RM0456
Reference manual

STM32U575/585 Arm®-based 32-bit MCUs

Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32U575/585 microcontrollers memory and peripherals.

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

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

STM32U575/585 microcontrollers include ST state-of-the-art patented technology.

Related documents

- STM32U585xx datasheet (DS13086)
- STM32U575xx datasheet (DS13737)
- STM32U585xx and STM32U575xx device errata sheet (ES0499)
- STM32 Cortex®-M33 MCUs programming manual (PM0264)

Contents

List of tables

List of figures

1 Documentation conventions

1.1 General information

The STM32U575/585 devices have an Arm®(a) Cortex®-M33 core.

1.2 List of abbreviations for registers

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

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

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

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

1.3 Glossary

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

- **Word**: data of 32-bit length.
- **Half-word**: data of 16-bit length.
- **Byte**: data of 8-bit length.
- **AHB**: advanced high-performance bus.
- **APB**: advanced peripheral bus.

1.4 Availability of peripherals

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

2 Memory and bus architecture

2.1 System architecture

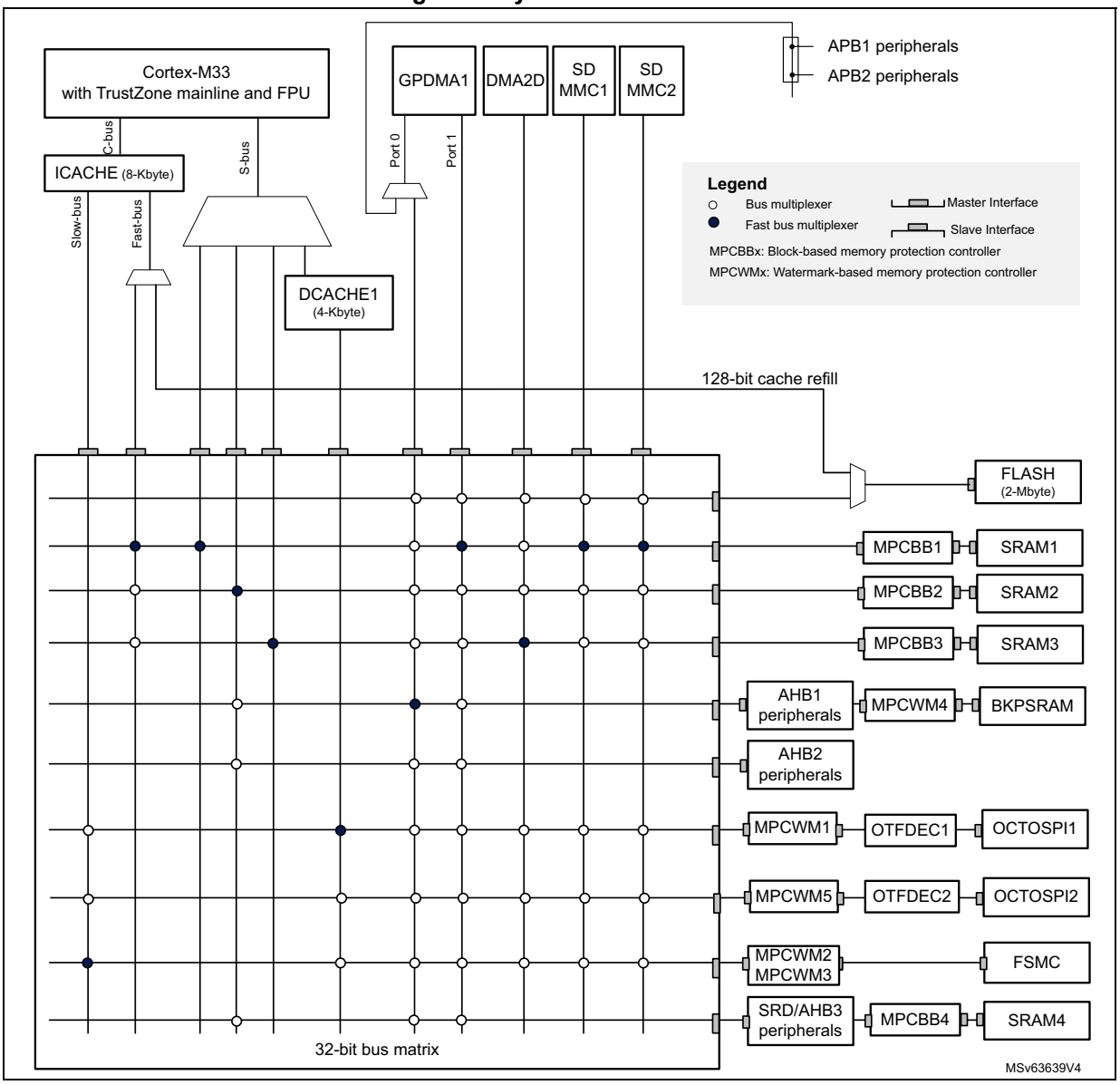
The STM32U575/585 architecture relies on a Arm Cortex-M33 core optimized for execution thanks to an instruction cache having a direct access to the embedded Flash memory.

This architecture also features a 32-bit multilayer AHB bus matrix that interconnects:

- up to 11 masters:
  - Fast C-bus, connecting Cortex-M33 with Arm TrustZone® mainline and FPU core C-bus to the internal SRAMs through the instruction cache
  - Slow C-bus, connecting Cortex-M33 with Arm TrustZone mainline and FPU core C-bus to the external memories through the instruction cache
  - Cortex-M33 with Arm TrustZone mainline and FPU core S-bus (three masters connected to three internal SRAMs without latency)
  - Cortex-M33 with Arm TrustZone mainline and FPU core S-bus connected to the external memories through the data cache
  - GPDMA1 (general purpose DMA featuring two master ports)
  - DMA2D
  - SDMMC1
  - SDMMC2
- up to 10 slaves:
  - internal Flash memory (2 Mbytes)
  - internal SRAM1 (192 Kbytes)
  - internal SRAM2 (64 Kbytes)
  - internal SRAM3 (512 Kbytes)
  - AHB1 peripherals and backup RAM (2-Kbyte BKPSRAM) including AHB to APB bridges and APB peripherals (connected to APB1 and APB2)
  - AHB2 peripherals
  - FSMC (flexible static memory controller)
  - OCTOSPI1
  - OCTOSPI2
  - SmartRun domain (SRD) AHB3 peripherals and SRAM4 (16 Kbytes), including AHB to APB bridge and APB peripherals (connected to APB3)

The bus matrix provides access from a master to a slave, enabling concurrent access and efficient operation even when several high-speed peripherals work simultaneously. This architecture is shown in the figure below.

Figure 1. System architecture



2.1.1 Fast C-bus

This bus connects the C-bus of the Cortex-M33 core to the internal Flash memory and to the bus matrix via the instruction cache. This bus is used for instruction fetch and data access to the internal memories mapped in code region. This bus targets the internal Flash memory and the internal SRAMs (SRAM1, SRAM2 and SRAM3).

SRAM1, SRAM2 and SRAM3 are accessible on this bus with a continuous mapping.

2.1.2 Slow C-bus

This bus connects the C-bus of the Cortex-M33 core to the bus matrix via the instruction cache. This bus is used for instruction fetch and data access to the external memories mapped in code region. This bus targets the external memories (FSMC and OCTOSPIs).

2.1.3 S-bus

This bus connects the system bus of the Cortex-M33 core to the bus matrix. This bus is used by the core to access data located in a peripheral or SRAM area. This bus targets the internal SRAMs (SRAM1, SRAM2, SRAM3, SRAM4 and BKPSRAM), the AHB1 peripherals including the APB1 and APB2 peripherals, the AHB2 peripherals and the SRD peripherals.

SRAM1, SRAM2 and SRAM3 are accessible on this bus with a continuous mapping.

Note: The bus matrix has a zero latency when accessing SRAM1, SRAM2 and SRAM3.

2.1.4 DCache S-bus

This bus connects the system bus of the Cortex-M33 core to the bus matrix via the data cache. This bus is used for instruction fetch and data access to the external memories mapped in data region. This bus targets the external memories (FSMC and OCTOSPIs).

Note: Fetching instructions through this bus is less efficient than fetching instructions through the slow C-bus.

2.1.5 GPDMA-bus

These buses connect the two AHB master interfaces of the GPDMA to the bus matrix. These buses target the internal Flash memory, the internal SRAMs (SRAM1, SRAM2, SRAM3, SRAM4 and BKPSRAM), the AHB1 peripherals including the APB1 and APB2 peripherals, the AHB2 peripherals, the SRD peripherals and the external memories through FSMC or OCTOSPIs.

2.1.6 SDMMC1 and SDMMC2 controllers DMA buses

These buses connect the SDMMC1 and SDMMC2 DMA master interfaces to the bus matrix. These buses are used only by the SDMMC1 and SDMMC2 DMA to load/store data from/to the memory. These buses target the data memories: internal Flash memory, internal SRAMs (SRAM1, SRAM2 and SRAM3) and external memories through FSMC or OCTOSPIs.

2.1.7 Bus matrix

The bus matrix manages the access arbitration between masters. The arbitration uses a Round-Robin algorithm. This bus matrix features a fast bus multiplexer used to connect each master to a given slave without latency (see *Figure 1*). For the same master, other slaves undergo a latency of at least one cycle at each new access.

2.1.8 AHB/APB bridges

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

Refer to *Section 2.3.2: Memory map and register boundary addresses* for the address mapping of the peripherals connected to these bridges.

After each device reset, all peripheral clocks are disabled (except for the internal SRAMs and Flash memory interfaces). Before using a peripheral, its clock must be enabled in the RCC_AHBxENR and RCC_APBxENR registers.

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

2.1.9 SmartRun domain (SRD)

The SRD architecture relies on a DMA allowing autonomous operation during low-power modes down to Stop 2.

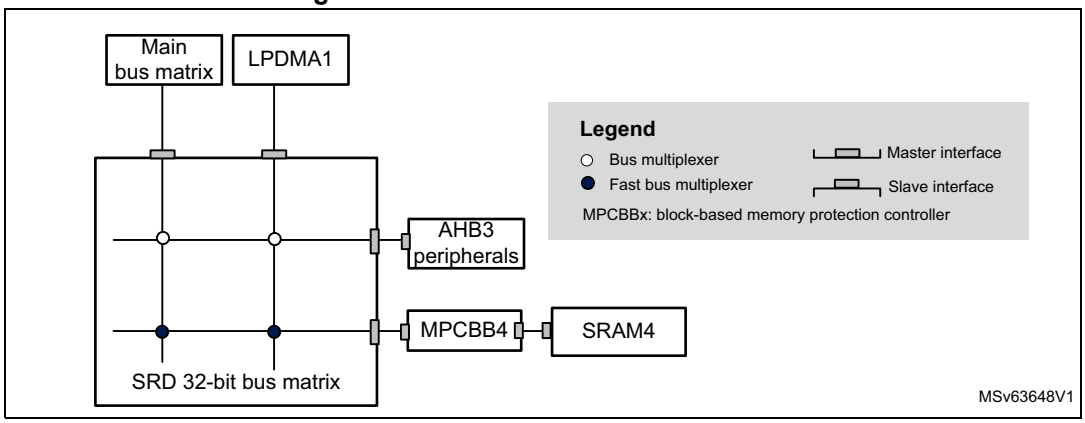
This architecture also features a 32-bit AHB bus matrix that interconnects:

- two masters:
  - the main AHB bus matrix
  - LPDMA1 (low-power DMA featuring one master port)
- two slaves:
  - AHB3 peripherals including AHB to APB bridge connected to APB3
  - internal SRAM4

Note: The SRAM4 is the only SRAM that can be accessed by the LPDMA1.

This architecture is shown in the figure below.

Figure 2. SmartRun domain architecture



2.2 Arm TrustZone security architecture

The security architecture is based on Arm TrustZone with the Armv8-M mainline 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) defines the access permissions based on secure and non-secure states.

- SAU: Up to eight SAU configurable regions are available for security attribution.
- IDAU: provides a first memory partition as non-secure or non-secure callable attributes. The IDAU memory map partition is not configurable and fixed by hardware implementation (refer to *Figure 3: Memory map based on IDAU mapping*). It is then

combined with the results from the SAU security attribution and the higher security state is selected.

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

The table below 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 configuration regions

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, non-secure or NSC(1)	Secure, non-secure or NSC
Code - Flash memory 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	
	0x1800_0000 0x1FFF_FFFF			
SRAM	0x2000_0000 0x2FFF_FFFFF	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, non-secure or NSC	Secure, non-secure or NSC

1. NSC = non-secure callable

2.2.1 Default Arm TrustZone security state

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

- CPU:
  - Cortex-M33 is in secure state after reset. The boot address must be at a secure address.
- Memory map:
  - SAU is fully secure after reset. Consequently, all memory map is fully secure. Up to height SAU configurable regions are available for security attribution.
- Flash memory:
  - The Flash memory security area is defined by watermark user options.
  - Flash block-based security attributions are non-secure after reset.
- SRAMs:

  - All SRAMs are secure after reset. MPCBBx (block-based memory protection controller) are secure.
- External memories:
  - FSMC and OCTOSPIs banks are secure after reset. MPCWMx (watermark-based memory protection controller) are secure.
- Peripherals (see *Table 2* and *Table 3* for a list of securable and TrustZone-aware peripherals)
  - Securable peripherals are non-secure after reset.
  - TrustZone-aware peripherals are non-secure after reset. Their secure configuration registers are secure.
- 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 (see *GTZC TrustZone system architecture*).

2.2.2 Arm TrustZone peripheral classification

When the TrustZone security is active, a peripheral can be either securable or TrustZone-aware type as follows:

- Securable: peripheral protected by an AHB/APB firewall gate that is controlled from TZSC controller to define security properties
- TrustZone-aware: peripheral connected directly to AHB or APB bus and implementing a specific TrustZone behavior such as a subset of registers being secure.

Refer to *GTZC TrustZone system architecture* for more details.

The tables below list the securable and TrustZone-aware peripherals within the system.

Table 2. Securable peripherals by TZSC

Bus	Peripheral
AHB3	ADF1
	DAC1
	ADC4

Table 2. Securable peripherals by TZSC (continued)

Bus	Peripheral
AHB2	OCTOSPI2 registers
	OCTOSPI1 registers
	FSMC registers
	SDMMC1
	SDMMC2
	OCTOSPIM
	SAES
	PKA
	RNG
	HASH
	AES
	OTG_FS
	DCMI
	ADC1
AHB1	DCACHE1 registers
	ICACHE registers
	DMA2D
	TSC
	CRC
	RAMCFG
	MDF1
	FMAC
	CORDIC
APB3	VREFBUF
	COMP
	OPAMP
	LPTIM4
	LPTIM3
	LPTIM1
	I2C3
	LPUART1
	SPI3

Table 2. Securable peripherals by TZSC (continued)

Bus	Peripheral
APB2	SAI2
	SAI1
	TIM17
	TIM16
	TIM15
	USART1
	TIM8
	SPI1
	TIM1
APB1	UCPD1
	FDCAN1
	LPTIM2
	I2C4
	CRS
	I2C2
	I2C1
	UART5
	UART4
	USART3
	USART2
	SPI2
	IWDG
	WWDG
	TIM7
	TIM6
	TIM5
	TIM4
	TIM3
	TIM2

Table 3. TrustZone-aware peripherals

Bus	Peripheral
AHB3	GTZC2
	EXTI
	LPDMA1
	RCC
	PWR
	LPGPIO1
AHB2	OTFDEC1(1)
	OTFDEC2(1)
	GPIOI
	GPIOH
	GPIOG
	GPIOF
	GPIOE
	GPIOD
	GPIOC
	GPIOB
	GPIOA
AHB1	GTZC1
	FLASH
	GPDMA1
APB3	TAMP
	RTC
	SYSCFG

1. Always secure when TZEN = 1.

2.3 Memory organization

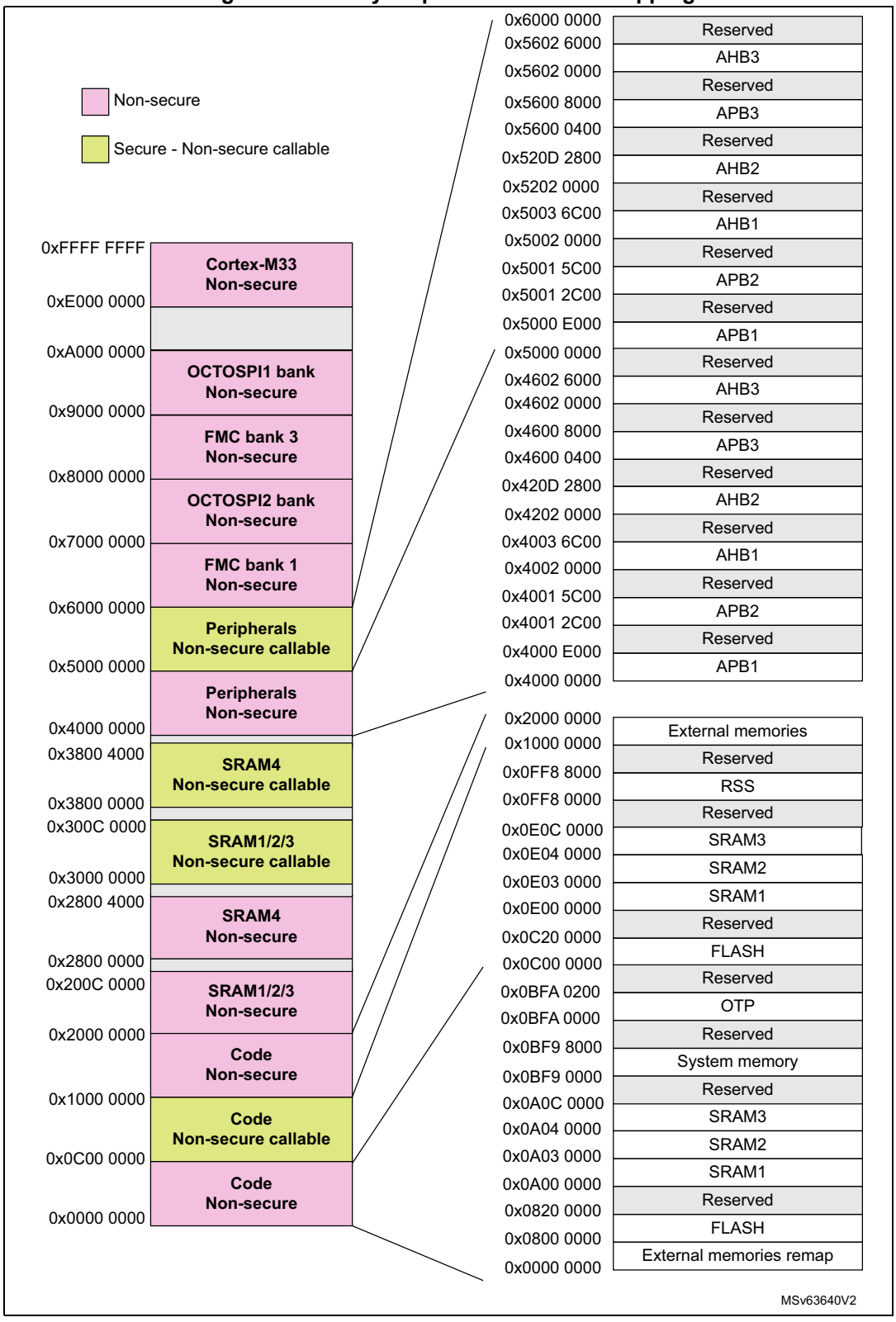
2.3.1 Introduction

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 3. Memory map based on IDAU mapping



All the memory map areas that are not allocated to on-chip memories and peripherals are considered “Reserved”. The following table gives the boundary addresses of the peripherals available in the devices.

Table 4. Memory map and peripheral register boundary addresses

Bus	Secure boundary address	Non-secure boundary address	Size (bytes)	Peripheral	Peripheral register map
AHB3	0x5602 6000 - 0x5FFF FFFF	0x4602 6000 - 0x4FFF FFFF	164 M	Reserved	-
	0x5602 5000- 0x5602 5FFF	0x4602 5000- 0x4602 5FFF	4 K	LPDMA1	*LPDMA register map*
	0x5602 4000- 0x5602 4FFF	0x4602 4000- 0x4602 4FFF	4 K	ADF1	*ADF register map*
	0x5602 3C00 - 0x5602 3FFF	0x4602 3C00 - 0x4602 3FFF	1 K	Reserved	-
	0x5602 3800 - 0x5602 3BFF	0x4602 3800 - 0x4602 3BFF	1 K	GTZC2_ MPCBB4	*GTZC2 MPCBB4 register map*
	0x5602 3400 - 0x5602 37FF	0x4602 3400 - 0x4602 37FF	1 K	GTZC2_ TZIC	*GTZC2 TZIC register map*
	0x5602 3000 - 0x5602 33FF	0x4602 3000 - 0x4602 33FF	1 K	GTZC2_ TZSC	*GTZC2 TZSC register map*
	0x5602 2400 - 0x5602 2FFF	0x4602 2400 - 0x4602 2FFF	3 K	Reserved	-
	0x5602 2000 - 0x5602 23FF	0x4602 2000 - 0x4602 23FF	1 K	EXTI	*EXTI register map*
	0x5602 1C00 - 0x5602 1FFF	0x4602 1C00 - 0x4602 1FFF	1 K	Reserved	-
	0x5602 1800 - 0x5602 1BFF	0x4602 1800 - 0x4602 1BFF	1 K	DAC1	*DAC register map*
	0x5602 1400 - 0x5602 17FF	0x4602 1400 - 0x4602 17FF	1 K	Reserved	-
	0x5602 1000 - 0x5602 13FF	0x4602 1000 - 0x4602 13FF	1 K	ADC4	*ADC register map*
	0x5602 0C00 - 0x5602 0FFF	0x4602 0C00 - 0x4602 0FFF	1 K	RCC	*RCC register map*
	0x5602 0800 - 0x5602 0BFF	0x4602 0800 - 0x4602 0BFF	1 K	PWR	*PWR register map*
	0x5602 0400 - 0x5602 07FF	0x4602 0400 - 0x4602 07FF	1 K	Reserved	-
	0x5602 0000 - 0x5602 03FF	0x4602 0000 - 0x4602 03FF	1 K	LPGPIO1	*LPGPIO register map*

Table 4. Memory map and peripheral register boundary addresses (continued)

Bus	Secure boundary address	Non-secure boundary address	Size (bytes)	Peripheral	Peripheral register map
APB3	0x5600 8000 - 0x5601 FFFF	0x4600 8000 - 0x4601 FFFF	96 K	Reserved	-
	0x5600 7C00 - 0x5600 7FFF	0x4600 7C00 - 0x4600 7FFF	1 K	TAMP	*TAMP register map*
	0x5600 7800 - 0x5600 7BFF	0x4600 7800 - 0x4600 7BFF	1 K	RTC	*RTC register map*
	0x5600 7400 - 0x5600 77FF	0x4600 7400 - 0x4600 77FF	1 K	VREFBUF	*VREFBUF register map*
	0x5600 5800 - 0x5600 73FF	0x4600 5800 - 0x4600 73FF	7 K	Reserved	-
	0x5600 5400 - 0x5600 57FF	0x4600 5400 - 0x4600 57FF	1 K	COMP	*COMP register map*
	0x5600 5000 - 0x5600 53FF	0x4600 5000 - 0x4600 53FF	1 K	OPAMP	*OPAMP register map*
	0x5600 4C00 - 0x5600 4FFF	0x4600 4C00 - 0x4600 4FFF	1 K	LPTIM4	*LPTIM register map*
	0x5600 4800 - 0x5600 4BFF	0x4600 4800 - 0x4600 4BFF	1 K	LPTIM3	
	0x5600 4400 - 0x5600 47FF	0x4600 4400 - 0x4600 47FF	1 K	LPTIM1	
	0x5600 2C00 - 0x5600 43FF	0x4600 2C00 - 0x4600 43FF	6 K	Reserved	-
	0x5600 2800 - 0x5600 2BFF	0x4600 2800 - 0x4600 2BFF	1 K	I2C3	*I2C register map*
	0x5600 2400 - 0x5600 27FF	0x4600 2400 - 0x4600 27FF	1 K	LPUART1	*LPUART register map*
	0x5600 2000 - 0x5600 23FF	0x4600 2000 - 0x4600 23FF	1 K	SPI3	*SPI register map*
	0x5600 0800 - 0x5600 1FFF	0x4600 0800 - 0x4600 1FFF	6 K	Reserved	-
	0x5600 0400 - 0x5600 07FF	0x4600 0400 - 0x4600 07FF	1 K	SYSCFG	*SYSCFG register map*
AHB2	0x520D 2800 - 0x5600 03FF	0x420D 2800 - 0x4600 03FF	64.7 M	Reserved	-
	0x520D 2400 - 0x520D 27FF	0x420D 2400 - 0x420D 27FF	1 K	OCTOSPI2 registers	*OCTOSPI register map*
	0x520D 1800 - 0x520D 23FF	0x420D 1800 - 0x420D 23FF	3 K	Reserved	-
	0x520D 1400 - 0x520D 17FF	0x420D 1400 - 0x420D 17FF	1 K	OCTOSPI1 registers	*OCTOSPI register map*
	0x520D 0800 - 0x520D 13FF	0x420D 0800 - 0x420D 13FF	3 K	Reserved	-
	0x520D 0400 - 0x520D 07FF	0x420D 0400 - 0x420D 07FF	1 K	FSMC registers	*FMC register map*
	0x520C F800 - 0x520D 03FF	0x420C F800 - 0x420D 03FF	3K	Reserved	-
	0x520C F400 - 0x520C F7FF	0x420C F400 - 0x420C F7FF	1 K	DLYBOS2	*DLYB register map*
	0x520C F000 - 0x520C F3FF	0x420C F000 - 0x420C F3FF	1 K	DLYBOS1	
	0x520C 9000 - 0x520C EFFF	0x420C 9000 - 0x420C EFFF	24 K	Reserved	-
	0x520C 8C00 - 0x520C 8FFF	0x420C 8C00 - 0x420C 8FFF	1 K	SDMMC2	*SDMMC register map*
	0x520C 8800 - 0x520C 8BFF	0x420C 8800 - 0x420C 8BFF	1 K	DLYBSD2	*DLYB register map*
	0x520C 8400 - 0x520C 87FF	0x420C 8400 - 0x420C 87FF	1 K	DLYBSD1	
	0x520C 8000 - 0x520C 83FF	0x420C 8000 - 0x420C 83FF	1 K	SDMMC1	*SDMMC register map*
	0x520C 5800 - 0x520C 7FFF	0x420C 5800 - 0x420C 7FFF	10 K	Reserved	-

Table 4. Memory map and peripheral register boundary addresses (continued)

Bus	Secure boundary address	Non-secure boundary address	Size (bytes)	Peripheral	Peripheral register map
AHB2 (cont'd)	0x520C 5400 - 0x520C 57FF	0x420C 5400 - 0x420C 57FF	1 K	OTFDEC2	*OTFDEC register map*
	0x520C 5000 - 0x520C 53FF	0x420C 5000 - 0x420C 53FF	1 K	OTFDEC1	
	0x520C 4400 - 0x520C 4FFF	0x420C 4400 - 0x420C 4FFF	3 K	Reserved	-
	0x520C 4000 - 0x520C 43FF	0x420C 4000 - 0x420C 43FF	1 K	OCTOSPIM	*OCTOSPIM register map*
	0x520C 2000 - 0x520C 3FFF	0x420C 2000 - 0x420C 3FFF	8 K	PKA	*PKA register map*
	0x520C 1000 - 0x520C 1FFF	0x420C 1000 - 0x420C 1FFF	4 K	Reserved	-
	0x520C 0C00 - 0x520C 0FFF	0x420C 0C00 - 0x420C 0FFF	1 K	SAES	*SAES register map*
	0x520C 0800 - 0x520C 0BFF	0x420C 0800 - 0x420C 0BFF	1 K	RNG	*RNG register map*
	0x520C 0400 - 0x520C 07FF	0x420C 0400 - 0x420C 07FF	1 K	HASH	*HASH register map*
	0x520C 0000 - 0x520C 03FF	0x420C 0000 - 0x420C 03FF	1 K	AES	*AES register map*
	0x5204 0000 - 0x520B FFFF	0x4204 0000 - 0x420B FFFF	512 K	OTG_FS	*OTG_FS register map*
	0x5202 C800 - 0x5203 7FFF	0x4202 C800 - 0x4203 FFFF	78 K	Reserved	-
	0x5202 C400 - 0x5202 C7FF	0x4202 C400 - 0x4202 C7FF	1 K	PSSI	*PSSI register map*
	0x5202 C000 - 0x5202 C3FF	0x4202 C000 - 0x4202 C3FF	1 K	DCMI	*DCMI register map*
	0x5202 8400 - 0x5202 BFFF	0x4202 8400 - 0x4202 BFFF	15 K	Reserved	-
	0x5202 8000 - 0x5202 83FF	0x4202 8000 - 0x4202 83FF	1 K	ADC1	*ADC register map*
	0x5202 2400 - 0x5202 7FFF	0x4202 2400 - 0x4202 7FFF	23 K	Reserved	-
	0x5202 2000 - 0x5202 23FF	0x4202 2000 - 0x4202 23FF	1 K	GPIOI	*GPIO register map*
	0x5202 1C00 - 0x5202 1FFF	0x4202 1C00 - 0x4202 1FFF	1 K	GPIOH	
	0x5202 1800 - 0x5202 1BFF	0x4202 1800 - 0x4202 1BFF	1 K	GPIOG	
	0x5202 1400 - 0x5202 17FF	0x4202 1400 - 0x4202 17FF	1 K	GPIOF	
	0x5202 1000 - 0x5202 13FF	0x4202 1000 - 0x4202 13FF	1 K	GPIOE	
	0x5202 0C00 - 0x5202 0FFF	0x4202 0C00 - 0x4202 0FFF	1 K	GPIOD	
	0x5202 0800 - 0x5202 0BFF	0x4202 0800 - 0x4202 0BFF	1 K	GPIOC	
	0x5202 0400 - 0x5202 07FF	0x4202 0400 - 0x4202 07FF	1 K	GPIOB	
	0x5202 0000 - 0x5202 03FF	0x4202 0000 - 0x4202 03FF	1 K	GPIOA	

Table 4. Memory map and peripheral register boundary addresses (continued)

Bus	Secure boundary address	Non-secure boundary address	Size (bytes)	Peripheral	Peripheral register map
AHB1	0x5003 6C00 - 0x5201 FFFF	0x4003 6C00 - 0x4201 FFFF	32.7 M	Reserved	-
	0x5003 6400 - 0x5003 6BFF	0x4003 6400 - 0x4003 6BFF	2 K	BKPSRAM	-
	0x5003 3800 - 0x5003 63FF	0x4003 3800 - 0x4003 63FF	11 K	Reserved	-
	0x5003 3400 - 0x5003 37FF	0x4003 3400 - 0x4003 37FF	1 K	GTZC1_ MPCBB3	*GTZC1 MPCBBz register map (z = 1 to 3)*
	0x5003 3000 - 0x5003 33FF	0x4003 3000 - 0x4003 33FF	1 K	GTZC1_ MPCBB2	
	0x5003 2C00 - 0x5003 2FFF	0x4003 2C00 - 0x4003 2FFF	1 K	GTZC1_ MPCBB1	
	0x5003 2800 - 0x5003 2BFF	0x4003 2800 - 0x4003 2BFF	1 K	GTZC1_ TZIC	*GTZC1 TZIC register map*
	0x5003 2400 - 0x5003 27FF	0x4003 2400 - 0x4003 27FF	1 K	GTZC1_ TZSC	*GTZC1 TZSC register map*
	0x5003 1800 - 0x5003 23FF	0x4003 1800 - 0x4003 23FF	3 K	Reserved	-
	0x5003 1400 - 0x5003 17FF	0x4003 1400 - 0x4003 17FF	1 K	DCACHE1	*DCACHE register map*
	0x5003 0800 - 0x5003 13FF	0x4003 0800 - 0x4003 13FF	3 K	Reserved	-
	0x5003 0400 - 0x5003 07FF	0x4003 0400 - 0x4003 07FF	1 K	ICACHE	*ICACHE register map*
	0x5002 BC00 - 0x5003 03FF	0x4002 BC00 - 0x4003 03FF	18 K	Reserved	-
	0x5002 B000 - 0x5002 BBFF	0x4002 B000 - 0x4002 BBFF	3 K	DMA2D	*DMA2D register map*
	0x5002 7000 - 0x5002 AFFF	0x4002 7000 - 0x4002 AFFF	16 K	Reserved	-
	0x5002 6000 - 0x5002 6FFF	0x4002 6000 - 0x4002 6FFF	4 K	RAMCFG	*RAMCFG register map*
	0x5002 5000 - 0x5002 5FFF	0x4002 5000 - 0x4002 5FFF	4 K	MDF1	*MDF register map*
	0x5002 4400 - 0x5002 4FFF	0x4002 4400 - 0x4002 4FFF	3 K	Reserved	-
	0x5002 4000 - 0x5002 43FF	0x4002 4000 - 0x4002 43FF	1 K	TSC	*TSC register map*
	0x5002 3400 - 0x5002 3FFF	0x4002 3400 - 0x4002 3FFF	3 K	Reserved	-
	0x5002 3000 - 0x5002 33FF	0x4002 3000 - 0x4002 33FF	1 K	CRC	*CRC register map*
	0x5002 2400 - 0x5002 2FFF	0x4002 2400 - 0x4002 2FFF	3 K	Reserved	-
	0x5002 2000 - 0x5002 23FF	0x4002 2000 - 0x4002 23FF	1 K	FLASH registers	*FLASH register map*
	0x5002 1800 - 0x5002 1FFF	0x4002 1800 - 0x4002 1FFF	2 K	Reserved	-
	0X5002 1400 - 0x5002 17FF	0X4002 1400 - 0x4002 17FF	1 K	FMAC	*FMAC register map*
	0X5002 1000 - 0x5002 13FF	0X4002 1000 - 0x4002 13FF	1 K	CORDIC	*CORDIC register map*
	0x5002 0000 - 0x5002 0FFF	0x4002 0000 - 0x4002 0FFF	4 K	GPDMA1	*GPDMA register map*

Table 4. Memory map and peripheral register boundary addresses (continued)

Bus	Secure boundary address	Non-secure boundary address	Size (bytes)	Peripheral	Peripheral register map
APB2	0x5001 5C00 - 0x5001 FFFF	0x4001 5C00 - 0x4001 FFFF	41 K	Reserved	-
	0x5001 5800 - 0x5001 5BFF	0x4001 5800 - 0x4001 5BFF	1 K	SAI2	*SAI register map*
	0x5001 5400 - 0x5001 57FF	0x4001 5400 - 0x4001 57FF	1 K	SAI1	
	0x5001 4C00 - 0x5001 53FF	0x4001 4C00 - 0x4001 53FF	2 K	Reserved	-
	0x5001 4800 - 0x5001 4BFF	0x4001 4800 - 0x4001 4BFF	1 K	TIM17	*TIM16/TIM17 register map*
	0x5001 4400 - 0x5001 47FF	0x4001 4400 - 0x4001 47FF	1 K	TIM16	
	0x5001 4000 - 0x5001 43FF	0x4001 4000 - 0x4001 43FF	1 K	TIM15	*TIM15 register map*
	0x5001 3C00 - 0x5001 3FFF	0x4001 3C00 - 0x4001 3FFF	1 K	Reserved	-
	0x5001 3800 - 0x5001 3BFF	0x4001 3800 - 0x4001 3BFF	1 K	USART1	*USART register map*
	0x5001 3400 - 0x5001 37FF	0x4001 3400 - 0x4001 37FF	1 K	TIM8	*TIMx register map*
	0x5001 3000 - 0x5001 33FF	0x4001 3000 - 0x4001 33FF	1 K	SPI1	*SPI register map*
	0x5001 2C00 - 0x5001 2FFF	0x4001 2C00 - 0x4001 2FFF	1 K	TIM1	*TIMx register map*
APB1	0x5000 E000 - 0x5001 2BFF	0x4000 E000 - 0x4001 2BFF	19 K	Reserved	-
	0x5000 DC00 - 0x5000 DFFF	0x4000 DC00 - 0x4000 DFFF	1 K	UCPD1	*UCPD register map*
	0x5000 B000 - 0x5000 DBFF	0x4000 B000 - 0x4000 DBFF	11 K	Reserved	-
	0x5000 AC00 - 0x5000 AFFF	0x4000 AC00 - 0x4000 AFFF	1 K	FDCAN1 RAM	-
	0x5000 A800 - 0x5000 ABFF	0x4000 A800 - 0x4000 ABFF	1 K	Reserved	-
	0x5000 A400 - 0x5000 A7FF	0x4000 A400 - 0x4000 A7FF	1 K	FDCAN1	*FDCAN register map*
	0x5000 9800 - 0x5000 A3FF	0x4000 9800 - 0x4000 A3FF	3 K	Reserved	-
	0x5000 9400 - 0x5000 97FF	0x4000 9400 - 0x4000 97FF	1 K	LPTIM2	*LPTIM register map*
	0x5000 8800 - 0x5000 93FF	0x4000 8800 - 0x4000 93FF	3 K	Reserved	-
	0x5000 8400 - 0x5000 87FF	0x4000 8400 - 0x4000 87FF	1 K	I2C4	*I2C register map*
	0x5000 6400 - 0x5000 83FF	0x4000 6400 - 0x4000 83FF	8 K	Reserved	-
	0x5000 6000 - 0x5000 63FF	0x4000 6000 - 0x4000 63FF	1 K	CRS	*CRS register map*
	0x5000 5C00 - 0x5000 5FFF	0x4000 5C00 - 0x4000 5FFF	1 K	Reserved	-
	0x5000 5800 - 0x5000 5BFF	0x4000 5800 - 0x4000 5BFF	1 K	I2C2	*I2C register map*
	0x5000 5400 - 0x5000 57FF	0x4000 5400 - 0x4000 57FF	1 K	I2C1	
	0x5000 5000 - 0x5000 53FF	0x4000 5000 - 0x4000 53FF	1 K	UART5	*USART register map*
	0x5000 4C00 - 0x5000 4FFF	0x4000 4C00 - 0x4000 4FFF	1 K	UART4	
	0x5000 4800 - 0x5000 4BFF	0x4000 4800 - 0x4000 4BFF	1 K	USART3	
	0x5000 4400 - 0x5000 47FF	0x4000 4400 - 0x4000 47FF	1 K	USART2	
	0x5000 3C00 - 0x5000 43FF	0x4000 3C00 - 0x4000 43FF	2 K	Reserved	-

Table 4. Memory map and peripheral register boundary addresses (continued)

Bus	Secure boundary address	Non-secure boundary address	Size (bytes)	Peripheral	Peripheral register map
APB1 (cont'd)	0x5000 3800 - 0x5000 3BFF	0x4000 3800 - 0x4000 3BFF	1 K	SPI2	*SPI register map*
	0x5000 3400 - 0x5000 37FF	0x4000 3400 - 0x4000 37FF	1 K	Reserved	-
	0x5000 3000 - 0x5000 33FF	0x4000 3000 - 0x4000 33FF	1 K	IWDG	*IWDG register map*
	0x5000 2C00 - 0x5000 2FFF	0x4000 2C00 - 0x4000 2FFF	1 K	WWDG	*WWDG register map*
	0x5000 1800 - 0x5000 2BFF	0x4000 1800 - 0x4000 2BFF	5 K	Reserved	-
	0x5000 1400 - 0x5000 17FF	0x4000 1400 - 0x4000 17FF	1 K	TIM7	*TIMx register map*
	0x5000 1000 - 0x5000 13FF	0x4000 1000 - 0x4000 13FF	1 K	TIM6	
	0x5000 0C00 - 0x5000 0FFF	0x4000 0C00 - 0x4000 0FFF	1 K	TIM5	*TIMx register map*
	0x5000 0800 - 0x5000 0BFF	0x4000 0800 - 0x4000 0BFF	1 K	TIM4	
	0x5000 0400 - 0x5000 07FF	0x4000 0400 - 0x4000 07FF	1 K	TIM3	
	0x5000 0000 - 0x5000 03FF	0x4000 0000 - 0x4000 03FF	1 K	TIM2	

2.3.3 Embedded SRAM

The devices feature up to 786-Kbyte SRAMs:

- 192-Kbyte SRAM1
- 64-Kbyte SRAM2
- 512-Kbyte SRAM3
- 16-Kbyte SRAM4
- 2-Kbyte BKPSRAM

These SRAMs can be accessed as bytes, half-words (16 bits) or full words (32 bits). These memories can be addressed both by CPU and DMAs.

The CPU can access the SRAM1, SRAM2 and SRAM3 through the system bus or through the C-bus depending on the selected address. The CPU can access the SRAM4 and BKPSRAM through the system bus only.

When the TrustZone security is enabled, all SRAMs are secure after reset. The SRAM can be programmed as non-secure with a block granularity. For more details, refer to *Section 5: Global TrustZone controller (GTZC)*.

SRAM features are detailed in *Section 6.3.1: Internal SRAMs features*.

2.3.4 Flash memory overview

The Flash memory is composed of two distinct physical areas:

- The main Flash memory block, that contains the application program and user data.
- The information block, that is composed of the following parts:
  - option bytes for hardware and memory protection user configuration
  - system memory that contains 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 7: Embedded Flash memory (FLASH)* for more details.

3 System security

The STM32U575/585 are designed with a comprehensive set of security features, some of which being based on the standard Arm TrustZone technology.

These security features simplify the process of evaluating IoT devices against security standards. They also significantly reduce the cost and complexity of software development for OEM 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 STM32U575/585 devices.

3.1 Key security features

- Resource isolation using privilege mode and 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, I^2C, SPI, FDCAN and JTAG
- Secure boot thanks to the unique boot entry feature and hide-protect area (HDP) mechanism
- Secure storage, featuring:
  - Non-volatile on-chip secure storage, protected with secure and HDP areas
  - Battery-powered volatile secure storage, automatically erased in case of tamper
  - Write-only key registers in the AES engines
  - Device 96-bit unique ID and JTAG 32-bit device-specific ID
  - On-chip enhance storage technology, using hardware secret non-volatile derived hardware unique keys (DHUK), and application-defined volatile boot hardware key (BHK), both loadable by hardware to the DPA-resistant SAES engine
- General purpose cryptographic acceleration
  - AES 256-bit engine, supporting ECB, CBC, CTR, GCM and CCM chaining modes
  - Secure AES 256-bit security co-processor, supporting ECB and CBC chaining modes with side-channel counter-measures and mitigations
  - HASH processor, supporting MD5/SHA-1 checksums and SHA-2 secure hash
  - Public key accelerator (PKA) for RSA/DH (up to 4096 bits) and ECC (up to 640 bits), implementing side-channel counter measures and mitigations when manipulating secrets
  - True random number generator (RNG), NIST SP800-90B pre-certified
- On-the-fly decryption of encrypted image stored on external Flash memory connected through the OCTOSPI
  - 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 life-cycle scheme with readout protection (RDP), including support for product decommissioning (auto-erase)
  - Debug protection, depending on the RDP level
  - Optional password-based RDP level regressions, including for RDP Level 2
- 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
  - Eight active inputs, eight active output tamper pins, available in all power modes

3.2 Secure install

The secure firmware install (SFI) is an STMicroelectronics secure service authenticated and decrypted by the immutable RSS code stored in the device. The SFI 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 the internal Flash memory or encrypted in an external Flash memory, is also protected, using the AES.

The SFI native service leverages the following hardware security features:

- secure boot (see *Section 3.3*)
- resource isolation using TrustZone (see *Section 3.5*)
- temporal isolation using hide protection (see *Section 3.6.1*)
- secure execution (see *Section 3.7*)
- secure storage, with associated cryptographic engines (see *Section 3.8* and *Section 3.9*)

Further information can be found in the application note *Overview secure firmware install (SFI)* (AN4992).

3.3 Secure boot

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

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

The device 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 the user manual *Getting started with STM32CubeU5 TFM application* (UM2851).

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

- resource isolation using TrustZone (see *Section 3.5*)
- temporal isolation using hide protection (see *Section 3.6.1*)

- secure execution (see *Section 3.7*)
- secure install and update (see *Section 3.2* and *Section 3.4*)
- secure storage, with associated cryptographic engines if available (see *Section 3.8* and *Section 3.9*)

This section describes the features specifically designed for secure boot.

3.3.1 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 the system Flash memory (see the next section), or in the 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 unmodifiable secure address defined by SECBOOTADD0 option bytes. All these option bytes cannot be modified by the application anymore when BOOT_LOCK is set.

Note: As long as it is cleared, the BOOT_LOCK option bit can be set without any constraint. But once set, the BOOT_LOCK option bit cannot be cleared when RDP level > 0.

For more information on the boot mechanisms, refer to *Section 4: Boot modes*.

3.3.2 Immutable root of trust in system Flash memory

The immutable root-of-trust code stored in the system Flash memory is first used to initiate SFI, allowing secure and counted installation of OEM firmware in untrusted production environment (such as OEM contract manufacturer).

The STMicroelectronics immutable code also includes secure runtime services that can be called at runtime when a secure application sets the SYSCGFG_RSSCMDR register to a non-null value before triggering a system reset. This runtime feature is deactivated when the BOOT_LOCK secure option bit is set, and the secure address defined by SECBOOTADD0 is set on the secure user flash memory.

3.4 Secure update

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

The device 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.

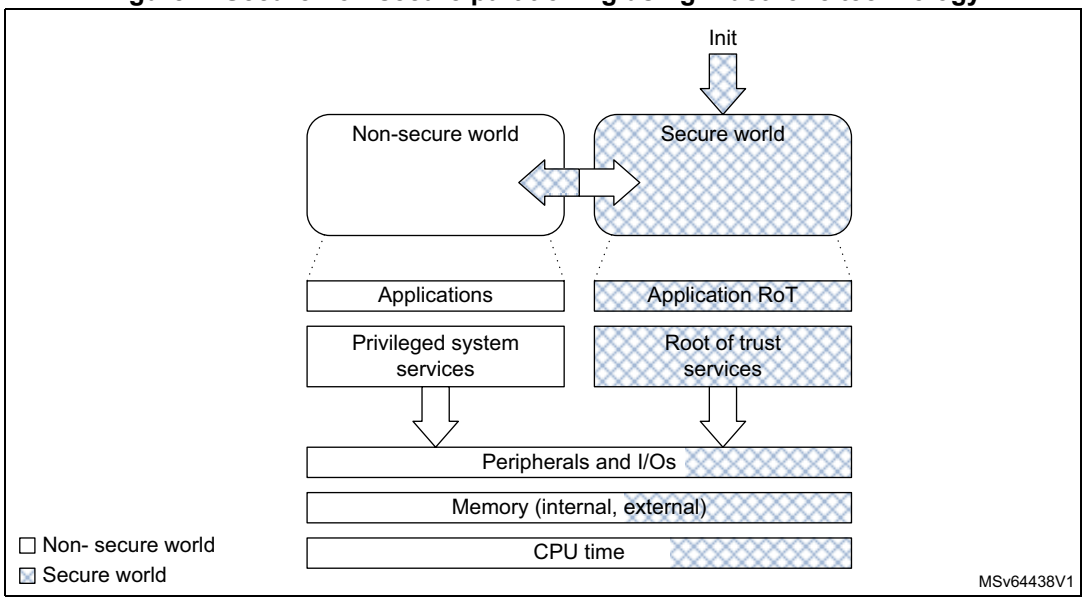
A firmware update can be done either on a single firmware image including both secure and non-secure parts, or on the secure (respectively non-secure) part of the firmware image, independently.

In the devices, the secure update application leverages the same hardware security as the firmware install described in *Section 3.2*. For more information, refer to the user manual *Getting started with STM32CubeU5 TFM application* (UM2851).

3.5 Resource isolation using TrustZone

In the STM32U575/585 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 in the figure below.

Figure 4. Secure/non-secure partitioning using TrustZone technology



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

3.5.1 TrustZone security architecture

The Armv8-M TrustZone technology is a comprehensive hardware architecture that proposes to developers a comprehensive, holistic protection across the entire processor and system. The 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 the TrustZone protection against transactions coming from other masters in the system than the Cortex-M33
- TrustZone-aware embedded Flash memory and peripherals

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

3.5.2 Armv8-M security extension of Cortex-M33

The Arm security extension of the Cortex-M33 is an evolution, not a revolution. It uses the programmer model from 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. Armv8-M also uses the nested vectored interrupt controller (NVIC).

The Armv8-M TrustZone implementation in STM32U575/585 devices 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-Gbyte address space
- a low interrupt latency for both secure and non-secure domains, and 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 3.11*

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

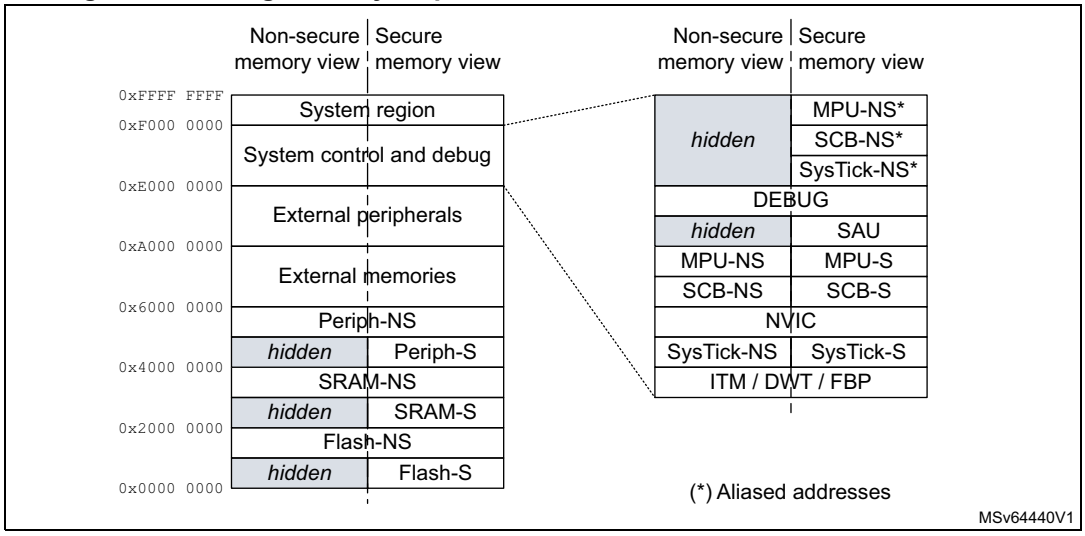
3.5.3 Memory and peripheral allocation using IDAU/SAU

Security attributes

As illustrated on *Figure 5*, the Armv8-M non-secure memory view is similar to Armv7-M (that can be found in Cortex-M4), with the difference that the secure memory is hidden. The secure memory view shows the Flash memory, SRAM and peripherals that are only accessible while the Cortex processor executes in Secure state.

The figure below shows the 32-bit address space viewed after the SAU configuration by the secure code.

Figure 5. Sharing memory map between CPU in secure and non-secure state

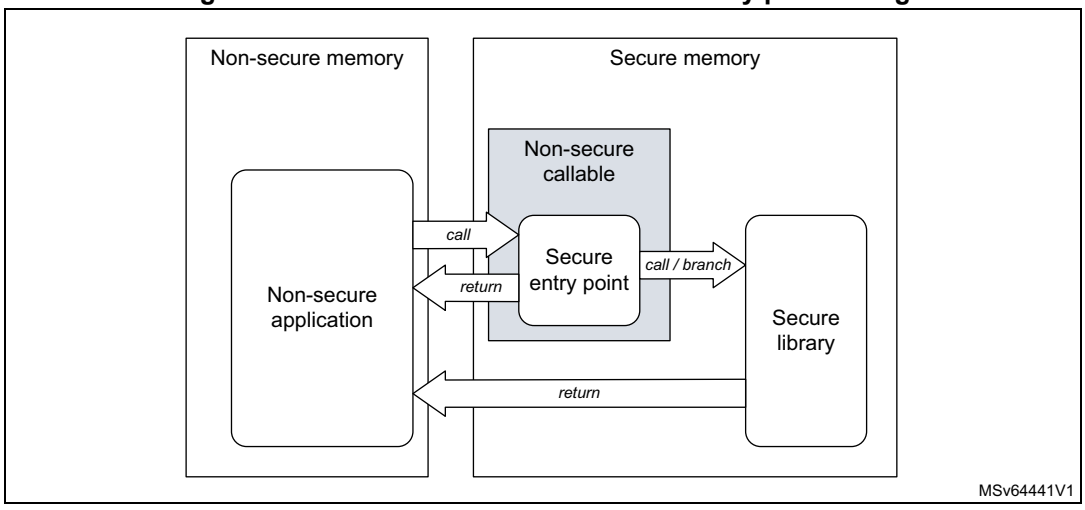


The Cortex processor state (and associated rights) depends on the security attribute assigned to the memory region where it is executed:

- 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 the 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 the non-secure code to call a secure function (see the figure below).

Figure 6. 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 in the table below.

Table 5. 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-privilege state. When the TrustZone is enabled, the SAU defaults all addresses as secure (S). A secure boot application can then program the SAU to create NSC or NS regions, as shown in the previous table.

Note: The SAU/IDAU settings are applicable only to the Cortex-M33*. The other masters like DMA are not affected by these policies.*

A memory space not covered by an SAU region is fixed as secure.

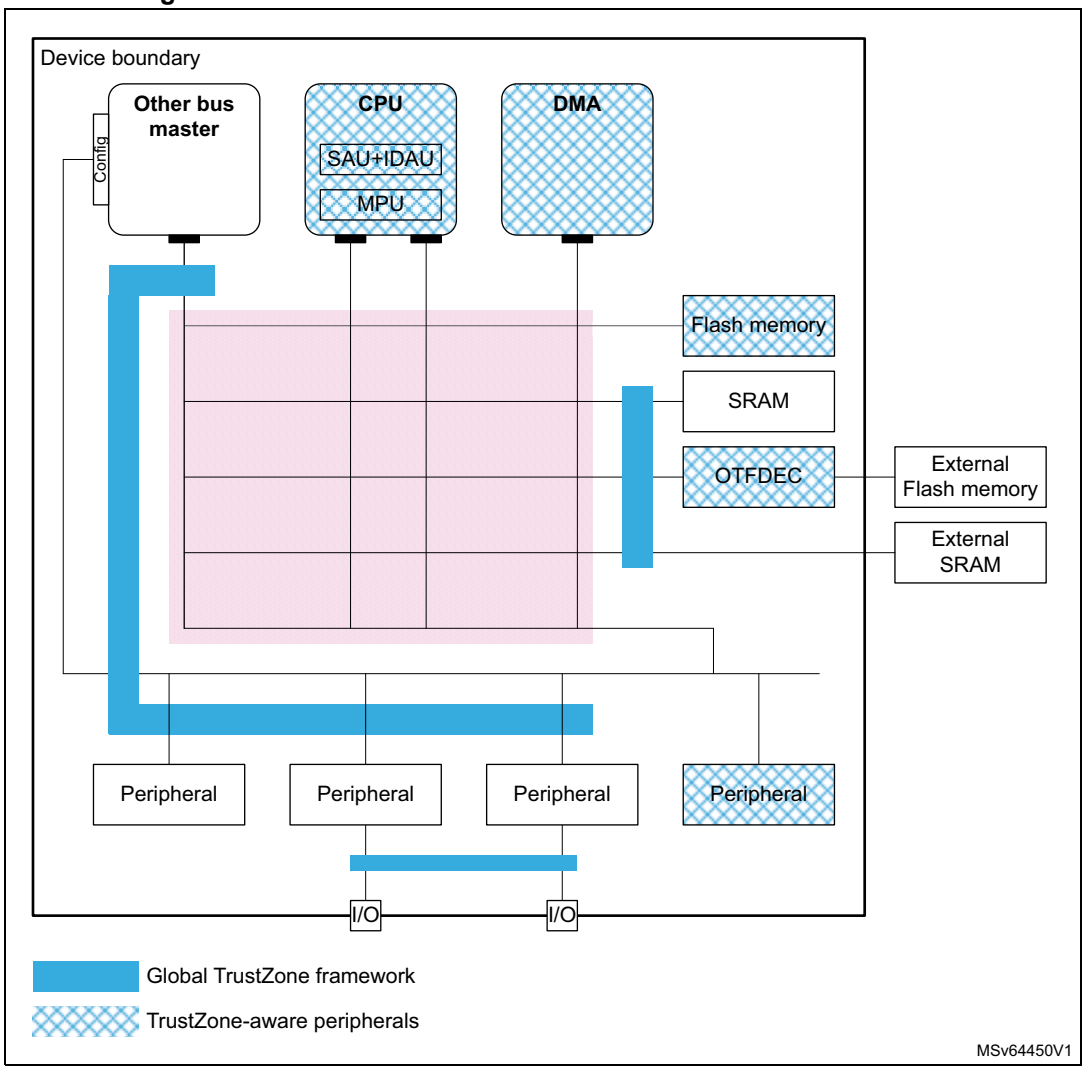
For more information on memory security attribution using IDAU/SAU, refer to the application note *TrustZone features on STM32L5 and STM32U5 Series* (AN5347).

3.5.4 Memory and peripheral allocation using GTZC

Global TrustZone framework architecture

On top of the Armv8-M TrustZone security extension in Cortex-M33, the devices embed 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 7. Global TrustZone framework and TrustZone awareness



Securing peripherals with TZSC

When the TrustZone security is active, a peripheral is either securable through the TZSC in GTZC, or is natively TrustZone-aware, as shown in the previous figure:

- 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 the GTZC, if this peripheral is master on the interconnect (SDMMC), it automatically issues secure transactions. The SDMMC is an example of securable master. TrustZone-aware AHB masters like Cortex-M33 or DMAs, drive a 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 (see Section 3.6.2). 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 and privilege for all securable external memories, programming the MPCWM resources as defined in the table below.

Table 6. MPCWMx resources

Memory	MPC resource	Type of filtering	Number of regions	Default security	On-the-fly decryption(1)
OCTOSPI1	MPCWM1	Non-secure privileged or unprivileged region (watermarks)	2	Secure privileged(2)	Yes
FMC_NOR bank	MPCWM2		2		No
FMC_NAND bank	MPCWM3		1		
Backup SRAM (BKPSRAM)	MPCWM4		1		
OCTOSPI2	MPCWM5		2		Yes

1. Using the OTFDEC.
2. Assuming TrustZone is activated on the device, non-secure unprivileged otherwise.

The MPCBB resources in GTZC provide the capability to configure the security and privilege of embedded SRAM blocks, as defined in the table below.

Table 7. MPCBBx resources

Memory	MPC resource	Type of filtering	Memory size (Kbytes)	Block size (Bytes)	Number of super-blocks	Default security
SRAM1	GTZC1_MPCBB1	Block based, managing security and privilege	192	512(1)	12	Secure privileged(2)
SRAM2	GTZC1_MPCBB2		64		4	
SRAM3	GTZC1_MPCBB3		512		32	
SRAM4	GTZC2_MPCBB4		16		1	

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

Applying GTZC configurations

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

- statically programmed during the secure boot, locked and not changed afterwards
- dynamically re-programmed using a specific application code or real-time kernel

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

- MPCBB secure blocks or MPCWM non-secure region size can be changed by a secure software. This software must be privileged for MPCWM, can be unprivileged if the particular block is not privileged-only.

- The 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 these peripherals can be found in *Section 5: Global TrustZone controller (GTZC)*.

Note: When the TrustZone is deactivated, the resource isolation hardware GTZC can still be used to isolate peripherals to privileged code only (see Section 3.6.2).

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

TrustZone-aware peripherals

The devices include the following TrustZone-aware peripherals:

- GPIOA to GPIOI, configured in LPGPIO alternate function or not
- GTZCx_MPCBB, GTZCx_TZIC and GTZCx_TZSC (GTZC blocks)
- OTFDEC1/2, writable only in secure if TZEN = 1
- EXTI
- Flash memory
- RCC and PWR
- GPDMA and LPDMA
- SYSCFG registers
- RTC and TAMP
- MCU debug unit DBGMCU

The way illegal accesses to those peripherals are monitored through the TZIC registers is described in *Section 5: Global TrustZone controller (GTZC)*.

For more details, refer to *Section 3.5.5*.

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 peripherals, 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 and privileged.

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.

3.5.5 Managing security in TrustZone-aware peripherals

This section gives more details on how the security is implemented in the TrustZone-aware peripherals listed in the previous section.

Embedded Flash memory

When the TrustZone security is enabled through option bytes (TZEN = 1), the whole Flash memory is secure after reset and the following protections, shown in the figure below, are available to the application:

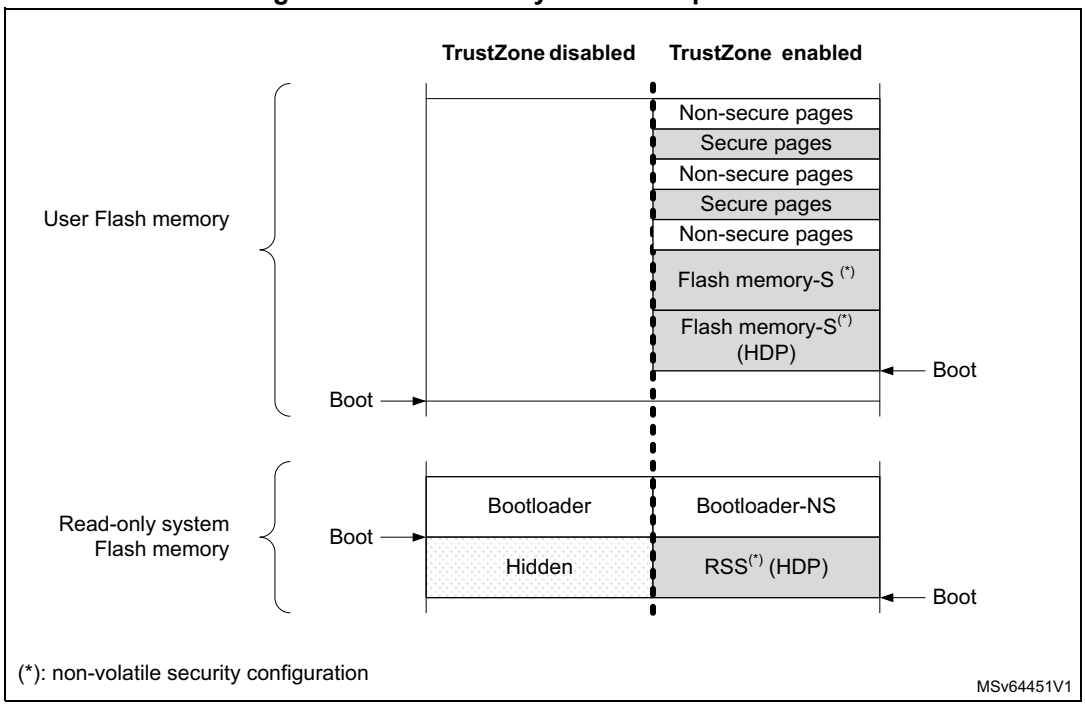
- non-volatile user secure areas, defined with non-volatile secure user option bytes
  - watermark-based secure only area (x2)
  - secure hide protection (HDP) area, stickily hidden after boot (x2)
- volatile user secure pages, defined with volatile secure registers (lost after reset)
  - Any page set 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 are aligned on the Flash memory page granularity.

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

Erase or program operations can be available to secure (resp. non-secure) code only for secure (resp. non-secure) pages or memory. A Flash memory is considered secure if at least one page is secure.

Figure 8. 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 the application sets the HDPx_ACCDIS bit, data read, write and instruction fetch on this HDP area are denied until next system reset.

For example, the software code in the secure Flash HDP area can be executed only once, with any further access to this area denied until the next system reset. Additionally, any Flash memory page belonging to an active HDP area cannot be erased anymore.

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

See *Section 7: Embedded Flash memory (FLASH)* for more details.

On-the-fly encryption/decryption (OTFDEC)

When the TrustZone security is activated (TZEN = 1), the OTFDEC 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 each OTFDEC are write-only.

See *Section 3.9.3* for more details on this cryptographic engine.

Direct memory access controllers (LPDMA and GPDMA)

When a DMA channel x is defined as secure (SECx = 1 in LP/GPDMA_SECCFGR), the source and destination transfers can be independently set as secure or non-secure by a secure application using SSEC and DSEC bits in LP/GPDMA_CxTR1. The table below summarizes these security options available in each DMA channel.

Table 8. DMA channel use (security)(1)

Destination type	Secure DMA channel x (SECx = 1)		Non-secure DMA channel y (SECy = 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.

When a channel is configured as secure:

- Registers allocated to this channel (excluding LP/GPDMA_SECCFGR, LP/GPDMA_PRIVCFGR and LP/GPDMA_RCFGLOCKR) are read as zero. Write are ignored for non-secure accesses. A secure illegal access event may also be triggered toward the TZIC peripheral.

  - Writes to LP/GPDMA_SECCFGR and LP/GPDMA_RCFGLOCKR must be secure. For each bits in LP/GPDMA_PRIVCFGR, write must be secure if corresponding bit in LP/GPDMA_SECCFGR is set.
- In linked-list mode, the loading of the next linked-list data structure from memory is performed with secure transfers.
- When switching to a non-secure state, the secure application must abort the channel or wait until the secure channel is completed before doing the switch.

Note: DMA secure channels are not available when TrustZone is deactivated.

When a channel is configured as non-secure, in linked-list mode, the loading of the next linked-list data structure from memory is performed with non-secure transfers.

See *Section 18: Low-power direct memory access controller (LPDMA)* and *Section 17: General purpose direct memory access controller (GPDMA)* for more details.

Power control (PWR)

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

- low-power mode setup
- wakeup (WKUP) pins definition
- voltage detection and monitoring
- Backup domain control

Other PWR configuration bits become secure:

- when the system clock selection is secure in the RCC: the voltage scaling (VOS) and the regulator booster (BOOSTEN) configurations become secure.
- when a GPIO is configured as secure: its corresponding bit for pull-up/pull-down configuration in Standby mode becomes secure.
- when the USB Type-C/USB power delivery interface (UCPD) is configured as secure in TZSC: PWR_UCPDR register becomes secure.

See *Section 10: 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 become TrustZone-aware:

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

Note: Refer to Section 3.5.4 for the list of securable and TrustZone-aware peripherals.

The SHSI configuration and status bits are secured when the SAES is configured as secure.

Additionally, the following configurations can be made secure-only using RCC_SECCFGR:

- external clock (such as HSE or LSE), internal oscillator (such as HSI, MSI, or LSI)
- main PLL and AHB prescaler
- system clock source selection
- reset flag clearing

- automatic internal oscillator waking up configuration

See *Section 11: 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.

After a Backup domain power-on reset, all RTC registers can be read or written in both secure and non-secure modes. The secure boot code can then change this security setup, making registers Alarm A, alarm B, wakeup timer and timestamp secure or not, using RTC_SECCFGR.

When the SEC bit is set in secure-only RTC_SECCFGR:

- Writing the RTC registers is possible only in secure mode.
- Reading RTC_SECCFGR, RTC_PRIVCFGR, 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 SEC is cleared in secure-only RTC_SECCFGR, it is still possible to restrict access in secure mode to some RTC registers by setting dedicated control bits: INITSEC, CALSEC, TSSEC, WUTSEC, ALRASEC and ALRBSEC.

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

See *Section 54: Real-time clock (RTC)* for more 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.

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

When TAMPSEC is set in TAMP_SECCFGR:

- 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_SECCFGR, TAMP_PRIVCFGR and TAMP_MISR) returns zero if the access is non-secure. Backup registers have their own read protection (see below).

The application can also:

- make TAMP_COUNTR register read and write secure-only by setting the CNT1SEC bit in TAMP_SECCFGR secure register
- in backup registers increase security for two of the three protection zones configured using BKPRWSEC[7:0] and BKPWSEC[7:0] bitfields in TAMP_SECCFGR:
  - 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 55: Tamper and backup registers (TAMP)* for more details.

General-purpose I/Os (GPIO)

When the 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 a 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 devices 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.

The TrustZone-aware logic around GPIO ports, used as alternate function, is summarized in the table below.

Table 9. 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 an analog function with an analog switch is used, the connection to the peripherals listed in the table below, is blocked by hardware when the peripheral is non-secure and the I/O is secure.

Table 10. Non-secure peripheral functions that are not connected to secure I/Os

Peripheral	Analog function(1)	Input	Output	How to set a peripheral or function as secure
ADC1	ADC1_INy (y = 1 to 17)	X	-	Set ADC12SEC in GTZC1_TZSC_ SECCFGR3
ADC4	ADC4_INy (y = 1 to 19)			Set ADC4SEC in GTZC2_TZSC_ SECCFGR1
OPAMPx (x = 1, 2)	OPAMPx_VINy (x = 1, 2; y = 1, 2)			Set OPAMPSEC in GTZC2_TZSC_ SECCFGR1
COMPx (x = 1, 2)	COMPx_INy (x = 1; y = 1 to 3) COMPx_INy (x = 2; y = 1, 2)			Set COMPSEC in GTZC2_TZSC_ SECCFGR1

1. Used to find the I/O corresponding to the signal/function on the package (refer to the product datasheet).

Finally, regarding GPIO and security, the table below summarizes the list of peripheral functions that do not have any hardware protection linked to TrustZone. The listed signals (input and/or outputs) are not blocked when the I/O is set as secure, and the associated peripheral function is non secure.

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

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

It is important that, for each case described in the table below, the 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 must configure the TAMP as secure even if it is not used by the secure application.

Note: How to make a peripheral secure is summarized in the right column of the table below.

Table 11. Non-secure peripheral functions that can be connected to secure I/Os

Peripheral	Signal[1]	Input	Output	How to set the peripheral or function as secure
DAC	DAC1_OUTx (x = 1, 2)	-	X	Set DAC1SEC in GTZC2_TZSC_SECCFGR1.
USB	OTG_FS_VBUS	X	X	Set OTGSEC in GTZC1_TZSC_ SECCFGR3.
UCPD	UCPD1_CCx (x = 1, 2)	X	X	Set UCPD1SEC in GTZC1_TZSC_ SECCFGR1.
	UCPD1_DBx (x = 1, 2)	X	-	
TAMP	TAMP_INx (x = 1 to 8)	X	-	Set TAMPSEC in TAMP_SECCFGR
	TAMP_OUTx (x = 1 to 8)	-	X	
RTC	RTC_OUTx (x = 1, 2)	-	X	Set SEC in RTC_SECCFGR.
	RTC_TS	X	-	Set TSSEC in RTC_SECCFGR.
PWR	WKUPx (x = 1 to 8)	X	-	Set WUPxSEC in PWR_SECCFGR.
RCC	LSCO	-	X	Set LSESEC in RCC_SECCFGR.
EXTI	EXTIx (x= 0 to 22)	X	-	Set SECx bit in EXTI_SECCFGR.

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

Refer to *Section 13: General-purpose I/Os (GPIO)* for more details.

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_SECCFGR1. When an input event is configured as secure, only a secure application can change the configuration (including security if applicable), change the masking or clear the status of this input event.

The security configuration in EXTI_SECCFGR1 can be globally locked after reset in EXTI_LOCKR.

See *Section 21: Extended interrupts and event controller (EXTI)* for more details.

System configuration controller (SYSCFG)

Like all TrustZone-aware peripherals, when the TrustZone security is activated (TZEN = 1), 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 15: System configuration controller (SYSCFG)* for more details.

Microcontroller debug unit (DBGMCU)

The MCU debug component (DBGMCU) helps the debugger, providing support for:

- low-power mode behavior during debug
- peripheral freeze during debug, applicable to I2Cs, IWDG, WWDG, timers, low-power timers, and LP/GPDMA channels

The DBGCMU is a TrustZone-aware peripheral, managing accesses to its control registers as described in the table below.

Table 12. TrustZone-aware DBGMCU access management

Debug profile	Peripheral status(1)	DBG_xx_STOP control bits	
		Write access	Read access
Non-secure invasive (SPIDEN = 0)	NS	Yes (S(2) or NS)	Yes (S or NS)
	S	None (S or NS)	
Secure invasive (SPIDEN = 1)	NS	Yes (S or NS)	
	S	Yes (S only)	

1. As reported by the GTZC, the TrustZone-aware peripheral or the DMA channel.
2. Secure access from debugger is converted to non-secure access in the device.

Refer to *Section 65.12: Microcontroller debug unit (DBGMCU)* for more details.

3.5.6 Activating TrustZone security

The TrustZone is deactivated by default in all STM32U575/585 devices. It can be activated by setting the TZEN user option bit in FLASH_OPTR when in 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 (with their HDP zone), are defined with non-volatile registers FLASH_SECWMxR (x = 1, 2). Default secure option bytes setup is all user Flash secure, without HDP area defined.
  - Volatile block-based security attributions of the Flash memory are non-secure. The secure boot code can change this setup, making blocks secure.

- Embedded SRAM memories
  - All SRAMs are secure, as defined in GTZC/MPCBB (see *Section 3.5.4*). The secure boot code can change this security setup, making blocks secure or not.
- External memories
  - All memory devices connected to the FSMC and OCTOSPIs are secure, as defined in GTZC/MPCWM (see *Section 3.5.4*). The secure-boot code can change this security setup, making components secure or not.
- All GPIOs are secure.
- All LP/GPDMA channels are non-secure.
- Backup registers are non-secure.
- 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 Section 3.5.4 for the list of securable and TrustZone-aware peripherals.

3.5.7 Deactivating TrustZone security

Once TrustZone is activated, it can only be deactivated during a RDP regression to Level 0.

Note: Such RDP regression triggers the erase of embedded memories (SRAM2, Flash memory), and the reset of all peripherals, including the OTFDEC and all crypto engines.

After the TrustZone deactivation, most of the features mentioned in *Section 3.5* are no longer available:

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

Note: When the TrustZone is deactivated, the resource isolation using privilege stays available (see Section 3.6.2 for details).

For more information, refer to the application note *Arm® TrustZone® features for STM32L5 and STM32U5 Series* (AN5347).

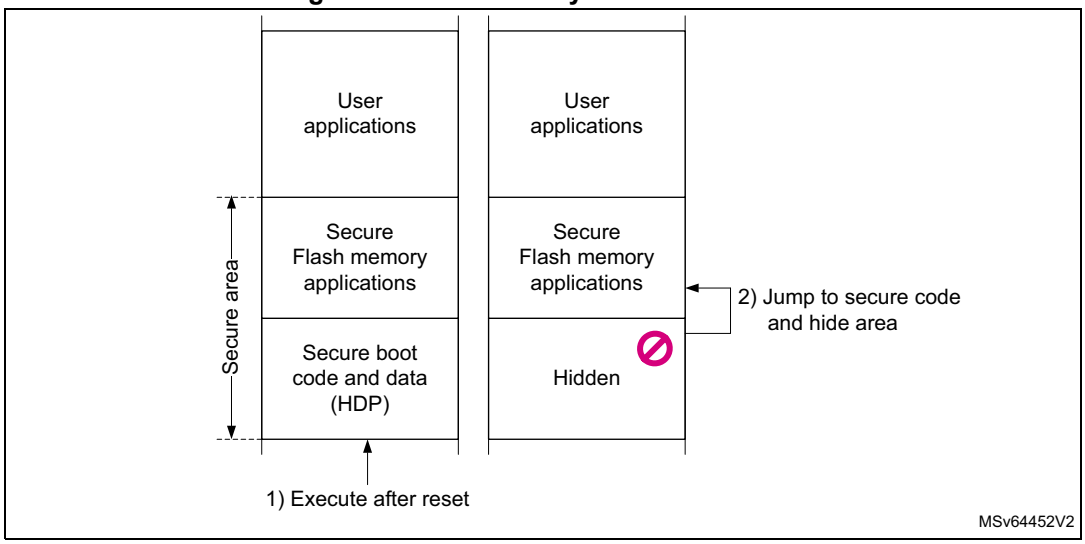
3.6 Other resource isolations

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

3.6.1 Temporal isolation using secure hide protection (HDP)

When the TrustZone security is enabled (TZEN = 1), the embedded Flash memory allows an HDP area per watermarked-secure area of each bank (8-Kbyte page granularity) to be defined. The code executed in this HDP area, with its related data and keys, can be hidden after boot until the next system reset. The hide protection principle is pictured on the figure below.

Figure 9. Flash memory secure HDP area



When the HDPxEN and HDPx_ACCDIS bits (x = 1, 2) are set, data read, write and instruction fetch on the area defined by SECWMx_PSTRT and HDPx_PEND option bytes, are denied until the next device reset.

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

The HDP area can be resized by a secure application if the area is not hidden and if RDP Level ≠ 2.

3.6.2 Resource isolation using Cortex privileged mode

In parallel to the TrustZone isolation described in *Section 3.5*, the hardware and software resources of STM32U575/585 devices can be partitioned so that they are restricted to software running in Cortex privileged mode.

Thanks to this hardware isolation technology, available even if TrustZone is deactivated (TZEN = 0), critical code or data can be protected against intentional or unintentional tampering from the more exposed unprivileged code.

Memory and peripheral privileged allocation using MPU

The Cortex-M33 MPU divides the unified memory into eight regions, each aligned to a multiple of 32 bytes. Each memory regions can be programmed to generate faults when accessed inappropriately by unprivileged software.

Memory and peripheral privileged allocation using GTZC

For the Cortex-M33 master, to complement the coarse isolation provided by the MPU, the GTZC reinforces, in a flexible way, the isolation between privileged and unprivileged tasks, for peripherals and selected memories.

For masters other than the Cortex-M33, the GTZC can assign them as unprivileged initiators, automatically protecting resources defined as privileged against this master.

- Securing peripherals with TZSC (privileged-only)

  In the devices, a peripheral is either securable privileged-only through GTZC, or is natively privileged-aware:

  – A securable privileged-only peripheral or memory is protected by an AHB/APB firewall gate that is controlled by the TZSC.

  – A privileged-aware peripheral or memory is connected directly to AHB or APB interconnect, implementing a specific behavior such as a subset of registers or a memory area is privilege-only.

    When such peripheral is made privileged-only with GTZC, if it is master on the interconnect (SDMMC), it automatically issues privileged transactions. Privilege-aware masters like GPDMA or LPDMA, drive privileged signal in the AHB interconnect according to their internal privileged mode, independently to the GTZC.

    The list of securable peripherals can be found in *Section 5: Global TrustZone controller (GTZC)*.

- Securing memories with TZSC and MPCBB (privileged-only)

  The TZSC logic in GTZC provides the capability to manage the privilege level for all securable external memories, programming the MPCWM resources defined in *Section 3.5.4*.

  Similarly, the GTZC provides the capability to configure the privilege level of embedded SRAM blocks, programming the MPCBB resources defined in *Section 3.5.4*.

- Error management (privileged-only)

  – Any unprivileged transaction trying to access a privileged resource is considered as illegal. There is no illegal access event generated for illegal unprivileged read and write accesses.

  – The addressed resource follows a silent-fail behavior, returning all zero data for read and ignoring any write.

  – When an illegal unprivileged access occurs, no bus error is generated, except when this access is an instruction fetch, accessing a privileged memory or a peripheral register.

Managing security in privileged-aware peripherals

TrustZone-aware peripherals also implement privileged-only access mode. The privileged protection is valid even if TZEN = 0:

- *Embedded Flash memory*

  By default all embedded Flash registers can be read or programmed in both privileged and unprivileged modes.

  When secure privileged bit SPRIV is set in FLASH_PRIVCFGR, reading and writing the Flash secure registers are possible only in privileged mode. Write access to this bit is ignored if TrustZone is deactivated (TZEN = 0).

  When non-secure privileged bit NSPRIV is set in FLASH_PRIVCFGR, reading and writing the Flash non-secure registers are possible only in privileged mode.

  Regarding privileged protection of the embedded Flash memory, the devices offer the following features:

  – The system Flash memory can be accessed both in privileged and unprivileged modes.

  – Each watermark-based secure area, including its secure HDP area, is accessible in secure-privileged and secure-unprivileged mode, if applicable.

  – Each 8-Kbyte page of the embedded Flash memory can be programmed on-the-fly as privileged only, using the block-based privileged configuration registers FLASH_PRIV1BBRx and FLASH_PRIV2BBRx. An unprivileged page is accessible by a privileged or unprivileged access.

Note: Switching a page from privileged to unprivileged does not erase the content of the page.

When applicable, an erase or program operation is always available to privileged code, and is available to unprivileged code only for unprivileged pages or unprivileged memory.

- *On-the-fly encryption/decryption (OTFDEC)*

  When privileged bit PRIV is set in OTFDEC_PRIVCFGR, the OTFDEC can only be initialized by a privileged application.

Note: OTFDEC_PRIVCFGR can be read by both privileged and unprivileged code.

- *Direct memory access controllers (LPDMA and GPDMA)*

  When a DMA channel x is defined as privileged (PRIVx = 1 in LP/GPDMA_PRIVCFGR), special rules apply when accessing privileged/unprivileged source or destination. Those rules are summarized on the table below.

Table 13. DMA channel use (privilege)

Destination	Privileged DMA channel x (PRIVx = 1)		Unprivileged DMA channel y (PRIVy = 0)	
	Privileged source	Unprivileged source	Privileged source	Unprivileged source
Privileged	OK		Transfer blocked(1)	
Unprivileged			Transfer blocked	OK

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

See *Section 18: Low-power direct memory access controller (LPDMA)* and *Section 17: General purpose direct memory access controller (GPDMA)* for more details.

- *Power control (PWR)*

  By default, after a power-on or a system reset, all PWR registers but PWR_PRIVCFGR, can be read or written in both privileged and unprivileged modes.

  When secure privileged bit SPRIV is set in PWR_PRIVCFGR, reading and writing the PWR securable registers are possible only in privileged mode. Write access to this bit is ignored if TrustZone is disabled (TZEN = 0).

  When non-secure privileged bit NSPRIV is set in PWR_PRIVCFGR, reading and writing the PWR non-secure registers are possible only in privileged mode.

  See *Section 10: Power control (PWR)* for details.

- *Secure clock and reset (RCC)*

  By default, after a power-on or a system reset, all RCC registers but RCC_PRIVCFGR can be read or written in both privileged and unprivileged modes.

  When secure privileged bit SPRIV is set in RCC_PRIVCFGR, reading and writing the RCC securable bits are possible only in privileged mode. Write access to this bit is ignored if TrustZone is disabled (TZEN = 0).

  When non-secure privileged bit NSPRIV is set in RCC_PRIVCFGR, reading and writing the RCC non-secure bits are possible only in privileged mode.

  See *Section 11: Reset and clock control (RCC)* for details.

- *Real time clock (RTC)*

  By default after a Backup domain reset, all RTC registers but RTC_PRIVCFGR, can be read or written in both privileged and unprivileged modes.

  When PRIV bit is set in privileged-only RTC_PRIVCFGR:

  – Writing the RTC registers is possible only in privileged mode.

  – Reading the RTC_SECCFGR, RTC_PRIVCFGR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in privileged and unprivileged modes.

  All the other RTC registers can be read only in privileged mode.

  When PRIV bit is cleared in privileged-only RTC_PRIVCFGR register, it is still possible to restrict access to privileged mode to some RTC registers by setting dedicated control bits: INITPRIV, CALPRIV, TSPRIV, WUTPRIV, ALRAPRV or ALRBPRIV.

  See *Section 54: Real-time clock (RTC)* chapter for details.

- *Tamper and backup registers (TAMP)*

  By default after any Backup domain reset, all TAMP registers but TAMP_PRIVCFGR can be read or written in both privileged and unprivileged modes.

  When PRIV bit is set in privileged-only TAMP_PRIVCFGR:

  – Writing the TAMP registers is possible only in privileged mode, except for the backup registers and the monotonic counters that have their own protection setting.

  – Reading the TAMP_SECCFGR or TAMP_PRIVCFGR is always possible in privilege and unprivilege modes. All the other TAMP registers can be read only in privilege mode, except for the backup registers and the monotonic counters that have their own protection setting.

The application can also:

– make TAMP_COUNT1R register read and write privileged-only by setting the CNTPRIV bit in TAMP_PRIVCFGR

– increase security for two of the three protection zones in backup registers, using BKPRWPRIV and BKPWPRIV bits in TAMP_PRIVCFGR:

– Make protection zone 1 read privileged, write privileged.

– Make protection zone 2 read privileged or unprivileged, write privileged.

– Protection zone 3 is always read and write privileged or unprivileged.

- *General-purpose I/Os (GPIO)*

  All GPIO registers can be read and written by privileged and unprivileged accesses, whatever the security state (secure or non-secure).

- *Extended interrupts and event controller (EXTI)*

  The EXTI peripheral is able to protect event register bits from being modified by unprivileged accesses. The protection is individually activated per input event via the register bits in the privileged-only EXTI_PRIVCFGR1 register. When an input event is configured as privileged, only a privileged application can change the configuration (including security if applicable), change the masking or clear the status of this input event.

  The security configuration in EXTI_PRIVCFGR1 can be globally locked after reset in EXTI_LOCKR.

  See *Section 21: Extended interrupts and event controller (EXTI)* for more details.

- *System configuration controller (SYSCFG)*

  All SYSCFG registers can be read and written in both privileged and unprivileged modes, except:

  – FPUSEC bit in SYSCFG_SECCFGR registers (privileged only)

  – SYSCFG registers for CPU configuration: SYSCFG_CSLOCKR, SYSCFG_FPUIMR and SYSCFG_CNSLCKR

  See *Section 15: System configuration controller (SYSCFG)* for more details.

3.7 Secure execution

Through a mix of special software and hardware features, the 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.

3.7.1 Memory protection unit (MPU)

The Cortex-M33 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-M33 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.

3.7.2 Embedded Flash memory write protection

The embedded Flash memory write protection (WRP) prevents illegal 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: two write-protected areas can be defined in each bank, with the granularity of a 8-Kbyte page.

WRP areas can be modified through option byte changes unless corresponding FLASH_WRPxA/BR has its UNLOCK option bit cleared (meaning ROM emulation). UNLOCK can be set only when regressing from RDP Level 1 to Level 0.

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

3.7.3 Tamper detection and response

Principle

The devices include active protection of critical security assets against temperature, voltage and frequency attacks, with the following 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 V_{DD} or V_{BAT}

See *Section 55: Tamper and backup registers (TAMP)* for more details.

Tamper detection sources

The devices support eight active input/output pins, allowing four independent active-tamper meshes, or up to seven meshes if the same output pin is shared by several input pins (for a total of eight active-tamper I/Os). The active-tamper balls are mapped in the center of packages that can be used in POS market (such as WLCSP90).

The active pins are clocked by the LSE, and are functional in all system operating modes (Run, Sleep, Stop, Standby or Shutdown), and in V_{BAT} mode.

Detection time is programmable, and a digital filtering is available (tamper triggered after two false comparison in four consecutive comparison samples).

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

The internal tamper sources are listed in the table below.

Table 14. Internal tampers in TAMP

Tamper input	NOER bit number in TAMP_CR3	Tamper source
itamp1	0	Backup domain voltage continuous monitoring, functional in V_{BAT} mode
itamp2	1	Temperature monitoring, functional in V_{BAT} mode
itamp3	2	LSE monitoring(1), functional in V_{BAT} mode
itamp4	-	Not used
itamp5	4	RTC calendar overflow (rtc_calovf)
itamp6	5	JTAG/SWD access when RDP > 0
itamp7, 12, 13	6, 11, 12	Voltage monitoring (V_{CORE}, V_{REF+}), through ADC analog watchdog, functional down to Stop 2 mode
itamp8	7	Monotonic counter overflow (generated internally)
itamp9	8	Fault generation for cryptographic peripherals (SAES, PKA, AES, RNG)
itamp11	10	IWDG timeout and potential tamper (IWDG reset when at least one enabled tamper flag is set)

1. LSE missing or over frequency detection (> 2MHz), Glitch filter (> 2 MHz).

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 device secrets if the corresponding TAMPxNOER bit is cleared in TAMP_CR2 (for tamper pins) or TAMP_CR3 (for internal tamper). These erasable secrets are:
  - symmetric keys stored in backup registers (x32), in SAES, AES, HASH and OTFDEC (encrypted Flash memory regions are read as zero)
  - asymmetric keys stored in PKA SRAM, erased when V_{DD} is present
  - other secrets stored in SRAM2 and CPU instruction cache memory (SRAM2 erased when V_{DD} is present)
  - non-volatile information used to derive the DHUK in SAES is zeroed until complete SRAM2 erase
  - 2-Kbyte backup SRAM (depending on configuration bit), erased when V_{DD} is present
  - ICACHE and DCACHE erased when V_{DD} is present

Read/write accesses by software to all these secrets can be blocked, by setting the BKBLOCK bit in TAMP_CR2. The device secrets access is possible only when BKBLOCK is cleared, and no tamper flag is set for any enabled tamper source.

If V_{DD} is not present, the secrets that are erased when V_{DD} is present, are only erased at the next V_{DD} power on.

Note: Device secret erase is also triggered by setting the BKERASE bit in TAMP_CR2, or by performing a RDP regression as defined in Section 3.10.1.

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

Software filtering mechanism

Each tamper source can be configured not to launch an immediate erase, by setting the corresponding TAMPxNOER bit in TAMP_CR2 (for external tamper pin) or TAMP_CR3 (for internal tamper).

In such situation, when the tamper flag is raised, access to below secrets is blocked until all tamper flags are cleared:

- DHUK in SAES: fixed to a dummy value
- Backup registers, backup SRAM, SRAM2: read-as-zero, write-ignored
- AES, SAES and HASH peripherals: automatically reset by RCC
- PKA peripheral: reset, with memory use blocked (meaning PKA not usable)

Once the application, notified by the tamper event, analyzes the situation, there are two possible cases:

- The application launches secrets erase with a software command (confirmed tamper).
- The application just clears the flags to release secrets blocking (false tamper).

Note: If the tamper software fails to react to such a tamper flag, an IWDG reset triggers automatically the erase of secrets.

Tamper detection and low-power modes

The effect of low-power modes on a tamper detection are summarized on the table below.

Table 15. 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 that 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.

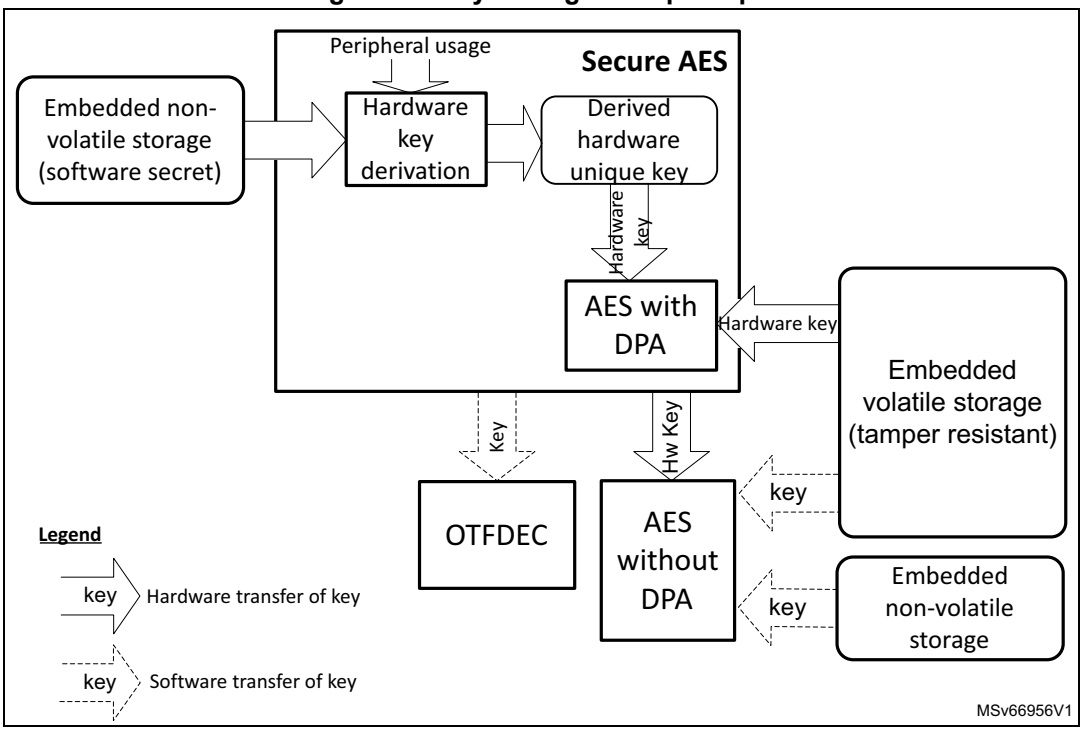
3.8 Secure storage

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

Figure 10 shows how key management service application can use the AES engine for example to compute external image decryption keys. A non-volatile key can be stored in the embedded secure HDP area (see *Section 3.6.1*), while volatile key storage consists in the battery-powered, tamper-protected SRAM or registers in TrustZone-aware TAMP.

Figure 10 also shows keys that are manipulated by software (like OTFDEC keys), or keys that are managed only by hardware (like DHUK). More information on those hardware keys can be found in *Section 3.8.1*.

Figure 10. Key management principle



Details on tamper protection is found in *Section 3.7.3*, while TAMP TrustZone features are briefly described in *Section 3.5.5*.

3.8.1 Hardware secret key management

As shown in the previous figure, the devices propose a better protection for application keys, using hardware secret keys. These AES keys can be made usable to the application, without exposing them in clear-text (unencrypted). Such keys also become immediately unusable in case of tamper.

There are three different sources of hardware secret keys:

- DHUK: derived keys based on 256-bit non-volatile device unique secret in Flash memory. The generation of this key takes into account the TrustZone state and key use state (KMOD).

Note: DHUK is the same for all devices when RDP = 0 (debug/development mode).

- BHK: 256-bit application key stored in tamper-resistant volatile storage in TAMP
  This key is written at boot time, then read/write locked to application until next reset.
- XORK: result of a XOR of BHK and DHUK

These keys can be used:

- as normal key, loading in write-only key registers (software key mode)
- as encryption/decryption key for another key, to be used in the DPA-resistant SAES (wrapped key mode)
- as encryption/decryption key for another key, to be used in a faster AES engine (shared key mode)

3.8.2 Unique ID

The devices store a 96-bit ID that is unique to each device (see *Section 66.1: Unique device ID register (96 bits)*).

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.

Alternatively, the 256-bit device unique key (DHUK) can be used (see *Section 3.8.1*).

3.9 Crypto engines

The 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 provide 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.

3.9.1 Crypto engines features

Table 16 lists the accelerated cryptographic operations available in the devices. Two AES accelerators are available (both can be reserved to secure application only).

Note: Additional operations can be added using firmware.

The PKA can accelerate asymmetric crypto operations (like key pair generation, ECC scalar multiplication, point on curve check). See Section 45: Public key accelerator (PKA) for more details.

Table 16. Accelerated cryptographic operations

Operations	Algo-rithm	Specification	Key lengths (in bit)	Modes
Get entropy	RNG	NIST SP800-90B(1)	N/A	Software and hardware(2) modes running in parallel
Encryption, decryption	AES	FIPS PUB 197 NIST SP800-38A	128, 256	ECB, CBC, CTR(3)
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(4)	RSA	IETF RFC 8017 NIST SP800-56B	Up to 4160	RSAES-OAEP
Signature(4) with hashing Signature verification	RSA	IETF RFC 8017 FIPS PUB 186-4	Up to 4160	PKCS1-v1_5, PSS
	ECDSA	ANSI X9.62 IETF RFC 7027 FIPS PUB 186-4	Up to 640	Refer to the table 'Family of supported curves for ECC operations' in PKA section for details
Key agreement	ECDH	ANSI X9.42		

1. Certifiable using STMicroelectronics reviewed documents.
2. Random numbers distribution to SAES and PKA using a dedicated hardware bus.
3. ECB and CBC chaining modes protected against side-channel and timing attacks in SAES (see *Section 3.9.2*).
4. Private key cryptography protected against side-channel and timing attacks.

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.

3.9.2 Secure AES co-processor (SAES)

The devices provide an additional on-chip hardware AES encryption and decryption engine, that implements counter-measures and mitigations against power and electromagnetic side-channel attacks.

Clocked by a dedicated 48 MHz SHSI clock, SAES is also slower than the AES, in order to provide best-in-class side-channel protections. The SAES engine supports 128-bit or 256-bit key in electronic code book (ECB) or cipher block chaining (CBC) mode.

As shown in *Section 3.8*, the SAES can be used for extra-secure on-chip storage for sensitive information. It can also be made secure-only.

For more information, refer to *Section 42: Secure AES coprocessor (SAES)*.

3.9.3 On-the-fly decryption engine (OTFDEC)

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

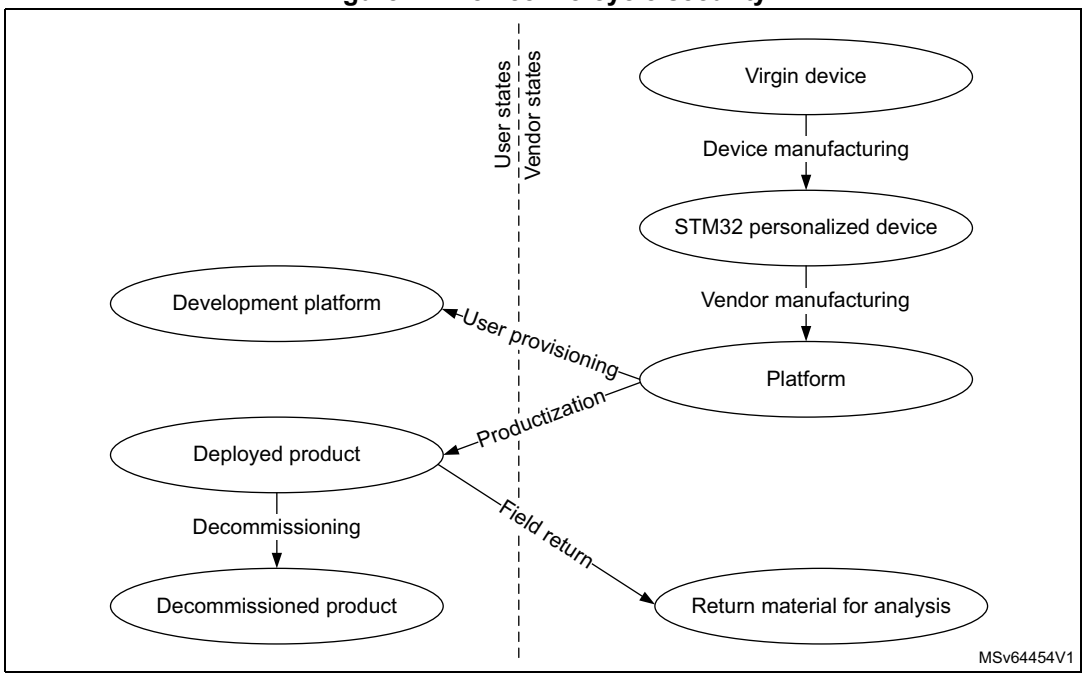
When a tamper event is confirmed in TAMP, all OTFDEC keys are erased and encrypted regions are read as zero until the OTFDEC is properly initialized again.

An OTFDEC typical use is detailed in *Section 3.12.2*. For more details on the peripheral programming, refer to *Section 44: On-the-fly decryption engine (OTFDEC)*.

3.10 Product life-cycle

A typical IoT device life-cycle is summarized in the figure below. For each step, the devices propose secure life-cycle management mechanisms embedded in the hardware.

Figure 11. Device life-cycle security



More details on the various phases and associated transitions, found either at the vendor or end-user premises, are summarized in *Table 17*.

Table 17. Main product life-cycle 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 devices loose the ability to operate securely (like connecting to a managed IoT network).

The features described hereafter contribute to secure the device life-cycle.

3.10.1 Life-cycle management with readout protection (RDP)

The readout protection mechanism (full hardware feature) controls the access to the devices debug, test and provisioned secrets, as summarized in the table below.

Table 18. Typical product life-cycle phases

RDP protection level		Debug	Comments
Level 0	Device 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). Both OEM1 and OEM2 unlocking keys can be provisioned in the Flash user options. The DHUK in SAES peripheral is the same for all devices.
Level 0.5(2)	Device 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. Both OEM1 and OEM2 unlocking keys can be provisioned in the Flash user options. The DHUK is different for every device.
Level 1	Device memories 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. Both OEM1 and OEM2 unlocking keys can be provisioned in the Flash user options. The DHUK is different for every device.
Level 2	Device 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, unless OEM2 unlocking key is activated (see *Table 19*). The DHUK is different for every device.

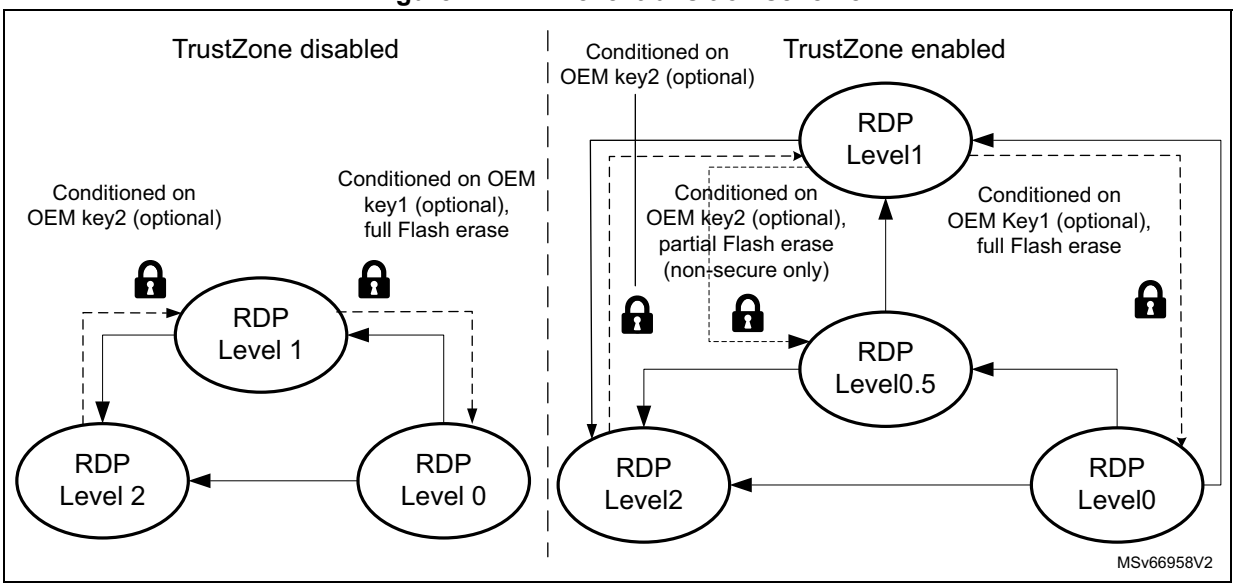
1. Debug is not available when executing RSS code.
2. Only applicable when TrustZone security is activated in the product.

3. External Flash memory area decrypted on-the-fly with the OTFDEC.

Note: OEM1KEY option byte can be modified when OEM1LOCK = 0 (RDP = 0.5 or 1 only).
OEM2KEY option byte can be modified when OEM2LOCK = 0 (RDP = 1 only).

The supported transitions, summarized in the figure below, can be requested (when available) through the debug interface or via the system bootloader.

Figure 12. RDP level transition scheme



As shown in the previous figure, the user Flash memory is automatically erased, either partially or in totality, during a RDP regression from RDP1. Those regressions can be conditioned to dedicated 64-bit password keys, if provisioned by the OEM (see next subsection). During the regression from RDP Level1 to RDP Level 0.5, only non-secure embedded Flash memory is erased, keeping functional, for example, the secure boot and the secure firmware update. In all regressions from Level 1, the OTP area in the Flash memory is kept, all SRAMS and targeted device secrets are erased. These secrets, also erased as response to tamper, are defined in *Section 3.7.3*.

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

For more details on RDP, refer to *Section 7: Embedded Flash memory (FLASH)*.

RDP unlocking sequences

The use of the two OEM password keys described in the last figure, is further described hereafter.

Note: The devices support both permanent RDP Level 2 (legacy mode) or password-based RDP Level 2 regression to Level 1. This Level 2 regression does not erase the application code, and it does not change the RDP Level 1 protections in place.

Details on the password-based regression can be found in *Table 19*.

Table 19. OEM1/2 RDP unlocking methods

OEM1 password options			OEM2 password options		
OEM1 LOCK	**Initial RDP level**	**RDP regression**	**OEM2 LOCK**	**Initial RDP level**	**RDP regression**
1	1	Regression to Level 0 possible only through OEM1 unlock sequence (see below)	1	1	Regression to Level 0.5 possible only through OEM2.1 unlock sequence (see below)
				2	Automatic regression to Level 1 triggered upon successful OEM2.2 unlock sequence (see below)
0		Regression to Level 0 always granted	0	1	Regression to Level 0.5 always granted
				2	Regression to Level 1 never granted RDP remains a permanent state.

- OEM1 unlock sequence, starting at RDP Level 1:
  - Shift password key through JTAG/SWD (see the note below).
  - If this key matches the OEM1KEY provisioned in the device, the application can trigger a regression sequence to Level 0. After the regression is completed, the whole embedded Flash memory and device secrets are erased. The OTP area is not erased.
  - In case of mismatch value, the RDP regression is blocked and RDP Level 1 protections are enforced until next power-on reset.
- OEM2.1 unlock sequence, starting at RDP Level 1:
  - Shift password key through JTAG/SWD under reset (see the note below).
  - If this key matches the OEM2KEY provisioned in the device, the application can trigger a regression sequence to Level 0.5. After the regression is completed, the non-secure embedded Flash memory and device secrets are erased. The OTP area is not erased.
  - In case of mismatch value, the RDP regression is blocked and RDP Level 1 protections are enforced until next power-on reset.
- OEM2.2 unlock sequence, starting at RDP Level 2:
  - Shift password key through JTAG/SWD under reset (see the note below).
  - If this key matches the OEM2KEY provisioned in the device, the device automatically triggers a regression sequence to Level 1. After the regression is completed, a power-on reset has to be performed by the user.
  - In case of mismatch value, the RDP regression is blocked and RDP Level 2 protections are enforced until next power-on reset.

Note: Unlocking the device with a password is possible only once per power cycle.

Shifting the password key through JTAG/SWD corresponds to writing two 32-bit key words, AUTH_KEY[31:0], then AUTH_KEY[63:32], in DBGMCU_DBG_AUTH_HOST register.

JTAG 32-bit device specific ID

Unless the JTAG port is de-activated (OEM2LOCK = 0 and RDP Level = 2), a 32-bit device specific quantity can always be read though the JTAG port. This information is stored in DBGMCU_DBG_AUTH_DEVICE.

The OEM can use this 32-bit information to derive the expected OEM password keys to unlock this specific device.

3.10.2 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 the user fears 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)
  - RDP Level 2
  - non-secure boot address option bytes set in user Flash memory
- If TrustZone is enabled (TZEN = 1)
  - RDP Level 2
  - BOOT_LOCK = 1
  - secure boot address option bytes set in user secure Flash memory

As described in the previous section, the customer can decide to allow any RDP Level 2 part to regress to RDP Level 1, provided the OEM Key2 has been successfully provisioned, and OEM2LOCK option bit is set.

3.11 Access controlled debug

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

3.11.1 Debug protection with readout protection (RDP)

As described in *Section 3.10.1,* the hardware RDP mechanism automatically controls the accesses to the device debug and test. The protection of these debug features are defined in the table below. Possible password-based regressions are described in *Section 3.10.1*.

Table 20. Debug protection with RDP

RDP protection level		Debug features protection
Level 0	Device open	Any debug(1)
Level 0.5(2)	Device partially closed	Secure debug is no longer available.
Level 1	Device memories protected	Non-secure debug can no longer debug code and data stored in the embedded Flash memory, the encrypted external Flash memory(3), SRAM2 and backup registers.
Level 2	Device closed	JTAG is physically deactivated, unless it is kept operational only for password key injection (OEM2LOCK = 1). See *Section 3.10.1* for details.

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 the OTFDEC.

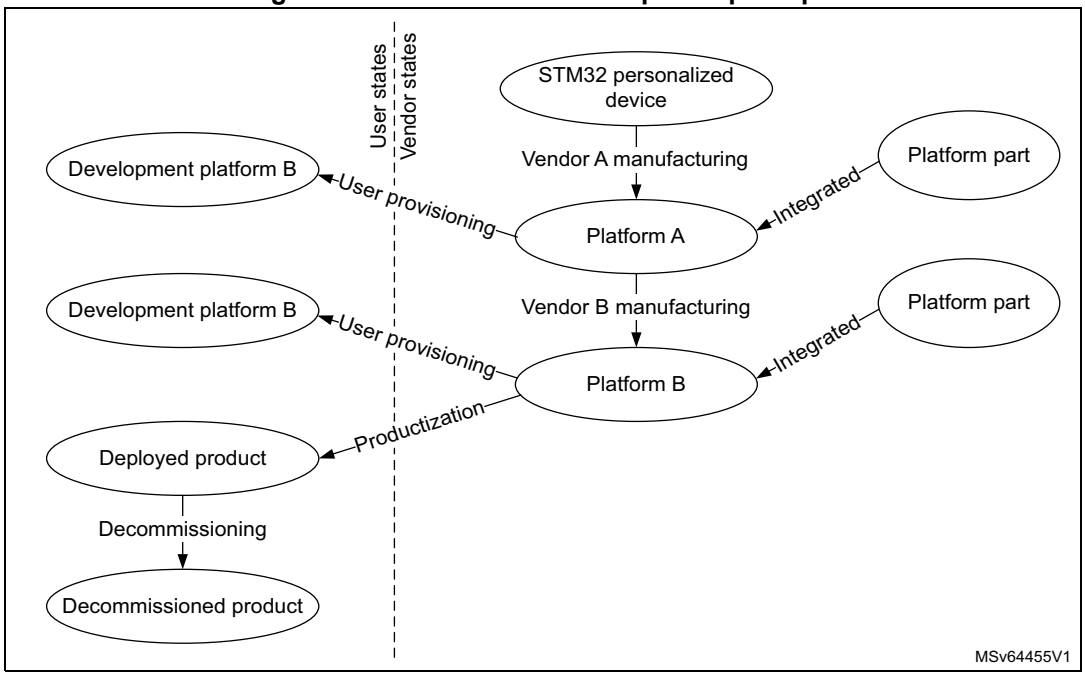
3.12 Software intellectual property protection and collaborative development

Thanks to software intellectual property protection and collaborative model, the devices allow the design of solutions integrating innovative third-party libraries.

Collaborative development is summarized on the figure below. 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, who does 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 13. Collaborative development principle



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

3.12.1 Software intellectual property protection with RDP

As described in *Section 3.10.1,* the hardware RDP mechanism automatically controls the accesses to secrets provisioned in the device.

The protection of these secrets are defined in the table below.

Table 21. Software intellectual property protection with RDP

RDP protection level		Secrets protection
Level 0	Device open	No special protections.
Level 0.5(1)	Device partially closed	All peripherals and memories mapped as secure during secure boot cannot be dumped, debugged or traced
Level 1	Device memories 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 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.

3.12.2 Software intellectual property protection with OTFDEC

As described in *Section 3.9.3,* the OTFDEC associated with the OCTOSPI is able to decrypt on the fly, the encrypted images stored in external SPI Flash memories.

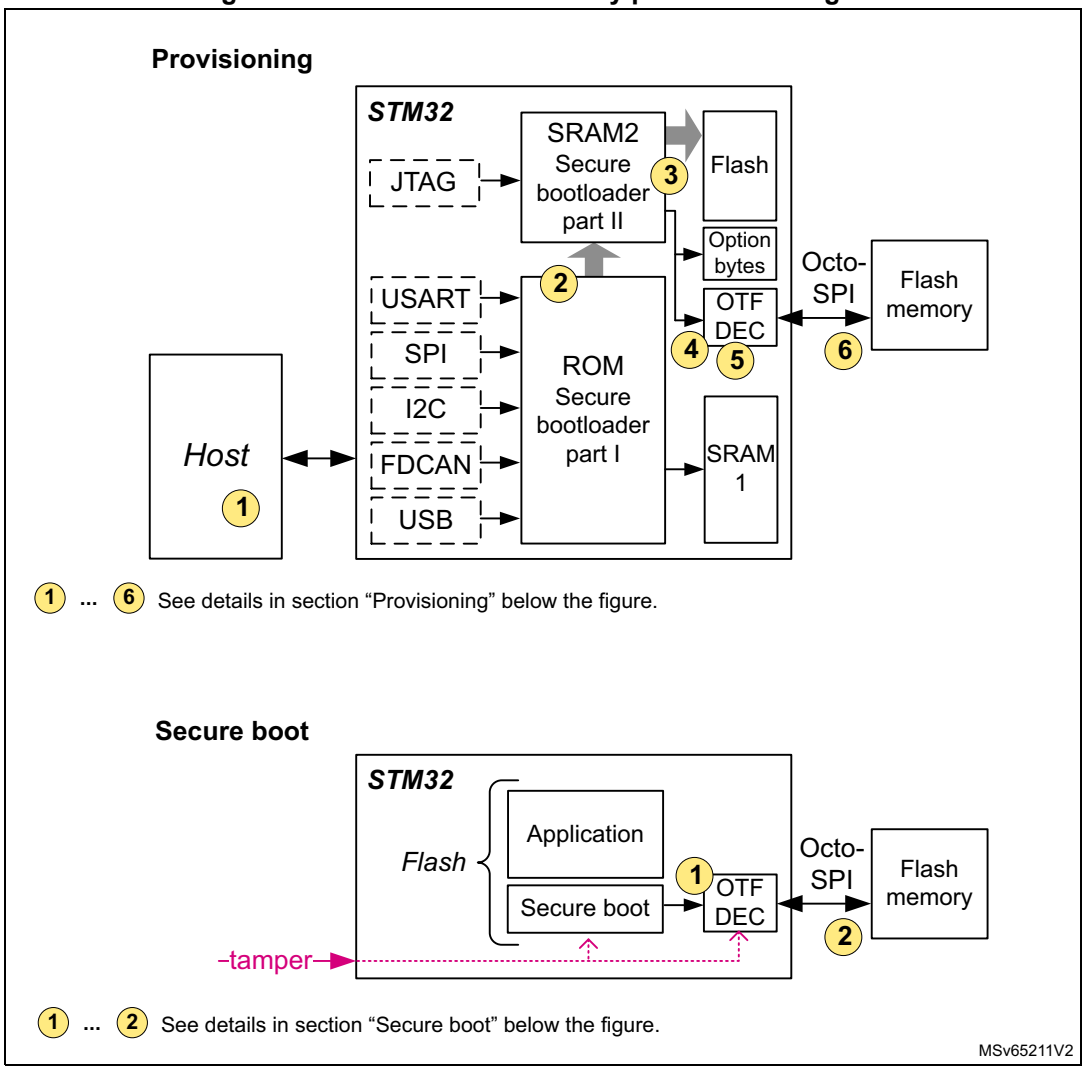
Thanks to this feature, the devices allow the installation of intellectual properties, in one of the following ways:

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

Figure 14 illustrates this last case, with the provisioning, in a non-trusted environment, of software intellectual properties both in the embedded Flash memory and in an external SPI Flash memory (encrypted).

Note: Since the OTFDEC uses the 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 14. 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. The user creates a 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 the system memory, loads the second part of the secure bootloader in SRAM, through the supported communication ports (USART, SPI, I2C, FDCAN, USB and JTAG). This second part runs in the secure SRAM 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. The internal Flash memory is programmed with decrypted option bytes, internal firmware and data, and external firmware and data AES keys. Alternatively, device unique external firmware AES keys can be used instead of such global keys.
4. The OTFDEC is properly initialized with encrypted region 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 the OTFDEC.
6. After a chunk OTFDEC re-encryption, the user external Flash memory programmer is responsible for programming the last encrypted chunks to the external SPI Flash memories through the OCTOSPI.

Alternatively, external firmware and data AES keys for OTFDEC can be generated on the device, if they are not generated by the signing tool.

Secure boot

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

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

For more information on SFI solutions for the devices, refer to the application note *Overview secure firmware install (SFI) (*AN4992).

3.12.3 Other software intellectual property protections

The device additional protections to software intellectual property are:

- Invasive attacks such as physical tampering or perturbation are countered by detection then decommissioning of the device before the detected attack succeeds.
- Non-invasive attacks, such as side channel attacks, are countered by not leaking secret information via side channels such as timing, power and EM emissions.

4 Boot modes

At startup, a BOOT0 pin, NBOOT0 and NSBOOTADDx[24:0]/SECBOOTADD0[24:0] option bytes are used to select the boot memory address that includes:

- Boot from any address in user Flash memory.
- 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.

The bootloader, located in the system memory, 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).

Table 22 and *Table 23* detail the boot modes when TrustZone is disabled or enabled.

Table 22. Boot modes when TrustZone is disabled (TZEN = 0)

NBOOT0 FLASH_ OPTR[27]	BOOT0 pin PH3	NSWBOOT0 FLASH_ OPTR[26]	Boot address option-byte 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]	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]	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 23. Boot modes when TrustZone is enabled (TZEN = 1)

BOOT_LOCK	NBOOT0 FLASH_OPTR[27]	BOOT0 pin PH3	NSWBOOT0 FLASH_OPTR[26]	RSS com-mand	Boot address option-byte 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	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
	-	-	-	≠ 0	N/A	RSS	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 are used 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 either in the secure Flash memory or the non-secure Flash memory depending on TrustZone security option as described in the table below.

Table 24. 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 in secure Flash memory – 0x0C00 0000-0x0C1F FFFF Otherwise the boot address is forced to 0x0FF8 0000.	N/A
1		Any boot address
2		Boot address only in Flash memory – 0x0800 0000-0x081F FFFF Otherwise boot address forced is 0x0800 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 nSWBOOT0 in FLASH_OPTR) is also resampled when exiting Standby mode. Consequently, the BOOT0 pin or user option bit 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 as follows:

- in input mode during the complete reset phase if the option bit NSWBOOT0 is set in FLASH_OPTR, and then switches automatically in analog mode after reset is released (BOOT0 pin)
- in input mode from the reset phase to the completion of the option byte loading if the bit NSWBOOT0 is cleared into FLASH_OPTR (BOOT0 value coming from the option bit), and then switches automatically to the analog mode even if the reset phase is not complete

Embedded bootloader

The embedded bootloader is located in the system memory, programmed by ST during production. Refer to the application note *STM32 microcontroller system memory boot mode* (AN2606).

Embedded root security services (RSS)

The embedded RSS are located in the secure information block, programmed by ST during production. Refer to the application note *Overview secure firmware install (SFI)* (AN4992).

5 Global TrustZone controller (GTZC)

5.1 Introduction

This section describes the global TrustZone controller (GTZC) block that contains the following sub-blocks:

- **TZSC**: TrustZone security controller

  This sub-block defines the secure/privileged state of slave peripherals. It also controls the sub-region area size and properties for the watermark memory peripheral controller (MPCWM). The TZSC 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**: memory protection controller - block based

  This sub-block configures the internal RAM in a TrustZone-system product having segmented SRAM (pages of 512 bytes) with programmable-security and privileged attributes.
- **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 such as:

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

5.2 GTZC main features

The GTZC main features are listed below:

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

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
- a non-secure or public world (such as usual non secure operating system and user space)

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

AHB and APB Peripherals can be categorized as:

- **privileged**: 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 (such as TZIC)
- **securable**: peripherals protected by an AHB/APB firewall stub that is controlled from TZSC to define security properties (optional)
- **non-secure and unprivileged:** 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 privileged/unprivileged.

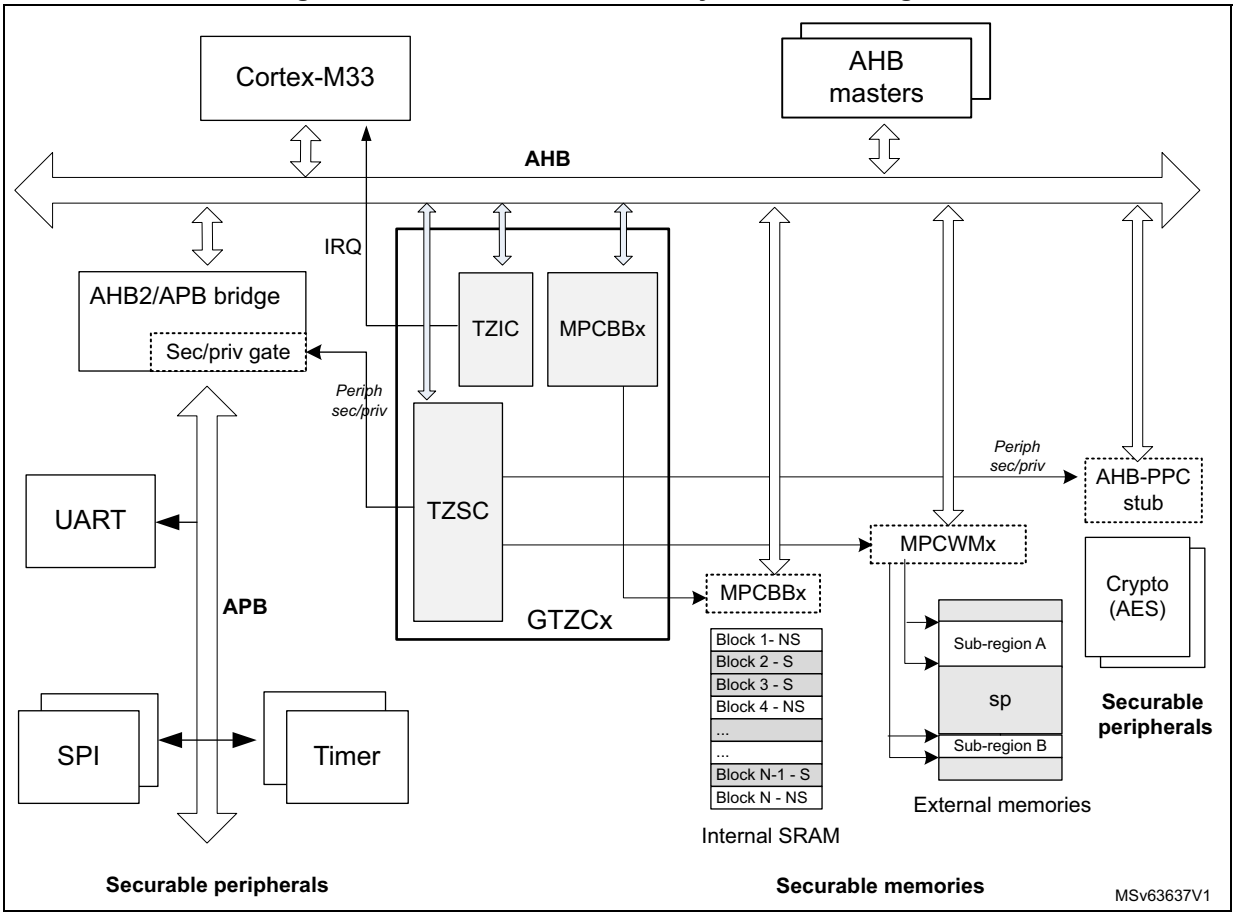
Application information

The TZSC, MPCBB and TZIC 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 region or secure SRAM. Same remark applies to the GTZC_TZSC_SECCFGRx/PRIVCFGRx registers that define secure/privileged state of each peripheral.

The Armv8-M security architecture with secure, securable and TrustZone-aware peripherals is shown in the figure below.

Figure 15. GTZC in Armv8-M subsystem block diagram



5.3 GTZC implementation

The STM32U575/585 devices embed two instances of GTZC.

Table 25. GTZC features

GTZC sub-blocks	GTZC1	GTZC2
TZSC	X	X
TZIC	X	X
MPCBB sub-block (number of MPCBB)	X (3)	X (1)

The tables below shows the address offset of GTZC sub-blocks versus GTZC base address (refer to *Section 2.3* for GTZC1 and GTZC2 base addresses).

Table 26. GTZC1 sub-blocks address offset

GTZC1 sub-block	Address offset
GTZC1_TZSC	0x0
GTZC1_TZIC	0x400
GTZC1_MPCBB1	0x800
GTZC1_MPCBB2	0xC00
GTZC1_MPCBB3	0x1000

Table 27. GTZC2 sub-blocks address offset

GTZC2 sub-blocks	Address offset
GTZC2_TZSC	0x0
GTZC2_TZIC	0x400
GTZC2_MPCBB4	0x800

The table below describes the characteristics of the available MPCWM.

Table 28. MPCWM resource assignment

GTZC	MPC	Target memory interface	Number of sec/non-sec and priv/unpriv regions	Watermark granularity (bytes)
GTZC1	MPCWM1	OCTOSPI1	2	128 K
	MPCWM2	FSMC_NOR bank	2	128 K
	MPCWM3	FSMC_NAND bank	1	128 K
	MPCWM4	BKPSRAM	1	32
	MPCWM5	OCTOSPI2	2	128 K

The table below describes the characteristics of the available MPCBB.

Table 29. MPCBB resource assignment

GTZC	MPC	Resource	Memory size (Kbytes)	Block size (bytes)	Number of blocks	Number of super-blocks
GTZC1	MPCBB1	SRAM1	192	512	384	12
	MPCBB2	SRAM2	64		128	4
	MPCBB3	SRAM3	512		1024	32
GTZC2	MPCBB4	SRAM4	16		32	1

5.4 GTZC functional description

5.4.1 GTZC block diagram

The figure below describes the combined feature of TZSC, MPCBB and TZIC. Each sub-block is controlled by its own AHB configuration port.

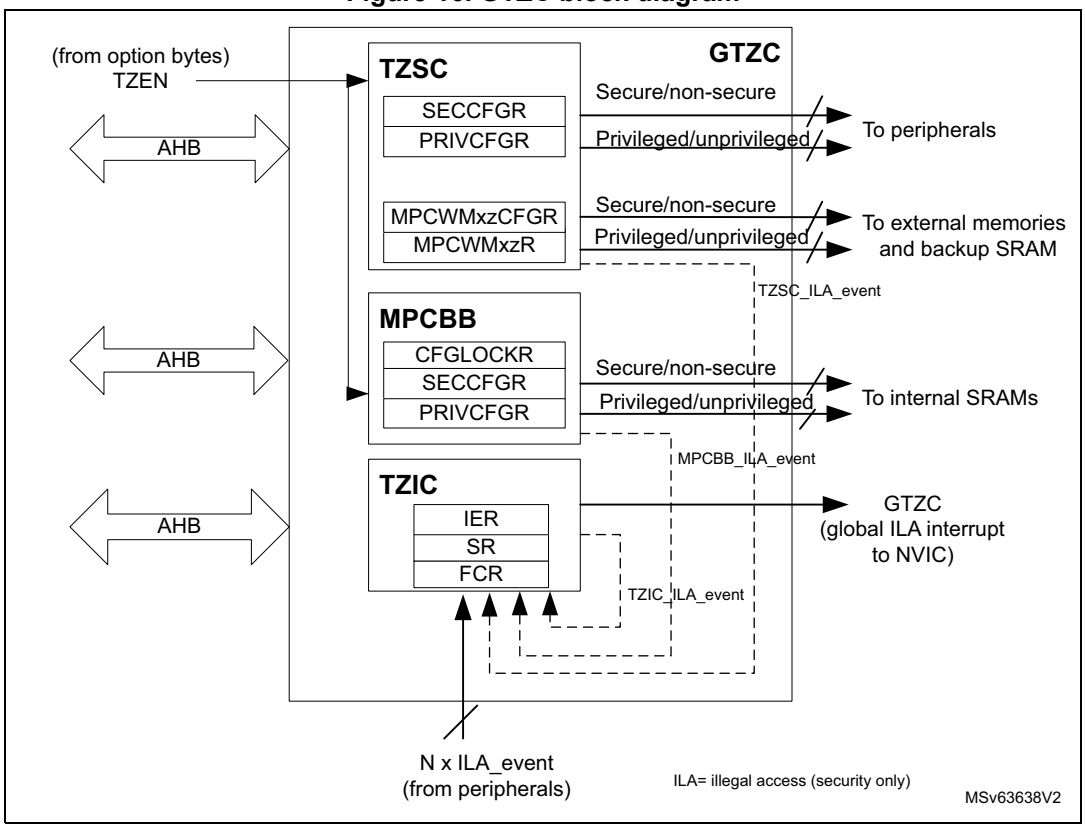
The TZSC defines which peripheral is secure and/or privileged.The privileged configuration bit of a peripheral can be modified by a secure privileged transaction when the peripheral is configured as secure. Otherwise, a privileged transaction (non-secure) is sufficient.

On the opposite, the secure configuration bit of a peripheral can be modified only with a secure privileged transaction if the peripheral is configured as privileged. Otherwise, a secure transaction (unprivileged) is sufficient.

The secure configuration bit of a given ram block can be modified only with a secure privileged transaction if the same RAM block is configured as privileged. Otherwise, a secure transaction (unprivileged) is sufficient.

The TZIC gathers illegal events generated within the system when an illegal access is detected. TZIC can then generate a secure interrupt towards the CPU if needed.

Figure 16. GTZC block diagram



5.4.2 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 interrupt for illegal write access and a bus error for illegal fetch access. However some exceptions exist on secure and privileged configuration registers: these later ones authorize non secure read access to secure registers (see GTZC_TZSC_SECCFGRx and GTZC_TZSC_PRIVCFGRx).

- Illegal secure access

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

  Correct TZIC settings 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 bit in the GTZC_MPCBBz_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 is considered as illegal and generate a bus error.

- Illegal unprivileged access

  Any unprivileged transaction trying to access a privileged resource is considered as illegal. There is no illegal access event generated for illegal read and write access. The addressed resource follows a silent-fail behavior, returning all zero data for read and ignoring any write. No bus error is generated. A bus error is generated when any unprivileged execute transaction tries to access a privileged memory.

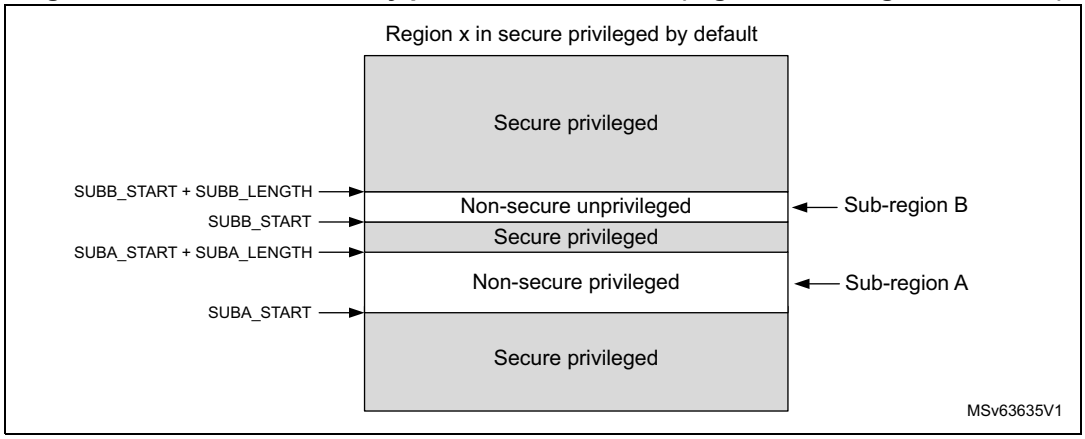
5.4.3 TrustZone security controller (TZSC)

The TZSC is composed of a configurable set of registers, providing the following features:

- Control of secure and privileged state for all peripherals, done through:
  - GTZC_TZSC_SECCFGRx registers to control AHB/APB firewall stubs for the securable peripherals
  - GTZC_TZSC_PRIVCFGRx registers to control AHB/APB firewall stubs for the privileged peripherals
- For watermark memory protection controller (external memories and backup SRAM), two independent regions can be defined and the following fields are used to program:
  - the start of the first protected sub-region on external memory/backup SRAM: SUBA_START[10:0]
  - the length of the first protected sub-region on external memory/backup SRAM: SUBA_LENGTH[11:0]
  - the start of the second protected sub-region on external memory/backup SRAM: SUBB_START[10:0]
  - the length of the second protected sub-region on external memory/backup SRAM: SUBB_LENGTH[11:0]

A control register for each sub-region can be used to enable/disable the watermark memory protection controller as well as defining the right attributes of each sub-region.

Figure 17. Watermark memory protection controller (region x/sub-regions A and B)



In the figure above, region x represents the external memory or backup SRAM region (such as FSMC bank, OCTOSPI1, OCTOSPI2 or BKPSRAM). Secure and privileged attributes of sub-regions A and B are independently configurable. When no sub-regions are defined or enabled on the region x, then the default attribute of the region x is set as “secure-privileged”.

The tables below describe the secure/privileged properties of the common area of sub-region A and B when an overlap exists.

Table 30. Secure properties of sub-regions A and B

Sub-region A	Sub-region B	Properties of overlapped region A and B
Non-secure	Non-secure	Non-secure
Non-secure	Secure	Non-secure
Secure	Non-secure	Non-secure
Secure	Secure	Secure

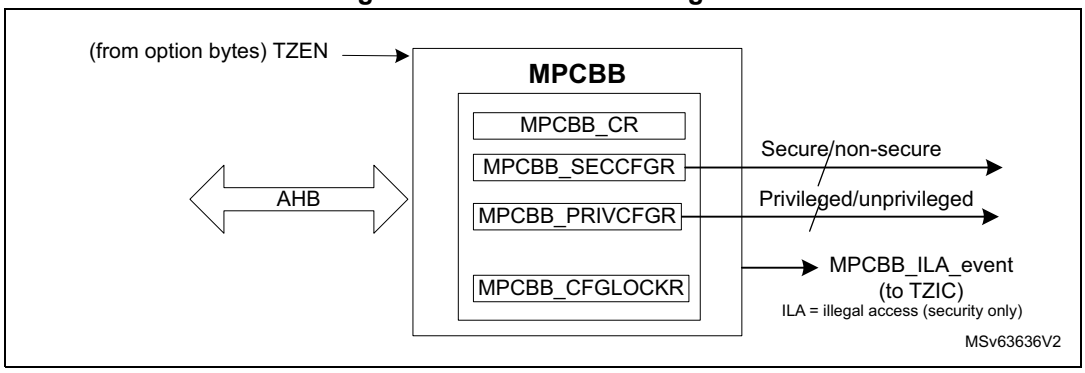
Table 31. Privileged properties of sub-regions A and B

Sub-region A	Sub-region B	Properties of overlapped region A and B
Unprivileged	Unprivileged	Unprivileged
Unprivileged	Privileged	Unprivileged
Privileged	Unprivileged	Unprivileged
Privileged	Privileged	Privileged

5.4.4 Memory protection controller - block based (MPCBB)

The MPCBB is composed of a configurable set of registers allowing to define security and privileged policy for internal SRAM memories. The security and privileged policy can be individually configured per each 512-byte block of SRAM.

Figure 18. MPCBB block diagram



In order to setup the MPCBB, the following actions are needed (for example at boot time):

- Secure firmware must define which memory blocks are secure by setting the correct bits in GTZC_MPCBBz_SECCFGRx.
- Privileged firmware must define which memory blocks are privileged by setting the correct bits in GTZC_MPCBBz_PRIVCFGRx.

A MPCBB super-block is made of 32 consecutive blocks. For each super-block, secure application can lock all related security/privileged bits using the correct bits in GTZC_MPCBBz_CFGLOCKR1. This lock remains active until the next system reset.

*Note: The block size is 512 bytes.The super-block size is 512 * 32 = 16 Kbytes.*

5.4.5 TrustZone illegal access controller (TZIC)

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

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

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

5.4.6 Power-on/reset state

The power-on and reset state of the TZSC clear to 0 all bits of GTZC_TZSC_SECCFGRx and GTZC_TZSC_PRIVCFGRx, meaning that all securable peripherals are respectively set to non-secure and unprivileged.

For internal SRAMx (x = 1 to 4), all GTZC_MPCBBz_SECCFGRx and GTZC_MPCBBz_PRIVCFGRx are set:

- to 0xFFFF FFFF, making these internal memories block secure and privileged by default when TrustZone security is enabled at system level (TZEN = 1).
- to 0x0000 0000, making these internal memories block non-secure and unprivileged by default when TrustZone security is disabled at system level (TZEN = 0)

For external memories and backup SRAM, all GTZC_TZSC_MPCWMxzR registers are set:

- to 0x0000 0000, making these memories secure and privileged by default when TrustZone security is enabled a system level (TZEN = 1).
- to 0x0800 0000, making these memories non-secure and non-privilege by default when TrustZone security is disabled at system level (TZEN = 0)

Secure boot code can then program the security settings, making components secure or not as needed.

5.5 GTZC interrupts

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

Table 32. GTZC interrupt request

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop mode	Exit Standby mode
GTZC	Illegal access	All flags in GTZC_TZIC_SRx	All bits in GTZC_TZIC_IERx	Write 1 in the bit GTZC_TZIC_FCRx	Yes	Yes	No

5.6 GTZC1 TZSC registers

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

5.6.1 GTZC1 TZSC control register (GTZC1_TZSC_CR)

Address offset: 0x000

Reset value: 0x0000 0000

Secure privileged 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCK
															rs

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

Bit 0 **LCK**: lock the configuration of GTZC1_TZSC_SECCFGRx and GTZC1_TZSC_PRIVCFGRx registers until next reset

This bit is cleared by default and once set, it can not be reset until system reset.

0: configuration of all GTZC1_TZSC_SECCFGRx and GTZC1_TZSC_PRIVCFGRx registers not locked

1: configuration of all GTZC1_TZSC_SECCFGRx and GTZC1_TZSC_PRIVCFGRx registers locked

5.6.2 GTZC1 TZSC secure configuration register 1 (GTZC1_TZSC_SECCFGR1)

Address offset: 0x010

Reset value: 0x0000 0000

Write-secure access only.

This register can be written only by secure privileged transaction when corresponding GTZC1_TZSC_PRIVCFGR register signal is set to 1. If a given PRIV bit is not set, the equivalent SEC bit can be written by secure unprivileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged 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.	UCPD1SEC	FDCAN1SEC	LPTIM2SEC	I2C4SEC
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRSSEC	I2C2SEC	I2C1SEC	UART5SEC	UART4SEC	USART3SEC	USART2SEC	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

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

Bit 19 **UCPD1SEC**: secure access mode for UCPD1

0: non-secure

1: secure

Bit 18 **FDCAN1SEC**: secure access mode for FDCAN1

0: non-secure

1: secure

Bit 17 **LPTIM2SEC**: secure access mode for LPTIM2

0: non-secure

1: secure

Bit 16 **I2C4SEC**: secure access mode for I2C4

0: non-secure

1: secure

Bit 15 **CRSSEC**: secure access mode for CRS
0: non-secure
1: secure

Bit 14 **I2C2SEC**: secure access mode for I2C2
0: non-secure
1: secure

Bit 13 **I2C1SEC**: secure access mode for I2C1
0: non-secure
1: secure

Bit 12 **UART5SEC**: secure access mode for UART5
0: non-secure
1: secure

Bit 11 **UART4SEC**: secure access mode for UART4
0: non-secure
1: secure

Bit 10 **USART3SEC**: secure access mode for USART3
0: non-secure
1: secure

Bit 9 **USART2SEC**: secure access mode for USART2
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

5.6.3 GTZC1 TZSC secure configuration register 2 (GTZC1_TZSC_SECCFGR2)

Address offset: 0x014

Reset value: 0x0000 0000

Write-secure access only.

This register can be written only by secure privileged transaction when corresponding GTZC1_TZSC_PRIVCFGR register signal is set to 1. If a given PRIV is not set, the equivalent SEC bit can be written by secure unprivileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged 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.	SAI2SEC	SAI1SEC	TIM17SEC	TIM16SEC	TIM15SEC	USART1SEC	TIM8SEC	SPI1SEC	TIM1SEC
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SAI2SEC**: secure access mode for SAI2
0: non-secure
1: secure

Bit 7 **SAI1SEC**: secure access mode for SAI1
0: non-secure
1: secure

Bit 6 **TIM17SEC**: secure access mode for TIM7
0: non-secure
1: secure

Bit 5 **TIM16SEC**: secure access mode for TIM6
0: non-secure
1: secure

Bit 4 **TIM15SEC**: secure access mode for TIM5
0: non-secure
1: secure

Bit 3 **USART1SEC**: secure access mode for USART1
0: non-secure
1: secure

Bit 2 **TIM8SEC**: secure access mode for TIM8
0: non-secure
1: secure

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

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

5.6.4 GTZC1 TZSC secure configuration register 3 (GTZC1_TZSC_SECCFGR3)

Address offset: 0x018

Reset value: 0x0000 0000

Write-secure access only.

This register can be written only by secure privileged transaction when corresponding GTZC1_TZSC_PRIVCFGR register signal is set to 1. If a given PRIV is not set, the equivalent SEC bit can be written by secure unprivileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged 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.	RAMCFGSEC	OCTOSPI2_REGSEC	OCTOSPI1_REGSEC	FSMC_REGSEC	SDMMC2SEC	SDMMC1SEC	OCTOSPIMSEC
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAESSEC	PKASEC	RNGSEC	HASHSEC	AESSEC	OTGSEC	DCMISEC	ADC12SEC	DCACHE1_REGSEC	ICACHE_REGSEC	DMA2DSEC	TSCSEC	CRCSEC	FMACSEC	CORDICSEC	MDF1SEC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 22 **RAMCFGSEC**: secure access mode for RAMCFG
0: non-secure
1: secure

Bit 21 **OCTOSPI2_REGSEC**: secure access mode for OCTOSPI2 registers
0: non-secure
1: secure

Bit 20 **OCTOSPI1_REGSEC**: secure access mode for OCTOSPI1 registers
0: non-secure
1: secure

Bit 19 **FSMC_REGSEC**: secure access mode for FSMC registers
0: non-secure
1: secure

Bit 18 **SDMMC2SEC**: secure access mode for SDMMC1
0: non-secure
1: secure

Bit 17 **SDMMC1SEC**: secure access mode for SDMMC2
0: non-secure
1: secure

Bit 16 **OCTOSPIMSEC**: secure access mode for OCTOSPIM
0: non-secure
1: secure

Bit 15 **SAESSEC**: secure access mode for SAES
0: non-secure
1: secure

Bit 14 **PKASEC**: secure access mode for PKA
0: non-secure
1: secure

Bit 13 **RNGSEC**: secure access mode for RNG
0: non-secure
1: secure

Bit 12 **HASHSEC**: secure access mode for HASH
0: non-secure
1: secure

Bit 11 **AESSEC**: secure access mode for AES
0: non-secure
1: secure

Bit 10 **OTGSEC**: secure access mode for OTG_FS
0: non-secure
1: secure

Bit 9 **DCMISEC**: secure access mode for DCMI and PSSI
0: non-secure
1: secure

Bit 8 **ADC12SEC**: secure access mode for ADC1
0: non-secure
1: secure

Bit 7 **DCACHE1_REGSEC**: secure access mode for DCACHE1 registers
0: non-secure
1: secure

Bit 6 **ICACHE_REGSEC**: secure access mode for ICACHE registers
0: non-secure
1: secure

Bit 5 **DMA2DSEC**: secure access mode for register of DMA2D
0: non-secure
1: secure

Bit 4 **TSCSEC**: secure access mode for TSC
0: non-secure
1: secure

Bit 3 **CRCSEC**: secure access mode for CRC
0: non-secure
1: secure

Bit 2 **FMACSEC**: secure access mode for FMAC
0: non-secure
1: secure

Bit 1 **CORDICSEC**: secure access mode for CORDIC
0: non-secure
1: secure

Bit 0 **MDF1SEC**: secure access mode for MDF1
0: non-secure
1: secure

5.6.5 GTZC1 TZSC privilege configuration register 1 (GTZC1_TZSC_PRIVCFGR1)

Address offset: 0x020

Reset value: 0x0000 0000

Write-privileged access only.

This register can be read or written only by secure privileged transaction when corresponding GTZC1_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, privileged 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.	UCPD1PRIV	FDCAN1PRIV	LPTIM2PRIV	I2C4PRIV
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRSPRIV	I2C2PRIV	I2C1PRIV	UART5PRIV	UART4PRIV	USART3PRIV	USART2PRIV	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

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

Bit 19 **UCPD1PRIV**: privileged access mode for UCPD1
0: unprivileged
1: privileged

Bit 18 **FDCAN1PRIV**: privileged access mode for FDCAN1
0: unprivileged
1: privileged

Bit 17 **LPTIM2PRIV**: privileged access mode for LPTIM2
0: unprivileged
1: privileged

Bit 16 **I2C4PRIV**: privileged access mode for I2C4
0: unprivileged
1: privileged

Bit 15 **CRSPRIV**: privileged access mode for CRS
0: unprivileged
1: privileged

Bit 14 **I2C2PRIV**: privileged access mode for I2C2
0: unprivileged
1: privileged

Bit 13 **I2C1PRIV**: privileged access mode for I2C1
0: unprivileged
1: privileged

Bit 12 **UART5PRIV**: privileged access mode for UART5
0: unprivileged
1: privileged

Bit 11 **UART4PRIV**: privileged access mode for UART4
0: unprivileged
1: privileged

Bit 10 **USART3PRIV**: privileged access mode for USART3
0: unprivileged
1: privileged

Bit 9 **USART2PRIV**: privileged access mode for USART2
0: unprivileged
1: privileged

Bit 8 **SPI2PRIV**: privileged access mode for SPI2
0: unprivileged
1: privileged

Bit 7 **IWDGPRIV**: privileged access mode for IWDG
0: unprivileged
1: privileged

Bit 6 **WWDGPRIV**: privileged access mode for WWDG
0: unprivileged
1: privileged

Bit 5 **TIM7PRIV**: privileged access mode for TIM7
0: unprivileged
1: privileged

Bit 4 **TIM6PRIV**: privileged access mode for TIM6
0: unprivileged
1: privileged

Bit 3 **TIM5PRIV**: privileged access mode for TIM5
0: unprivileged
1: privileged

Bit 2 **TIM4PRIV**: privileged access mode for TIM4
0: unprivileged
1: privileged

Bit 1 **TIM3PRIV**: privileged access mode for TIM3
0: unprivileged
1: privileged

Bit 0 **TIM2PRIV**: privileged access mode for TIM2
0: unprivileged
1: privileged

5.6.6 GTZC1 TZSC privilege configuration register 2 (GTZC1_TZSC_PRIVCFGR2)

Address offset: 0x024

Reset value: 0x0000 0000

Write-privileged access only.

This register can be read or written only by secure privileged transaction when corresponding GTZC1_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, privileged 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.	SAI2PRIV	SAI1PRIV	TIM17PRIV	TIM16PRIV	TIM15PRIV	USART1PRIV	TIM8PRIV	SPI1PRIV	TIM1PRIV
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SAI2PRIV**: privileged access mode for SAI2
0: unprivileged
1: privileged

Bit 7 **SAI1PRIV**: privileged access mode for SAI1
0: unprivileged
1: privileged

Bit 6 **TIM17PRIV**: privileged access mode for TIM17
0: unprivileged
1: privileged

Bit 5 **TIM16PRIV**: privileged access mode for TIM16
0: unprivileged
1: privileged

Bit 4 **TIM15PRIV**: privileged access mode for TIM15
0: unprivileged
1: privileged

Bit 3 **USART1PRIV**: privileged access mode for USART1
0: unprivileged
1: privileged

Bit 2 **TIM8PRIV**: privileged access mode for TIM8
0: unprivileged
1: privileged

Bit 1 **SPI1PRIV**: privileged access mode for SPI1PRIV
0: unprivileged
1: privileged

Bit 0 **TIM1PRIV**: privileged access mode for TIM1
0: unprivileged
1: privileged

5.6.7 GTZC1 TZSC privilege configuration register 3 (GTZC1_TZSC_PRIVCFGR3)

Address offset: 0x028

Reset value: 0x0000 0000

Write-privileged access only.

This register can be read or written only by secure privileged transaction when corresponding GTZC1_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, privileged 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.	RAMCFGPRIV	OCTOSPI2_REGPRIV	OCTOSPI1_REGPRIV	FSMC_REGPRIV	SDMMC2PRIV	SDMMC1PRIV	OCTOSPIMPRIV
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAESPRIV	PKAPRIV	RNGPRIV	HASHPRIV	AESPRIV	OTGPRIV	DCMIPRIV	ADC12PRIV	DCACHE1_REGPRIV	ICACHE_REGPRIV	DMA2DPRIV	TSCPRIV	CRCPRIV	FMACPRIV	CORDICPRIV	MDF1PRIV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 22 **RAMCFGPRIV**: privileged access mode for RAMCFG
- 0: unprivileged
- 1: privileged

Bit 21 **OCTOSPI2_REGPRIV**: privileged access mode for OCTOSPI2
- 0: unprivileged
- 1: privileged

Bit 20 **OCTOSPI1_REGPRIV**: privileged access mode for OCTOSPI1
- 0: unprivileged
- 1: privileged

Bit 19 **FSMC_REGPRIV**: privileged access mode for FSMC registers
- 0: unprivileged
- 1: privileged

Bit 18 **SDMMC2PRIV**: privileged access mode for SDMMC1
- 0: unprivileged
- 1: privileged

Bit 17 **SDMMC1PRIV**: privileged access mode for SDMMC2
0: unprivileged
1: privileged

Bit 16 **OCTOSPIMPRIV**: privileged access mode for OCTOSPIM
0: unprivileged
1: privileged

Bit 15 **SAESPRIV**: privileged access mode for SAES
0: unprivileged
1: privileged

Bit 14 **PKAPRIV**: privileged access mode for PKA
0: unprivileged
1: privileged

Bit 13 **RNGPRIV**: privileged access mode for RNG
0: unprivileged
1: privileged

Bit 12 **HASHPRIV**: privileged access mode for HASH
0: unprivileged
1: privileged

Bit 11 **AESPRIV**: privileged access mode for AES
0: unprivileged
1: privileged

Bit 10 **OTGPRIV**: privileged access mode for OTG_FS
0: unprivileged
1: privileged

Bit 9 **DCMIPRIV**: privileged access mode for DCMI and PSSI
0: unprivileged
1: privileged

Bit 8 **ADC12PRIV**: privileged access mode for ADC1
0: unprivileged
1: privileged

Bit 7 **DCACHE1_REGPRIV**: privileged access mode for DCACHE1 registers
0: unprivileged
1: privileged

Bit 6 **ICACHE_REGPRIV**: privileged access mode for ICACHE registers
0: unprivileged
1: privileged

Bit 5 **DMA2DPRIV**: privileged access mode for register of DMA2D
0: unprivileged
1: privileged

Bit 4 **TSCPRIV**: privileged access mode for TSC
0: unprivileged
1: privileged

Bit 3 **CRCPRIV**: privileged access mode for CRC
0: unprivileged
1: privileged

Bit 2 **FMACPRIV**: privileged access mode for FMAC
0: unprivileged
1: privileged

Bit 1 **CORDICPRIV**: privileged access mode for CORDIC
0: unprivileged
1: privileged

Bit 0 **MDF1PRIV**: privileged access mode for MDF1
0: unprivileged
1: privileged

5.6.8 GTZC1 TZSC memory x sub-region z watermark configuration register (GTZC1_TZSC_MPCWMxzCFGR) (z = A to B)

Address offset: Block A: 0x40 + 0x10 *(x - 1) (x = 1 to 5)
Address offset: Block B: 0x48 + 0x10 *(x - 1) (x = 1, 2, 5)
Reset value: 0x0000 0000

Secure privilege 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.	PRIV	SEC	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
						rw	rw							rs	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **PRIV**: Privileged sub-region z of base region x
This bit is taken into account only if SREN is set.
0: Privileged and unprivileged accesses are granted in sub-region z.
1: Only privileged accesses are granted in sub-region z of region x.

Bit 8 **SEC**: Secure sub-region z of base region x
This bit is taken into account only if SREN is set.
0: Only non-secure data accesses are granted to sub-region z of region x.
1: Only secure data accesses are granted to sub-region z of region x.

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

Bit 1 **SRLOCK**: Sub-region z lock

This bit, once set, can be cleared only by a system reset.
0: GTZC1_TZSC_MPCWMxCFGR, GTZC1_TZSC_MPCWMxAR and GTZC1_TZSC_MPCWMxBR can be written.
1: Writes to GTZC1_TZSC_MPCWMxCFGR, GTZC1_TZSC_MPCWMxAR and GTZC1_TZSC_MPCWMxBR are ignored.

Bit 0 **SREN**: Sub-region z enable

0: Sub-region z is disabled. Access control of base region x applies to any access between this sub-region start- and end-addresses.
1: Sub-region z of region x is enabled. Access control defined in GTZC1_TZSC_MPCWMx_CFGR applies to any access between this sub-region start- and end-addresses, both defined in GTZC1_TZSC_MPCWMxAR and GTZC1_TZSC_MPCWMxBR .

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

5.6.9 GTZC1 TZSC memory x sub-region A watermark register (GTZC1_TZSC_MPCWMxAR)

Address offset: 0x44 + 0x10 *(x - 1) (x = 1 to 5)
Reset value: 0x0000 0000
The given reset value is valid when TZEN = 1. The reset value is 0x0800 0000 when TZEN = 0.
Secure privilege access only.

When SUBA_START + SUBA_LENGTH is higher than the maximum size allowed for the memory, a saturation of SUBA_LENGTH is applied automatically.
When an overlap of sub-region A and B exists, secure/privileged attributes of both sub-regions apply on the common section (see *Section 5.4.3: TrustZone security controller (TZSC)*)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	SUBA_LENGTH[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.	SUBA_START[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 **SUBA_LENGTH[11:0]**: Length of sub-region A in region x

This field defines the length of the sub-region A, to be multiplied by the granularity defined in *Table 28*.

When SUBA_START + SUBA_LENGTH is higher than the maximum size allowed for the memory, a saturation of SUBA_LENGTH is applied automatically.

If SUBA_LENGTH = 0, the sub-region A is disabled.(SREN bit in GTZC1_TZSC_MPCMWxACFGR is cleared).

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

Bits 10:0 **SUBA_START[10:0]**: Start of sub-region A in region x

This field defines the address offset of the sub-region A, to be multiplied by the granularity defined in *Table 28*, versus the start of the region x.

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

5.6.10 GTZC1 TZSC memory x sub-region B watermark register (GTZC1_TZSC_MPCWMxBR)

Address offset: 0x4C + 0x10 *(x - 1) (x = 1, 2, 5)
Reset value: 0x0000 0000
The given reset value is valid when TZEN = 1. The reset value is 0x0800 0000 when TZEN = 0.
Secure privilege access only.
When SUBB_START + SUBB_LENGTH is higher than the maximum size allowed for the memory, a saturation of SUBB_LENGTH is applied automatically.
When an overlap of sub-region A and B exists, secure/privileged attributes of both sub-regions apply on the common section (see *Section 5.4.3: TrustZone security controller (TZSC)*)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	SUBB_LENGTH[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.	SUBB_START[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 **SUBB_LENGTH[11:0]**: Length of sub-region B in region x
This field defines the length of the sub-region B, to be multiplied by the granularity defined in *Table 28*.
When SUBB_START + SUBB_LENGTH is higher than the maximum size allowed for the memory, a saturation of SUBB_LENGTH is applied automatically.
If SUBB_LENGTH = 0, the sub-region B is disabled.(SREN bit in GTZC1_TZSC_MPCMWxBCFGR is cleared).

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

Bits 10:0 **SUBB_START[10:0]**: Start of sub-region B in region x
This field defines the address offset of the sub-region B, to be multiplied by the granularity defined in *Table 28*, versus the start of the region x.
External memories that are watermark controlled, start fully non-secure at reset when TZEN = 0. When TZEN = 1, external memories start fully secure (inverted reset-value).

5.6.11 GTZC1 TZSC register map

Table 33. GTZC1 TZSC 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	GTZC1_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
0x004-0x00C	Reserved	Reserved																															

Table 33. GTZC1 TZSC 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
0x010	GTZC1_TZSC_ SECCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1SEC	FDCAN1SEC	LPTIM2SEC	I2C4SEC	CRSSEC	I2C2SEC	I2C1SEC	UART5SEC	UART4SEC	USART3SEC	USART2SEC	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
0x014	GTZC1_TZSC_ SECCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2SEC	SAI1SEC	TIM17SEC	TIM16SEC	TIM15SEC	USART1SEC	TIM8SEC	SPI1SEC	TIM1SEC
	Reset value																								0	0	0	0	0	0	0	0	0
0x018	GTZC1_TZSC_ SECCFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGSEC	OCTOSPI2_REGSEC	OCTOSPI1_REGSEC	FSMC_REGSEC	SDMMC2SEC	SDMMC1SEC	OCTOSPIMSEC	SAESSEC	PKASEC	RNGSEC	HASHSEC	AESSEC	OTGSEC	DCMISEC	ADC12SEC	DCACHE1_REGSEC	ICACHE_REGSEC	DMA2DSEC	TSCSEC	CRCSEC	FMACSEC	CORDICSEC	MDF1SEC
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	Reserved	Reserved																															
0x020	GTZC1_TZSC_ PRIVCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1PRIV	FDCAN1PRIV	LPTIM2PRIV	I2C4PRIV	CRSPRIV	I2C2PRIV	I2C1PRIV	UART5PRIV	UART4PRIV	USART3PRIV	USART2PRIV	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
0x024	GTZC1_TZSC_ PRIVCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2PRIV	SAI1PRIV	TIM17PRIV	TIM16PRIV	TIM15PRIV	USART1PRIV	TIM8PRIV	SPI1PRIV	TIM1PRIV
	Reset value																								0	0	0	0	0	0	0	0	0
0x028	GTZC1_TZSC_ PRIVFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGPRIV	OCTOSPI2_REGPRIV	OCTOSPI1_REGPRIV	FSMC_REGPRIV	SDMMC2PRIV	SDMMC1PRIV	OCTOSPIMPRIV	SAESPRIV	PKAPRIV	RNGPRIV	HASHPRIV	AESPRIV	OTGPRIV	DCMIPRIV	ADC12PRIV	DCACHE1_REGPRIV	ICACHE_REGPRIV	DMA2DPRIV	TSCPRIV	CRCPRIV	FMACPRIV	CORDICPRIV	MDF1PRIV
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C-0x03C	Reserved	Reserved																															
0x040	GTZC1_TZSC_ MPCWM1ACFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
	Reset value																							0	0							0	0
0x44	GTZC1_TZSC_ MPCWM1AR	Res.	Res.	Res.	Res.	SUBA_LENGTH[11:0]												Res.	Res.	Res.	Res.	Res.	SUBA_START[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
0x048	GTZC1_TZSC_ MPCWM1BCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
	Reset value																							0	0							0	0
0x04C	GTZC1_TZSC_ MPCWM1BR	Res.	Res.	Res.	Res.	SUBB_LENGTH[11:0]												Res.	Res.	Res.	Res.	Res.	SUBB_START[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

Table 33. GTZC1 TZSC 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
0x050	GTZC1_TZSC_ MPCWM2ACFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
	Reset value																							0	0							0	0
0x54	GTZC1_TZSC_ MPCWM2AR	Res.	Res.	Res.	Res.	SUBA_LENGTH[11:0]												Res.	Res.	Res.	Res.	Res.	SUBA_START[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
0x058	GTZC1_TZSC_ MPCWM2BCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
	Reset value																							0	0							0	0
0x05C	GTZC1_TZSC_ MPCWM2BR	Res.	Res.	Res.	Res.	SUBB_LENGTH[11:0]												Res.	Res.	Res.	Res.	Res.	SUBB_START[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
0x060	GTZC1_TZSC_ MPCWM3ACFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
	Reset value																							0	0							0	0
0x64	GTZC1_TZSC_ MPCWM3AR	Res.	Res.	Res.	Res.	SUBA_LENGTH[11:0]												Res.	Res.	Res.	Res.	Res.	SUBA_START[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
0x068-0x06C	Reserved	Reserved																															
0x070	GTZC1_TZSC_ MPCWM4ACFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
	Reset value																							0	0							0	0
0x74	GTZC1_TZSC_ MPCWM4AR	Res.	Res.	Res.	Res.	SUBA_LENGTH[11:0]												Res.	Res.	Res.	Res.	Res.	SUBA_START[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
0x078-0x07C	Reserved	Reserved																															
0x080	GTZC1_TZSC_ MPCWM5ACFGR	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
	Reset value																							0	0							0	0
0x84	GTZC1_TZSC_ MPCWM5AR	Res.	Res.	Res.	Res.	SUBA_LENGTH[11:0]												Res.	Res.	Res.	Res.	Res.	SUBA_START[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
0x088	GTZC1_TZSC_ MPCWM5BCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
	Reset value																							0	0							0	0
0x08C	GTZC1_TZSC_ MPCWM5BR	Res.	Res.	Res.	Res.	SUBB_LENGTH[11:0]												Res.	Res.	Res.	Res.	Res.	SUBB_START[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 *Table 26: GTZC1 sub-blocks address offset*.

5.7 GTZC1 TZIC registers

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

5.7.1 GTZC1 TZIC interrupt enable register 1 (GTZC1_TZIC_IER1)

Address offset: 0x000

Reset value: 0x0000 0000

Secure privileged 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.	UCPD1IE	FDCAN1IE	LPTIM2IE	I2C4IE
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRSIE	I2C2IE	I2C1IE	UART5IE	UART4IE	USART3IE	USART2IE	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

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

Bit 19 **UCPD1IE**: illegal access interrupt enable for UCPD1
0: interrupt disabled
1: interrupt enabled

Bit 18 **FDCAN1IE**: illegal access interrupt enable for FDCAN1
0: interrupt disabled
1: interrupt enabled

Bit 17 **LPTIM2IE**: illegal access interrupt enable for LPTIM2
0: interrupt disabled
1: interrupt enabled

Bit 16 **I2C4IE**: illegal access interrupt enable for I2C4
0: interrupt disabled
1: interrupt enabled

Bit 15 **CRSIE**: illegal access interrupt enable for CRS
0: interrupt disabled
1: interrupt enabled

Bit 14 **I2C2IE**: illegal access interrupt enable for I2C2
0: interrupt disabled
1: interrupt enabled

Bit 13 **I2C1IE**: illegal access interrupt enable for I2C1
0: interrupt disabled
1: interrupt enabled

Bit 12 **UART5IE**: illegal access interrupt enable for UART5
0: interrupt disabled
1: interrupt enabled

Bit 11 **UART4IE**: illegal access interrupt enable for UART4
0: interrupt disabled
1: interrupt enabled

Bit 10 **USART3IE**: illegal access interrupt enable for USART3
0: interrupt disabled
1: interrupt enabled

Bit 9 **USART2IE**: illegal access interrupt enable for USART2
0: interrupt disabled
1: interrupt enabled

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

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

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

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

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

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

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

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

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

5.7.2 GTZC1 TZIC interrupt enable register 2 (GTZC1_TZIC_IER2)

Address offset: 0x004

Reset value: 0x0000 0000

Secure privileged 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.	SAI2IE	SAI1IE	TIM17IE	TIM16IE	TIM15IE	USART1IE	TIM8IE	SPI1IE	TIM1IE
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SAI2IE**: illegal access interrupt enable for SAI2
0: interrupt disabled
1: interrupt enabled

Bit 7 **SAI1IE**: illegal access interrupt enable for SAI1
0: interrupt disabled
1: interrupt enabled

Bit 6 **TIM17IE**: illegal access interrupt enable for TIM7
0: interrupt disabled
1: interrupt enabled

Bit 5 **TIM16IE**: illegal access interrupt enable for TIM6
0: interrupt disabled
1: interrupt enabled

Bit 4 **TIM15IE**: illegal access interrupt enable for TIM5
0: interrupt disabled
1: interrupt enabled

Bit 3 **USART1IE**: illegal access interrupt enable for USART1
0: interrupt disabled
1: interrupt enabled

Bit 2 **TIM8IE**: illegal access interrupt enable for TIM8
0: interrupt disabled
1: interrupt enabled

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

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

5.7.3 GTZC1 TZIC interrupt enable register 3 (GTZC1_TZIC_IER3)

Address offset: 0x008

Reset value: 0x0000 0000

Secure privileged 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.	RAMCFGIE	OCTOSPI2_REGIE	OCTOSPI1_REGIE	FSMC_REGIE	SDMMC2IE	SDMMC1IE	OCTOSPIMIE
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAESIE	PKAIE	RNGIE	HASHIE	AESIE	OTGIE	DCMIIE	ADC12IE	DCACHE1_REGIE	ICACHE_REGIE	DMA2DIE	TSCIE	CRCIE	FMACIE	CORDICIE	MDF1IE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 22 **RAMCFGIE**: illegal access interrupt enable for RAMCFG
- 0: interrupt disabled
- 1: interrupt enabled

Bit 21 **OCTOSPI2_REGIE**: illegal access interrupt enable for OCTOSPI2 registers
- 0: interrupt disabled
- 1: interrupt enabled

Bit 20 **OCTOSPI1_REGIE**: illegal access interrupt enable for OCTOSPI1 registers
- 0: interrupt disabled
- 1: interrupt enabled

Bit 19 **FSMC_REGIE**: illegal access interrupt enable for FSMC registers
- 0: interrupt disabled
- 1: interrupt enabled

Bit 18 **SDMMC2IE**: illegal access interrupt enable for SDMMC1
- 0: interrupt disabled
- 1: interrupt enabled

Bit 17 **SDMMC1IE**: illegal access interrupt enable for SDMMC2
- 0: interrupt disabled
- 1: interrupt enabled

Bit 16 **OCTOSPIMIE**: illegal access interrupt enable for OCTOSPIM
- 0: interrupt disabled
- 1: interrupt enabled

Bit 15 **SAESIE**: illegal access interrupt enable for SAES
0: interrupt disabled
1: interrupt enabled

Bit 14 **PKAIE**: illegal access interrupt enable for PKA
0: interrupt disabled
1: interrupt enabled

Bit 13 **RNGIE**: illegal access interrupt enable for RNG
0: interrupt disabled
1: interrupt enabled

Bit 12 **HASHIE**: illegal access interrupt enable for HASH
0: interrupt disabled
1: interrupt enabled

Bit 11 **AESIE**: illegal access interrupt enable for AES
0: interrupt disabled
1: interrupt enabled

Bit 10 **OTGIE**: illegal access interrupt enable for OTG_FS
0: interrupt disabled
1: interrupt enabled

Bit 9 **DCMIIE**: illegal access interrupt enable for DCMI and PSSI
0: interrupt disabled
1: interrupt enabled

Bit 8 **ADC12IE**: illegal access interrupt enable for ADC1
0: interrupt disabled
1: interrupt enabled

Bit 7 **DCACHE1_REGIE**: illegal access interrupt enable for DCACHE1 registers
0: interrupt disabled
1: interrupt enabled

Bit 6 **ICACHE_REGIE**: illegal access interrupt enable for ICACHE registers
0: interrupt disabled
1: interrupt enabled

Bit 5 **DMA2DIE**: illegal access interrupt enable for register of DMA2D
0: interrupt disabled
1: interrupt enabled

Bit 4 **TSCIE**: illegal access interrupt enable for TSC
0: interrupt disabled
1: interrupt enabled

Bit 3 **CRCIE**: illegal access interrupt enable for CRC
0: interrupt disabled
1: interrupt enabled

Bit 2 **FMACIE**: illegal access interrupt enable for FMAC
0: interrupt disabled
1: interrupt enabled

Bit 1 **CORDICIE**: illegal access interrupt enable for CORDIC
0: interrupt disabled
1: interrupt enabled

Bit 0 **MDF1IE**: illegal access interrupt enable for MDF1
0: interrupt disabled
1: interrupt enabled

5.7.4 GTZC1 TZIC interrupt enable register 4 (GTZC1_TZIC_IER4)

Address offset: 0x00C

Reset value: 0x0000 0000

Secure privileged 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.	MPCBB3_REGIE	SRAM3IE	MPCBB2_REGIE	SRAM2IE	MPCBB1_REGIE	SRAM1IE	Res.	Res.	Res.	Res.	OCTOSPI2_MEMIE	BKPSRAMIE	FSMC_MEMIE	OCTOSPI1_MEMIE
		rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TZIC1IE	TZSC1IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTFDEC2IE	OTFDEC1IE	FLASHIE	FLASH_REGIE	GPDMA1IE
rw	rw										rw	rw	rw	rw	rw

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

Bit 29 **MPCBB3_REGIE**: illegal access interrupt enable for MPCBB3 registers
0: interrupt disabled
1: interrupt enabled

Bit 28 **SRAM3IE**: illegal access interrupt enable for SRAM3
0: interrupt disabled
1: interrupt enabled

Bit 27 **MPCBB2_REGIE**: illegal access interrupt enable for MPCBB2 registers
0: interrupt disabled
1: interrupt enabled

Bit 26 **SRAM2IE**: illegal access interrupt enable for SRAM2
0: interrupt disabled
1: interrupt enabled

Bit 25 **MPCBB1_REGIE**: illegal access interrupt enable for MPCBB1 registers
0: interrupt disabled
1: interrupt enabled

Bit 24 **SRAM1IE**: illegal access interrupt enable for SRAM1
0: interrupt disabled
1: interrupt enabled

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **OCTOSPI2_MEMIE**: illegal access interrupt enable for OCTOSPI2 memory bank
0: interrupt disabled
1: interrupt enabled

Bit 18 **BKPSRAMIE**: illegal access interrupt enable for MPCWM3 (BKPSRAM) memory bank
0: interrupt disabled
1: interrupt enabled

Bit 17 **FSMC_MEMIE**: illegal access interrupt enable for MPCWM2 (FSMC NAND) and MPCWM3 (FSMC NOR)
0: interrupt disabled
1: interrupt enabled

Bit 16 **OCTOSPI1_MEMIE**: illegal access interrupt enable for MPCWM1 (OCTOSPI1) memory bank
0: interrupt disabled
1: interrupt enabled

Bit 15 **TZIC1IE**: illegal access interrupt enable for GTZC1 TZIC registers
0: interrupt disabled
1: interrupt enabled

Bit 14 **TZSC1IE**: illegal access interrupt enable for GTZC1 TZSC registers
0: interrupt disabled
1: interrupt enabled

Bits 13:5 Reserved, must be kept at reset value.

Bit 4 **OTFDEC2IE**: illegal access interrupt enable for OTFDEC2
0: interrupt disabled
1: interrupt enabled

Bit 3 **OTFDEC1IE**: illegal access interrupt enable for OTFDEC1
0: interrupt disabled
1: interrupt enabled

Bit 2 **FLASHIE**: illegal access interrupt enable for FLASH memory
0: interrupt disabled
1: interrupt enabled

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

Bit 0 **GPDMA1IE**: illegal access interrupt enable for GPDMA1
0: interrupt disabled
1: interrupt enabled

5.7.5 GTZC1 TZIC status register 1 (GTZC1_TZIC_SR1)

Address offset: 0x010

Reset value: 0x0000 0000

Secure privileged 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.	UCPD1F	FDCAN1F	LPTIM2F	I2C4F
												r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRSF	I2C2F	I2C1F	UART5F	UART4F	USART3F	USART2F	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

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

Bit 19 **UCPD1F**: illegal access flag for UCPD1
- 0: no illegal access event
- 1: illegal access event

Bit 18 **FDCAN1F**: illegal access flag for FDCAN1
- 0: no illegal access event
- 1: illegal access event

Bit 17 **LPTIM2F**: illegal access flag for LPTIM2
- 0: no illegal access event
- 1: illegal access event

Bit 16 **I2C4F**: illegal access flag for I2C4
- 0: no illegal access event
- 1: illegal access event

Bit 15 **CRSF**: illegal access flag for CRS
- 0: no illegal access event
- 1: illegal access event

Bit 14 **I2C2F**: illegal access flag for I2C2
- 0: no illegal access event
- 1: illegal access event

Bit 13 **I2C1F**: illegal access flag for I2C1
- 0: no illegal access event
- 1: illegal access event

Bit 12 **UART5F**: illegal access flag for UART5
- 0: no illegal access event
- 1: illegal access event

Bit 11 **UART4F**: illegal access flag for UART4
- 0: no illegal access event
- 1: illegal access event

Bit 10 **USART3F**: illegal access flag for USART3
0: no illegal access event
1: illegal access event

Bit 9 **USART2F**: illegal access flag for USART2
0: no illegal access event
1: illegal access event

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

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

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

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

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

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

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

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

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

5.7.6 GTZC1 TZIC status register 2 (GTZC1_TZIC_SR2)

Address offset: 0x014

Reset value: 0x0000 0000

Secure privileged 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.	SAI2F	SAI1F	TIM17F	TIM16F	TIM15F	USART1F	TIM8F	SPI1F	TIM1F
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SAI2F**: illegal access flag for SAI2
0: no illegal access event
1: illegal access event

Bit 7 **SAI1F**: illegal access flag for SAI1
0: no illegal access event
1: illegal access event

Bit 6 **TIM17F**: illegal access flag for TIM7
0: no illegal access event
1: illegal access event

Bit 5 **TIM16F**: illegal access flag for TIM6
0: no illegal access event
1: illegal access event

Bit 4 **TIM15F**: illegal access flag for TIM5
0: no illegal access event
1: illegal access event

Bit 3 **USART1F**: illegal access flag for USART1
0: no illegal access event
1: illegal access event

Bit 2 **TIM8F**: illegal access flag for TIM8
0: no illegal access event
1: illegal access event

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

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

5.7.7 GTZC1 TZIC status register 3 (GTZC1_TZIC_SR3)

Address offset: 0x018

Reset value: 0x0000 0000

Secure privileged 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.	RAMCFGF	OCTOSPI2_REGF	OCTOSPI1_REGF	FSMC_REGF	SDMMC2F	SDMMC1F	OCTOSPIMF
									r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAESF	PKAF	RNGF	HASHF	AESF	OTGF	DCMIF	ADC12F	DCACHE1_REGF	ICACHE_REGF	DMA2DF	TSCF	CRCF	FMACF	CORDICF	MDF1F
r	r	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 **RAMCFGF**: illegal access flag for RAMCFG
- 0: no illegal access event
- 1: illegal access event

Bit 21 **OCTOSPI2_REGF**: illegal access flag for OCTOSPI2 registers
- 0: no illegal access event
- 1: illegal access event

Bit 20 **OCTOSPI1_REGF**: illegal access flag for OCTOSPI1 registers
- 0: no illegal access event
- 1: illegal access event

Bit 19 **FSMC_REGF**: illegal access flag for FSMC registers
- 0: no illegal access event
- 1: illegal access event

Bit 18 **SDMMC2F**: illegal access flag for SDMMC1
- 0: no illegal access event
- 1: illegal access event

Bit 17 **SDMMC1F**: illegal access flag for SDMMC2
- 0: no illegal access event
- 1: illegal access event

Bit 16 **OCTOSPIMF**: illegal access flag for OCTOSPIM
- 0: no illegal access event
- 1: illegal access event

Bit 15 **SAESF**: illegal access flag for SAES
- 0: no illegal access event
- 1: illegal access event

Bit 14 **PKAF**: illegal access flag for PKA
0: no illegal access event
1: illegal access event

Bit 13 **RNGF**: illegal access flag for RNG
0: no illegal access event
1: illegal access event

Bit 12 **HASHF**: illegal access flag for HASH
0: no illegal access event
1: illegal access event

Bit 11 **AESF**: illegal access flag for AES
0: no illegal access event
1: illegal access event

Bit 10 **OTGF**: illegal access flag for OTG_FS
0: no illegal access event
1: illegal access event

Bit 9 **DCMIF**: illegal access flag for DCMI and PSSI
0: no illegal access event
1: illegal access event

Bit 8 **ADC12F**: illegal access flag for ADC1
0: no illegal access event
1: illegal access event

Bit 7 **DCACHE1_REGF**: illegal access flag for DCACHE1 registers
0: no illegal access event
1: illegal access event

Bit 6 **ICACHE_REGF**: illegal access flag for ICACHE registers
0: no illegal access event
1: illegal access event

Bit 5 **DMA2DF**: illegal access flag for register of DMA2D
0: no illegal access event
1: illegal access event

Bit 4 **TSCF**: illegal access flag for TSC
0: no illegal access event
1: illegal access event

Bit 3 **CRCF**: illegal access flag for CRC
0: no illegal access event
1: illegal access event

Bit 2 **FMACF**: illegal access flag for FMAC
0: no illegal access event
1: illegal access event

Bit 1 **CORDICF**: illegal access flag for CORDIC
0: no illegal access event
1: illegal access event

Bit 0 **MDF1F**: illegal access flag for MDF1
0: no illegal access event
1: illegal access event

5.7.8 GTZC1 TZIC status register 4 (GTZC1_TZIC_SR4)

Address offset: 0x01C

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	MPCBB3_REGF	SRAM3F	MPCBB2_REGF	SRAM2F	MPCBB1_REGF	SRAM1F	Res.	Res.	Res.	Res.	OCTOSPI2_MEMF	BKPSRAMF	FSMC_MEMF	OCTOSPI1_MEMF
		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
TZIC1F	TZSC1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTFDEC2F	OTFDEC1F	FLASHF	FLASH_REGF	GPDMA1F
r	r										r	r	r	r	r

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

Bit 29 **MPCBB3_REGF**: illegal access flag for MPCBB3 registers
- 0: no illegal access event
- 1: illegal access event

Bit 28 **SRAM3F**: illegal access flag for SRAM3
- 0: no illegal access event
- 1: illegal access event

Bit 27 **MPCBB2_REGF**: illegal access flag for MPCBB2 registers
- 0: no illegal access event
- 1: illegal access event

Bit 26 **SRAM2F**: illegal access flag for SRAM2
- 0: no illegal access event
- 1: illegal access event

Bit 25 **MPCBB1_REGF**: illegal access flag for MPCBB1 registers
- 0: no illegal access event
- 1: illegal access event

Bit 24 **SRAM1F**: illegal access flag for SRAM1
- 0: no illegal access event
- 1: illegal access event

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **OCTOSPI2_MEMF**: illegal access flag for OCTOSPI2 memory bank
- 0: no illegal access event
- 1: illegal access event

Bit 18 **BKPSRAMF**: illegal access flag for MPCWM3 (BKPSRAM) memory bank
- 0: no illegal access event
- 1: illegal access event

Bit 17 **FSMC_MEMF**: illegal access flag for MPCWM2 (FSMC NAND) and MPCWM3 (FSMC NOR)
0: no illegal access event
1: illegal access event

Bit 16 **OCTOSPI1_MEMF**: illegal access flag for MPCWM1 (OCTOSPI1) memory bank
0: no illegal access event
1: illegal access event

Bit 15 **TZIC1F**: illegal access flag for GTZC1 TZIC registers
0: no illegal access event
1: illegal access event

Bit 14 **TZSC1F**: illegal access flag for GTZC1 TZSC registers
0: no illegal access event
1: illegal access event

Bits 13:5 Reserved, must be kept at reset value.

Bit 4 **OTFDEC2F**: illegal access flag for OTFDEC2
0: no illegal access event
1: illegal access event

Bit 3 **OTFDEC1F**: illegal access flag for OTFDEC1
0: no illegal access event
1: illegal access event

Bit 2 **FLASHF**: illegal access flag for FLASH memory
0: no illegal access event
1: illegal access event

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

Bit 0 **GPDMA1F**: illegal access flag for GPDMA1
0: no illegal access event
1: illegal access event

5.7.9 GTZC1 TZIC flag clear register 1 (GTZC1_TZIC_FCR1)

Address offset: 0x020

Reset value: 0x0000 0000

Secure privileged 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.	CUCPD1F	CFDCAN1F	CLPTIM2F	CI2C4F
												w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCRSF	CI2C2F	CI2C1F	CUART5F	CUART4F	CUSART3F	CUSART2F	CSPI2F	CIWDGF	CWWDGF	CTIM7F	CTIM6F	CTIM5F	CTIM4F	CTIM3F	CTIM2F
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

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

Bit 19 **CUCPD1F**: clear the illegal access flag for UCPD1
0: no action
1: status flag cleared

Bit 18 **CFDCAN1F**: clear the illegal access flag for FDCAN1
0: no action
1: status flag cleared

Bit 17 **CLPTIM2F**: clear the illegal access flag for LPTIM2
0: no action
1: status flag cleared

Bit 16 **CI2C4F**: clear the illegal access flag for I2C4
0: no action
1: status flag cleared

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

Bit 14 **CI2C2F**: clear the illegal access flag for I2C2
0: no action
1: status flag cleared

Bit 13 **CI2C1F**: clear the illegal access flag for I2C1
0: no action
1: status flag cleared

Bit 12 **CUART5F**: clear the illegal access flag for UART5
0: no action
1: status flag cleared

Bit 11 **CUART4F**: clear the illegal access flag for UART4
0: no action
1: status flag cleared

Bit 10 **CUSART3F**: clear the illegal access flag for USART3
0: no action
1: status flag cleared

Bit 9 **CUSART2F**: clear the illegal access flag for USART2
0: no action
1: status flag cleared

Bit 8 **CSPI2F**: clear the illegal access flag for SPI2
0: no action
1: status flag cleared

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

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

Bit 5 **CTIM7F**: clear the illegal access flag for TIM7
0: no action
1: status flag cleared

Bit 4 **CTIM6F**: clear the illegal access flag for TIM6
0: no action
1: status flag cleared

Bit 3 **CTIM5F**: clear the illegal access flag for TIM5
0: no action
1: status flag cleared

Bit 2 **CTIM4F**: clear the illegal access flag for TIM4
0: no action
1: status flag cleared

Bit 1 **CTIM3F**: clear the illegal access flag for TIM3
0: no action
1: status flag cleared

Bit 0 **CTIM2F**: clear the illegal access flag for TIM2
0: no action
1: status flag cleared

5.7.10 GTZC1 TZIC flag clear register 2 (GTZC1_TZIC_FCR2)

Address offset: 0x024

Reset value: 0x0000 0000

Secure privileged 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.	CSAI2F	CSAI1F	CTIM17F	CTIM16F	CTIM15F	CUSART1F	CTIM8F	CSPI1F	CTIM1F
							w	w	w	w	w	w	w	w	w

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **CSAI2F**: clear the illegal access flag for SAI2
0: no action
1: status flag cleared

Bit 7 **CSAI1F**: clear the illegal access flag for SAI1
0: no action
1: status flag cleared

Bit 6 **CTIM17F**: clear the illegal access flag for TIM7
0: no action
1: status flag cleared

Bit 5 **CTIM16F**: clear the illegal access flag for TIM6
0: no action
1: status flag cleared

Bit 4 **CTIM15F**: clear the illegal access flag for TIM5
0: no action
1: status flag cleared

Bit 3 **CUSART1F**: clear the illegal access flag for USART1
0: no action
1: status flag cleared

Bit 2 **CTIM8F**: clear the illegal access flag for TIM8
0: no action
1: status flag cleared

Bit 1 **CSPI1F**: clear the illegal access flag for SPI1
0: no action
1: status flag cleared

Bit 0 **CTIM1F**: clear the illegal access flag for TIM1
0: no action
1: status flag cleared

5.7.11 GTZC1 TZIC flag clear register 3 (GTZC1_TZIC_FCR3)

Address offset: 0x028

Reset value: 0x0000 0000

Secure privilege 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.	CRAMCFGF	COCTOSPI2_REGF	COCTOSPI1_REGF	CFSMC_REGF	CSDMMC2F	CSDMMC1F	COCTOSPIMF
									w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSAESF	CPKAF	CRNGF	CHASHF	CAESF	COTGF	CDCMIF	CADC12F	CDCACHE1_REGF	CICACHE_REGF	CDMA2DF	CTSCF	CCRCF	CFMACF	CCORDICF	CMDF1F
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

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

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

Bit 21 **COCTOSPI2_REGF**: clear the illegal access flag for OCTOSPI2 registers
0: no action
1: status flag cleared

Bit 20 **COCTOSPI1_REGF**: clear the illegal access flag for OCTOSPI1 registers
0: no action
1: status flag cleared

Bit 19 **CFSMC_REGF**: clear the illegal access flag for FSMC registers
0: no action
1: status flag cleared

Bit 18 **CSDMMC2F**: clear the illegal access flag for SDMMC1
0: no action
1: status flag cleared

Bit 17 **CSDMMC1F**: clear the illegal access flag for SDMMC2
0: no action
1: status flag cleared

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

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

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

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

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

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

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

Bit 9 **CDCMIF**: clear the illegal access flag for DCMI and PSSI
0: no action
1: status flag cleared

Bit 8 **CADC12F**: clear the illegal access flag for ADC1
0: no action
1: status flag cleared

Bit 7 **CDCACHE1_REGF**: clear the illegal access flag for DCACHE1 registers
0: no action
1: status flag cleared

Bit 6 **CICACHE_REGF**: clear the illegal access flag for ICACHE registers
0: no action
1: status flag cleared

Bit 5 **CDMA2DF**: clear the illegal access flag for register of DMA2D
0: no action
1: status flag cleared

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

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

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

Bit 1 **CCORDICF**: clear the illegal access flag for CORDIC
0: no action
1: status flag cleared

Bit 0 **CMDF1F**: clear the illegal access flag for MDF1
0: no action
1: status flag cleared

5.7.12 GTZC1 TZIC flag clear register 4 (GTZC1_TZIC_FCR4)

Address offset: 0x02C

Reset value: 0x0000 0000

Secure privilege access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	CMPCBB3_REGF	CSRAM3F	CMPCBB2_REGF	CSRAM2F	CMPCBB1_REGF	CSRAM1F	Res.	Res.	Res.	Res.	COCTOSPI2_MEMF	CBKPSRAMF	CFSMC_MEMF	COCTOSPI1_MEMF
		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
CTZIC1F	CTZSC1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COTFDEC2F	COTFDEC1F	CFLASHF	CFLASH_REGF	CGPDMA1F
w	w										w	w	w	w	w

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

Bit 29 **CMPCBB3_REGF**: clear the illegal access flag for MPCBB3 registers
0: no action
1: status flag cleared

Bit 28 **CSRAM3F**: clear the illegal access flag for SRAM3
0: no action
1: status flag cleared

Bit 27 **CMPCBB2_REGF**: clear the illegal access flag for MPCBB2 registers
0: no action
1: status flag cleared

Bit 26 **CSRAM2F**: clear the illegal access flag for SRAM2
0: no action
1: status flag cleared

Bit 25 **CMPCBB1_REGF**: clear the illegal access flag for MPCBB1 registers
0: no action
1: status flag cleared

Bit 24 **CSRAM1F**: clear the illegal access flag for SRAM1
0: no action
1: status flag cleared

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **COCTOSPI2_MEMF**: clear the illegal access flag for OCTOSPI2 memory bank
0: no action
1: status flag cleared

Bit 18 **CBKPSRAMF**: clear the illegal access flag for MPCWM3 (BKPSRAM) memory bank
0: no action
1: status flag cleared

Bit 17 **CFSMC_MEMF**: clear the illegal access flag for MPCWM2 (FSMC NAND) and MPCWM3 (FSMC NOR)
0: no action
1: status flag cleared

Bit 16 **COCTOSPI1_MEMF**: clear the illegal access flag for MPCWM1 (OCTOSPI1) memory bank
0: no action
1: status flag cleared

Bit 15 **CTZIC1F**: clear the illegal access flag for GTZC1 TZIC registers
0: no action
1: status flag cleared

Bit 14 **CTZSC1F**: clear the illegal access flag for GTZC1 TZSC registers
0: no action
1: status flag cleared

Bits 13:5 Reserved, must be kept at reset value.

Bit 4 **COTFDEC2F**: clear the illegal access flag for OTFDEC2
0: no action
1: status flag cleared

Bit 3 **COTFDEC1F**: clear the illegal access flag for OTFDEC1
0: no action
1: status flag cleared

Bit 2 **CFLASHF**: clear the illegal access flag for FLASH memory
0: no action
1: status flag cleared

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

Bit 0 **CGPDMA1F**: clear the illegal access flag for GPDMA1
0: no action
1: status flag cleared

5.7.13 GTZC1 TZIC register map

Table 34. GTZC1 TZIC 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	GTZC1_TZIC_IER1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1IE	FDCAN1IE	LPTIM2IE	I2C4IE	CRSIE	I2C2IE	I2C1IE	UART5IE	UART4IE	USART3IE	USART2IE	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
0x004	GTZC1_TZIC_IER2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2IE	SAI1IE	TIM17IE	TIM16IE	TIM15IE	USART1IE	TIM8IE	SPI1IE	TIM1IE
	Reset value																								0	0	0	0	0	0	0	0	0
0x008	GTZC1_TZIC_IER3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGIE	OCTOSPI2_REGIE	OCTOSPI1_REGIE	FSMC_REGIE	SDMMC2IE	SDMMC1IE	OCTOSPIMIE	SAESIE	PKAIE	RNGIE	HASHIE	AESIE	OTGIE	DCMIIE	ADC12IE	DCACHE1_REGIE	ICACHE_REGIE	DMA2DIE	TSCIE	CRCIE	FMACIE	CORDICIE	MDF1IE
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	GTZC1_TZIC_IER4	Res.	Res.	MPCBB3_REGIE	SRAM3IE	MPCBB2_REGIE	SRAM2IE	MPCBB1_REGIE	SRAM1IE	Res.	Res.	Res.	Res.	OCTOSPI2_MEMIE	BKPSRAMIE	FSMC_MEMIE	OCTOSPI1_MEMIE	TZIC1IE	TZSC1IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTFDEC2IE	OTFDEC1IE	FLASHIE	FLASH_REGIE	GPDMA1IE
	Reset value			0	0	0	0	0	0					0	0	0	0	0	0										0	0	0	0	0
0x010	GTZC1_TZIC_SR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1F	FDCAN1F	LPTIM2F	I2C4F	CRSF	I2C2F	I2C1F	UART5F	UART4F	USART3F	USART2F	SPI2F	IWDGF	WWDGF	TIM7F	TIM6F	TIM5F	TIM4F	TIM3F	TIM2F
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	GTZC1_TZIC_SR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2F	SAI1F	TIM17F	TIM16F	TIM15F	USART1F	TIM8F	SPI1F	TIM1F
	Reset value																								0	0	0	0	0	0	0	0	0

Table 34. GTZC1 TZIC 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
0x018	GTZC1_TZIC_SR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGF	OCTOSPI2F	OCTOSPI1F	FSMCF	SDMMC2F	SDMMC1F	OCTOSPIMF	SAESF	PKAF	RNGF	HASHF	AESF	OTGF	DCMIF	ADC12F	DCACHE1F	ICACHEF	DMA2DF	TSCF	CRCF	FMACF	CORDICF	MDF1F
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	GTZC1_TZIC_SR4	Res.	Res.	MPCBB3_REGF	SRAM3F	MPCBB2_REGF	SRAM2F	MPCBB1_REGF	SRAM1F	Res.	Res.	Res.	Res.	OCTOSPI2_MEMIF	BKPSRAMIF	FSMC_MEMFE	OCTOSPI1_MEMFE	TZIC1F	TZSC1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTFDEC2F	OTFDEC1F	FLASHF	FLASH_REGF	GPDMA1F
	Reset value			0	0	0	0	0	0					0	0	0	0	0	0										0	0	0	0	0
0x020	GTZC1_TZIC_ FCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CUCPD1F	CFDCAN1F	CLPTIM2F	CI2C4F	CCRSF	CI2C2F	CI2C1F	CUART5F	CUART4F	CUSART3F	CUSART2F	CSPI2F	CIWDGF	CWWDGF	CTIM7F	CTIM6F	CTIM5F	CTIM4F	CTIM3F	CTIM2F
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	GTZC1_TZIC_ FCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSAI2F	CSAI1F	CTIM17F	CTIM16F	CTIM15F	CUSART1F	CTIM8F	CSPI1F	CTIM1F
	Reset value																								0	0	0	0	0	0	0	0	0
0x028	GTZC1_TZIC_ FCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRAMCFGF	COCTOSPI2F	COCTOSPI1F	CFSMCF	CSDMMC2F	CSDMMC1F	COCTOSPIMF	CSAESF	CPKAF	CRNGF	CHASHF	CAESF	COTGF	CDCMIF	ADC12F	CDCACHE1F	CICACHEF	CDMA2DF	CTSCF	CCRCF	CFMACF	CCORDICF	CMDF1F
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	GTZC1_TZIC_ FCR4	Res.	Res.	CMPCBB3_REGF	CSRAM3F	CMPCBB2_REGF	CSRAM2F	CMPCBB1_REGF	CSRAM1F	Res.	Res.	Res.	Res.	COCTOSPI2_MEMIF	CBKPSRAMIF	CFSMC_MEMFE	COCTOSPI1_MEMFE	CTZIC1F	CTZSC1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COTFDEC2F	COTFDEC1F	CFLASHF	CFLASH_REGF	CGPDMA1F
	Reset value			0	0	0	0	0	0					0	0	0	0	0	0										0	0	0	0	0

Refer to *Table 26: GTZC1 sub-blocks address offset*.

5.8 GTZC1 MPCBBz registers (z = 1 to 3)

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

5.8.1 GTZC1 SRAMz MPCBB control register (GTZC1_MPCBBz_CR) (z = 1 to 3)

Address offset: 0x000

Reset value: 0x0000 0000

Secure privileged 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.	GLOCK
															rs

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 access not allowed on non-secure SRAM block

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

Bit 30 **INVSECSTATE**: SRAMx clocks security state

This bit is used to define the internal SRAMs clocks control in RCC as secure or not.

0: SRAMs clocks are secure if a secure area exists in the MPCBB. It is non secure if there is no secure area.

1: SRAMs clocks are non-secure even if a secure area exists in the MPCBB, and secure even if no secure block is set in the MPCBB.

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

Bit 0 **GLOCK**: lock the control register of the MPCBB until next reset

This bit is cleared by default and once set, it can not be reset until system reset.

0: control register not locked

1: control register locked

5.8.2 GTZC1 SRAMz MPCBB configuration lock register 1 (GTZC1_MPCBBz_CFGLOCKR1) (z = 1 to 3)

Address offset: 0x010

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPLCK31	SPLCK30	SPLCK29	SPLCK28	SPLCK27	SPLCK26	SPLCK25	SPLCK24	SPLCK23	SPLCK22	SPLCK21	SPLCK20	SPLCK19	SPLCK18	SPLCK17	SPLCK16
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
SPLCK15	SPLCK14	SPLCK13	SPLCK12	SPLCK11	SPLCK10	SPLCK9	SPLCK8	SPLCK7	SPLCK6	SPLCK5	SPLCK4	SPLCK3	SPLCK2	SPLCK1	SPLCK0
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **SPLCKy**: Security/privilege configuration lock for super-block (y = 31 to 0)
This bit is set by software and can be cleared only by system reset.
0: GTZC1_MPCBBz_SECCFGRy and GTZC1_MPCBBz_PRIVCFGRy can be written.
1: Writes to GTZC1_MPCBBz_SECCFGRy and GTZC1_MPCBBz_PRIVCFGRy are ignored

5.8.3 GTZC1 SRAMz MPCBB security configuration for super-block x register (GTZC1_MPCBBz_SECCFGRx) (z = 1 to 3)

Address offset: 0x100 + 0x04 * x, (x = 0 to 31)

Reset value: 0xFFFF FFFF

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

Write access to this register is secure only. Any read is allowed.

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 **SECy**: Security configuration for block y (y = 31 to 0)
0: Non-secure access only to block y, belonging to super-block x. Secure access is also allowed if the SRWILADIS bit is set in GTZC1_MPCBBz_CR.
1: Secure access only to block y, belonging to super-block x.
Unprivileged write to this bit is ignored if PRIVy bit is set in GTZC1_MPCBBz_PRIVCFGRx.
Writes are ignored if SPLCKx bit is set in GTZC1_MPCBBz_CFGLOCKR1.

5.8.4 GTZC1 SRAMz MPCBB privileged configuration for super-block x register (GTZC1_MPCBBz_PRIVCFGRx) (z = 1 to 3)

Address offset: 0x200 + 0x04 * x, (x = 0 to 31)

Reset value: 0xFFFF FFFF

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

Write access to this register is privileged only. Any read is allowed.

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 **PRIVy**: Privileged configuration for block y, belonging to super-block x (y = 31 to 0).
0: Privileged and unprivileged access to block y, belonging to super-block x
1: Only privileged access to block y, belonging to super-block x
Non-secure write to this bit is ignored if SECy bit is set in GTZC1_MPCBBz_SECCFGRx.
Writes are ignored if SPLCKx bit is set in GTZC1_MPCBBz_CFGLOCKR1.

5.8.5 GTZC1 MPCBBz register map (z = 1 to 3)

Table 35. GTZC1 MPCBBz register map and reset values (z = 1 to 3)

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	GTZC1_ MPCBBz_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.	GLOCK
	Reset value	0	0																														0
0x004 to 0x00C	Reserved	Reserved																															
0x010	GTZC1_MPCBBz_ CFGLOCKR1	SPLCK31	SPLCK30	SPLCK29	SPLCK28	SPLCK27	SPLCK26	SPLCK25	SPLCK24	SPLCK23	SPLCK22	SPLCK21	SPLCK20	SPLCK19	SPLCK18	SPLCK17	SPLCK16	SPLCK15	SPLCK14	SPLCK13	SPLCK12	SPLCK11	SPLCK10	SPLCK9	SPLCK8	SPLCK7	SPLCK6	SPLCK5	SPLCK4	SPLCK3	SPLCK2	SPLCK1	SPLCK0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018-0x0FC	Reserved	Reserved																															
0x100 + 0x04 *x (x = 0 to 31)	GTZC1_MPCBBz_ SECCFGRx	SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	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
0x180 to 0x1FC	Reserved	Reserved																															
0x200 + 0x04 *x (x = 0 to 31)	GTZC1_MPCBBz_ PRIVCFGRx	PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	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 *Table 26: GTZC1 sub-blocks address offset*.

5.9 GTZC2 TZSC registers

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

5.9.1 GTZC2 TZSC control register (GTZC2_TZSC_CR)

Address offset: 0x000

Reset value: 0x0000 0000

Secure privilege 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCK
															rs

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

Bit 0 **LCK**: lock the configuration of GTZC2_TZSC_SECCFGRx and GTZC2_TZSC_PRIVCFGRx registers until next reset

This bit is cleared by default and once set, it can not be reset until system reset.

0: configuration of all GTZC2_TZSC_SECCCFGRx and all GTZC2_TZSC_PRIVCFGRx registers not locked

1: configuration of all GTZC2_TZSC_SECCCFGRx and all GTZC2_TZSC_PRIVCFGRx registers locked

5.9.2 GTZC2 TZSC secure configuration register 1 (GTZC2_TZSC_SECCFGR1)

Address offset: 0x010

Reset value: 0x0000 0000

Write-secure access only.

This register can be written only by secure privileged transaction when corresponding GTZC2_TZSC_PRIVCFGR register signal is set to 1. If a given PRIV bit is not set, the equivalent SEC bit can be written by secure unprivileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged 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.	ADF1SEC	DAC1SEC	Res.	VREFBUFSEC	ADC4SEC	COMPSEC	OPAMPSEC	LPTIM4SEC	LPTIM3SEC	LPTIM1SEC	I2C3SEC	LPUART1SEC	SPI3SEC
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **ADF1SEC**: secure access mode for ADF1
0: non-secure
1: secure

Bit 11 **DAC1SEC**: secure access mode for DAC1
0: non-secure
1: secure

Bit 10 Reserved, must be kept at reset value.

Bit 9 **VREFBUFSEC**: secure access mode for VREFBUF
0: non-secure
1: secure

Bit 8 **ADC4SEC**: secure access mode for ADC4
0: non-secure
1: secure

Bit 7 **COMPSEC**: secure access mode for COMP
0: non-secure
1: secure

Bit 6 **OPAMPSEC**: secure access mode for OPAMP
0: non-secure
1: secure

Bit 5 **LPTIM4SEC**: secure access mode for LPTIM4
0: non-secure
1: secure

Bit 4 **LPTIM3SEC**: secure access mode for LPTIM3
0: non-secure
1: secure

Bit 3 **LPTIM1SEC**: secure access mode for LPTIM1
0: non-secure
1: secure

Bit 2 **I2C3SEC**: secure access mode for I2C3
0: non-secure
1: secure

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

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

5.9.3 GTZC2 TZSC privilege configuration register 1 (GTZC2_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 GTZC2_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.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	ADF1PRIV	DAC1PRIV	Res.	VREFBUFPRIV	ADC4PRIV	COMPPRIV	OPAMPPRIV	LPTIM4PRIV	LPTIM3PRIV	LPTIM1PRIV	I2C3PRIV	LPUART1PRIV	SPI3PRIV
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **ADF1PRIV**: privileged access mode for ADF1
0: unprivileged
1: privileged

Bit 11 **DAC1PRIV**: privileged access mode for DAC1
0: unprivileged
1: privileged

Bit 10 Reserved, must be kept at reset value.

Bit 9 **VREFBUFPRIV**: privileged access mode for VREFBUF
0: unprivileged
1: privileged

Bit 8 **ADC4PRIV**: privileged access mode for ADC4
0: unprivileged
1: privileged

Bit 7 **COMPPRIV**: privileged access mode for COMP
0: unprivileged
1: privileged

Bit 6 **OPAMPPRIV**: privileged access mode for OPAMP
0: unprivileged
1: privileged

Bit 5 **LPTIM4PRIV**: privileged access mode for LPTIM4
0: unprivileged
1: privileged

Bit 4 **LPTIM3PRIV**: privileged access mode for LPTIM3
0: unprivileged
1: privileged

Bit 3 **LPTIM1PRIV**: privileged access mode for LPTIM1
0: unprivileged
1: privileged

Bit 2 **I2C3PRIV**: privileged access mode for I2C3
0: unprivileged
1: privileged

Bit 1 **LPUART1PRIV**: privileged access mode for LPUART1
0: unprivileged
1: privileged

Bit 0 **SPI3PRIV**: privileged access mode for SPI3
0: unprivileged
1: privileged

5.9.4 GTZC2 TZSC register map

Table 36. GTZC2 TZSC 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	GTZC2_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
0x004-0x00C	Reserved	Reserved																															
0x010	GTZC2_TZSC_SECCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1SEC	DAC1SEC	Res.	VREFBUFSEC	ADC4SEC	COMPSEC	OPAMPSEC	LPTIM4SEC	LPTIM3SEC	LPTIM1SEC	I2C3SEC	LPUART1SEC	SPI3SEC
	Reset value																				0	0		0	0	0	0	0	0	0	0	0	0
0x014-0x01C	Reserved	Reserved																															
0x020	GTZC2_TZSC_PRIVCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1PRIV	DAC1PRIV	Res.	VREFBUFPRIV	ADC4PRIV	COMPPRIV	OPAMPPRIV	LPTIM4PRIV	LPTIM3PRIV	LPTIM1PRIV	I2C3PRIV	LPUART1PRIV	SPI3PRIV
	Reset value																				0	0		0	0	0	0	0	0	0	0	0	0

Refer to *Table 27: GTZC2 sub-blocks address offset*.

5.10 GTZC2 TZIC registers

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

5.10.1 GTZC2 TZIC interrupt enable register 1 (GTZC2_TZIC_IER1)

Address offset: 0x000

Reset value: 0x0000 0000

Secure privilege 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.	ADF1IE	DAC1IE	Res.	VREFBUFIE	ADC4IE	COMPIE	OPAMPIE	LPTIM4IE	LPTIM3IE	LPTIM1IE	I2C3IE	LPUART1IE	SPI3IE
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **ADF1IE**: illegal access interrupt enable for ADF1
0: interrupt disabled
1: interrupt enabled

Bit 11 **DAC1IE**: illegal access interrupt enable for DAC1
0: interrupt disabled
1: interrupt enabled

Bit 10 Reserved, must be kept at reset value.

Bit 9 **VREFBUFIE**: illegal access interrupt enable for VREFBUF
0: interrupt disabled
1: interrupt enabled

Bit 8 **ADC4IE**: illegal access interrupt enable for ADC4
0: interrupt disabled
1: interrupt enabled

Bit 7 **COMPIE**: illegal access interrupt enable for COMP
0: interrupt disabled
1: interrupt enabled

Bit 6 **OPAMPIE**: illegal access interrupt enable for OPAMP
0: interrupt disabled
1: interrupt enabled

Bit 5 **LPTIM4IE**: illegal access interrupt enable for LPTIM4
0: interrupt disabled
1: interrupt enabled

Bit 4 **LPTIM3IE**: illegal access interrupt enable for LPTIM3
0: interrupt disabled
1: interrupt enabled

Bit 3 **LPTIM1IE**: illegal access interrupt enable for LPTIM1
0: interrupt disabled
1: interrupt enabled

Bit 2 **I2C3IE**: illegal access interrupt enable for I2C3
0: interrupt disabled
1: interrupt enabled

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

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

5.10.2 GTZC2 TZIC interrupt enable register 2 (GTZC2_TZIC_IER2)

Address offset: 0x004

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	MPCBB4_REGIE	SRAM4IE	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
TZIC2IE	TZSC2IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTIIE	LPDMA1IE	RCCIE	PWRIE	TAMPIE	RTCIE	SYSCFGIE
rw	rw								rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **MPCBB4_REGIE**: illegal access interrupt enable for MPCBB4 registers
0: interrupt disabled
1: interrupt enabled

Bit 24 **SRAM4IE**: illegal access interrupt enable for SRAM4
0: interrupt disabled
1: interrupt enabled

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 **TZIC2IE**: illegal access interrupt enable for GTZC2 TZIC registers
0: interrupt disabled
1: interrupt enabled

Bit 14 **TZSC2IE**: illegal access interrupt enable for GTZC2 TZSC registers
0: interrupt disabled
1: interrupt enabled

Bits 13:7 Reserved, must be kept at reset value.

Bit 6 **EXTIIE**: illegal access interrupt enable for EXTI
0: interrupt disabled
1: interrupt enabled

Bit 5 **LPDMA1IE**: illegal access interrupt enable for LPDMA
0: interrupt disabled
1: interrupt enabled

Bit 4 **RCCIE**: illegal access interrupt enable for RCC
0: interrupt disabled
1: interrupt enabled

Bit 3 **PWRIE**: illegal access interrupt enable for PWR
0: interrupt disabled
1: interrupt enabled

Bit 2 **TAMPIE**: illegal access interrupt enable for TAMP
0: interrupt disabled
1: interrupt enabled

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

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

5.10.3 GTZC2 TZIC status register 1 (GTZC2_TZIC_SR1)

Address offset: 0x010

Reset value: 0x0000 0000

Secure privileged 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.	ADF1F	DAC1F	Res.	VREFBUFF	ADC4F	COMPF	OPAMPF	LPTIM4F	LPTIM3F	LPTIM1F	I2C3F	LPUART1F	SPI3F
			r	r		r	r	r	r	r	r	r	r	r	r

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **ADF1F**: illegal access flag for ADF1
0: no illegal access event
1: illegal access event

Bit 11 **DAC1F**: illegal access flag for DAC1
0: no illegal access event
1: illegal access event

Bit 10 Reserved, must be kept at reset value.

Bit 9 **VREFBUFF**: illegal access flag for VREFBUF
0: no illegal access event
1: illegal access event

Bit 8 **ADC4F**: illegal access flag for ADC4
0: no illegal access event
1: illegal access event

Bit 7 **COMPF**: illegal access flag for COMP
0: no illegal access event
1: illegal access event

Bit 6 **OPAMPF**: illegal access flag for OPAMP
0: no illegal access event
1: illegal access event

Bit 5 **LPTIM4F**: illegal access flag for LPTIM4
0: no illegal access event
1: illegal access event

Bit 4 **LPTIM3F**: illegal access flag for LPTIM3
0: no illegal access event
1: illegal access event

Bit 3 **LPTIM1F**: illegal access flag for LPTIM1
0: no illegal access event
1: illegal access event

Bit 2 **I2C3F**: illegal access flag for I2C3
0: no illegal access event
1: illegal access event

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

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

5.10.4 GTZC2 TZIC status register 2 (GTZC2_TZIC_SR2)

Address offset: 0x014

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	MPCBB4_REGF	SRAM4F	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
TZIC2F	TZSC2F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTIF	LPDMA1F	RCCF	PWRF	TAMPF	RTCF	SYSCFGF
r	r								r	r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **MPCBB4_REGF**: illegal access flag for MPCBB4 registers
0: no illegal access event
1: illegal access event

Bit 24 **SRAM4F**: illegal access flag for SRAM4
0: no illegal access event
1: illegal access event

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 **TZIC2F**: illegal access flag for GTZC2 TZIC registers
0: no illegal access event
1: illegal access event

Bit 14 **TZSC2F**: illegal access flag for GTZC2 TZSC registers
0: no illegal access event
1: illegal access event

Bits 13:7 Reserved, must be kept at reset value.

Bit 6 **EXTIF**: illegal access flag for EXTI
0: no illegal access event
1: illegal access event

Bit 5 **LPDMA1F**: illegal access flag for LPDMA
0: no illegal access event
1: illegal access event

Bit 4 **RCCF**: illegal access flag for RCC
0: no illegal access event
1: illegal access event

Bit 3 **PWRF**: illegal access flag for PWR
0: no illegal access event
1: illegal access event

Bit 2 **TAMPF**: illegal access flag for TAMP
0: no illegal access event
1: illegal access event

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

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

5.10.5 GTZC2 TZIC flag clear register 1 (GTZC2_TZIC_FCR1)

Address offset: 0x020

Reset value: 0x0000 0000

Secure privileged 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.	CADF1F	CDAC1F	Res.	CVREFBUFF	CADC4F	CCOMPF	COPAMPF	CLPTIM4F	CLPTIM3F	CLPTIM1F	CI2C3F	CLPUART1F	CSPI3F
			w	w		w	w	w	w	w	w	w	w	w	w

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **CADF1F**: clear the illegal access flag for ADF1
0: no action
1: status flag cleared

Bit 11 **CDAC1F**: clear the illegal access flag for DAC1
0: no action
1: status flag cleared

Bit 10 Reserved, must be kept at reset value.

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

Bit 8 **CADC4F**: clear the illegal access flag for ADC4
0: no action
1: status flag cleared

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

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

Bit 5 **CLPTIM4F**: clear the illegal access flag for LPTIM4
0: no action
1: status flag cleared

Bit 4 **CLPTIM3F**: clear the illegal access flag for LPTIM3
0: no action
1: status flag cleared

Bit 3 **CLPTIM1F**: clear the illegal access flag for LPTIM1
0: no action
1: status flag cleared

Bit 2 **CI2C3F**: clear the illegal access flag for I2C3
0: no action
1: status flag cleared

Bit 1 **CLPUART1F**: clear the illegal access flag for LPUART1
0: no action
1: status flag cleared

Bit 0 **CSPI3F**: clear the illegal access flag for SPI3
0: no action
1: status flag cleared

5.10.6 GTZC2 TZIC flag clear register 2 (GTZC2_TZIC_FCR2)

Address offset: 0x024

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CMPCBB4_REGF	CSRAM4F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						w	w								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTZIC2F	CTZSC2F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEXTIF	CLPDMA1F	CRCCF	CPWRF	CTAMPF	CRTCF	CSYSCFGF
w	w								w	w	w	w	w	w	w

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **CMPCBB4_REGF**: clear the illegal access flag for MPCBB4 registers
0: no action
1: status flag cleared

Bit 24 **CSRAM4F**: clear the illegal access flag for SRAM4
0: no action
1: status flag cleared

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 **CTZIC2F**: clear the illegal access flag for GTZC2 TZIC registers
0: no action
1: status flag cleared

Bit 14 **CTZSC2F**: clear the illegal access flag for GTZC2 TZSC registers
0: no action
1: status flag cleared

Bits 13:7 Reserved, must be kept at reset value.

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

Bit 5 **CLPDMA1F**: clear the illegal access flag for LPDMA
0: no action
1: status flag cleared

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

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

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

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

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

5.10.7 GTZC2 TZIC register map

Table 37. GTZC2 TZIC 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	GTZC2_TZIC_IER1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1IE	DAC1IE	Res.	VREFBUFIE	ADC4IE	COMPIE	OPAMPIE	LPTIM4IE	LPTIM3IE	LPTIM1IE	I2C3IE	LPUART1IE	SPI3IE
	Reset value																				0	0		0	0	0	0	0	0	0	0	0	0
0x004	GTZC2_TZIC_IER2	Res.	Res.	Res.	Res.	Res.	Res.	MPCBB4_REGIE	SRAM4IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TZIC2IE	TZSC2IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTIIE	LPDMA1IE	RCCIE	PWRIE	TAMPIE	RTCIE	SYSCFGIE
	Reset value							0	0									0	0								0	0	0	0	0	0	0
0x008-0x00C	Reserved	Reserved																															
0x010	GTZC2_TZIC_SR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1F	DAC1F	Res.	VREFBUFF	ADC4F	COMPF	OPAMPF	LPTIM4F	LPTIM3F	LPTIM1F	I2C3F	LPUART1F	SPI3F
	Reset value																				0	0		0	0	0	0	0	0	0	0	0	0
0x014	GTZC2_TZIC_SR2	Res.	Res.	Res.	Res.	Res.	Res.	MPCBB4_REGF	SRAM4F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TZIC2F	TZSC2F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTIF	LPDMA1F	RCCF	PWRF	TAMPF	RTCF	SYSCFGF
	Reset value							0	0									0	0								0	0	0	0	0	0	0

Table 37. GTZC2 TZIC 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
0x018-0x01C	Reserved	Reserved																															
0x020	GTZC2_TZIC_FCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CADF1F	CDAC1F	Res.	CVREFBUFF	CADC4F	CCOMPF	COPAMPF	CLPTIM4F	CLPTIM3F	CLPTIM1F	CI2C3F	CLPUART1F	CSPI3F
	Reset value																				0	0		0	0	0	0	0	0	0	0	0	0
0x024	GTZC2_TZIC_FCR2	Res.	Res.	Res.	Res.	Res.	Res.	CMPCBB4_REGF	CSRAM4F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTZIC2F	CTZSC2F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEXTIF	CLPDMA1F	CRCCF	CPWRF	CTAMPF	CRTCF	CSYSCFGF
	Reset value							0	0									0	0								0	0	0	0	0	0	0

Refer to *Table 27: GTZC2 sub-blocks address offset*.

5.11 GTZC2 MPCBB4 registers

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

5.11.1 GTZC2 SRAM4 MPCBB control register (GTZC2_MPCBB4_CR)

Address offset: 0x000

Reset value: 0x0000 0000

Secure privileged 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.	GLOCK
															rs

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 access not allowed on non-secure SRAM block

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

Bit 30 **INVSECSTATE**: SRAMx clocks security state
This bit is used to define the internal SRAMs clocks control in RCC as secure or not.
0: SRAMs clocks are secure if a secure area exists in the MPCBB. It is non secure if there is no secure area.
1: SRAMs clocks are non-secure even if a secure area exists in the MPCBB, and secure even if no secure block is set in the MPCBB.

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

Bit 0 **GLOCK**: lock the control register of the MPCBB until next reset
This bit is cleared by default and once set, it can not be reset until system reset.
0: control register not locked
1: control register locked

5.11.2 GTZC2 SRAM4 MPCBB configuration lock register 1 (GTZC2_MPCBB4_CFGLOCKR1)

Address offset: 0x010

Reset value: 0x0000 0000

Secure privileged 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPLCK0
															rs

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

Bit 0 **SPLCK0**: Security/privilege configuration lock for super-block 0
This bit is set by software and can be cleared only by system reset.
0: GTZC2_MPCBB4_SECCFGR0 and GTZC2_MPCBB4_PRIVCFGR0 can be written.
1: Writes to GTZC2_MPCBB4_SECCFGR0 and GTZC1_MPCBB4_PRIVCFGR0 are ignored.

5.11.3 GTZC2 SRAM4 MPCBB security configuration for super-block 0 register (GTZC2_MPCBB4_SECCFGR0)

Address offset: 0x100

Reset value: 0xFFFF FFFF

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

Write access to this register is secure only. Any read is allowed.

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 **SECy**: Security configuration for block y, belonging to super-block 0 (y = 31 to 0)
0: Non-secure access only to block y, belonging to super-block 0. Secure access is also allowed if the SRWILADIS bit is set in GTZC2_MPCBB4_CR.
1: Secure access only to block y, belonging to super-block 0.
Unprivileged write to this bit is ignored if PRIVy bit is set in GTZC2_MPCBB4_PRIVCFGR0.
Write are ignored if SPLCK0 bit is set in GTZC2_MPCBB4_CFGLOCKR1.

5.11.4 GTZC2 SRAM4 MPCBB privileged configuration for super-block 0 register (GTZC2_MPCBB4_PRIVCFGR0)

Address offset: 0x200

Reset value: 0xFFFF FFFF

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

Write access to this register is privileged only. Any read is allowed.

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 **PRIVy**: Privileged configuration for block y, belonging to super-block 0 (y = 31 to 0).
0: Privileged and unprivileged access to block y, belonging to super block 0
1: privileged access only to block y, belonging to super-block 0
Non-secure write to this bit is ignored if SECy bit is set in GTZC2_MPBCC4_SECCFGR0.
Write are ignored if SPLCK0 bit is set in GTZC2_MPCBB4_CFGLOCKR1.

5.11.5 GTZC2 MPCBB4 register map

Table 38. GTZC2 MPCBB4 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	GTZC2_ MPCBB4_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.	GLOCK
	Reset value	0	0																														0
0x004-0x00C	Reserved	Reserved																															
0x010	GTZC2_MPCBB4_ CFGLOCKR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPLCK0
	Reset value																																0
0x014-0x0FC	Reserved	Reserved																															
0x100	GTZC2_MPCBB4_ SECCFGR0	SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	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
0x104-0x1FC	Reserved	Reserved																															
0x200	GTZC2_MPCBB4_ PRIVCFGR0	PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	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 *Table 27: GTZC2 sub-blocks address offset*.

6 RAMs configuration controller (RAMCFG)

6.1 Introduction

The RAMCFG configures the features of the internal SRAMs (SRAM1, SRAM2, SRAM3, SRAM4, and BKPSRAM).

6.2 RAMCFG main features

The internal SRAM supports some of the features listed hereafter, configured in RAMCFG:

- Error code correction (ECC):
  - Single error detection and correction with interrupt generation
  - Double error detection with interrupt or NMI generation
  - Status with failing address
- Write protection (1-Kbyte granularity)
- Programmable wait states for voltage scaling range 4
- SRAM software erase

6.3 RAMCFG functional description

6.3.1 Internal SRAMs features

Five SRAMs are embedded in the devices, each with specific features:

- SRAM1, SRAM2, SRAM3 are the main SRAMs. SRAM4 is in the SRAM used for peripherals low-power background autonomous mode (LPBAM) in Stop 2 mode.

  These SRAMs are made of several blocks that can be powered down in Stop mode to reduce consumption:
  - SRAM1: three 64-Kbyte blocks (total 192 Kbytes)
  - SRAM2: 8-Kbyte + 56-Kbyte blocks (total 64 Kbytes). In addition SRAM2 blocks can be retained in Standby mode.
  - SRAM3: eight 64-Kbyte blocks (total 512 Kbytes)
  - SRAM4: 16-Kbyte

  The backup SRAM (BKPSRAM) can be retained in all low-power modes and when V_{DD} is off in V_{BAT} mode.

  Refer to *Section 10: Power control (PWR)* for more details.
- All internal SRAMs are erased by hardware in case of Readout protection (RDP) level regression to Level 0.5 or Level 0. Refer to *Section 7: Embedded Flash memory (FLASH)* for more details.
- SRAM2 is erased when a system reset occurs if the SRAM2_RST option bit is selected in the Flash memory user option bytes. SRAM1, SRAM3 and SRAM4 are erased when a system reset occurs if the SRAM134_RST option bit is selected in the Flash memory

user option bytes. Refer to *Section 7: Embedded Flash memory (FLASH)* for more details.

- SRAM2 and optionally backup SRAM are protected by the tamper detection circuit, and are erased by hardware in case of tamper detection. They are also erased by hardware in case of a Backup domain reset. Refer to *Section 55: Tamper and backup registers (TAMP)* for more details.
- The RAMCFG embeds the registers related to the internal SRAMs ECC, write protection, wait state configuration and software erase.

The table below summarizes the features supported by each internal SRAM

Table 39. Internal SRAMs features

SRAM feature	SRAM1 (192 Kbytes)	SRAM2 (64 Kbytes)	SRAM3 (512 Kbytes)	SRAM4 (16 Kbytes)	BKPSRAM (2 Kbytes)
LPBAM in Stop 0/1 modes	X	X	X	X	X
LPBAM in Stop 2 mode	-	-	-	X	-
Optional retention in Standby mode	-	X	-	-	X
Optional retention in V_{BAT} mode	-	-	-	-	X
Erased with RDP regression	X	X	X	X	X
Erased or blocked by tamper detection, and erased with Backup domain reset	-	X	-	-	X(1)
Optionally erased with system reset	X	X	X	X	-
Software erase	X	X	X	X	X
ECC	-	X	X	-	X
Write protection	-	X	-	-	-
Wait states	X	X	X	X	X

1. Optional: BKPSRAM can be configured to be erased or not on tamper detection.

6.3.2 Error code correction (SRAM2, SRAM3, BKPSRAM)

The ECC is supported by SRAM2, SRAM3 and BKPSRAM when enabled with the SRAM2_ECC, SRAM3_ECC and BKPRAM_ECC user option bits. Refer to *Section 7: Embedded Flash memory (FLASH)* for more details.

Seven ECC bits are added per 32 bits of SRAM, allowing two bits error detection and one bit error correction on memory read access.

As the ECC is calculated and checked for a 32-bit word, the byte and half-word write accesses are managed by the SRAM interface by first reading the whole word, then write the word again with the new byte/half-word value. ECC double errors are also detected during these byte or half-word AHB write accesses (read/modify/write done by interface). The byte or half-word write access latency is WSC[2:0] + 2 AHB clock cycles (refer to *Section 6.3.4: Read access latency*).

Caution: In case of a byte or half-word write on SRAM with ECC, the read/modify/write operation is done in a buffer. The buffer content is written into the SRAM two AHB clock cycles after the SRAM AHB is released (when SRAM is no more accessed).

Single and double ECC errors

When a single error is detected, it is automatically corrected and the SEDC/CSEDC bits are set in the *RAMCFG memory interrupt status register (RAMCFG_MxISR)* and *RAMCFG memory x interrupt clear register x (RAMCFG_MxICR)* respectively. An interrupt is generated if enabled by the SEIE bit in the *RAMCFG memory x interrupt enable register (RAMCFG_MxIER)*. The failing address is stored in the *RAMCFG memory x ECC single error address register (RAMCFG_MxSEAR)* if the ALE bit is set in the *RAMCFG memory x control register (RAMCFG_MxCR)*.

Caution: Single errors cannot be detected when the SEDC bit is set.

When a double error is detected, the DED and CDED bits are set in the *RAMCFG memory interrupt status register (RAMCFG_MxISR)* and *RAMCFG memory x interrupt clear register x (RAMCFG_MxICR)* respectively. An interrupt or NMI is generated if enabled by the DEIE or ECCNMI bit in the *RAMCFG memory x interrupt enable register (RAMCFG_MxIER)*. The failing address is stored in the *RAMCFG memory x ECC double error address register (RAMCFG_MxDEAR)* if the ALE bit is set in the *RAMCFG memory x control register (RAMCFG_MxCR)*.

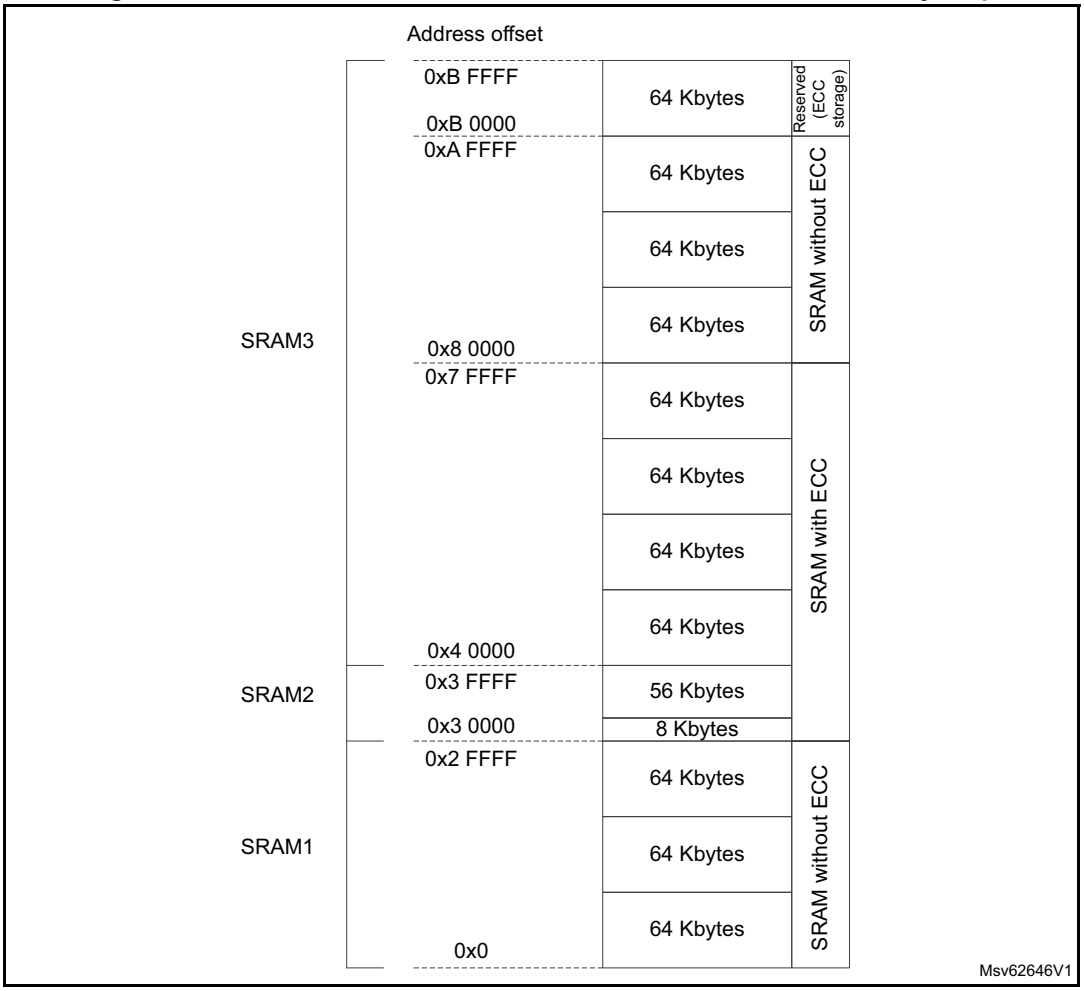
Caution: Double errors cannot be detected when the DED bit is set.

SRAM3 ECC specific management

When the ECC is enabled for SRAM3, only the first 256 Kbytes of SRAM3 are with ECC. The next 192 Kbytes are without ECC, and the last block is used to store the ECC, so cannot be used for application.

The figure below shows the SRAM areas, when SRAM2 and SRAM3 ECC are enabled.

Figure 19. SRAM1, SRAM2 with ECC and SRAM3 with ECC memory map



When ECC is enabled by user option bits, the ECCE bit is automatically set after system reset in the related *RAMCFG memory x control register (RAMCFG_MxCR)*.

The ECC can be deactivated by executing the following software sequence:

1. Write 0xAE in the *RAMCFG memory x ECC key register (RAMCFG_MxECCKEYR)*.
2. Write 0x75 in the *RAMCFG memory x ECC key register (RAMCFG_MxECCKEYR)*.
3. Write 0 in the ECCE bit of the *RAMCFG memory x control register (RAMCFG_MxCR)*.

In case ECC is deactivated (ECCE = 0), the SRAM3 ECC storage area (from offset 0xB0000 to offset 0xBFFFF) can be read and written as other SRAM3 areas. In order to test the ECC mechanism, only the first 256 Kbytes of SRAM3 can be modified, 1 or 2 bits by word (for single or double error test respectively).

The procedure to check ECC is the following:

1. On an erased memory, write data with ECC ON.
2. Disable ECC.
3. Write same data with 1- or 2-bit modification (for single or double error test respectively).
4. Enable ECC.
5. Read data. Enabled interrupt is generated because of single or double error.

Warning: The ECC fault injection test triggers a system break events in TIM1/TIM8/TIM15/TIM16/TIM17, if the SPL bit is set in SYSCFG_CFGR2.
This implies that the test must be performed while the PWM outputs of the timers are in idle state.

6.3.3 Write protection (SRAM2)

The SRAM2 is made of 64 1-Kbyte pages. Each 1-Kbyte page can be write-protected by setting its corresponding PxWP (x = 0 to 63) bit in the *RAMCFG memory 2 write protection register 1 (RAMCFG_M2WPR1)* and *RAMCFG memory 2 write protection register 2 (RAMCFG_M2WPR2)*.

6.3.4 Read access latency

To correctly read data from SRAMs, the number of wait states must be correctly programmed in the WSC[2:0] field of the *RAMCFG memory x control register (RAMCFG_MxCR)*, depending on AHB clock frequency (HCLK) and voltage scaling range, as shown in the table below.

Table 40. Number of wait states versus HCLK frequency and voltage range scaling

Wait states (WS) (Latency)	HCLK (MHz)			
	V_{CORE} range 1	V_{CORE} range 2	V_{CORE} range 3	V_{CORE} range 4 LPBAM(1)
0 WS (1 AHB cycle)	≤ 160	≤ 110	≤ 55	≤ 16
1 WS (2 AHB cycle)	-	-	-	≤ 25

1. LPBAM: low-power background autonomous mode. The system clock can be requested in Stop mode to perform DMA transfers.

6.3.5 Software erase

SRAM erase can be requested by executing this software sequence:

1. Write 0xCA in the *RAMCFG memory x erase key register (RAMCFG_MxERKEYR)*.
2. Write 0x53 in the *RAMCFG memory x erase key register (RAMCFG_MxERKEYR)*.
3. Write 1 in the SRAMER bit of the *RAMCFG memory x control register (RAMCFG_MxCR)*.

SRAMBUSY flag is set in the related SRAM interrupt status register as long as the erase is on going.

The total duration of each SRAM erase is N AHB clock cycles, where N is the size of the SRAM in 32-bit words.

If the SRAM is read or written while an erase is on going, wait states are inserted on the AHB bus until the end of the erase operation.

6.4 RAMCFG low-power modes

Table 41. Effect of low-power modes on RAMCFG

Mode	Description
Sleep	No effect. RAMCFG interrupts cause the device to exit the Sleep mode.
Stop	The content of RAMCFG registers is kept. The ECC is functional and ECC error interrupt or NMI causes the device to exit from Stop 0 and Stop 1 modes.
Standby	The RAMCFG peripheral is powered down and must be reinitialized after exiting Standby.

6.5 RAMCFG interrupts

The table below gives the list of RAMCFG interrupt requests.

Table 42. RAMCFG interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit the Sleep mode	Exit the Stop mode	Exit the Standby modes
RAMCFG	ECC single error detection and correction	SEDC	SEIE	Write 1 in CSEDC	Yes	Yes(1)	No
	ECC double error detection	DED	DEIE = 1 and ECCNMI = 0	Write 1 in CDED	Yes	Yes(1)	No
NMI	ECC double error detection	DED	ECCNMI	Write 1 in CDED	Yes	Yes(1)	No

1. Stop 0 and Stop 1 modes only

6.6 RAMCFG registers

In the registers described below, x refers to:

- SRAM1/2/3/4 when x = 1/2/3/4 respectively

- BKPSRAM when x = 5

6.6.1 RAMCFG memory x control register (RAMCFG_MxCR)

Address offset: 0x040 * (x - 1), (x = 1 to 5)

Reset value: 0x0000 000X

ECCE reset value depends on ECC enable user option bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WSC[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.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
							rs				rw				rw

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

Bits 18:16 **WSC[2:0]**: Wait state configuration

This field is used to program the number of wait states inserted on the AHB when reading the SRAM, depending on its access time.

000: 0 wait state

001: 1 wait state

...

111: 7 wait states (not needed)

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

Bit 8 **SRAMER**: SRAM erase

This bit can be set by software only after writing the unlock sequence in the ERASEKEY field of the RAMCFG_MxERKEYR register. Setting this bit starts the SRAM erase. This bit is automatically cleared by hardware at the end of the erase operation.

0: No erase operation on going

1: Erase operation on going

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **ALE**: Address latch enable

0: Failing address not stored in the SRAMx ECC single/double error address registers

1: Failing address stored in the SRAMx ECC single/double error address registers

Note: This bit is reserved and must be kept at reset value in SRAM1 and SRAM4 control registers.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **ECCE**: ECC enable.

This bit reset value is defined by the user option bit configuration. When set, it can be cleared by software only after writing the unlock sequence in the RAMCFG_MxECCKEYR register.

0: ECC disabled

1: ECC enabled

Note: This bit is reserved and must be kept at reset value in SRAM1 and SRAM4 control registers.

6.6.2 RAMCFG memory x interrupt enable register (RAMCFG_MxIER)

Address offset: 0x004 + 0x040 * (x - 1), (x = 2, 3, 5)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	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.	ECCNMI	Res.	DEIE	SEIE
												rs		rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ECCNMI**: Double error NMI

This bit is set by software and cleared only by a global RAMCFG reset.

0: NMI not generated in case of ECC double error

1: NMI generated in case of ECC double error

Note: if ECCNMI is set, the RAMCFG maskable interrupt is not generated whatever DEIE bit value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DEIE**: ECC double error interrupt enable

0: Double error interrupt disabled

1: Double error interrupt enabled

Bit 0 **SEIE**: ECC single error interrupt enable

0: Single error interrupt disabled

1: Single error interrupt enabled

6.6.3 RAMCFG memory interrupt status register (RAMCFG_MxISR)

Address offset: 0x008 + 0x040 * (x - 1), (x = 1 to 5)

Reset value: 0x0000 0000

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMBUSY**: SRAM busy with erase operation

0: No erase operation on going

1: Erase operation on going

Note: Depending on the SRAM, the erase operation can be performed due to software request, system reset if the option bit is enabled, tamper detection or readout protection regression. Refer to Table 39: Internal SRAMs features.

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

Bit 1 **DED**: ECC double error detected

0: No double error

1: Double error detected

Note: This bit is reserved and must be kept at reset value in SRAM1 and SRAM4 interrupt status registers.

Bit 0 **SEDC**: ECC single error detected and corrected

0: No single error

1: Single error detected and corrected

Note: This bit is reserved and must be kept at reset value in SRAM1 and SRAM4 interrupt status registers.

6.6.4 RAMCFG memory x ECC single error address register (RAMCFG_MxSEAR)

Address offset: 0x00C + 0x040 * (x - 1), (x = 2, 3, 5)

Reset value: 0x0000 0000

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

Bits 31:0 **ESEA[31:0]**: ECC single error address

When the ALE bit is set in the RAMCFG_MxCR register, this field is updated with the address corresponding to the ECC single error.

6.6.5 RAMCFG memory x ECC double error address register (RAMCFG_MxDEAR)

Address offset: 0x010 + 0x040 * (x - 1), (x = 2, 3, 5)

Reset value: 0x0000 0000

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

Bits 31:0 **EDEA[31:0]**: ECC double error address

When the ALE bit is set in the RAMCFG_MxCR register, this field is updated with the address corresponding to the ECC double error.

6.6.6 RAMCFG memory x interrupt clear register x (RAMCFG_MxICR)

Address offset: 0x014 + 0x040 * (x - 1), (x = 2, 3, 5)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	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.	CDED	CSEDC
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **CDED**: Clear ECC double error detected

Writing 1 to this flag clears the DED bit in the RAMCFG_MxISR register. Reading this flag returns the DED value.

Bit 0 **CSEDC**: Clear ECC single error detected and corrected

Writing 1 to this flag clears the SEDC bit in the RAMCFG_MxISR register. Reading this flag returns the SEDC value.

6.6.7 RAMCFG memory 2 write protection register 1 (RAMCFG_M2WPR1)

Address offset: 0x058

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 **PyWP**: SRAM2 1-Kbyte page y write protection (y = 31 to 0)
These bits are set by software and cleared only by a global RAMCFG reset.
0: Write protection of SRAM2 1-Kbyte page y is disabled.
1: Write protection of SRAM2 1-Kbyte page y is enabled.

6.6.8 RAMCFG memory 2 write protection register 2 (RAMCFG_M2WPR2)

Address offset: 0x05C

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 **PyWP**: SRAM2 1-Kbyte page y write protection (y = 63 to 32)
These bits are set by software and cleared only by a global RAMCFG reset.
0: Write protection of SRAM2 1-Kbyte page y is disabled.
1: Write protection of SRAM2 1-Kbyte page y is enabled.

6.6.9 RAMCFG memory x ECC key register (RAMCFG_MxECCKEYR)

Address offset: 0x024 + 0x040 * (x - 1), (x = 2, 3, 5)

Reset value: 0x0000 0000

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

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

Bits 7:0 **ECCKEY[7:0]**: ECC write protection key

The following steps are required to unlock the write protection of the ECCE bit in the RAMCFG_MxCR register.

1) Write 0xAE into ECCKEY[7:0].

2) Write 0x75 into ECCKEY[7:0].

Note: Writing a wrong key reactivates the write protection.

6.6.10 RAMCFG memory x erase key register (RAMCFG_MxERKEYR)

Address offset: 0x028 + 0x040 * (x - 1), (x = 1 to 5)

Reset value: 0x0000 0000

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

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

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

Bits 7:0 **ERASEKEY[7:0]**: Erase write protection key

The following steps are required to unlock the write protection of the SRAMER bit in the RAMCFG_MxCR register.

1) Write 0xCA into ERASEKEY[7:0].

2) Write 0x53 into ERASEKEY[7:0].

Note: Writing a wrong key reactivates the write protection.

6.6.11 RAMCFG register map

Table 43. RAMCFG 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	RAMCFG_M1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WSC[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
	Reset value														0	0	0								0				0				x
0x04	Reserved	Reserved																															
0x08	RAMCFG_M1ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																								0								
0x0C to 0x24	Reserved	Reserved																															
0x28	RAMCFG_ M1ERKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x2C to 0x3C	Reserved	Reserved																															

Table 43. RAMCFG 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
0x40	RAMCFG_M2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WSC[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
	Reset value														0	0	0								0				0				x
0x44	RAMCFG_M2IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCNMI	Res.	DEIE	SEIE
	Reset value																													0		0	0
0x048	RAMCFG_M2ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	DED	SEDC
	Reset value																								0							0	0
0x04C	RAMCFG_ M2SEAR	ESEA[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	RAMCFG_ M2DEAR	EDEA[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	RAMCFG_M2ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDED	CSEDC
	Reset value																															0	0
0x058	RAMCFG_ M2WPR1	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
0x05C	RAMCFG_ M2WPR2	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
0x060	Reserved	Reserved																															
0x064	RAMCFG_ M2ECCKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x068	RAMCFG_ M2ERKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x06C to 0x07C	Reserved	Reserved																															
0x080	RAMCFG_M3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WSC[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
	Reset value														0	0	0								0				0				0
0x084	RAMCFG_M3IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCNMI	Res.	DEIE	SEIE
	Reset value																													0		0	0
0x088	RAMCFG_M3ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	DED	SEDC
	Reset value																								0							0	0
0x08C	RAMCFG_ M3SEAR	ESEA[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	RAMCFG_ M3DEAR	EDEA[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 43. RAMCFG 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
0x094	RAMCFG_M3ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDED	CSEDC
	Reset value																															0	0
0x098 to 0x0A0	Reserved	Reserved																															
0x0A4	RAMCFG_ M3ECCKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x0A8	RAMCFG_ M3ERKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x0AC to 0x0BC	Reserved	Reserved																															
0x0C0	RAMCFG_M4CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WSC[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
	Reset value														0	0	0								0				0				x
0x0C4	Reserved	Reserved																															
0x0C8	RAMCFG_M4ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																								0								
0x0CC to 0x0E4	Reserved	Reserved																															
0x0E8	RAMCFG_ M4ERKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[2:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x0EC to 0x0FC	Reserved	Reserved																															
0x100	RAMCFG_M5CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WSC[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
	Reset value														0	0	0								0				0				x
0x104	RAMCFG_M5IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCNMI	Res.	DEIE	SEIE
	Reset value																													0		0	0
0x108	RAMCFG_M5ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	DED	SEDC
	Reset value																								0							0	0
0x10C	RAMCFG_ M5SEAR	ESEA[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
0x110	RAMCFG_ M5DEAR	EDEA[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
0x114	RAMCFG_M5ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDED	CSEDC
	Reset value																															0	0
0x118 to 0x120	Reserved	Reserved																															
0x124	RAMCFG_ M5ECCKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 43. RAMCFG 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
0x128	RAMCFG_ M5ERKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

7 Embedded Flash memory (FLASH)

7.1 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. It is optimized in terms of power consumption with dedicated modes when the MCU is in low-power modes.

7.2 FLASH main features

- Up to 2 Mbytes of Flash memory supporting read-while-write capability (RWW).
- Memory organization
  - Dual bank architecture (bank 1 and bank 2)
  - Main memory: up to 1 Mbyte per bank
  - Information block: 64.5 Kbytes in bank 1
- 128-bit wide data read with prefetch
- Standard and burst programming modes
- Read, program and erase operations in all voltage ranges
- 10 kcycles endurance on all Flash memory. 100 kcycles on 256 Kbytes per bank
- Page erase, bank erase and mass erase (both banks)
- Bank swapping: the user Flash memory address mapping of each bank can be swapped.
- Product security activated by TrustZone option bit (TZEN)
- Device life cycle managed by readout protection option byte (RDP)
- Four write protection areas (two per bank)
- TrustZone support:
  - Two secure areas (1 per bank)
  - Two secure HDP (hide protection) areas part of the secure areas (one per bank)
- Configurable protection against unprivileged accesses with Flash page granularity
- Error code correction: 9 bits ECC per 128-bit quad-word allowing two bits error detection and one bit error correction
- Option-byte loader
- Advanced low-power modes (low-power read mode, bank power-down mode)

7.3 FLASH functional description

7.3.1 Flash memory organization

The Flash memory has the following main features:

- Capacity up to 2 Mbytes
- Dual-bank mode:
  - 1 Mbyte per bank for main memory
  - 8 Kbytes page size
  - 137 bits wide data read and write (128 effective bits plus 9 ECC bits)
  - Page, bank and mass erase

The Flash memory is organized as follows:

- Main memory block organized as two banks of 1 Mbyte each containing 128 pages of 8 Kbytes
- An information block containing:
  - 32 Kbytes for system memory. This area is immutable and reserved for use by STMicroelectronics. It contains the bootloader that is used to reprogram the Flash memory through one of the user communication interfaces such as USB (DFU). The system memory is programmed by STMicroelectronics when the device is manufactured. For further details, refer to the application note *STM32 microcontroller system memory boot mode (*AN2606).
  - 32 Kbytes immutable secure area containing the root security services (RSS and RSS library) developed by STMicroelectronics
  - 512 bytes OTP (one-time programmable) bytes for user data (32 quad-words). The OTP data cannot be erased and can be written only once.
  - 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 the table below.

Table 44. Flash module 2-Mbyte dual bank organization

Flash area		Flash memory address	Size	Name
Main memory	Bank 1	0x0800 0000 - 0x0800 1FFF	8 Kbytes	Page 0
		0x0800 2000 - 0x0800 3FFF	8 Kbytes	Page 1
		...	...	...
		0x080F E000 - 0x080F FFFF	8 Kbytes	Page 127
	Bank 2	0x0810 0000 - 0x0810 1FFF	8 Kbytes	Page 0
		0x0810 2000 - 0x0810 3FFF	8 Kbytes	Page 1
		...	...	...
		0x081F E000 - 0x081F FFFF	8 Kbytes	Page 127
Non-secure information block		0x0BF9 0000 - 0x0BF9 7FFF	32 Kbytes	System memory
		0x0BFA 0000 - 0x0BFA 01FF	512 bytes	OTP area

Table 44. Flash module 2-Mbyte dual bank organization (continued)

Flash area	Flash memory address	Size	Name
Secure information block	0x0FF8 0000 - 0x0FF8 5FFF	24 Kbytes	RSS
	0x0FF8 6000 - 0x0FF8 7FFF	8 Kbytes	RSS library

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

7.3.2 Error code correction (ECC)

Data in Flash memory are 137-bit words: Nine bits are added per quad-word (128 bits). The ECC mechanism supports:

- one error detection and correction
- two errors detection

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

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

When an ECC error is detected, the address of the failing quad-word and its associated bank are saved in ADDR_ECC[19:0] and BK_ECC in the *FLASH ECC register (FLASH_ECCR)*. ADDR_ECC[3:0] 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.

Caution: When the ECCC flag is set, a further two-errors detection is not able to generate the NMI or break signal to timers. It is therefore recommended to clear the ECCC flag as soon as a correction is operated, to preserve the ECC error detection capability. In case of a double ECC error detection (ECCD flag set and NMI triggered), the software must clean the cache in the NMI handler. Refer to *STM32U5 Series safety manual* (UM2875) for the full description of the implications on safety standards compliance.

Caution: For an erased Flash line, 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.

The following addresses in the system Flash memory are used to store words including ECC errors to allow run-time tests by software on ECC correction detection capability:

- 0x0BFA1F00 (embeds a word with 1-bit error)
- 0x0BFA1F80 (embeds a word with 2-bit error)

In case the second address is read, for instance by the debugger memory viewer, a NMI is generated.

7.3.3 Read access latency

To correctly read data from Flash memory, the number of wait states (latency) must be correctly programmed in the *FLASH access control register (FLASH_ACR)* according to the frequency of the CPU clock (HCLK) and the internal voltage range of the device V_{CORE}. Refer to *Section 10.5.4: Dynamic voltage scaling management*.

The table below shows the correspondence between wait states and CPU clock frequency.

Table 45. Number of wait states according to CPU clock (HCLK) frequency (LPM = 0)

Wait states (WS) (latency)	HCLK (MHz)			
	V_{CORE} range 1	V_{CORE} range 2	V_{CORE} range 3	V_{CORE} range 4
0 WS (1 CPU cycle)	≤ 32	≤ 30	≤ 24	≤ 12
1 WS (2 CPU cycles)	≤ 64	≤ 60	≤ 48	≤ 25
2 WS (3 CPU cycles)	≤ 96	≤ 90	≤ 55	-
3 WS (4 CPU cycles)	≤ 128	≤ 110	-	-
4 WS (5 CPU cycles)	≤ 160	-	-	-

The Flash memory supports a low-power read mode when setting the LPM bit in the *FLASH access control register (FLASH_ACR)*. The table below shows the correspondence between wait states and CPU clock frequency when LPM bit is set.

Table 46. Number of wait states according to CPU clock (HCLK) frequency (LPM = 1)

Wait states (WS) (Latency)	HCLK (MHz)	
	V_{CORE} range 1/2/3	V_{CORE} range 4
0 WS (1 CPU cycle)	WS ≥ HCLK (MHz) / 10 -1 Maximum HCLK frequency is given by *Table 45*	≤ 8
1 WS (2 CPU cycles)		≤ 16
2 WS (3 CPU cycles)		≤ 25
3 WS (4 CPU cycles)		-
...		-
15 WS (16 CPU cycles)		-

After reset, the CPU clock frequency is 4 MHz, 0 wait state (WS) is configured in the *FLASH access control register (FLASH_ACR)* and the normal read mode is selected (LPM = 0).

Instruction prefetch

The Cortex-M33 fetches instructions and literal pools (constants/data) over the C-Bus and through the instruction cache if it is enabled. The prefetch block aims at increasing the efficiency of C-Bus accesses in case the instruction cache is enabled by reducing the cache refill latency. Prefetch is efficient in case of sequential code; prefetch in the Flash memory allows the next sequential instruction line to be read from the Flash memory while the current instruction line is being filled in instruction cache and executed by the CPU.

Prefetch is enabled by setting the PRFTEN bit in the *FLASH access control register (FLASH_ACR)*. PRFTEN must be set only if at least one wait state is needed to access the Flash memory.

Note: Prefetch tends to increase the code execution performance at the cost extra Flash memory accesses. It must be used carefully in low-power applications.

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

Increase the CPU frequency

1. Program the new number of wait states to the LATENCY bits in the *FLASH access control register (FLASH_ACR)*.
2. Check that the new number of wait states is taken into account to access the Flash memory by reading back the *FLASH access control register (FLASH_ACR)*.
3. Modify the CPU clock source by writing the SW bits in the RCC_CFGR1 register.
4. Modify the CPU clock prescaler, if needed, by writing the HPRE bits in RCC_CFGR2.
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_CFGR1 and RCC_CFGR2 registers.

Decrease the CPU frequency

1. Modify the CPU clock source by writing the SW bits in the RCC_CFGR1 register.
2. Modify the CPU clock prescaler, if needed, by writing the HPRE bits in RCC_CFGR2.
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_CFGR1 and RCC_CFGR2 registers.
4. Program the new number of wait states to the LATENCY bits in the *FLASH access control register (FLASH_ACR)*.
5. Check that the new number of wait states is used to access the Flash memory by reading back the *FLASH access control register (FLASH_ACR)*.

The software sequences detailed below must be applied in order to modify the read mode.

From normal read mode to low-power read mode

1. Set the LPM bit in the *FLASH access control register (FLASH_ACR)*.
2. Check that the low-power read mode is activated by reading the *FLASH access control register (FLASH_ACR)*.

From low-power read mode to normal read mode

1. Reset the LPM bit in the *FLASH access control register (FLASH_ACR)*.
2. Check that the normal read mode is activated by reading the *FLASH access control register (FLASH_ACR)*.

7.3.4 Bank power-down mode

After reset, both banks are in normal mode. In order to reduce power consumption, each bank can be independently put in power-down mode by setting the PDREQx bit in the *FLASH access control register (FLASH_ACR)*.

Request entry in power-down mode for bank x

- Check that bank x is not in power-down mode and no request to put it in power-down mode is pending (PDx bit in *FLASH non-secure status register (FLASH_NSSR)* and PDREQx bit in *FLASH access control register (FLASH_ACR)* must be reset).
- Unlock PDKEYxR with correct keys. Refer to *FLASH bank 1 power-down key register (FLASH_PDKEY1R)* or *FLASH bank 2 power-down key register (FLASH_PDKEY2R)*
- Set the PDREQx bit in the *FLASH access control register (FLASH_ACR)*.
- Check that the PDx bit in the *FLASH non-secure status register (FLASH_NSSR)* is set. The PDREQx bit in the *FLASH access control register (FLASH_ACR)* is automatically reset and the PDKEYxR is locked.

Note: If bank x is currently being accessed, the power-down request is delayed until the access is completed.

Requesting power-down entry for a bank already in power-down mode has no effect. The PDREQx bit in the *FLASH access control register (FLASH_ACR)* is automatically reset and the PDKEYxR is locked.

Return to normal mode

Any access to a bank in power-down mode automatically wakes up the bank. A penalty of 5 µs minimum is taken to wake up the bank.

Wake up bank 1 (respectively bank 2) is done in one of the following cases:

- upon a valid read access to bank 1 (resp. bank 2)
- upon a valid write access to bank 1(resp. bank 2)
- upon a valid bank erase on bank 1 (resp. bank 2)

Waking up both bank 1 and bank 2 is done in one of the following cases:

- upon a valid mass erase
- upon an option byte modification
- upon an option byte loading
- upon system reset

Note: The software can reduce the Flash bank wake-up time by enabling HSI16 before waking up the bank.

7.3.5 Flash memory 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 (such as I/Os, USB, CAN, UART, I2C or SPI) to download programming data into the 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.

An ongoing 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 7.3.10: Read-while-write (RWW)*).

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 TrustZone that defines secure and non-secure areas in the Flash memory. 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 7.5: FLASH TrustZone security and privilege protections*.

Unlock the secure/non-secure Flash control registers

After reset, write is not allowed in the Flash control registers (*FLASH secure control register (FLASH_SECCR)* and *FLASH non-secure control register (FLASH_NSCR)*) in order to protect the Flash memory against possible unwanted operations due, for example, to electric disturbances.

The following sequence is used to unlock these registers:

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 secure control register (FLASH_SECCR)* or *FLASH non-secure control register (FLASH_NSCR)* 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 (resp. FLASH_SECCR) register can be locked again by software by setting the LOCK bit in the FLASH_NSCR (resp. FLASH_SECCR) register.

*Note: The FLASH_*NSCR and FLASH_SECCR *registers cannot be written when the BSY bits are set. Any attempt to write them with the BSY bits set, causes the AHB bus to stall until the BSY bits are cleared.*

Wait for data-to-write flags (WDW)

The WDW flags in the *FLASH non-secure status register (FLASH_NSSR)* and *FLASH secure status register (FLASH_SECSR)* are both set when a secure or non-secure write access has been done in the write buffer. They are cleared when the BSY flags are set (meaning that the write buffer is freed and the programming operation actually starts in the Flash memory) or in case of error.

It is the software responsibility to ensure that the four words in the same quad-word are all written.

Flash secure and non-secure busy flags

The BSY flags in the *FLASH non-secure status register (FLASH_NSSR)* and *FLASH secure status register (FLASH_SECSR)* are both set when a secure or non-secure Flash operation is started:

- Erase operation: setting the STRT bit in the *FLASH non-secure control register (FLASH_NSCR)* or *FLASH secure control register (FLASH_SECCR)*
- Write operation: setting the PG bit in the *FLASH non-secure control register (FLASH_NSCR)* or *FLASH secure control register (FLASH_SECCR)* and writing a quad-word in the Flash memory
- Option-byte programming: setting the OPTSTRT in the *FLASH non-secure control register (FLASH_NSCR)*

7.3.6 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.

Page erase

To erase a page, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the *FLASH non-secure status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. Set the PER bit and select the page to erase (PNB) with the associated bank (BKER) in the *FLASH non-secure control register (FLASH_NSCR)* or *FLASH secure control register (FLASH_SECCR)*.
4. Set the STRT bit in the *FLASH non-secure control register (FLASH_NSCR)* or *FLASH secure control register (FLASH_SECCR)*.
5. Wait for the BSY bit to be cleared in the *FLASH non-secure status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.

Bank 1 or bank 2 mass erase

To perform a bank mass erase, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the *FLASH non-secure status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.
2. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
3. Set the MER1 or MER2 bit (depending on the bank) in the *FLASH non-secure control register (FLASH_NSCR)* or *FLASH secure control register (FLASH_SECCR)*. Both banks can be selected in the same operation, in that case it corresponds to a mass erase.
4. Set the STRT bit in the *FLASH non-secure control register (FLASH_NSCR)* or *FLASH secure control register (FLASH_SECCR)*.

5. Wait for the BSY bit to be cleared in the *FLASH non-secure status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.
6. The MER1 or MER2 bits can be cleared if no more bank erase is requested.

Mass erase

To perform a mass erase, follow the procedure below:

1. Check that no Flash memory operation is ongoing by checking the BSY bit in the *FLASH non-secure status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.
2. Check and clear all non-secure error programming flags due to a previous programming. If not, the PGSERR bit is set.
3. Set the MER1 bit and MER2 bits in the *FLASH non-secure control register (FLASH_NSCR)* or *FLASH secure control register (FLASH_SECCR)*.
4. Set the STRT bit in the *FLASH non-secure control register (FLASH_NSCR)* or *FLASH secure control register (FLASH_SECCR)*.
5. Wait for the BSY bit to be cleared in the *FLASH non-secure status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.
6. The MER1 and MER2 bit can be cleared if no more mass erase is requested.

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

To erase a page, a bank or to perform a mass erase, the software must have sufficient privilege (see Table 64 and Table 65).

7.3.7 Flash main memory programming sequences

The Flash memory is programmed 137 bits at a time (128-bit data + 9 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 PROGERR flag in the *FLASH non-secure status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.

It is only possible to program quad-word (4 x 32-bit data).

- Any attempt to write byte or half-word sets the SIZERR flag in the *FLASH non-secure status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.
- Any attempt to write a quad-word that is not aligned with a quad-word address sets the PGAERR flag in the *FLASH non-secure status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.

Flash programming

The Flash memory programming sequence is as follows:

1. Check that no Flash main memory operation is ongoing by checking the BSY bit in the *FLASH non-secure status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.
2. Check that the write buffer is empty by checking the WDW bit in the *FLASH non-secure status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.
3. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.

4. Set the PG bit in the *FLASH non-secure control register (FLASH_NSCR)* or *FLASH secure control register (FLASH_SECCR)*.
5. Perform the data write operation at the desired Flash memory address, or in the OTP area. Only a quad-word can be programmed and OTP can be only programmed in non-secure access:
   – Write a first word in an address aligned on a quad-word address. The WDW bits in the *FLASH non-secure status register (FLASH_NSSR)* and *FLASH secure status register (FLASH_SECSR)* are set to indicate that more data can be written in the write buffer.
   – Write the second, third and fourth word in the same quad-word.
6. The BSY bit gets set. WDW is reset automatically.
7. Wait until the BSY bit is cleared in the *FLASH non-secure status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*. The software must make sure that BSY is set or WDW is cleared before waiting for BSY to get cleared.
8. If the EOP flag is set in the *FLASH non-secure status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)* (meaning that the programming operation has succeeded and the EOPIE bit is set), it must be cleared by software.
9. Clear the PG bit in the *FLASH non-secure control register (FLASH_NSCR)* or *FLASH secure control register (FLASH_SECCR)* if there is no more programming request.

Note: When the Flash memory interface received a good sequence (a quad-word), programming is automatically launched and the BSY bits are set. The internal oscillator HSI16 (16 MHz) is enabled automatically when PG bit is set, and disabled automatically when PG bit is cleared, except if the HSI16 is previously enabled with HSION in RCC_CR register.

No option bytes modification nor erase request is allowed when the WDW bit is set.

Programming is possible only if the privileged and security attributes are respected. Refer to Section 7.7: Summary of Flash memory and Flash memory registers access control.

If the user needs to program only one word, the quad-word must be completed with the erase value 0xFFFF FFFF to launch automatically the programming.

ECC is calculated from the quad-word to program.

Flash burst programming (8 quad-words)

The Flash memory burst programming sequence is as follows:

1. Check that no Flash main memory operation is ongoing by checking the BSY bit in the *FLASH non-secure status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.
2. Check that the write buffer is empty by checking the WDW bit in the *FLASH non-secure status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.
3. Check and clear all error programming flags due to a previous programming. If not, PGSERR is set.
4. Set the BWR and PG bits in the *FLASH non-secure control register (FLASH_NSCR)* or *FLASH secure control register (FLASH_SECCR)*.
5. Perform the data write operation at the desired Flash memory address, or in the OTP area. Only 8 quad-words can be programmed:
   – Write a first 32-bit word in an address aligned on a 8*quad-word address (multiple of 0x80). The WDW bits in the *FLASH non-secure status register (FLASH_NSSR)*

and *FLASH secure status register (FLASH_SECSR)* are set to indicate that more data can be written in the write buffer.

– Write the 31 other 32-bit words consecutively.

6. The BSY bit gets set. WDW is reset automatically.
7. Wait until the BSY bit is cleared in the *FLASH non-secure status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*. The software must make sure that BSY is set or WDW is cleared before waiting for BSY to get cleared
8. If the EOP flag is set in the *FLASH non-secure status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)* (meaning that the programming operation has succeeded and the EOPIE bit is set), it has to be cleared by software.
9. Clear the BWR and PG bits in the *FLASH non-secure control register (FLASH_NSCR)* or *FLASH secure control register (FLASH_SECCR)* if there is no more programming request.

Note: When the Flash memory interface received a good sequence, programming is automatically launched and the BSY bits are set. The internal oscillator HSI16 (16 MHz) is enabled automatically when PG bit is set, and disabled automatically when PG bit is cleared, except if the HSI16 is previously enabled with HSION in RCC_CR register.

No option bytes modification nor erase request is allowed when the WDW bit is set.

Programming is possible only if the privileged and security attributes are respected. Refer to Section 7.7: Summary of Flash memory and Flash memory registers access control.

7.3.8 Flash memory endurance

Each Flash memory page can be written and erased 10 000 times. In addition, up to 256 Kbytes (32 pages) per bank feature an increased endurance of 100 kcycles, that can be used for data storage that usually needs more intensive cycling capability than code storage. Any Flash page can be chosen to be cycled more than 10 000 times (up to 100 000 times). It is the application responsibility to limit the size of the Flash area cycled more than 10 000 times to 256 Kbytes per bank.

7.3.9 Flash memory errors flags

Flash programming errors

Several kind of errors can be detected during secure and non-secure operations. In case of error, the Flash memory 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 operation.

- **PROGERR**: secure/non-secure programming error

  It is set when the 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, when the address is already partially programmed (contains 0 and 1)

- **SIZERR**: secure/non-secure size programming error

  Only 32-bit data can be written. SIZERR flag is set if a byte or a half-word is written.
- **PGAERR**: secure/non-secure alignment programming error

  It is set when the first word to be programmed is not aligned with a quad-word address, or the second, third or forth word does not belong to the same quad-word address.

  For burst programming, it is set when the first word to be programmed is not aligned on a 8 *quad-word address or if the following word writes are not done at consecutive 32-bit addresses.
- **PGSERR**: programming sequence error

  PGSERR is set if one of the following conditions occurs during a erase or program operation:
  - A data is written when PG is cleared.
  - A program operation is requested during erase: PG is set while MER1 or MER2 or PER is set.
  - In the erase sequence, PG is set while STRT is already set.
  - In the erase sequence, if STRT is set while MER1 and MER2 and PER are cleared.
  - If page and mass erase are requested at the same time, STRT and PER are set while MER1 or MER2 is set.
  - If an operation is started while the write buffer is waiting for the next data, STRT or OPTSTRT is set while WDW is already set.
  - If STRT and OPTSTRT are set at the same time.
  - A non-secure PGSERR is set if the non secure STRT bit is set by a secure access.
  - A secure PGSERR is set if PROGERR, SIZERR, PGAERR, WRPERR or PGSERR is already set due to a previous programming error.
  - A non-secure PGSERR is set if PROGERR, SIZERR, PGAERR, WRPERR, PGSERR or OPTWERR is already set due to a previous programming error.
- **WRPERR**: write protection error
  - Refer to *Table 59* to *Table 62* for all the conditions of WRPERR flag setting.
- **OPTWERR**: option bytes write error

  OPTWERR is set if when user option bytes are modified with an invalid configuration. It is set when attempting:
  - to program an invalid secure watermark-based area. Refer to *Table 50*
  - to set or clear the TZEN option bit when RDP is not at correct level (refer to *Rules for modifying specific option bytes*)
  - to clear the BOOT_LOCK option bit when RDP is not at correct level (refer to *Rules for modifying specific option bytes*)
  - 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 DUALBANK option bit while BOOT_LOCK and TZEN are set
  - to modify SECWM1Rx (resp. SECWM2Rx) while HDP1_ACCDIS (resp. HDP2_ACCDIS) is set
  - to modify the option bytes, except the SWAP_BANK option bit, when RDP is set to two

- to regress from RDP level 0.5 to RDP level 0
- to modify OEM1KEYRx while RDP level is 0.5 or 1 and OEM1LOCK bit is set
- to modify OEM2KEYRx while RDP level is 1 and OEM2LOCK bit is set
- to regress from RDP level 1 to RDP level 0 while OEM1LOCK bit is set and a wrong OEM1KEY is shifted through JTAG or SWD
- to regress from RDP level 1 to RDP level 0.5 while OEM2LOCK bit is set and a wrong OEM2KEY is shifted through JTAG or SWD
- to modify WRPxyR while its UNLOCK bit is cleared
- to set the UNLOCK bit in the WRPxyR when RDP is not at correct level (refer to *Rules for modifying specific option bytes*)

If an error occurs during a secure or non-secure program or erase operation, one of the following programming error flags is set:

- non-secure programming error flags: PROGERR, SIZERR, PGAERR, PGSERR, OPTWRERR or WRPERR is set in the *FLASH non-secure status register (FLASH_NSSR)*.

  If the non-secure error interrupt enable bit ERRIE is set in the *FLASH non-secure control register (FLASH_NSCR)*, an interrupt is generated and the operation error flag OPERR is set in the FLASH_NSSR register.
- Secure programming error flags: PROGERR, SIZERR, PGAERR, PGSERR or WRPERR is set in the *FLASH secure status register (FLASH_SECSR)*.

  If the secure error interrupt enable bit ERRIE is set in the *FLASH secure control register (FLASH_SECCR)*, an interrupt is generated and the operation error flag OPERR 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.

Any programming error flushes the write buffer.

7.3.10 Read-while-write (RWW)

The Flash memory is divided into two banks allowing read-while-write operations. This feature allows a read operation to be performed 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).

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).

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).

Note: Due to the Cortex-M33 unified C-Bus, user software must ensure to not stall C-Bus with multiple consecutive writes. It is recommended to wait for the BSY flag to be cleared before programming the next quad-word.

7.3.11 Power-down during Flash programming or erase operation

The contents of the Flash memory currently being accessed are not guaranteed if a power-down occurs during a Flash memory program or erase operation.

7.3.12 Reset during Flash programming or erase operation

The contents of the Flash memory currently being accessed are not guaranteed if a reset occurs during a Flash memory program or erase operation. The status of the Flash memory can be recovered from the *FLASH operation status register (FLASH_OPSR)* when a system reset occurs during a Flash memory program or erase operation.

It is the software responsibility to check the status of the Flash memory and to take corrective actions. This must be done after each system reset before any other programming or erase operation is performed.

The table below describes the corrective action to be taken according to the status provided in the CODE_OP field of the *FLASH operation status register (FLASH_OPSR)*.

Table 47. Flash operation interrupted by a system reset

CODE_OP	Operation interrupted	Address	Bank	System Flash	Corrective action
0x0	No operation	Reserved			None
0x1	Single write	ADDR_OP	BK_OP	SYSF_OP	Page erase and single write at same location
0x2	Burst write	ADDR_OP	BK_OP	SYSF_OP	Page erase and burst write at same location
0x3	Page erase	ADDR_OP	BK_OP	Reserved	Erase same page
0x4	Bank erase	Reserved	BK_OP	Reserved	Erase same bank
0x5	Mass erase	Reserved			Mass erase
0x6	Option change	Reserved			Option change
0x7	Reserved				

Note: For single and burst write, it is mandatory to perform a page erase because the current Flash memory locations may no longer be writable. Consequently, the remaining page content must be saved before page erase and restored afterwards.

For OTP write, it is not possible to perform a page erase. The OTP quad-word is lost.

For burst write, ADDR_OP gives the first address of burst. User must restart the same burst operation.

For page erase, ADDR_OP gives the first address of erased page.

7.4 FLASH option bytes

7.4.1 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 7.4.2: Option-byte programming*). The user option bytes are accessible through the Flash memory registers interface.

The table below describes the organization of all user option bytes available in the Flash memory interface registers.

Table 48. 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	IO_VDDIO2_HSLV	IO_VDD_HSLV	PA15_PUPEN	NBOOT0	NSWBOOT0	SRAM2_RST	SRAM2_ECC	SRAM3_ECC	BKPRAM_ECC	DUALBANK	SWAP_BANK	WWDG_SW	IWDG_STDBY	IWDG_STOP	IWDG_SW	SRAM1345_RST	NRST_SHDW	NRST_STDBY	NRST_STOP	Res.	BOR_LEV[2:0]	RDP										*Section 7.9.13*
NSBOOTADD0[24:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.		*Section 7.9.14*
NSBOOTADD1[24:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.		*Section 7.9.15*
SECBOOTADD0[24:0]																								Res.							BOOT_LOCK	*Section 7.9.16*
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 7.9.17*
HDP1EN	Res.	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.	*Section 7.9.18*
UNLOCK	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 7.9.19*
UNLOCK	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 7.9.20*
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 7.9.21*
HDP2EN	Res.	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.	*Section 7.9.22*
UNLOCK	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 7.9.23*
UNLOCK	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 7.9.24*
OEM1KEY[31:0]																																*Section 7.9.25*
OEM1KEY[63:32]																																*Section 7.9.26*
OEM2KEY[31:0]																																*Section 7.9.27*
OEM2KEY[63:32]																																*Section 7.9.28*

7.4.2 Option-byte programming

After reset, the options related to the operation bits OPTSTRT and OBL_LAUNCH in the FLASH_NSCR register, 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 register with the LOCK clearing sequence (refer to *Unlock the secure/non-secure Flash control registers*).
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 the LOCK bit in FLASH_NSCR *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 BSY 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_NSCR register.
5. Wait for the BSY bit to be cleared.
6. Set the OBL_LAUNCH option bit to start option bytes loading.

Note: If the OPTWERR or PGSERR error bit is set, the old option byte values are kept.

Option byte loading

After the BSY bit is cleared, all new options are updated into the Flash memory 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 reflect the options state of the system.

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 is only possible when RDP is changing from level 1 to level 0.

- BOOT_LOCK option bit
  - BOOT_LOCK has only effect when TZEN is set.
  - BOOT_LOCK can be set without any constraint.
  - Deactivation of BOOT_LOCK is only possible when RDP is level 0.
- SWAP_BANK option bit
  - It cannot be modified when BOOT_LOCK and TZEN option bits are set.
- SECBOOTADD0 option bytes
  - It cannot be modified when BOOT_LOCK option bit is set.
- DUALBANK option bit
  - It cannot be modified when BOOT_LOCK and TZEN option bits are set.
- SECWMxRy option bits
  - It is not possible to modify secure (SECWMx_PSTRT[6:0] and SECWMx_PEND[6:0] option bits), and HDP (HDPx_PEND[6:0] and HDPxEN option bits) area in bank x when the HDPx_ACCDIS bit is set.
- RDP option bits
  - Refer to *Device life cycle managed by readout protection (RDP) transitions*.
- WRPxyR option bits
  - These bits cannot be modified when their UNLOCK bit is cleared.
- UNLOCK option bits
  - These bits can be set only when regressing from RDP level 1 to level 0.

If the 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.

7.5 FLASH TrustZone security and privilege protections

7.5.1 TrustZone security protection

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

When TrustZone is active (TZEN = 1), the following additional security features are available:

- Secure watermark-based user options bytes defining secure and 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.

Activate TrustZone security

When the TrustZone is activated (TZEN is modified from 0 to 1), the secure watermark-based user options bytes are set to default secure state: all Flash memory is secure, and no HDP area, as shown in the table below.

Table 49. Default secure option bytes after TZEN activation

Secure watermark option bytes values after OBL when TZEN is activated (from 0 to 1)	Security attribute
SECWMx_PSTRT = 0 and SECWMx_PEND = 0x7F	All Flash memory secure
HDPxEN = 0 and HDPx_PEND = 0	No secure HDP area

Illegal access generation

A non-secure access to 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 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 TZIC_IER2 register.

Deactivate TrustZone security

Deactivation of TZEN (from 1 to 0) is only possible when the RDP is changing from level 1 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 (refer to *Section 7.5.2*)
- Block-based secure area (refer to *Section 7.5.4*)
- RDP level 0.5 (refer to *Section 7.6.2: Readout protection (RDP)*)
- Secure interrupts (refer to *Section 7.8: FLASH interrupts*)
- All secure registers are RAZ/WI

7.5.2 Watermark-based secure Flash memory area protection

When TrustZone security is active (TZEN = 1), a part of the Flash memory can be protected against non-secure read and write accesses. 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: 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 secure watermark1 register 1 (FLASH_SECWM1R1)* and *FLASH secure watermark2 register 1 (FLASH_SECWM2R1)*.

The SECWMx_PSTRT and SECWMx_PEND option bytes can only be modified by secure firmware when the HDPx_ACCDIS bit is reset. If the HDPx_ACCDIS bit is set, the SECWMx_PSTRT and SECWMx_PEND cannot be modified until next system reset.

Table 50. Secure watermark-based area

Secure watermark option bytes values (x = 1,2)	Secure watermark protection area
SECWMx_PSTRT > SECWMx_PEND	No secure area
SECWMx_PSTRT = SECWMx_PEND	One page defined by SECWMx_PSTRT is secure watermark-based protected
SECWMx_PSTRT < SECWMx_PEND	The area between SECWMx_PSTRT and SECWMx_PEND is secure watermark-based protected.

Caution: Switching a Flash memory area from secure to no-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.

7.5.3 Secure hide protection (HDP)

The secure HDP area is part of the Flash memory watermark-based secure area. Access to the hide protection area can be denied by setting the HDPx_ACCDIS bit in the *FLASH secure HDP control register (FLASH_SECHDPCR)*.

When the HDPx_ACCDIS bit is set, instruction fetch, data read, write and erase operations 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 HDPx_ACCDIS bit can be only cleared by a system reset.

Note: It is the software responsibility to take any appropriate action to protect the HDP code before resetting the HDPxEN bit such as erasing the HDP area and flushing the instruction cache.

One non-volatile secure hide protection (HDP) area per bank can be defined 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 on the fly by the secure firmware if the HDPx_ACCDIS bit is reset. If the HDPx_ACCDIS bit is set, the HDPxEN bit and secure watermark configuration cannot be modified until next system 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 secure watermark1 register 1 (FLASH_SECWM1R1)*, *FLASH secure watermark1 register 2 (FLASH_SECWM1R2)*, *FLASH secure watermark2 register 1 (FLASH_SECWM2R1)* and *FLASH secure watermark2 register 2 (FLASH_SECWM2R2)*.

For example, to protect by HDP from the address 0x0C00 4000 (included) to the address 0x0C00 5FFF (included):

- If the banks are not swapped, the option bytes registers must be programmed with:
  - SECWM1_PSTRT = 0x2
  - HDP1_PEND = 0x3
- If the two banks are swapped, the protection must apply to bank 2 and the option bytes registers must be programmed with:
  - SECWM2_PSTRT = 0x2
  - HDP2_PEND = 0x3

Note: For more details on the bank swapping mechanism, refer to Section 7.5.8.

If an invalid secure HDP area is defined as described in the table below, the OPTWERR flag error is set and option bytes modification is discarded.

Table 51. Secure hide protection

HDPx watermark option bytes values (x = 1,2)		Hide protection area
HDPxEN = 0	-	No secure HDP area
HDPxEN = 1	SECWMx_PSTRT ≤ HDPx_PEND ≤ SECWMx_PEND	The area between SECWMx_PSTRT and HDPx_PEND is secure HDP protected.
	Others	Invalid secure area. Hide protection area is defined outside the secure area.

The table below summarizes the possible secure and HPD protection area configurations.

Table 52. Secure and HDP protections

Secure and HDP watermark option byte values		Protections area
HDPxEN	Option bytes	
x	SECWMx_PSTRT > SECWMx_PEND	No secure area
0	SECWMx_PSTRT ≤ SECWMx_PEND	No secure HDP area Secure between SECWMx_PSTRT and SECWMx_PEND – If SECWMx_PSTRT = SECWMx_PEND, one page defined by SECWMx_PSTRT is secure protected.
1	SECWMx_PSTRT ≤ HDPx_PEND ≤ SECWMx_PEND	The area between SECWMx_PSTRT and HDPx_PEND is secure HDP protected. – If SECWMx_PSTRT = HDPx_PEND, one page defined by HDPx_PEND is secure HDP protected.
	Others	Invalid secure area HDP area is defined outside the secure area.

7.5.4 Block-based secure Flash memory area protection

Any page can be programmed on the fly as secure or non-secure using the block-based configuration registers. FLASH_SECBB1Rx (resp. FLASH_SECBB2Rx) registers are used to configure the security attribute for pages in bank 1 (resp. bank 2).

When the page security attribute, bit i in SECyBBRx, is set, the security attribute is the same as the secure watermark-based area. The secure page is only accessible by a secure access.

If the SECyBBi bit is set or reset for a page already included in a secure watermark-based area, the page keeps the watermark-based protection security attributes.

To modify a block-based page security attribution, the following actions are recommended:

- Check that no Flash memory operation is ongoing on the related page.
- Add an ISB instruction after modifying the page security attribute bit i in SECyBBRx.

Caution: Switching a page or memory block from secure to non-secure does not erase the content of the associated block. User secure software must perform the following needed operations before switching a block to non-secure attribute:

- Erase page content,
- Invalidate the instruction cache.

Note: For SECyBBRx bit i access control, refer to Table 66: SECyBBRx registers access when TrustZone is active (TZEN = 1).

7.5.5 Flash security attribute state

The Flash memory 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 INV bit in the FLASH_SECCR register.

The FLASHEN and FLASHSMEN bits security attributes in RCC follow the Flash memory security attribute. It is possible to override the Flash memory security attribute in RCC using the INV bit in the FLASH_SECCR register. A secure firmware setting this INV bit allows a non-secure firmware to disable the Flash memory clock when the Flash memory is in power down or when the MCU enters low-power modes.

Table 53. Flash security state

Secure area	INV bit	Flash security state
None	0	Non-secure
	1	Secure
Yes	0	Secure
	1	Non-secure

7.5.6 Block-based privileged Flash memory area protection

Any page can be programmed on the fly as privileged or unprivileged using the block-based configuration registers. FLASH_PRIVBB1Rx (resp. FLASH_PRIVBB2Rx) registers are used to configure the privilege attribute for pages in bank 1 (resp. bank 2).

When the page privilege attribute, bit i in PRIVyBBRx, is set, the page is only accessible by a privileged access. An unprivileged page is accessible by a privileged or unprivileged access.

To modify a block-based privilege attribution, the following actions are recommended:

- Check that no Flash operation is ongoing on the related page.
- Add an ISB instruction after modifying the page security attribute bit i in PRIVyBBRx.

Caution: Switching a page or memory block from privileged to unprivileged does not erase the content of the associated block.

Note: For PRIVyBBRx bit i access control, refer to Table 67: PRIVyBBRx registers access when TrustZone is active (TZEN = 1) and Table 68: PRIVyBBRx registers access when TrustZone is disabled (TZEN = 0).

7.5.7 Flash memory registers privileged and unprivileged modes

The Flash memory registers can be read and written by privileged and unprivileged accesses depending on the SPRIV and NSPRIV bits in *FLASH privilege configuration register (FLASH_PRIVCFGR)*, with the following rules:

- When the SPRIV (resp. NSPRIV) bit is reset, all secure (resp. non-secure) Flash memory registers can be read and written by both privileged or unprivileged access.
- When the SPRIV (resp. NSPRIV) bit is set, all secure (resp. non-secure) Flash memory registers can be read and written by privileged access only. Unprivileged access to a privileged registers is RAZ/WI.

Table 63: Flash registers access summarizes the Flash memory registers access control.

7.5.8 Flash memory bank attributes in case of bank swap

The SWAP_BANK option bit modifies the address of each bank in the memory map. When SWAP_BANK is reset, the Flash memory bank 1 is at the lower address range. When SWAP_BANK is set, the Flash memory bank 1 is at the higher address range.

Flash memory bank attributes follow their bank so there is no need to modify the following registers when swapping banks:

- FLASH secure watermark y register x FLASH_SECWMyRx
- FLASH write protection x area y FLASH_WRPxyR (refer to *Section 7.6.1*)
- FLASH secure block based bank y register x FLASH_SECyBBRx
- FLASH privilege block based bank y register x FLASH_PRIVyBBRx
- PDREQx bits in the *FLASH access control register (FLASH_ACR)*
- PDx bits in the *FLASH non-secure status register (FLASH_NSSR)*

Note: The BK_ECC bit in the FLASH ECC register (FLASH_ECCR) always refers to bank 1 (resp. bank 2) when it is low (resp. high), whatever SWAP_BANK value.

The BK_OP bit in the FLASH operation status register (FLASH_OPSR) always refers to bank 1 (resp. bank 2) when it is low (resp. high), whatever SWAP_BANK value.

The figures below show how security attributes and protections behave in case of bank swap.

Figure 20. Flash memory security attributes and protections in case of no bank swap (SWAP_BANK = 0)

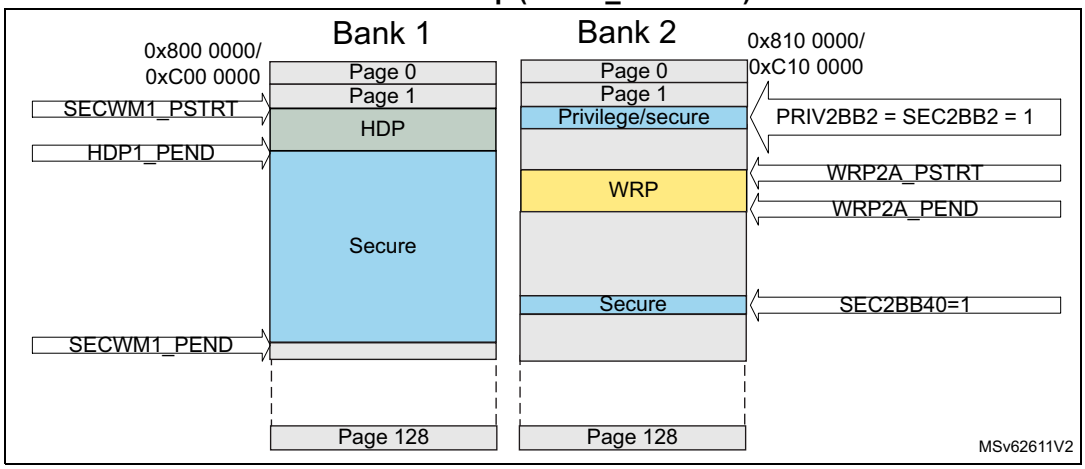
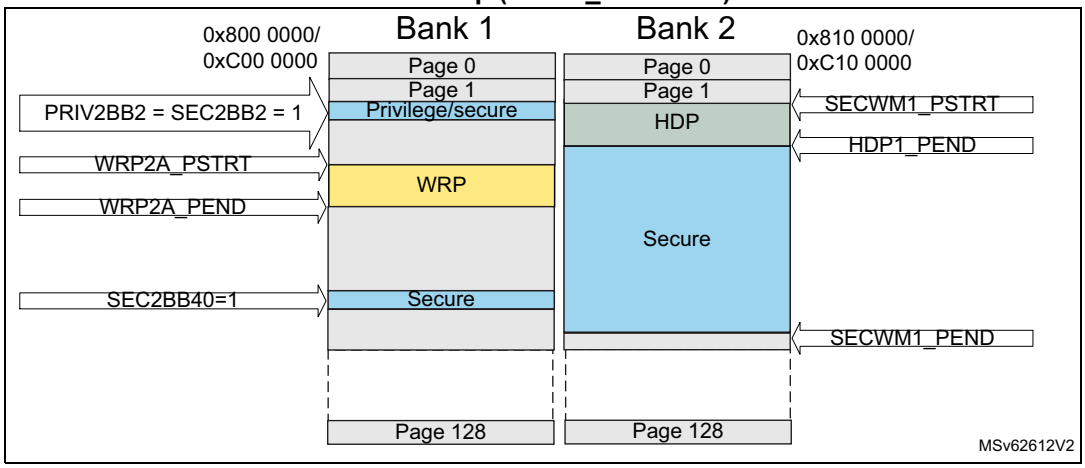


Figure 21. Flash memory security attributes and protections in case of bank swap (SWAP_BANK = 1)



7.6 FLASH memory protection

The Flash memory interface implements the following protection mechanisms:

- write protection (WRP)
- readout protection (RDP)
- additional secure protections when TrustZone is active (refer to *Section 7.5*)
  - up to two secure watermark-based non-volatile areas
  - up to two secure hide protection areas
  - secure block-based volatile areas with page granularity
- privileged block-based volatile areas with page granularity (refer to *Section 7.5.6*)

7.6.1 Write protection (WRP)

The user area in Flash memory can be protected against unwanted write operations. Two write-protected (WRP) areas can be defined in each bank, with page 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 WRP1 area A address register (FLASH_WRP1AR)*, *FLASH WRP1 area B address register (FLASH_WRP1BR)*, *FLASH WPR2 area A address register (FLASH_WRP2AR)* and *FLASH WPR2 area B address register (FLASH_WRP2BR)*.

The bank “x” WRP “y” area (x = 1,2 and y = A,B) is defined as follows:

- from the address: bank “x” base address + [WRPxy_PSTRT x 0x2000] (included)
- to the address: bank “x” base address + [(WRPxy_PEND+1) x 0x2000] (excluded)

For example, to protect by WRP from the address 0x0806 2000 (included) to the address 0x0807 3FFF (included):

- If the banks are not swapped, FLASH_WRP1AR register must be programmed with:
  - WRP1A_PSTRT = 0x31
  - WRP1A_PEND = 0x39

  WRP1B_PSTRT and WRP1B_PEND 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_PSTRT = 0x31
  - WRP2A_PEND = 0x39

  WRP2B_PSTRT and WRP2B_PEND in FLASH_WRP2BR can be used instead (area “B in bank 2).

Note: For more details on the bank swapping mechanism, refer to Section 7.5.8: Flash memory bank attributes in case of bank swap.

When WRP is active, protected Flash memory pages 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 (WRPERR) is set in the FLASH_NSSR or FLASH_SECSR register. This flag is also set for any write access to the following:

- system Flash memory
- OTP area

Note: When the memory readout protection level 1 is selected (RDP level = 1), it is not possible to program or erase the 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 memory, even if WRP is not activated.

When the memory readout protection level 0.5 is selected (RDP level = 0.5), it is not possible to program or erase the Flash secure memory if the CPU debug features are connected (JTAG or single wire), even if WRP is not activated.

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

Table 54. WRP protection

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

Write protection lock

Each WRP area can be independently locked by writing 0 to the UNLOCK bit in the *FLASH WRP1 area A address register (FLASH_WRP1AR)*, *FLASH WRP1 area B address register (FLASH_WRP1BR)*, *FLASH WPR2 area A address register (FLASH_WRP2AR)* or *FLASH WPR2 area B address register (FLASH_WRP2BR)*. Once a WRP area is locked, it is not possible to modify its settings. In order to unlock a WRP area, a regression to RDP level 0 must be launched.

In order to make the WRP area immutable and act as a ROM, the following actions are needed:

- If RDP level is 0, 0.5 or 1, provision a OEM1KEY in order to prevent a regression to RDP level 0 for users not knowing the key.
- If RDP level is 2, either provision a OEM1KEY (refer to first bullet) or do not provision a OEM2KEY (preventing regression from level 2 to level 1).

For more information on RDP regressions, refer to *Device life cycle managed by readout protection (RDP) transitions*.

7.6.2 Readout protection (RDP)

The readout protection protects the Flash main memory, the option bytes, the backup registers, the backup RAM and the SRAMs. In order to reach the best protection level, it is recommended to activate TrustZone and to set the RDP Level 2 with password authentication regression enabled (refer to *Readout protection levels when TrustZone is enabled*).

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 55. 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

  When the readout protection level 1 is set:
  - **User mode:** code executing in user mode (**boot Flash**) can access the 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 the microcontroller boots from RAM or system memory, the Flash main memory, the backup registers, the backup RAM and the SRAM2 are totally inaccessible; any read or write access to the Flash main memory 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

  When the readout protection level 2 is set:
  - The protection level 1 is guaranteed.
  - All debug features are disabled:

    . if OEM2 key has not been provided, JTAG and SWD are definitively disabled.

    . if OEM2 key has been provided under a lower RDP protection, JTAG and SWD remain enabled under reset only to interface with DBGMCU_SR, DBGMCU_DBG_AUTH_HOST and DBGMCU_DBG_AUTH_DEVICE registers to obtain device identification and provide OEM2 key to request RDP regression.
  - The boot from SRAM (boot RAM mode) and the boot from system memory (bootloader mode) are no longer available.
  - Only boot from main Flash memory is possible; all operations are allowed on the Flash main memory. Read, erase and program accesses to the 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 unless an OEM2 key has been provisioned (refer to *OEM2 RDP lock mechanism*).

Note: The debug feature is also disabled under reset.

STMicroelectronics is not able to perform analysis on defective parts on which the level 2 protection has been set. Regress parts to RDP level 1 before returning them for analysis (refer to OEM2 RDP lock mechanism).

Table 56. Access status versus protection level and execution modes when TZEN = 0

Area	RDP level	User execution (boot from Flash)			Debug/boot from RAM/ bootloader(1)		
		Read	Write	Erase	Read	Write	Erase
Flash main memory	1	Yes	Yes	Yes	No	No	No(4)
	2	Yes	Yes	Yes	N/A	N/A	N/A
System memory (2)	1	Yes	No	No	Yes	No	No
	2	Yes	No	No	N/A	N/A	N/A
Option bytes(3)	1	Yes	Yes(4)	N/A	Yes	Yes(4)	N/A
	2	Yes	No(5)	N/A	N/A	N/A	N/A
OTP	1	Yes	Yes(6)	N/A	Yes	Yes(6)	N/A
	2	Yes	Yes(6)	N/A	N/A	N/A	N/A
Backup registers	1	Yes	Yes	N/A	No	No	N/A(7)
	2	Yes	Yes	N/A	N/A	N/A	N/A
SRAM2/backup RAM	1	Yes	Yes	N/A	No	No	N/A(8)
	2	Yes	Yes	N/A	N/A	N/A	N/A
OTFDEC regions (OCTOSPI)	1	Yes	Yes	Yes	No	Yes	Yes(9)
	2	Yes	Yes	Yes	N/A	N/A	N/A

1. When the protection level 2 is active, the debug port, the boot from RAM and the boot from system memory are disabled.
2. The system memory is only read-accessible, whatever the protection level (0, 1 or 2) and execution mode.
3. Option bytes are only accessible through the Flash registers interface and OPTSTRT bit.
4. The Flash main memory is erased when the RDP option byte changes from level 1 to level 0.
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 57. Flash memory readout protection status (TZEN = 1)

RDP byte value	Readout protection level
0xAA	Level 0
0x55	Level 0.5

Table 57. Flash memory readout protection status (TZEN = 1) (continued)

RDP byte value	Readout protection level
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.

  – **RSS mode**: 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 the 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

  When the readout protection level 1 is set:

  – **User mode:** code executing in user mode (**boot Flash**) can access the 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, the backup RAM and the SRAM2 are totally inaccessible; any read or write access to the Flash main memory 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 RSS code.

  – **Boot RAM mode**: boot from SRAM is not possible.

- Level 2: no debug

  When the readout protection level 2 is set:

  – The protection level 1 is guaranteed.

  – All debug features are disabled

  . if OEM2 key has not been provided, JTAG and SWD are definitively disabled.

  . if OEM2 key has been provided under a lower RDP protection, JTAG and SWD remain enabled under reset only to interface with DBGMCU_SR, DBGMCU_DBG_AUTH_HOST and DBGMCU_DBG_AUTH_DEVICE registers to obtain device identification and provide OEM2 key to request RDP regression.

  – The boot from SRAM (boot RAM mode) and the boot from system memory (boot loader mode) are no longer available.

  – Boot from RSS is possible.

  – When booting from main 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 unless an OEM2 key has been provisioned (refer to *OEM2 RDP lock mechanism*).

Note: The debug feature is also disabled under reset.

STMicroelectronics is not able to perform analysis on defective parts on which the level 2 protection has been set. Regress parts to RDP level 1 before returning them for analysis (refer to OEM2 RDP lock mechanism).

Table 58. 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(5)
	2	Yes	Yes	Yes	N/A	N/A	N/A
System memory (3)	0.5	Yes	No	No	Yes	No	No
	1	Yes	No	No	Yes	No	No
	2	Yes	No	No	N/A	N/A	N/A
Option bytes(4)	0.5	Yes	Yes (5)	N/A	Yes	Yes (5)	N/A
	1	Yes	Yes(5)	N/A	Yes	Yes(5)	N/A
	2	Yes	No(6)	N/A	N/A	N/A	N/A
OTP	0.5	Yes	Yes(7)	N/A	Yes	Yes(7)	N/A
	1	Yes	Yes(7)	N/A	Yes	Yes(7)	N/A
	2	Yes	Yes(7)	N/A	N/A	N/A	N/A

Table 58. Access status versus protection level and execution modes when TZEN = 1 (continued)

Area	RDP level	User execution (boot from Flash)			Debug/bootloader(1)		
		Read	Write	Erase	Read	Write	Erase
Backup registers	0.5	Yes	Yes	N/A	Yes(2)	Yes(2)	N/A(8)
	1	Yes	Yes	N/A	No	No	N/A(8)
	2	Yes	Yes	N/A	N/A	N/A	N/A
SRAM2/ backup RAM	0.5	Yes	Yes	N/A	Yes(2)	Yes(2)	N/A(9)
	1	Yes	Yes	N/A	No	No	N/A(9)
	2	Yes	Yes	N/A	N/A	N/A	N/A
OTFDEC regions (OCTOSPI)	0.5	Yes	Yes	Yes	No	Yes	Yes(10)
	1	Yes	Yes	Yes	No	Yes	Yes(10)
	2	Yes	Yes	Yes	N/A	N/A	N/A

1. When the protection level 2 is active, the debug port and the bootloader mode are disabled.
2. Depends on TrustZone security access rights.
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 OPTSTRT bit.
5. The Flash main memory is erased when the RDP option byte regresses from level 1 to level 0.
6. SWAP_BANK option bit can be modified.
7. OTP can only be written once.
8. The backup registers are erased when RDP changes from level 1 to level 0 and when RDP changes from level 1 to level 0.5.
9. All SRAMs are erased when RDP changes from level 1 to level 0 and when RDP changes from level 1 to level 0.5.
10. The OTFDEC keys are erased when the RDP option byte changes from level 1 to level 0 and when RDP changes from level 1 to level 0.5.

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 from level 1. Once in level 2, it is no longer possible to modify the readout protection level unless an OEM2 key is provisioned (refer to *OEM2 RDP lock mechanism*).

When the RDP is reprogrammed to the value 0xAA to move from level 1 to level 0, a mass erase of the Flash main memory and all SRAMs is performed. The backup registers, the OTFDEC keys, ICACHE, DCACHE, and PKA SRAM are also erased. The OTP area is not erased.

At RDP level 0.5, it is not possible to request RDP level 0. Instead, a RDP increase to level 1 followed by a RDP regression to level 0 is required.

When the RDP is programmed to the value 0x55 to move from level 1 to level 0.5, a partial mass erase of the 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, ICACHE, DCACHE, PKA SRAM, 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.

Note: Full mass erase is performed only when level 1 is active and level 0 requested. When the protection level is increased (0 to 0.5, 0 to 1, 0.5 to 1, 1 to 2, 0 to 2 or 0.5 to 2), there is no mass erase.

To validate the readout protection level change, the option bytes must be reloaded through the OBL_LAUNCH bit in FLASH non-secure control register (FLASH_NSCR).

Before launching a RDP regression, the software must invalidate the ICACHE and wait for the BUSYF bit to get cleared.

Figure 22. RDP level transition scheme when TrustZone is disabled (TZEN = 0)

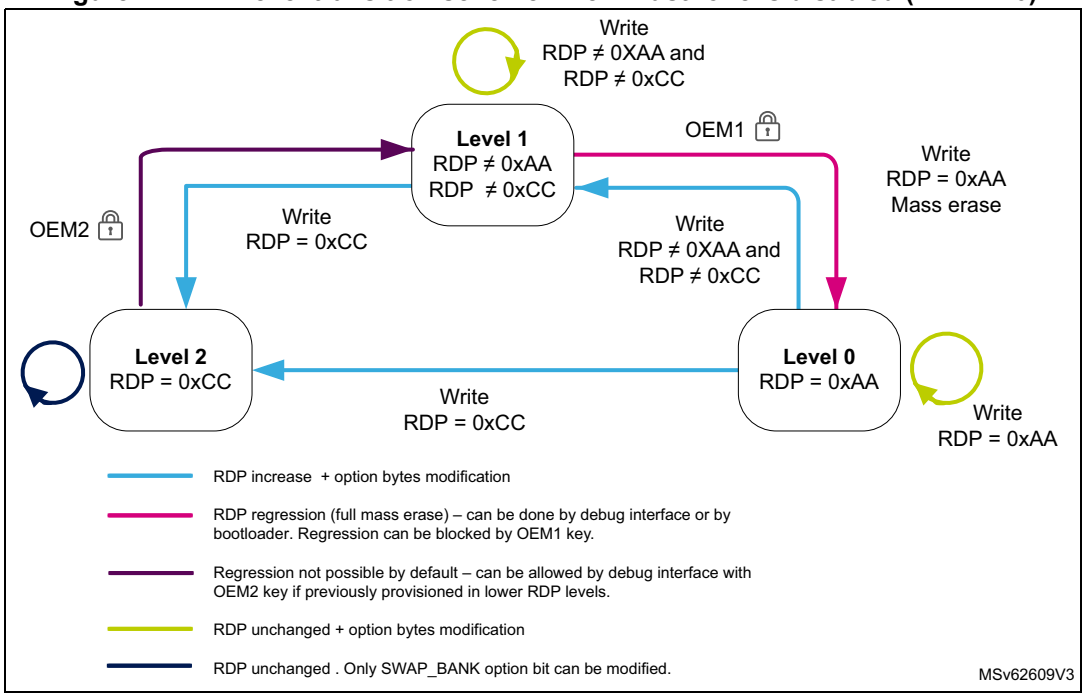
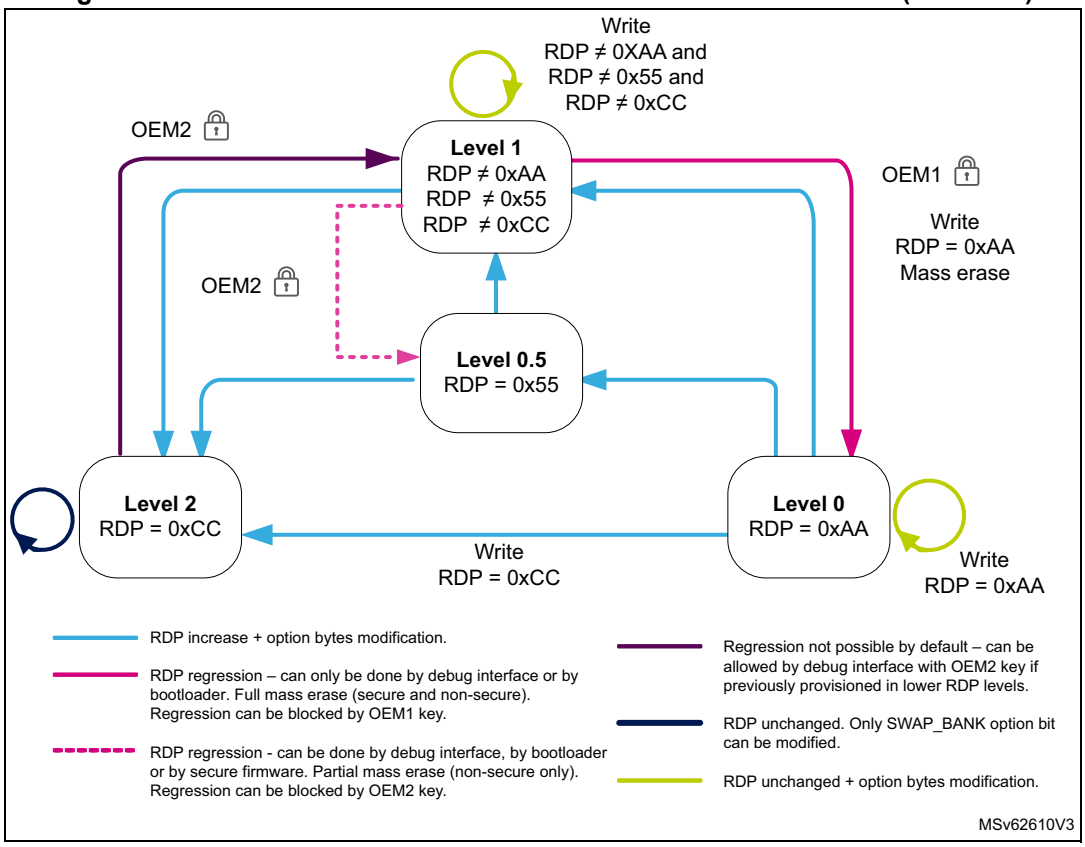


Figure 23. RDP level transition scheme when TrustZone is enabled (TZEN = 1)



OEM1/OEM2 lock activation

Two 64-bit keys (OEM1KEY and OEM2KEY) can be defined in order to lock the RDP regression. Each 64-bit key is coded on two registers *FLASH OEM1 key register 1 (FLASH_OEM1KEYR1)* (resp. *FLASH OEM2 key register 1 (FLASH_OEM2KEYR1)*) and *FLASH OEM1 key register 2 (FLASH_OEM1KEYR2)* resp. *FLASH OEM2 key register 2 (FLASH_OEM2KEYR2)*. OEM1KEY and OEM2KEY cannot be read through these registers. They are read as zero.

OEM1KEYcan be modified:

- in readout protection level 0
- in readout protection level 0.5 or 1 if OEM1LOCK = 0 in *FLASH non-secure status register (FLASH_NSSR)*

OEM2KEYcan be modified:

- in readout protection level 0 or 0.5
- in readout protection level 1 if OEM2LOCK = 0 in *FLASH non-secure status register (FLASH_NSSR)*

When attempting to modify the *FLASH OEM1 key register 1 (FLASH_OEM1KEYR1)*, *FLASH OEM1 key register 2 (FLASH_OEM1KEYR2)* or *FLASH OEM2 key register 1 (FLASH_OEM2KEYR1)*, *FLASH OEM2 key register 2 (FLASH_OEM2KEYR2)* without following these rules, the user option modification is not done and the OPTWERR bit is set.

In order to activate OEM1 lock mechanism, the following steps are needed:

- Check that the OEM1LOCK bit is not set or that the readout protection is at level 0.
- Write a 64-bit key in *FLASH OEM1 key register 1 (FLASH_OEM1KEYR1)* and *FLASH OEM1 key register 2 (FLASH_OEM1KEYR2)*.
- Launch option modification by setting the OPTSTRT bit in the *FLASH non-secure control register (FLASH_NSCR)*.
- Wait for the BSY bit to be cleared and check that OPTWERR is not set.
- Set the OBL_LAUNCH option bit to start option bytes loading or perform a power-on reset.
- Check that OEM1LOCK is set.

In order to activate OEM2 lock mechanism, the following steps are needed:

- Check that the OEM2LOCK bit is not set or that the readout protection is at level 0 or 0.5.
- Write a 64-bit key in *FLASH OEM2 key register 1 (FLASH_OEM2KEYR1)* and *FLASH OEM2 key register 2 (FLASH_OEM2KEYR2)*.
- Launch option modification by setting the OPTSTRT bit in the *FLASH non-secure control register (FLASH_NSCR)*.
- Wait for the BSY bit to be cleared and check that OPTWERR is not set.
- Set the OBL_LAUNCH option bit to start option bytes loading or perform a power-on reset.
- Check that OEM2LOCK is set.

Note: The OEM1KEY and OEM2KEY must not contain only 1 or only 0.

OEM1 RDP lock mechanism

The OEM1 RDP lock mechanism is active when the OEM1LOCK bit is set. It blocks the RDP level 1 to RDP level 0 regression.

In order to regress from RDP level 1 to RDP level 0, the following unlock sequence must be applied:

- Shift OEM1KEY[31:0] then OEM1KEY[63:32] through JTAG or SWD in the DBGMCU_DBG_AUTH_HOST register.
- If this key matches the OEM1KEY value, the RDP regression can be launched by setting the OPTSTRT bit.
- If the key does not match the OEM1KEY value, the RDP regression and any access to the Flash memory are blocked until a next power-on reset.

Attempting to regress from RDP level 1 to RDP level 0 without following this sequence sets the OPTWERR option bit and the option bytes remain unchanged.

When the lock mechanism is not activated (OEM1LOCK =0), the regression from RDP level 1 to RDP level 0 is always granted.

OEM2 RDP lock mechanism

The OEM2 RDP lock mechanism is active when the OEM2LOCK bit is set. It allows the following actions:

- Block RDP level 1 to RDP level 0.5 regression.
- Authorize RDP level 2 to RDP level 1 regression.

In order to regress from RDP level 1 to RDP level 0.5, the following unlock sequence must be applied:

- Shift OEM2KEY[31:0] then OEM2KEY[63:32] through JTAG or SWD under reset in the DBGMCU_DBG_AUTH_HOST register.
- If this key matches the OEM2KEY value, the RDP regression can be launched by setting the OPTSTRT bit.
- If the key does not match the OEM2KEY value, the RDP regression and any access to the Flash memory are blocked until a next power-on reset.

In order to regress from RDP level 2 to RDP level 1, the following unlock sequence must be applied:

- Shift OEM2KEY[31:0] then OEM2KEY[63:32] through JTAG or SWD under reset in the DBGMCU_DBG_AUTH_HOST register.
- If this key matches the OEM2KEY value:
  - the RDP regression is launched by hardware (it is not possible to execute instructions when the key is matching).
  - apply a power-on reset (cycle V_{DD} power supply OFF and ON).
- if the key does not match the OEM2KEY value, the RDP regression and any access to the Flash memory are blocked until a next power-on reset.

Attempting to regress from RDP level 2 to RDP level 1 without following these sequences, leaves option bytes unchanged.

Attempting to regress from RDP level 1 to RDP level 0.5 without following these sequences, sets the OPTWERR option bit and the option bytes remain unchanged.

When the lock mechanism is not activated (OEM2LOCK =0), the following happens:

- The regression from RDP level 1 to RDP level 0.5 is always granted.
- The regression from RDP level 2 to RDP level 1 is never granted. 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.

7.7 Summary of Flash memory and Flash memory registers access control

The tables below summarize all the Flash memory and registers accesses status versus RDP level, WRP and HDP protections.

Table 59. Flash memory 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 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="3">Bus error</td></tr>
<tr><td>Read</td><td>RAZ, Flash illegal access event</td></tr>
<tr><td>Write</td><td rowspan="2">WI, secure WRPERR flag set, Flash illegal access event</td><td rowspan="2">WI, secure WRPERR flag set</td><td rowspan="2">No WRP: OK
WRP pages: WI and secure WRPERR flag set</td></tr>
<tr><td>Page erase</td><td>WI, secure WRPERR 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>No WRP: OK
WRP pages: WI and non secure WRPERR flag set</td><td colspan="2" rowspan="2">WI, non secure WRPERR flag set,
Flash illegal access event</td><td rowspan="2">WI, non secure WRPERR flag set</td></tr>
<tr><td>Page erase</td></tr>
</table>

1. RDP level 1 no intrusion = when booting from user Flash memory and no debug access.
2. RDP level 1 with intrusion = when debug access detected.
3. Others refers to the other Flash memory secure configurations than the one described for HDP protections. Example: Flash memory secure and HDP area enabled but ACCDIS = 0.

Table 60. Flash memory access versus RDP level when TrustZone is disabled (TZEN = 0)

<table>
<tr><th>Access type</th><th>RDP level 0, RDP leve -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="2">Bus error</td></tr>
<tr><td>Read</td></tr>
<tr><td>Write</td><td rowspan="2">No WRP: OK
WRP pages: WI and non secure WRPERR flag set</td><td rowspan="2">WI and non secure WRPERR flag set</td></tr>
<tr><td>Erase</td></tr>
</table>

1. RDP Level 1 no intrusion = when booting from user Flash memory and no debug access.
2. RDP Level 1 with intrusion = when booting from RAM or system memory or debug access detected.

Table 61. Flash memory 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 memory	Non-secure or secure Flash memory
			HDP area (HDPxEN=1 and ACCDIS = 1)	Others(3)		
Secure	Bank or mass erase	WI, secure WRPERR flag set, Flash memory illegal access event	WI, secure WRPERR flag set	No WRP: OK WRP pages: WI and secure WRPERR flag set	WI, secure WRPERR flag set, Flash memory illegal access event	WI, secure WRPERR flag set
Non-secure	Bank or mass erase	No WRP: OK WRP pages: WI and non-secure WRPERR flag set	WI, non-secure WRPERR flag set, Flash memory illegal access event			WI, non secure WRPERR flag set

1. RDP Level 1 no intrusion = when booting from user Flash memory and no debug access.
2. RDP Level 1 with intrusion = when debug access detected.
3. Others refers to the other Flash memory secure configurations than the one described for HDP protections. Example: Flash memory secure and HDP area enabled but ACCDIS = 0.

Table 62. Flash system memory, OTP and RSS accesses(1)

Access type		System memory (bootloader)	OTP	RSS
Secure (TZEN = 1)	Fetch	Bus error		RAZ
	Read	RAZ, Flash memory register illegal access event	OK	
	Write	WI, secure WRPERR flag set, Flash memory illegal access event		WI, secure WRPERR flag set
Non-secure (TZEN = 0 or TZEN = 1)	Fetch	OK	Bus error	Bus error
	Read		OK	RAZ(2)
	Write	WI and non secure WRPERR flag set	OK if not virgin: WI, non secure PROGERR flag set	WI, non secure WRPERR flag set

1. Valid for all RDP levels.
2. Flash memory illegal access event is generated when TZEN = 1.

Table 63. Flash registers access(1)

Access type			Non-secure register		Secure register	
			NSPRIV = 1	NSPRIV = 0	SPRIV = 1	SPRIV = 0
Fetch	Secure/ non-secure	Privileged/ unprivileged	Bus error			
Read/ Write	Secure(2)	Privileged	OK			
		Unprivileged	RAZ/WI	OK	RAZ/WI	OK
	Non-secure(3)	Privileged	OK		RAZ/WI and a Flash memory register illegal access event(4)	
		Unprivileged	RAZ/WI	OK		

1. Except SECyBBRx, PRIVyBBRx and PRIVCFGR registers.
2. Secure access is only valid when TrustZone is active (TZEN = 1).
3. Non-secure access are valid when TrustZone is active or disabled.
4. Flash register illegal access event is only generated when TZEN = 1.

Table 64. Flash page access versus privilege mode(1)

Access type		Unprivileged page	Privileged page
Fetch, Read/Write, Page erase	Privileged	OK	
Fetch, Read	Unprivileged	OK	RAZ
Write, Page erase	Unprivileged		WI, secure or non secure WRPERR flag set

1. When TZEN = 1, access must be granted by security firewall before privilege is considered.

Table 65. Flash mass erase versus privilege mode(1)

Access type		Unprivileged Flash memory	Privileged Flash memory	Mix unprivileged and privileged Flash memory
Mass erase	Privileged	OK		
Mass erase	Unprivileged	OK	WI, secure or non-secure WRPERR flag set	

1. When TZEN = 1, access must be granted by security firewall before privilege is considered.

Table 66. SECyBBRx registers access when TrustZone is active (TZEN = 1)

Access type			Bit i in PRIVyBBRx	Bit i in SECyBBRx
Fetch	Secure/non-secure	Privileged/unprivileged	-	Bus error
Read	Secure/non-secure	Privileged/unprivileged	-	OK

Table 66. SECyBBRx registers access when TrustZone is active (TZEN = 1) (continued)

Access type			Bit i in PRIVyBBRx	Bit i in SECyBBRx
Write	Secure	Privileged	-	OK
		Unprivileged	0	OK for bit i
		Unprivileged	1	WI for bit i
	Non-secure	Privileged/unprivileged	-	WI and a Flash memory register illegal access event

Table 67. PRIVyBBRx registers access when TrustZone is active (TZEN = 1)

Access type			Page secure state (watermark or blocked based)	Bit i in PRIVyBBRx
Fetch	Privileged/unprivileged	Secure/non-secure	-	Bus error
Read	Privileged/unprivileged	Secure/non-secure	-	OK for all bits
Write	Privileged	Secured	-	OK for all bits
		Non-secure	Non-secure	OK for bit i
		Non-secure	Secure	WI for bit i
	Unprivileged	Secured/non-secure	-	WI for all bits

Table 68. PRIVyBBRx registers access when TrustZone is disabled (TZEN = 0)

Access type		PRIVyBBRx
Fetch	Privileged/unprivileged	Bus error
Read	Privileged/unprivileged	OK
Write	Privileged	OK
	Unprivileged	WI

7.8 FLASH interrupts

Table 69. Flash interrupt requests

Interrupt vector	Interrupt event	Event flag	Event flag/interrupt clearing method	Interrupt enable control bit	Exit Sleep mode	Exit Stop and Standby modes
FLASH_S	Secure end of operation	Secure EOP(1)	Write secure EOP = 1	Secure EOPIE	Yes	No
	Secure operation error	Secure OPERR(2)	Write secure OPERR = 1	Secure ERRIE	Yes	No

Table 69. Flash interrupt requests (continued)

Interrupt vector	Interrupt event	Event flag	Event flag/interrupt clearing method	Interrupt enable control bit	Exit Sleep mode	Exit Stop and Standby modes
FLASH	Non-secure end of operation	Non-secure EOP(1)	Write non-secure EOP = 1	Non-secure EOPIE	Yes	No
	Non-secure operation error	Non-secure OPERR(2)	Write non-secure OPERR = 1	Non-secure ERRIE	Yes	No
	ECC correction	ECCC	Write ECCC=1	ECCCIE	Yes	No

1. Secure EOP (resp. non-secure EOP) is set only if secure EOPIE (resp. non-secure EOPIE) is set.
2. Secure OPERR (resp. non-secure OPERR) is set only if secure ERRIE (resp. non-secure ERRIE) is set.

7.9 FLASH registers

7.9.1 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 unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x00

Reset value: 0x0000 0000

Access: no wait state when no Flash memory read is ongoing; 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.	SLEEP_PD	PDREQ2	PDREQ1	LPM	Res.	Res.	PRFTEN	Res.	Res.	Res.	Res.	LATENCY[3:0]			
	rw	rs	rs	rw			rw					rw	rw	rw	rw

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

Bit 14 **SLEEP_PD**: Flash memory power-down mode during Sleep mode

This bit determines whether the Flash memory is in power-down mode or Idle mode when the device is in Sleep mode.

0: Flash in Idle mode during Sleep mode

1: Flash in power-down mode during Sleep mode

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

Bit 13 **PDREQ2**: Bank 2 power-down mode request
This bit is write-protected with FLASH_PDKEY2R. This bit requests bank 2 to enter power-down mode. When bank 2 enters power-down mode, this bit is cleared by hardware and the PDKEY2R is locked.
0: No request for bank 2 to enter power-down mode
1: Bank 2 requested to enter power-down mode

Bit 12 **PDREQ1**: Bank 1 power-down mode request
This bit is write-protected with FLASH_PDKEY1R. This bit requests bank 1 to enter power-down mode. When bank 1 enters power-down mode, this bit is cleared by hardware and the PDKEY1R is locked.
0: No request for bank 1 to enter power-down mode
1: Bank 1 requested to enter power-down mode

Bit 11 **LPM**: Low-power read mode
This bit puts the Flash memory in low-power read mode.
0: Flash not in low-power read mode
1: Flash in low-power read mode

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **PRFTEN**: Prefetch enable
This bit enables the prefetch buffer in the embedded Flash memory.
0: Prefetch disabled
1: Prefetch enabled

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

Bits 3:0 **LATENCY[3:0]**: Latency
These bits represent the ratio between the HCLK (AHB clock) period and the Flash memory access time.
0000: Zero wait state
0001: One wait state
0010: Two wait states
...
1111: Fifteen wait states

7.9.2 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 unprivileged access when NSPRIV =1 in the FLASH_PRIVCFGR register.

Address offset: 0x08

Reset value: 0x0000 0000

Access: one wait state; word access

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

Bits 31:0 **NSKEY[31:0]**: Flash memory non-secure key

The following values must be written consecutively to unlock the FLASH_NSCR register, allowing the Flash memory non-secure programming/erasing operations:
KEY1: 0x4567 0123
KEY2: 0xCDEF 89AB

7.9.3 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 unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x0C

Reset value: 0x0000 0000

Access: one wait state; word access

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

Bits 31:0 **SECKEY[31:0]**: Flash memory secure key

The following values must be written consecutively to unlock the FLASH_SECCR register, allowing the Flash memory secure programming/erasing operations:
KEY1: 0x4567 0123
KEY2: 0xCDEF 89AB

7.9.4 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 unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x10

Reset value: 0x0000 0000

Access: one wait state; word access

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

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

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

7.9.5 FLASH bank 1 power-down key register (FLASH_PDKEY1R)

This register is non-secure. It can be read and written by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x18

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
PDKEY1[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
PDKEY1[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **PDKEY1[31:0]**: Bank 1 power-down key

The following values must be written consecutively to unlock the PDREQ1 bit in FLASH_ACR:
PDKEY1_1: 0x0415 2637
PDKEY1_2: 0xFAFB FCFD

7.9.6 FLASH bank 2 power-down key register (FLASH_PDKEY2R)

This register is non-secure. It can be read and written by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x1C

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
PDKEY2[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
PDKEY2[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **PDKEY2[31:0]**: Bank 2 power-down key

The following values must be written consecutively to unlock the PDREQ2 bit in FLASH_ACR:
PDKEY2_1: 0x4051 6273
PDKEY2_2: 0xAFBF CFDF

7.9.7 FLASH non-secure 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 unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x20

Reset value: 0x000X 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.	PD2	PD1	OEM2LOCK	OEM1LOCK	WDW	BSY
										r	r	r	r	r	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.	PGSERR	SIZERR	PGAERR	WRPERR	PROGERR	Res.	OPERR	EOP
		rc_w1						rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **PD2**: Bank 2 in power-down mode

This bit indicates that the Flash memory bank 2 is in power-down state. It is reset when bank 2 is in normal mode or being awaken.

Bit 20 **PD1**: Bank 1 in power-down mode

This bit indicates that the Flash memory bank 1 is in power-down state. It is reset when bank 1 is in normal mode or being awaken.

Bit 19 **OEM2LOCK**: OEM2 lock

This bit indicates that the OEM2 RDP key read during the OBL is not virgin. When set, the OEM2 RDP lock mechanism is active.

Bit 18 **OEM1LOCK**: OEM1 lock

This bit indicates that the OEM1 RDP key read during the OBL is not virgin. When set, the OEM1 RDP lock mechanism is active.

Bit 17 **WDW**: Non-secure wait data to write

This bit indicates that the Flash memory write buffer has been written by a secure or non-secure operation. It is set when the first data is stored in the buffer and cleared when the write is performed in the Flash memory.

Bit 16 **BSY**: Non-secure busy

This indicates that a Flash memory secure or non-secure operation is in progress. This bit is set at the beginning of a Flash operation and reset when the operation finishes or when an error occurs.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OPTWERR**: Option write error

This bit is set by hardware when the options bytes are written with an invalid configuration. It is cleared by writing 1.

Refer to *Section 7.3.9: Flash memory errors flags* for full conditions of error flag setting.

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **PGSERR**: 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 7.3.9: Flash memory errors flags* for full conditions of error flag setting.

Bit 6 **SIZERR**: Non-secure size error

This bit is set by hardware when the size of the access is a byte or half-word during a non-secure program sequence. Only quad-word programming is allowed by means of successive word accesses. This bit is cleared by writing 1.

Bit 5 **PGAERR**: Non-secure programming alignment error

This bit is set by hardware when the first word to be programmed is not aligned with a quad-word address, or the second, third or forth word does not belong to the same quad-word address. This bit is cleared by writing 1.

Bit 4 **WRPERR**: Non-secure write protection error

This bit is set by hardware when a non-secure address to be erased/programmed belongs to a write-protected part (by WRP, HDP or RDP level 1) of the Flash memory. This bit is cleared by writing 1.

Refer to *Section 7.3.9: Flash memory errors flags* for full conditions of error flag setting.

Bit 3 **PROGERR**: Non-secure programming error

This bit is set by hardware when a non-secure quad-word address to be programmed contains a value different from all 1 before programming, except if the data to write is all 0. This bit is cleared by writing 1.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **OPERR**: Non-secure operation error

This bit is 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). This bit is cleared by writing 1.

Bit 0 **EOP**: Non-secure end of operation

This bit is set by hardware when one or more Flash memory non-secure operation (program/erase) has been completed successfully. This bit is set only if the non-secure end of operation interrupts are enabled (EOPIE = 1 in FLASH_NSCR). This bit is cleared by writing 1.

7.9.8 FLASH secure 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 unprivileged access when SPRIV = 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.	WDW	BSY
														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.	PGSERR	SIZERR	PGAERR	WRPERR	PROGERR	Res.	OPERR	EOP
								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **WDW**: Secure wait data to write

This bit indicates that the Flash memory write buffer has been written by a secure or non-secure operation. It is set when the first data is stored in the buffer and cleared when the write is performed in the Flash memory.

Bit 16 **BSY**: Secure busy

This bit indicates that a Flash memory 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.

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

Bit 7 **PGSERR**: 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 7.3.9: Flash memory errors flags* for full conditions of error flag setting.

Bit 6 **SIZERR**: Secure size error

This bit is set by hardware when the size of the access is a byte or half-word during a secure program sequence. Only quad-word programming is allowed by means of successive word accesses.This bit is cleared by writing 1.

Bit 5 **PGAERR**: Secure programming alignment error

This bit is set by hardware when the first word to be programmed is not aligned with a quad-word address, or the second, third or forth word does not belong to the same quad-word address.This bit is cleared by writing 1.

Bit 4 **WRPERR**: Secure write protection error

This bit is set by hardware when a secure address to be erased/programmed belongs to a write-protected part (by WRP, HDP or RDP level 1) of the Flash memory.This bit is cleared by writing 1.

Refer to *Section 7.3.9: Flash memory errors flags* for full conditions of error flag setting.

Bit 3 **PROGERR**: Secure programming error

This bit is set by hardware when a secure quad-word address to be programmed contains a value different from all 1 before programming, except if the data to write is all 0. This bit is cleared by writing 1.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **OPERR**: Secure operation error

This bit is 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). This bit is cleared by writing 1.

Bit 0 **EOP**: Secure end of operation

This bit is set by hardware when one or more Flash memory secure operation (program/erase) has been completed successfully. This bit is set only if the secure end of operation interrupts are enabled (EOPIE = 1 in FLASH_SECCR). This bit is cleared by writing 1.

7.9.9 FLASH non-secure control register (FLASH_NSCR)

This register can only be written when the BSY or OBL_LAUNCH are reset. Otherwise, the write access is stalled until the BSY bits 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 unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x28

Reset value: 0xC000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	OPTLOCK	Res.	Res.	OBL_LAUNCH	Res.	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	Res.	OPTSTRT	STRT
rs	rs			rc_w1		rw	rw							rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MER2	BWR	Res.	Res.	BKER	Res.	PNB[6:0]							MER1	PER	PG
rw	rw			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LOCK:** Non-secure 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:** Option lock

This bit is set only. When set, all bits concerning user options in FLASH_NSCR register are locked. This bit is cleared by hardware after detecting the unlock sequence. The LOCK bit in the FLASH_NSCR 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 **ERRIE**: Non-secure error interrupt enable
This bit enables the interrupt generation when the OPERR bit in the FLASH_NSSR is set to 1.
0: Non-secure OPERR error interrupt disabled
1: Non-secure OPERR error interrupt enabled

Bit 24 **EOPIE**: Non-secure end of operation interrupt enable
This bit enables the interrupt generation when the EOP bit in the FLASH_NSSR is set to 1.
0: Non-secure EOP Interrupt disabled
1: Non-secure EOP 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 BSY bit is cleared in FLASH_NSSR.

Bit 16 **STRT**: Non-secure start
This bit triggers a non-secure erase operation when set. If MER1, MER2 and PER bits are reset and the STRT bit is set, the PGSERR bit in FLASH_NSSR is set (this condition is forbidden).
This bit is set only by software and is cleared when the BSY bit is cleared in FLASH_NSSR.

Bit 15 **MER2**: Non-secure bank 2 mass erase
This bit triggers the bank 2 non-secure mass erase (all bank 2 user pages) when set.

Bit 14 **BWR**: Non-secure burst write programming mode
When set, this bit selects the burst write programming mode.

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **BKER**: Non-secure bank selection for page erase
0: Bank 1 selected for non-secure page erase
1: Bank 2 selected for non-secure page erase

Bit 10 Reserved, must be kept at reset value.

Bits 9:3 **PNB[6:0]**: Non-secure page number selection
These bits select the page to erase.
0000000: page 0
0000000: page 1
...
1111111: page 127

Bit 2 **MER1**: 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 **PER**: Non-secure page erase
0: Non-secure page erase disabled
1: Non-secure page erase enabled

Bit 0 **PG**: Non-secure programming
0: Non-secure Flash programming disabled
1: Non-secure Flash programming enabled

7.9.10 FLASH secure control register (FLASH_SECCR)

This register can only be written when the BSY or OBL_LAUNCH are reset. Otherwise, the write access stalls until the BSY bits 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 unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x2C

Reset value: 0x8000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	INV	Res.	Res.	Res.	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT
rs		rw				rw	rw								rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MER2	BWR	Res.	Res.	BKER	Res.	PNB[6:0]							MER1	PER	PG
rw	rw			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LOCK:** Secure 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 **INV:** Flash memory security state invert
This bit inverts the Flash memory security state.

Bits 28:26 Reserved, must be kept at reset value.

Bit 25 **ERRIE**: Secure error interrupt enable
This bit enables the interrupt generation when the OPERR bit in the FLASH_SECSR is set to 1.
0: Secure OPERR error interrupt disabled
1: Secure OPERR error interrupt enabled

Bit 24 **EOPIE**: Secure End of operation interrupt enable
This bit enables the interrupt generation when the EOP bit in the FLASH_SECSR is set to 1.
0: Secure EOP Interrupt disabled
1: Secure EOP Interrupt enabled

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **STRT**: Secure start
This bit triggers a secure erase operation when set. If MER1, MER2 and PER bits are reset and the STRT bit is set, the PGSERR in the FLASH_SECSR is set (this condition is forbidden).
This bit is set only by software and is cleared when the BSY bit is cleared in FLASH_SECSR.

Bit 15 **MER2**: Secure bank 2 mass erase
This bit triggers the bank 2 secure mass erase (all bank 2 user pages) when set.

Bit 14 **BWR**: Secure burst write programming mode
When set, this bit selects the burst write programming mode.

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **BKER**: Secure bank selection for page erase
0: Bank 1 selected for secure page erase
1: Bank 2 selected for secure page erase

Bit 10 Reserved, must be kept at reset value.

Bits 9:3 **PNB[6:0]**: Secure page number selection
These bits select the page to erase:
00000000: page 0
00000001: page 1
...
11111111: page 127

Bit 2 **MER1**: Secure bank 1 mass erase
This bit triggers the bank 1 secure mass erase (all bank 1 user pages) when set.

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

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

7.9.11 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 unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x30

Reset value: 0x0000 0000

Access: no wait state; word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECCD	ECCC	Res.	Res.	Res.	Res.	Res.	ECCIE	Res.	SYSF_ECC	BK_ECC	Res.	ADDR_ECC[19:16]			
rc_w1	rc_w1						rw		r	r		r	r	r	r

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

Bit 31 **ECCD**: ECC detection

This bit is set by hardware when two ECC errors have been detected (only if ECCC and ECCD were previously cleared). When this bit is set, a NMI is generated. This bit is cleared by writing 1.

Bit 30 **ECCC**: ECC correction

This bit is set by hardware when one ECC error has been detected and corrected (only if ECCC and ECCD were previously cleared). An interrupt is generated if ECCIE is set. This bit is cleared by writing 1.

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

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

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

0: ECCC interrupt disabled

1: ECCC interrupt enabled.

Bit 23 Reserved, must be kept at reset value.

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

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

Bit 21 **BK_ECC**: ECC fail bank

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

Bit 20 Reserved, must be kept at reset value.

Bits 19:0 **ADDR_ECC[19:0]**: ECC fail address

This field indicates which address is concerned by the ECC error correction or by the double ECC error detection. The address is given by bank from address 0x0 0000 to 0xF FFF0.

7.9.12 FLASH operation status register (FLASH_OPSR)

This register is non-secure. It can be read and written by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x34

Reset value: 0xX0XX XXXX

(0xX0XX XXXX after system reset and 0x0000 0000 after power-on 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
CODE_OP[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	SYSF_OP	BK_OP	Res.	ADDR_OP[19:16]			
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
ADDR_OP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 **CODE_OP[2:0]**: Flash memory operation code

This field indicates which Flash memory operation has been interrupted by a system reset:
000: No Flash operation interrupted by previous reset
001: Single write operation interrupted
010: Burst write operation interrupted
011: Page erase operation interrupted
100: Bank erase operation interrupted
101: Mass erase operation interrupted
110: Option change operation interrupted
111: Reserved

Bits 28:23 Reserved, must be kept at reset value.

Bit 22 **SYSF_OP**: Operation in system Flash memory interrupted

This bit indicates that the reset occurred during an operation in the system Flash memory.

Bit 21 **BK_OP**: Interrupted operation bank

This bit indicates which Flash memory bank was accessed when reset occurred
0: Bank 1
1: Bank 2

Bit 20 Reserved, must be kept at reset value.

Bits 19:0 **ADDR_OP[19:0]**: Interrupted operation address

This field indicates which address in the Flash memory was accessed when reset occurred. The address is given by bank from address 0x0 0000 to 0xF FFF0.

7.9.13 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 unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x40

Reset value: 0xXXXX XXXX (register bits 0 to 31 are loaded with values from the Flash memory at OBL)

ST production value: 0x1FEF F8AA

Access: no wait state when no option bytes modification is ongoing; word, half-word and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TZEN	IO_VDDIO2_HSLV	IO_VDD_HSLV	PA15_PUPEN	NBOOT0	NSWBOOT0	SRAM2_RST	SRAM2_ECC	SRAM3_ECC	BKPRAM_ECC	DUALBANK	SWAP_BANK	WWDG_SW	IWDG_STDBY	IWDG_STOP	IWDG_SW
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
SRAM1345_RST	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	rw

Bit 31 **TZEN:** Global TrustZone security enable
0: Global TrustZone security disabled
1: Global TrustZone security enabled

Bit 30 **IO_VDDIO2_HSLV:** High-speed IO at low V_{DDIO2} voltage configuration bit
This bit can be set only with V_{DDIO2} below 2.5 V.
0: High-speed IO at low V_{DDIO2} voltage feature disabled (V_{DDIO2} can exceed 2.5 V)
1: High-speed IO at low V_{DDIO2} voltage feature enabled (V_{DDIO2} remains below 2.5 V)

Bit 29 **IO_VDD_HSLV:** High-speed IO at low V_{DD} voltage configuration bit
This bit can be set only with V_{DD} below 2.5V
0: High-speed IO at low V_{DD} voltage feature disabled (V_{DD} can exceed 2.5 V)
1: High-speed IO at low V_{DD} voltage feature enabled (V_{DD} remains below 2.5 V)

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 not erased when a system reset occurs

Bit 24 **SRAM2_ECC:** SRAM2 ECC detection and correction enable
0: SRAM2 ECC check enabled
1: SRAM2 ECC check disabled

Bit 23 **SRAM3_ECC:** SRAM3 ECC detection and correction enable
0: SRAM3 ECC check enabled
1: SRAM3 ECC check disabled

Bit 22 **BKPRAM_ECC:** Backup RAM ECC detection and correction enable
0: Backup RAM ECC check enabled
1: Backup RAM ECC check disabled

Bit 21 **DUALBANK:** Dual-bank on 1-Mbyte Flash memory devices
0: Single bank Flash with contiguous address in bank 1
1: Dual-bank Flash with contiguous addresses

Bit 20 **SWAP_BANK:** Swap banks
0: Bank 1 and bank 2 addresses not swapped
1: Bank 1 and bank 2 addresses swapped

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

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

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

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

Bit 15 **SRAM1345_RST:** SRAM1, SRAM3 and SRAM4 erase upon system reset
0: SRAM1, SRAM3 and SRAM4 erased when a system reset occurs
1: SRAM1, SRAM3 and SRAM4 not erased when a system reset occurs

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

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

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

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **BOR_LEV[2:0]:** BOR reset level

These bits contain the V_{DD} supply level threshold that activates/releases the reset.
000: BOR level 0 (reset level threshold around 1.7 V)
001: BOR level 1 (reset level threshold around 2.0 V)
010: BOR level 2 (reset level threshold around 2.2 V)
011: BOR level 3 (reset level threshold around 2.5 V)
100: BOR level 4 (reset level threshold 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 7.6.2: Readout protection (RDP) for more details.

7.9.14 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 unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x44

Reset value: 0xXXXX XXXX (option bytes are loaded with values from the Flash memory at reset release)

ST production value: 0x0800 007F

Access: no wait state when no option bytes modification is ongoing; 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]															
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
NSBOOTADD0[8:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw							

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. These bits correspond to address [31:7]. The NSBOOTADD0 option bytes are selected following the BOOT0 pin or NSWBOOT0 state.
Examples:
NSBOOTADD0[24:0] = 0x0100000: Boot from non-secure Flash memory (0x0800 0000)
NSBOOTADD0[24:0] = 0x017F200: Boot from system memory bootloader (0x0BF9 0000)
NSBOOTADD0[24:0] = 0x0400000: Boot from non-secure SRAM1 on S-Bus (0x2000 0000)

Bits 6:0 Reserved, must be kept at reset value.

7.9.15 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 unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x48

Reset value: 0xXXXX XXXX (option bytes are loaded with values from the Flash memory at reset release)

ST production value: 0x0BF9 007F

Access: no wait state when no option bytes modification is ongoing; 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]															
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
NSBOOTADD1[8:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw							

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. These bits correspond to address [31:7]. The NSBOOTADD0 option bytes are selected following the BOOT0 pin or NSWBOOT0 state.

Examples:

NSBOOTADD1[24:0] = 0x0100000: Boot from non-secure Flash memory (0x0800 0000)

NSBOOTADD1[24:0] = 0x017F200: Boot from system memory bootloader (0x0BF9 0000)

NSBOOTADD1[24:0] = 0x0400000: Boot from non-secure SRAM1 on S-Bus (0x2000 0000)

Bits 6:0 Reserved, must be kept at reset value.

7.9.16 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 unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x4C

Reset value: 0xXXXX XXXX (option bytes are loaded with values from the Flash memory at reset release)

ST production value: 0x0C00 007C

Access: no wait state when no option bytes modification is ongoing; 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]															
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
SECBOOTADD0[8:0]									Res.	Res.	Res.	Res.	Res.	Res.	BOOT_LOCK
rw	rw	rw	rw	rw	rw	rw	rw	rw							rw

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. This bits correspond to address [31:7] The SECBOOTADD0 option bytes are selected following the BOOT0 pin or NSWBOOT0 state.

Examples:

SECBOOTADD0[24:0] = 0x018 0000: Boot from secure Flash memory (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, this bit can only be cleared when RDP is at Level 0.

7.9.17 FLASH secure watermark1 register 1 (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 unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x50

Reset value: 0xXXXX XXXX (register bits are loaded with values from the Flash memory at OBL, reserved bits are read as 1)

ST production value: 0xFFFF FF80

Access: no wait state when no option bytes modification is ongoing; 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
This field contains the last page of the secure area in bank 1.

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

Bits 6:0 **SECWM1_PSTRT[6:0]:** Start page of first secure area
This field contains the first page of the secure area in bank 1.

7.9.18 FLASH secure watermark1 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 unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x54

Reset value: 0xXXXX XXXX (register bits are loaded with values from the Flash memory at OBL)

ST production value: 0x7F80 7F80

Access: no wait state when no option bytes modification is ongoing; 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 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
This field contains the last page of the HDP area in bank 1.

Bits 15:0 Reserved, must be kept at reset value.

7.9.19 FLASH WRP1 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 unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x58

Reset value: 0xXXXX XXXX (register bits are loaded with values from the Flash memory at OBL, reserved bits are read as 1)

ST production value: 0xFF80 FFFF

Access: no wait state when no option bytes modification is ongoing; word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1A_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.	WRP1A_PSTRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bit 31 **UNLOCK**: Bank 1 WPR first area A unlock
0: WRP1A start and end pages locked
1: WRP1A start and end pages unlocked

Bits 30:23 Reserved, must be kept at reset value.

Bits 22:16 **WRP1A_PEND[6:0]**: Bank 1 WPR first area A end page
This field contains the last page of the first WPR area in bank 1.

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
This field contains the first page of the first WPR area for bank 1.

7.9.20 FLASH WRP1 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 unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x5C

Reset value: 0xXXXX XXXX (register bits are loaded with values from the Flash memory at OBL)

ST production value: 0xFF80 FFFF

Access: no wait state when no option bytes modification is ongoing; word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP1B_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.	WRP1B_PSTRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bit 31 **UNLOCK**: Bank 1 WPR second area B unlock
0: WRP1B start and end pages locked
1: WRP1B start and end pages unlocked

Bits 30:23 Reserved, must be kept at reset value.

Bits 22:16 **WRP1B_PEND[6:0]**: Bank 1 WRP second area B end page
This field contains the last page of the second WRP area in bank 1.

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
This field contains the first page of the second WRP area for bank 1.

7.9.21 FLASH secure watermark2 register 1 (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 unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x60

Reset value: 0xXXXX XXXX (register bits are loaded with values from the Flash memory at OBL)

ST production value: 0xFFFF FF80

Access: no wait state when no option bytes modification is ongoing; 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
This field contains the last page of the secure area in bank 2.

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

Bits 6:0 **SECWM2_PSTRT[6:0]**: Start page of second secure area
This field contains the first page of the secure area in bank 2.

7.9.22 FLASH secure watermark2 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 unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x64

Reset value: 0xXXXX XXXX (register bits are loaded with values from the Flash memory at OBL)

ST production value: 0x7F80 7F80

Access: no wait state when no option bytes modification is ongoing; 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
HDP2_PEND contains the last page of the HDP area in bank 2.

Bits 15:0 Reserved, must be kept at reset value.

7.9.23 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 unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x68

Reset value: 0xXXXX XXXX (register bits are loaded with values from the Flash memory at OBL)

ST production value: 0xFF80 FFFF

Access: no wait state when no option bytes modification is ongoing; word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2A_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.	WRP2A_PSTRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bit 31 **UNLOCK**: Bank 2 WPR first area A unlock
0: WRP2A start and end pages locked
1: WRP2A start and end pages unlocked

Bits 30:23 Reserved, must be kept at reset value.

Bits 22:16 **WRP2A_PEND[6:0]**: Bank 2 WPR first area A end page
This field contains the last page of the first WRP area in bank 2.

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
This field contains the first page of the first WRP area for bank 2.

7.9.24 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 unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x6C

Reset value: 0xXXXX XXXX (register bits are loaded with values from the Flash memory at OBL)

ST production value: 0xFF80 FFFF

Access: no wait state when no option bytes modification is ongoing; word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRP2B_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.	WRP2B_PSTRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bit 31 **UNLOCK**: Bank 2 WPR second area B unlock
0: WRP2B start and end pages locked
1: WRP2B start and end pages unlocked

Bits 30:23 Reserved, must be kept at reset value.

Bits 22:16 **WRP2B_PEND[6:0]**: Bank 2 WPR second area B end page
This field contains the last page of the second WRP area in bank 2.

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
This field contains the first page of the second WRP area for bank 2.

7.9.25 FLASH OEM1 key register 1 (FLASH_OEM1KEYR1)

This register is non-secure. It can be written by both secure and non-secure access. This register is read as zero. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x70

Reset value: 0x0000 0000

Access: no wait state when no option bytes modification is ongoing; word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEM1KEY[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
OEM1KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEM1KEY[31:0]**: OEM1 least significant bytes key

7.9.26 FLASH OEM1 key register 2 (FLASH_OEM1KEYR2)

This register is non-secure. It can be written by both secure and non-secure access. This register is read as zero. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x74

Reset value: 0x0000 0000

Access: no wait state when no option bytes modification is ongoing; word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEM1KEY[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
OEM1KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEM1KEY[63:32]**: OEM1 most significant bytes key

7.9.27 FLASH OEM2 key register 1 (FLASH_OEM2KEYR1)

This register is non-secure. It can be written by both secure and non-secure access. This register is read as zero. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x78

Reset value: 0x0000 0000

Access: no wait state when no option bytes modification is ongoing; word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEM2KEY[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
OEM2KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEM2KEY[31:0]**: OEM2 least significant bytes key

7.9.28 FLASH OEM2 key register 2 (FLASH_OEM2KEYR2)

This register is non-secure. It can be written by both secure and non-secure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x7C

Reset value: 0x0000 0000

Access: no wait state when no option bytes modification is ongoing; word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEM2KEY[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
OEM2KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OEM2KEY[63:32]**: OEM2 most significant bytes key

7.9.29 FLASH secure block based bank 1 register x (FLASH_SECBB1Rx)

This register is secure. It can be written only by secure access. This register can be protected against unprivileged access (refer to *Table 66: SECyBBRx registers access when TrustZone is active (TZEN = 1)*).

Address offset: 0x80 + 4 *(x - 1), (x = 1 to 4)

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
SEC1BB31	SEC1BB30	SEC1BB29	SEC1BB28	SEC1BB27	SEC1BB26	SEC1BB25	SEC1BB24	SEC1BB23	SEC1BB22	SEC1BB21	SEC1BB20	SEC1BB19	SEC1BB18	SEC1BB17	SEC1BB16
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
SEC1BB15	SEC1BB14	SEC1BB13	SEC1BB12	SEC1BB11	SEC1BB10	SEC1BB9	SEC1BB8	SEC1BB7	SEC1BB6	SEC1BB5	SEC1BB4	SEC1BB3	SEC1BB2	SEC1BB1	SEC1BB0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SEC1BBi:** page secure/non-secure attribution (i = 31 to 0)

Each bit is used to set one page security attribution in bank 1.

0: Page (32 * (x - 1) + i) in bank 1 not block-based secure

1: Page (32 * (x - 1) + i) in bank 1 block-based secure

7.9.30 FLASH secure block based bank 2 register x (FLASH_SECBB2Rx)

This register is secure. It can be written only by a secure access. This register can be protected against unprivileged access (refer to *Table 66: SECyBBRx registers access when TrustZone is active (TZEN = 1)*).

Address offset: 0xA0 + 4 *(x - 1), (x = 1 to 4)

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
SEC2BB31	SEC2BB30	SEC2BB29	SEC2BB28	SEC2BB27	SEC2BB26	SEC2BB25	SEC2BB24	SEC2BB23	SEC2BB22	SEC2BB21	SEC2BB20	SEC2BB19	SEC2BB18	SEC2BB17	SEC2BB16
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
SEC2BB15	SEC2BB14	SEC2BB13	SEC2BB12	SEC2BB11	SEC2BB10	SEC2BB9	SEC2BB8	SEC2BB7	SEC2BB6	SEC2BB5	SEC2BB4	SEC2BB3	SEC2BB2	SEC2BB1	SEC2BB0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SEC2BBi:** page secure/non-secure attribution (i = 31 to 0)
Each bit is used to set one page security attribution in bank 2.
0: Page (32 * (x - 1) + i) in bank 2 not block-based secure
1: Page (32 * (x - 1) + i) in bank 2 block-based secure

7.9.31 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 unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0xC0

Reset value: 0x0000 0000

Access: no wait state; word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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 granted
1: Access to HDP2 area denied (SECWM2Ry option bytes modification bocked -refer to *Rules for modifying specific option bytes*)

Bit 0 **HDP1_ACCDIS:** HDP1 area access disable
When set, this bit is only cleared by a system reset.
0: Access to HDP1 area granted
1: Access to HDP1 area denied (SECWM1Ry option bytes modification blocked - refer to *Rules for modifying specific option bytes*)

7.9.32 FLASH privilege configuration register (FLASH_PRIVCFGR)

This register can be read by both privileged and unprivileged access. NSPRIV is a non-secure bit. SPRIV is a secure bit.

Address offset: 0xC4.

Reset value: 0x0000 0000

Access: no wait state; word, half-word and byte access

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **NSPRIV**: Privileged protection for non-secure registers

This bit can be read by both privileged or unprivileged, secure and non-secure access.

0: Non-secure Flash registers can be read and written by privileged or unprivileged access.

1: Non-secure Flash registers can be read and written by privileged access only.

The NSPRIV bit can be written by a secure or non-secure privileged access. A secure or non-secure unprivileged write access on NSPRIV bit is ignored.

Bit 0 **SPRIV**: Privileged protection for secure registers

This bit can be accessed only when TrustZone is enabled (TZEN = 1). This bit can be read by both privileged or unprivileged, secure and non-secure access.

0: Secure Flash registers can be read and written by privileged or unprivileged access.

1: Secure Flash registers can be read and written by privileged access only.

The SPRIV bit can be written only by a secure privileged access. A non-secure write access on SPRIV bit is ignored. A secure unprivileged write access on SPRIV bit is ignored.

7.9.33 FLASH privilege block based bank 1 register x (FLASH_PRIVBB1Rx)

This register is privileged. It can be read written only by a privileged access. This register can be protected against non-secure access (refer to *Table 67: PRIVyBBRx registers access when TrustZone is active (TZEN = 1)*).

Address offset: 0xD0 + 4 *(x - 1), (x = 1 to 4)

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
PRIV1BB31	PRIV1BB30	PRIV1BB29	PRIV1BB28	PRIV1BB27	PRIV1BB26	PRIV1BB25	PRIV1BB24	PRIV1BB23	PRIV1BB22	PRIV1BB21	PRIV1BB20	PRIV1BB19	PRIV1BB18	PRIV1BB17	PRIV1BB16
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
PRIV1BB15	PRIV1BB14	PRIV1BB13	PRIV1BB12	PRIV1BB11	PRIV1BB10	PRIV1BB9	PRIV1BB8	PRIV1BB7	PRIV1BB6	PRIV1BB5	PRIV1BB4	PRIV1BB3	PRIV1BB2	PRIV1BB1	PRIV1BB0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIV1BBi:** page privileged/unprivileged attribution (i = 31 to 0)

Each bit is used to set one page privilege attribution in bank 1.

0: Page (32 * (x - 1) + i) in bank 1 accessible by unprivileged access

1: Page (32 * (x - 1) + i) in bank 1 only accessible by privileged access

7.9.34 FLASH privilege block based bank 2 register x (FLASH_PRIVBB2Rx)

This register is privilege. It can be read written only by a privileged access. This register can be protected against non-secure access (refer to *Table 67: PRIVyBBRx registers access when TrustZone is active (TZEN = 1)*).

Address offset: 0xF0 + 4 *(x - 1), (x = 1 to 4)

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
PRIV2BB31	PRIV2BB30	PRIV2BB29	PRIV2BB28	PRIV2BB27	PRIV2BB26	PRIV2BB25	PRIV2BB24	PRIV2BB23	PRIV2BB22	PRIV2BB21	PRIV2BB20	PRIV2BB19	PRIV2BB18	PRIV2BB17	PRIV2BB16
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
PRIV2BB15	PRIV2BB14	PRIV2BB13	PRIV2BB12	PRIV2BB11	PRIV2BB10	PRIV2BB9	PRIV2BB8	PRIV2BB7	PRIV2BB6	PRIV2BB5	PRIV2BB4	PRIV2BB3	PRIV2BB2	PRIV2BB1	PRIV2BB0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIV2BBi:** page privileged/unprivileged attribution (i = 31 to 0)
Each bit is used to set one page security attribution in bank 2.
0: Page (32 * (x - 1) + i) in bank 2 accessible by unprivileged access
1: Page (32 * (x - 1) + i) in bank 2 only accessible by privileged access

7.9.35 FLASH register map

Table 70. FLASH 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.	Res.	SLEEP_PD	PDREQ2	PDREQ1	LPM	Res.	Res.	PRFTEN	Res.	Res.	Res.	Res.	LATENCY [3:0]			
	Reset value																		0	0	0	0			0					0	0	0	0
0x04	Reserved	Reserved																															
0x08	FLASH_NSKEYR	NSKEY[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	SECKEY[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	OPTKEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	Reserved	Reserved																															
0x18	FLASH_PDKEY1R	PDKEY1[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 70. FLASH 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
0x1C	FLASH_PDKEY2R	PDKEY2[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.	PD2	PD1	OEM2LOCK	OEM1LOCK	WDW	BSY	Res.	Res.	OPTWERR	Res.	Res.	Res.	Res.	Res.	PGSERR	SIZERR	PGAERR	WRPERR	PROGERR	Res.	OPERR	EOP
	Reset value											0	0	X	X	X	X			0						0	0	0	0	0		0	0
0x24	FLASH_SECSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDW	BSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PGSERR	SIZERR	PGAERR	WRPERR	PROGERR	Res.	OPERR	EOP
	Reset value															0	0									0	0	0	0	0		0	0
0x28	FLASH_NSCR	LOCK	OPTLOCK	Res.	Res.	OBL_LAUNCH	Res.	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	Res.	OPTSTRT	STRT	MER2	BWR	Res.	Res.	BKER	Res.	PNB[6:0]							MER1	PER	PG
	Reset value	1	1			0		0	0							0	0	0	0			0		0	0	0	0	0	0	0	0	0	0
0x2C	FLASH_SECCR	LOCK	Res.	INV	Res.	Res.	Res.	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT	MER2	BWR	Res.	Res.	BKER	Res.	PNB[6:0]							MER1	PER	PG
	Reset value	1		0				0	0								0	0	0			0		0	0	0	0	0	0	0	0	0	0
0x30	FLASH_ECCR	ECCD	ECCC	Res.	Res.	Res.	Res.	Res.	ECCCIE	Res.	SYSF_ECC	BK_ECC	Res.	ADDR_ECC[19:0]																			
	Reset value	0	0						0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	FLASH_OPSR	CODE_OP[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	SYSF_OP	BK_OP	Res.	ADDR_OP[19:0]																			
	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
0x38 to 0x3C	Reserved	Reserved																															
0x40	FLASH_OPTR	TZEN	IO_VDDIO2_HSLV	IO_VDD_HSLV	PA15_PUPEN	NBOOT0	NSWBOOT0	SRAM2_RST	SRAM2_ECC	SRAM3_ECC	BKPRAM_ECC	DUALBANK	SWAP_BANK	WWDG_SW	IWDG_STDBY	IWDG_STOP	IWDG_SW	SRAM1345_RST	NRST_SHDW	NRST_STDBY	NRST_STOP	Res.	BOR_LEV[2:0]			RDP[7:0]							
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X
0x44	FLASH_ NSBOOTADD0R	NSBOOTADD0[24:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X							
0x48	FLASH_ NSBOOTADD1R	NSBOOTADD1[24:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X							

Table 70. FLASH 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
0x4C	FLASH_ SECBOOTADD0R	SECBOOTADD0[24:0]																									Res.	Res.	Res.	Res.	Res.	Res.	BOOT_LOCK
	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
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]						
	Reset value										X	X	X	X	X	X	X										X	X	X	X	X	X	X
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.
	Reset value	X									X	X	X	X	X	X	X																
0x58	FLASH_WRP1AR	UNLOCK	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]						
	Reset value	X									X	X	X	X	X	X	X										X	X	X	X	X	X	X
0x5C	FLASH_WRP1BR	UNLOCK	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]						
	Reset value	X									X	X	X	X	X	X	X										X	X	X	X	X	X	X
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]						
	Reset value										X	X	X	X	X	X	X										X	X	X	X	X	X	X
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.
	Reset value	X									X	X	X	X	X	X	X																
0x68	FLASH_WRP2AR	UNLOCK	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]						
	Reset value	X									X	X	X	X	X	X	X	X									X	X	X	X	X	X	X
0x6C	FLASH_WRP2BR	UNLOCK	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]						
	Reset value	X									X	X	X	X	X	X	X	X									X	X	X	X	X	X	X
0x70	FLASH_ OEM1KEYR1	OEM1KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74	FLASH_ OEM1KEYR2	OEM1KEY[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
0x78	FLASH_ OEM2KEYR1	OEM2KEY[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
0x7C	FLASH_ OEM2KEYR2	OEM2KEY[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

Table 70. FLASH 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
0x80	FLASH_ SECB1BR1	SEC1BB31	SEC1BB30	SEC1BB29	SEC1BB28	SEC1BB27	SEC1BB26	SEC1BB25	SEC1BB24	SEC1BB23	SEC1BB22	SEC1BB21	SEC1BB20	SEC1BB19	SEC1BB18	SEC1BB17	SEC1BB16	SEC1BB15	SEC1BB14	SEC1BB13	SEC1BB12	SEC1BB11	SEC1BB10	SEC1BB9	SEC1BB8	SEC1BB7	SEC1BB6	SEC1BB5	SEC1BB4	SEC1BB3	SEC1BB2	SEC1BB1	SEC1BB0
	Reset value	0	0	0	0	0	0	0	0	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	FLASH_ SECBB1R2	SEC1BB31	SEC1BB30	SEC1BB29	SEC1BB28	SEC1BB27	SEC1BB26	SEC1BB25	SEC1BB24	SEC1BB23	SEC1BB22	SEC1BB21	SEC1BB20	SEC1BB19	SEC1BB18	SEC1BB17	SEC1BB16	SEC1BB15	SEC1BB14	SEC1BB13	SEC1BB12	SEC1BB11	SEC1BB10	SEC1BB9	SEC1BB8	SEC1BB7	SEC1BB6	SEC1BB5	SEC1BB4	SEC1BB3	SEC1BB2	SEC1BB1	SEC1BB0
	Reset value	0	0	0	0	0	0	0	0	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	FLASH_ SECBB1R3	SEC1BB31	SEC1BB30	SEC1BB29	SEC1BB28	SEC1BB27	SEC1BB26	SEC1BB25	SEC1BB24	SEC1BB23	SEC1BB22	SEC1BB21	SEC1BB20	SEC1BB19	SEC1BB18	SEC1BB17	SEC1BB16	SEC1BB15	SEC1BB14	SEC1BB13	SEC1BB12	SEC1BB11	SEC1BB10	SEC1BB9	SEC1BB8	SEC1BB7	SEC1BB6	SEC1BB5	SEC1BB4	SEC1BB3	SEC1BB2	SEC1BB1	SEC1BB0
	Reset value	0	0	0	0	0	0	0	0	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	FLASH_ SECBB1R4	SEC1BB31	SEC1BB30	SEC1BB29	SEC1BB28	SEC1BB27	SEC1BB26	SEC1BB25	SEC1BB24	SEC1BB23	SEC1BB22	SEC1BB21	SEC1BB20	SEC1BB19	SEC1BB18	SEC1BB17	SEC1BB16	SEC1BB15	SEC1BB14	SEC1BB13	SEC1BB12	SEC1BB11	SEC1BB10	SEC1BB9	SEC1BB8	SEC1BB7	SEC1BB6	SEC1BB5	SEC1BB4	SEC1BB3	SEC1BB2	SEC1BB1	SEC1BB0
	Reset value	0	0	0	0	0	0	0	0	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 to 0x9C	Reserved	Reserved																															
0xA0	FLASH_ SECBB2R1	SEC2BB31	SEC2BB30	SEC2BB29	SEC2BB28	SEC2BB27	SEC2BB26	SEC2BB25	SEC2BB24	SEC2BB23	SEC2BB22	SEC2BB21	SEC2BB20	SEC2BB19	SEC2BB18	SEC2BB17	SEC2BB16	SEC2BB15	SEC2BB14	SEC2BB13	SEC2BB12	SEC2BB11	SEC2BB10	SEC2BB9	SEC2BB8	SEC2BB7	SEC2BB6	SEC2BB5	SEC2BB4	SEC2BB3	SEC2BB2	SEC2BB1	SEC2BB0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA4	FLASH_ SECBB2R2	SEC2BB31	SEC2BB30	SEC2BB29	SEC2BB28	SEC2BB27	SEC2BB26	SEC2BB25	SEC2BB24	SEC2BB23	SEC2BB22	SEC2BB21	SEC2BB20	SEC2BB19	SEC2BB18	SEC2BB17	SEC2BB16	SEC2BB15	SEC2BB14	SEC2BB13	SEC2BB12	SEC2BB11	SEC2BB10	SEC2BB9	SEC2BB8	SEC2BB7	SEC2BB6	SEC2BB5	SEC2BB4	SEC2BB3	SEC2BB2	SEC2BB1	SEC2BB0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA8	FLASH_ SECBB2R3	SEC2BB31	SEC2BB30	SEC2BB29	SEC2BB28	SEC2BB27	SEC2BB26	SEC2BB25	SEC2BB24	SEC2BB23	SEC2BB22	SEC2BB21	SEC2BB20	SEC2BB19	SEC2BB18	SEC2BB17	SEC2BB16	SEC2BB15	SEC2BB14	SEC2BB13	SEC2BB12	SEC2BB11	SEC2BB10	SEC2BB9	SEC2BB8	SEC2BB7	SEC2BB6	SEC2BB5	SEC2BB4	SEC2BB3	SEC2BB2	SEC2BB1	SEC2BB0
	Reset value	0	0	0	0	0	0	0	0	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	FLASH_ SECBB2R4	SEC2BB31	SEC2BB30	SEC2BB29	SEC2BB28	SEC2BB27	SEC2BB26	SEC2BB25	SEC2BB24	SEC2BB23	SEC2BB22	SEC2BB21	SEC2BB20	SEC2BB19	SEC2BB18	SEC2BB17	SEC2BB16	SEC2BB15	SEC2BB14	SEC2BB13	SEC2BB12	SEC2BB11	SEC2BB10	SEC2BB9	SEC2BB8	SEC2BB7	SEC2BB6	SEC2BB5	SEC2BB4	SEC2BB3	SEC2BB2	SEC2BB1	SEC2BB0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB0 to 0xBC	Reserved	Reserved																															
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.	NSPRIV	SPRIV
	Reset value																															0	0

Table 70. FLASH 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
0xC8	Reserved	Reserved																															
0xD0	FLASH_ PRIVBB1R1	PRIV1BB31	PRIV1BB30	PRIV1BB29	PRIV1BB28	PRIV1BB27	PRIV1BB26	PRIV1BB25	PRIV1BB24	PRIV1BB23	PRIV1BB22	PRIV1BB21	PRIV1BB20	PRIV1BB19	PRIV1BB18	PRIV1BB17	PRIV1BB16	PRIV1BB15	PRIV1BB14	PRIV1BB13	PRIV1BB12	PRIV1BB11	PRIV1BB10	PRIV1BB9	PRIV1BB8	PRIV1BB7	PRIV1BB6	PRIV1BB5	PRIV1BB4	PRIV1BB3	PRIV1BB2	PRIV1BB1	PRIV1BB0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD4	FLASH_ PRIVBB1R2	PRIV1BB31	PRIV1BB30	PRIV1BB29	PRIV1BB28	PRIV1BB27	PRIV1BB26	PRIV1BB25	PRIV1BB24	PRIV1BB23	PRIV1BB22	PRIV1BB21	PRIV1BB20	PRIV1BB19	PRIV1BB18	PRIV1BB17	PRIV1BB16	PRIV1BB15	PRIV1BB14	PRIV1BB13	PRIV1BB12	PRIV1BB11	PRIV1BB10	PRIV1BB9	PRIV1BB8	PRIV1BB7	PRIV1BB6	PRIV1BB5	PRIV1BB4	PRIV1BB3	PRIV1BB2	PRIV1BB1	PRIV1BB0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD8	FLASH1_ PRIVBBR3	PRIV1BB31	PRIV1BB30	PRIV1BB29	PRIV1BB28	PRIV1BB27	PRIV1BB26	PRIV1BB25	PRIV1BB24	PRIV1BB23	PRIV1BB22	PRIV1BB21	PRIV1BB20	PRIV1BB19	PRIV1BB18	PRIV1BB17	PRIV1BB16	PRIV1BB15	PRIV1BB14	PRIV1BB13	PRIV1BB12	PRIV1BB11	PRIV1BB10	PRIV1BB9	PRIV1BB8	PRIV1BB7	PRIV1BB6	PRIV1BB5	PRIV1BB4	PRIV1BB3	PRIV1BB2	PRIV1BB1	PRIV1BB0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xDC	FLASH_ PRIVBB1R4	PRIV1BB31	PRIV1BB30	PRIV1BB29	PRIV1BB28	PRIV1BB27	PRIV1BB26	PRIV1BB25	PRIV1BB24	PRIV1BB23	PRIV1BB22	PRIV1BB21	PRIV1BB20	PRIV1BB19	PRIV1BB18	PRIV1BB17	PRIV1BB16	PRIV1BB15	PRIV1BB14	PRIV1BB13	PRIV1BB12	PRIV1BB11	PRIV1BB10	PRIV1BB9	PRIV1BB8	PRIV1BB7	PRIV1BB6	PRIV1BB5	PRIV1BB4	PRIV1BB3	PRIV1BB2	PRIV1BB1	PRIV1BB0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE0 to 0xEC	Reserved	Reserved																															
0xF0	FLASH_ PRIVBB2R1	PRIV2BB31	PRIV2BB30	PRIV2BB29	PRIV2BB28	PRIV2BB27	PRIV2BB26	PRIV2BB25	PRIV2BB24	PRIV2BB23	PRIV2BB22	PRIV2BB21	PRIV2BB20	PRIV2BB19	PRIV2BB18	PRIV2BB17	PRIV2BB16	PRIV2BB15	PRIV2BB14	PRIV2BB13	PRIV2BB12	PRIV2BB11	PRIV2BB10	PRIV2BB9	PRIV2BB8	PRIV2BB7	PRIV2BB6	PRIV2BB5	PRIV2BB4	PRIV2BB3	PRIV2BB2	PRIV2BB1	PRIV2BB0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xF4	FLASH_ PRIVBB2R2	PRIV2BB31	PRIV2BB30	PRIV2BB29	PRIV2BB28	PRIV2BB27	PRIV2BB26	PRIV2BB25	PRIV2BB24	PRIV2BB23	PRIV2BB22	PRIV2BB21	PRIV2BB20	PRIV2BB19	PRIV2BB18	PRIV2BB17	PRIV2BB16	PRIV2BB15	PRIV2BB14	PRIV2BB13	PRIV2BB12	PRIV2BB11	PRIV2BB10	PRIV2BB9	PRIV2BB8	PRIV2BB7	PRIV2BB6	PRIV2BB5	PRIV2BB4	PRIV2BB3	PRIV2BB2	PRIV2BB1	PRIV2BB0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xF8	FLASH_ PRIVBB2R3	PRIV2BB31	PRIV2BB30	PRIV2BB29	PRIV2BB28	PRIV2BB27	PRIV2BB26	PRIV2BB25	PRIV2BB24	PRIV2BB23	PRIV2BB22	PRIV2BB21	PRIV2BB20	PRIV2BB19	PRIV2BB18	PRIV2BB17	PRIV2BB16	PRIV2BB15	PRIV2BB14	PRIV2BB13	PRIV2BB12	PRIV2BB11	PRIV2BB10	PRIV2BB9	PRIV2BB8	PRIV2BB7	PRIV2BB6	PRIV2BB5	PRIV2BB4	PRIV2BB3	PRIV2BB2	PRIV2BB1	PRIV2BB0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFC	FLASH_ PRIVBB2R4	PRIV2BB31	PRIV2BB30	PRIV2BB29	PRIV2BB28	PRIV2BB27	PRIV2BB26	PRIV2BB25	PRIV2BB24	PRIV2BB23	PRIV2BB22	PRIV2BB21	PRIV2BB20	PRIV2BB19	PRIV2BB18	PRIV2BB17	PRIV2BB16	PRIV2BB15	PRIV2BB14	PRIV2BB13	PRIV2BB12	PRIV2BB11	PRIV2BB10	PRIV2BB9	PRIV2BB8	PRIV2BB7	PRIV2BB6	PRIV2BB5	PRIV2BB4	PRIV2BB3	PRIV2BB2	PRIV2BB1	PRIV2BB0
	Reset value	0	0	0	0	0	0	0	0	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.

8 Instruction cache (ICACHE)

8.1 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 internal and external memories.

Some specific features like dual master ports, hit-under-miss and critical-word-first refill policy, allow close to zero-wait-state performance in most use cases.

8.2 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 AHB master ports: master1 and master2 ports (outputs to Fast and Slow buses of main AHB bus matrix, 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 memories.
  - Minimal impact on interrupt latency, thanks to dual master
  - Optimal cache line refill thanks to WRAPw bursts of the size of the cache line (32-bit word size, w, aligned on cache line size)
  - n-way 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 Code region, for execution from C-AHB code interface)
- 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

8.3 ICACHE implementation

Table 71. 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 slave interface	32 bits
Data size of AHB fast master1 interface	128 bits
Data size of AHB slow master2 interface	32 bits

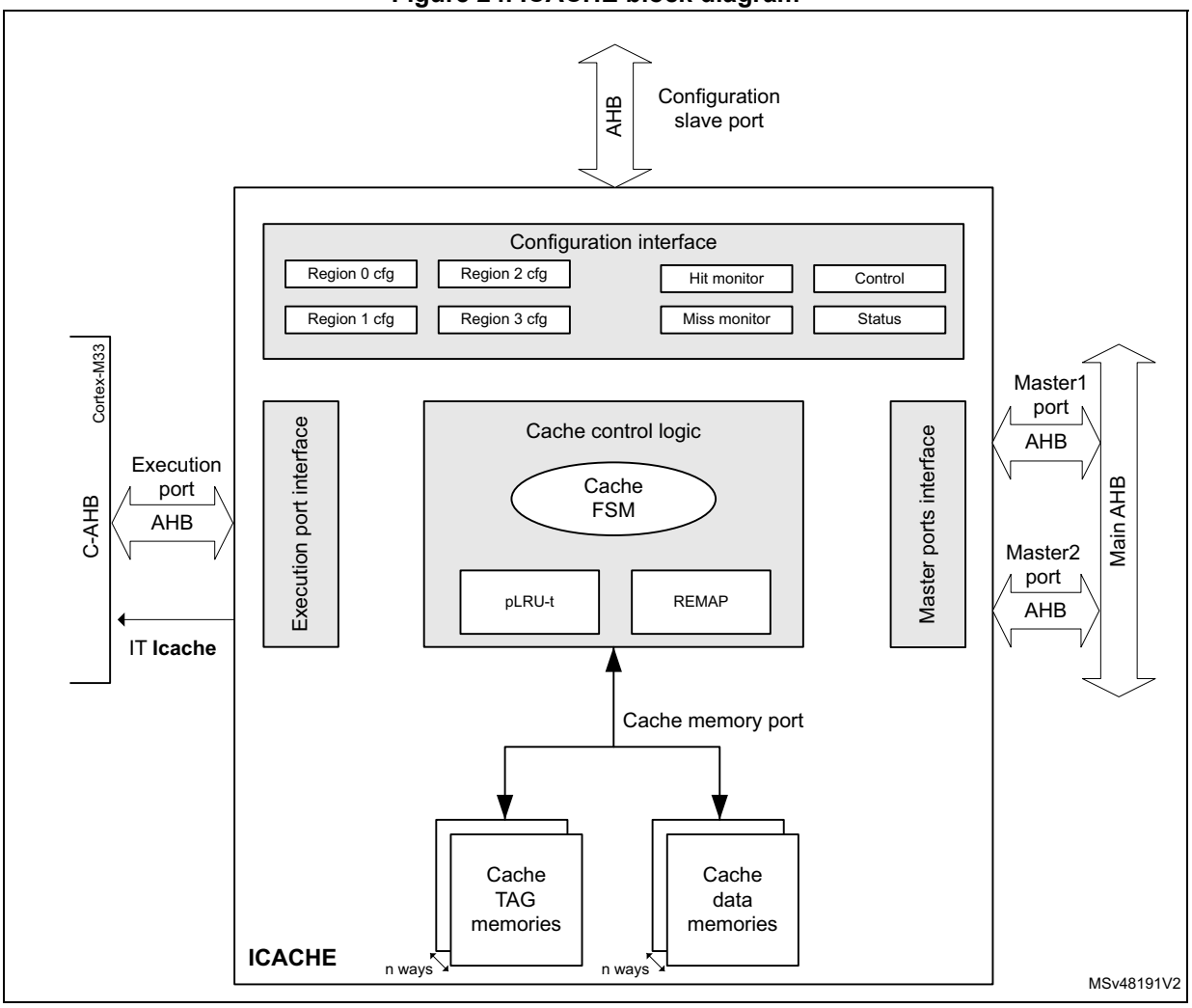
8.4 ICACHE functional description

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.

8.4.1 ICACHE block diagram

Figure 24. ICACHE block diagram



8.4.2 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).

8.4.3 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):

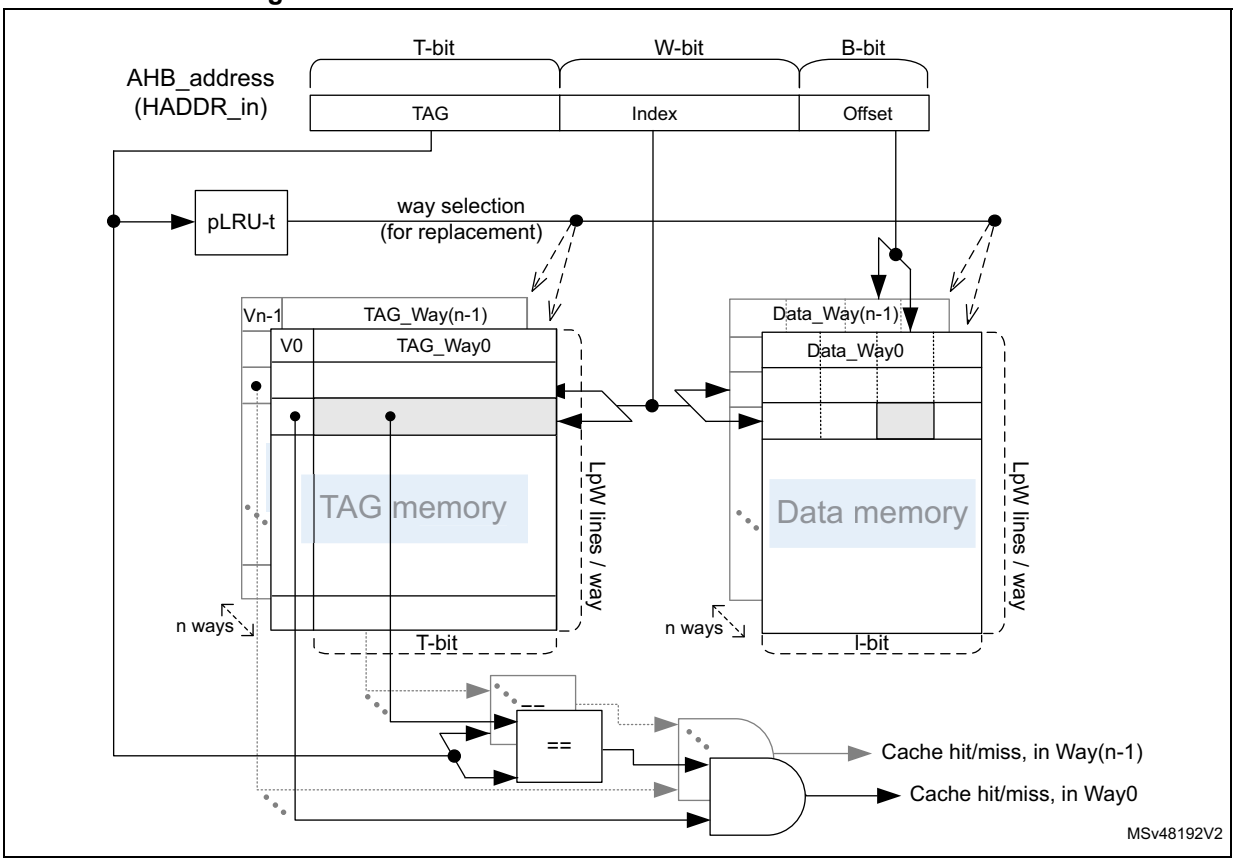
- HADDR_in[B-1:0]: address byte offset, indicates which byte to select inside a cache line.
- HADDR_in[B+W-1:B]: address way index, indicates which cache line to select inside each way.
- HADDR_in[31:B+W]: tag address, to be compared to TAG memory address to check if the requested data is already available (meaning valid) inside the ICACHE.

The table below gives a summary of ICACHE main parameters for TAG memory dimensioning and *Figure 25* shows the functional view of TAG and data memories, for a n-way set associative ICACHE.

Table 72. 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 25. ICACHE TAG and data memories functional view



8.4.4 Direct mapped ICACHE (1-way cache)

Default configuration (at reset) is n-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 73. 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 8.4.8*), since only one way (the unique one) is possible for any data refill.

8.4.5 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 26*).

ICACHE is by default disabled at boot.

8.4.6 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 the definition of 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 bus matrix. 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 8.4.7*.

An incoming memory request to ICACHE is defined as cacheable according to its AHB transaction memory lookup attribute, as shown in *Table 74*. This AHB attribute depends on the MPU programming for the addressed region.

Table 74. 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 8.4.7*).

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 8.4.8*.

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.

The table below summarizes product memories programmable configurations.

Table 75. 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 code in external memories fetched on C-AHB bus (else on S-AHB bus)

8.4.7 Address remapping

ICACHE allows an alias address to be defined 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.

The number of regions depends on the ICACHE hardware configuration embedded in a given product. 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.

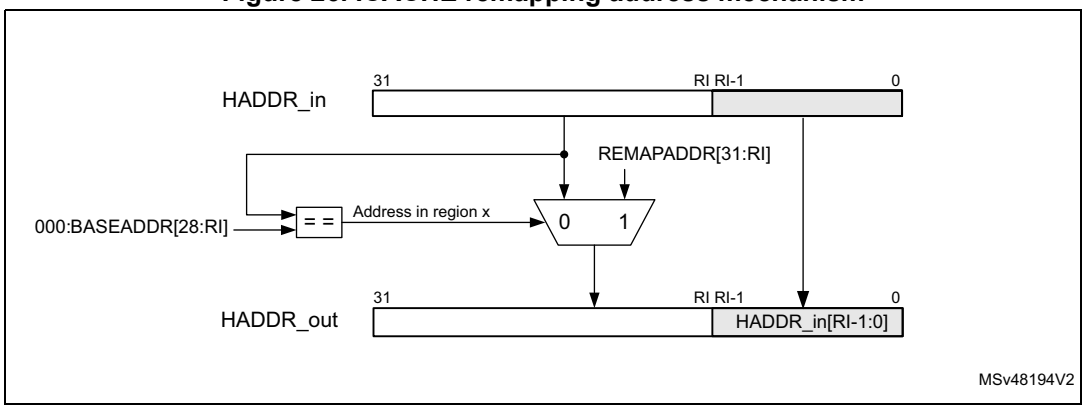
Remap regions can have different size: each region size can be programmed in the RSIZE field of its ICACHE_CRRx register. The size of each region is a power of two multiple of range granularity (2 Mbytes), with a minimum region size of 2 Mbytes and a maximum region size of 128 Mbytes.

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 26. 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 76. 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

Table 76. ICACHE remap region size, base address and remap address (continued)

Region size (Mbytes)	Base address size (MSBs)	Remap address (MSBs)
32	4	7
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 76*), 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 bus matrix (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 subregions.*

8.4.8 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).

  The burst generated on ICACHE master bus is WRAPw (w being the cache line width, in words) in case no address remap occurs. 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 bus matrix 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 n can be used (the line pointed by the address index, in each of the n 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.

8.4.9 Dual master cache

ICACHE can implement a dual port AHB master on main AHB bus matrix: master1 and master2 ports. This is used 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) memories on master2 port (Slow bus)

For systems not implementing external memories, it is possible to decouple the traffic to the internal FLASH memory from the traffic to the internal SRAM (when remapped by the ICACHE). This feature is used 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.

8.4.10 ICACHE security

ICACHE implements a v8-M TrustZone as defined by Arm.

ICACHE configuration registers are protected at system level.

8.4.11 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.

8.4.12 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 cacheable AHB-transactions on slave cache port that hit ICACHE content.

  The hit monitor 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 8.4.8*).
- The miss monitor counts the cacheable AHB-transactions on slave cache port that miss ICACHE content.

  The miss monitor 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.

8.4.13 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).

8.5 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 25*). 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 code execution, it is a little less efficient than with a n-way set associative cache mode.

8.6 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 77. 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.

8.7 ICACHE registers

8.7.1 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

8.7.2 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

8.7.3 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.

8.7.4 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.

8.7.5 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

8.7.6 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

8.7.7 ICACHE region x configuration register (ICACHE_CRRx)

Address offset: 0x020 + 4 * x, (x = 0 to 3)

Reset value: 0x0000 0200

Define an alias address in Code region for other regions, making them cacheable.

BASEDADDR and REMAPADDR fields are write locked (read only) when EN = 1 in ICACHE_CR.

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 8.4.7*). 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 8.4.7*). If the programmed value has more LSBs, the useless bits are ignored.

8.7.8 ICACHE register map

Table 78. 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
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																															

Table 78. 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
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.

9 Data cache (DCACHE)

9.1 Introduction

The data cache (DCACHE) is introduced on S-AHB system bus of Cortex-M33 processor to improve the performance of data traffic to/from external memories.

Some specific features like hit-under-miss and critical-word-first refill policy allow optimum performance on external memories data accesses.

9.2 DCACHE main features

The main features of DCACHE are described below:

- Bus interface
  - one 32-bit AHB slave port, the system port (input from Cortex-M33 S-AHB system interface)
  - one 32-bit AHB master port (output to main AHB bus matrix)
  - one 32-bit AHB slave port for control (input from AHB peripherals interconnect, for DCACHE 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
  - Optimized 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
  - Supports both write-back and write-through policies (selectable with AHB bufferable attribute)
  - Read and write-back always allocate
  - Write-through always non-allocate (write-around)
  - Supports byte, half-word and word writes
- 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 for read transactions, minimizing processor stalls
  - Possibility to configure burst type of all AHB memory transactions: INCRw or WRAPw (size w aligned on cache line size)
- Performance counters

  DCACHE implements four performance counters:
  - Two hit monitor counters (32-bit): number of read hits, number of write hits
  - Two miss monitor counters (16-bit): number of read misses, number of write misses

- Error management
  - Possibility to detect error for master port request initiated by DCACHE itself (a cache line written back into main memory, because of an eviction or a clean operation), to flag this error, and optionally to raise an interrupt
- TrustZone security support
- Maintenance operations
  - Cache invalidate: full cache invalidation, fast command, non interruptible
  - Cache invalidate range: invalidates cache lines (reset valid bit = 0) whose address belongs to defined range, background task, interruptible
  - Cache clean range: cleans cache lines (if dirty bit = 1, write back line, then clear dirty bit) whose address belongs to defined range, background task, interruptible
  - Cache clean and invalidate range: cleans and invalidates cache lines (if dirty bit = 1, write back line, then clear valid bit) whose address belongs to defined range, background task, interruptible

9.3 DCACHE implementation

DCACHE1 is placed on Cortex-M33 S-AHB bus and caches only the external RAM memory region (OCTOSPI and FMC), in address range [0x6000 0000:0x9FFF FFFF] of the memory map.

Indeed, by placing a bus matrix demultiplexing node in front of DCACHE1, S-AHB bus memory requests addressing SRAM region or peripherals region (respectively in ranges [0x2000 0000:0x3FFF FFFF] and [0x4000 0000:0x5FFF FFFF]) are routed directly to the main AHB bus matrix and DCACHE1 is bypassed.

Table 79. DCACHE features

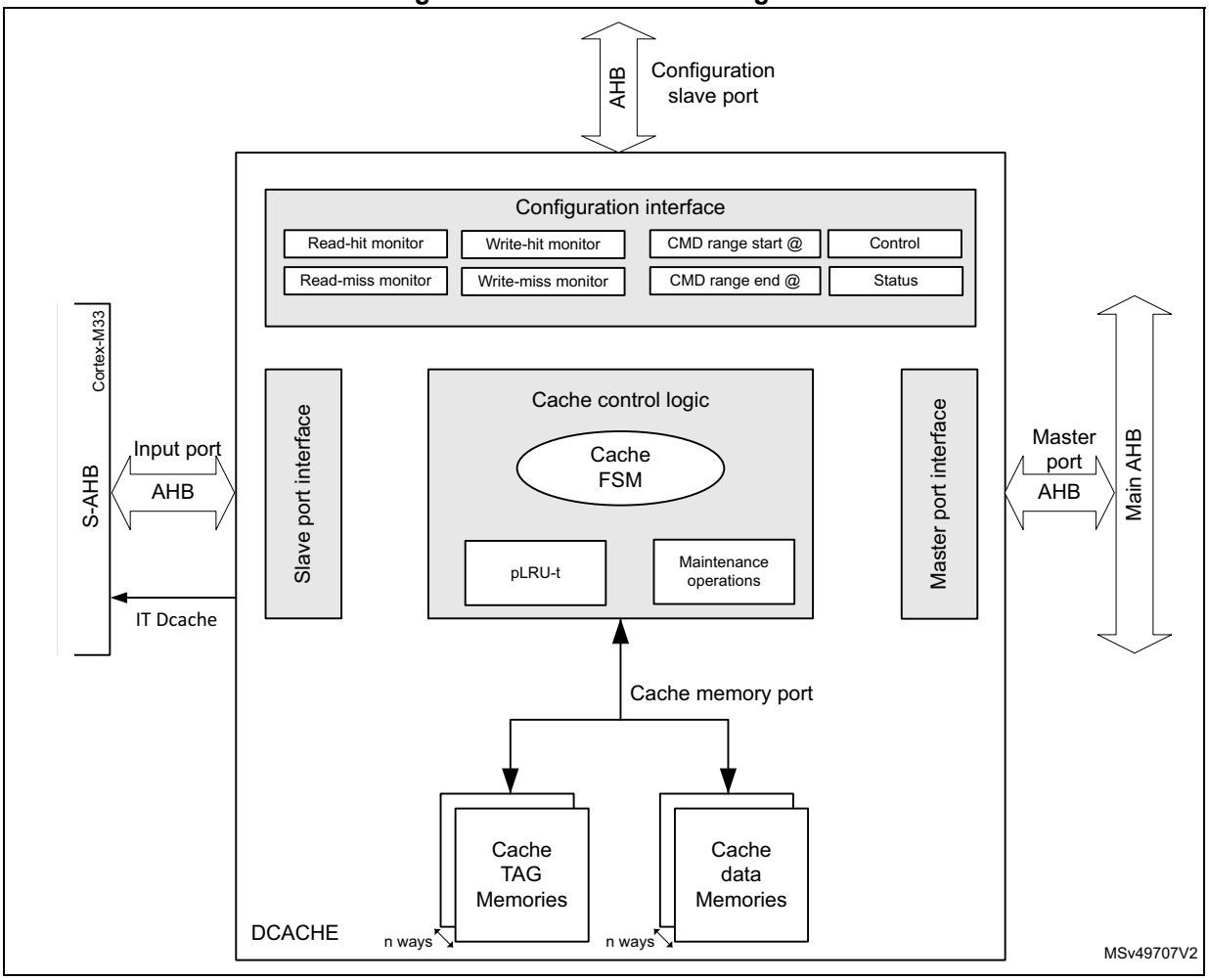
Features	DCACHE1
Number of ways	2
Cache size	4 Kbytes
Cache line width	16 bytes
Data size of AHB Master interface	32 bits

9.4 DCACHE functional description

The purpose of the data cache is to cache external memories data loads and stores coming from the processor. These accesses include the instruction fetches that may occur at an external memory address. As such DCACHE manages both read and write transactions.

9.4.1 DCACHE block diagram

Figure 27. DCACHE block diagram



9.4.2 DCACHE reset and clocks

DCACHE is clocked on the Cortex-M33 S-AHB bus clock.

When the DCACHE reset signal is released, a cache invalidate procedure is automatically launched, making DCACHE busy (DCACHE_SR = 0x0000 0001).

When this procedure is finished:

- DCACHE is invalidated: “cold cache”, with all cache line valid, dirty and privilege bits = 0 (DCACHE must be filled up)
- DCACHE_SR = 0x0000 0002 (reflecting the cache is no more busy)
- DCACHE is disabled: the EN bit in DCACHE_CR holds its reset state (= 0).

Note: When disabled, DCACHE is bypassed: the slave input requests are just forwarded to the master port.

9.4.3 DCACHE TAG memory

The DCACHE TAG memory contains:

- address tags, that indicate which data are contained in the cache data memories
- validity bits
- dirty bits
- privilege bits

There is one valid bit, one dirty bit and one privilege bit per cache line (per way).

The valid bit enables/disables access to data cache line: if the line is not valid, the data access (read or write) is performed in main memory.

Valid bit is set when cache line is written (refilled by either a read miss or a write-back miss).

Valid bits are reset in any of the below cases:

- after DCACHE reset is released
- when cache is disabled, by setting the EN bit low in DCACHE_CR register (by software)
- when executing one of the DCACHE invalidate commands, setting by software the CACHEINV bit high or CACHECMD = 0b010 or 0b011 in DCACHE_CR register (see *Section 9.4.8*).

The dirty bit indicates that the cache line has up-to-date values with respect to main memory content (in other words, cache has last right value, main memory is not up to date).

Dirty bit is set when cache line is written by a slave port write transaction (only in case of an access with write-back attribute).

Dirty bits are reset in any of the below cases:

- after DCACHE reset is released
- when a line refill is performed on a read miss (on a write-back miss, the refilled cache line is modified by the written data and dirty bit = 1)
- when cache invalidation is performed
- when executing one of the DCACHE clean operations (cache line written back to the main memory), setting by software CACHECMD = 0b001 or 0b011 in DCACHE_CR register (see *Section 9.4.8*).

The privilege bit indicates if the data is managed by a privileged entity. It is assigned according to the value of AHB privileged attribute at input slave port, for the first access to this line (it is written only during the line refill, on read miss or write-back miss). The privilege bit holds same polarity as the privileged attribute: 1 for privileged access, 0 for unprivileged access.

Privilege bits are reset when cache is invalidated and after DCACHE reset is released.

When a cacheable transaction is received at input slave port, its AHB address (HADDR_in) is split into the following fields (see the table below for B and W values):

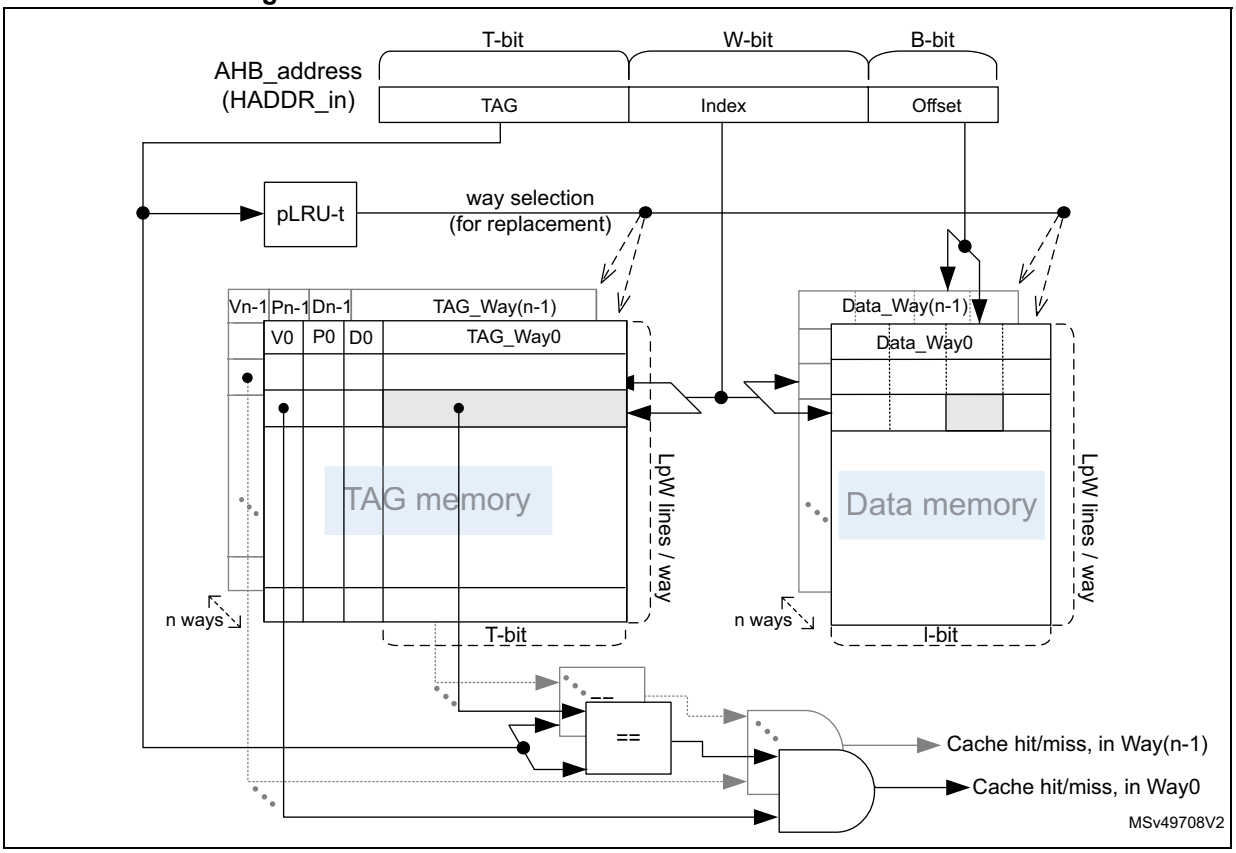
- HADDR_in[B-1:0]: address byte offset, indicates which byte to select inside a cache line.
- HADDR_in[B+W-1:B]: address way index, indicates which cache line to select inside each way.
- HADDR_in[31:B+W]: tag address, to be compared to TAG memory address to check if the requested data is already available (meaning valid) inside the DCACHE.

The table below gives DCACHE main parameters for TAG memory dimensioning. *Figure 28.* shows the functional view of TAG and data memories, for a n-way set associative DCACHE.

Table 80. TAG memory dimensioning parameters

Parameter	Value	Example
Cache size	S Kbytes = s bytes (s = 1024 x S)	4 Kbytes = 4096 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	128 lines/way
Address byte offset size	B = log2(L) bit	4-bit
Address way index size	W = log2(LpW) bit	7-bit
TAG address size	T = (32 - W - B) bit	21-bit

Figure 28. DCACHE TAG and data memories functional view



9.4.4 DCACHE enable

In order to activate the DCACHE functioning, the EN bit must be set in DCACHE_CR register.

When DCACHE is disabled, it is bypassed and all transactions are copied from slave port to master port in the same clock cycle, and no comparison is performed with TAG address.

DCACHE is by default disabled at boot.

9.4.5 Cacheable and non-cacheable traffic

DCACHE is developed for Cortex-M33 core and caches the memory regions addressed by the AHB bus connected to it.

In addition, the AHB bus traffic to the memory regions can be cacheable or non-cacheable. An incoming memory request to DCACHE is defined as cacheable according to its AHB transaction memory lookup attribute.

In case of write transaction, the DCACHE write policy is defined as write-through or write-back according to its AHB transaction memory bufferable attribute (see the table below).

These AHB attributes depend on the memory protection unit (MPU) programming for the addressed region.

Table 81. DCACHE cacheability for AHB transaction

AHB lookup attribute	AHB bufferable attribute	Cacheability
0	x	Read and write: non cacheable
1	0	Read: cacheable Write: (cacheable) write-through
1	1	Read: cacheable Write: (cacheable) write-back

In case of non cacheable access, DCACHE is bypassed, meaning that the AHB transaction is propagated unchanged to the master output port.

The bypass does not increase the latency of the access to the targeted memory.

In case of cacheable access, DCACHE behaves as explained in *Section 9.4.6*.

Cacheable memory regions are defined and programmed by the user in the MPU, responsible for the generation of the AHB attribute signals for any transaction addressing a given region.

9.4.6 Cacheable accesses

When DCACHE receives a cacheable transaction from Cortex-M33, on its slave port, DCACHE checks if the address requested is present in its TAG memory and if the corresponding cache line is valid.

For **read transaction**, there are then three alternatives:

- Address is present inside the TAG memory, cache line is valid: **cache read 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 read miss**, the data is read from main memory and provided to the processor, and a cache line refill is performed.

  The critical-word-first refill 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).

  The kind of burst generated on DCACHE master bus depends on the HBURST bit programmed in DCACHE_CR register: either INCRw or WRAPw (w being the cache line width, in words).

  The AHB transaction attributes are also propagated from the slave input (missing) request to the master output refill request.
- Address is not present in the TAG memory but belongs to the refill burst from main memory that is currently ongoing: **cache read hit** as well (hit-under-miss feature).

  Whatever the line refill is due to a read or write (missing) transaction, DCACHE is capable of providing the requested read data as soon as the data is available at its master interface, thus avoiding a miss (with data fetch from main memory).

For **write-back transaction** (write transaction, with write-back bufferable attribute), there are three alternatives as well:

- Address is present inside the TAG memory, cache line is valid: **cache write-back hit**, the data is written in cache.
- Address is not present in the TAG memory (or cache line is not valid): **cache write-back miss**.

  First, a line allocation is performed by reading the entire cache line data from main memory. The kind of burst generated on DCACHE master bus for this line refill depends on the HBURST bit programmed in DCACHE_CR register: either INCRw or WRAPw (w being the cache line width, in words), and the AHB transaction attributes are propagated from the slave port initial request.

  Once the refilled line is written in DCACHE, the initial data provided on slave port is written in this DCACHE line (it overwrites the data part of the cache line that was refilled just before).
- Address is not present in the TAG memory but belongs to the refill burst from main memory that is currently ongoing: **cache write-back hit** as well (hit-under-miss feature).

  Whatever the line refill is due to a read or write (missing) transaction, DCACHE is capable of writing incoming data directly inside the refilled line, thus avoiding a miss (with refill from main memory).

For **write-through transaction** (write transaction, with write-through bufferable attribute), only two alternatives exist:

- Address is present inside the TAG memory, cache line is valid: **cache write-through hit**, the data is written both in cache and in main memory (through master port).
- Address is not present in the TAG memory (or cache line is not valid): **cache write-through miss**, the data incoming at slave port is written only in main memory (unlike the write-back miss, there is no line allocation and data written in cache).

In case of cache refill (due to cache miss), DCACHE selects which cache line is written with the refill data: as a 2-way set associative cache, 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 evicted first:

- if the dirty tag of this line equals 0 (clean data), the line is simply invalidated.
- if it equals 1 (dirty data), the line must be written back in the main memory.

  In such a case, DCACHE generates a burst write transaction on its master port, with burst type set to INCRw (w being the cache line width, in words), and with AHB memory transaction attribute signals set as below:

  – data (not instruction)
  – privileged = TAG privilege bit
  – write-back (even if it does not care)
  – normal memory
  – cacheable
  – allocate (even if it does not care)
  – non shareable

  These AHB attributes cannot be propagated from the slave port (as it is the case for all other transactions emitted by DCACHE) because the evicting transaction has no relation with the initial missing transaction. Note this AHB attributes setting is fixed, excepted the privileged bit that is copied from the TAG privilege bit of the evicted line.

9.4.7 DCACHE security

DCACHE implements a v8-M TrustZone as defined by Arm.

DCACHE configuration registers are protected at system level.

9.4.8 DCACHE maintenance

DCACHE features several maintenance operations that the software can programmed in DCACHE_CR control register:

- **Full invalidate**: invalidates the whole cache, non interruptible task.

  The software can invalidate the whole content of DCACHE by programming the CACHEINV bit in DCACHE_CR register.

  When CACHEINV is set, the DCACHE control logic sets the BUSYF flag in DCACHE_SR status register, and performs the operation of cache invalidation, reseting each TAG valid bit to 0 (one valid bit per cache line). Each dirty and privilege bits are also reset during cache invalidation to prevent unknown values at next cache line validation. The CACHEINV bit of DCACHE_CR is automatically cleared.

  Once the full invalidate operation is finished, DCACHE automatically clears the BUSYF flag and sets the BSYENDF flag in DCACHE_SR register.

  If enabled on this flag condition (BSYENDIE = 1 in DCACHE_IER), the **DCACHE** interrupt is raised. Then, the (empty) cache is available again.

  This full invalidate operation is not interruptible, meaning that cache does not treat any cacheable request while BUSYF flag is set. However, non-cacheable traffic is treated (since the request address is not compared to TAG ones), DCACHE being bypassed in the same clock cycle (same behavior as when DCACHE is disabled).

- **Invalidate range**: invalidates a certain range of addresses in the cache, back-ground task (interruptible).

  The software can invalidate a given data region in DCACHE by programming DCACHE_CR register with bits STARTCMD = 1 and CACHECMD = 0b010, after the address range was programmed into DCACHE_CMDRSADDRR (range start address) and DCACHE_CMDREADDRR (range end address).

  DCACHE control logic then parses the whole TAG memory. If the read line address (TAG Address + Line index) falls in the programmed address range (DCACHE_CMDRSADDRR ≤ Line Addr ≤ DCACHE_CMDREADDRR), the corresponding cache line is invalidated (line TAG bits cleared, i.e. Valid bit = Dirty bit = Privilege bit = 0).

  When STARTCMD is set, the DCACHE control logic sets the BUSYCMDF flag in DCACHE_SR and launches the invalidate range operation. The STARTCMD bit of DCACHE_CR is also automatically cleared.

  Once the operation is finished (all TAG memory parsed), DCACHE automatically clears the BUSYCMDF flag and sets the CMDENDF flag in DCACHE_SR register.

  If enabled on this flag condition (CMDENDIE = 1 in DCACHE_IER), the **DCACHE** interrupt is raised.

  During this invalidate range operation, DCACHE is interruptible, meaning it can accept new incoming requests that take higher priority than the invalidation process. In other words, the TAG memory is accessed for invalidate range operation only if not already accessed by an external cache request. This implies that invalidate range execution is usually not performed in one go, but can be interrupted.

- **Clean range**: cleans a certain range of addresses in the cache, back-ground task (interruptible).

  Cleaning a cache line means making sure that main memory content is up-to-date with the data which might have been modified in cache. So the clean operation consists in performing the write-back in main memory of the cache lines that are tagged as “dirty” (the ones with TAG dirty bit set).

  The software can clean a given data region in DCACHE by programming DCACHE_CR register with bits STARTCMD = 1, and CACHECMD = 0b001, after the address range was programmed into DCACHE_CMDRSADDRR (range start address) and DCACHE_CMDREADDRR (range end address).

  DCACHE control logic then parses the whole TAG memory. If the read line address (TAG Address + Line index) falls in the programmed address range (DCACHE_CMDRSADDRR ≤ Line Addr ≤ DCACHE_CMDREADDRR) and the corresponding line is dirty, this line is cleaned, meaning the whole cache line is written-back in memory through the DCACHE master port and its TAG dirty bit is cleared.

  When STARTCMD is set, the DCACHE control logic sets the BUSYCMDF flag in DCACHE_SR and launches the clean range operation. The STARTCMD bit of DCACHE_CR is also automatically cleared.

  Once the operation is finished (all TAG memory parsed), DCACHE automatically clears the BUSYCMDF flag and sets the CMDENDF flag in DCACHE_SR register.

  If enabled on this flag condition (CMDENDIE = 1 in DCACHE_IER), the **DCACHE** interrupt is raised.

  During this clean range operation, DCACHE is interruptible, meaning it can accept new incoming requests that take higher priority than the cleaning process. In other words, the TAG memory is accessed for clean range operation only if not already accessed by

an external cache request. This implies that clean range execution is usually not performed in one go, but can be interrupted.

It is under the software responsibility that no bus initiator attempts to change the content of the region being cleaned until clean range is completed. For that, the software should take advantage of BUSYCMDF flag in DCACHE_SR and polls this flag to prevent any spurious access to the area being cleaned. Alternatively it can also rely on the command end flag (CMDENDF) or on the **DCACHE** interrupt to detect the end of the clean range execution.

- **Clean and invalidate range**: cleans and invalidates a certain range of addresses in the cache, back-ground task (interruptible).

  Same as clean range, with an additional reset of its TAG valid bit when a cache line is cleaned.

  The software can launch this clean and invalidate range operation, by programming DCACHE_CR register with bits STARTCMD = 1, and CACHECMD = 0b011, after the address range was programmed into DCACHE_CMDRSADDRR (range start address) and DCACHE_CMDREADDRR (range end address).

  It sets and clears the same flags, and potentially the same interrupt as invalidate range or clean range operations.

9.4.9 DCACHE performance monitoring

DCACHE provides four monitors for performance analysis: two 32-bit hit monitors and two 16-bits miss monitors.

- The read-hit and the write-hit monitors count the AHB transactions at the input of DCACHE (slave port) that do not generate a transaction on DCACHE output (master 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 9.4.6*).

- The read-miss and the write-miss monitors count the AHB transactions at the input of DCACHE (slave port) that generate a transaction on DCACHE output (master 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.

The software can perform the following tasks:

- Enable/stop the read (write) hit monitor, through the R(W)HITMEN bit in DCACHE_CR.
- Reset the read (write) hit monitor, by setting the R(W)HITMRST bit in DCACHE_CR.
- Enable/stop the read (write) miss monitor, through the R(W)MISSMEN bit in DCACHE_CR.
- Reset the read (write) miss monitor, by setting the R(W)MISSMRST bit in DCACHE_CR.

To reduce power consumption, these monitors are disabled (stopped) by default.

9.4.10 DCACHE boot

DCACHE is disabled (EN = 0 in DCACHE_CR) at boot.

Once boot is finished, DCACHE can be enabled (software setting the EN bit to 1 in DCACHE_CR).

9.5 DCACHE low-power modes

At product level, using DCACHE reduces the power consumption by loading/storing data from/to the internal DCACHE most of the time, rather than from the bigger and then more power consuming main memories. This reduction is even much higher, since the cached main memories are external.

9.6 DCACHE error management and interrupts

A transaction initiated on the DCACHE master port may return an error (a write attempt into a read-only memory, for instance). If the master port request was propagated from a slave port request, the error is propagated back to the slave port. If ever the master port request is initiated by DCACHE itself (a cache line is written back into the main memory because of an eviction or a clean operation), DCACHE receives this functional error and flags it internally by setting the ERRF flag in DCACHE_SR.

In such a case, an interrupt is generated if the corresponding interrupt enable bit is set (ERRIE = 1 in DCACHE_IER).

Another case of potential interrupt generation is at the end of a full invalidate operation: when the cache busy state is finished, DCACHE sets the BSYENDF flag in DCACHE_SR.

An interrupt is then generated if the corresponding interrupt enable bit is set (BSYENDIE = 1 in DCACHE_IER).

Last case is at the end of invalidate and/or clean range operations: when the command busy state is finished, DCACHE sets the CMDENDF flag in DCACHE_SR.

An interrupt is also generated if the corresponding interrupt enable bit is set (CMDENDIE = 1 in DCACHE_IER).

All DCACHE interrupts use the same **DCACHE** interrupt vector.

Table 82. DCACHE interrupts

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method
DCACHE	Functional error	ERRF flag in DCACHE_SR	ERRIE bit in DCACHE_IER	Set CERRF bit to 1 in DCACHE_FCR
	End of busy state (full invalidate finished)	BSYENDF flag in DCACHE_SR	BSYENDIE bit in DCACHE_IER	Set CBSYENDF bit to 1 in DCACHE_FCR
	End of cache operations (address range based)	CMDENDF flag in DCACHE_SR	CMDENDIE bit in DCACHE_IER	Set CCMDENDF bit to 1 in DCACHE_FCR

DCACHE also propagates all AHB bus errors (such as security issues, address decoding issues) from master port back to the S-AHB slave port.

9.7 DCACHE registers

9.7.1 DCACHE control register (DCACHE_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HBURST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WMISSMRST	WHITMRST	WMISSMEN	WHITMEN	RMISSMRST	RHITMRST	RMISSMEN	RHITMEN
rw								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.	STARTCMD	CACHECMD[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	CACHEINV	EN
				w	rw	rw	rw							w	rw

Bit 31 **HBURST:** output burst type for cache master port read accesses
Can be set by software, only when EN = 0.
Master port write accesses are always done in INCR burst type.
0: WRAP
1: INCR

Bits 30:24 Reserved, must be kept at reset value.

Bit 23 **WMISSMRST:** write-miss monitor reset
0: no effect
1: reset cache write-miss monitor

Bit 22 **WHITMRST:** write-hit monitor reset
0: no effect
1: reset cache write-hit monitor

Bit 21 **WMISSMEN:** write-miss monitor enable
0: cache write-miss monitor switched off. Stopping the monitor does not reset it.
1: cache write-miss monitor enabled

Bit 20 **WHITMEN:** write-hit monitor enable
0: cache write-hit monitor switched off. Stopping the monitor does not reset it.
1: cache write-hit monitor enabled

Bit 19 **RMISSMRST:** read-miss monitor reset
0: no effect
1: reset cache read-miss monitor

Bit 18 **RHITMRST:** read-hit monitor reset
0: no effect
1: reset cache read-hit monitor

Bit 17 **RMISSMEN:** read-miss monitor enable
0: cache read-miss monitor switched off. Stopping the monitor does not reset it.
1: cache read-miss monitor enabled

Bit 16 **RHITMEN:** read-hit monitor enable
0: cache read-hit monitor switched off. Stopping the monitor does not reset it.
1: cache read-hit monitor enabled

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **STARTCMD:** starts maintenance command (maintenance operation defined in CACHECMD).
Can be set by software, only when EN = 1, BUSYCMDF = 0, BUSYF = 0 and CACHECMD = 0b001, 0b010 or 0b011.
Cleared by hardware when the BUSYCMDF flag is set (during cache maintenance operation). Writing 0 has no effect.
0: command operation (cache maintenance) finished
1: start maintenance command (cache maintenance)

Bits 10:8 **CACHECMD[2:0]:** cache command maintenance operation (cleans and/or invalidates an address range)
Can be set and cleared by software, only when no maintenance command is ongoing (BUSYCMDF = 0).
000: no operation
001: clean range
010: invalidate range
011: clean and invalidate range
others: reserved

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

Bit 1 **CACHEINV:** full cache invalidation
Can be set by software, only when EN = 1.
Cleared by hardware when the BUSYF flag is set (during full cache invalidation 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

9.7.2 DCACHE status register (DCACHE_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.	CMDENDF	BUSYCMDF	ERRF	BSYENDF	BUSYF
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CMDENDF:** command end flag
Cleared by writing DCACHE_FCR.CCMDENDF = 1.
0: cache busy or in idle
1: CACHECMD command finished

Bit 3 **BUSYCMDF:** command busy flag
0: cache not busy on a CACHECMD command
1: cache busy on a CACHECMD command (clean and/or invalidate an address range)

Bit 2 **ERRF:** cache error flag
Cleared by writing DCACHE_FCR.CERRF = 1.
0: no error
1: an error occurred during the operation (eviction or clean operation write-back error).

Bit 1 **BSYENDF:** full invalidate busy end flag
Cleared by writing DCACHE_FCR.CBSYENDF = 1.
0: cache busy or in idle
1: full invalidate CACHEINV operation finished

Bit 0 **BUSYF:** full invalidate busy flag
0: cache not busy on a CACHEINV operation
1: cache executing a full invalidate CACHEINV operation

9.7.3 DCACHE interrupt enable register (DCACHE_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.	CMDENDIE	Res.	ERRIE	BSYENDIE	Res.
											rw		rw	rw	

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CMDENDIE**: interrupt enable on command end

Set by software to enable an interrupt generation at the end of a cache command (clean and/or invalidate an address range)

0: interrupt disabled on command end

1: interrupt enabled on command end

Bit 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 (eviction or clean operation write-back error)

0: interrupt disabled on error

1: interrupt enabled on error

Bit 1 **BSYENDIE:** interrupt enable on busy end

Set by SW to enable an interrupt generation at the end of a cache full invalidate operation.

0: Interrupt disabled on busy end

1: Interrupt enabled on busy end

Bit 0 Reserved, must be kept at reset value.

9.7.4 DCACHE flag clear register (DCACHE_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.	CCMDENDF	Res.	CERRF	CBSYENDF	Res.
											w		w	w	

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CCMDENDF:** clear command end flag
Set by software.
0: no effect
1: clears CMDENDF flag in DCACHE_SR

Bit 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 DCACHE_SR

Bit 1 **CBSYENDF:** clear full invalidate busy end flag
Set by software.
0: no effect
1: clears BSYENDF flag in DCACHE_SR

Bit 0 Reserved, must be kept at reset value.

9.7.5 DCACHE read-hit monitor register (DCACHE_RHMONR)

Address offset: 0x010

Reset value: 0x0000 0000

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

Bits 31:0 **RHITMON[31:0]:** cache read-hit monitor counter

9.7.6 DCACHE read-miss monitor register (DCACHE_RMMONR)

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
RMISSMON[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 **RMISSMON[15:0]:** cache read-miss monitor counter

9.7.7 DCACHE write-hit monitor register (DCACHE_WHMONR)

Address offset: 0x020

Reset value: 0x0000 0000

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

Bits 31:0 **WHITMON[31:0]:** cache write-hit monitor counter

9.7.8 DCACHE write-miss monitor register (DCACHE_WMMONR)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WMISSMON[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 **WMISSMON[15:0]:** cache write-miss monitor counter

9.7.9 DCACHE command range start address register (DCACHE_CMDRSADDRR)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMDSTARTADDR[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
CMDSTARTADDR[15:4]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:4 **CMDSTARTADDR[31:4]:** start address of range to which the cache maintenance command specified in DCACHE_CR.CACHECMD field applies

This register must be set before DCACHE_CR.CACHECMD is written.

.

Bits 3:0 Reserved, must be kept at reset value.

9.7.10 DCACHE command range end address register (DCACHE_CMDREADDRR)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMDENDADDR[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
CMDENDADDR[15:4]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:4 **CMDENDADDR[31:4]:** end address of range to which the cache maintenance command specified in DCACHE_CR.CACHECMD field applies

This register must be set before DCACHE_CR.CACHECMD is written.

Bits 3:0 Reserved, must be kept at reset value.

9.7.11 DCACHE register map

Table 83. DCACHE 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	**DCACHE_CR**	HBURST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WMISSMRST	WHITMRST	WMISSMEN	WHITMEN	RMISSMRST	RHITMRST	RMISSMEN	RHITMEN	Res.	Res.	Res.	Res.	STARTCMD	CACHECMD [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	CACHEINV	EN
	Reset value	0								0	0	0	0	0	0	0	0					0	0	0	0							0	0
0x004	**DCACHE_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.	CMDENDF	BUSYCMDF	ERRF	BSYENDF	BUSYF
	Reset value																												0	0	0	0	1
0x008	**DCACHE_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.	CMDENDIE	Res.	ERRIE	BSYENDIE	Res.
	Reset value																												0		0	0	

Table 83. DCACHE 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	**DCACHE_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.	CCMDENDF	Res.	CERRF	CBSYENDF	Res.
	Reset value																												0		0	0	
0x010	**DCACHE_RHMONR**	RHITMON[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	**DCACHE_RMMONR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RMISSMON[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	**DCACHE_WHMONR**	WHITMON[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	**DCACHE_WMMONR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WMISSMON[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	**DCACHE_CMDRSADDRR**	CMDSTARTADDR[31:4]																												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				
0x02C	**DCACHE_CMDREADDRR**	CMDENDADDR[31:4]																												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				

Refer to *Section 2.3* for the register boundary addresses.

10 Power control (PWR)

10.1 Introduction

The power controller manages all device power supplies and power modes transitions.

10.2 PWR main features

The power controller (PWR) main features are:

- Power supplies and supply domains
  - Core domain (V_{CORE})
  - V_{DD} domain
  - Backup domain
  - Analog domain (V_{DDA})
  - Supply for the SMPS power stage (available on SMPS packages)
  - V_{DDIO2} domain on port PG[15:2]
  - V_{DDUSB} for USB transceiver
- System supply voltage regulation
  - SMPS step-down converter
  - Linear voltage regulator (LDO)
- Power supply supervision
  - BOR monitor
  - PVD monitor
  - PVM monitor (V_{DDA}, V_{DDUSB}, V_{DDIO2})
  - Out of functional range temperature monitor
  - Out of functional range Backup domain voltage monitor
- Power management
  - Operating modes
  - Voltage scaling control
  - Low-power modes
- V_{BAT} battery charging
- TrustZone security and privileged protection

10.3 PWR pins and internal signals

Table 84. PWR input/output pins

Pin name	Signal type	Description
VDD	Supply	Main supply
GND	Supply	Main ground
VDDA	Supply	Analog peripherals supply
VSSA	Supply	Analog peripherals ground
VDDIO2	Supply	Independent I/O supply
VDDUSB	Supply	USB supply
VDD11 (packages with SMPS)/ VCAP (packages without SMPS)	Supply	Logic supply (V_{CORE})
VBAT	Supply	Backup domain supply
VDDSMPS	Supply	SMPS supply
VSSSMPS	Supply	SMPS ground
VLXSMPS	Supply	SMPS output
VREF+	Supply	ADC/DAC high reference voltage
VREF-	Supply	ADC/DAC low reference voltage
WKUPx (x = 1 to 8)	Input	Wakeup pins
CSLEEP	Output	MCU in Sleep mode
CDSTOP	Output	CPU domain in Stop mode
SRDSTOP	Output	SmartRun domain in Stop mode

Table 85. PWR internal input/output signals

Internal signal name	Signal type	Description
WKUPx_y (x = 1 to 8, y = 1 to 4)	Input	Wakeup event source selection

Each of the eight wakeup events, WKUPx, can be generated from four pins or internal events, selected by WUSELx[1:0] in the PWR_WUCR3 register (x = 1 to 8).

Table 86. PWR wakeup source selection

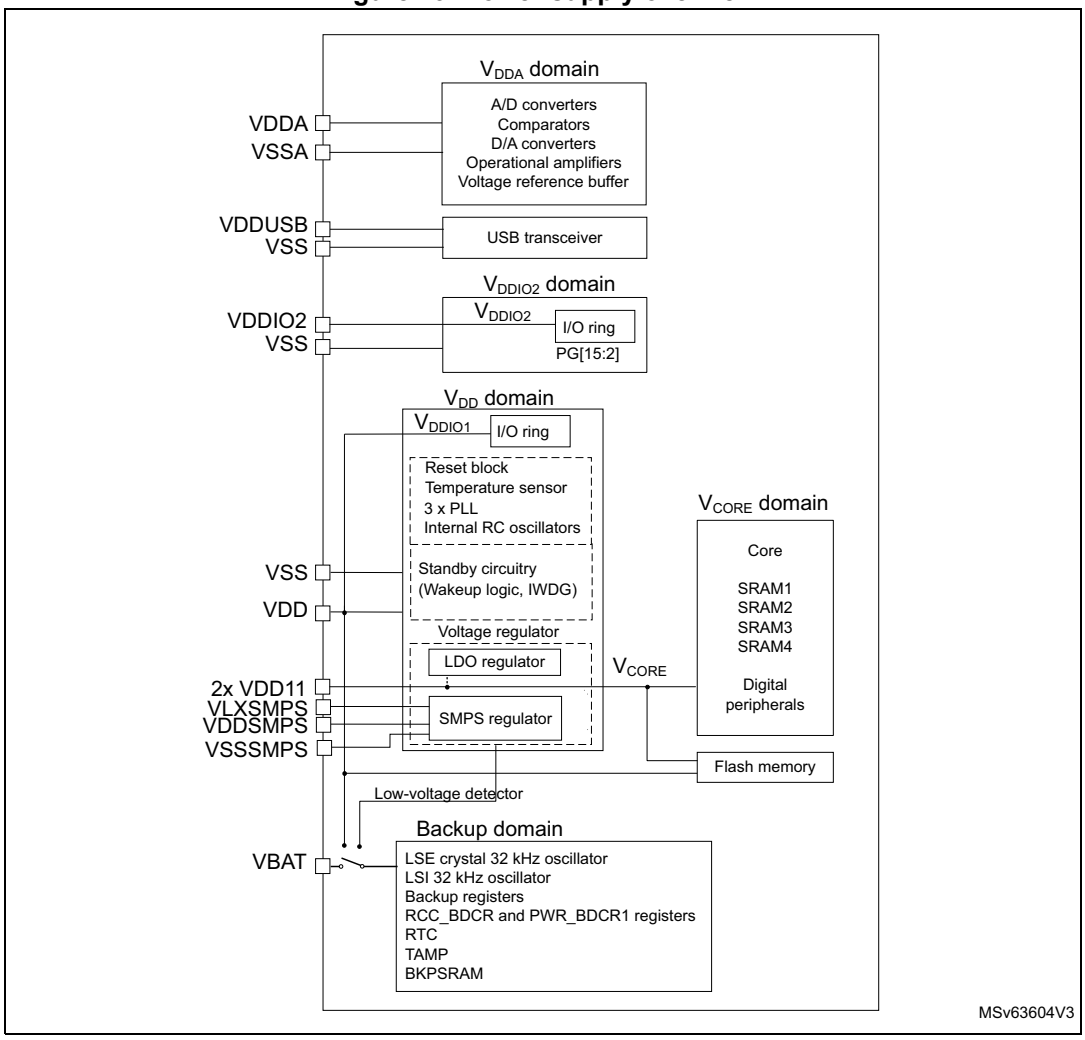
Wakeup event	Internal signal source (x = 1 to 8)			
	WKUPx_0 (WUSELx = 00)	WKUPx_1 (WUSELx = 01)	WKUPx_2 (WUSELx = 10)	WKUPx_3 (WUSELx = 11)
WKUP1	PA0	PB2	PE4	Reserved
WKUP2	PA4	PC13	PE5	Reserved
WKUP3	PE6	PA1	PB6	Reserved
WKUP4	PA2	PB1	PB7	Reserved

Table 86. PWR wakeup source selection (continued)

Wakeup event	Internal signal source (x = 1 to 8)			
	WKUPx_0 (WUSELx = 00)	WKUPx_1 (WUSELx = 01)	WKUPx_2 (WUSELx = 10)	WKUPx_3 (WUSELx = 11)
WKUP5	PC5	PA3	PB8	Reserved
WKUP6	PB5	PA5	PE7	RTC_ALRA_S or RTC_ALRB_S or RTC_WUT_S or RTC_TS_S
WKUP7	PB15	PA6	PE8	RTC_ALRA or RTC_ALRB or RTC_WUT or RTC_TS
WKUP8	PF2	PA7	PB10	TAMP or TAMP_S

10.4 PWR power supplies and supply domains

Figure 29. Power supply overview



10.4.1 External power supplies

The devices require a 1.71 V to 3.6 V V_{DD} operating voltage supply. Several independent supplies can be provided for specific peripherals. Those supplies must not be provided without a valid operating supply on the VDD pin:

- V_{DD} = 1.71 V to 3.6 V (functionality guaranteed down to V_{BOR} minimum value)

  V_{DD} is the external power supply for the I/Os, the internal regulator and the system analog such as reset, power management and internal clocks. It is provided externally through the VDD pins.
- V_{DDA} = 1.58 V (COMPs) / 1.6 V (DACs, OPAMPs) / 1.62 (ADCs) / 1.8 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 and must be connected to VDD or VSS (preferably to VDD) when these peripherals are not used.
- V_{DDSMPS} = 1.71 V to 3.6 V

  V_{DDSMPS} is the external power supply for the SMPS step-down converter. It is provided externally through VDDSMPS supply pin, and must be connected to the same supply as VDD pin.
- V_{LXSMPS} is the switched SMPS step-down converter output.

Note: The SMPS power supply pins are available only on a specific package with SMPS step-down converter 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 and must preferably be connected to V_{DD} when the USB is not 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 must preferably be connected to V_{DD} when PG[15:2] are not used.
- V_{BAT} = 1.65 V to 3.6 V (functionality guaranteed down to V_{BOR_VBAT} minimum value, refer to the product datasheet)

  V_{BAT} is the power supply when V_{DD} is not present (through power switch) for RTC, TAMP, external and internal clocks 32 kHz oscillator, backup registers and optionally backup SRAM.
- 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.

  V_{REF+} can be grounded when ADC and DAC are not active.

  The internal voltage reference buffer supports four output voltages, that are configured with VRS bit in the VREFBUF_CSR register:
  - V_{REF+} around 1.5 V. This requires $V_{DDA} \geq 1.8$ V.
  - V_{REF+} around 1.8 V. This requires $V_{DDA} \geq 2.1$ V.
  - V_{REF+} around 2.048 V. This requires $V_{DDA} \geq 2.4$ V.
  - V_{REF+} around 2.5 V. This requires $V_{DDA} \geq 2.8$ V.

VREF- and VREF+ pins are not available on all packages. When not available, they are bonded to VSSA and VDDA, respectively.

When the VREF+ is double-bonded with VDDA in a package, the internal voltage reference buffer is not available and must be kept disabled.

V_{REF-} must always be equal to V_{SSA}.

10.4.2 Internal regulators

The devices embed two regulators: one LDO and one SMPS in parallel to provide the V_{CORE} supply for digital peripherals, SRAMs (except BKPSRAM) and embedded Flash memory. The SMPS generates this voltage on VDD11 (two pins) with a total external capacitor of 4.7 µF typical and requires an external coil of 2.2 µH typical.

The LDO generates this voltage on VCAP (one or two pins depending on packages) with a total of external capacitor of 4.7 µF typical.

Both regulators can provide four different voltages (voltage scaling) and can operate in Stop mode.

It is possible to switch from SMPS to LDO and from LDO to SMPS on the fly.

10.4.3 Power-up and power-down power sequences

During power-up and power-down phases, the following power sequence requirements must be respected:

- When V_{DD} is below 1 V, other power supplies (V_{DDA}, V_{DDIO2}, V_{DDUSB}) must remain below V_{DD} + 300 mV.
- When V_{DD} is above 1 V, all power supplies are independent.

During the power-down phase, V_{DD} can temporarily become lower than other supplies only if the energy provided to the MCU remains below 1 mJ. This allows external decoupling capacitors to be discharged with different time constants during the power-down transient phase.

10.4.4 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 that can be separately filtered and shielded from noise on the PCB:

- The analog peripherals voltage supply input is available on a separate VDDA pin.
- An isolated supply ground connection is provided on VSSA pin.

The V_{DDA} supply voltage can be different from V_{DD}. The presence of V_{DDA} must be checked before enabling any of the analog peripherals supplied by V_{DDA} (A/D converter, D/A converter, comparators, operational amplifiers, voltage reference buffer).

After reset, the analog features supplied by V_{DDA} are logically and electrically isolated and therefore are not available. The isolation must be removed before using the analog peripherals, by setting the ASV bit in the PWR_SVMCR register, once the V_{DDA} supply is present.

The V_{DDA} supply can be monitored by the analog voltage monitors (AVM), and compared with two thresholds (1.6 V for AVM1 or 1.8 V for AVM2), refer to *Section 10.6.3: 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 VREF+, 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 can output a configurable voltage: 1.5 V, 1.8 V, 2.048 V or 2.4 V. The internal voltage reference can also provide the voltage to external components through VREF+ pin. Refer to the device datasheet and to *Section 32: Voltage reference buffer (VREFBUF)* for further information.

10.4.5 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 VDDIO2 pin. The V_{DDIO2} voltage level is completely independent from V_{DD} or V_{DDA}. The VDDIO2 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 IO2SV bit in the PWR_SVMCR register, once the V_{DDIO2} supply is present.

The V_{DDIO2} supply is monitored by the IO2 voltage monitoring (IO2VM) and compared with the internal reference voltage (3/4 V_{REFINT}, around 0.9 V), refer to *Section 10.6.3: Peripheral voltage monitoring (PVM)* for more details.

10.4.6 Independent USB transceivers supply

The USB transceivers are supplied from a separate VDDUSB 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 OTG peripheral, by setting the USV bit in the PWR_SVMCR register, once the V_{DDUSB} supply is present.

The V_{DDUSB} supply is monitored by the USB voltage monitoring (UVM) and compared with the internal reference voltage (V_{REFINT}, around 1.2 V), refer to *Section 10.6.3: Peripheral voltage monitoring (PVM)* for more details.

10.4.7 Battery Backup domain

To retain the content of the backup registers, backup SRAM, and supply the RTC and TAMP functions when V_{DD} is turned off, the VBAT pin can be connected to an optional backup voltage supplied by a battery or by another source.

The Backup domain supply is V_{SW}, which is the output of a power switch between V_{DD} and V_{BAT}. The switch between V_{DD} and V_{BAT} supplies is automatically controlled by the brownout reset circuitry.

When V_{DD} is below the lowest brownout reset threshold (V_{BOR0}), the VBAT pin powers the RTC and TAMP peripherals, the LSI and LSE oscillators. The backup SRAM is optionally powered by VBAT pin when BREN is set in PWR_BDCR1.

The following pins functions are also powered by the VBAT pin:

- PC13, PC14, and PC15 that can be configured by the RTC, the TAMP, or the LSE (see *Section 54.3: RTC functional description*)
- PE3, PE4, PE5, PE6, PC13, PA0, PA1, and PC5 when they are configured by the TAMP as tamper pins
- PB2 when configured by the RTC as RTC_OUT2 output

When V_{DD} is higher than V_{BOR0}, the VDD pin powers all previous functions.

Note: Due to the fact that the analog power switch can transfer only a limited amount of current (3 mA), the use of GPIO PC13 to PC15 in output mode is restricted: the speed must be limited to 2 MHz with a maximum load of 30 pF and these I/Os must not be used as a current source (for example to drive a LED).

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.

Backup domain access

After a system reset, the Backup domain (RCC Backup domain control register RCC_BDCR, RTC registers, TAMP registers, backup registers and backup SRAM) 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 *RCC AHB3 peripheral clock enable register (RCC_AHB3ENR)*.
2. Set the DBP bit in the *PWR disable Backup domain register (PWR_DBPR)* to enable access to the Backup domain.

V_{BAT} battery charging

When V_{DD} is present, It is possible to charge the external battery on VBAT through an internal resistance.

The V_{BAT} charging is done either through a 5 kΩ resistor or through a 1.5 kΩ resistor depending on the VBRS bit value in the PWR_BDCR2 register.

The battery charging is enabled by setting VBE bit in the PWR_BDCR2 register. It is automatically disabled in V_{BAT} mode.

10.5 PWR system supply voltage regulation

10.5.1 SMPS and LDO embedded regulators

The devices embed two internal regulators, that can be selected when the application runs, depending on the application requirements:

- a SMPS step-down converter
- a linear voltage regulator (LDO)

The SMPS allows the power consumption to be reduced but some applications can be perturbed by the noise generated by the SMPS, requiring the application to switch to LDO.

The LDO and the SMPS regulators have two modes: Main regulator mode (used when performance is needed), and Low-power regulator mode. LDO or SMPS can be used in all voltage scaling ranges, and in all Stop modes.

10.5.2 LDO and SMPS versus reset, voltage scaling, and low-power modes

After reset, the regulator is the LDO, in range 4. Switching to SMPS provides lower consumption in particular at high V_{DD} voltage. It is possible to switch from LDO to SMPS, or from SMPS to LDO in any range, by configuring the REGSEL bit.

When exiting the Stop or Standby mode, the regulator is the same than when entering low-power modes. The voltage range is the range 4.

10.5.3 LDO and SMPS step down converter fast startup

After BOR reset, the LDO and SMPS regulators starts in slow-startup mode. This slow-startup feature is selected to limit the inrush current after power-on reset. This increases the wakeup time when exiting Stop or Standby mode.

However, it is possible to configure a faster startup on the fly and it is applied for next startup either after a system reset or wakeup from low-power mode except Shutdown and V_{BAT} modes. The fast startup is selected by setting the FSTEN bit in the PWR_CR3 register.

10.5.4 Dynamic voltage scaling management

The dynamic voltage scaling is a power management technique that consists in increasing or decreasing the voltage used for the digital peripherals (V_{CORE}), according to the application performance and power consumption needs.

Dynamic voltage scaling to increase V_{CORE} is known as overvolting. It 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 regulator operates in the following ranges:

- Range 1: high performance

  It provides a typical output voltage at 1.2 V. It is used when the system clock frequency

is up to 160 MHz.

- Range 2: medium-high performance

  It provides a typical output voltage at 1.1 V. It is used when the system clock frequency is up to 110 MHz.

- Range 3: medium-low power range

  It provides a typical output voltage at 1.0 V. The system clock frequency can be up to 55 MHz.

- Range 4: low power range

  It provides a typical output voltage at 0.9 V. The system clock frequency can be up to 25 MHz.

Voltage scaling is selected through the VOS[1:0] bits in the PWR_VOSR register. The EPOD (embedded power distribution) booster must be enabled and ready before increasing the system clock frequency above 55 MHz in range 1 and range 2.

The sequence to switch the voltage scaling from range “L” (lower power) to range “P” (higher performance) with L > P is the following:

1. If target SYSCLK > 55 MHz:
   a) Configure the PLL1MBOOST[3:0] in the RCC_PLL1CFGR to generate a booster clock frequency between 4 and 16 MHz.
   b) Switch on the PLL1 oscillator clock source.
   c) Select the PLL1 clock source (PLL1SRC[1:0] in the RCC_PLL1CFGR).
2. Program the VOS[1:0] bits to range P in the PWR_VOSR register.
3. Wait until the VOSRDY flag is set in the PWR_VOSR register.
4. If target SYSCLK > 55 MHz:
   a) Set BOOSTEN in the PWR_VOSR register. This step can be done together with VOS programming.
   b) Wait until the BOOSTRDY flag is set in the PWR_VOSR register.
5. Adjust number of wait states according new frequency target in range “P” (LATENCY bits in the FLASH_ACR and WSC bits in the RAMCFG_MxCR).
6. Configure and enable the PLL if needed.
7. Configure and switch to new system frequency.

The sequence to switch the voltage scaling from range “P” (higher performance) to range “L” (lower power) with L > P is:

1. Reduce the system frequency to a value lower than range “L” maximum frequency.
2. Adjust number of wait states according new frequency target (LATENCY bits in the FLASH_ACR and WSC bits in the RAMCFG_MxCR).
3. If new SYSCLK ≤ 55 MHz, clear BOOSTEN in the PWR_VOSR register if it was set.
4. Program the VOS bits to range “L” in the PWR_VOSR register. This step can be done together with BOOSTEN clearing.

10.6 PWR power supply supervision

10.6.1 Brownout reset (BOR)

The device has an integrated Brownout reset (BOR) circuitry. The BOR is active in all power modes except Shutdown mode, and cannot be disabled.

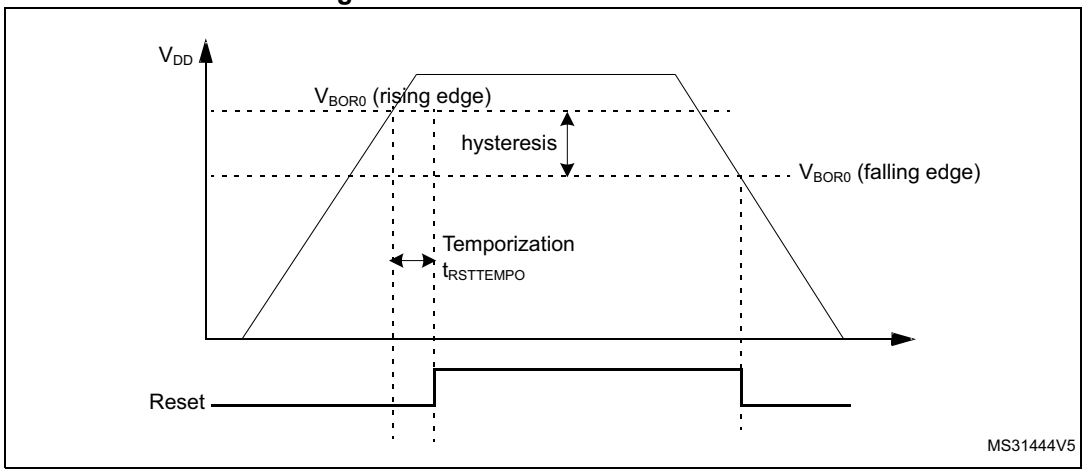
Five BOR thresholds can be selected through option bytes. BOR0 provides the always enabled power-on/power-down functionality, independent from any other higher BOR level selection.

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 Brownout reset thresholds, refer to the electrical characteristics section in the datasheet.

During Standby mode, 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 control register 1 (PWR_CR1)*.

Figure 30. Brownout reset waveform



1. The reset temporization $t_{RSTTEMPO}$ is present only for the BOR lowest threshold (V_{BOR0}).

10.6.2 Programmable voltage detector (PVD)

The PVD can be used to monitor the V_{DD} power supply by comparing it to a threshold selected by the PVDLS[2:0] bits in the *PWR supply voltage monitoring control register (PWR_SVMCR)*.

The PVD is enabled by setting the PVDE bit.

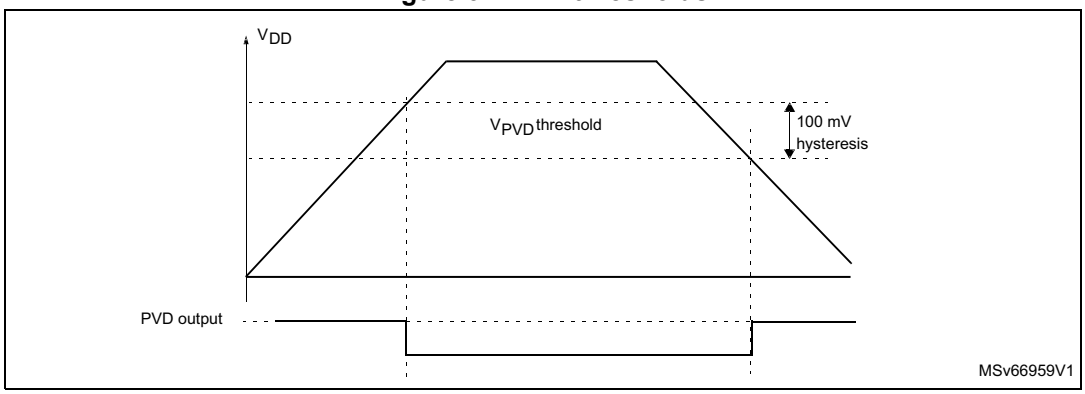
A PVDO flag is available in the *PWR supply voltage monitoring control register (PWR_SVMCR)* to indicate if V_{DD} is higher or lower than the PVD threshold. This event is internally connected to the EXTI and can generate an interrupt if enabled through the EXTI registers (refer to *Table 99: PWR interrupt requests*).

The rising/falling edge sensitivity of the EXTI Line must be configured according to PVD output behavior. For example, if the EXTI line is configured to rising edge sensitivity, the

interrupt is generated when V_{DD} drops below the PVD threshold. As an example, the service routine can perform emergency shutdown tasks.

The PVD can remain active in Stop 0, Stop 1, Stop 2 modes, and the PVM interrupt can wake up from the Stop mode. The PVD is not functional in Stop 3 mode.

Figure 31. PVD thresholds



10.6.3 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):

- The UVM monitors the USB supply V_{DDUSB}. VDDUSBRDY indicates if the V_{DDUSB} independent power supply is higher or lower than the V_{UVM} threshold.
- The IO2VM monitors the PG[15:2] supply V_{DDIO2}. VDDIO2RDY indicates if the V_{DDIO2} independent power supply is higher or lower than the V_{IO2VM} threshold.
- The AVM1 monitors the analog supply V_{DDA}. VDDA1RDY indicates if the V_{DDA} independent power supply is higher or lower than the V_{AVM1} threshold.
- The AVM2 monitors the analog supply V_{DDA}. VDDA2RDY indicates if the V_{DDA} independent power supply is higher or lower than the V_{AVM2} 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 PVM threshold and/or when it rises above the PVM threshold, depending on EXTI line rising/falling edge configuration. Refer to *Table 99: PWR interrupt requests*.

Each PVM can remain active in Stop 0, Stop 1, Stop 2 modes, and the PVM interrupt can wake up from the Stop mode. The PVM is not functional in Stop 3 mode.

Table 87. PVM features

PVM	Power supply	PVM threshold
UVM	V_{DDUSB}	V_{UVM} (around 1.2 V)
IO2VM	V_{DDIO2}	V_{IO2VM} (around 0.9 V)
AVM1	V_{DDA}	V_{AVM1} (around 1.6 V)
AVM2	V_{DDA}	V_{AVM2} (around 1.8 V)

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 assumes 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 OTG peripheral:

1. If V_{DDUSB} is independent from V_{DD}:
   a) Enable the UVM by setting UVMEN bit in the *PWR supply voltage monitoring control register (PWR_SVMCR)*.
   b) Wait for the UVM wakeup time.
   c) Wait until VDDUSBRDY bit is set in the *PWR supply voltage monitoring status register (PWR_SVMSR)*.
   d) Disable the UVM for consumption saving (optional).
2. Set the USV bit in the *PWR supply voltage monitoring control register (PWR_SVMCR)* 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 IO2VM by setting IO2VM bit in the *PWR supply voltage monitoring control register (PWR_SVMCR)*.
   b) Wait for the IO2CVM wakeup time.
   c) Wait until VDDIO2RDY bit is set in the *PWR supply voltage monitoring status register (PWR_SVMSR)*.
   d) Disable the IO2VM for consumption saving (optional).
2. Set the IO2SV bit in the *PWR supply voltage monitoring control register (PWR_SVMCR)* 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 AVM1 or AVM2 by setting AVM1EN or AVM2EN bit in the *PWR supply voltage monitoring control register (PWR_SVMCR)*.
   b) Wait for the AVM wakeup time.
   c) Wait until VDDA1RDY or VDDA2RDY bit is set in the *PWR supply voltage monitoring status register (PWR_SVMSR)*.
   d) Disable the AVM for consumption saving (optional).
2. Set the ASV bit in the *PWR supply voltage monitoring control register (PWR_SVMCR)* to remove the V_{DDA} power isolation.

10.6.4 Backup domain voltage and temperature monitoring

When the Backup domain voltage and temperature monitoring is enabled (MONEN = 1 in the PWR_DBPR register), the Backup domain voltage and the temperature are monitored.

If the Backup domain voltage monitoring internal tamper is enabled in the TAMP peripheral (ITAMP1E = 1 in the TAMP_CR1 register), a tamper event is generated when the Backup domain voltage is above the functional range. In case the Backup domain voltage is below the functional range, a Brownout reset is generated, erasing all device including Backup domain.

Note: The Backup domain voltage is V_{DD} when present, V_{BAT} otherwise.

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 functional range.

10.7 PWR power management

10.7.1 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 these 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 10.7.5: Sleep mode*.

- Stop 0, Stop 1, Stop 2, Stop 3 modes:

  Stop mode achieves the lowest power consumption while retaining the content of SRAM and registers. All clocks in the core domain are stopped. The PLL, the MSI (MSIS and MSIK) RC, the HSI16 RC and the HSE crystal oscillators are disabled. The LSE or LSI is still running.

  The RTC can remain active (Stop mode with RTC, Stop mode without RTC).

  Some peripherals are autonomous and can operate in Stop mode by requesting their kernel clock and their bus (APB or AHB) when needed, in order to transfer data with DMA (GPDMA1 or LPDMA1 depending on peripherals and power mode).

  In Stop 2 and Stop 3 modes, most of the core domain is put in a lower leakage mode. Stop 0 and 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 regulator remains in main regulator mode, allowing a very fast wakeup time but with much higher consumption.

  Stop 3 is the lowest power mode with full retention, but the functional peripherals and sources of wakeup are reduced to the same ones than in Standby mode.

  The system clock when exiting from Stop mode can be either MSIS up to 24 MHz or HSI16, depending on software configuration.

  Refer to *Section 10.7.6: Stop 0 mode*, *Section 10.7.7: Stop 1 mode*, *Section 10.7.8: Stop 2 mode* and *Section 10.7.9: Stop 3 mode*.

- Standby mode:

  The Standby mode is used to achieve the lowest power consumption with BOR. The

internal regulator is switched off so that the core domain is powered off. The PLL, the MSI (MSIS and MSIK) RC, the HSI16 RC and the HSE crystal oscillators are also switched off.

The RTC can remain active (Standby mode with RTC, Standby mode without RTC).

The Brownout reset (BOR) always remains active in Standby mode.

The state of each I/O during Standby mode can be selected by software: I/O with internal pull-up, internal pull-down or floating.

After entering Standby mode, SRAMs and register contents are lost except for registers and backup SRAM in the Backup domain and Standby circuitry. Optionally, the full SRAM2 or 8 Kbytes or 56 Kbytes can be retained in Standby mode, supplied by the low-power regulator (standby with RAM2 retention mode).

The BORL (Brownout reset detector low) can be configured in ultra-low-power mode to further reduce power consumption during standby mode.

The device exits Standby mode when an external reset (NRST pin), an IWDG reset, WKUP pin event (configurable rising or falling edge), a RTC event occurs (alarm, periodic wakeup, timestamp), or a tamper detection. The tamper detection can be raised either due to external pins or due to an internal failure detection.

The system clock after wakeup is MSIS up to 4 MHz.

Refer to *Section 10.7.10: Standby mode*.

- Shutdown mode:

  The Shutdown mode allows the lowest power consumption. The internal regulator is switched off so that the core domain is powered off. The PLL, the HSI16, the MSI (MSIS and MSIK), the LSI and the HSE oscillators are also switched off.

  The RTC can remain active (Shutdown mode with RTC, Shutdown mode without RTC). 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.

  SRAMs and register contents are lost except for registers in the Backup domain.

  The device exits Shutdown mode when an external reset (NRST pin), a WKUP pin event (configurable rising or falling edge), or a RTC event occurs (alarm, periodic wakeup, timestamp), or a tamper detection.

  The system clock after wakeup is MSIS at 4 MHz.

  Refer to *Section 10.7.11: Shutdown mode*.

The table below shows the power modes overview.

Table 88. Low-power mode summary

Mode name	Entry	Wakeup source(1)	Wakeup system clock	Effect on clocks	Voltage regulators
Sleep (Sleep-now or Sleep-on-exit)	WFI or Return from ISR	Any interrupt except OTG_FS and UCPD in range 4	Same as before entering Sleep mode	CPU clock OFF No effect on other clocks or analog clock sources	Range 1, 2, 3, 4
	WFE	Wakeup event			

Table 88. Low-power mode summary (continued)

Mode name	Entry	Wakeup source[1]	Wakeup system clock	Effect on clocks	Voltage regulators
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_CFGR1 MSIS with the frequency before entering the Stop mode, limited to 24 MHz, when STOPWUCK = 0	All clocks OFF except LSI and LSE MSIK or HSI16 can be enabled temporarily when requested by a peripheral during LPBAM.	Range 1, 2, 3, 4
Stop 1	LPMS = 001 + SLEEPDEEP bit + WFI or Return from ISR or WFE				Low-power regulator (SMPS or LDO)
Stop 2	LPMS = 010 + SLEEPDEEP bit + WFI or Return from ISR or WFE				
Stop 3	LPMS = 011 + SLEEPDEEP bit + WFI or Return from ISR or WFE	WKUP pin edge, RTC event, external reset in NRST pin, IWDG reset		All clocks OFF except LSI and LSE	
Standby with SRAM2_ 8 Kbytes	LPMS = 10x+ RRS1 = 1 + SLEEPDEEP bit + WFI or Return from ISR or WFE		MSIS from 1 MHz up to 4 MHz		
Standby with SRAM2_Full	LPMS = 10x+ RRS1 = RRS2 = 1+ SLEEPDEEP bit + WFI or Return from ISR or WFE				
Standby	LPMS = 10x + RRS1 = RRS2 = 0 + SLEEPDEEP bit + WFI or Return from ISR or WFE				OFF
Shutdown	LPMS = 11x + SLEEPDEEP bit + WFI or Return from ISR or WFE	WKUP pin edge, RTC event, external reset in NRST pin	MSIS 4 MHz	All clocks OFF except LSE	OFF

1. Refer to the next table.

Table 89. Functionalities depending on the working mode(1)

Peripheral	Run	Sleep	Stop 0/1		Stop 2		Stop 3		Standby		Shutdown		VBAT
			-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	
CPU	Y	-	-	-	-	-	-	-	-	-	-	-	-
Flash memory (2 Mbytes)	O(2)	O(2)	-	-	-	-	-	-	-	-	-	-	-
SRAM1 (192 Kbytes)	Y(3)	Y(4)	O(7)	-	O(7)	-	O(7)	-	-	-	-	-	-
SRAM2 (64 Kbytes)	Y(3)	Y(4)	O(7)	O(5)	O(7)	-	O(7)	-	O(6)	-	-	-	-
SRAM3 (512 Kbytes)	Y(3)	Y(4)	O(7)	O(5)	O(7)	-	O(7)	-	-	-	-	-	-
SRAM4 (16 Kbytes)	Y(3)	Y(4)	O(7)	-	O(7)	-	O(7)	-	-	-	-	-	-
BKPSRAM	O	O	O	O(5)	O		O		O				O
FSMC	O	O	-	-	-	-	-	-	-	-	-	-	-
OCTOSPIx (x =1,2)	O	O	-	-	-	-	-	-	-	-	-	-	-
Backup registers	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	-	-	-	-	-	-	-
Peripheral voltage monitor	O	O	O	O	O	O	-	-	-	-	-	-	-
GPDMA	O	O	O	O(8)	-	-	-	-	-	-	-	-	-
LPDMA	O	O	O	O(9)	O	O(9)	-	-	-	-	-	-	-
DMA2D	O	O											
High-speed internal (HSI16)	O	O	(10)	-	(10)	-	-	-	-	-	-	-	-
Oscillator HSI48	O	O	-	-	-	-	-	-	-	-	-	-	-
High-speed external (HSE)	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
Multi-speed internal (MSIS and MSIK)	O	O	(10)	-	(10)	-	-	-	-	-	-	-	-
Clock security system (CSS)	O	O	-	-	-	-	-	-	-	-	-	-	-

Table 89. Functionalities depending on the working mode[1] (continued)

Peripheral	Run	Sleep	Stop 0/1		Stop 2		Stop 3		Standby		Shutdown		VBAT
			-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	
Clock security system on LSE	O	O	O	O	O	O	O	O	O	O	O	O	O
Backup domain voltage monitoring, temperature monitoring	O	O	O	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 TAMP tamper pins	8	8	8	O	8	O	8	O	8	O	8	O	8
OTG_FS, UCPD	O[11]	O[11]	-	O	-	-	-	-	-	-	-	-	-
USARTx (x=1,2,3,4,5)	O	O	O[12]	O[12]	-	-	-	-	-	-	-	-	-
Low-power UART (LPUART)	O	O	O[12]	O[12]	O[12]	O[12]	-	-	-	-	-	-	-
I2Cx (x = 1,2,4)	O	O	O[13]	O[13]	-	-	-	-	-	-	-	-	-
I2C3	O	O	O[13]	O[13]	O[13]	O[13]	-	-	-	-	-	-	-
SPIx (x = 1,2)	O	O	O[14]	O[14]	-	-	-	-	-	-	-	-	-
SPI3	O	O	O[14]	O[14]	O[14]	O[14]							
FDCAN1	O	O	-	-	-	-	-	-	-	-	-	-	-
SDMMCx (x = 1,2)	O	O	-	-	-	-	-	-	-	-	-	-	-
SAIx (x = 1,2)	O	O	-	-	-	-	-	-	-	-	-	-	-
ADC1	O	O	-	-	-	-	-	-	-	-	-	-	-
ADC4	O	O	O[15]	O[15]	O[15]	O[15]	-	-	-	-	-	-	-
DAC1 (2 converters)	O	O	O	-	O	-	-	-	-	-	-	-	-
VREFBUF	O	O	O	-	O	-	-	-	-	-	-	-	-
OPAMPx (x = 1,2)	O	O	O	-	O	-	-	-	-	-	-	-	-
COMPx (x = 1,2)	O	O	O	O	O	O	-	-	-	-	-	-	-
Temperature sensor	O	O	O	-	O	-	-	-	-	-	-	-	-
Timers (TIMx)	O	O	-	-	-	-	-	-	-	-	-	-	-
LPTIMx (x = 1,3,4)	O	O	O[16]	O[16]	O[16]	O[16]	-	-	-	-	-	-	-
LPTIM2	O	O	O[16]	O[16]	-	-	-	-	-	-	-	-	-

Table 89. Functionalities depending on the working mode(1) (continued)

Peripheral	Run	Sleep	Stop 0/1		Stop 2		Stop 3		Standby		Shutdown		VBAT
			-	Wakeup capability	-	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	-	-	-	-	-	-	-	-	-	-	-
SysTick timer	O	O	-	-	-	-	-	-	-	-	-	-	-
Multi-function digital filter (MDF)	O	O	O(17)	O(17)	-	-	-	-	-	-	-	-	-
Audio digital filter (ADF)	O	O	O(17)	O(17)	O(17)	O(17)	-	-	-	-	-	-	-
Digital camera interface (DCMI)	O	O	-	-	-	-	-	-	-	-	-	-	-
Parallel synchronous slave interface (PSSI)	O	O	-	-	-	-	-	-	-	-	-	-	-
CORDIC co-processor (CORDIC)	O	O	-	-	-	-	-	-	-	-	-	-	-
Filter mathematical accelerator (FMAC)	O	O	-	-	-	-	-	-	-	-	-	-	-
Touch sensing controller (TSC)	O	O	-	-	-	-	-	-	-	-	-	-	-
Random number generator (RNG)	O	O	-	-	-	-	-	-	-	-	-	-	-
AES and secure AES	O	O	-	-	-	-	-	-	-	-	-	-	-
Public key accelerator (PKA)	O	O	-	-	-	-	-	-	-	-	-	-	-
On-the-fly decryption (OTFDEC)	O	O	-	-	-	-	-	-	-	-	-	-	-
HASH accelerator	O	O	-	-	-	-	-	-	-	-	-	-	-
CRC calculation unit	O	O	-	-	-	-	-	-	-	-	-	-	-
GPIOs	O	O	O	O	O	O	(18)	24 pins	(18)	24 pins	(19)	24 pins	-

1. 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 memory can be configured in power-down mode. By default, it is not in power-down mode.
3. The SRAMs can be powered on or off independently.
4. The SRAM clock can be gated on or off independently.

5. ECC error interrupt or NMI wakes up from Stop mode.
6. 8 Kbytes, 56 Kbytes or full SRAM2 content can be preserved.
7. Sub-blocks or full SRAM1 and SRAM3, full SRAM2 and SRAM4 can be powered-off to save power consumption. SRAM1, SRAM2, SRAM3 and SRAM4 can be accessed by GPDMA in Stop 0 and Stop 1 modes. SRAM4 can be accessed by LPDMA in Stop 0, Stop 1 and Stop 2 modes.
8. GPDMA transfers are functional and autonomous in Stop mode, and generates a wakeup interrupt on transfer events.
9. LPDMA transfers are functional and autonomous in Stop mode, and generates a wakeup interrupt on transfer events.
10. Some peripherals with autonomous mode and wakeup from Stop capability can request HSI16, MSIS or MSIK to be enabled. In this case, the oscillator is woken up by the peripheral, and is automatically put off when no peripheral needs it.
11. OTG_FS is functional in voltage scaling range 1, 2 and 3.
12. USART and LPUART reception and transmission is functional and autonomous in Stop mode, in asynchronous and in SPI master modes, and generates a wakeup interrupt on transfer events.
13. I2C reception and transmission is functional and autonomous in Stop mode, and generates a wakeup interrupt on transfer events.
14. SPI reception and transmission is functional and autonomous in Stop mode, and generates a wakeup interrupt on transfer events.
15. ADC conversion is functional and autonomous in Stop mode, and generates a wakeup interrupt on conversion events.
16. LPTIM is functional and autonomous in Stop mode, and generates a wakeup interrupt on events.
17. MDF and ADF is functional and autonomous in Stop mode, and generates a wakeup interrupt on events.
18. I/Os can be configured with internal pull-up, pull-down or floating in Standby mode.
19. 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.

In addition, the power consumption in Run mode can be reduced by one of the following means:

- Slowing down the system clocks and configuring voltage scaling to lower-power ranges.
- Gating the clocks to the APB and AHB peripherals when they are unused.
- Powering off unused RAMs

When a SRAM has been powered off, it can be powered on again by following the procedure:

1. Reset SRAMxPD in PWR_CR1.
2. Wait for 1.6 µs.
3. Set SRAMxEN in RCC_AHBxENR.

Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop 0, Stop 1, Stop 2, Stop 3, 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 control registers, the software can be debugged even when using the low-power modes extensively. For more details, refer to *Section 65.2.5: Debug and low-power modes*.

10.7.2 Low-power background autonomous mode (LPBAM)

The devices support a low-power background autonomous mode (LPBAM), that allows peripherals to be functional and autonomous in Stop 0, Stop 1 and Stop 2 modes (without any software running).

Stop 0 and Stop 1 modes

In Stop 0 and Stop 1 modes, the autonomous peripherals are ADC4, DAC1, LPTIMx (x = 1 to 4), USARTx (x = 1 to 5), LPUART1, SPIx (x = 1 to 3), I2Cx (x = 1 to 4), MDF1, ADF1, GPDMA1 and LPDMA1:

- ADC4, DAC1, LPTIM1, LPTIM3, LPUART1, SPI3, I2C3 and ADF1 are autonomous only with LPDMA1 and SRAM4.
- LPTIM2, USARTx (x = 1 to 5), SPI1, SPI2, I2C1, I2C2, I2C4 and MDF1 are autonomous only with GPDMA1 and SRAM1 to SRAM4.

Stop 2 mode

In Stop 2 mode, the autonomous peripherals are ADC4, DAC1, LPTIM1, LPTIM3, LPUART1, SPI3, I2C3, ADF1 and LPDMA1. In this mode, the SRAM4 can be accessed by the LPDMA1.

LPBAM features

Those autonomous peripherals support the following features:

- Functionality in Stop mode thanks to its own independent clock (named kernel clock) request capability: the peripheral kernel clock is automatically switched on when requested by a peripheral, and automatically switched off when no peripheral requests it.
- DMA transfers supported in Stop mode thanks to the system clock request capability: the system clock (MSIS or HSI16) automatically switched on when requested by a peripheral, and automatically switched off when no peripheral requests it. When the system clock is requested by an autonomous peripheral, the system clock is woken up and distributed to all peripherals enabled in the RCC. This allows the DMA to access the enabled SRAM, and any enabled peripheral register (for instance GPIO or LPGPIO registers).
- Automatic start of the peripheral thanks to the hardware synchronous or asynchronous triggers (such as I/Os edge detection and low-power timer event)
- Wakeup from Stop mode with peripheral interrupt

The GPDMA1 and LPDMA1 are fully functional and the linked-list is updated in Stop mode, allowing the different DMA transfers to be linked without any CPU wakeup. This can be used to chain different peripherals transfers, or to write peripherals registers in order to change their configuration while remaining in Stop mode.

The DMA transfers from memory to memory can be started by hardware synchronous or asynchronous triggers, and the DMA transfers between peripherals and memories can also be gated by those triggers.

Here below some use-cases that can be done while remaining in Stop mode:

- ADC or DAC conversion triggered by a low-power timer (or any other trigger)
  - Wakeup from Stop mode on analog watchdog if the ADC conversion result is out of the programmed thresholds
  - Wakeup from Stop mode on DMA buffer event
- Audio digital filter data transfer into SRAM
  - Wakeup from Stop on sound activity detection
- I2C slave reception or transmission, SPI reception, UART/LPUART reception
  - Wakeup at the end of peripheral transfer or on DMA buffer event
- I2C master transfer, SPI transmission, UART/LPUART transmission, triggered by a low-power timer (or any other trigger)
  - Example: Sensor periodic read
  - Wakeup at the end of peripheral transfer or on DMA buffer event
- Bridges between peripherals
  - Example: ADC converted data transferred by communication peripherals
- Data transfer from/to GPIO/LPGPIO to/from SRAM for:
  - Controlling external components
  - Implementing data transmission and reception protocols
- Data transfer from a SRAM to another one

10.7.3 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 11: Reset and clock control (RCC)*.

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. This bit must be set for the peripherals requesting clocks in Stop mode for LPBAM.

Disabling the peripherals autonomous clock in Stop 2 mode can be performed automatically by resetting the corresponding bit in the RCC_AHB3AMENR and RCC_APB3AMENR registers.

10.7.4 Low-power modes

Entering into a low-power mode

The MCU enters in low-power modes 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.

Caution: The peripherals with autonomous mode feature are able to generate an AHB or APB clock request, depending on their internal events. If a clock request is present when WFI or WFE is executed, the low-power mode entry is delayed until the clock request is released.

Exiting a low-power mode

The way the MCU exits the Sleep or Stop mode depends 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:
  - an 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) must be cleared. Only NVIC interrupts with high enough priority wake up 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) must be cleared. All NVIC interrupts wake up the MCU, even the disabled ones. Only enabled NVIC interrupts with high enough priority wake up and interrupt the MCU.
  - an 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.

The MCU exits Stop 3, Standby or Shutdown mode through an external reset (NRST pin), an IWDG reset, a rising edge on one of the enabled WKUPx pins or a RTC/TAMP event (see *Figure 622: RTC block diagram*).

After waking up from Standby or Shutdown mode, the program execution restarts in the same way as after a reset (boot pin sampling, option bytes loading, reset vector is fetched).

Caution: When the device is in Stop mode, a peripheral interrupt powers on an internal oscillator. The corresponding NVIC interrupt channel must be enabled to allow the interrupt to exit the device from Stop mode. It is not allowed to disable a peripheral interrupt by disabling only the NVIC channel while keeping the peripheral interrupt enable, as the device could remain in Stop mode with clock ON.
The peripherals with autonomous mode feature are able to generate an AHB or APB clock request when the device is in Stop mode, depending on their internal events. The software must ensure that either DMA transfer or interrupt is served, by configuring properly and in a consistent way the RCC, the autonomous peripherals, the DMA channels and NVIC. Note that when an autonomous peripheral requests the bus clock in Stop mode, the AHB and APB clocks are distributed to all enabled peripherals (limited to SmartRun domain peripherals in Stop 2 mode). Consequently, enabled peripherals, even without autonomous mode capability, are temporarily clocked and can also generate an interrupt during this time. These peripherals interrupts wake up the device from Stop mode.

10.7.5 Sleep mode

I/O states in Sleep mode

In Sleep mode, all I/O pins keep the same state as in Run mode.

Entering Sleep mode

The MCU enters the Sleep mode as described in *Entering into a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 system control register is clear (see the table below for details on how to enter the Sleep mode).

Exiting Sleep mode

The MCU exits the Sleep mode as described in *Exiting a low-power mode* (see the table below for details on how to exit the Sleep mode).

Table 90. Sleep mode

Sleep mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP = 0 – No interrupt (for WFI) or event (for WFE) pending Refer to the Cortex-M33 system control register.
	On return from ISR while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 – No interrupt pending Refer to the Cortex-M33 system control register.

Table 90. Sleep mode (continued)

Sleep mode	Description
Mode exit	If WFI or Return from ISR was used for entry Interrupt (see *Table 153: STM32U575/585 vector table*) If WFE was used for entry and SEVONPEND = 0: Wakeup event (see *Section 17.3: EXTI functional description*) If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC (see *Table 153: STM32U575/585 vector table*) or Wakeup event (see *Section 17.3: EXTI functional description*)
Wakeup latency	None

10.7.6 Stop 0 mode

The Stop 0 mode is based on the Cortex-M33 Deepsleep mod combined with the peripheral clock gating. The voltage regulator is configured in main regulator mode. In Stop 0 mode, all clocks in the core domain are stopped. The PLL, MSIS, MSIK, HSI16 and HSE oscillators are disabled.

Some peripherals with the LPBAM capability can switch on HSI16 or MSIS or MSIK for transferring data (see *Section 10.7.2* for details).

All SRAMs and register contents are preserved, but the SRAMs can be totally or partially switched off to further reduced consumption.

The BOR is always available in Stop 0 mode.

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 Stop 0 mode

The MCU enters the Stop 0 mode as described in *Entering into a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 system control register is set (see *Table 91* for details on how to enter the Stop 0 mode).

If the 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 the individual control bits:

- The Independent watchdog (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 52.4: IWDG functional description*).
- The real-time clock (RTC) is configured by the RTCEN bit in the *RCC Backup domain control register (RCC_BDCR)*.
- The internal RC oscillator LSI clock or LSI clock divided by 128, is configured by the LSION and LSIPREDIV bits in RCC_BCDR.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in RCC_BCDR.

Several peripherals can be autonomous in Stop 0 mode and can add consumption if they are enabled (see *Section 10.7.2* for more details).

The OPAMPs, the COMPs, the PVM and the PVD can be used in Stop 0 mode. If they are not needed, they must be disabled by software to save their power consumptions.

The ADCx (x = 1, 4), the DAC1 (two channels), the temperature sensor and the VREFBUF can consume power during the Stop 0 mode, unless they are disabled before entering this mode.

Exiting Stop 0 mode

The MCU exits the Stop 0 mode as described in *Exiting a low-power mode* (see *Table 91* for details on how to exit Stop 0 mode).

When exiting Stop 0 mode by issuing an interrupt or a wakeup event, HSI16 is selected as system clock if the bit STOPWUCK is set in *RCC clock configuration register 1 (RCC_CFGR1)*. The MSIS oscillator is selected as system clock if STOPWUCK is cleared. The MSIS selection allows a wakeup at higher frequency (up to 24 MHz).

Several peripherals are autonomous in Stop mode, and can generate interrupts with wakeup from Stop capability. All peripheral clocks must be enabled to allow a wakeup from Stop interrupt (see *Peripheral clock gating*).

When exiting the Stop 0 mode, the MCU is in Run mode, range 4.

Table 91. 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) 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 pending – LPMS = 000 in PWR_CR1
	Note: To enter Stop 0 mode, all EXTI line pending bits (in the 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 the program execution continues.

Table 91. Stop 0 mode (continued)

Stop 0 mode	Description
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 (see *Table 153: STM32U575/585 vector table*). - any peripheral interrupt occurring when the AHB/APB clocks are present due to an autonomous peripheral clock request (the peripheral vector must be enabled in the NVIC) If WFE was used for entry and SEVONPEND = 0: - any EXTI line configured in event mode (see *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 (see *Table 153: STM32U575/585 vector table*). - any EXTI line configured in event mode (see *Section 17.3: EXTI functional description*) - any peripheral interrupt occurring when the AHB/APB clocks are present due to an autonomous peripheral clock request
	Note: All peripheral clocks must be enabled to allow this peripheral to generate a wakeup from Stop interrupt ([PERIPH]EN, [PERIPH]SMEN and [PERIPH]AMEN bits must be set in the RCC, and a functional independent clock must be selected).
Wakeup latency	Longest wakeup time between: MSIS or HSI16 wakeup time and Flash wakeup time from Stop 0 mode.

10.7.7 Stop 1 mode

The Stop 1 mode is the same as Stop 0 mode except that the regulator is in low-power mode (see the table below for details on how to enter and exit Stop 1 mode).

Table 92. 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) 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 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 the program execution continues.

Table 92. Stop 1 mode (continued)

Stop 1 mode	Description
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 (see *Table 153: STM32U575/585 vector table*). - any peripheral interrupt occurring when the AHB/APB clocks are present due to an autonomous peripheral clock request (the peripheral vector must be enabled in the NVIC) If WFE was used for entry and SEVONPEND = 0: - any EXTI line configured in event mode (see *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 (see *Table 153: STM32U575/585 vector table*). - any EXTI line configured in event mode (see *Section 17.3: EXTI functional description*) - any peripheral interrupt occurring when the AHB/APB clocks are present due to an autonomous peripheral clock request
	Note: All peripheral clocks must be enabled to allow this peripheral to generate a wakeup from Stop interrupt ([PERIPH]EN, [PERIPH]SMEN and [PERIPH]AMEN bits must be set in the RCC, and a functional independent clock must be selected).
Wakeup latency	Longest wakeup time between: MSIS or HSI16 wakeup time and regulator wakeup time from low-power mode + Flash wakeup time from Stop 1 mode.

10.7.8 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 core domain are stopped. The PLL, MSIS, MSIK, HSI16 and HSE oscillators are disabled.

Some peripherals with the LPBAM capability can switch on HSI16 or MSIS or MSIK for transferring data (see *Section 10.7.2* for more details).

All SRAMs and register contents are preserved, but the SRAMs can be totally or partially switched off to further reduced consumption.

The BOR is always available in 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 MCU enters the Stop 2 mode as described in *Entering into a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 system control register is set (see *Table 93* for details on how to enter the Stop 2 mode).

If the 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:

- The independent watchdog (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 52.4: IWDG functional description*).
- The real-time clock (RTC) is configured by the RTCEN bit in the *RCC Backup domain control register (RCC_BDCR)*.
- The internal RC oscillator LSI clock or LSI clock divided by 128, is configured by the LSION and LSIPREDIV bits in RCC_BDCR.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in RCC_BDCR.

Several peripherals can be autonomous in Stop 2 mode and can add consumption if they are enabled (see *Section 10.7.2* for more details).

The OPAMPs, the COMPs, the PVM and the PVD can be used in Stop 2 mode. If they are not needed, they must be disabled by software to save their power consumptions.

The ADCx (x = 1, 4), the DAC1 (two channels), the temperature sensor and the VREFBUF can consume power during the Stop 2 mode, unless they are disabled before entering this mode.

Caution: All the peripherals that 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 configuring the RCC registers.

Exiting Stop 2 mode

The MCU exits the Stop 2 mode as defined in *Exiting a low-power mode* (see *Table 93* for details on how to exit Stop 2 mode).

When exiting Stop 2 mode by issuing an interrupt or a wakeup event, HSI16 is selected as system clock if the bit STOPWUCK is set in *RCC clock configuration register 1 (RCC_CFGR1)*. MSIS is selected as system clock if STOPWUCK is cleared. The MSI selection allows a wakeup at higher frequency (up to 24 MHz).

Several peripherals are autonomous in Stop mode, and can generate interrupts with wakeup from Stop capability. All peripheral clocks must be enabled to allow a wakeup from Stop interrupt (see *Peripheral clock gating*).

When exiting the Stop 2 mode, the MCU is in Run mode, range 4.

Table 93. 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) 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 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 the 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 (see *Table 153: STM32U575/585 vector table*). - any peripheral interrupt occurring when the AHB/APB clocks are present due to an autonomous peripheral clock request (the peripheral vector must be enabled in the NVIC) If WFE was used for entry and SEVONPEND = 0: - any EXTI line configured in event mode (see *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 (see *Table 153: STM32U575/585 vector table*). - any EXTI line configured in event mode (see *Section 17.3: EXTI functional description*) - any peripheral interrupt occurring when the AHB/APB clocks are present due to an autonomous peripheral clock request
	Note: All peripheral clocks must be enabled to allow this peripheral to generate a wakeup from Stop interrupt ([PERIPH]EN, [PERIPH]SMEN and [PERIPH]AMEN bits must be set in the RCC, and a functional independent clock must be selected).
Wakeup latency	Longest wakeup time between: MSIS or HSI16 wakeup time and regulator wakeup time from low-power mode + Flash wakeup time from Stop 2 mode.

10.7.9 Stop 3 mode

The Stop 3 mode is based on the Cortex-M33 Deepsleep mode combined with peripheral clock gating. In Stop 3 mode, all clocks in the core domain are stopped. The PLL, MSIS, MSIK, HSI16 and HSE oscillators are disabled.

All SRAMs and register contents are preserved, but the SRAMs can be totally or partially switched off to further reduce consumption.

The BOR is always available in Stop 3 mode.

All other peripherals must be either disabled by clearing the enable bit in the peripheral itself, or put under reset state by configuring RCC registers.

I/O states in Stop 3 mode

In the Stop 3 mode, the I/Os are by default in floating state. If the APC bit in the PWR_APCR register is set, the I/Os can be configured either with a pull-up (see PWR_PUCRx registers), or with a pull-down (see PWR_PDCRx registers), or can be kept in analog state if none of the PWR_PUCRx or PWR_PDCRx register is 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 I/O.

Some I/Os (listed in *Section 13.3.1: 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 Stop 3 mode setting their respective bit to 1 in the PWR_PUCRx or PWR_PDCRx registers, or to be configured to floating state if the bit is kept at 0.

The RTC outputs on PC13 and PB2 are functional in Stop 3 mode. PC14 and PC15 used for LSE are also functional. The 24 wakeup pins multiplexed on eight events (WKUPx, x = 1 to 8) and the eight RTC tampers pins are available.

Entering Stop 3 mode

The MCU enters the Stop 3 mode as described in *Entering into a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 System Control register is set (see *Table 94* for details on how to enter the Stop 3 mode).

If the Flash memory programming is ongoing, the Stop 3 mode entry is delayed until the memory access is finished.

If an access to the APB domain is ongoing, the Stop 3 mode entry is delayed until the APB access is finished.

In Stop 3 mode, the following features can be selected by programming individual control bits:

- The independent watchdog (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 30.3: IWDG functional description*).
- The real-time clock (RTC) is configured by the RTCEN bit in the *RCC Backup domain control register (RCC_BDCR)*.
- The internal RC oscillator LSI clock or LSI clock divided by 128, is configured by the LSION and LSIPREDIV bits in RCC_BDCR.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in RCC_BDCR.

Exiting Stop 3 mode

The MCU exits the Stop 3 mode as described in *Exiting a low-power mode* (see *Table 94* for details on how to exit Stop 3 mode).

When exiting Stop 3 mode by issuing an interrupt or a wakeup event, HSI16 is selected as system clock if the STOPWUCK bit is set in *RCC clock configuration register 1 (RCC_CFGR1)*. MSIS is selected as system clock if STOPWUCK is cleared. The MSIS selection allows a wakeup at higher frequency (up to 24 MHz).

When exiting the Stop 3 mode, the MCU is in Run mode, range 4.

Table 94. Stop 3 mode

Stop 3 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) pending – LPMS =“011 in PWR_CR1 – WUFx bits cleared in PWR_WUSR
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending – LPMS = 011 in PWR_CR1 – WUFx bits cleared in PWR_WUSR – RTC/TAMP flags corresponding to the chosen wakeup source, cleared
	Note: To enter Stop 3 mode, all WUFx, and the RTC/TAMP flags generating wakeup interrupts must be cleared. Otherwise, the Stop 3 mode entry procedure is completed but the Stop 3 is exited immediately after entry.
Mode exit	WKUPx pin edge, RTC/TAMP event/interrupt, NRST pin external reset, IWDG reset, BOR reset
Wakeup latency	Longest wakeup time between: MSIS or HSI16 wakeup time and regulator wakeup time from low-power mode + Flash wakeup time from Stop 3 mode.

10.7.10 Standby mode

The lowest power mode in which the BOR is active is the Standby mode. It is based on the Cortex-M33 Deepsleep mode, with the voltage regulators disabled (except when SRAM2 content is preserved). The PLL, HSI16, MSIS, MSIK and HSE oscillators are also switched off.

The SRAMs and register contents are lost except for registers in the Backup domain and Standby circuitry (see *Figure 29: Power supply overview*). SRAM2 content can be partially or fully preserved depending on RRSB1 and RRSB2 bits configuration in PWR_CR1. 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 ULPMEN bit in the PWR_CR1 register must be configured to 1 to reach the lowest power consumption by forcing the BOR in ultra-low-power mode.

I/O states in Standby mode

In the Standby mode, the I/Os are by default in floating state. If the APC bit in the PWR_APCR register is set, the I/Os can be configured either with a pull-up (see PWR_PUCRx registers), or with a pull-down (see PWR_PDCRx registers), or can be kept in analog state if none of the PWR_PUCRx or PWR_PDCRx register is 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 I/O.

Some I/Os (listed in *Section 13.3.1: 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 to 1 in the PWR_PUCRx or PWR_PDCRx registers, or to be configured to floating state if the bit is kept at 0.

The RTC outputs on PC13 and PB2 are functional in Standby mode. PC14 and PC15 used for LSE are also functional. The 24 wakeup pins multiplexed on eight events (WKUPx, x = 1 to 8) and the eight RTC tampers pins are available.

Entering Standby mode

The MCU enters the Standby mode as described in *Entering into a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 system control register is set (see *Table 95* for details on how to enter Standby mode).

In Standby mode, the following features can be selected by programming individual control bits:

- The independent watchdog (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 52.4: IWDG functional description*).
- The real-time clock (RTC) is configured by the RTCEN bit in *RCC Backup domain control register (RCC_BDCR)*.
- The internal RC oscillator LSI clock or LSI clock divided by 128, is configured by the LSION and LSIPRE bits in RCC_BDCR.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in RCC_BDCR.

Exiting Standby mode

The MCU exits the Standby mode as described in *Exiting a low-power mode*. The SBF status flag in the *PWR status register (PWR_SR)* indicates that the MCU was in Standby mode. All registers are reset after wakeup from Standby except for *PWR control register 3 (PWR_CR3)* (see *Table 95* for more details on how to exit Standby mode).

When exiting Standby mode, I/Os that were configured with pull-up or pull-down during Standby through PWR_PUCRx or PWR_PDCRx, keep this configuration upon exiting Standby mode until the APC bit in PWR_CR3 is cleared. Once APC is cleared, the I/Os are either configured to their reset values or to the pull-up/pull-down state according to the GPIOx_PUPDR registers. The content of the PWR_PUCRx or PWR_PDCRx registers is not lost and can be re-used for a sub-sequent entering into Standby mode.

Some I/Os (listed in *Section 13.3.1: 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 I/Os, with a pull-up or pull-down pre-defined after reset (some JTAG/SW I/Os) or with the 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 APC is cleared, releasing the PWR_PUCRx or PWR_PDCRx programmed value.

Table 95. 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) pending – LPMS = 10x in PWR_CR1 (1) – WUFx bits cleared in PWR_WUSR
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending – LPMS = 10x in PWR_CR1(1) – WUFx bits cleared in PWR_WUSR – RTC/TAMP flags corresponding to the chosen wakeup source, cleared
Mode exit	WKUPx pin edge, RTC/TAMP event/interrupt, NRST pin external reset, IWDG reset, BOR reset
Wakeup latency	Reset phase

1. The Standby mode is also entered if LPMS = 11X in PWR_CR1 with BREN = 1 in PWR_BDCR1.

10.7.11 Shutdown mode

The lowest power consumption is reached in Shutdown mode. It is based on the Deepsleep mode with the voltage regulator disabled. The core domain is consequently powered off. The PLL, HSI16, MSIS, MSIK and HSE oscillators are also switched off.

The SRAMs 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, I/Os are by default in floating state. If the APC bit in the PWR_APCR register is set, the I/Os can be configured either with a pull-up (see PWR_PUCRx registers (x=A,B,C,D,E,F,G,H)), or with a pull-down (see 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 is 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 I/O. However this configuration is lost when exiting the Shutdown mode due to the power-on reset.

Some I/Os (listed in *Section 13.3.1: 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 Shutdown mode setting to 1 their respective bit in the PWR_PUCRx or PWR_PDCRx registers, or to be configured to floating state if the bit is kept at 0.

The RTC outputs on PC13 and PB2 are functional in Shutdown mode. PC14 and PC15 used for LSE are also functional. The 24 wakeup pins multiplexed on eight events (WKUPx, x = 1 to 8) and the eight RTC tampers pins are available.

Entering Shutdown mode

The MCU enters the Shutdown mode as described in *Entering into a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 system control register is set (see *Table 96* for details on how to enter Shutdown mode).

In Shutdown mode, the following features can be selected by programming individual control bits:

- The real-time clock (RTC) is configured by the RTCEN bit in the Backup domain control register (RCC_BDCR). Caution: in case of V_{DD} power-down, the RTC content is lost.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in the Backup domain control register (RCC_BDCR).

Caution: The Shutdown mode cannot be entered if the BREN bit is set in the *PWR Backup domain control register 1 (PWR_BDCR1)*. If BREN = 1, the Standby mode is entered instead of Shutdown mode.

Exiting Shutdown mode

The MCU exits the Shutdown mode as described in *Exiting 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 a wakeup from Shutdown (see *Table 96* 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 13.3.1: General-purpose I/O (GPIO)*).

Table 96. 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) pending – LPMS = 11X in PWR_CR1 with BREN = 0 in PWR_BDCR1 – WUFx bits cleared in PWR_WUSR
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXT = 1 – No interrupt pending – LPMS = 11X in PWR_CR1 – WUFx bits cleared in PWR_WUSR – RTC/TAMP flags corresponding to the chosen wakeup source, cleared
Mode exit	WKUPx pin edge, RTC/TAMP event/interrupt, NRST pin external reset
Wakeup latency	Reset phase

10.7.12 Power modes output pins

In order to help the debug, three signals are available as device pins alternate functions:

- **CSLEEP**

  When set, CSLEEP indicates that the CPU is in Sleep mode: WFI or WFE has been executed.

  When cleared, CSLEEP indicates that the CPU is in Run mode.

- **CDSTOP**

  When set, CDSTOP indicates that the CPU domain (CD) is in CStop mode, meaning that the following conditions are filled:

  – WFI or WFE has been executed with CPU SLEEPDEEP = 1.

  – No AHB/APB clock is running in the CPU domain.

  When cleared, CDSTOP indicates that the CPU domain is not in CStop mode: AHB/APB clocks run in the CPU domain.

- **SRDSTOP**

  When set, SRDSTOP indicates that the SmartRun domain (SRD) is in DStop mode, meaning that the following conditions are filled:

  – WFI or WFE has been executed with CPU SLEEPDEEP = 1.

  – No AHB/APB clock is running in the SRD domain.

  When cleared, SRDSTOP indicates that the SmartRun domain is not in DStop mode: AHB/APB clocks run in the SRD domain.

Note: The AHB/APB clocks run after WFI or WFE has been executed if an autonomous peripheral requests its bus clock in Stop mode. The peripherals bus clock request can delay or prevent the device to enter low-power modes (refer to Section 10.7.2: Low-power background autonomous mode (LPBAM) and Section 10.7.4: Low-power modes).

The table below explains the MCU power mode depending on these signals states.

Table 97. Power modes output states versus MCU power modes

CSLEEP	CDSTOP	SRDSTOP	MCU power modes(1)
0	0	0	Run mode
1	0	0	Sleep mode or Stop 0 or Stop 1 mode, with AHB/APB clocks running in CPU domain (CD) and SmartRun domain (SRD)
1	1	0	Stop 0, Stop 1 or Stop 2 mode, with AHB/APB clocks running in SmartRun domain (SRD)
1	1	1	Stop 0, Stop 1, or Stop 2 mode

1. CSLEEP, CDSTOP and SRDSTOP are generated in core domain, consequently they are not driven in Stop 3, Standby and Shutdown modes.

10.8 PWR security and privileged protection

10.8.1 PWR security protection

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 the following features to be secured through the PWR_SECCFGR register:

- Low-power mode
- Wake-up (WKUP) pins
- Voltage detection and monitoring
- V_{BAT} mode
- I/Os pull-up/pull-down configuration

Other PWR configuration bits are secure when:

- The system clock selection is secure in RCC: the voltage scaling (VOS) configuration and the regulator booster (BOOSTEN) are secure.
- A GPIO is configured as secure: its corresponding bit for pull-up/pull-down configuration in Standby mode is secure.
- The UCPD1 is secure in the GTZC: the PWR_UCPDR register is secure.

Table 98 gives a summary of the PWR secured bits following the security configuration bit in PWR_SECCFGR. As soon as at least one function is configured to be secure, the PWR clock control is also secure in the RCC.

A non-secure access to a secure-protected register bit is denied:

- The secured bits are not written (WI) with a non-secure write access.
- The secured bits are read as 0 (RAZ) with a non-secure read access.

A non-secure write access to PWR_SECCFGR is WI and generates an illegal access event and an interrupt if enabled in the GTZC. It can be read with a non-secure read access.

When the TrustZone security is disabled (TZEN = 0), PWR_SECCFGR is RAZ/WI and all other registers are non-secure.

Table 98. PWR Security configuration summary

Secure configuration register	Security configuration bit	Register name	Secured bits	Non-secure access on secure bits
PWR_SECCFGR	Not applicable(1)	PWR_SECCFGR	All bits	Read OK. WI and illegal access event
PWR_SECCFGR	At least one bit is set	PWR_PRIVCFGR	SPRIV	Read OK. WI
PWR_SECCFGR	LPMSEC	PWR_CR1	All bits	RAZ/WI
		PWR_CR2	All bits	
		PWR_SR	CSSF	WI

Table 98. PWR Security configuration summary (continued)

Secure configuration register	Security configuration bit	Register name	Secured bits	Non-secure access on secure bits
PWR_SECCFGR	VDMSEC	PWR_CR3	All bits	RAZ/WI
		PWR_SVMCR	All bits	RAZ/WI
PWR_SECCFGR	VBSEC	PWR_BDCR1	All bits	RAZ/WI
		PWR_BDCR2	All bits	RAZ/WI
		PWR_DBPR	All bits	RAZ/WI
PWR_SECCFGR	APCSEC	PWR_APCR	All bits	RAZ/WI
PWR_SECCFGR	WUPxSEC (x = 1 to 8)	PWR_WUCR1	WUPENx	RAZ/WI
		PWR_WUCR2	WUPPx	RAZ/WI
		PWR_WUCR3	WUSELx	RAZ/WI
		PWR_WUSCR	CWUFx	WI
GTZC_TZSC_ SECCFGR	UCPD1SEC	PWR_UCPDR	All bits	RAZ/WI
RCC_SECCFGR	SYSCLKSEC	PWR_VOSR	VOS[1:0], BOOSTEN	RAZ/WI
GPIOx_SECCFGR (x=A,B..I)	SECy (y=0..15)	PWR_PUCRx (x = A to I)	PUy (y = 0 to 15)	RAZ/WI
		PWR_PDCRx (x = A to I)	PDy (y = 0 to 15)	RAZ/WI

1. PWR_SECCFGR is always secure.

10.8.2 PWR privileged protection

By default, after a reset, all PWR registers can be read or written with both privileged and unprivileged accesses, except PWR_PRIVCFGR that can be written with privileged access only. PWR_PRIVCFGR can be read by secure and non secure, privileged and unprivileged accesses.

The SPRIV bit in PWR_PRIVCFGR can be written with secure privileged access only. This bit configures the privileged access of all PWR secure functions (defined by PWR_SECCFGR, GTZC, RCC or GPIO as shown in *Table 98: PWR Security configuration summary*).

When the SPRIV bit is set in PWR_PRIVCFGR:

- The PWR secure bits can be written only with privileged access, including PWR_SECCFGR.
- The PWR secure bits can be read only with privileged access except PWR_SECCFGR and PWR_PRIVCFGR that can be read by privileged or unprivileged access.
- An unprivileged access to a privileged PWR bit or register is discarded: the bits are read as zero and the write to these bits is ignored (RAZ/WI).

The NSPRIV bit of PWR_PRIVCFGR can be written with privileged access only, secure or non-secure. This bit configures the privileged access of all PWR securable functions that are configured as non-secure (defined by PWR_SECCFGR, GTZC, RCC or GPIO as shown in *Table 98: PWR Security configuration summary*)).

When the NSPRIV bit is set in PWR_PRIVCFGR:

- The PWR securable bits that are configured as non-secure, can be written only with privileged access.
- The PWR securable bits that are configured as non-secure, can be read only with privileged access except PWR_PRIVCFGR that can be read by privileged or unprivileged accesses.
- The VOSRDY and BOOSTRDY bits in PWR_VOSR, PWR_SR, PWR_SVMSR, PWR_BDSR and PWR_WUSR, can be read with privileged or unprivileged accesses.
- An unprivileged access to a privileged PWR bit or register is discarded: the bits are read as zero and the write to these bits is ignored (RAZ/WI).

10.9 PWR interrupts

The table below gives a summary of the interrupt sources and the way to control them.

Table 99. PWR interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep, Stop 0, 1, 2 modes	Exit Stop 3, Standby, Shutdown modes
PWR_S3WU(1)	Wakeup interrupt flag	WUFx (x = 1 to 8)	WUPENx (x = 1 to 8)	Write CWUFx = 1 (x = 1 to 8)	No	Yes(2)
PVD_PVM	Programmable voltage detector through EXTI line 16	PVDO	EXTI line 16 enabled	Write EXTI PIF16 = 1	Yes	No
	USB supply voltage monitor through EXTI line 19	VDDUSBRDY	EXTI line 19 enabled	Write EXTI PIF19 = 1	Yes	No
	V_{DDIO2} supply voltage monitor through EXTI line 20	VDDIO2RDY	EXTI line 20 enabled	Write EXTI PIF20 = 1		
	Analog supply voltage monitor1 through EXTI line 21	VDDA1RDY	EXTI line 21 enabled	Write EXTI PIF21 = 1		
	Analog supply voltage monitor2 through EXTI line 22	VDDA2RDY	EXTI line 22 enabled	Write EXTI PIF22 = 1		

1. The PWR_S3WU interrupt is generated only when the device is in Stop 3 mode (not applicable in Run, Sleep, Stop 0, Stop 1 and Stop 2 modes).
2. Only an interrupt can wake up from Stop 3 mode (not possible with an event).

10.10 PWR registers

10.10.1 PWR control register 1 (PWR_CR1)

This register is protected against non-secure access when LPMSEC = 1 in PWR_SECCFGR.

This register is protected against unprivileged access when LPMSEC = 1 and SPRIV = 1 in PWR_PRIVCFGR, or when LPMSEC = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x00

Reset value: 0x0000 0000

(reset value not affected by exit 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.	SRAM4PD	SRAM3PD	SRAM2PD	SRAM1PD	ULPMEN	RRSB2	RRSB1	Res.	Res.	LPMS[2:0]		
				rw	rw	rw	rw	rw	rw	rw			rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SRAM4PD**: SRAM4 power down
This bit is used to reduce the consumption by powering off the SRAM4.
0: SRAM4 powered on
1: SRAM4 powered off

Bit 10 **SRAM3PD**: SRAM3 power down
This bit is used to reduce the consumption by powering off the SRAM3.
0: SRAM3 powered on
1: SRAM3 powered off

Bit 9 **SRAM2PD**: SRAM2 power down
This bit is used to reduce the consumption by powering off the SRAM2.
0: SRAM2 powered on
1: SRAM2 powered off

Bit 8 **SRAM1PD**: SRAM1 power down
This bit is used to reduce the consumption by powering off the SRAM1.
0: SRAM1 powered on
1: SRAM1 powered off

Bit 7 **ULPMEN**: BOR ultra-low power mode
This bit is used to reduce the consumption by configuring the BOR in discontinuous mode.
0: BOR operating in continuous (normal) mode in Standby mode
1: BOR operating in discontinuous (ultra-low power) mode in Standby mode

Caution: This bit must be set to reach the lowest power consumption in Standby mode.

Bit 6 **RRSB2**: SRAM2 page 2 retention in Stop 3 and Standby modes

This bit is used to keep the SRAM2 page 2 content in Stop 3 and Standby modes. The SRAM2 page 2 corresponds to the last 56 Kbytes of the SRAM2 (from SRAM2 base address + 0x2000 to SRAM2 base address + 0xFFFF).

0: SRAM2 page2 content not retained in Stop3 and Standby modes

1: SRAM2 page2 content retained in Stop 3 and Standby modes

Note: This bit has no effect in Shutdown mode.
The backup SRAM is also retained when this bit is set.

Bit 5 **RRSB1**: SRAM2 page 1 retention in Stop 3 and Standby modes

This bit is used to keep the SRAM2 page 1 content in Stop 3 and Standby modes. The SRAM2 page 1 corresponds to the first 8 Kbytes of the SRAM2 (from SRAM2 base address to SRAM2 base address + 0x1FFF).

0: SRAM2 page1 content not retained in Stop 3 and Standby modes

1: SRAM2 page1 content retained in Stop 3 and Standby modes

Note: This bit has no effect in Shutdown mode.
The backup SRAM is also retained when this bit is set.

Bits 4:3 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 the CPU enters the Deepsleep mode.

000: Stop 0 mode

001: Stop 1 mode

010: Stop 2 mode

011: Stop 3 mode

10x: Standby mode (Standby mode also entered if LPMS = 11X in PWR_CR1 with BREN = 1 in PWR_BDCR1)

11x: Shutdown mode if BREN = 0 in PWR_BDCR1

10.10.2 PWR control register 2 (PWR_CR2)

This register is protected against non-secure access when LPMSEC = 1 in PWR_SECCFGR.

This register is protected against unprivileged access when LPMSEC = 1 and SPRIV = 1 in PWR_PRIVCFGR, or when LPMSEC = 0 and NSPRIV = 1.

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRDRUN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM3PDS8	SRAM3PDS7	SRAM3PDS6	SRAM3PDS5	SRAM3PDS4	SRAM3PDS3	SRAM3PDS2	SRAM3PDS1
rw								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.	FLASHFWU	SRAM4FWU	PKARAMPDS	PRAMPDS	DMA2DRAMPDS	DC1RAMPDS	ICRAMPDS	Res.	SRAM4PDS	SRAM2PDS2	SRAM2PDS1	Res.	SRAM1PDS3	SRAM1PDS2	SRAM1PDS1
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw		rw	rw	rw

Bit 31 **SRDRUN**: SmartRun domain in Run mode
0: SmartRun domain AHB3 and APB3 clocks disabled by default in Stop 0,1, 2 modes
1: SmartRun domain AHB3 and APB3 clocks kept enabled in Stop 0,1, 2 modes

Bits 30:24 Reserved, must be kept at reset value.

Bit 23 **SRAM3PDS8**: SRAM3 page 8 (64 Kbytes) power-down in Stop modes (Stop 0, 1, 2, 3)
0: SRAM3 page 8 content retained in Stop modes
1: SRAM3 page 8 content lost in Stop modes

Bit 22 **SRAM3PDS7**: SRAM3 page 7 (64 Kbytes) power-down in Stop modes (Stop 0, 1, 2, 3)
0: SRAM3 page 7 content retained in Stop modes
1: SRAM3 page 7 content lost in Stop modes

Bit 21 **SRAM3PDS6**: SRAM3 page 6 (64 Kbytes) power-down in Stop modes (Stop 0, 1, 2, 3)
0: SRAM3 page 6 content retained in Stop modes
1: SRAM3 page 6 content lost in Stop modes

Bit 20 **SRAM3PDS5**: SRAM3 page 5 (64 Kbytes) power-down in Stop modes (Stop 0, 1, 2, 3)
0: SRAM3 page 5 content retained in Stop modes
1: SRAM3 page 5 content lost in Stop modes

Bit 19 **SRAM3PDS4**: SRAM3 page 4 (64 Kbytes) power-down in Stop modes (Stop 0, 1, 2, 3)
0: SRAM3 page 4 content retained in Stop modes
1: SRAM3 page 4 content lost in Stop modes

Bit 18 **SRAM3PDS3**: SRAM3 page 3 (64 Kbytes) power-down in Stop modes (Stop 0, 1, 2, 3)
0: SRAM3 page 3 content retained in Stop modes
1: SRAM3 page 3 content lost in Stop modes

Bit 17 **SRAM3PDS2**: SRAM3 page 2 (64 Kbytes) power-down in Stop modes (Stop 0, 1, 2, 3)
0: SRAM3 page 2 content retained in Stop modes
1: SRAM3 page 2 content lost in Stop modes

Bit 16 **SRAM3PDS1**: SRAM3 page 1 (64 Kbytes) power-down in Stop modes (Stop 0, 1, 2, 3)
0: SRAM3 page 1 content retained in Stop modes
1: SRAM3 page 1 content lost in Stop modes

Bit 15 Reserved, must be kept at reset value.

Bit 14 **FLASHFWU**: Flash memory fast wakeup from Stop 0 and Stop 1 modes
This bit is used to obtain the best trade-off between low-power consumption and wakeup time when exiting the Stop 0 or Stop 1 modes.
When this bit is set, the Flash memory remains in normal mode in Stop 0 and Stop 1 modes, which offers a faster startup time with higher consumption.
0: Flash memory enters low-power mode in Stop 0 and Stop 1 modes (lower-power consumption).
1: Flash memory remains in normal mode in Stop 0 and Stop 1 modes (faster wakeup time).

Bit 13 **SRAM4FWU**: SRAM4 fast wakeup from Stop 0, Stop 1 and Stop 2 modes
This bit is used to obtain the best trade-off between low-power consumption and wakeup time. SRAM4 wakeup time increases the wakeup time when exiting Stop 0, 1 and 2 modes, and also increases the LPDMA access time to SRAM4 during Stop modes.
0: SRAM4 enters low-power mode in Stop 0, 1 and 2 modes (source biasing for lower-power consumption).
1: SRAM4 remains in normal mode in Stop 0, 1 and 2 modes (higher consumption but no SRAM4 wakeup time).

Bit 12 **PKARAMPDS**: PKA SRAM power-down in Stop modes (Stop 0, 1, 2, 3)
0: PKA SRAM content retained in Stop modes
1: PKA SRAM content lost in Stop modes

Bit 11 **PRAMPDS**: FMAC, FDCAN and USB peripherals SRAM power-down in Stop modes (Stop 0, 1, 2, 3)
0: FMAC, FDCAN and USB peripherals SRAM content retained in Stop modes
1: FMAC, FDCAN and USB peripherals SRAM content lost in Stop modes

Bit 10 **DMA2DRAMPDS**: DMA2D SRAM power-down in Stop modes (Stop 0, 1, 2, 3)
0: DMA2D SRAM content retained in Stop modes
1: DMA2D SRAM content lost in Stop modes

Bit 9 **DC1RAMPDS**: DCACHE1 SRAM power-down in Stop modes (Stop 0, 1, 2, 3)
0: DCACHE1 SRAM content retained in Stop modes
1: DCACHE1 SRAM content lost in Stop modes

Bit 8 **ICRAMPDS**: ICACHE SRAM power-down in Stop modes (Stop 0, 1, 2, 3)
0: ICACHE SRAM content retained in Stop modes
1: ICACHE SRAM content lost in Stop modes

Bit 7 Reserved, must be kept at reset value.

Bit 6 **SRAM4PDS**: SRAM4 power-down in Stop modes (Stop 0, 1, 2, 3)
0: SRAM4 content retained in Stop modes
1: SRAM4 content lost in Stop modes

Bit 5 **SRAM2PDS2**: SRAM2 page 2 (56 Kbytes) power-down in Stop modes (Stop 0, 1, 2)
0: SRAM2 page 2 content retained in Stop modes
1: SRAM2 page 2 content lost in Stop modes
Note: The SRAM2 page 2 retention in Stop 3 is controlled by RRSB2 bit in PWR_CR1.

Bit 4 **SRAM2PDS1**: SRAM2 page 1 (8 Kbytes) power-down in Stop modes (Stop 0, 1, 2)
0: SRAM2 page 1 content retained in Stop modes
1: SRAM2 page 1 content lost in Stop modes
Note: The SRAM2 page 1 retention in Stop 3 is controlled by RRSB1 bit in PWR_CR1.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SRAM1PDS3**: SRAM1 page 3 (64 Kbytes) power-down in Stop modes (Stop 0, 1, 2, 3)
0: SRAM1 page 3 content retained in Stop modes
1: SRAM1 page 3 content lost in Stop modes

Bit 1 **SRAM1PDS2**: SRAM1 page 2 (64 Kbytes) power-down in Stop modes (Stop 0, 1, 2, 3)
0: SRAM1 page 2 content retained in Stop modes
1: SRAM1 page 2 content lost in Stop modes

Bit 0 **SRAM1PDS1**: SRAM1 page 1 (64 Kbytes) power-down in Stop modes (Stop 0, 1, 2, 3)
0: SRAM1 page 1 content retained in Stop modes
1: SRAM1 page 1 content lost in Stop modes

10.10.3 PWR control register 3 (PWR_CR3)

This register is protected against non-secure access when VDMSEC = 1 in PWR_SECCFGR.

This register is protected against unprivileged access when VDMSEC = 1 and SPRIV = 1 in PWR_PRIVCFGR, or when VDMSEC = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x08

Power-on reset value: 0x0000 0000

Exit from Standby modes: not affected

System reset: not affected, except REGSEL that is 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.	Res.
15	14	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.	FSTEN	REGSEL	Res.
													rw	rw	

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **FSTEN**: Fast soft start
- 0: LDO/SMPS fast startup disabled (limited inrush current)
- 1: LDO/SMPS fast startup enabled

Bit 1 **REGSEL**: Regulator selection
- 0: LDO selected
- 1: SMPS selected

Note: REGSEL is reserved and must be kept at reset value in packages without SMPS.

Bit 0 Reserved, must be kept at reset value.

10.10.4 PWR voltage scaling register (PWR_VOSR)

Some register fields are protected against non-secure access depending on RCC_SECCFGR.

These fields can be protected against unprivileged access depending on PWR_PRIVCFGR.

Address offset: 0x0C

Reset value: 0x0000 8000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOSTEN	VOS[1:0]	
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VOSRDY	BOOSTRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r														

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

Bit 18 **BOOSTEN**: EPOD booster enable

This bit is protected against non-secure access when SYSCLKSEC = 1 in RCC_SECCFGR. It is protected against unprivileged access when SYSCLKSEC = 1 in RCC_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SYSCLKSEC = 0 and NSPRIV = 1.

This bit must be set in range 1 and range 2 before increasing the system clock frequency above 55 MHz.

This bit is reset when going into Stop modes (0, 1, 2, 3).

0: Booster disabled

1: Booster enabled

Bits 17:16 **VOS[1:0]**: Voltage scaling range selection

This field is protected against non-secure access when SYSCLKSEC = 1 in RCC_SECCFGR. It is protected against unprivileged access when SYSCLKSEC = 1 in RCC_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SYSCLKSEC = 0 and NSPRIV = 1.

00: Range 4 (lowest power)

01: Range 3

10: Range 2

11: Range 1 (highest frequency)

Bit 15 **VOSRDY:** Ready bit for V_{CORE} voltage scaling output selection

0: Not ready, voltage level < VOS selected level

1: Ready, voltage level ≥ VOS selected level

Bit 14 **BOOSTRDY:** EPOD booster ready

This bit is set to 1 by hardware when the power booster startup time is reached. The system clock frequency can be switched higher than 55 MHz only after this bit is set.

0: Power booster not ready

1: Power booster ready

Bits 13:0 Reserved, must be kept at reset value.

10.10.5 PWR supply voltage monitoring control register (PWR_SVMCR)

This register is protected against non-secure access when VDMSEC = 1 in PWR_SECCFGR.

This register is protected against unprivileged access when VDMSEC = 1 and SPRIV = 1 in PWR_PRIVCFGR, or when VDMSEC = 0 and NSPRIV = 1.

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ASV	IO2SV	USV	AVM2EN	AVM1EN	IO2VMEN	UVMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDLS[2:0]			PVDE	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ASV**: V_{DDA} independent analog supply valid
This bit is used to validate the V_{DDA} supply for electrical and logical isolation purpose. Setting this bit is mandatory to use the analog peripherals. If V_{DDA} is not always present in the application, the V_{DDA} voltage monitor can be used to determine whether this supply is ready or not.
0: V_{DDA} not present: logical and electrical isolation is applied to ignore this supply.
1: V_{DDA} valid

Bit 29 **IO2SV**: 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 V_{DDIO2} voltage monitor can be used to determine whether this supply is ready or not.
0: V_{DDIO2} not present: logical and electrical isolation is applied to ignore this supply.
1: V_{DDIO2} valid

Bit 28 **USV**: V_{DDUSB} independent 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 OTG peripheral. If V_{DDUSB} is not always present in the application, the V_{DDUSB} voltage monitor can be used to determine whether this supply is ready or not.
0: V_{DDUSB} not present: logical and electrical isolation is applied to ignore this supply.
1: V_{DDUSB} valid

Bit 27 **AVM2EN**: V_{DDA} independent analog supply voltage monitor 2 enable (1.8 V threshold)
0: V_{DDA} voltage monitor 2 disabled
1: V_{DDA} voltage monitor 2 enabled

Bit 26 **AVM1EN**: V_{DDA} independent analog supply voltage monitor 1 enable (1.6 V threshold)
0: V_{DDA} voltage monitor 1 disabled
1: V_{DDA} voltage monitor 1 enabled

Bit 25 **IO2VMEN**: V_{DDIO2} independent I/Os voltage monitor enable
0: V_{DDIO2} voltage monitor disabled
1: V_{DDIO2} voltage monitor enabled

Bit 24 **UVMEN**: V_{DDUSB} independent USB voltage monitor enable
0: V_{DDUSB} voltage monitor disabled
1: V_{DDUSB} voltage monitor enabled

Bits 23:8 Reserved, must be kept at reset value.

Bits 7:5 **PVDLS[2:0]**: Programmable voltage detector (PVD) level selection
These bits select the voltage threshold detected by the PVD:
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)

Bit 4 **PVDE**: Programmable voltage detector enable
0: PVD disabled
1: PVD enabled

Bits 3:0 Reserved, must be kept at reset value.

10.10.6 PWR wakeup control register 1 (PWR_WUCR1)

Each register bit WUPENx (x = 1 to 8) is protected against non-secure access when WUPxSEC = 1 in PWR_SECCFGR.

Each bit WUPENx is protected against unprivileged access when WUPxSEC = 1 in PWR_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when WUPxSEC = 0 and NSPRIV =1 .

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x14

Reset value: 0x0000 0000

(reset value not affected by exit 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.	WUPEN8	WUPEN7	WUPEN6	WUPEN5	WUPEN4	WUPEN3	WUPEN2	WUPEN1
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 7 **WUPEN8**: Wakeup pin WKUP8 enable
0: Wakeup pin WKUP8 disabled
1: Wakeup pin WKUP8 enabled

Bit 6 **WUPEN7**: Wakeup pin WKUP7 enable
0: Wakeup pin WKUP7 disabled
1: Wakeup pin WKUP7 enabled

Bit 5 **WUPEN6**: Wakeup pin WKUP6 enable
0: Wakeup pin WKUP6 disabled
1: Wakeup pin WKUP6 enabled

Bit 4 **WUPEN5**: Wakeup pin WKUP5 enable
0: Wakeup pin WKUP5 disabled
1: Wakeup pin WKUP5 enabled

Bit 3 **WUPEN4**: Wakeup pin WKUP4 enable
0: Wakeup pin WKUP4 disabled
1: Wakeup pin WKUP4 enabled

Bit 2 **WUPEN3**: Wakeup pin WKUP3 enable
0: Wakeup pin WKUP3 disabled
1: Wakeup pin WKUP3 enabled

Bit 1 **WUPEN2**: Wakeup pin WKUP2 enable
0: Wakeup pin WKUP2 disabled
1: Wakeup pin WKUP2 enabled

Bit 0 **WUPEN1**: Wakeup pin WKUP1 enable
0: Wakeup pin WKUP1 disabled
1: Wakeup pin WKUP1 enabled

10.10.7 PWR wakeup control register 2 (PWR_WUCR2)

Each register bit WUPPx (x = 1 to 8) is protected against non-secure access when WUPxSEC = 1 in PWR_SECCFGR.

Each bit WUPPx is protected against unprivileged access when WUPxSEC = 1 in PWR_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when WUPxSEC = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x18

Reset value: 0x0000 0000

(reset value not affected by exit 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.	WUPP8	WUPP7	WUPP6	WUPP5	WUPP4	WUPP3	WUPP2	WUPP1
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 7 **WUPP8**: Wakeup pin WKUP8 polarity
This bit must be configured when WUPEN8 = 0.
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

Bit 6 **WUPP7**: Wakeup pin WKUP7 polarity
This bit must be configured when WUPEN7 = 0.
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

Bit 5 **WUPP6**: Wakeup pin WKUP6 polarity
This bit must be configured when WUPEN6 = 0.
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

Bit 4 **WUPP5**: Wakeup pin WKUP5 polarity
This bit must be configured when WUPEN5 = 0.
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

Bit 3 **WUPP4**: Wakeup pin WKUP4 polarity
This bit must be configured when WUPEN4 = 0.
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

Bit 2 **WUPP3**: Wakeup pin WKUP3 polarity
This bit must be configured when WUPEN3 = 0.
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

Bit 1 **WUPP2**: Wakeup pin WKUP2 polarity
This bit must be configured when WUPEN2 = 0.
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

Bit 0 **WUPP1**: Wakeup pin WKUP1 polarity.
This bit must be configured when WUPEN1 = 0.
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

10.10.8 PWR wakeup control register 3 (PWR_WUCR3)

Each register field WUSELx (x = 1 to 8) is protected against non-secure access when WUPxSEC = 1 in PWR_SECCFGR.

Each field WUSELx is protected against unprivileged access when WUPxSEC = 1 in PWR_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when WUPxSEC = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x1C

Reset value: 0x0000 0000

(reset value not affected by exit from Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUSEL8[1:0]		WUSEL7[1:0]		WUSEL6[1:0]		WUSEL5[1:0]		WUSEL4[1:0]		WUSEL3[1:0]		WUSEL2[1:0]		WUSEL1[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 15:14 **WUSEL8[1:0]**: Wakeup pin WKUP8 selection
This field must be configured when WUPEN8 = 0.
00: WKUP8_0
01: WKUP8_1
10: WKUP8_2
11: WKUP8_3

Bits 13:12 **WUSEL7[1:0]**: Wakeup pin WKUP7 selection
This field must be configured when WUPEN7 = 0.
00: WKUP7_0
01: WKUP7_1
10: WKUP7_2
11: WKUP7_3

Bits 11:10 **WUSEL6[1:0]**: Wakeup pin WKUP6 selection
This field must be configured when WUPEN6 = 0.
00: WKUP6_0
01: WKUP6_1
10: WKUP6_2
11: WKUP6_3

Bits 9:8 **WUSEL5[1:0]**: Wakeup pin WKUP5 selection
This field must be configured when WUPEN5 = 0.
00: WKUP5_0
01: WKUP5_1
10: WKUP5_2
11: WKUP5_3

Bits 7:6 **WUSEL4[1:0]**: Wakeup pin WKUP4 selection
This field must be configured when WUPEN4 = 0.
00: WKUP4_0
01: WKUP4_1
10: WKUP4_2
11: WKUP4_3

Bits 5:4 **WUSEL3[1:0]**: Wakeup pin WKUP3 selection
This field must be configured when WUPEN3 = 0.
00: WKUP3_0
01: WKUP3_1
10: WKUP3_2
11: WKUP3_3

Bits 3:2 **WUSEL2[1:0]**: Wakeup pin WKUP2 selection
This field must be configured when WUPEN2 = 0.
00: WKUP2_0
01: WKUP2_1
10: WKUP2_2
11: WKUP2_3

Bits 1:0 **WUSEL1[1:0]**: Wakeup pin WKUP1 selection
This field must be configured when WUPEN1 = 0.
00: WKUP0_0
01: WKUP0_1
10: WKUP0_2
11: WKUP0_3

10.10.9 PWR Backup domain control register 1 (PWR_BDCR1)

This register is write-protected when DBP is cleared in PWR_DBPR.

This register is protected against non-secure access when VBSEC = 1 in PWR_SECCFGR.

This register is protected against unprivileged access when VBSEC = 1 and SPRIV = 1 in PWR_PRIVCFGR, or when VBSEC = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x20

Backup domain reset value: 0x0000 0000

Power-on reset: not affected

Exit from Standby modes: not affected

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.	MONEN	Res.	Res.	Res.	BREN
											rw				rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **MONEN**: Backup domain voltage and temperature monitoring enable
0: Backup domain voltage and temperature monitoring disabled
1: Backup domain voltage and temperature monitoring enabled

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **BREN:** Backup RAM retention in Standby and V_{BAT} modes
When this bit is set, the backup RAM content is kept in Standby(1) and V_{BAT} modes.
If BREN is reset, the backup RAM can still be used in Run, Sleep and Stop modes. However, its content is lost in Standby, Shutdown and V_{BAT} modes. This bit can be written only when the regulator is LDO, which must be configured before switching to SMPS.
0: Backup RAM content lost in Standby(1) and V_{BAT} modes
1: Backup RAM content preserved in Standby and V_{BAT} modes
Note: Backup RAM cannot be preserved in Shutdown mode.

1. The Backup SRAM content is lost in Standby mode without SRAM2 retention. If either RRSB1 or RRSB2 bit is set in Standby mode, the backup SRAM is also retained.

10.10.10 PWR Backup domain control register 2 (PWR_BDCR2)

This register is protected against non-secure access when VBSEC = 1 in PWR_SECCFGR.

This register is protected against unprivileged access when VBSEC = 1 and SPRIV = 1 in PWR_PRIVCFGR, or when VBSEC = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x24

Power-on reset value: 0x0000 0000

Exit from Standby modes: not affected

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.	Res.	Res.	VBRS	VBE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **VBRS**: V_{BAT} charging resistor selection
0: Charge V_{BAT} through a 5 kΩ resistor
1: Charge V_{BAT} through a 1.5 kΩ resistor

Bit 0 **VBE**: V_{BAT} charging enable
0: V_{BAT} battery charging disabled
1: V_{BAT} battery charging enabled

10.10.11 PWR disable Backup domain register (PWR_DBPR)

This register is protected against non-secure access when VBSEC = 1 in PWR_SECCFGR.

This register is protected against unprivileged access when VBSEC = 1 and SPRIV = 1 in PWR_PRIVCFGR, or when VBSEC = 0 and NSPRIV = 1.

Address offset: 0x28

Reset value: 0x0000 0000

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

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

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

Bit 0 **DBP**: Disable Backup domain write protection

In reset state, all registers and SRAM in Backup domain are protected against parasitic write access. This bit must be set to enable the write access to these registers.

0: Write access to Backup domain disabled

1: Write access to Backup domain enabled

10.10.12 PWR USB Type-C and Power Delivery register (PWR_UCPDR)

This register is protected against non-secure access when UCPD1SEC = 1 in TZSC_SECCFGR.

This register is protected against unprivileged access when UCPD1SEC = 1 and SPRIV = 1 in PWR_PRIVCFGR, or when UCPD1SEC = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x2C

Reset value: 0x0000 0000

(reset value not affected by exit from Standby modes)

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

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **UCPD_STBY**: UCPD Stop 3 and Standby modes

When set, this bit is used to memorize the UCPD configuration in Stop 3 and Standby modes. This bit must be written to 1 just before entering Stop 3 or Standby mode when using UCPD. It must be written to 0 after exiting Stop 3 or Standby mode and before writing any UCPD registers.

Bit 0 **UCPD_DBDIS**: UCPD dead battery disable

After exiting reset, the USB Type-C “dead battery” behavior is enabled, which may have a pull-down effect on CC1 and CC2 pins. It is recommended to disable it in all cases, either to stop this pull-down or to handover control to the UCPD (the UCPD must be initialized before doing the disable).

0: UCPD dead battery pull-down behavior enabled on UCPDx_CC1 and UCPDx_CC2 pins

1: UCPD dead battery pull-down behavior disabled on UCPDx_CC1 and UCPDx_CC2 pins

10.10.13 PWR security configuration register (PWR_SECCFGR)

This register can be written only when the access is secure. It can be read by secure or non-secure access.

This register is write-protected against unprivileged write access when SPRIV = 1 in PWR_PRIVCFGR.

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
APCSEC	VBSEC	VDMSEC	LPMSEC	Res.	Res.	Res.	Res.	WUP8SEC	WUP7SEC	WUP6SEC	WUP5SEC	WUP4SEC	WUP3SEC	WUP2SEC	WUP1SEC
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **APCSEC**: Pull-up/pull-down secure protection

0: PWR_APCR can be read and written with secure or non-secure access.

1: PWR_APCR can be read and written only with secure access.

Bit 14 **VBSEC**: Backup domain secure protection

0: PWR_BDCR1, PWR_BDCR2 and PWR_DBPR can be read and written with secure or non-secure access.

1: PWR_BDCR1, PWR_BDCR2 and PWR_DBPR can be read and written only with secure access.

Bit 13 **VDMSEC**: Voltage detection and monitoring secure protection

0: PWR_SVMCR and PWR_CR3 can be read and written with secure or non-secure access.

1: PWR_SVMCR and PWR_CR3 can be read and written only with secure access.

Bit 12 **LPMSEC**: Low-power modes secure protection

0: PWR_CR1, PWR_CR2 and CSSF in the PWR_SR can be read and written with secure or non-secure access.

1: PWR_CR1, PWR_CR2, and CSSF in the PWR_SR can be read and written only with secure access.

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **WUP8SEC**: WUP8 secure protection

0: Bits related to the WKUP8 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written with secure or non-secure access.

1: Bits related to the WKUP8 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written only with secure access.

Bit 6 **WUP7SEC**: WUP7 secure protection

0: Bits related to the WKUP7 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written with secure or non-secure access.

1: Bits related to the WKUP7 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written only with secure access.

Bit 5 **WUP6SEC**: WUP6 secure protection

0: Bits related to the WKUP6 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written with secure or non-secure access.

1: Bits related to the WKUP6 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written only with secure access.

Bit 4 **WUP5SEC**: WUP5 secure protection

0: Bits related to the WKUP5 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written with secure or non-secure access.

1: Bits related to the WKUP5 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written only with secure access.

Bit 3 **WUP4SEC**: WUP4 secure protection

0: Bits related to the WKUP4 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written with secure or non-secure access.

1: Bits related to the WKUP4 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written only with secure access.

Bit 2 **WUP3SEC**: WUP3 secure protection

0: Bits related to the WKUP3 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written with secure or non-secure access.

1: Bits related to the WKUP3 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written only with secure access.

Bit 1 **WUP2SEC**: WUP2 secure protection

0: Bits related to the WKUP2 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written with secure or non-secure access.

1: Bits related to the WKUP2 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written only with secure access.

Bit 0 **WUP1SEC**: WUP1 secure protection

0: Bits related to the WKUP1 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written with secure or non-secure access.

1: Bits related to the WKUP1 pin in PWR_WUCR1, PWR_WUCR2, PWR_WUCR3 and PWR_WUSCR can be read and written only with secure access.

10.10.14 PWR privilege control register (PWR_PRIVCFGR)

This register can be written only when the access is privileged. It can be read by privileged or unprivileged access.

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NSPRIV	SPRIV
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **NSPRIV**: PWR non-secure functions privilege configuration

This bit is set and reset by software. It can be written only by privileged access, secure or non-secure.

0: Read and write to PWR non-secure functions can be done by privileged or unprivileged access.

1: Read and write to PWR non-secure functions can be done by privileged access only.

Bit 0 **SPRIV**: PWR secure functions privilege configuration

This bit is set and reset by software. It can be written only by a secure privileged access.

0: Read and write to PWR secure functions can be done by privileged or unprivileged access.

1: Read and write to PWR secure functions can be done by privileged access only.

10.10.15 PWR status register (PWR_SR)

Some register fields are protected against non-secure access depending on PWR_SECCFGR.

Some register fields are protected against unprivileged access depending on PWR_PRIVCFGR.

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SBF	STOPF	CSSF
													r	r	w

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **SBF**: Standby flag

This bit is set by hardware when the device enters the Standby mode, and is cleared by writing 1 to the CSSF bit, or by a power-on reset. It is not cleared by the system reset.

0: The device did not enter Standby mode.

1: The device entered Standby mode.

Bit 1 **STOPF**: Stop flag

This bit is set by hardware when the device enters a Stop mode, and is cleared by software by writing 1 to the CSSF bit.

0: The device did not enter any Stop mode.

1: The device entered a Stop mode.

Bit 0 **CSSF**: Clear Stop and Standby flags

This bit is protected against non-secure access when LPMSEC = 1 in PWR_SECCFGR.

This bit is protected against unprivileged access when LPMSEC = 1 and SPRIV = 1 in PWR_PRIVCFGR, or when LPMSEC = 0 and NSPRIV = 1.

Writing 1 to this bit clears the STOPF and SBF flags.

10.10.16 PWR supply voltage monitoring status register (PWR_SVMSR)

Address offset: 0x3C

Reset value: 0x0000 8000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VDDA2RDY	VDDA1RDY	VDDIO2RDY	VDDUSBRDY	Res.	Res.	Res.	Res.	Res.	Res.	ACTVOS[1:0]	
				r	r	r	r							r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACTVOSRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDO	Res.	Res.	REGS	Res.
r											r			r	

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

Bit 27 **VDDA2RDY**: V_{DDA} ready versus 1.8 V voltage monitor

0: V_{DDA} is below the threshold of the V_{DDA} voltage monitor 2 (around 1.8 V).

1: V_{DDA} is equal or above the threshold of the V_{DDA} voltage monitor 2 (around 1.8 V).

Bit 26 **VDDA1RDY**: V_{DDA} ready versus 1.6V voltage monitor

0: V_{DDA} is below the threshold of the V_{DDA} voltage monitor 1 (around 1.6 V).

1: V_{DDA} is equal or above the threshold of the V_{DDA} voltage monitor 1 (around 1.6 V).

Bit 25 **VDDIO2RDY**: V_{DDIO2} ready

0: V_{DDIO2} is below the threshold of the V_{DDIO2} voltage monitor.

1: V_{DDIO2} is equal or above the threshold of the V_{DDIO2} voltage monitor.

Bit 24 **VDDUSBRDY**: V_{DDUSB} ready

0: V_{DDUSB} is below the threshold of the V_{DDUSB} voltage monitor.

1: V_{DDUSB} is equal or above the threshold of the V_{DDUSB} voltage monitor.

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **ACTVOS[1:0]**: VOS currently applied to V_{CORE}
This field provides the last VOS value.
00: Range 4 (lowest power)
01: Range 3
10: Range 2
11: Range 1 (highest frequency)

Bit 15 **ACTVOSRDY**: Voltage level ready for currently used VOS
0: V_{CORE} is above or below the current voltage scaling provided by ACTVOS[1:0].
1: V_{CORE} is equal to the current voltage scaling provided by ACTVOS[1:0]

Bits 14:5 Reserved, must be kept at reset value.

Bit 4 **PVDO**: Programmable voltage detector output
0: V_{DD} is equal or above the PVD threshold selected by PVDLS[2:0].
1: V_{DD} is below the PVD threshold selected by PVDLS[2:0].

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **REGS**: Regulator selection
0: LDO selected
1: SMPS selected

Bit 0 Reserved, must be kept at reset value.

10.10.17 PWR Backup domain status register (PWR_BDSR)

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x40

Backup domain reset value: 0x0000 0000

Power-on reset: not affected

Exit from Standby modes: not affected
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.	TEMPH	TEMPL	VBATH	Res.
												r	r	r	

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **TEMPH**: Temperature level monitoring versus high threshold
0: Temperature < high threshold
1: Temperature ≥ high threshold

Bit 2 **TEMPL**: Temperature level monitoring versus low threshold
0: Temperature > low threshold
1: Temperature ≤ low threshold

Bit 1 **VBATH**: Backup domain voltage level monitoring versus high threshold
0: Backup domain voltage level < high threshold
1: Backup domain voltage level ≥ high threshold

Bit 0 Reserved, must be kept at reset value.

10.10.18 PWR wakeup status register (PWR_WUSR)

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x44

Reset value: 0x0000 0000

(reset value not affected by exit from Standby modes)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUF8	WUF7	WUF6	WUF5	WUF4	WUF3	WUF2	WUF1
								r	r	r	r	r	r	r	r

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

Bit 7 **WUF8**: Wakeup flag 8
This bit is set when a wakeup event is detected on WKUP8 pin. This bit is cleared by writing 1 in the CWUF8 bit of PWR_WUSCR when WUSEL ≠ 11, or by hardware when WUPEN8 = 0.
If WUSEL = 11, this bit is cleared by hardware when all internal wakeup source are cleared.

Bit 6 **WUF7**: Wakeup flag 7
This bit is set when a wakeup event is detected on WKUP7 pin. This bit is cleared by writing 1 in the CWUF7 bit of PWR_WUSCR when WUSEL ≠ 11, or by hardware when WUPEN7 = 0.
If WUSEL = 11, this bit is cleared by hardware when all internal wakeup source are cleared.

Bit 5 **WUF6**: Wakeup flag 6
This bit is set when a wakeup event is detected on WKUP6 pin. This bit is cleared by writing 1 in the CWUF6 bit of PWR_WUSCR when WUSEL ≠ 11, or by hardware when WUPEN6 = 0.
If WUSEL = 11, this bit is cleared by hardware when all internal wakeup source are cleared.

Bit 4 **WUF5**: Wakeup flag 5
This bit is set when a wakeup event is detected on WKUP5 pin. This bit is cleared by writing 1 in the CWUF5 bit of PWR_WUSCR when WUSEL ≠ 11, or by hardware when WUPEN5 = 0.

Bit 3 **WUF4**: Wakeup flag 4
This bit is set when a wakeup event is detected on WKUP4 pin. This bit is cleared by writing 1 in the CWUF4 bit of PWR_WUSCR when WUSEL ≠ 11, or by hardware when WUPEN4 = 0.

Bit 2 **WUF3**: Wakeup flag 3
This bit is set when a wakeup event is detected on WKUP3 pin. This bit is cleared by writing 1 in the CWUF3 bit of PWR_WUSCR when WUSEL ≠ 11, or by hardware when WUPEN3 = 0.

Bit 1 **WUF2**: Wakeup flag 2

This bit is set when a wakeup event is detected on WKUP2 pin. This bit is cleared by writing 1 in the CWUF2 bit of PWR_WUSCR when WUSEL ≠ 11, or by hardware when WUPEN2 = 0.

Bit 0 **WUF1**: Wakeup flag 1

This bit is set when a wakeup event is detected on WKUP1 pin. This bit is cleared by writing 1 in the CWUF1 bit of PWR_WUSCR when WUSEL ≠ 11, or by hardware when WUPEN1 = 0.

10.10.19 PWR wakeup status clear register (PWR_WUSCR)

Each register bit CWUFx (x = 1 to 8) is protected against non-secure access when WUPxSEC = 1 in PWR_SECCFGR.

Each bit CWUFx is protected against unprivileged access when WUPxSEC = 1 in PWR_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when WUPxSEC = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CWUF 8	CWUF 7	CWUF 6	CWUF 5	CWUF 4	CWUF 3	CWUF 2	CWUF 1
								w	w	w	w	w	w	w	w

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

Bit 7 **CWUF8**: Wakeup flag 8

Writing 1 to this bit clears the WUF8 flag in PWR_WUSR.

Bit 6 **CWUF7**: Wakeup flag 7

Writing 1 to this bit clears the WUF7 flag in PWR_WUSR.

Bit 5 **CWUF6**: Wakeup flag 6

Writing 1 to this bit clears the WUF6 flag in PWR_WUSR.

Bit 4 **CWUF5**: Wakeup flag 5

Writing 1 to this bit clears the WUF5 flag in PWR_WUSR.

Bit 3 **CWUF4**: Wakeup flag 4

Writing 1 to this bit clears the WUF4 flag in PWR_WUSR.

Bit 2 **CWUF3**: Wakeup flag 3

Writing 1 to this bit clears the WUF3 flag in PWR_WUSR.

Bit 1 **CWUF2**: Wakeup flag 2

Writing 1 to this bit clears the WUF2 flag in PWR_WUSR.

Bit 0 **CWUF1**: Wakeup flag 1

Writing 1 to this bit clears the WUF1 flag in PWR_WUSR.

10.10.20 PWR apply pull configuration register (PWR_APCR)

This register is protected against non-secure access when APCSEC = 1 in PWR_SECCFGR.

This register is protected against unprivileged access when APCSEC = 1 and SPRIV = 1 in PWR_PRIVCFGR, or when APCSEC = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x4C

Reset value: 0x0000 0000

(reset value not affected by exit 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.	Res.	Res.	APC
															rw

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

Bit 0 **APC**: Apply pull-up and pull-down configuration

When this bit is set, the I/O pull-up and pull-down configurations defined in PWR_PUCRx and PWR_PDCRx are applied. When this bit is cleared, PWR_PUCRx and PWR_PDCRx are not applied to the I/Os.

10.10.21 PWR port A pull-up control register (PWR_PUCRA)

Each register bit PUy is protected against non-secure access when SECy = 1 in GPIOA_SECCFGR.

Each register bit PUy is protected against unprivileged access when SECy = 1 in GPIOA_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x50

Reset value: 0x0000 0000

(reset value not affected by exit 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
PU15	Res.	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **PU15**: Port A pull-up bit 15
When set, this bit activates the pull-up on PA15 when the APC bit is set in PWR_APCR. The pull-up is not activated if the corresponding PD15 bit is also set.

Bit 14 Reserved, must be kept at reset value.

Bits 13:0 **PUy**: Port A pull-up bit y (y = 13 to 0)
When set, each bit activates the pull-up on PAy when the APC bit is set in PWR_APCR. The pull-up is not activated if the corresponding PDy bit is also set.

10.10.22 PWR port A pull-down control register (PWR_PDCRA)

Each register bit PDy is protected against non-secure access when SECy = 1 in GPIOA_SECCFGR.

Each register bit PDy is protected against unprivileged access when SECy = 1 in GPIOA_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x54

Reset value: 0x0000 0000

(reset value not affected by exit 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.	PD14	Res.	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 14 **PD14**: Port A pull-down bit 14
When set, this bit activates the pull-down on PA14 when the APC bit is set in PWR_APCR.

Bit 13 Reserved, must be kept at reset value.

Bits 12:0 **PDy**: Port A pull-down bit y (y = 12 to 0)
When set, each bit activates the pull-down on PAy when the APC bit is set in PWR_APCR.

10.10.23 PWR port B pull-up control register (PWR_PUCRB)

Each register bit PUy is protected against non-secure access when SECy = 1 in GPIOB_SECCFGR.

Each register bit PUy is protected against unprivileged access when SECy = 1 in GPIOB_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x58

Reset value: 0x0000 0000

(reset value not affected by exit 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
PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PUy**: Port B pull-up bit y (y = 15 to 0)

When set, each bit activates the pull-up on PBy when the APC bit is set in PWR_APCR. The pull-up is not activated if the corresponding PDy bit is also set.

10.10.24 PWR port B pull-down control register (PWR_PDCRB)

Each register bit PDy is protected against non-secure access when SECy = 1 in GPIOB_SECCFGR.

Each register bit PDy is protected against unprivileged access when SECy = 1 in GPIOB_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x5C

Reset value: 0x0000 0000

(reset valuet not affected by exit 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
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	Res.	PD3	PD2	PD1	PD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:5 **PDy**: Port B pull-down bit y (y = 15 to 5)
When set, each bit activates the pull-down on PBy when the APC bit is set in PWR_APCR.

Bit 4 Reserved, must be kept at reset value.

Bits 3:0 **PDy**: Port B pull-down bit y (y = 3 to 0)
When set, each bit activates the pull-down on PBy when the APC bit is set in PWR_APCR.

10.10.25 PWR port C pull-up control register (PWR_PUCRC)

Each register bit PUy is protected against non-secure access when SECy = 1 in GPIOC_SECCFGR.

Each register bit PUy is protected against unprivileged access when SECy = 1 in GPIOC_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x60

Reset value: 0x0000 0000

(reset value not affected by exit 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
PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PUy**: Port C pull-up bit y (y = 15 to 0)
When set, each bit activates the pull-up on PCy when the APC bit is set in PWR_APCR. The pull-up is not activated if the corresponding PDy bit is also set.

10.10.26 PWR port C pull-down control register (PWR_PDCRC)

Each register bit PDy is protected against non-secure access when SECy = 1 in GPIOC_SECCFGR.

Each register bit PDy is protected against unprivileged access when SECy = 1 in GPIOC_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x64

Reset value: 0x0000 0000

(reset value not affected by exit 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
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PDy**: Port C pull-down bit y (y = 15 to 0)
When set, each bit activates the pull-down on PCy when the APC bit is set in PWR_APCR.

10.10.27 PWR port D pull-up control register (PWR_PUCRD)

Each register bit PUy is protected against non-secure access when SECy = 1 in GPIOD_SECCFGR.

Each register bit PUy is protected against unprivileged access when SECy = 1 in GPIOD_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x68

Reset value: 0x0000 0000

(reset value not affected by exit 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
PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PUy**: Port D pull-up bit y (y = 15 to 0)
When set, each bit activates the pull-up on PDy when the APC bit is set in PWR_APCR. The pull-up is not activated if the corresponding PDy bit is also set.

10.10.28 PWR port D pull-down control register (PWR_PDCRD)

Each register bit PDy is protected against non-secure access when SECy = 1 in GPIOD_SECCFGR.

Each register bit PDy is protected against unprivileged access when SECy = 1 in GPIOD_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x6C

Reset value: 0x0000 0000

(reset value not affected by exit 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
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PDy**: Port D pull-down bit y (y = 15 to 0)
When set, each bit activates the pull-down on PDy when the APC bit is set in PWR_APCR.

10.10.29 PWR port E pull-up control register (PWR_PUCRE)

Each register bit PUy is protected against non-secure access when SECy = 1 in GPIOE_SECCFGR.

Each register bit PUy is protected against unprivileged access when SECy = 1 in GPIOE_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x70

Reset value: 0x0000 0000

(reset value not affected by exit 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
PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PUy**: Port E pull-up bit y (y = 15 to 0)

When set, each bit activates the pull-up on PEy when the APC bit is set in PWR_APCR. The pull-up is not activated if the corresponding PDy bit is also set.

10.10.30 PWR port E pull-down control register (PWR_PDCRE)

Each register bit PDy is protected against non-secure access when SECy = 1 in GPIOE_SECCFGR.

Each register bit PDy is protected against unprivileged access when SECy = 1 in GPIOE_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy =0 and NSPRIV =1 .

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x74

Reset value: 0x0000 0000

(reset value not affected by exit 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
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PDy**: Port E pull-down bit y (y = 15 to 0)
When set, each bit activates the pull-down on PEy when the APC bit is set in PWR_APCR.

10.10.31 PWR port F pull-up control register (PWR_PUCRF)

Each register bit PUy is protected against non-secure access when SECy = 1 in GPIOF_SECCFGR.

Each register bit PUy is protected against unprivileged access when SECy = 1 in GPIOF_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x78

Reset value: 0x0000 0000

(reset value not affected by exit 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
PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PUy**: Port F pull-up bit y (y = 15 to 0)
When set, each bit activates the pull-up on PFy when the APC bit is set in PWR_APCR. The pull-up is not activated if the corresponding PDy bit is also set.

10.10.32 PWR port F pull-down control register (PWR_PDCRF)

Each register bit PDy is protected against non-secure access when SECy = 1 ie GPIOF_SECCFGR.

Each register bit PDy is protected against unprivileged access when SECy = 1 in GPIOF_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV =1 .

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x7C

Reset value: 0x0000 0000

(reset value not affected by exit 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
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PDy**: Port F pull-down bit y (y = 15 to 0)
When set, each bit activates the pull-down on PFy when the APC bit is set in PWR_APCR.

10.10.33 PWR port G pull-up control register (PWR_PUCRG)

Each register bit PUy is protected against non-secure access when SECy = 1 in GPIOG_SECCFGR.

Each register bit PUy is protected against unprivileged access when SECy = 1 in GPIOG_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x80

Reset value: 0x0000 0000

(reset value not affected by exit 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
PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PUy**: Port G pull-up bit y (y = 15 to 0)
When set, each bit activates the pull-up on PGy when the APC bit is set in PWR_APCR. The pull-up is not activated if the corresponding PDy bit is also set.

10.10.34 PWR port G pull-down control register (PWR_PDCRG)

Each register bit PDy is protected against non-secure access when SECy = 1 in GPIOG_SECCFGR.

Each register bit PDy is protected against unprivileged access when SECy =1 in GPIOG_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x84

Reset value: 0x0000 0000

(reset value not affected by exit 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
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PDy**: Port G pull-down bit y (y = 15 to 0)
When set, eah bit activates the pull-down on PGy when the APC bit is set in PWR_APCR .

10.10.35 PWR port H pull-up control register (PWR_PUCRH)

Each register bit PUy is protected against non-secure access when SECy = 1 in GPIOH_SECCFGR.

Each register bit PUy is protected against unprivileged access when SECy = 1 in GPIOH_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x88

Reset value: 0x0000 0000

(reset value not affected by exit 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
PU15	PU14	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PUy**: Port H pull-up bit y (y = 15 to 0)

When set, each bit activates the pull-up on PHy when the APC bit is set in PWR_APCR. The pull-up is not activated if the corresponding PDy bit is also set.

10.10.36 PWR port H pull-down control register (PWR_PDCRH)

Each register bit PDy is protected against non-secure access when SECy = 1 in GPIOH_SECCFGR.

Each register bit PDy is protected against unprivileged access when SECy = 1 in GPIOH_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x8C

Reset value: 0x0000 0000

(reset value not affected by exit 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
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PDy**: Port H pull-down bit y (y = 15 to 0)
When set, each bit activates the pull-down on PHy when the APC bit is set in PWR_APCR.

10.10.37 PWR port I pull-up control register (PWR_PUCRI)

Each register bit PUy is protected against non-secure access when SECy = 1 in GPIOI_SECCFGR.

Each register bit PUy is protected against unprivileged access when SECy=1 in GPIOI_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x90

Reset value: 0x0000 0000

(reset value not affected by exit 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.	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **PUy**: Port I pull-up bit y (y = 7 to 0)
When set, each bit activates the pull-up on PIy when the APC bit is set in PWR_APCR. The pull-up is not activated if the corresponding PDy bit is also set.

10.10.38 PWR port I pull-down control register (PWR_PDCRI)

Each register bit PDy is protected against non-secure access when SECy = 1 in GPIOI_SECCFGR.

Each register bit PDy is protected against unprivileged access when SECy = 1 in GPIOI_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when SECy = 0 and NSPRIV = 1.

Access: 14 AHB clock cycles added compared to a standard AHB access

Address offset: 0x94

Reset value: 0x0000 0000

(reset value not affected by exit 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.	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **PDy**: Port I pull-down bit y (y = 7 to 0)

When set, each bit activates the pull-down on PIy when the APC bit is set in PWR_APCR.

10.10.39 PWR register map

Table 100. PWR register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**PWR_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM4PD	SRAM3PD	SRAM2PD	SRAM1PD	ULPMEN	RRSB2	RRSB1	Res.	Res.	LPMS [2:0]		
	Reset value																					0	0	0	0	0	0	0			0	0	0
0x04	**PWR_CR2**	SRDRUN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM3PDS8	SRAM3PDS7	SRAM3PDS6	SRAM3PDS5	SRAM3PDS4	SRAM3PDS3	SRAM3PDS2	SRAM3PDS1	Res.	FLASHFWU	SRAM4FWU	PKARAMPDS	PRAMPDS	DMA2DRAMPDS	DC1RAMPDS	ICRAMPDS	Res.	SRAM4PDS	SRAM2PDS2	SRAM2PDS1	Res.	SRAM1PDS3	SRAM1PDS2	SRAM1PDS1
	Reset value	0								0	0	0	0	0	0	0	0		0	0	0	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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSTEN	REGSEL	Res.
	Reset value																														0	0	
0x0C	**PWR_VOSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOSTEN	VOS[1:0]		VOSRDY	BOOSTRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0	1	0														

Table 100. PWR register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x10	**PWR_SVMCR**	Res.	ASV	IO2SV	USV	AVM2EN	AVM1EN	IO2VMEN	UVMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDLS[2:0]			PVDE	Res.	Res.	Res.	Res.
	Reset value		0	0	0	0	0	0	0																	0	0	0	0				
0x14	**PWR_WUCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUPEN8	WUPEN7	WUPEN6	WUPEN5	WUPEN4	WUPEN3	WUPEN2	WUPEN1
	Reset value																									0	0	0	0	0	0	0	0
0x18	**PWR_WUCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUPP8	WUPP7	WUPP6	WUPP5	WUPP4	WUPP3	WUPP2	WUPP1
	Reset value																									0	0	0	0	0	0	0	0
0x1C	**PWR_WUCR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUSEL8[1:0]		WUSEL7[1:0]		WUSEL6[1:0]		WUSEL5[1:0]		WUSEL4[1:0]		WUSEL3[1:0]		WUSEL2[1:0]		WUSEL1[1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**PWR_BDCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MONEN	Res.	Res.	Res.	BREN
	Reset value																												0				0
0x24	**PWR_BDCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBRS	VBE
	Reset value																															0	0
0x28	**PWR_DBPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBP
	Reset value																																0
0x2C	**PWR_UCPDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD_STBY	UCPD_DBDIS
	Reset value																															0	0
0x30	**PWR_SECCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APCSEC	VBSEC	VDMSEC	LPMSEC	Res.	Res.	Res.	Res.	WUP8SEC	WUP7SEC	WUP6SEC	WUP5SEC	WUP4SEC	WUP3SEC	WUP2SEC	WUP1SEC
	Reset value																	0	0	0	0					0	0	0	0	0	0	0	0
0x34	**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.	NSPRIV	SPRIV
	Reset value																															0	0
0x38	**PWR_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.	SBF	STOPF	CSSF
	Reset value																														0	0	0
0x3C	**PWR_SVMSR**	Res.	Res.	Res.	Res.	VDDA2RDY	VDDA1RDY	VDDIO2RDY	VDDUSBRDY.	Res.	Res.	Res.	Res.	Res.	Res.	ACTVOS [1:0]		ACTVOSRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDO	Res.	Res.	REGS	Res.
	Reset value					0	0	0	0							0	0	1											0			0	

Table 100. PWR register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x40	**PWR_BDSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TEMPH	TEMPL	VBATH	Res.
	Reset value																													0	0	0	
0x44	**PWR_WUSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUF8	WUF7	WUF6	WUF5	WUF4	WUF3	WUF2	WUF1
	Reset value																									0	0	0	0	0	0	0	0
0x48	**PWR_WUSCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CWUF8	CWUF7	CWUF6	CWUF5	CWUF4	CWUF3	CWUF2	CWUF1
	Reset value																									0	0	0	0	0	0	0	0
0x4C	**PWR_APCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APC
	Reset value																																0
0x50	**PWR_PUCRA**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU15	Res.	PU13	PU12	PU11	PU10	PU9	PU8	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
	Reset value																	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x54	**PWR_PDCRA**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PD14	Res.	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
	Reset value																		0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x58	**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
0x5C	**PWR_PDCRB**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	Res.	PD3	PD2	PD1	PD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0
0x60	**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
0x64	**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
0x68	**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
0x6C	**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
0x70	**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
0x74	**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
0x78	**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
0x7C	**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
0x80	**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

Table 100. PWR register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x84	**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
0x88	**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
0x8C	**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
0x90	**PWR_PUCRI**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
	Reset value																									0	0	0	0	0	0	0	0
0x94	**PWR_PDCRI**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
	Reset value																									0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

11 Reset and clock control (RCC)

11.1 Introduction

The reset and clock control (RCC) manages the different kind of reset, and generates all clocks for the bus and peripherals.

11.2 RCC pins and internal signals

The table below lists the RCC inputs and output signals connected to package pins or balls.

Table 101. 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
MCO	O	Output clock for external devices
LSCO	O	Low-speed output clock for external devices
AUDIOCLK	I	External kernel clock input for SAI1, SAI2, MDF1 and ADF1

11.3 RCC reset functional description

There are three types of reset:

- a system reset
- a power reset
- a Backup domain reset

11.3.1 Power reset

A power reset is generated when one of the following events occurs:

- a Brownout reset (BOR)
- when exiting Standby mode
- when exiting Shutdown mode

A BOR sets all registers to their reset values except the ones in the Backup domain.

When exiting Standby mode, all registers in the core domain are set to their reset value. Registers outside the core domain (RTC, TAMP, WKUP, IWDG, and Standby/Shutdown mode control) are not impacted.

When exiting Shutdown mode, a Brownout reset is generated, resetting all registers except those in the Backup domain.

11.3.2 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

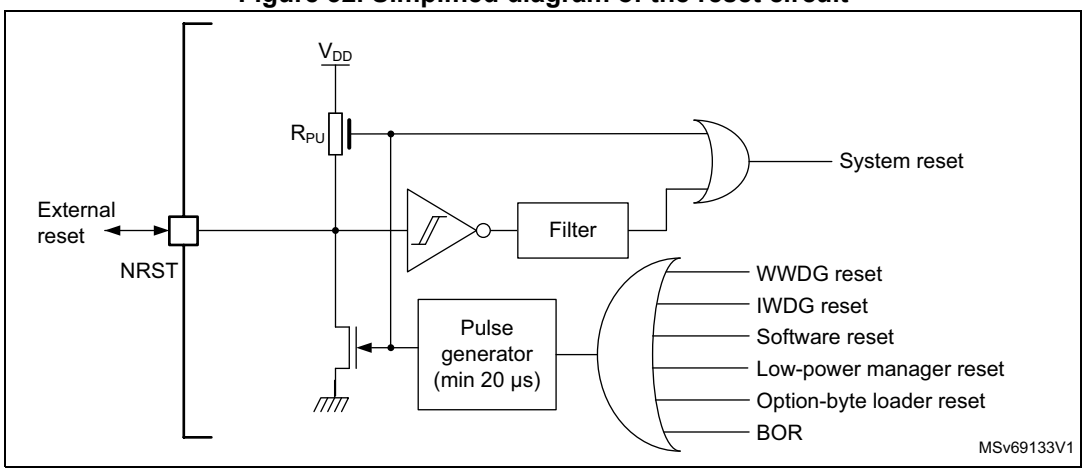
The reset source can be identified by checking the reset flags in *RCC control/status register (RCC_CSR)*.

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 32. 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, the following 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 7.4.1: Option bytes description*.

Option byte loader reset

The option byte loader reset is generated when the OBL_LAUNCH bit is set in the FLASH_NSCR register. This bit is used to launch the option byte loading by software.

11.3.3 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 affects the LSE oscillator, the RTC, the backup registers, the backup SRAM, the SRAM2, all secrets protected by tamper (refer to *Table 528: TAMP interconnection*) and the RCC_BDCR register.

11.4 RCC clocks functional description

Four different clock sources can be used to drive the system clock (SYSCLK):

- HSI16: high-speed internal 16 MHz RC oscillator clock
- MSIS: multi-speed internal RC oscillator clock
- HSE: high-speed external crystal or clock, from 4 to 50 MHz
- PLL1 clock

The MSIS is used as system clock source after startup from reset, configured at 4 MHz.

The devices have the following additional clock sources:

- MSIK: multi-speed internal RC oscillator clock used for peripherals kernel clocks
- LSI: 32 kHz/250 Hz low-speed internal RC that drives the independent watchdog and optionally the RTC used for auto-wakeup from Stop and Standby modes
- LSE: 32.768 kHz low-speed external crystal or clock that optionally drives the real-time clock (rtc_ck)
- HSI48: internal 48 MHz RC that potentially drives the OTG_FS, the SDMMC and the RNG
- SHSI: secure high-speed internal 48 MHz RC that drives the SAES
- PLL2 and PLL3 clocks

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 and APB domains is 160 MHz.

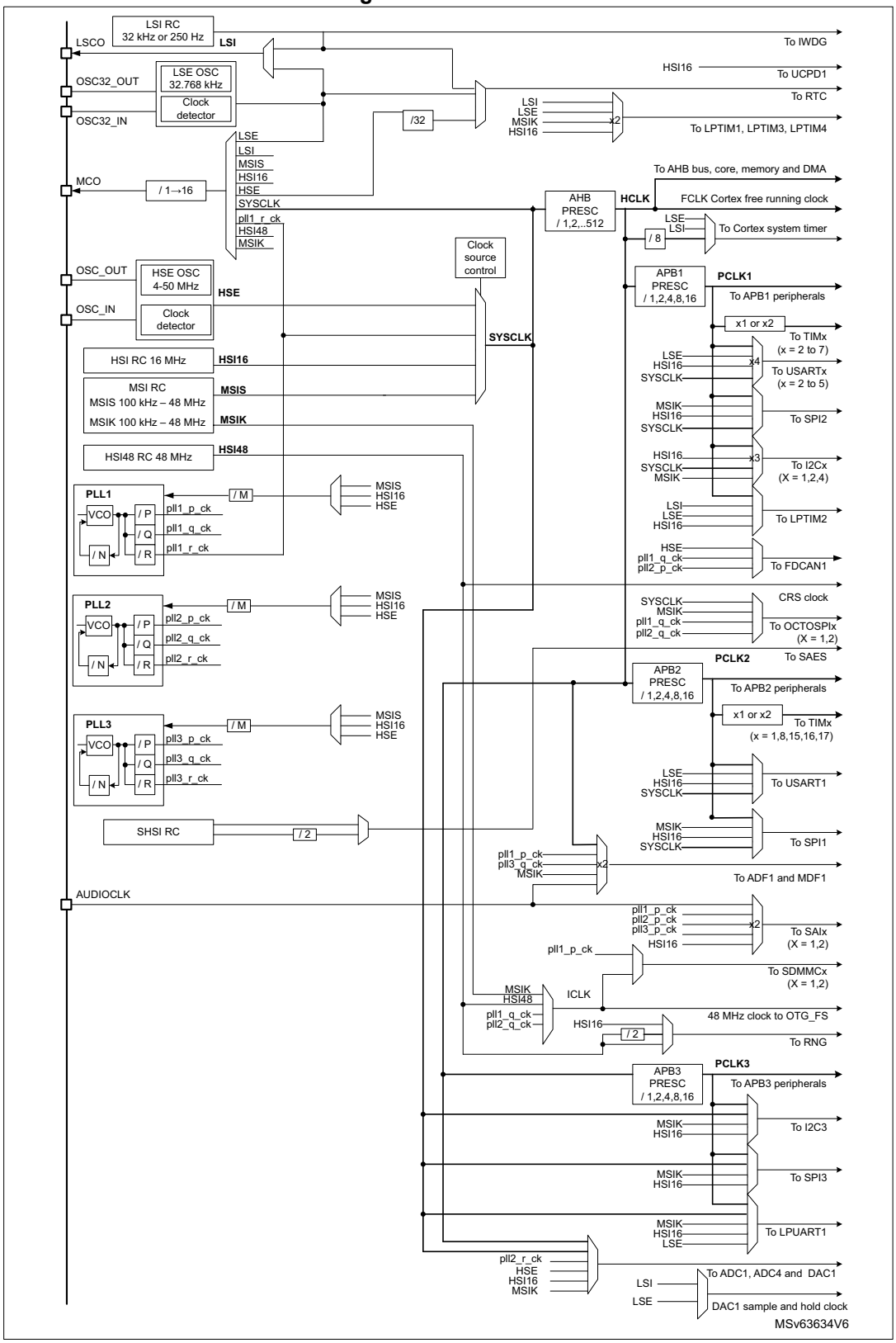
All the peripheral clocks are derived from their bus clock (HCLK, PCLK1, PCLK2 or PCLK3) except the following ones that receive an independent kernel clock. This kernel clock can be selected by software between several sources thanks to RCC_CCIPRx registers (x = 1,2,3): OTG_FS, SDMMCx (x = 1,2), RNG, ADCx (x = 1,4), DAC1, U(S)ARTx (x = 1 to 5), LPUART1, I2Cx (x = 1 to 4), SPIx (x = 1 to 3), OCTOSPIx (x = 1,2), SAIx (x = 1,2), MDF1, ADF1, FDCAN1, LPTIMx (x = 1 to 4), SAES.

In addition, the RTC kernel clock is selected by software in RCC_BDCR. The IWDG clock is always the LSI 32 kHz clock.

The RCC feeds the Cortex system timer (SysTick) external clock with the AHB clock (HCLK) divided by eight, or LSE or LSI. 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 33. Clock tree



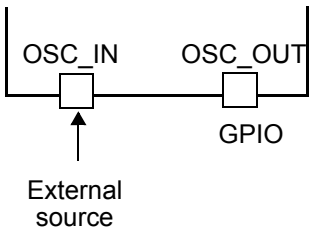
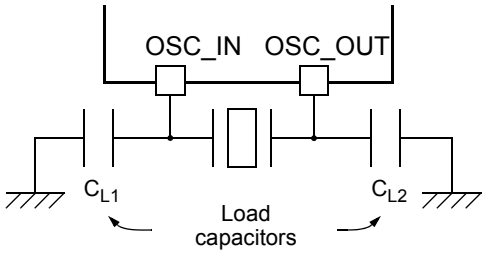
11.4.1 HSE clock

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 must be placed as close as possible to the oscillator pins in order to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected oscillator.

Figure 34. HSE/ LSE clock sources

Clock source	Hardware configuration
External clock	
Crystal/ceramic resonators	

External crystal/ceramic resonator (HSE crystal)

The 4 to 50 MHz external oscillator has the advantage of producing a very accurate rate on the main clock.

The associated hardware configuration is shown in *Figure 34*. 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 50 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 34*).

11.4.2 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, Stop 2 or Stop 3). Refer to *Section 11.8.6: RCC clock configuration register 1 (RCC_CFGR1)*. It can also be used as a backup clock source (auxiliary clock) if the HSE crystal oscillator fails. Refer to *Section 11.4.11: 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 3 (RCC_ICSCR3)*.

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 3 (RCC_ICSCR3)*.

For more details on how to measure the HSI16 frequency variation, refer to *Section 11.4.19: 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.

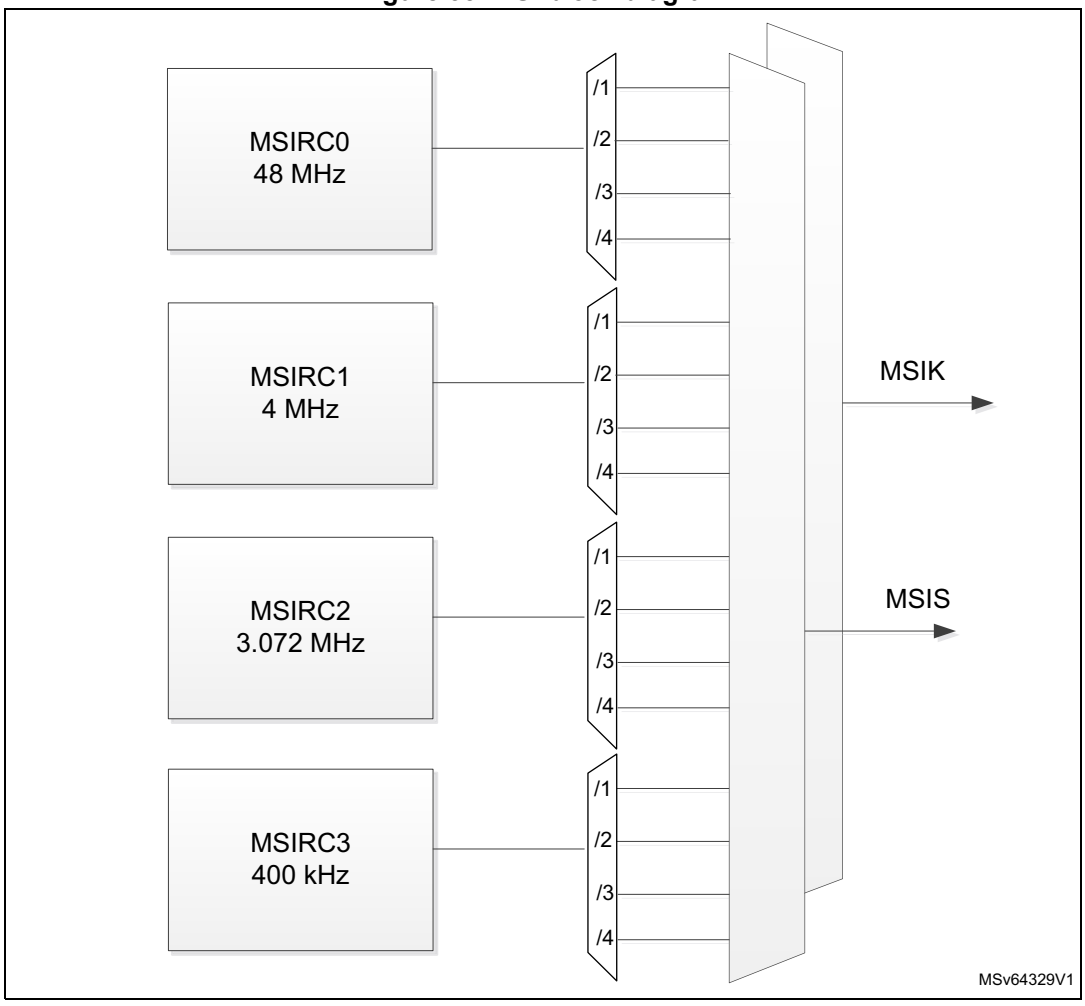
The HSI16 RC can be switched on and off using the HSION bit in the *RCC clock control register (RCC_CR)*.

The HSI16 signal can also be used as a backup source (Auxiliary clock) if the HSE crystal oscillator fails. Refer to *Section 11.4.11: Clock security system (CSS)*.

11.4.3 MSI (MSIS and MSIK) clocks

The MSI is made of four internal RC oscillators: MSIRC0 at 48 MHz, MSIRC1 at 4 MHz, MSIRC2 at 3.072 MHz and MSIRC3 at 400 kHz. Each oscillator feeds a prescaler providing a division by 1, 2, 3 or 4. Two output clocks are generated from these divided oscillators: MSIS, that can be selected as system clock, and MSIK, that can be selected by some peripherals as kernel clock.

Figure 35. MSI block diagram



MSIS and MSIK frequency range can be adjusted by software, by using respectively the MSISRANGE[3:0] and MSIKRANGE[3:0] fields in the *RCC internal clock sources calibration*

register 1 (RCC_ICSCR1), with MSIRGSEL = 1. Sixteen frequency ranges are available, generated from the four internal RCs, as shown in the table below.

Table 102. MSIS and MSIK ranges per internal MSIRCs

MSIRC0	MSIRC1	MSIRC2	MSIRC3
Range 0: 48 MHz	Range 4: 4 MHz	Range 8: 3.072 MHz	Range 12: 400 kHz
Range 1: 24 MHz	Range 5: 2 MHz	Range 9: 1.536 MHz	Range 13: 200 kHz
Range 2: 16 MHz	Range 6: 1.33 MHz	Range 10: 1.024 MHz	Range 14: 133 kHz
Range 3: 12 MHz	Range 7: 1 MHz	Range 11: 0.768 MHz	Range 15: 100 kHz

The MSIS clock is used as system clock after restart from Reset, wakeup from Standby and Shutdown low-power modes. After restart from Reset or when exiting Shutdown mode, the MSIS and MSIK frequencies are set to their default value 4 MHz. The frequency range at wakeup from Standby mode can be adjusted by software, using respectively the MSISSRANGE[3:0] and MSIKSRANGE[3:0] fields with MSIRGSEL=0. Refer to *Section 11.8.50: RCC control/status register (RCC_CSR)*.

The MSIS clock can be selected as system clock after a wakeup from Stop mode (Stop 0, Stop 1, Stop 2 or Stop 3) depending on STOPWUCK in the *RCC clock control register (RCC_CR)*. It can also be used as a backup clock source (auxiliary clock) if the HSE crystal oscillator fails. Refer to *Section 11.4.11: Clock security system (CSS)*.

The MSI 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, the MSI provides a very accurate clock source that can be used by the OTG_FS peripheral, and feeds the PLL.

The MSISRDY and MSIKRDY flags in the *RCC clock control register (RCC_CR)* indicate whether the MSIS and MSIK RC are stable or not. At startup, the MSIS and MSIK RC output clocks are not released until their respective bit is set by hardware. The MSIS and MSIK RC can be switched on and off by using the MSISON and MSIKON bits 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 either the MSIS or the MSIK in a PLL-mode. This mode is enabled:

- for MSIS by setting the MSIPLLEN bit with MSIPLLSEL = 1 in the *RCC clock control register (RCC_CR)*
- for MSIK by setting the MSIPLLEN bit with MSIPLLSEL = 0

In case MSIS and MSIK ranges are generated from the same MSIRC source, the PLL-mode is applied on both MSIS and MSIK. When configured in PLL-mode, the MSIS or MSIK automatically calibrates itself thanks to the LSE. This mode is available for all MSI frequency ranges. At 48 MHz, the MSIK in PLL-mode can be used for the OTG_FS device, avoiding the need of an external high-speed crystal.

If LSE clocks pulses are stopped, the MSI PLL-mode is automatically unlocked, and the MSI accuracy is consequently degraded. On all STM32U575/585 revisions except rev. X, the MSI PLL-mode unlock event is connected to an EXTI line: this is used to generate an event or interrupt supporting wakeup from Stop 0, Stop 1, or Stop 2 mode (see *Table 107: Interrupt sources and control* and *Table 156: EXTI line connections*).

MSI PLL-mode stabilization time

When MSIPLLEN = 1, the final accuracy after enabling the MSI (by writing MSISON = 1 or MSIKON = 1 or following a peripheral clock request in Stop mode) is reached after a stabilization time t_{STAB}(MSI) when MSIPLLFAST = 0. This stabilization time is needed even if the LSE is kept enable. Refer to datasheet for t_{STAB}(MSI) value.

If MSIPLLEN = 1 with MSIPLLFAST =1 , the MSI oscillator is kept powered on when a request to switch it off is received (either by writing MSISON = 0 and MSIKON =0 , or because no peripheral requests this clock in Stop mode). In this case the MSI PLL-mode accuracy is kept when the MSI is switched on again, providing that the t_{STAB}(MSI) stabilization time is reached before switching off the MSI. This mode can be used for autonomous peripherals requiring accuracy in Stop mode, with an extra consumption as the oscillator remains powered on, but gated off when disabled.

Software calibration

The MSIRCx (x = 0 to 3) oscillators 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 MSICALx[4:0] bits (x = 0 to 3) in the *RCC internal clock sources calibration register 1 (RCC_ICSCR1)*. If the application is subject to voltage or temperature variations, this may affect the RC oscillator speed. The MSIRCx frequency can be trimmed in the application by using the MSITRIMx[4:0] bits (x = 0 to 3) in the RCC_ICSCR register.

Note: The final accuracy after applying the calibration value is reached after a stabilization time. This stabilization time is needed after reset of exiting Standby or Shutdown mode. It is also needed when switching from PLL-mode to normal mode.

The hardware auto calibration with LSE must not be used in conjunction with software calibration.

For more details on how to measure the MSI frequency variation, refer to *Section 11.4.19: Internal/external clock measurement with TIM15/TIM16/TIM17*.

11.4.4 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 OTG_FS 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 that is subject to manufacturing process variations.

For more details on how to configure and use the CRS peripheral, refer to *Section 12: Clock recovery system (CRS)*.

The HSI48RDY flag in the RCC_CR 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_CR register.

11.4.5 SHSI clock

The SHSI is an internal securable RC oscillator dedicated to clock the SAES. The SHSIRDY flag in the *RCC clock control register (RCC_CR)* indicates if the SHSI RC is stable or not. At startup, the SHSI RC output clock is not released until this bit is set by hardware.

The SHSI RC can be switched on and off using the SHSION bit in the *RCC clock control register (RCC_CR)*.

11.4.6 PLL

The RCC features three PLLs:

- a main PLL, PLL1, that is generally used to provide clocks to the CPU and to some peripherals
- two dedicated PLL2 and PLL3, that are used to generate the kernel clock for peripherals

The PLLs integrated into the RCC are completely independent. They offer the following features:

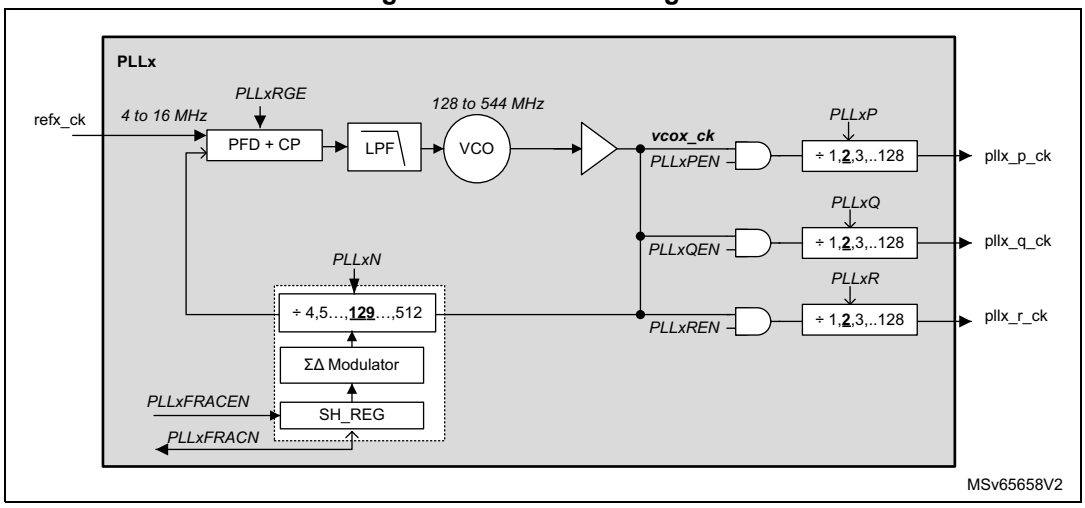
- Input frequency range: 4 to 16 MHz
- Capability to work either in integer or Fractional mode
- 13-bit sigma-delta (ΣΔ) modulator, allowing to fine-tune the VCO frequency by steps of 11 to 0.3 ppm
- The ΣΔ modulator can be updated on-the-fly, without generating frequency overshoots on PLLs outputs.
- Each PLL offers three outputs with post-dividers.

The PLLs are controlled via the registers RCC_PLLxDIVR, RCC_PLLxFRACR, RCC_PLLxCFGR and RCC_CR (x = 1,2,3).

The frequency of the reference clock provided to the PLLs (refx_ck) must range from 4 to 16 MHz. The user application must program properly the PLLxM (x = 1,2,3) dividers in the *RCC PLL1 configuration register (RCC_PLL1CFGR)*, *RCC PLL2 configuration register (RCC_PLL2CFGR)* and *RCC PLL3 configuration register (RCC_PLL3CFGR)* in order to match this condition. In addition, PLLxRGE must be set according to the reference input frequency to guarantee an optimal performance of the PLL.

To reduce the power consumption, it is recommended to configure the VCOx output to the lowest frequency.

Figure 36. PLL block diagram



PLLxN loop divider must be programmed to achieve the expected frequency at VCO output. In addition, the VCO output range must be respected.

The PLLx operates in integer mode when PLLxFRACEN = 0, and the PLL is enabled with PLLxON. The fractional mode can be enabled at any time by setting PLLxFRACN to the required value, and subsequently setting PLLxFRACEN from 0 to 1. The ∑Δ modulator is designed to minimize the jitter impact while allowing very small step frequency adjustments. To update the fractional value, first set PLLxFRACEN to 0 before updating the PLLxFRACN value, and subsequently set PLLxFRACEN to 1. The old PLLxFRACN value is kept used until the new value is activated by setting PLLxFRACEN from 0 to 1. PLLxFRACN must only be updated by software when PLLxFRACEN = 0.

The PLLs can be enabled by setting PLLxON to 1 in the *RCC clock control register (RCC_CR)*. The PLLxRDY bit indicates that the PLL is ready (meaning locked).

Note: Before enabling the PLLx, make sure that the reference frequency (refx_ck) provided to the PLL is stable. The following PLLx parameters cannot be changed once the PLLx is enabled: PLLxSRC, PLLxN, PLLxRGE, PLLxP, PLLxQ, and PLLxR.

The hardware prevents writing PLLxON to 0 if the PLL is currently used to deliver the system clock.

The following PLL parameters cannot be changed once the PLL is enabled: PLLxN, PLLxRGE, PLLxP, PLLxQ, and PLLxR.

To insure an optimal behavior of the PLL when one of the post-dividers (PLLxP, PLLxQ or PLLxR) is not used, the application must clear the enable bit (PLLxPEN, PLLxQEN, PLLxREN), and configure the corresponding post-dividers to their minimum value (PLLxR = 1, PLLxP = 0, or PLLxQ = 0).

If the above rules are not respected, the PLL output frequency is not guaranteed.

Output frequency computation

When the PLL operates in integer mode (SH_REG = 0), the VCO frequency (F_{VCO}) is given by the following formula (x = 1, 2, 3):

$$F_{VCOx} = F_{refx\_ck} \times PLLxN$$

When the PLLx operates in fractional mode, the PLLxFRACN value can be changed on-the-fly without disturbing the PLLx output.

This feature can be used either to generate a specific frequency from any crystal value with a good accuracy, or to fine-tune the frequency on-the-fly.

For each PLL, the VCO frequency is given by the following formula:

$$F_{VCOx} = F_{refx\_ck} \times \left(PLLxN + \frac{PLLxFRACN}{2^{(13)}} \right)$$

For both integer and fractional mode, the PLLx output frequency is given by the following formula:

$$F_{pllx\_y\_ck} = (F_{VCOx} / (PLLxy + 1)) \text{ with y = P, Q or R}$$

The PLLs are disabled by hardware:

- when the system enters Stop or Standby mode
- when an HSE failure occurs, when HSE or PLL (clocked by HSE) are used as system clock

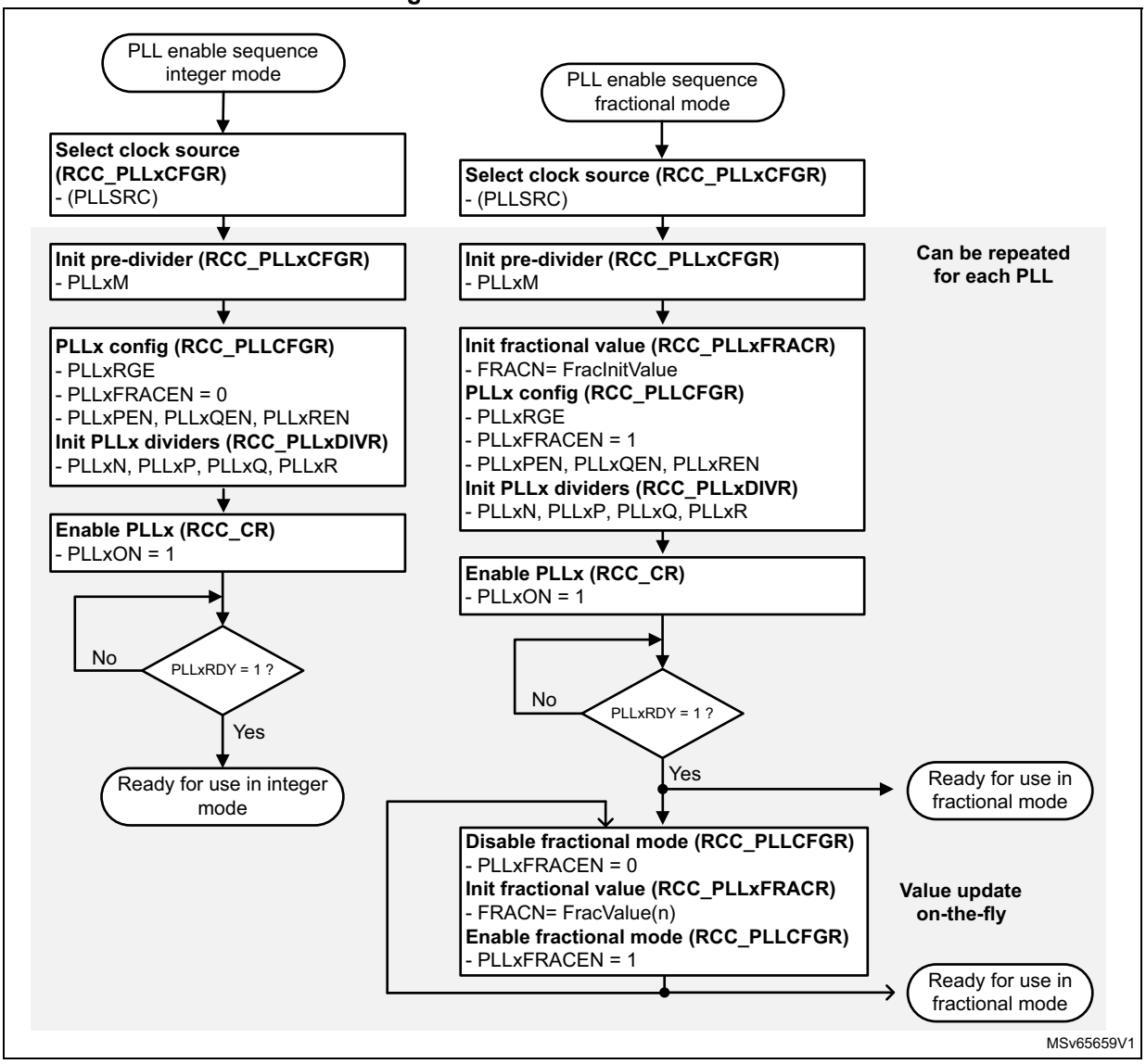
The fractional information used by the PLL is reset when disabling the PLL.

PLL initialization phase

Here below the recommended PLL initialization sequence in integer and fractional mode. The PLLx are supposed to be disabled at the start of the initialization sequence:

1. Initialize the PLL registers according to the required frequency.
   - For integer mode, set PLLxFRACEN to 0 in the *RCC PLL1 configuration register (RCC_PLL1CFGR)*, *RCC PLL2 configuration register (RCC_PLL2CFGR)* and *RCC PLL3 configuration register (RCC_PLL3CFGR)*.
   - For fractional mode, set PLLxFRACN to the required initial value (FracInitValue) and then set PLLxFRACEN to 1.
2. Once the PLLxON bit is set to 1, the application must wait until PLLxRDY is set to 1. As long as PLLxRDY = 0, PLLxFRACEN must not be altered.
3. Once the PLLxRDY bit is set to 1, the PLLx is ready to be used.
4. If the application intends to tune the PLLx frequency on-the-fly, then:
   a) PLLxFRACEN must be set to 0 to update the PLLxFRACN value while keeping the PLL running.
   b) A new value can be uploaded into PLLxFRACN (FracValue(n).
   c) PLLxFRACEN must be set to 1 to activate the new programed value in PLLxFRACN, that is taken into account by the PLL.

Figure 37. PLL initialization flow



Note: When the PLLxRDY goes to 1, it means that the difference between the PLLx output frequency and the target value is lower than ± 2 %.

11.4.7 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)*. If the LSE is used by other peripherals of functions than RTC, TAMP, and LSECSS, the LSESYSEN bit must be also be set in *RCC Backup domain control register (RCC_BDCR)* (refer to *Section : LSE when used by peripherals other than RTC/TAMP, and RCC functions*).

The crystal oscillator driving strength is configured using the LSEDRV[1:0] bits, according to crystal specification, 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 must be programmed before enabling the LSE.

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. This mode is selected by setting the LSEBYP and LSEON bits in the *RCC Backup domain control register (RCC_BDCR)*. The external clock signal (square, sinus or triangle) with ~50 % duty cycle, must drive the OSC32_IN pin while the OSC32_OUT pin can be used as GPIO (see *Figure 34*).

LSE when used by peripherals other than RTC/TAMP, and RCC functions

By default, when enabled, the LSE is sent only to RTC and TAMP (assuming that RTCSEL = 01).

It the LSE is needed for other peripherals (such as peripheral clock or trigger source), or if the LSE is used by a RCC function (such as LSCO, MCO, MSI PLL mode), the sequence below must be done:

1. Set the LSEON in RCC_BDCR and wait the LSE clock ready bit (LSERD = 1 in *RCC Backup domain control register (RCC_BDCR)*.
2. Set the LSESYSEN bit in RCC_BDCR.
3. Wait the LSESYS clock is ready (LSESYSRDY = 1 in RCC_BDCR).

The LSE consumption is increased when LSESYSEN = 1.

11.4.8 LSI clock

The LSI RC acts as a low-power clock source that can be kept running in Stop and Standby modes for the independent watchdog (IWDG) and RTC. The clock frequency is either 32 kHz or 250 Hz depending on the LSIPREDIV bit in *RCC Backup domain control register (RCC_BDCR)*. Setting LSIPREDIV allows a lower consumption (refer to the electrical characteristics section of the datasheet for more details).

When the IWDG is enabled or when the RTC or TAMP is clocked by the LSI, the LSIPREDIV cannot be changed anymore.

The LSI RC can be switched on and off using the LSION bit in the *RCC Backup domain control register (RCC_BDCR)*.

The LSIRDY flag in the *RCC Backup domain control register (RCC_BDCR)* 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)*.

11.4.9 System clock (SYSCLK) selection

Four different clock sources can be used to drive the system clock (SYSCLK):

- MSIS oscillator
- HSI16 oscillator
- HSE oscillator
- PLL

The system clock maximum frequency is 160 MHz. After a system reset, the MSIS 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 clock control register (RCC_CR)* indicate which clocks are ready and which clock is currently used as a system clock.

The table below gives the different bus frequencies depending on the product voltage range.

Table 103. Bus maximum frequency

Product voltage range	AHB1/AHB2/AHB3/APB1/APB2/APB3
Range 1	160 MHz
Range 2	110 MHz
Range 3	55 MHz
Range 4	25 MHz

11.4.10 Clock source frequency versus voltage scaling

The table below gives the different clock source frequencies depending on the product voltage range.

Table 104. Clock source maximum frequency

Product voltage range	Clock frequency					
	MSIS MSIK	HSI16	HSI48	SHSI	HSE	PLL outputs (VCO min to max)
Range 1	All ranges	Allowed	Allowed	Allowed	50 MHz	208 MHz(1) (128 to 544 MHz)
Range 2	All ranges	Allowed	Allowed	Allowed	50 MHz	110 MHz (128 to 544 MHz)
Range 3	All ranges	Allowed	Allowed	Allowed	50 MHz	55 MHz (128 to 330 MHz)
Range 4	Up to 24 MHz range	Allowed	Allowed (divided by 2)	Allowed (divided by 2)	25 MHz	Not allowed

1. Max frequency can vary depending on peripherals connected to PLL outputs.

11.4.11 Clock security system (CSS)

The CSS can be activated by software. In this case, the clock detector is enabled after the HSE oscillator wakeup time, 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 some timers break input and an interrupt is generated to inform the software about the failure (clock security system interrupt CSSI). This allows 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 MSIS or the HSI16 oscillator depending on the STOPWUCK configuration in the *RCC clock control register (RCC_CR)*, 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.

11.4.12 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 is enabled (LSEON enabled) and ready (LSERDY set by hardware), and after the RTC clock has been selected by RTCSEL.

The CSS on LSE is working in all modes including VBAT. It works also under system reset (excluding power-on reset).

The clock security system on LSE detects when the LSE disappears or in case of over frequency. In addition, the glitches on LSE can be filtered by setting LSEGFON. LSEGFON must be written when the LSE is disabled (LSEON = 0 and LSERDY = 0).

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 detection event is connected to the internal tamper 3 of the TAMP peripheral:

- On STM32U575/585 rev.X devices, the internal tamper 3 must be enabled (ITAMP3E = 1 in TAMP_CR1 register), and the associated interrupt enabled (ITAMP3IE in TAMP_IER) in order to wake up from the low-power modes. This erases also the TAMP backup registers and backup SRAM, unless ITAMP3NOER = 1 in TAMP_CR3 (see *Section 55: Tamper and backup registers (TAMP)* for more details).
- On all other STM32U575/585 revisions, the CSS on LSE detection event is also connected to an EXTI line: this is used to generate an event or interrupt supporting wakeup from Stop 0, Stop 1, or Stop 2 mode, without requiring to enable tamper detection (see *Table 107: Interrupt sources and control* and *Table 156: EXTI line connections*).

In case of CSS on LSE detection event (LSECSSD = 1 in the RCC_BDCR), 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.

Refer to datasheet for CSS on LSE electrical characteristics.

11.4.13 ADC and DAC clocks

The ADC and DAC kernel clock source is selected thanks to ADCDACSEL[2:0] in the *RCC peripherals independent clock configuration register 3 (RCC_CCIPR3)*. The ADC clock ratio must be around 50 %. For this reason, the AHB clock, when selected as ADC clock, must not be divided with HPRE prescaler. If pll2_r_ck is selected as ADC clock, the PLL2R division factor must be even (division by 2 or 4 for example).

If the application requires that the ADC or DAC is precisely triggered by a TIMx timer without any uncertainty, the HCLK must be selected as ADC and DAC kernel clock source. The other clock sources are asynchronous to TIMx timers therefore an uncertainty of the trigger instant is added by the resynchronization between the two clock domains. The LPTIMx timers are also asynchronous.

The DAC requires an additional low-power clock (LSI or LSE) to operate in sample and hold mode, available in Stop mode. This clock is selected with DAC1SEL in the RCC_CCIPR3.

11.4.14 RTC and TAMP clock

The RTCCLK clock source is used by RTC and TAMP, and 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 TAMP does not require any kernel clock if only the backup registers are used, with tampers in edge detection mode. All other tamper detection modes require a kernel clock (refer to *Section 55: Tamper and backup registers (TAMP)* for more details).

The LSE and the LSI clocks are in the Backup domain, whereas the HSE clock is not. Consequently:

- If LSE or LSI is selected as RTC and TAMP clock, these peripherals continue to work even if the V_{DD} supply is switched off, provided the V_{BAT} supply is maintained.
- If the HSE clock divided by a prescaler is used as the RTC or TAMP 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 core domain). Depending on the TAMP configuration, this one can remain functional if used in a mode that does not need any kernel clock.

When the RTC and TAMP clock is LSE or LSI, the RTC remains clocked and functional under system reset.

If the LSE is needed only for the RTC or TAMP, LSESYSEN must be kept at reset value to get the lowest consumption.

11.4.15 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.

11.4.16 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.

11.4.17 OCTOSPI clock

The OCTOSPIx kernel clock, selected by OCTOSPIxSEL[1:0], can be up to 200 MHz when pll1_q_ck is used.

11.4.18 Clock-out capability

- **MCO**

  The microcontroller clock output (MCO) capability allows the clock to be output onto the external MCO pin. One of the following clock signals can be selected as MCO clock.

  – LSI
  – LSE
  – SYSCLK
  – HSI16
  – HSI48
  – HSE
  – PLLCLK
  – MSIS
  – MSIK

  The selection is controlled by the MCOSEL[3:0] bits in the *RCC clock control register (RCC_CR)*. The selected clock can be divided with the MCOPRE[2:0] field in the *RCC clock control register (RCC_CR)*.

- **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 all Stop modes, Standby and Shutdown modes. This output is not available in VBAT mode. 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.

11.4.19 Internal/external clock measurement with TIM15/TIM16/TIM17

The frequency of all on-board clock sources can be indirectly measured by mean of the TIM15, TIM16 or TIM17 channel 1 input capture and LPTIM1 or LPTIM2 channel 2 input capture.

HSI16 and MSI calibration using LSE

The primary purpose of connecting the LSE to the channel 1 input capture of TIM15, TIM16 and TIM17, and to the channel 2 input capture of LPTIM2, is to be able to precisely measure the HSI16 and MSI system clocks (for this, either HSI16 or MSIS must be used as system clock source). The number of HSI16 (MSIS 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), the internal clock frequency can be determined with the same resolution, and the source can be trimmed to compensate the manufacturing, process, temperature and/or voltage related frequency deviations.

The four oscillators of MSI and HSI16 oscillator 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.

Note: When the LSE is available, the MSI can be automatically trimmed by LSE using the PLL-mode.

HSI16 and MSI calibration using HSE

If the HSE is available, it can be used as system clock and the timer input capture must be connected either to MSI (divided by 1024 or by 4) or to HSI/256. The timers 16 and 17 channel 1 input capture, as well and the LPTIM2 input capture 2, are connected to the divided oscillator only when TIMICSEL[2:0] is different from 0xx in the *RCC peripherals independent clock configuration register 1 (RCC_CCIPR1)*.

Considering that the timers counter is 16-bit, and that the ratio between HSE and the input capture signal must be the highest possible, a division by 1024 must be selected when the MSIRC0, MSIRC1 or MSIRC2 is measured, and a division by 4 when the MSIRC4 is measured.

LSI calibration

The calibration of the LSI follows the same principle, but changing the reference clock. The LSI clock must be connected to the channel 1 input capture of the TIM16 or TIM17, or to the channel 2 input capture of the LPTIM1. 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.

11.4.20 Peripherals clock gating and autonomous mode

Peripherals clock gating in Run mode

Each peripheral clock can be enabled by the corresponding EN bit in the RCC_AHBxENR and RCC_APBxENR 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 the clock is active after 2 cycles of the peripheral bus clock.

Caution: Just after enabling the clock for a peripheral, the software must wait for these 2 clock cycles before accessing the peripheral registers.

Peripherals clock gating in Sleep and Stop modes

When a peripheral is enabled, its clock can be automatically gated off when the device is in Sleep mode, by clearing the peripheral SMEN bit in the RCC_AHBxSMENR and RCC_APBxSMENR registers. Both EN and SMEN bit of the peripheral must be set to keep the clock on in Sleep mode.

The SMEN bit of the peripheral is also used to allow peripheral clocking in Stop 0 and Stop 1 modes, upon peripheral request.

When the clock is requested by a peripheral, this clock is distributed to all enabled peripherals. Therefore the SMEN bit must be cleared before entering Stop mode, if the peripheral is not used in Stop mode.

Caution: The SMEN bit of the peripheral must be set to allow the generation of an interrupt capable to wake up the device from Stop mode. This is not necessary when the peripheral wakeup interrupt is generated though the EXTI.

Peripherals clock gating and autonomous mode in Stop 0/1/2 modes

Some peripherals support autonomous mode (refer to *Table 105: Autonomous peripherals*). These peripherals are able to generate a kernel clock request and a AHB/APB bus clock request when they need, in order to operate and update their status register even in Stop mode. Depending on the peripheral configuration, either a DMA request or an interrupt can be associated to the peripheral event.

Upon an AHB or APB bus clock request from an autonomous peripheral, either MSI or HSI16 oscillator is woken up, depending on the oscillator selected by STOPWUCK in the *RCC clock configuration register 1 (RCC_CFGR1)*.

If the autonomous peripheral is configured with DMA requests enabled, a data transfer is performed thanks to the AHB/APB clock. The bus clocks as well as the oscillator (HSI16 or MSI) are automatically switched off as soon as the transfer is finished, if no other peripheral requests it. The device automatically goes back in Stop mode.

If the autonomous peripheral is configured with interrupt enabled, the interrupt wakes up the device into Run mode.

The autonomous peripherals mapped on AHB3 or APB3 belong to the SmartRun domain and are autonomous in Stop 0, Stop 1 and Stop 2 with the LPDMA1 and SRAM4.

The autonomous peripherals mapped on AHB1, AHB2, APB1 and APB2, belong to the CPU domain and are autonomous in Stop 0 and Stop 1 only with the GPDMA1 and SRAM1, SRAM2, SRAM3 or SRAM4.

The table below shows the list of peripherals with autonomous mode capability.

Table 105. Autonomous peripherals

Domain	Peripheral	Autonomous in Stop 0, 1 modes	Autonomous in Stop 2 mode	Associated DMA	Associated SRAM
CPU domain (CD)	U(S)ARTx (x = 1 to 5)	Yes(1)	No	GPDMA1	SRAM1 SRAM2 SRAM3 SRAM4(2)
	SPIx (x = 1,2)				
	I2Cx (x = 1,2,4)				
	LPTIM2				
	MDF1				
	GPDMA1			-	
SmartRun domain (SRD)	LPUART1	Yes (3)	Yes(3)	LPDMA1	SRAM4
	SPI3				
	I2C3				
	LPTIMx (x = 1,3,4)				
	ADF1				
	DAC1				
	ADC4				
	LPDMA1			-	

1. Enabled if both xxEN and xxSMEN bits of the peripheral are set (xx = instance name)
2. SRAM4 belongs to SmartRun domain (SRD) but can be addressed by GPDMA 1 in Stop 0 and Stop 1 modes.
3. Enabled if all xxEN, xxSMEN and xxAMEN bits of the peripheral are set (xx = instance name)

For peripherals in CPU domain, the autonomous mode is enabled in Stop 0 and Stop 1 modes if both xxEN and xxSMEN bits of the peripheral are set.

For peripherals in SmartRun domain, the autonomous mode is enabled in Stop 0, Stop 1 and Stop 2 modes if both xxEN and xxSMEN bits of the peripheral are set, plus the xxAMEN bit of the peripheral in the *RCC SmartRun domain peripheral autonomous mode register (RCC_SRDAMR)*.

If an autonomous peripheral requests its kernel clock in Stop 0, Stop 1 or Stop 2 mode, the internal oscillator (HSI16 or MSI) is woken up if it was off, and the kernel clock is propagated only to the peripheral requesting it. When the peripheral releases its kernel clock request, the HSI16 or MSI is switched off if no other peripheral requests it.

If an autonomous peripheral belonging to CPU domain requests its bus clock (AHB1, AHB2, APB1 or APB2 clock) in Stop 0 or Stop 1 mode, the internal oscillator (HSI16 or MSI depending on STOPWUCK value in RCC_CFGR1) is woken up if it was off, and the system clock is propagated to all peripherals configured with both xxEN and xxSMEN bits set.

If an autonomous peripheral belonging to SmartRun domain requests its bus clock (AHB3 or APB3 clock) in Stop 0, Stop 1 or Stop 2 mode, the internal oscillator (HSI16 or MSI depending on STOPWUCK value in RCC_CFGR1) is woken up if it was off, and the HCLK3/PCLK3 clocks are propagated to all peripherals of the SmartRun domain configured with xxEN, xxSMEN and xxAMEN set.

Caution: The AMEN bit of the peripheral must be set to allow the generation of an interrupt capable to wake up the device from Stop mode. This is not necessary when the peripheral wakeup interrupt is generated though the EXTI.

Note: MSIK or HSI16 can be forced to remain ON in Stop 0, Stop 1 or Stop 2 mode, by configuring MSIKERON or HSIKERON is the RCC_CR. In this case, the oscillator is propagated only to the kernel clock of the enabled autonomous peripherals with this oscillator selected as kernel clock. This allows the peripheral baudrates or conversion rates increase, as there is no need to wait for the oscillator wakeup time when the peripheral requests its kernel clock.

The LSE or LSI selected as peripheral kernel clock remains always ON in Stop modes.

The AHB3 and APB3 clocks can be forced to remain ON by setting the SRDRUN bit in the PWR_CR2 register. This allows the LPDMA1 latency to be improved as there is no need to wait for the oscillator wakeup time when the peripheral requests its bus clock.

11.5 RCC security and privilege functional description

11.5.1 RCC TrustZone security protection modes

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 *RCC secure 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, LSCO, HSI48 configuration and status bits
- PLL1, PLL2, PLL3, AHB and APB prescalers configuration and status bits
- system clock (SYSCLK) and ICLK source clock selection and status bits
- MCO clock output configuration and STOPWUCK and STOPKERWUCK bit
- Remove reset flag RMVF configuration

If SPRIV is set in the *RCC privilege configuration register (RCC_PRIVCFGR)*, the RCC_SECCFGR register can be written only by secure and privileged access. If SPRIV is cleared in RCC_PRIVCFGR, RCC_SECCFGR can be written only by secure access, privileged or unprivileged.

RCC_SECCFGR can be read by secure, non-secure, privileged and unprivileged access.

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_SRDAMEN, RCC_CCIPR1, RCC_CCIPR2, RCC_CCIPR3 and RCC_BDCR registers.

The SHSI configuration and status bits are secured wen the SAES is configured as secure.

BDRST in RCC_BDCR is secure when at least one function is secure in RTC or TAMP.

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 GTZC TZSC secure configuration registers.
- For TrustZone-aware peripherals, a security feature of this peripheral is enabled through its dedicated bits.

Table 106 summarizes 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 returns 0 and write accesses are ignored. No illegal access event is 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 RCC_SECCFGR is ignored and generates an illegal access event. An illegal access interrupt is generated if the RCC illegal access interrupt is enabled in the GTZC TZIC registers. RCC_SECCFGR can be read by secure or non-secure access.

When the TrustZone security is disabled (TZEN = 0 in FLASH_OPTR register), all registers are non-secure. RCC_SECCFGR write accesses are ignored.

Table 106. RCC security configuration summary

Configuration bit in RCC_SECCFGR	Secured bits	Corresponding register
HSISEC	HSION, HSIKERON, HSIRDY	RCC_CR
	HSICAL[7:0], HSITRIM[6:0]	RCC_ICSCR3
	HSIRDYIE	RCC_CIER
	HSIRDYIF	RCC_CIFR
	HSIRDYC	RCC_CICR
HSESEC	HSEON, HSERDY, HSEBYP, HSECSSON, HSEEXT	RCC_CR
	HSERDYIE, HSECSSF	RCC_CIER
	HSERDYIF, HSECSSF	RCC_CIFR
	HSERDYC, HSECSSC	RCC_CICR

Table 106. RCC security configuration summary (continued)

Configuration bit in RCC_SECCFGR	Secured bits	Corresponding register
MSISEC	MSISON, MSIKERON, MSISRDY, MSIPLLEN, MSIKON, MSIKRDY, MSIPLLSEL, MSIPLLFAST	RCC_CR
	MISIRANGE[3:0], MISIKRANGE[3:0], MSIRGSEL, MSIBIAS, MSICAL0[4:0], MSICAL1[4:0], MSICAL2[4:0], MSICAL3[4:0]	RCC_ICSCR1
	MSITRIM0[4:0], MSITRIM1[4:0], MSITRIM2[4:0], MSITRIM3[4:0]	RCC_ICSCR2
	MSISRDYIE, MSIKRDYIE	RCC_CIER
	MSISRDYIF, MSIKRDYIF	RCC_CIFR
	MSISRDYIC, MSIKRDYIC	RCC_CICR
	MSISSRANGE[3:0], MSIKSRANGE[3:0]	RCC_CSR
LSISEC	LSION, LSIRDY, LSIPREDIV, LSCOSEL, LSCOEN	RCC_BDCR
	LSIRDYIE	RCC_CIER
	LSIRDYIF	RCC_CIFR
	LSIRDYC	RCC_CICR
LSESEC	LSECSSON, LSECSSD, LSEDRV[1:0], LSEBYP, LSERDY, LSEON, LSEGFON, LSESYSRDY, LSESYSEN, LSCOSEL, LSCOEN	RCC_BDCR
	LSERDYIE	RCC_CIER
	LSERDYF	RCC_CIFR
	LSERDYC	RCC_CICR
SYSCLKSEC	SW[1:0], SWS[1:0], STOPWUCK, STOPKERWUCK, MCOSEL[3:0], MCOPRE[2:0]	RCC_CFGR1
	SYSTICKSEL[1:0]	RCC_CCIPR1
	VOS[1:0]	PWR_VOSR
PRESCSEC	HPRE[3:0], PPRE1[2:0], PPRE2[2:0]	RCC_CFGR2
	PPRE3[2:0]	RCC_CFGR3
PLL1SEC	PLL1SRC[1:0], PLL1RGE[1:0], PLL1FRACEN, PLL1M[3:0], PLL1MBOOST[3:0], PLL1PEN, PLL1QEN, PLL1REN	RCC_PLL1CFGR
	PLL1N[8:0], PLL1P[6:0], PLL1Q[6:0], PLL1R[6:0]	RCC_PLL1DIVR
	PLL1FRACN[12:0]	RCC_PLL1FRACR
	PLL1RDY, PLL1ON	RCC_CR
	PLL1RDYIE	RCC_CIER
	PLL1RDYF	RCC_CIFR
	PLL1RDYC	RCC_CICR

Table 106. RCC security configuration summary (continued)

Configuration bit in RCC_SECCFGR	Secured bits	Corresponding register
PLL2SEC	PLL2SRC[1:0], PLL2RGE[1:0], PLL2FRACEN, PLL2M[3:0], PLL2PEN, PLL2QEN, PLL2REN	RCC_PLL2CFGR
	PLL2N[8:0], PLL2P[6:0], PLL2Q[6:0], PLL2R[6:0]	RCC_PLL2DIVR
	PLL2FRACN[12:0]	RCC_PLL2FRACR
	PLL2RDY, PLL2ON	RCC_CR
	PLL2RDYIE	RCC_CIER
	PLL2RDYF	RCC_CIFR
	PLL2RDYC	RCC_CICR
PLL3SEC	PLL3SRC[1:0], PLL3RGE[1:0], PLL3FRACEN, PLL3M[3:0], PLL3PEN, PLL3QEN, PLL3REN	RCC_PLL3CFGR
	PLL3N[8:0], PLL3P[6:0], PLL3Q[6:0], PLL3R[6:0]	RCC_PLL3DIVR
	PLL3FRACN[12:0]	RCC_PLL3FRACR
	PLL3RDY, PLL3ON	RCC_CR
	PLL3RDYIE	RCC_CIER
	PLL3RDYF	RCC_CIFR
	PLL3RDYC	RCC_CICR
HSI48SEC(1)	HSI48ON, HSI48RDY	RCC_CR
	HSI48CAL[8:0]	RCC_CRRCR
	HSI48RDYE	RCC_CIER
	HSI48RDYF	RCC_CIFR
	HSI48RDYC	RCC_CICR
ICLKSEL	ICLKSEL[1:0]	RCC_CCIPR1
RMVFSEC	RMVF	RCC_CSR

1. TRIM field of the HSI48 is located in CRS peripheral. Be sure to secure it using CRSSEC bit in GTZC1 TZSC secure configuration register 1.

11.5.2 RCC privilege protection modes

By default, after reset, all RCC registers can be read or written with both privileged and unprivileged access except *RCC privilege configuration register (RCC_PRIVCFGR)* that can be written with privileged access only. RCC_PRIVCFGR can be read by secure and non secure, privileged and unprivileged access.

The SPRIV bit in RCC_PRIVCFGR can be written with secure privileged access only. This bit configures the privileged access of all RCC secure functions (as defined by *RCC secure configuration register (RCC_SECCFGR)* or by the GTZC for securable peripherals, or by the peripheral itself in case of TrustZone-aware peripherals).

When the SPRIV bit is set in RCC_PRIVCFGR:

- Writing the RCC secure bits is possible only with privileged access, including RCC_SECCFGR.
- The RCC secure bits can be read only with privileged access except RCC_SECCFGR and RCC_PRIVCFGR that can be read by privileged or unprivileged access.
- An unprivileged access to a privileged RCC bit or register is discarded: the bits are read as zero and the write to these bits is ignored (RAZ/WI).

The NSPRIV bit in RCC_PRIVCFGR can be written with privileged access only, secure or non-secure. This bit configures the privileged access of all RCC non-secure functions (as defined by RCC_SECCFGR, or by the GTZC for securable peripherals, or by the peripheral itself in case of TrustZone-aware peripherals).

When the NSPRIV bit is set in RCC_PRIVCFGR:

- Writing the RCC non-secure bits is possible only with privileged access.
- The RCC non-secure bits can be read only with privileged access except RCC_PRIVCFGR that can be read by privileged or unprivileged access.
- An unprivileged access to a privileged RCC bit or register is discarded: the bits are read as zero and the write to these bits is ignored (RAZ/WI).

11.6 RCC low-power modes

- AHB and APB peripheral clocks, including DMA clock, can be disabled by software.
- Sleep mode stops the CPU clock. The memory interface clocks (Flash memory, caches and all SRAM interfaces) can be stopped by software during Sleep mode. The AHB to APB bridge clocks are disabled by hardware during Sleep mode when all the clocks of the peripherals connected to them are disabled.
- Stop modes (Stop 0, Stop 1, Stop 2, Stop 3) stop all the clocks in the core domain and disable the PLLs, HSI16, HSI48, SHSI, MSI and HSE oscillators. However, HSI16 or MSI can be switched ON if the peripheral requests it for autonomous mode purpose, or to generate a wakeup interrupt (see *Section 11.4.20: Peripherals clock gating and autonomous mode* for more details). LSI and LSE remain active in Stop modes.
- Standby and Shutdown modes stop all the clocks in the core domain and disable the PLLs, HSI16, HSI48, SHSI, MSI and HSE oscillators.

The CPU Deepsleep mode can be overridden for debugging by setting the DBG_STOP or DBG_STANDBY bit in the DBGMCU_CR register.

When exiting Stop modes (Stop 0, Stop 1, Stop 2 or Stop 3), the system clock is either MSIS or HSI16, depending on the software configuration of STOPWUCK in the *RCC clock configuration register 1 (RCC_CFGR1)*. The frequency (range and user trim) of the MSIS and MSIK oscillators is the one configured before entering Stop mode, except if above 24 MHz. In this case, the MSIS or MSIK range is the 24 MHz range. The user trim of HSI16 is kept. If MSI is in PLL-mode before entering Stop mode with MSIPLLFAST = 0, the PLL-mode stabilization time must be waited for after wakeup even if the LSE was kept ON during the Stop mode. The PLL-mode accuracy is kept after wakeup from Stop 0, Stop 1 or Stop 2 mode without stabilization time if MSIPLLFAST = 1. The MSIPLLFAST bit has no effect when exiting Stop 3 mode.

The other internal oscillator can be automatically woken up in addition to the one used by the system clock, in order to avoid waiting for the other oscillator wakeup time when the device is back in Run mode. This is done thanks to STOPKERWUCK in RCC_CFGR1.

When leaving the Standby and Shutdown modes, the system clock is MSIS. The MSIS and MSIK frequency at wakeup from Standby mode is configured with MSISSRANGE and MSIKSRANGE in RCC_CSR, from 1 to 4 MHz. The MSI frequency at wakeup from Shutdown mode is 4 MHz. The user trim is lost.

If a Flash memory programming operation is ongoing, Stop, Standby or Shutdown mode entry is delayed until the Flash memory interface access is finished. If an access to the APB domain is ongoing, Stop, Standby or Shutdown mode entry is delayed until the APB access is finished. If an autonomous peripheral generates a system clock request, Stop, Standby or Shutdown mode entry is delayed until the system clock request is released.

11.7 RCC interrupts

The table below summarizes the interrupt sources and the way to control them.

Table 107. Interrupt sources and control

Interrupt vector	Interrupt event flag	Description	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop, Standby, Shutdown modes
RCC	LSIRDYF	LSI ready	LSIRDYIE and LSISEC = 0	Set LSIRDYC to 1	Yes	No
	LSERDYF	LSE ready	LSERDYIE and LSESEC = 0	Set LSERDYC to 1	Yes	No
	HSIDRYF	HSI ready	HSIDRYIE and HSISEC = 0	Set HSIRDYC to 1	Yes	No
	HSERDYF	HSE ready	HSERDYIE and HSESEC = 0	Set HSERDYC to 1	Yes	No
	MSISRDYF	MSIS ready	MSISRDYIE and MSISEC = 0	Set MSISRDYC to 1	Yes	No
	MSIKRDYF	MSIK ready	MSIKRDYIE and MSISEC = 0	Set MSIKRDYC to 1	Yes	No
	SHSIRDYF	SHSI ready	SHSIRDYIE and SAESSEC = 0 (in GTZC)	Set SHSIRDYC to 1	Yes	No
	HSI48RDYF	HSI48 ready	HSI48RDYIE and HSI48SEC = 0	Set HSI48RDYC to 1	Yes	No
	PLL1RDYF	PLL1 ready	PLL1RDYIE and PLL1SEC = 0	Set PLL1RDYC to 1	Yes	No
	PLL2RDYF	PLL2 ready	PLL2RDYIE and PLL2SEC = 0	Set PLL2RDYC to 1	Yes	No
	PLL3RDYF	PLL3 ready	PLL3RDYIE and PLL3SEC = 0	Set PLL3DYC to 1	Yes	No
RCC_S(1)	LSIRDYF	LSI ready	LSIRDYIE and LSISEC = 1	Set LSIRDYC to 1	Yes	No
	LSERDYF	LSE ready	LSERDYIE and LSESEC = 1	Set LSERDYC to 1	Yes	No
	HSIDRYF	HSI ready	HSIDRYIE and HSISEC = 1	Set HSIRDYC to 1	Yes	No
	HSERDYF	HSE ready	HSERDYIE and HSESEC = 1	Set HSERDYC to 1	Yes	No
	MSISRDYF	MSIS ready	MSISRDYIE and MSISEC = 1	Set MSISRDYC to 1	Yes	No
	MSIKRDYF	MSIK ready	MSIKRDYIE and MSISEC = 1	Set MSIKRDYC to 1	Yes	No

Table 107. Interrupt sources and control (continued)

Interrupt vector	Interrupt event flag	Description	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop, Standby, Shutdown modes
RCC_S(1)	SHSIRDYF	SHSI ready	SHSIRDYIE and SAESSEC(2) = 1	Set SHSIRDYC to 1	Yes	No
	HSI48RDYF	HSI48 ready	HSI48RDYIE and HSI48SEC = 1	Set HSI48RDYC to 1	Yes	No
	PLL1RDYF	PLL1 ready	PLL1RDYIE and PLL1SEC = 1	Set PLL1RDYC to 1	Yes	No
	PLL2RDYF	PLL2 ready	PLL2RDYIE and PLL2SEC = 1	Set PLL2RDYC to 1	Yes	No
	PLL3RDYF	PLL3 ready	PLL3RDYIE and PLL3SEC = 1	Set PLL3RDYC to 1	Yes	No
TAMP	ITAMP3F(3)	LSE CSS failure	LSECSSON and ITAMP3E(3) and ITAMP3IE(3)	Set CITAMP3F(3) to 1	Yes	Yes
NMI	HSECSSF	HSE CSS failure	-(4)	Set HSECSSC to 1	Yes	No
LSECSS(5)	Through EXTI	LSE CSS failure	Through EXTI	Through EXTI	Yes	Yes(6)/No
MSI_PLL_UNLOCK(5)	Through EXTI	MSI PLL-mode unlock(7)	Through EXTI	Through EXTI	Yes	Yes(6)/No

1. The RCC secure interrupt vector is used only when TrustZone is enabled.
2. The SAESSEC bit is in the GTZC peripheral.
3. The LSE CSS failure event (LSECSSD) is connected to TAMP internal tamper 3. In order to get the interrupt associated to this event, the internal tamper 3 must be enabled, and the internal tamper 3 interrupt must be enabled. The ITAMP3F, ITAMP3E, ITAMP3IE and CITAMP3F bits are in the TAMP peripheral. Consequently, the LSE CSS tamper interrupt erases or blocks the device secrets as described in *Table 528: TAMP interconnection*.
4. It is not possible to mask this interrupt when the security system feature is enabled (HSECSSON = 1).
5. Not available in STM32U575/585 rev. X. Available in all other STM32U575/585 revisions.
6. This interrupt can wake up from Stop 0, Stop 1, and Stop 2 modes only.
7. This interrupt indicates that the MSI has left the PLL_mode, due to LSE missing pulses. As a consequence, the MSI frequency accuracy is degraded.

11.8 RCC registers

11.8.1 RCC clock control register (RCC_CR)

Address offset: 0x000

Reset value: 0x0000 0035

HSEBYP and HSEEXT are cleared upon power-on reset. They are 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
Res.	Res.	PLL3 RDY	PLL3ON	PLL2 RDY	PLL2ON	PLL1 RDY	PLL1ON	Res.	Res.	Res.	HSEEXT	CSSON	HSEBYP	HSE RDY	HSEON
		r	rw	r	rw	r	rw				rw	rs	rw	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHSIRDY	SHSION	HSI48RDY	HSI48ON	Res.	HSIRDY	HSIKERON	HSION	MSIPLLFAST	MSIPLLSEL	MSIKRDY	MSIKON	MSIPLLEN	MSISRDY	MSIKERON	MSISON
r	rw	r	rw		r	rw	rw	rw	rw	r	rw	rw	r	rw	rw

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

Bit 29 **PLL3RDY**: PLL3 clock ready flag
Set by hardware to indicate that the PLL3 is locked.
0: PLL3 unlocked
1: PLL3 locked

Bit 28 **PLL3ON**: PLL3 enable
Set and cleared by software to enable PLL3.
Cleared by hardware when entering Stop, Standby or Shutdown mode.
0: PLL3 OFF
1: PLL3 ON

Bit 27 **PLL2RDY**: PLL2 clock ready flag
Set by hardware to indicate that the PLL2 is locked.
0: PLL2 unlocked
1: PLL2 locked

Bit 26 **PLL2ON**: PLL2 enable
Set and cleared by software to enable PLL2.
Cleared by hardware when entering Stop, Standby or Shutdown mode.
0: PLL2 OFF
1: PLL2 ON

Bit 25 **PLL1RDY**: PLL1 clock ready flag
Set by hardware to indicate that the PLL1 is locked.
0: PLL1 unlocked
1: PLL1 locked

Bit 24 **PLL1ON**: PLL1 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 PLL1 clock is used as the system clock.

0: PLL1 OFF

1: PLL1 ON

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

Bit 20 **HSEEXT**: HSE external clock bypass mode

Set and reset by software to select the external clock mode in bypass mode. External clock mode must be configured with HSEON bit to be used by the device. This bit can be written only if the HSE oscillator is disabled. This bit is active only if the HSE bypass mode is enabled.

0: external HSE clock analog mode

1: external HSE clock digital mode (through I/O Schmitt trigger)

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 six 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

Bit 15 **SHSIRDY**: SHSI clock ready flag

Set by hardware to indicate that the SHSI oscillator is stable. This bit is set only when SHSI is enabled by software by setting SHSION.

0: SHSI oscillator not ready

1: SHSI oscillator ready

Note: Once the SHSION bit is cleared, SHSIRDY goes low after six SHSI clock cycles.

Bit 14 **SHSION**: SHSI clock enable
Set and cleared by software.
Cleared by hardware to stop the SHSI when entering in Stop, Standby or Shutdown modes.
0: SHSI oscillator OFF
1: SHSI oscillator ON

Bit 13 **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 12 **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

Bit 11 Reserved, must be kept at reset value.

Bit 10 **HSIRDY**: HSI16 clock ready flag
Set by hardware to indicate that HSI16 oscillator is stable. This bit is set only when HSI16 is enabled by software by setting HSION.
0: HSI16 oscillator not ready
1: HSI16 oscillator ready
Note: Once the HSION bit is cleared, HSIRDY goes low after six HSI16 clock cycles.

Bit 9 **HSIKERON**: HSI16 enable for some peripheral kernels
Set and cleared by software to force HSI16 ON even in Stop modes. Keeping the HSI16 ON in Stop mode allows the communication speed not to be reduced by the HSI16 startup time. This bit has no effect on HSION value.
Refer to *Peripherals clock gating and autonomous mode* for more details.
The HSIKERON must be configured at 0 before entering Stop 3 mode.
0: No effect on HSI16 oscillator
1: HSI16 oscillator 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 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

Bit 7 **MSIPLLFAST:** MSI PLL mode fast startup
Set and reset by software to enable/disable the fast PLL mode start-up of the MSI clock source. This bit is used only if PLL mode is selected (MSIPLLEN = 1).

Caution: The fast start-up feature is not active the first time the PLL mode is selected. The fast start-up is active when the MSI in PLL mode returns from switch off.

0: MSI PLL normal start-up
1: MSI PLL fast start-up

Bit 6 **MSIPLLSEL:** MSI clock with PLL mode selection

Set and cleared by software to select which MSI output clock uses the PLL mode. This bit can be written only when the MSI PLL mode is disabled (MSIPLLEN = 0).

0: PLL mode applied to MSIK (MSI kernel) clock output

1: PLL mode applied to MSIS (MSI system) clock output

Note: If the MSI kernel clock output uses the same oscillator source than the MSI system clock output, then the PLL mode is applied to both clock outputs.

Bit 5 **MSIKRDY**: MSIK clock ready flag

Set by hardware to indicate that the MSIK is stable. This bit is set only when MSI kernel oscillator is enabled by software by setting MSIKON.

0: MSIK (MSI kernel) oscillator not ready

1: MSIK (MSI kernel) oscillator ready

Note: Once the MSIKON bit is cleared, MSIKRDY goes low after six MSIK oscillator clock cycles.

Bit 4 **MSIKON**: MSIK clock enable

Set and cleared by software.

Cleared by hardware to stop the MSIK when entering Stop, Standby or Shutdown mode.

Set by hardware to force the MSIK oscillator ON when exiting Standby or Shutdown mode.

Set by hardware to force the MSIK oscillator ON when STOPWUCK = 0 or STOPKERWUCK = 0 when exiting Stop modes or in case of a failure of the HSE oscillator.

0: MSIK (MSI kernel) oscillator disabled

1: MSIK (MSI kernel) oscillator enabled

Bit 3 **MSIPLLEN**: MSI clock PLL-mode 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). A hardware protection prevents from enabling MSIPLLEN if LSE is not ready.

This bit is cleared by hardware when LSE is disabled (LSEON = 0) or when the CSS on LSE detects a LSE failure (see RCC_CSR).

0: MSI PLL-mode OFF

1: MSI PLL-mode ON

Bit 2 **MSISRDY**: MSIS clock ready flag

Set by hardware to indicate that the MSIS oscillator is stable. This bit is set only when MSIS is enabled by software by setting MSISON.

0: MSIS (MSI system) oscillator not ready

1: MSIS (MSI system) oscillator ready

Note: Once the MSISON bit is cleared, MSISRDY goes low after six MSIS clock cycles.

Bit 1 **MSIKERON**: MSI enable for some peripheral kernels

Set and cleared by software to force MSI ON even in Stop modes. Keeping the MSI ON in Stop mode allows the communication speed not to be reduced by the MSI startup time. This bit has no effect on MSISON and MSIKON values (see *Peripherals clock gating and autonomous mode* for more details).

The MSIKERON must be configured at 0 before entering Stop 3 mode.

0: No effect on MSI oscillator

1: MSI oscillator forced ON even in Stop mode

Bit 0 **MSISON**: MSIS clock enable

Set and cleared by software.

Cleared by hardware to stop the MSIS oscillator when entering Stop, Standby or Shutdown mode.

Set by hardware to force the MSIS oscillator ON when exiting Standby or Shutdown mode.

Set by hardware to force the MSIS oscillator ON when STOPWUCK = 0 when exiting Stop modes or in case of a failure of the HSE oscillator.

Set by hardware when used directly or indirectly as system clock.

0: MSIS (MSI system) oscillator OFF

1: MSIS (MSI system) oscillator ON

11.8.2 RCC internal clock sources calibration register 1 (RCC_ICSCR1)

Address offset: 0x008

Reset value: 0x440X XXXX

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
MSISRANGE[3:0]				MSIKRANGE[3:0]				MSIRGSEL	MSIBIAS	Res.	Res.	MSICAL0[4:1]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSICAL0[0]	MSICAL1[4:0]					MSICAL2[4:0]					MSICAL3[4:0]				
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **MSISRANGE[3:0]**: MSIS clock ranges

These bits are configured by software to choose the frequency range of MSIS oscillator when MSIRGSEL is set. 16 frequency ranges are available:
0000: range 0 around 48 MHz
0001: range 1 around 24 MHz
0010: range 2 around 16 MHz
0011: range 3 around 12 MHz
0100: range 4 around 4 MHz (reset value)
0101: range 5 around 2 MHz
0110: range 6 around 1.33 MHz
0111: range 7 around 1 MHz
1000: range 8 around 3.072 MHz
1001: range 9 around 1.536 MHz
1010: range 10 around 1.024 MHz
1011: range 11 around 768 kHz
1100: range 12 around 400 kHz
1101: range 13 around 200 kHz
1110: range 14 around 133 kHz
1111: range 15 around 100 kHz

Note: MSISRANGE can be modified when MSIS is OFF (MSISON = 0) or when MSIS is ready (MSISRDY = 1). MSISRANGE must NOT be modified when MSIS is ON and NOT ready (MSISON = 1 and MSISRDY = 0)

MSISRANGE is kept when the device wakes up from Stop mode, except when the MSIS range is above 24 MHz. In this case MSISRANGE is changed by hardware into range 2 (24 MHz).

Bits 27:24 **MSIKRANGE[3:0]**: MSIK clock ranges

These bits are configured by software to choose the frequency range of MSIK oscillator when MSIRGSEL is set. 16 frequency ranges are available:
0000: range 0 around 48 MHz
0001: range 1 around 24 MHz
0010: range 2 around 16 MHz
0011: range 3 around 12 MHz
0100: range 4 around 4 MHz (reset value)
0101: range 5 around 2 MHz
0110: range 6 around 1.33 MHz
0111: range 7 around 1 MHz
1000: range 8 around 3.072 MHz
1001: range 9 around 1.536 MHz
1010: range 10 around 1.024 MHz
1011: range 11 around 768 kHz
1100: range 12 around 400 kHz
1101: range 13 around 200 kHz
1110: range 14 around 133 kHz
1111: range 15 around 100 kHz

Note: MSIKRANGE can be modified when MSIK is OFF (MSISON = 0) or when MSIK is ready (MSIKRDY = 1). MSIKRANGE must NOT be modified when MSIK is ON and NOT ready (MSIKON = 1 and MSIKRDY = 0)

MSIKRANGE is kept when the device wakes up from Stop mode, except when the MSIK range is above 24 MHz. In this case MSIKRANGE is changed by hardware into range 2 (24 MHz).

Bit 23 **MSIRGSEL**: MSI clock range selection

Set by software to select the MSIS and MSIK clocks range with MSISRANGE[3:0] and MSIKRANGE[3:0]. Write 0 has no effect.

After exiting Standby or Shutdown mode, or after a reset, this bit is at 0 and the MSIS and MSIK ranges are provided by MSISSRANGE[3:0] and MSIKSRANGE[3:0] in RCC_CSR.

0: MSIS/MSIK ranges provided by MSISSRANGE[3:0] and MSIKSRANGE[3:0] in RCC_CSR

1: MSIS/MSIK ranges provided by MSISRANGE[3:0] and MSIKRANGE[3:0] in RCC_ICSCR1

Bit 22 **MSIBIAS**: MSI bias mode selection

Set by software to select the MSI bias mode. By default, the MSI bias is in continuous mode in order to maintain the output clocks accuracy. Setting this bit reduces the MSI consumption when the regulator is in range 4, or when the device is in Stop 1 or Stop 2 mode, but it decreases the MSI accuracy

0: MSI bias continuous mode (clock accuracy fast settling time)

1: MSI bias sampling mode when the regulator is in range 4, or when the device is in Stop 1 or Stop 2 (ultra-low-power mode)

Bits 21:20 Reserved, must be kept at reset value.

Bits 19:15 **MSICAL0[4:0]**: MSIRC0 clock calibration for MSI ranges 0 to 3

These bits are initialized at startup with the factory-programmed MSIRC0 calibration trim value for ranges 0 to 3. When MSITRIM0 is written, MSICAL0 is updated with the sum of MSITRIM0[4:0] and the factory-programmed calibration trim value MSIRC0[4:0].

Caution: There is no hardware protection to limit a potential overflow due to the addition of MSITRIM bitfield and factory program bitfield for this calibration value. Control must be managed by software at user level.

Bits 14:10 **MSICAL1[4:0]**: MSIRC1 clock calibration for MSI ranges 4 to 7

These bits are initialized at startup with the factory-programmed MSIRC1 calibration trim value for ranges 4 to 7. When MSITRIM1 is written, MSICAL1 is updated with the sum of MSITRIM1[4:0] and the factory calibration trim value MSIRC1[4:0].

Caution: There is no hardware protection to limit a potential overflow due to the addition of MSITRIM bitfield and factory program bitfield for this calibration value. Control must be managed by software at user level.

Bits 9:5 **MSICAL2[4:0]**: MSIRC2 clock calibration for MSI ranges 8 to 11

These bits are initialized at startup with the factory-programmed MSIRC2 calibration trim value for ranges 8 to 11. When MSITRIM2 is written, MSICAL2 is updated with the sum of MSITRIM2[4:0] and the factory calibration trim value MSIRC2[4:0].

Caution: There is no hardware protection to limit a potential overflow due to the addition of MSITRIM bitfield and factory program bitfield for this calibration value. Control must be managed by software at user level.

Bits 4:0 **MSICAL3[4:0]**: MSIRC3 clock calibration for MSI ranges 12 to 15

These bits are initialized at startup with the factory-programmed MSIRC3 calibration trim value for ranges 12 to 15. When MSITRIM3 is written, MSICAL3 is updated with the sum of MSITRIM3[4:0] and the factory calibration trim value MSIRC2[4:0].

Caution: There is no hardware protection to limit a potential overflow due to the addition of MSITRIM bitfield and factory program bitfield for this calibration value. Control must be managed by software at user level.

11.8.3 RCC internal clock sources calibration register 2 (RCC_ICSCR2)

Address offset: 0x00C

Reset value: 0x0008 4210

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.	MSITRIM0[4:1]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSITRIM0[0]	MSITRIM1[4:0]					MSITRIM2[4:0]					MSITRIM3[4:0]				
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 19:15 **MSITRIM0[4:0]**: MSI clock trimming for ranges 0 to 3

These bits provide an additional user-programmable trimming value that is added to the factory-programmed calibration trim value MSIRC0[4:0] bits. It can be programmed to adjust to voltage and temperature variations that influence the frequency of the MSI.

Bits 14:10 **MSITRIM1[4:0]**: MSI clock trimming for ranges 4 to 7

These bits provide an additional user-programmable trimming value that is added to the factory-programmed calibration trim value MSIRC1[4:0] bits. It can be programmed to adjust to voltage and temperature variations that influence the frequency of the MSI.

Bits 9:5 **MSITRIM2[4:0]**: MSI clock trimming for ranges 8 to 11

These bits provide an additional user-programmable trimming value that is added to the factory-programmed calibration trim value MSIRC2[4:0] bits. It can be programmed to adjust to voltage and temperature variations that influence the frequency of the MSI.

Bits 4:0 **MSITRIM3[4:0]**: MSI clock trimming for ranges 12 to 15

These bits provide an additional user-programmable trimming value that is added to the factory-programmed calibration trim value MSIRC3[4:0] bits. It can be programmed to adjust to voltage and temperature variations that influence the frequency of the MSI.

Note: The hardware auto calibration with LSE must not be used in conjunction with software calibration.

11.8.4 RCC internal clock sources calibration register 3 (RCC_ICSCR3)

Address offset: 0x010

Reset value: 0x0010 0XXX

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.	HSITRIM[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.	HSICAL[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **HSITRIM[4:0]**: HSI clock trimming

These bits provide an additional user-programmable trimming value that is added to the HSICAL[11:0] bits. It can be programmed to adjust to voltage and temperature variations that influence the frequency of the HSI.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSICAL[11:0]**: HSI clock calibration

These bits are initialized at startup with the factory-programmed HSI calibration trim value. When HSITRIM is written, HSICAL is updated with the sum of HSITRIM and the factory trim value.

11.8.5 RCC clock recovery RC register (RCC_CRRCR)

Address offset: 0x014

Reset value: 0x0000 0XXX

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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSI48CAL[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **HSI48CAL[8:0]**: HSI48 clock calibration

These bits are initialized at startup with the factory-programmed HSI48 calibration trim value.

11.8.6 RCC clock configuration register 1 (RCC_CFGR1)

Address offset: 0x01C

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.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MCOPRE[2:0]			MCOSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOPKERWUCK	STOPWUCK	SWS[1:0]		SW[1:0]	
										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
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: MSIS clock selected
0011: HSI16 clock selected
0100: HSE clock selected
0101: Main PLL clock pll1_r_ck selected
0110: LSI clock selected
0111: LSE clock selected
1000: Internal HSI48 clock selected
1001: MSIK clock selected
Others: reserved

Note: This clock output may have some truncated cycles at startup or during MCO clock source switching.

Bits 23:6 Reserved, must be kept at reset value.

Bit 5 **STOPKERWUCK:** wakeup from Stop kernel clock automatic enable selection
Set and cleared by software to enable automatically another oscillator when exiting Stop mode. This oscillator can be used as independent kernel clock by peripherals.
0: MSIK oscillator automatically enabled when exiting Stop mode
1: HSI16 oscillator automatically enabled when exiting Stop mode

Bit 4 **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 RCC_CR and the system clock is HSE (SWS = 10) or a switch on HSE is requested (SW = 10).
0: MSIS oscillator selected as wakeup from stop clock and CSS backup clock
1: HSI16 oscillator selected as wakeup from stop clock and CSS backup clock

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: MSIS oscillator used as system clock
01: HSI16 oscillator used as system clock
10: HSE used as system clock
11: PLL pll1_r_ck 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 MSIS oscillator selection when exiting Standby or Shutdown mode. Configured by hardware to force MSIS or HSI16 oscillator selection when exiting Stop mode or in case of HSE oscillator failure, depending on STOPWUCK value.
00: MSIS selected as system clock
01: HSI16 selected as system clock
10: HSE selected as system clock
11: PLL pll1_r_ck selected as system clock

11.8.7 RCC clock configuration register 2 (RCC_CFGR2)

Address offset: 0x020

Reset value: 0x0000 0000

Access: 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB2 DIS	APB1 DIS	AHB2 DIS2	AHB2 DIS1	AHB1 DIS
											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.	PPRE2[2:0]			Res.	PPRE1[2:0]			HPRE[3:0]			
					rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **APB2DIS**: APB2 clock disable

This bit can be set in order to further reduce power consumption, when none of the APB2 peripherals are used and when their clocks are disabled in RCC_APB2ENR. When this bit is set, all the APB2 peripherals clocks are off.

0: APB2 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: APB2 clock disabled

Bit 19 **APB1DIS**: APB1 clock disable

This bit can be set in order to further reduce power consumption, when none of the APB1 peripherals (except IWDG) are used and when their clocks are disabled in RCC_APB1ENR. When this bit is set, all the APB1 peripherals clocks are off, except for IWDG.

0: APB1 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: APB1 clock disabled

Bit 18 **AHB2DIS2**: AHB2_2 clock disable

This bit can be set in order to further reduce power consumption, when none of the AHB2 peripherals from RCC_AHB2ENR2 are used and when their clocks are disabled in RCC_AHB2ENR2. When this bit is set, all the AHB2 peripherals clocks from RCC_AHB2EBNR2 are off.

0: AHB2_2 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: AHB2_2 clock disabled

Bit 17 **AHB2DIS1**: AHB2_1 clock disable

This bit can be set in order to further reduce power consumption, when none of the AHB2 peripherals from RCC_AHB2ENR1 (except SRAM2 and SRAM3) are used and when their clocks are disabled in RCC_AHB2ENR1. When this bit is set, all the AHB2 peripherals clocks from RCC_AHB2ENR1 are off, except for SRAM2 and SRAM3.

0: AHB2_1 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: AHB2_1 clock disabled

Bit 16 **AHB1DIS**: AHB1 clock disable

This bit can be set in order to further reduce power consumption, when none of the AHB1 peripherals (except those listed hereafter) are used and when their clocks are disabled in RCC_AHB1ENR. When this bit is set, all the AHB1 peripherals clocks are off, except for FLASH, BKPSRAM, ICACHE, DCACHE1 and SRAM1.

0: AHB1 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: AHB1 clock disabled

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

Bits 10:8 **PPRE2[2:0]**: APB2 prescaler

Set and cleared by software to control the division factor of the APB2 clock (PCLK2).

0xx: PCLK2 not divided

100: PCLK2 divided by 2

101: PCLK2 divided by 4

110: PCLK2 divided by 8

111: PCLK2 divided by 16

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **PPRE1[2:0]:** APB1 prescaler

Set and cleared by software to control the division factor of the APB1 clock (PCLK1).

0xx: PCLK1 not divided

100: PCLK1 divided by 2

101: PCLK1 divided by 4

110: PCLK1 divided by 8

111: PCLK1 divided by 16

Bits 3:0 **HPRE[3:0]**: AHB prescaler

Set and cleared by software to control the division factor of the AHB clock (HCLK).

Caution: Depending on the device voltage range, the software must set these bits correctly to ensure that the system frequency does not exceed the maximum allowed frequency (for more details, refer to *Table 103: Bus maximum frequency*). 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 is 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

11.8.8 RCC clock configuration register 3 (RCC_CFGR3)

Address offset: 0x024

Reset value: 0x0000 0000

Access: word, half-word and byte access

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB3 DIS	AHB3 DIS
														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.	PPRE3[2:0]			Res.	Res.	Res.	Res.
									rw	rw	rw				

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **APB3DIS**: APB3 clock disable

This bit can be set in order to further reduce power consumption, when none of the APB3 peripherals from RCC_APB3ENR are used and when their clocks are disabled in RCC_APB3ENR. When this bit is set, all the APB3 peripherals clocks are off.

0: APB3 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: APB3 clock disabled

Bit 16 **AHB3DIS**: AHB3 clock disable

This bit can be set in order to further reduce power consumption, when none of the AHB3 peripherals (except SRAM4) are used and when their clocks are disabled in RCC_AHB3ENR. When this bit is set, all the AHB3 peripherals clocks are off, except for SRAM4.

0: AHB3 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: AHB3 clock disabled

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

Bits 6:4 **PPRE3[2:0]**: APB3 prescaler

Set and cleared by software to control the division factor of the APB3 clock (PCLK3).

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 3:0 Reserved, must be kept at reset value.

11.8.9 RCC PLL1 configuration register (RCC_PLL1CFGR)

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.	PLL1REN	PLL1QEN	PLL1PEN
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL1MBOOST[3:0]				PLL1M[3:0]				Res.	Res.	Res.	PLL1FRACEN	PLL1RGE[1:0]		PLL1SRC[1:0]	
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 **PLL1REN**: PLL1 DIVR divider output enable

Set and reset by software to enable the pll1_r_ck output of the PLL1.

To save power, PLL1RENPLL2REN and PLL1R bits must be set to 0 when the pll1_r_ck is not used.

This bit can be cleared only when the PLL1 is not used as SYSCLK.

0: pll1_r_ck output disabled

1: pll1_r_ck output enabled

Bit 17 **PLL1QEN**: PLL1 DIVQ divider output enable
Set and reset by software to enable the pll1_q_ck output of the PLL1.
To save power, PLL1QEN and PLL1Q bits must be set to 0 when the pll1_q_ck is not used.
0: pll1_q_ck output disabled
1: pll1_q_ck output enabled

Bit 16 **PLL1PEN**: PLL1 DIVP divider output enable
Set and reset by software to enable the pll1_p_ck output of the PLL1.
To save power, PLL1PEN and PLL1P bits must be set to 0 when the pll1_p_ck is not used.
0: pll1_p_ck output disabled
1: pll1_p_ck output enabled

Bits 15:12 **PLL1MBOOST[3:0]**: Prescaler for EPOD booster input clock
Set and cleared by software to configure the prescaler of the PLL1, used for the EPOD booster. The EPOD booster input frequency is PLL1 input clock frequency/PLL1MBOOST.
This bit can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0) and EPOD Boost mode is disabled (see *Section 10: Power control (PWR)*).
0000: division by 1 (bypass)
0001: division by 2
0010: division by 4
0011: division by 6
0100: division by 8
0101: division by 10
0110: division by 12
0111: division by 14
1000: division by 16
others: reserved

Bits 11:8 **PLL1M[3:0]**: Prescaler for PLL1
Set and cleared by software to configure the prescaler of the PLL1. The VCO1 input frequency is PLL1 input clock frequency/PLL1M.
This bit can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
0000: division by 1 (bypass)
0001: division by 2
0010: division by 3
...
1111: division by 16

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **PLL1FRACEN**: PLL1 fractional latch enable
Set and reset by software to latch the content of PLL1FRACN into the ΣΔ modulator.
In order to latch the PLL1FRACN value into the ΣΔ modulator, PLL1FRACEN must be set to 0, then set to 1: the transition 0 to 1 transfers the content of PLL1FRACN into the modulator (see *PLL initialization phase* for details).

Bits 3:2 **PLL1RGE[1:0]**: PLL1 input frequency range
Set and reset by software to select the proper reference frequency range used for PLL1.
This bit must be written before enabling the PLL1.
00-01-10: PLL1 input (ref1_ck) clock range frequency between 4 and 8 MHz
11: PLL1 input (ref1_ck) clock range frequency between 8 and 16 MHz

Bits 1:0 **PLL1SRC[1:0]**: PLL1 entry clock source

Set and cleared by software to select PLL1 clock source. These bits can be written only when the PLL1 is disabled.
In order to save power, when no PLL1 is used, the value of PLL1SRC must be 0.
00: No clock sent to PLL1
01: MSIS clock selected as PLL1 clock entry
10: HSI16 clock selected as PLL1 clock entry
11: HSE clock selected as PLL1 clock entry

11.8.10 RCC PLL2 configuration register (RCC_PLL2CFGR)

Address offset: 0x02C

Reset value: 0x0000 0000

Access: no wait state; word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL2REN	PLL2QEN	PLL2PEN
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PLL2M[3:0]				Res.	Res.	Res.	PLL2FRACEN	PLL2RGE[1:0]		PLL2SRC[1:0]	
				rw	rw	rw	rw				rw	rw	rw	rw	rw

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

Bit 18 **PLL2REN**: PLL2 DIVR divider output enable

Set and reset by software to enable the pll2_r_ck output of the PLL2.
To save power, PLL2REN and PLL2R bits must be set to 0 when the pll2_r_ck is not used.
0: pll2_r_ck output disabled
1: pll2_r_ck output enabled

Bit 17 **PLL2QEN**: PLL2 DIVQ divider output enable

Set and reset by software to enable the pll2_q_ck output of the PLL2.
To save power, PLL2QEN and PLL2Q bits must be set to 0 when the pll2_q_ck is not used.
0: pll2_q_ck output disabled
1: pll2_q_ck output enabled

Bit 16 **PLL2PEN:** PLL2 DIVP divider output enable

Set and reset by software to enable the pll2_p_ck output of the PLL2.
To save power, PLL2PEN and PLL2P bits must be set to 0 when the pll2_p_ck is not used.
0: pll2_p_ck output disabled
1: pll2_p_ck output enabled

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **PLL2M[3:0]**: Prescaler for PLL2

Set and cleared by software to configure the prescaler of the PLL2. The VCO2 input frequency is PLL2 input clock frequency/PLL2M.
This bit can be written only when the PLL2 is disabled (PLL2ON = 0 and PLL2RDY = 0).
0000: division by 1 (bypass)
0001: division by 2
0010: division by 3
...
1111: division by 16

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **PLL2FRACEN**: PLL2 fractional latch enable

Set and reset by software to latch the content of PLL2FRACN into the ΣΔ modulator.
In order to latch the PLL2FRACN value into the ΣΔ modulator, PLL2FRACEN must be set to 0, then set to 1: the transition 0 to 1 transfers the content of PLL2FRACN into the modulator (see *PLL initialization phase* for details).

Bits 3:2 **PLL2RGE[1:0]**: PLL2 input frequency range

Set and reset by software to select the proper reference frequency range used for PLL2. This bit must be written before enabling the PLL2.
00-01-10: PLL2 input (ref2_ck) clock range frequency between 4 and 8 MHz
11: PLL2 input (ref2_ck) clock range frequency between 8 and 16 MHz

Bits 1:0 **PLL2SRC[1:0]**: PLL2 entry clock source

Set and cleared by software to select PLL2 clock source. These bits can be written only when the PLL2 is disabled.
In order to save power, when no PLL2 is used, the value of PLL2SRC must be 0.
00: No clock sent to PLL2
01: MSIS clock selected as PLL2 clock entry
10: HSI16 clock selected as PLL2 clock entry
11: HSE clock selected as PLL2 clock entry

11.8.11 RCC PLL3 configuration register (RCC_PLL3CFGR)

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.	PLL3REN	PLL3QEN	PLL3PEN
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PLL3M[3:0]				Res.	Res.	Res.	PLL3FRACEN	PLL3RGE[1:0]		PLL3SRC[1:0]	
				rw	rw	rw	rw				rw	rw	rw	rw	rw

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

Bit 18 **PLL3REN**: PLL3 DIVR divider output enable
Set and reset by software to enable the pll3_r_ck output of the PLL3.
To save power, PLL3REN and PLL3R bits must be set to 0 when the pll3_r_ck is not used.
0: pll3_r_ck output disabled
1: pll3_r_ck output enabled

Bit 17 **PLL3QEN**: PLL3 DIVQ divider output enable
Set and reset by software to enable the pll3_q_ck output of the PLL3.
To save power, PLL3QEN and PLL3Q bits must be set to 0 when the pll3_q_ck is not used.
0: pll3_q_ck output disabled
1: pll3_q_ck output enabled

Bit 16 **PLL3PEN:** PLL3 DIVP divider output enable
Set and reset by software to enable the pll3_p_ck output of the PLL3.
To save power, PLL3PEN and PLL3P bits must be set to 0 when the pll3_p_ck is not used.
0: pll3_p_ck output disabled
1: pll3_p_ck output enabled

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **PLL3M[3:0]**: Prescaler for PLL3
Set and cleared by software to configure the prescaler of the PLL3. The VCO3 input frequency is PLL3 input clock frequency/PLL3M.
This bit can be written only when the PLL3 is disabled (PLL3ON = 0 and PLL3RDY = 0).
0000: division by 1 (bypass)
0001: division by 2
0010: division by 3
...
1111: division by 16

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **PLL3FRACEN**: PLL3 fractional latch enable
Set and reset by software to latch the content of PLL3FRACN into the ΣΔ modulator.
In order to latch the PLL3FRACN value into the ΣΔ modulator, PLL3FRACEN must be set to 0, then set to 1: the transition 0 to 1 transfers the content of PLL3FRACN into the modulator (see *PLL initialization phase* for details).

Bits 3:2 **PLL3RGE[1:0]**: PLL3 input frequency range
Set and reset by software to select the proper reference frequency range used for PLL3.
This bit must be written before enabling the PLL3.
00-01-10: PLL3 input (ref3_ck) clock range frequency between 4 and 8 MHz
11: PLL3 input (ref3_ck) clock range frequency between 8 and 16 MHz

Bits 1:0 **PLL3SRC[1:0]**: PLL3 entry clock source
Set and cleared by software to select PLL3 clock source. These bits can be written only when the PLL3 is disabled.
In order to save power, when no PLL3 is used, the value of PLL3SRC must be 00.
00: No clock sent to PLL3
01: MSIS clock selected as PLL3 clock entry
10: HSI16 clock selected as PLL3 clock entry
11: HSE clock selected as PLL3 clock entry

11.8.12 RCC PLL1 dividers register (RCC_PLL1DIVR)

Address offset: 0x034

Reset value: 0x0101 0280

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.	PLL1R[6:0]							Res.	PLL1Q[6:0]						
	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
PLL1P[6:0]							PLL1N[8:0]								
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.

Bits 30:24 **PLL1R[6:0]**: PLL1 DIVR division factor

Set and reset by software to control the frequency of the pll1_r_ck clock.
These bits can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
Only division by 1 and even division factors are allowed.
0000000: Not allowed
0000001: pll1_r_ck = vco1_ck / 2 (default after reset)
0000010: pll1_r_ck = vco1_ck / 3
0000011: pll1_r_ck = vco1_ck / 4
...
1111111: pll1_r_ck = vco1_ck / 128

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **PLL1Q[6:0]**: PLL1 DIVQ division factor

Set and reset by software to control the frequency of the pll1_q_ck clock.
These bits can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
0000000: pll1_q_ck = vco1_ck
0000001: pll1_q_ck = vco1_ck / 2 (default after reset)
0000010: pll1_q_ck = vco1_ck / 3
0000011: pll1_q_ck = vco1_ck / 4
...
1111111: pll1_q_ck = vco1_ck / 128

Bits 15:9 **PLL1P[6:0]**: PLL1 DIVP division factor

Set and reset by software to control the frequency of the pll1_p_ck clock.
These bits can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
0000000: pll1_p_ck = vco1_ck
0000001: pll1_p_ck = vco1_ck / 2 (default after reset)
0000010: pll1_p_ck = vco1_ck
0000011: pll1_p_ck = vco1_ck / 4
...
1111111: pll1_p_ck = vco1_ck / 128

Bits 8:0 **PLL1N[8:0]**: Multiplication factor for PLL1 VCO

Set and reset by software to control the multiplication factor of the VCO.

These bits can be written only when the PLL is disabled (PLL1ON = 0 and PLL1RDY = 0).

0x003: PLL1N = 4

0x004: PLL1N = 5

0x005: PLL1N = 6

...

0x080: PLL1N = 129 (default after reset)

...

0x1FF: PLL1N = 512

Others: reserved

VCO output frequency = F_{ref1_ck} x PLL1N, when fractional value 0 has been loaded into PLL1FRACN, with:

- PLL1N between 4 and 512
- input frequency F_{ref1_ck} between 4 and 16 MHz

11.8.13 RCC PLL1 fractional divider register (RCC_PLL1FRACR)

Address offset: 0x038

Reset value: 0x0000 0000

Access: no wait state; word and half-word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL1FRACN[12:0]													Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:3 **PLL1FRACN[12:0]**: Fractional part of the multiplication factor for PLL1 VCO

Set and reset by software to control the fractional part of the multiplication factor of the VCO.

These bits can be written at any time, allowing dynamic fine-tuning of the PLL1 VCO.

VCO output frequency = F_{ref1_ck} x $\left(\text{PLL1N} + (\text{PLL1FRACN} / 2^{13})\right)$, with:

- PLL1N must be between 4 and 512.
- PLL1FRACN can be between 0 and 2^{13}- 1.
- The input frequency F_{ref1_ck} must be between 4 and 16 MHz.

To change the FRACN value on-the-fly even if the PLL is enabled, the application must proceed as follows:

- Set the bit PLL1FRACEN to 0.
- Write the new fractional value into PLL1FRACN.
- Set the bit PLL1FRACEN to 1.

Bits 2:0 Reserved, must be kept at reset value.

11.8.14 RCC PLL2 dividers configuration register (RCC_PLL2DIVR)

Address offset: 0x03C

Reset value: 0x0101 0280

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.	PLL2R[6:0]							Res.	PLL2Q[6:0]						
	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
PLL2P[6:0]								PLL2N[8:0]							
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.

Bits 30:24 **PLL2R[6:0]**: PLL2 DIVR division factor

Set and reset by software to control the frequency of the pll2_r_ck clock.
These bits can be written only when the PLL2 is disabled (PLL2ON = 0 and PLL2RDY = 0).
0000000: pll2_r_ck = vco2_ck
0000001: pll2_r_ck = vco2_ck / 2 (default after reset)
0000010: pll2_r_ck = vco2_ck / 3
0000011: pll2_r_ck = vco2_ck / 4
...
1111111: pll2_r_ck = vco2_ck / 128

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **PLL2Q[6:0]**: PLL2 DIVQ division factor

Set and reset by software to control the frequency of the pll2_q_ck clock.
These bits can be written only when the PLL2 is disabled (PLL2ON = 0 and PLL2RDY = 0).
0000000: pll2_q_ck = vco2_ck
0000001: pll2_q_ck = vco2_ck / 2 (default after reset)
0000010: pll2_q_ck = vco2_ck / 3
0000011: pll2_q_ck = vco2_ck / 4
...
1111111: pll2_q_ck = vco2_ck / 128

Bits 15:9 **PLL2P[6:0]**: PLL2 DIVP division factor

Set and reset by software to control the frequency of the pll2_p_ck clock.
These bits can be written only when the PLL2 is disabled (PLL2ON = 0 and PLL2RDY = 0).

0000000: pll2_p_ck = vco2_ck
0000001: pll2_p_ck = vco2_ck / 2 (default after reset)
0000010: pll2_p_ck = vco2_ck / 3
0000011: pll2_p_ck = vco2_ck / 4
...
1111111: pll2_p_ck = vco2_ck / 128

Bits 8:0 **PLL2N[8:0]**: Multiplication factor for PLL2 VCO

Set and reset by software to control the multiplication factor of the VCO.

These bits can be written only when the PLL is disabled (PLL2ON = 0 and PLL2RDY = 0).

0x003: PLL2N = 4

0x004: PLL2N = 5

0x005: PLL2N = 6

...

0x080: PLL2N = 129 (default after reset)

...

0x1FF: PLL2N = 512

Others: reserved

VCO output frequency = F_{ref2_ck} x PLL2N, when fractional value 0 has been loaded into PLL2FRACN, with:

- PLL2N between 4 and 512
- input frequency F_{ref2_ck} between 1MHz and 16MHz

11.8.15 RCC PLL2 fractional divider register (RCC_PLL2FRACR)

Address offset: 0x040

Reset value: 0x0000 0000

Access: no wait state; word and half-word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL2FRACN[12:0]													Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:3 **PLL2FRACN[12:0]**: Fractional part of the multiplication factor for PLL2 VCO

Set and reset by software to control the fractional part of the multiplication factor of the VCO.

These bits can be written at any time, allowing dynamic fine-tuning of the PLL2 VCO.

VCO output frequency = F_{ref2_ck} x $\left(\text{PLL2N} + (\text{PLL2FRACN} / 2^{13})\right)$, with

- PLL2N must be between 4 and 512.
- PLL2FRACN can be between 0 and 2^{13} - 1.
- The input frequency F_{ref2_ck} must be between 4 and 16 MHz.

In order to change the FRACN value on-the-fly even if the PLL is enabled, the application must proceed as follows:

- Set the bit PLL2FRACEN to 0.
- Write the new fractional value into PLL2FRACN.
- Set the bit PLL2FRACEN to 1.

Bits 2:0 Reserved, must be kept at reset value.

11.8.16 RCC PLL3 dividers configuration register (RCC_PLL3DIVR)

Address offset: 0x044

Reset value: 0x0101 0280

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.	PLL3R[6:0]							Res.	PLL3Q[6:0]						
	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
PLL3P[6:0]								PLL3N[8:0]							
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.

Bits 30:24 **PLL3R[6:0]**: PLL3 DIVR division factor

Set and reset by software to control the frequency of the pll3_r_ck clock.
These bits can be written only when the PLL3 is disabled (PLL3ON = 0 and PLL3RDY = 0).
0000000: pll3_r_ck = vco3_ck
0000001: pll3_r_ck = vco3_ck / 2 (default after reset)
0000010: pll3_r_ck = vco3_ck / 3
0000011: pll3_r_ck = vco3_ck / 4
...
1111111: pll3_r_ck = vco3_ck / 128

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **PLL3Q[6:0]**: PLL3 DIVQ division factor

Set and reset by software to control the frequency of the pll3_q_ck clock.
These bits can be written only when the PLL3 is disabled (PLL3ON = 0 and PLL3RDY = 0).
0000000: pll3_q_ck = vco3_ck
0000001: pll3_q_ck = vco3_ck / 2 (default after reset)
0000010: pll3_q_ck = vco3_ck / 3
0000011: pll3_q_ck = vco3_ck / 4
...
1111111: pll3_q_ck = vco3_ck / 128

Bits 15:9 **PLL3P[6:0]**: PLL3 DIVP division factor

Set and reset by software to control the frequency of the pll3_p_ck clock.
These bits can be written only when the PLL3 is disabled (PLL3ON = 0 and PLL3RDY = 0).

0000000: pll3_p_ck = vco3_ck
0000001: pll3_p_ck = vco3_ck / 2 (default after reset)
0000010: pll3_p_ck = vco3_ck / 3
0000011: pll3_p_ck = vco3_ck / 4
...
1111111: pll3_p_ck = vco3_ck / 128

Bits 8:0 **PLL3N[8:0]**: Multiplication factor for PLL3 VCO

Set and reset by software to control the multiplication factor of the VCO.

These bits can be written only when the PLL is disabled (PLL3ON = 0 and PLL3RDY = 0).

0x003: PLL3N = 4

0x004: PLL3N = 5

0x005: PLL3N = 6

...

0x080: PLL3N = 129 (default after reset)

...

0x1FF: PLL3N = 512

Others: reserved

VCO output frequency = F_{ref3_ck} x PLL3N, when fractional value 0 has been loaded into PLL3FRACN, with:

- PLL3N between 4 and 512
- input frequency F_{ref3_ck} between 4 and 16MHz

11.8.17 RCC PLL3 fractional divider register (RCC_PLL3FRACR)

Address offset: 0x048

Reset value: 0x0000 0000

Access: no wait state; word and half-word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL3FRACN[12:0]													Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:3 **PLL3FRACN[12:0]**: Fractional part of the multiplication factor for PLL3 VCO

Set and reset by software to control the fractional part of the multiplication factor of the VCO.

These bits can be written at any time, allowing dynamic fine-tuning of the PLL3 VCO.

VCO output frequency = F_{ref3_ck} x $\left(\text{PLL3N} + (\text{PLL3FRACN} / 2^{13})\right)$, with:

- PLL3N must be between 4 and 512.
- PLL3FRACN can be between 0 and 2^{13} - 1.
- The input frequency F_{ref3_ck} must be between 4 and 16 MHz.

In order to change the FRACN value on-the-fly even if the PLL is enabled, the application must proceed as follows:

- Set the bit PLL3FRACEN to 0.
- Write the new fractional value into PLL3FRACN.
- Set the bit PLL3FRACEN to 1.

Bits 2:0 Reserved, must be kept at reset value.

11.8.18 RCC clock interrupt enable register (RCC_CIER)

Address offset: 0x050

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.	SHSIRDYIE	MSIKRDYIE	Res.	Res.	PLL3RDYIE	PLL2RDYIE	PLL1RDYIE	HSI48RDYIE	HSERDYIE	HSIRDYIE	MSISRDYIE	LSERDYIE	LSIRDYIE
			rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **SHSIRDYIE**: SHSI ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the SHSI oscillator stabilization.
0: SHSI ready interrupt disabled
1: SHSI ready interrupt enabled

Bit 11 **MSIKRDYIE**: MSIK ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the MSIK oscillator stabilization.
0: MSIK ready interrupt disabled
1: MSIK ready interrupt enabled

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **PLL3RDYIE**: PLL3 ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by PLL3 lock.
0: PLL3 lock interrupt disabled
1: PLL3 lock interrupt enabled

Bit 7 **PLL2RDYIE**: PLL2 ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by PLL2 lock.
0: PLL2 lock interrupt disabled
1: PLL2 lock interrupt enabled

Bit 6 **PLL1RDYIE**: PLL ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by PLL1 lock.
0: PLL1 lock interrupt disabled
1: PLL1 lock interrupt enabled

Bit 5 **HSI48RDYIE**: HSI48 ready interrupt enable
Set and cleared by software to enable/disable interrupt caused by the HSI48 oscillator stabilization.
0: HSI48 ready interrupt disabled
1: HSI48 ready 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 **MSISRDYIE**: MSIS ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the MSIS oscillator stabilization.

0: MSIS ready interrupt disabled

1: MSIS 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

11.8.19 RCC clock interrupt flag register (RCC_CIFR)

Address offset: 0x054

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.	SHSIRDYF	MSIKRDYF	CSSF	Res.	PLL3RDYF	PLL2RDYF	PLL1RDYF	HSI48RDYF	HSERDYF	HSIRDYF	MSISRDYF	LSERDYF	LSIRDYF
			r	r	r		r	r	r	r	r	r	r	r	r

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **SHSIRDYF**: SHSI ready interrupt flag

Set by hardware when the SHSI clock becomes stable and SHSIRDYIE is set.

Cleared by software setting the SHSIRDYC bit.

0: No clock ready interrupt caused by the SHSI oscillator

1: Clock ready interrupt caused by the SHSI oscillator

Bit 11 **MSIKRDYF**: MSIK ready interrupt flag
Set by hardware when the MSIK clock becomes stable and MSIKRDYIE is set.
Cleared by software setting the MSIKRDYC bit.
0: No clock ready interrupt caused by the MSIK oscillator
1: Clock ready interrupt caused by the MSIK oscillator

Bit 10 **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 9 Reserved, must be kept at reset value.

Bit 8 **PLL3RDYF**: PLL3 ready interrupt flag
Set by hardware when the PLL3 locks and PLL3RDYIE is set.
Cleared by software setting the PLL3RDYC bit.
0: No clock ready interrupt caused by PLL3 lock
1: Clock ready interrupt caused by PLL3 lock

Bit 7 **PLL2RDYF**: PLL2 ready interrupt flag
Set by hardware when the PLL2 locks and PLL2RDYIE is set.
Cleared by software setting the PLL2RDYC bit.
0: No clock ready interrupt caused by PLL2 lock
1: Clock ready interrupt caused by PLL2 lock

Bit 6 **PLL1RDYF**: PLL1 ready interrupt flag
Set by hardware when the PLL1 locks and PLL1RDYIE is set.
Cleared by software setting the PLL1RDYC bit.
0: No clock ready interrupt caused by PLL1 lock
1: Clock ready interrupt caused by PLL1 lock

Bit 5 **HSI48RDYF**: HSI48 ready interrupt flag
Set by hardware when the HSI48 clock becomes stable and HSI48RDYIE is set.
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 4 **HSERDYF**: HSE ready interrupt flag
Set by hardware when the HSE clock becomes stable and HSERDYIE 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 HSIRDYIE is set in a response to setting the HSION (see 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 **MSISRDYF**: MSIS ready interrupt flag
Set by hardware when the MSIS clock becomes stable and MSISRDYIE is set.
Cleared by software setting the MSISRDYC bit.
0: No clock ready interrupt caused by the MSIS oscillator
1: Clock ready interrupt caused by the MSIS oscillator

Bit 1 **LSERDYF**: LSE ready interrupt flag
Set by hardware when the LSE clock becomes stable and LSERDYIE 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 LSIRDYIE 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

11.8.20 RCC clock interrupt clear register (RCC_CICR)

Address offset: 0x058

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.	SHSIRDYC	MSIKRDYC	CSSC	Res.	PLL3RDYC	PLL2RDYC	PLL1RDYC	HSI48RDYC	HSERDYC	HSIRDYC	MSISRDYC	LSERDYC	LSIRDYC
			w	w	w		w	w	w	w	w	w	w	w	w

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **SHSIRDYC**: SHSI oscillator ready interrupt clear
Writing this bit to 1 clears the SHSIRDYF flag. Writing 0 has no effect.

Bit 11 **MSIKRDYC**: MSIK oscillator ready interrupt clear
Writing this bit to 1 clears the MSIKRDYF flag. Writing 0 has no effect.

Bit 10 **CSSC**: Clock security system interrupt clear
Writing this bit to 1 clears the CSSF flag. Writing 0 has no effect.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **PLL3RDYC**: PLL3 ready interrupt clear
Writing this bit to 1 clears the PLL3RDYF flag. Writing 0 has no effect.

Bit 7 **PLL2RDYC**: PLL2 ready interrupt clear
Writing this bit to 1 clears the PLL2RDYF flag. Writing 0 has no effect.

Bit 6 **PLL1RDYC**: PLL1 ready interrupt clear
Writing this bit to 1 clears the PLL1RDYF flag. Writing 0 has no effect.

Bit 5 **HSI48RDYC**: HSI48 ready interrupt clear
Writing this bit to 1 clears the HSI48RDYF flag. Writing 0 has no effect.

Bit 4 **HSERDYC**: HSE ready interrupt clear
Writing this bit to 1 clears the HSERDYF flag. Writing 0 has no effect.

Bit 3 **HSIRDYC**: HSI16 ready interrupt clear
Writing this bit to 1 clears the HSIRDYF flag. Writing 0 has no effect.

Bit 2 **MSISRDYC**: MSIS ready interrupt clear
Writing this bit to 1 clears the MSISRDYF flag. Writing 0 has no effect.

Bit 1 **LSERDYC**: LSE ready interrupt clear
Writing this bit to 1 clears the LSERDYF flag. Writing 0 has no effect.

Bit 0 **LSIRDYC**: LSI ready interrupt clear
Writing this bit to 1 clears the LSIRDYF flag. Writing 0 has no effect.

11.8.21 RCC AHB1 peripheral reset register (RCC_AHB1RSTR)

Address offset: 0x060

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.	DMA2DRST	RAMCFGRST	TSCRST
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CRCRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDF1RST	FMACRST	CORDICRST	GPDMA1RST
			rw									rw	rw	rw	rw

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

Bit 18 **DMA2DRST**: DMA2D reset
Set and cleared by software.
0: No effect
1: Reset DMA2D

Bit 17 **RAMCFGRST**: RAMCFG reset
Set and cleared by software.
0: No effect
1: Reset RAMCFG

Bit 16 **TSCRST**: TSC 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 11:4 Reserved, must be kept at reset value.

Bit 3 **MDF1RST:** MDF1 reset
Set and cleared by software.
0: No effect
1: Reset MDF1

Bit 2 **FMACRST:** FMAC reset
Set and cleared by software.
0: No effect
1: Reset FMAC

Bit 1 **CORDICRST**: CORDIC reset
Set and cleared by software.
0: No effect
1: Reset CORDIC

Bit 0 **GPDMA1RST**: GPDMA1 reset
Set and cleared by software.
0: No effect
1: Reset GPDMA1

11.8.22 RCC AHB2 peripheral reset register 1 (RCC_AHB2RSTR1)

Address offset: 0x064

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.	SDMMC2RST	SDMMC1RST	Res.	Res.	OTFDEC2RST	OTFDEC1RST	Res.	OCTOSPIMRST	SAESRST	PKARST	RNGRST	HASHRST	AESRST
			rw	rw			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.	OTGRST	Res.	DCMI_PSSIRST	Res.	ADC1RST	Res.	GPIOIRST	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
	rw		rw		rw		rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **SDMMC2RST**: SDMMC2 reset
Set and cleared by software.
0: No effect
1: Reset SDMMC2

Bit 27 **SDMMC1RST**: SDMMC1 reset
Set and cleared by software.
0: No effect
1: Reset SDMMC1

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **OTFDEC2RST**: OTFDEC2 reset
Set and cleared by software.
0: No effect
1: Reset OTFDEC2

Bit 23 **OTFDEC1RST**: OTFDEC1 reset
Set and cleared by software.
0: No effect
1: Reset OTFDEC1

Bit 22 Reserved, must be kept at reset value.

Bit 21 **OCTOSPIMRST**: OCTOSPIM reset
Set and cleared by software.
0: No effect
1: Reset OCTOSPIM

Bit 20 **SAESRST**: SAES hardware accelerator reset
Set and cleared by software.
0: No effect
1: Reset SAES

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

Bit 15 Reserved, must be kept at reset value.

Bit 14 **OTGRST**: OTG_FS reset
Set and cleared by software.
0: No effect
1: Reset OTG_FS

Bit 13 Reserved, must be kept at reset value.

Bit 12 **DCMI_PSSIRST**: DCMI and PSSI reset
Set and cleared by software.
0: No effect
1: Reset DCMI and PSSI

Bit 11 Reserved, must be kept at reset value.

Bit 10 **ADC1RST**: ADC1 reset
Set and cleared by software.
0: No effect
1: Reset ADC1

Bit 9 Reserved, must be kept at reset value.

Bit 8 **GPIOIRST**: IO port I reset
Set and cleared by software.
0: No effect
1: Reset IO port I

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

11.8.23 RCC AHB2 peripheral reset register 2 (RCC_AHB2RSTR2)

Address offset: 0x068

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.	OCTOSPI2RST	Res.	Res.	Res.	OCTOSPI1RST	Res.	Res.	Res.	FSMCRST
							rw				rw				rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **OCTOSPI2RST**: OCTOSPI2 reset
Set and cleared by software.
0: No effect
1: Reset OCTOSPI2

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **OCTOSPI1RST**: OCTOSPI1 reset
Set and cleared by software.
0: No effect
1: Reset OCTOSPI1

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **FSMCRST**: Flexible memory controller reset
Set and cleared by software.
0: No effect
1: Reset FSMC

11.8.24 RCC AHB3 peripheral reset register (RCC_AHB3RSTR)

Address offset: 0x06C

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.	ADF1RST	LPDMA1RST	Res.	Res.	DAC1RST	ADC4RST	Res.	Res.	Res.	Res.	LPGPIO1RST
					rw	rw			rw	rw					rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **ADF1RST**: ADF1 reset
Set and cleared by software.
0: No effect
1: Reset ADF1

Bit 9 **LPDMA1RST**: LPDMA1 reset
Set and cleared by software.
0: No effect
1: Reset LPDMA1

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **DAC1RST**: DAC1 reset
Set and cleared by software.
0: No effect
1: Reset DAC1

Bit 5 **ADC4RST**: ADC4 reset
Set and cleared by software.
0: No effect
1: Reset ADC4 interface

Bits 4:1 Reserved, must be kept at reset value.

Bit 0 **LPGPIO1RST**: LPGPIO1 reset
Set and cleared by software.
0: No effect
1: Reset LPGPIO1

11.8.25 RCC APB1 peripheral reset register 1 (RCC_APB1RSTR1)

Address offset: 0x074

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.	CRSRST	Res.	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	Res.
							rw		rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
	rw									rw	rw	rw	rw	rw	rw

Bits 31:25 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 Reserved, must be kept at reset value.

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

Bits 16:15 Reserved, must be kept at reset value.

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 reset
Set and cleared by software.
0: No effect
1: Reset TIM7

Bit 4 **TIM6RST**: TIM6 reset
Set and cleared by software.
0: No effect
1: Reset TIM6

Bit 3 **TIM5RST**: TIM5 reset
Set and cleared by software.
0: No effect
1: Reset TIM5

Bit 2 **TIM4RST**: TIM4 reset
Set and cleared by software.
0: No effect
1: Reset TIM4

Bit 1 **TIM3RST**: TIM3 reset
Set and cleared by software.
0: No effect
1: Reset TIM3

Bit 0 **TIM2RST**: TIM2 reset
Set and cleared by software.
0: No effect
1: Reset TIM2

11.8.26 RCC APB1 peripheral reset register 2 (RCC_APB1RSTR2)

Address offset: 0x078

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.	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.	FDCAN1RST	Res.	Res.	Res.	LPTIM2RST	Res.	Res.	Res.	I2C4RST	Res.
						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

Bits 22:10 Reserved, must be kept at reset value.

Bit 9 **FDCAN1RST**: FDCAN1 reset
Set and cleared by software.
0: No effect
1: Reset FDCAN1

Bits 8:6 Reserved, must be kept at reset value.

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 Reserved, must be kept at reset value.

11.8.27 RCC APB2 peripheral reset register (RCC_APB2RSTR)

Address offset: 0x07C

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.	SAI2RST	SAI1RST	Res.	Res.	TIM17RST	TIM16RST	TIM15RST
									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.	Res.
	rw	rw	rw	rw											

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

Bit 22 **SAI2RST**: SAI2 reset
Set and cleared by software.
0: No effect
1: Reset SAI2

Bit 21 **SAI1RST**: 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 reset
Set and cleared by software.
0: No effect
1: Reset TIM17

Bit 17 **TIM16RST**: TIM16 reset
Set and cleared by software.
0: No effect
1: Reset TIM16

Bit 16 **TIM15RST**: TIM15 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 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 reset
Set and cleared by software.
0: No effect
1: Reset TIM1

Bits 10:0 Reserved, must be kept at reset value.

11.8.28 RCC APB3 peripheral reset register (RCC_APB3RSTR)

Address offset: 0x080

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.	VREFRST	Res.	Res.	Res.	Res.
											rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMPRST	OPAMPRST	LPTIM4RST	LPTIM3RST	LPTIM1RST	Res.	Res.	Res.	I2C3RST	LPUART1RST	SPI3RST	Res.	Res.	Res.	SYSCFGRST	Res.
rw	rw	rw	rw	rw				rw	rw	rw				rw	

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **VREFRST**: VREFBUF reset
Set and cleared by software.
0: No effect
1: Reset VREFBUF

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **COMPRST**: COMP reset
Set and cleared by software.
0: No effect
1: Reset COMP

Bit 14 **OPAMPRST**: OPAMP reset
Set and cleared by software.
0: No effect
1: Reset OPAMP

Bit 13 **LPTIM4RST**: LPTIM4 reset
Set and cleared by software.
0: No effect
1: Reset LPTIM4

Bit 12 **LPTIM3RST**: LPTIM3 reset
Set and cleared by software.
0: No effect
1: Reset LPTIM3

Bit 11 **LPTIM1RST**: LPTIM1 reset
Set and cleared by software.
0: No effect
1: Reset LPTIM1

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **I2C3RST**: I2C3 reset
Set and cleared by software.
0: No effect
1: Reset I2C3

Bit 6 **LPUART1RST**: LPUART1 reset
Set and cleared by software.
0: No effect
1: Reset LPUART1

Bit 5 **SPI3RST**: SPI3 reset
Set and cleared by software.
0: No effect
1: Reset SPI3

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **SYSCFGRST**: SYSCFG reset
Set and cleared by software.
0: No effect
1: Reset SYSCFG

Bit 0 Reserved, must be kept at reset value.

11.8.29 RCC AHB1 peripheral clock enable register (RCC_AHB1ENR)

Address offset: 0x088

Reset value: 0xD000 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
SRAM1EN	DCACHE1EN	Res.	BKPSRAMEN	Res.	Res.	Res.	GTZC1EN	Res.	Res.	Res.	Res.	Res.	DMA2DEN	RAMCFGEN	TSCEN
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.	CRCEN	Res.	Res.	Res.	FLASHEN	Res.	Res.	Res.	Res.	MDF1EN	FMACEN	CORDICEN	GPDMA1EN
			rw				rw					rw	rw	rw	rw

Bit 31 **SRAM1EN**: SRAM1 clock enable
Set and reset by software.
0: SRAM1 clock disabled
1: SRAM1 clock enabled

Bit 30 **DCACHE1EN**: DCACHE1 clock enable
Set and reset by software.
0: DCACHE1 clock disabled
1: DCACHE1 clock enabled
Note: DCACHE1 clock must be enabled when external memories are accessed through OCTOSPI1, OCTOSPI2, or FSMC, even if the DCACHE1 is bypassed.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **BKPSRAMEN**: BKPSRAM clock enable
Set and reset by software.
0: BKPSRAM clock disabled
1: BKPSRAM clock enabled

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

Bit 24 **GTZC1EN**: GTZC1 clock enable
Set and reset by software.
0: GTZC1 clock disabled
1: GTZC1 clock enabled

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

Bit 18 **DMA2DEN**: DMA2D clock enable
Set and cleared by software.
0: DMA2D clock disabled
1: DMA2D clock enabled

Bit 17 **RAMCFGEN**: RAMCFG clock enable
Set and cleared by software.
0: RAMCFG clock disabled
1: RAMCFG clock enabled

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 clock enable
Set and cleared by software. This bit can be disabled only when the Flash memory is in power down mode.
0: FLASH clock disabled
1: FLASH clock enabled

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

Bit 3 **MDF1EN**: MDF1 clock enable
Set and reset by software.
0: MDF1 clock disabled
1: MDF1 clock enabled

Bit 2 **FMACEN**: FMAC clock enable
Set and reset by software.
0: FMAC clock disabled
1: FMAC clock enabled

Bit 1 **CORDICEN**: CORDIC clock enable
Set and cleared by software.
0: CORDIC clock disabled
1: CORDIC clock enabled

Bit 0 **GPDMA1EN**: GPDMA1 clock enable
Set and cleared by software.
0: GPDMA1 clock disabled
1: GPDMA1 clock enabled

11.8.30 RCC AHB2 peripheral clock enable register 1 (RCC_AHB2ENR1)

Address offset: 0x08C

Reset value: 0xC000 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
SRAM3EN	SRAM2EN	Res.	SDMMC2EN	SDMMC1EN	Res.	Res.	OTFDEC2EN	OTFDEC1EN	Res.	OCTOSPIMEN	SAESEN	PKAEN	RNGEN	HASHEN	AESEN
rw	rw		rw	rw			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.	OTGEN	Res.	DCMI_PSSIEN	Res.	ADC1EN	Res.	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
	rw		rw		rw		rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SRAM3EN**: SRAM3 clock enable
Set and reset by software.
0: SRAM3 clock disabled
1: SRAM3 clock enabled

Bit 30 **SRAM2EN**: SRAM2 clock enable
Set and reset by software.
0: SRAM2 clock disabled
1: SRAM2 clock enabled

Bit 29 Reserved, must be kept at reset value.

Bit 28 **SDMMC2EN**: SDMMC2 clock enable
Set and cleared by software.
0: SDMMC2 clock disabled
1: SDMMC2 clock enabled

Bit 27 **SDMMC1EN**: SDMMC1 clock enable
Set and cleared by software.
0: SDMMC1 clock disabled
1: SDMMC1 clock enabled

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **OTFDEC2EN**: OTFDEC2 clock enable
Set and cleared by software.
0: OTFDEC2 clock disabled
1: OTFDEC2 clock enabled

Bit 23 **OTFDEC1EN**: OTFDEC1 clock enable
Set and cleared by software.
0: OTFDEC1 clock disabled
1: OTFDEC1 clock enabled

Bit 22 Reserved, must be kept at reset value.

Bit 21 **OCTOSPIMEN:** OCTOSPIM clock enable
Set and cleared by software.
0: OCTOSPIM clock disabled
1: OCTOSPIM clock enabled

Bit 20 **SAESEN:** SAES clock enable
Set and cleared by software.
0: SAES clock disabled
1: SAES clock enabled

Bit 19 **PKAEN:** PKA clock enable
Set and cleared by software.
0: PKA clock disabled
1: PKA clock enabled

Bit 18 **RNGEN:** RNG clock enable
Set and cleared by software.
0: RNG clock disabled
1: RNG 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 clock enable
Set and cleared by software.
0: AES clock disabled
1: AES clock enabled

Bit 15 Reserved, must be kept at reset value.

Bit 14 **OTGEN:** OTG_FS clock enable
Set and cleared by software.
0: OTG_FS clock disabled
1: OTG_FS clock enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **DCMI_PSSIEN:** DCMI and PSSI clock enable
Set and cleared by software.
0: DCMI and PSSI clock disabled
1: DCMI and PSSI clock enabled

Bit 11 Reserved, must be kept at reset value.

Bit 10 **ADC1EN:** ADC1 clock enable
Set and cleared by software.
0: ADC1 clock disabled
1: ADC1 clock enabled

Bit 9 Reserved, must be kept at reset value.

Bit 8 **GPIOIEN:** IO port I clock enable
Set and cleared by software.
0: IO port I clock disabled
1: IO port I clock enabled

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

11.8.31 RCC AHB2 peripheral clock enable register 2 (RCC_AHB2ENR2)

Address offset: 0x090

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.	OCTOSPI2EN	Res.	Res.	Res.	OCTOSPI1EN	Res.	Res.	Res.	FSMCEN
							rw				rw				rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **OCTOSPI2EN**: OCTOSPI2 clock enable
Set and cleared by software.
0: OCTOSPI2 clock disabled
1: OCTOSPI2 clock enabled

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **OCTOSPI1EN**: OCTOSPI1 clock enable
Set and cleared by software.
0: OCTOSPI1 clock disabled
1: OCTOSPI1 clock enabled

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **FSMCEN**: FSMC clock enable
Set and cleared by software.
0: FSMC clock disabled
1: FSMC clock enabled

11.8.32 RCC AHB3 peripheral clock enable register (RCC_AHB3ENR)

Address offset: 0x094

Reset value: 0x8000 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
SRAM4EN	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.	GTZC2EN	Res.	ADF1EN	LPDMA1EN	Res.	Res.	DAC1EN	ADC4EN	Res.	Res.	PWREN	Res.	LPGPIO1EN
			rw		rw	rw			rw	rw			rw		rw

Bit 31 **SRAM4EN**: SRAM4 clock enable
Set and reset by software.
0: SRAM4 clock disabled
1: SRAM4 clock enabled

Bits 30:13 Reserved, must be kept at reset value.

Bit 12 **GTZC2EN**: GTZC2 clock enable
Set and cleared by software.
0: GTZC2 clock disabled
1: GTZC2 clock enabled

Bit 11 Reserved, must be kept at reset value.

Bit 10 **ADF1EN**: ADF1 clock enable
Set and cleared by software.
0: ADF1 clock disabled
1: ADF1 clock enabled

Bit 9 **LPDMA1EN**: LPDMA1 clock enable
Set and cleared by software.
0: LPDMA1 clock disabled
1: LPDMA1 clock enabled

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **DAC1EN**: DAC1 clock enable
Set and cleared by software.
0: DAC1 clock disabled
1: DAC1 clock enabled

Bit 5 **ADC4EN**: ADC4 clock enable
Set and cleared by software.
0: ADC4 clock disabled
1: ADC4 clock enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **PWREN**: PWR clock enable
Set and cleared by software.
0: PWR clock disabled
1: PWR clock enabled

Bit 1 Reserved, must be kept at reset value.

Bit 0 **LPGPIO1EN**: LPGPIO1 enable
Set and cleared by software.
0: LPGPIO1 clock disabled
1: LPGPIO1 clock enabled

11.8.33 RCC APB1 peripheral clock enable register 1 (RCC_APB1ENR1)

Address offset: 0x09C

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.	CRSEN	Res.	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Res.
							rw		rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2EN	Res.	Res.	WWDGEN	Res.	Res.	Res.	Res.	Res.	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
	rw			rs						rw	rw	rw	rw	rw	rw

Bits 31:25 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 Reserved, must be kept at reset value.

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

Bits 16:15 Reserved, must be kept at reset value.

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**: WWDG 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: WWDG clock disabled
1: WWDG clock enabled

Bits 10:6 Reserved, must be kept at reset value.

Bit 5 **TIM7EN**: TIM7 clock enable
Set and cleared by software.
0: TIM7 clock disabled
1: TIM7 clock enabled

Bit 4 **TIM6EN**: TIM6 clock enable
Set and cleared by software.
0: TIM6 clock disabled
1: TIM6 clock enabled

Bit 3 **TIM5EN**: TIM5 clock enable
Set and cleared by software.
0: TIM5 clock disabled
1: TIM5 clock enabled

Bit 2 **TIM4EN**: TIM4 clock enable
Set and cleared by software.
0: TIM4 clock disabled
1: TIM4 clock enabled

Bit 1 **TIM3EN**: TIM3 clock enable
Set and cleared by software.
0: TIM3 clock disabled
1: TIM3 clock enabled

Bit 0 **TIM2EN**: TIM2 clock enable
Set and cleared by software.
0: TIM2 clock disabled
1: TIM2 clock enabled

11.8.34 RCC APB1 peripheral clock enable register 2 (RCC_APB1ENR2)

Address offset: 0x0A0

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.	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.	FDCAN1EN	Res.	Res.	Res.	LPTIM2EN	Res.	Res.	Res.	I2C4EN	Res.
						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

Bits 22:10 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 8:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2EN:** LPTIM2 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 Reserved, must be kept at reset value.

11.8.35 RCC APB2 peripheral clock enable register (RCC_APB2ENR)

Address offset: 0x0A4

Reset value: 0x0000 0000

Access: word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2EN	SAI1EN	Res.	Res.	TIM17EN	TIM16EN	TIM15EN
									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.	Res.
	rw	rw	rw	rw											

Bits 31:23 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 clock enable
Set and cleared by software.
0: TIM17 clock disabled
1: TIM17 clock enabled

Bit 17 **TIM16EN**: TIM16 clock enable
Set and cleared by software.
0: TIM16 clock disabled
1: TIM16 clock enabled

Bit 16 **TIM15EN**: TIM15 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 clock enable
Set and cleared by software.
0: TIM8 clock disabled
1: TIM8 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 clock enable
Set and cleared by software.
0: TIM1 clock disabled
1: TIM1 clock enabled

Bits 10:0 Reserved, must be kept at reset value.

11.8.36 RCC APB3 peripheral clock enable register (RCC_APB3ENR)

Address offset: 0x0A8

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.	RTCAPBEN	VREFEN	Res.	Res.	Res.	Res.
										rw	rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMPEN	OPAMPEN	LPTIM4EN	LPTIM3EN	LPTIM1EN	Res.	Res.	Res.	I2C3EN	LPUART1EN	SPI3EN	Res.	Res.	Res.	SYSCFGEN	Res.
rw	rw	rw	rw	rw				rw	rw	rw				rw	

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **RTCAPBEN**: RTC and TAMP APB clock enable
Set and cleared by software.
0: RTC and TAMP APB clock disabled
1: RTC and TAMP APB clock enabled

Bit 20 **VREFEN**: VREFBUF clock enable
Set and cleared by software.
0: VREFBUF clock disabled
1: VREFBUF clock enabled

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **COMPEN**: COMP clock enable
Set and cleared by software.
0: COMP clock disabled
1: COMP clock enabled

Bit 14 **OPAMPEN**: OPAMP clock enable
Set and cleared by software.
0: OPAMP clock disabled
1: OPAMP clock enabled

Bit 13 **LPTIM4EN**: LPTIM4 clock enable
Set and cleared by software.
0: LPTIM4 clock disabled
1: LPTIM4 clock enabled

Bit 12 **LPTIM3EN**: LPTIM3 clock enable
Set and cleared by software.
0: LPTIM3 clock disabled
1: LPTIM3 clock enabled

Bit 11 **LPTIM1EN**: LPTIM1 clock enable
Set and cleared by software.
0: LPTIM1 clock disabled
1: LPTIM1 clock enabled

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **I2C3EN**: I2C3 clock enable
Set and cleared by software.
0: I2C3 clock disabled
1: I2C3 clock enabled

Bit 6 **LPUART1EN**: LPUART1 clock enable
Set and cleared by software.
0: LPUART1 clock disabled
1: LPUART1 clock enabled

Bit 5 **SPI3EN**: SPI3 clock enable
Set and cleared by software.
0: SPI3 clock disabled
1: SPI3 clock enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **SYSCFGEN**: SYSCFG clock enable
Set and cleared by software.
0: SYSCFG clock disabled
1: SYSCFG clock enabled

Bit 0 Reserved, must be kept at reset value.

11.8.37 RCC AHB1 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB1SMENR)

Address offset: 0x0B0

Reset value: 0xFFFF FFFF

Access: no wait state, word, half-word and byte access

This register only configures the clock gating, not the clock source itself.

When a bit is set in Stop mode, the corresponding peripheral clock is enabled only when a peripheral (this one or another) requests AHB or APB clock (refer to *Section 11.4.20: Peripherals clock gating and autonomous mode*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAM1SMEN	DCACHE1SMEN	ICACHESMEN	BKPSRAMSMEN	Res.	Res.	Res.	GTZC1SMEN	Res.	Res.	Res.	Res.	Res.	DMA2DSMEN	RAMCFGSMEN	TSCSMEN
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.	CRCSMEN	Res.	Res.	Res.	FLASHSMEN	Res.	Res.	Res.	Res.	MDF1SMEN	FMACSMEN	CORDICSMEN	GPDMA1SMEN
			rw				rw					rw	rw	rw	rw

Bit 31 **SRAM1SMEN**: SRAM1 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: SRAM1 clocks disabled by the clock gating during Sleep and Stop modes
1: SRAM1 clocks enabled by the clock gating during Sleep and Stop modes

Bit 30 **DCACHE1SMEN**: DCACHE1 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: DCACHE1 clocks disabled by the clock gating during Sleep and Stop modes
1: DCACHE1 clocks enabled by the clock gating during Sleep and Stop modes

Bit 29 **ICACHESMEN**: 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 28 **BKPSRAMSMEN**: BKPSRAM clocks enable during Sleep and Stop modes
Set and cleared by software
0: BKPSRAM clocks disabled by the clock gating during Sleep and Stop modes
1: BKPSRAM clocks enabled by the clock gating during Sleep and Stop modes

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

Bit 24 **GTZC1SMEN**: GTZC1 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: GTZC1 clocks disabled by the clock gating during Sleep and Stop modes
1: GTZC1 clocks enabled by the clock gating during Sleep and Stop modes

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

Bit 18 **DMA2DSMEN**: DMA2D clocks enable during Sleep and Stop modes
Set and cleared by software.
0: DMA2D clocks disabled by the clock gating during Sleep and Stop modes
1: DMA2D clocks enabled by the clock gating during Sleep and Stop modes

Bit 17 **RAMCFGSMEN**: RAMCFG clocks enable during Sleep and Stop modes
Set and cleared by software.
0: RAMCFG clocks disabled by the clock gating during Sleep and Stop modes
1: RAMCFG clocks enabled by the clock gating during Sleep and Stop modes

Bit 16 **TSCSMEN**: TSC 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 11:9 Reserved, must be kept at reset value.

Bit 8 **FLASHSMEN**: FLASH clocks enable during Sleep and Stop modes
Set and cleared by software.
0: FLASH clocks disabled by the clock gating during Sleep and Stop modes
1: FLASH clocks enabled by the clock gating during Sleep and Stop modes

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

Bit 3 **MDF1SMEN**: MDF1 clocks enable during Sleep and Stop modes.
Set and cleared by software.
0: MDF1 clocks disabled by the clock gating during Sleep and Stop modes
1: MDF1 clocks enabled by the clock gating during Sleep and Stop modes
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bit 2 **FMACSMEN**: FMAC clocks enable during Sleep and Stop modes.
Set and cleared by software.
0: FMAC clocks disabled by the clock gating during Sleep and Stop modes
1: FMAC clocks enabled by the clock gating during Sleep and Stop modes

Bit 1 **CORDICSMEN**: CORDIC clocks enable during Sleep and Stop modes
Set and cleared by software during Sleep mode.
0: CORDIC clocks disabled by the clock gating during Sleep and Stop modes
1: CORDIC clocks enabled by the clock gating during Sleep and Stop modes

Bit 0 **GPDMA1SMEN**: GPDMA1 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: GPDMA1 clocks disabled by the clock gating during Sleep and Stop modes
1: GPDMA1 clocks enabled by the clock gating during Sleep and Stop modes
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

11.8.38 RCC AHB2 peripheral clocks enable in Sleep and Stop modes register 1 (RCC_AHB2SMENR1)

Address offset: 0x0B4

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word and byte access

This register only configures the clock gating, not the clock source itself.

When a bit is set in Stop mode, the corresponding peripheral clock is enabled only when a peripheral (this one or another) requests AHB or APB clock (refer to *Section 11.4.20: Peripherals clock gating and autonomous mode*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAM3SMEN	SRAM2SMEN	Res.	SDMMC2SMEN	SDMMC1SMEN	Res.	Res.	OTFDEC2SMEN	OTFDEC1SMEN	Res.	OCTOSPIMSMEN	SAESSMEN	PKASMEN	RNGSMEN	HASHSMEN	AESSMEN
rw	rw		rw	rw			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.	OTGSMEN	Res.	DCMI_PSSISMEN	Res.	ADC1SMEN	Res.	GPIOISMEN	GPIOHSMEN	GPIOGSMEN	GPIOFSMEN	GPIOESMEN	GPIODSMEN	GPIOCSMEN	GPIOBSMEN	GPIOASMEN
	rw		rw		rw		rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SRAM3SMEN**: SRAM3 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: SRAM3 clocks disabled by the clock gating during Sleep and Stop modes
1: SRAM3 clocks enabled by the clock gating during Sleep and Stop modes

Bit 30 **SRAM2SMEN**: SRAM2 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: SRAM2 clocks disabled by the clock gating during Sleep and Stop modes
1: SRAM2 clocks enabled by the clock gating during Sleep and Stop modes

Bit 29 Reserved, must be kept at reset value.

Bit 28 **SDMMC2SMEN**: SDMMC2 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: SDMMC2 clocks disabled by the clock gating during Sleep and Stop modes
1: SDMMC2 clocks enabled by the clock gating during Sleep and Stop modes

Bit 27 **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

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **OTFDEC2SMEN**: OTFDEC2 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: OTFDEC2 clocks disabled by the clock gating during Sleep and Stop modes
1: OTFDEC2 clocks enabled by the clock gating during Sleep and Stop modes

Bit 23 **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 22 Reserved, must be kept at reset value.

Bit 21 **OCTOSPIMSMEN**: OCTOSPIM clocks enable during Sleep and Stop modes
Set and cleared by software.
0: OCTOSPIM clocks disabled by the clock gating during Sleep and Stop modes
1: OCTOSPIM clocks enabled by the clock gating during Sleep and Stop modes

Bit 20 **SAESSMEN:** SAES accelerator clocks enable during Sleep and Stop modes
Set and cleared by software.
0: SAES clocks disabled by the clock gating during Sleep and Stop modes
1: SAES clocks enabled by the clock gating during Sleep and Stop modes

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 clock 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

Bit 15 Reserved, must be kept at reset value.

Bit 14 **OTGSMEN:** OTG_FS clocks enable during Sleep and Stop modes
Set and cleared by software.
0: OTG_FS clocks disabled by the clock gating during Sleep and Stop modes
1: OTG_FS clocks enabled by the clock gating during Sleep and Stop modes

Bit 13 Reserved, must be kept at reset value.

Bit 12 **DCMI_PSSISMEN:** DCMI and PSSI clocks enable during Sleep and Stop modes
Set and cleared by software.
0: DCMI and PSSI clocks disabled by the clock gating during Sleep and Stop modes
1: DCMI and PSSI clocks enabled by the clock gating during Sleep and Stop modes

Bit 11 Reserved, must be kept at reset value.

Bit 10 **ADC1SMEN:** ADC1 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: ADC1 clocks disabled by the clock gating during Sleep and Stop modes
1: ADC1 clocks enabled by the clock gating during Sleep and Stop modes

Bit 9 Reserved, must be kept at reset value.

Bit 8 **GPIOISMEN:** IO port I clocks enable during Sleep and Stop modes
Set and cleared by software.
0: IO port I clocks disabled by the clock gating during Sleep and Stop modes
1: IO port I clocks enabled by the clock gating during Sleep and Stop modes

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

11.8.39 RCC AHB2 peripheral clocks enable in Sleep and Stop modes register 2 (RCC_AHB2SMENR2)

Address offset: 0x0B8

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word and byte access

This register only configures the clock gating, not the clock source itself.

When a bit is set in Stop mode, the corresponding peripheral clock is enabled only when a peripheral (this one or another) requests AHB or APB clock (refer to *Section 11.4.20: Peripherals clock gating and autonomous 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.	OCTOSPI2SMEN	Res.	Res.	Res.	OCTOSPI1SMEN	Res.	Res.	Res.	FSMCSMEN
							rw				rw				rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **OCTOSPI2SMEN**: OCTOSPI2 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: OCTOSPI2 clocks disabled by the clock gating during Sleep and Stop modes
1: OCTOSPI2 clocks enabled by the clock gating during Sleep and Stop modes

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **OCTOSPI1SMEN**: OCTOSPI1 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 3:1 Reserved, must be kept at reset value.

Bit 0 **FSMCSMEN**: FSMC clocks enable during Sleep and Stop modes
Set and cleared by software.
0: FSMC clocks disabled by the clock gating during Sleep and Stop modes
1: FSMC clocks enabled by the clock gating during Sleep and Stop modes

11.8.40 RCC AHB3 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB3SMENR)

Address offset: 0x0BC

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word and byte access

This register only configures the clock gating, not the clock source itself.

When a bit is set in Stop mode, the corresponding peripheral clock is enabled only when a peripheral (this one or another) requests AHB or APB clock (refer to *Section 11.4.20: Peripherals clock gating and autonomous mode*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAM4SMEN	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.	GTZC2SMEN	Res.	ADF1SMEN	LPDMA1SMEN	Res.	Res.	DAC1SMEN	ADC4SMEN	Res.	Res.	PWRSMEN	Res.	LPGPIO1SMEN
			rw		rw	rw			rw	rw			rw		rw

Bit 31 **SRAM4SMEN**: SRAM4 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: SRAM4 clocks disabled by the clock gating during Sleep and Stop modes
1: SRAM4 clocks enabled by the clock gating during Sleep and Stop modes

Bits 30:13 Reserved, must be kept at reset value.

Bit 12 **GTZC2SMEN**: GTZC2 clock enable during Sleep and Stop modes
Set and cleared by software.
0: GTZC2 clock disabled by the clock gating during Sleep and Stop modes
1: GTZC2 clock enabled by the clock gating during Sleep and Stop modes

Bit 11 Reserved, must be kept at reset value.

Bit 10 **ADF1SMEN**: ADF1 clock enable during Sleep and Stop modes
Set and cleared by software.
0: ADF1 clock disabled by the clock gating during Sleep and Stop modes
1: ADF1 clock enabled by the clock gating during Sleep and Stop modes
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bit 9 **LPDMA1SMEN**: LPDMA1 clock enable during Sleep and Stop modes
Set and cleared by software.
0: LPDMA1 clock disabled by the clock gating during Sleep and Stop modes
1: LPDMA1 clock enabled by the clock gating during Sleep and Stop modes
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **DAC1SMEN**: DAC1 clock enable during Sleep and Stop modes
Set and cleared by software.
0: DAC1 clock disabled by the clock gating during Sleep and Stop modes
1: DAC1 clock enabled by the clock gating during Sleep and Stop modes
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bit 5 **ADC4SMEN**: ADC4 clock enable during Sleep and Stop modes
Set and cleared by software.
0: ADC4 clock disabled by the clock gating during Sleep and Stop modes
1: ADC4 clock enabled by the clock gating during Sleep and Stop modes
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **PWRSMEN**: PWR clock enable during Sleep and Stop modes
Set and cleared by software.
0: PWR clock disabled by the clock gating during Sleep and Stop modes
1: PWR clock enabled by the clock gating during Sleep and Stop modes

Bit 1 Reserved, must be kept at reset value.

Bit 0 **LPGPIO1SMEN**: LPGPIO1 enable during Sleep and Stop modes
Set and cleared by software.
0: LPGPIO1 clock disabled by the clock gating during Sleep and Stop modes
1: LPGPIO1 clock enabled by the clock gating during Sleep and Stop modes

11.8.41 RCC APB1 peripheral clocks enable in Sleep and Stop modes register 1 (RCC_APB1SMENR1)

Address offset: 0x0C4

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word and byte access

This register only configures the clock gating, not the clock source itself.

When a bit is set in Stop mode, the corresponding peripheral clock is enabled only when a peripheral (this one or another) requests AHB or APB clock (refer to *Section 11.4.20: Peripherals clock gating and autonomous mode*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRSSMEN	Res.	I2C2SMEN	I2C1SMEN	UART5SMEN	UART4SMEN	USART3SMEN	USART2SMEN	Res.
							rw		rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI2SMEN	Res.	Res.	WWDGSMEN	Res.	Res.	Res.	Res.	Res.	TIM7SMEN	TIM6SMEN	TIM5SMEN	TIM4SMEN	TIM3SMEN	TIM2SMEN
	rw			rw						rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 31:25 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 Reserved, must be kept at reset value.

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
Note: This bit must be set to allow the peripheral to wake up from 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
Note: This bit must be set to allow the peripheral to wake up from 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
Note: This bit must be set to allow the peripheral to wake up from 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
Note: This bit must be set to allow the peripheral to wake up from 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
Note: This bit must be set to allow the peripheral to wake up from 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
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bits 16:15 Reserved, must be kept at reset value.

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
Note: This bit must be set to allow the peripheral to wake up from 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

Bits 10:6 Reserved, must be kept at reset value.

Bit 5 **TIM7SMEN**: TIM7 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 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 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 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 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 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

11.8.42 RCC APB1 peripheral clocks enable in Sleep and Stop modes register 2 (RCC_APB1SMENR2)

Address offset: 0x0C8

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word and byte access

This register only configures the clock gating, not the clock source itself.

When a bit is set in Stop mode, the corresponding peripheral clock is enabled only when a peripheral (this one or another) requests AHB or APB clock (refer to *Section 11.4.20: Peripherals clock gating and autonomous 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.	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.	FDCAN1SMEN	Res.	Res.	Res.	LPTIM2SMEN	Res.	Res.	Res.	I2C4SMEN	Res.
						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

Bits 22:10 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 8:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2SMEN:** LPTIM2 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
Note: This bit must be set to allow the peripheral to wake up from 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
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bit 0 Reserved, must be kept at reset value.

11.8.43 RCC APB2 peripheral clocks enable in Sleep and Stop modes register (RCC_APB2SMENR)

Address offset: 0x0CC

Reset value: 0xFFFF FFFF

Access: word, half-word and byte access

This register only configures the clock gating, not the clock source itself.

When a bit is set in Stop mode, the corresponding peripheral clock is enabled only when a peripheral (this one or another) requests AHB or APB clock (refer to *Section 11.4.20: Peripherals clock gating and autonomous 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.	SAI2SMEN	SAI1SMEN	Res.	Res.	TIM17SMEN	TIM16SMEN	TIM15SMEN
									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.	Res.
	rw	rw	rw	rw											

Bits 31:23 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 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 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 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
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bit 13 T**IM8SMEN**: TIM8 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
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bit 11 **TIM1SMEN**: TIM1 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 10:0 Reserved, must be kept at reset value.

11.8.44 RCC APB3 peripheral clock enable in Sleep and Stop modes register (RCC_APB3SMENR)

Address offset: 0x0D0

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word and byte access

This register only configures the clock gating, not the clock source itself.

When a bit is set in Stop mode, the corresponding peripheral clock is enabled only when a peripheral (this one or another) requests AHB or APB clock (refer to *Section 11.4.20: Peripherals clock gating and autonomous 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.	RTCAPBSMEN	VREFSMEN	Res.	Res.	Res.	Res.
										rw	rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMPSMEN	OPAMPSMEN	LPTIM4SMEN	LPTIM3SMEN	LPTIM1SMEN	Res.	Res.	Res.	I2C3SMEN	LPUART1SMEN	SPI3SMEN	Res.	Res.	Res.	SYSCFGSMEN	Res.
rw	rw	rw	rw	rw				rw	rw	rw				rw	

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **RTCAPBSMEN**: RTC and TAMP APB clock enable during Sleep and Stop modes
Set and cleared by software.
0: RTC and TAMP APB clock disabled by the clock gating during Sleep and Stop modes
1: RTC and TAMP APB clock enabled by the clock gating during Sleep and Stop modes
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bit 20 **VREFSMEN**: VREFBUF clocks enable during Sleep and Stop modes
Set and cleared by software.
0: VREFBUF clocks disabled by the clock gating during Sleep and Stop modes
1: VREFBUF clocks enabled by the clock gating during Sleep and Stop modes

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **COMPSMEN**: COMP clocks enable during Sleep and Stop modes
Set and cleared by software.
0: COMP clocks disabled by the clock gating during Sleep and Stop modes
1: COMP clocks enabled by the clock gating during Sleep and Stop modes

Bit 14 **OPAMPSMEN**: OPAMP clocks enable during Sleep and Stop modes
Set and cleared by software.
0: OPAMP clocks disabled by the clock gating during Sleep and Stop modes
1: OPAMP clocks enabled by the clock gating during Sleep and Stop modes

Bit 13 **LPTIM4SMEN**: LPTIM4 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: LPTIM4 clocks disabled by the clock gating during Sleep and Stop modes
1: LPTIM4 clocks enabled by the clock gating during Sleep and Stop modes
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bit 12 **LPTIM3SMEN**: LPTIM3 clocks 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
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bit 11 **LPTIM1SMEN**: LPTIM1 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
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **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
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bit 6 **LPUART1SMEN**: LPUART1 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
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bit 5 **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
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **SYSCFGSMEN**: SYSCFG clocks enable during Sleep and Stop modes
Set and cleared by software.
0: SYSCFG clocks disabled by the clock gating during Sleep and Stop modes
1: SYSCFG clocks enabled by the clock gating during Sleep and Stop modes

Bit 0 Reserved, must be kept at reset value.

11.8.45 RCC SmartRun domain peripheral autonomous mode register (RCC_SRDAMR)

Address offset: 0x0D8

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
SRAM4AMEN	Res.	ADF1AMEN	LPDMA1AMEN	DAC1AMEN	LPGPIO1AMEN	ADC4AMEN	Res.	Res.	Res.	RTCAPBAMEN	VREFAMEN	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
COMPAMEN	OPAMPAMEN	LPTIM4AMEN	LPTIM3AMEN	LPTIM1AMEN	Res.	Res.	Res.	I2C3AMEN	LPUART1AMEN	SPI3AMEN	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw				rw	rw	rw					

Bit 31 **SRAM4AMEN**: SRAM4 autonomous mode enable in Stop 0,1,2 mode
Set and cleared by software.
0: SRAM4 autonomous mode disabled during Stop 0,1,2 mode
1: SRAM4 autonomous mode enabled during Stop 0,1,2 mode

Bit 30 Reserved, must be kept at reset value.

Bit 29 **ADF1AMEN**: ADF1 autonomous mode enable in Stop 0,1,2 mode
Set and cleared by software.
0: ADF1 autonomous mode disabled during Stop 0,1,2 mode
1: ADF1 autonomous mode enabled during Stop 0,1,2 mode
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bit 28 **LPDMA1AMEN**: LPDMA1 autonomous mode enable in Stop 0,1,2 mode
Set and cleared by software.
0: LPDMA1 autonomous mode disabled during Stop 0,1,2 mode
1: LPDMA1 autonomous mode enabled during Stop 0,1,2 mode
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bit 27 **DAC1AMEN**: DAC1 autonomous mode enable in Stop 0,1,2 mode
Set and cleared by software.
0: DAC1 autonomous mode disabled during Stop 0,1,2 mode
1: DAC1 autonomous mode enabled during Stop 0,1,2 mode
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bit 26 **LPGPIO1AMEN**: LPGPIO1 autonomous mode enable in Stop 0,1,2 mode
Set and cleared by software.
0: LPGPIO1 autonomous mode disabled during Stop 0,1,2 mode
1: LPGPIO1 autonomous mode enabled during Stop 0,1,2 mode

Bit 25 **ADC4AMEN**: ADC4 autonomous mode enable in Stop 0,1,2 mode
Set and cleared by software.
0: ADC4 autonomous mode disabled during Stop 0,1,2 mode
1: ADC4 autonomous mode enabled during Stop 0,1,2 mode
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bits 24:22 Reserved, must be kept at reset value.

Bit 21 **RTCAPBAMEN**: RTC and TAMP autonomous mode enable in Stop 0,1,2 mode
Set and cleared by software.
0: RTC and TAMP autonomous mode disabled during Stop 0,1,2 mode
1: RTC and TAMP autonomous mode enabled during Stop 0,1,2 mode
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bit 20 **VREFAMEN**: VREFBUF autonomous mode enable in Stop 0,1,2 mode
Set and cleared by software.
0: VREFBUF autonomous mode disabled during Stop 0,1,2 mode
1: VREFBUF autonomous mode enabled during Stop 0,1,2 mode

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **COMPAMEN**: COMP autonomous mode enable in Stop 0,1,2 mode
Set and cleared by software.
0: COMP autonomous mode disabled during Stop 0,1,2 mode
1: COMP autonomous mode enabled during Stop 0,1,2 mode

Bit 14 **OPAMPAMEN**: OPAMP autonomous mode enable in Stop 0,1,2 mode
Set and cleared by software.
0: OPAMP autonomous mode disabled during Stop 0,1,2 mode
1: OPAMP autonomous mode enabled during Stop 0,1,2 mode

Bit 13 **LPTIM4AMEN**: LPTIM4 autonomous mode enable in Stop 0,1,2 mode
Set and cleared by software.
0: LPTIM4 autonomous mode disabled during Stop 0,1,2 mode
1: LPTIM4 autonomous mode enabled during Stop 0,1,2 mode
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bit 12 **LPTIM3AMEN**: LPTIM3 autonomous mode enable in Stop 0,1,2 mode
Set and cleared by software.
0: LPTIM3 autonomous mode disabled during Stop 0,1,2 mode
1: LPTIM3 autonomous mode enabled during Stop 0,1,2 mode
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bit 11 **LPTIM1AMEN**: LPTIM1 autonomous mode enable in Stop 0,1,2 mode
Set and cleared by software.
0: LPTIM1 autonomous mode disabled during Stop 0,1,2 mode
1: LPTIM1 autonomous mode enabled during Stop 0,1,2 mode
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **I2C3AMEN**: I2C3 autonomous mode enable in Stop 0,1,2 mode
Set and cleared by software.
0: I2C3 autonomous mode disabled during Stop 0,1,2 mode
1: I2C3 autonomous mode enabled during Stop 0,1,2 mode
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bit 6 **LPUART1AMEN**: LPUART1 autonomous mode enable in Stop 0,1, 2 mode
Set and cleared by software.
0: LPUART1 autonomous mode disabled during Stop 0,1,2 mode
1: LPUART1 autonomous mode enabled during Stop 0,1,2 mode
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bit 5 **SPI3AMEN**: SPI3 autonomous mode enable in Stop 0,1, 2 mode
Set and cleared by software.
0: SPI3 autonomous mode disabled during Stop 0,1,2 mode
1: SPI3 autonomous mode enabled during Stop 0,1,2 mode
Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bits 4:0 Reserved, must be kept at reset value.

11.8.46 RCC peripherals independent clock configuration register 1 (RCC_CCIPR1)

Address offset: 0x0E0

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
TIMICSEL[2:0]			Res.	ICLKSEL[1:0]		FDCAN1SEL[1:0]		SYSTICKSEL [1:0]		SPI1SEL[1:0]		LPTIM2SEL[1:0]		SPI2SEL[1:0]	
rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2C4SEL[1:0]		I2C2SEL[1:0]		I2C1SEL[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 31:29 **TIMICSEL[2:0]**: Clocks sources for TIM16,TIM17 and LPTIM2 internal input capture
When the TIMICSEL2 bit is set, the TIM16, TIM17 and LPTIM2 internal input capture can be connected either to HSI/256, MSI/4 or MSI/1024. Depending on TIMICSEL[1:0] value, MSI is either MSIK or MSIS.
When TIMICSEL2 is cleared, the HSI, MSIK and MSIS clock sources cannot be selected as TIM16, TIM17 or LPTIM2 internal input capture.
0xx: HSI, MSIK and MSIS dividers disabled
100: HSI/256, MSIS/1024 and MSIS/4 generated and can be selected by TIM16, TIM17 and LPTIM2 as internal input capture
101: HSI/256, MSIS/1024 and MSIK/4 generated and can be selected by TIM16, TIM17 and LPTIM2 as internal input capture
110: HSI/256, MSIK/1024 and MSIS/4 generated and can be selected by TIM16, TIM17 and LPTIM2 as internal input capture
111: HSI/256, MSIK/1024 and MSIK/4 generated and can be selected by TIM16, TIM17 and LPTIM2 as internal input capture
Note: The clock division must be disabled (TIMICSEL configured to 0xx) before selecting or changing a clock sources division.

Bit 28 Reserved, must be kept at reset value.

Bits 27:26 **ICLKSEL[1:0]**: intermediate clock source selection
These bits are used to select the clock source used by OTG_FS and SDMMC.
00: HSI48 clock selected
01: PLL2 “Q” (pll2_q_ck) selected
10: PLL1 “Q” (pll1_q_ck) selected
11: MSIK clock selected

Bits 25:24 **FDCAN1SEL[1:0]**: FDCAN1 kernel clock source selection
These bits are used to select the FDCAN1 kernel clock source.
00: HSE clock selected
01: PLL1“Q” (pll1_q_ck) selected
10: PLL2 “P” (pll2_p_ck) selected
11: reserved

Bits 23:22 **SYSTICKSEL[1:0]**: SysTick clock source selection
These bits are used to select the SysTick clock source.
00: HCLK/8 selected
01: LSI selected
10: LSE selected
11: reserved

Note: When LSE or LSI is selected, the AHB frequency must be at least four times higher than the LSI or LSE frequency. In addition, a jitter up to one HCLK cycle is introduced, due to the LSE or LSI sampling with HCLK in the SysTick circuitry.

Bits 21:20 **SPI1SEL[1:0]**: SPI1 kernel clock source selection
These bits are used to select the SPI1 kernel clock source.
00: PCLK2 selected
01: SYSCLK selected
10: HSI16 selected
11: MSIK selected

Note: The SPI1 is functional in Stop 0 and Stop 1 mode only when the kernel clock is HSI16 or MSIK.

Bits 19:18 **LPTIM2SEL[1:0]**: Low-power timer 2 kernel clock source selection
These bits are used to select the LPTIM2 kernel clock source.
00: PCLK1 selected
01: LSI selected
10: HSI16 selected
11: LSE selected

Note: The LPTIM2 is functional in Stop 0 and Stop 1 mode only when the kernel clock is LSI, LSE or HSI16 if HSIKERON = 1.

Bits 17:16 **SPI2SEL[1:0]**: SPI2 kernel clock source selection
These bits are used to select the SPI2 kernel clock source.
00: PCLK1 selected
01: SYSCLK selected
10: HSI16 selected
11: MSIK selected

Note: The SPI2 is functional in Stop 0 and Stop 1 mode only when the kernel clock is HSI16 or MSIK.

Bits 15:14 **I2C4SEL[1:0]**: I2C4 kernel clock source selection
These bits are used to select the I2C4 kernel clock source.
00: PCLK1 selected
01: SYSCLK selected
10: HSI16 selected
11: MSIK selected
Note: The I2C4 is functional in Stop 0 and Stop 1 mode only when the kernel clock is HSI16 or MSIK.

Bits 13:12 **I2C2SEL[1:0]**: I2C2 kernel clock source selection
These bits are used to select the I2C2 kernel clock source.
00: PCLK1 selected
01: SYSCLK selected
10: HSI16 selected
11: MSIK selected
Note: The I2C2 is functional in Stop 0 and Stop 1 mode only when the kernel clock is HSI16 or MSIK.

Bits 11:10 **I2C1SEL[1:0]**: I2C1 kernel clock source selection
These bits are used to select the I2C1 kernel clock source.
00: PCLK1 selected
01: SYSCLK selected
10: HSI16 selected
11: MSIK selected
Note: The I2C1 is functional in Stop 0 and Stop 1 mode only when the kernel clock is HSI16 or MSIK.

Bits 9:8 **UART5SEL[1:0]**: UART5 kernel clock source selection
These bits are used to select the UART5 kernel clock source.
00: PCLK1 selected
01: SYSCLK selected
10: HSI16 selected
11: LSE selected
Note: The UART5 is functional in Stop 0 and Stop 1 mode only when the kernel clock is HSI16 or LSE.

Bits 7:6 **UART4SEL[1:0]**: UART4 kernel clock source selection
This bits are used to select the UART4 kernel clock source.
00: PCLK1 selected
01: SYSCLK selected
10: HSI16 selected
11: LSE selected
Note: The UART4 is functional in Stop 0 and Stop 1 mode only when the kernel clock is HSI16 or LSE.

Bits 5:4 **USART3SEL[1:0]**: USART3 kernel clock source selection
This bits are used to select the USART3 kernel clock source.
00: PCLK1 selected
01: SYSCLK selected
10: HSI16 selected
11: LSE selected
Note: The USART3 is functional in Stop 0 and Stop 1 mode only when the kernel clock is HSI16 or LSE.

Bits 3:2 **USART2SEL[1:0]**: USART2 kernel clock source selection
This bits are used to select the USART2 kernel clock source.
00: PCLK1 selected
01: SYSCLK selected
10: HSI16 selected
11: LSE selected
Note: The USART2 is functional in Stop 0 and Stop 1 mode only when the kernel clock is HSI16 or LSE.

Bits 1:0 **USART1SEL[1:0]**: USART1 kernel clock source selection
This bits are used to select the USART1 kernel clock source.
00: PCLK2 selected
01: SYSCLK selected
10: HSI16 selected
11: LSE selected
Note: The USART1 is functional in Stop 0 and Stop 1 mode only when the kernel clock is HSI16 or LSE.

11.8.47 RCC peripherals independent clock configuration register 2 (RCC_CCIPR2)

Address offset: 0x0E4

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.	OCTOSPISEL [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	RNGSEL[1:0]		SAESSEL	SAI2SEL[2:0]			SAI1SEL[2:0]			Res.	Res.	MDF1SEL[2:0]		
	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 **OCTOSPISEL[1:0]**: OCTOSPI1 and OCTOSPI2 kernel clock source selection
These bits are used to select the OCTOSPI1 and OCTOSPI2 kernel clock source.
00: SYSCLK selected
01: MSIK selected
10: PLL1 “Q” (pll1_q_ck) selected, can be up to 200 MHz
11: PLL2 “Q” (pll2_q_ck) selected, can be up to 200 MHz

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **SDMMCSEL**: SDMMC1 and SDMMC2 kernel clock source selection
This bit is used to select the SDMMC kernel clock source. It is recommended to change this bit only after reset and before enabling the SDMMC.
0: ICLK clock selected
1: PLL1 “P” (pll1_p_ck) selected, in case higher than 48 MHz is needed (for SDR50 mode)

Bits 13:12 **RNGSEL[1:0]**: RNGSEL kernel clock source selection
These bits are used to select the RNG kernel clock source.
00: HSI48 selected
01: HSI48 / 2 selected, can be used in range 4
10: HSI16 selected
11: reserved

Bit 11 **SAESSEL**: SAES kernel clock source selection
This bit is used to select the SAES kernel clock source.
0: SHSI selected
1: SHSI / 2 selected, can be used in range 4

Bits 10:8 **SAI2SEL[2:0]**: SAI2 kernel clock source selection
These bits are used to select the SAI2 kernel clock source.
000: PLL2 “P” (pll2_p_ck) selected
001: PLL3 “P” (pll3_p_ck) selected
010: PLL1 “P” (pll1_p_ck) selected
011: input pin AUDIOCLK selected
100: HSI16 clock selected
others: reserved
Note: If the selected clock is the external clock and this clock is stopped, a switch to another clock is impossible.

Bits 7:5 **SAI1SEL[2:0]**: SAI1 kernel clock source selection
These bits are used to select the SAI1 kernel clock source.
000: PLL2 “P” (pll2_p_ck) selected
001: PLL3 “P” (pll3_p_ck) selected
010: PLL1 “P” (pll1_p_ck) selected
011: input pin AUDIOCLK selected
100: HSI16 clock selected
others: reserved
Note: If the selected clock is the external clock and this clock is stopped, a switch to another clock is impossible.

Bits 4:3 Reserved, must be kept at reset value.

Bits 2:0 **MDF1SEL[2:0]**: MDF1 kernel clock source selection
These bits are used to select the MDF1 kernel clock source.
000: HCLK selected
001: PLL1 “P” (pll1_p_ck) selected
010: PLL3 “Q” (pll3_q_ck) selected
011: input pin AUDIOCLK selected
100: MSIK clock selected
others: reserved

11.8.48 RCC peripherals independent clock configuration register 3 (RCC_CCIPR3)

Address offset: 0x0E8

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.	ADF1SEL[2:0]		
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAC1SEL	ADCDACSEL[2:0]			LPTIM1SEL[1:0]		LPTIM34SEL[1:0]		I2C3SEL[1:0]		Res.	SPI3SEL[1:0]		LPUART1SEL[2:0]		
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.

Bits 18:16 **ADF1SEL[2:0]**: ADF1 kernel clock source selection

These bits are used to select the ADF1 kernel clock source.
000: HCLK selected
001: PLL1 “P” (pll1_p_ck) selected
010: PLL3 “Q” (pll3_q_ck) selected
011: input pin AUDIOCLK selected
100: MSIK clock selected
others: reserved

Note: The ADF1 is functional in Stop 0, Stop 1 and Stop 2 modes only when the kernel clock is AUDIOCLK or MSIK.

Bit 15 **DAC1SEL**: DAC1 sample and hold clock source selection

This bit is used to select the DAC1 sample and hold clock source.
0: LSE selected
1: LSI selected

Bits 14:12 **ADCDACSEL[2:0]**: ADC1, ADC4 and DAC1 kernel clock source selection

These bits are used to select the ADC1, ADC4 and DAC1 kernel clock source.
000: HCLK clock selected
001: SYSCLK selected
010: PLL2 “R” (pll2_r_ck) selected
011: HSE clock selected
100: HSI16 clock selected
101: MSIK clock selected
others: reserved

Note: The ADC1, ADC4 and DAC1 are functional in Stop 0, Stop 1 and Stop 2 modes only when the kernel clock is HSI16 or MSIK (only ADC4 and DAC1 are functional in Stop 2 mode).

Bits 11:10 **LPTIM1SEL[1:0]**: LPTIM1 kernel clock source selection
These bits are used to select the LPTIM1 kernel clock source.
00: MSIK clock selected
01: LSI selected
10: HSI16 selected
11: LSE selected
Note: The LPTIM1 is functional in Stop 0, Stop 1 and Stop 2 modes only when the kernel clock is LSI, LSE, HSI16 with HSIKERON = 1 or MSIK with MSIKERON = 1.

Bits 9:8 **LPTIM34SEL[1:0]**: LPTIM3 and LPTIM4 kernel clock source selection
These bits are used to select the LPTIM3 and LPTIM4 kernel clock source.
00: MSIK clock selected
01: LSI selected
10: HSI selected
11: LSE selected
Note: The LPTIM3 and LPTIM4 are functional in Stop 0, Stop 1 and Stop 2 modes only when the kernel clock is LSI, LSE, HSI16 with HSIKERON = 1 or MSIK with MSIKERON = 1.

Bits 7:6 **I2C3SEL[1:0]**: I2C3 kernel clock source selection
These bits are used to select the I2C3 kernel clock source.
00: PCLK3 selected
01: SYSCLK selected
10: HSI16 selected
11: MSIK selected
Note: The I2C3 is functional in Stop 0, Stop 1 and Stop 2 modes only when the kernel clock is HSI16 or MSIK.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **SPI3SEL[1:0]**: SPI3 kernel clock source selection
These bits are used to select the SPI3 kernel clock source.
00: PCLK3 selected
01: SYSCLK selected
10: HSI16 selected
11: MSIK selected
Note: The SPI3 is functional in Stop 0, Stop 1 and Stop 2 modes only when the kernel clock is HSI16 or MSIK.

Bits 2:0 **LPUART1SEL[2:0]**: LPUART1 kernel clock source selection
These bits are used to select the LPUART1 kernel clock source.
000: PCLK3 selected
001: SYSCLK selected
010: HSI16 selected
011: LSE selected
100: MSIK selected
others: reserved
Note: The LPUART1 is functional in Stop 0, Stop 1 and Stop 2 modes only when the kernel clock is HSI16, LSE or MSIK.

11.8.49 RCC Backup domain control register (RCC_BDCR)

Address offset: 0x00F0

Backup domain 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 this register are outside of the core domain. As a result, after reset, these bits are write-protected and the DBP bit in the PWR Backup domain control register 1 (PWR_BDCR1) must be set before these can be modified (see Section 10: Power control (PWR) for further information). These bits (except LSCOSEL, LSCOEN and BDRST) are only reset after a Backup domain reset (see Section 11.3.3: 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.	LSIPREDIV	LSIRDY	LSION	LSCOSEL	LSCOEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BDRST
			rw	rw	rw	rw	rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	Res.	Res.	LSEGFON	LSESYSRDY	Res.	RTCSEL[1:0]		LSESYSEN	LSECSSD	LSECSSON	LSEDRV[1:0]		LSEBYP	LSERDY	LSEON
rw			rw	r		rw	rw	rw	r	rw	rw	rw	rw	r	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **LSIPREDIV**: Low-speed clock divider configuration

Set and cleared by software to enable the LSI division. This bit can be written only when the LSI is disabled (LSION = 0 and LSIRDY = 0). If the LSI was previously enabled, it is necessary to wait for at least 60 µs after clearing LSION bit (synchronization time for LSI to be really disabled), before writing LSIPREDIV. The LSIPREDIV cannot be changed if the LSI is used by the IWDG or by the RTC.

0: LSI not divided

1: LSI divided by 128

Bit 27 **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 three internal low-speed oscillator clock cycles. This bit is set when the LSI is used by IWDG or RTC, even if LSION = 0.

0: LSI oscillator not ready

1: LSI oscillator ready

Bit 26 **LSION**: LSI oscillator enable

Set and cleared by software. The LSI oscillator is disabled 60 µs maximum after the LSION bit is cleared.

0: LSI oscillator OFF

1: LSI oscillator ON

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

Bit 15 **RTCEN**: RTC and TAMP clock enable
Set and cleared by software.
0: RTC and TAMP clock disabled
1: RTC and TAMP clock enabled

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

Bit 12 **LSEGFON**: LSE clock glitch filter enable
Set and cleared by hardware to enable the LSE glitch filter. This bit can be written only when the LSE is disabled (LSEON = 0 and LSERDY = 0)
0: LSE glitch filter disabled
1: LSE glitch filter enabled

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 six 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 and TAMP clock source selection
Set by software to select the clock source for the RTC and TAMP . Once the RTC and TAMP 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, which can be used by any peripheral when its source clock is the LSE, or at system level if one of LSCOSEL, MCO, or MSI PLL mode is needed.
0: LSE can be used only for RTC, TAMP, and CSS on LSE.
1: LSE can be used by any other peripheral or function.

Bit 6 **LSECSSD**: CSS on LSE failure Detection

Set by hardware to indicate when a failure is detected by the CCS on the external 32 kHz oscillator (LSE).

0: No failure detected on LSE

1: Failure detected on LSE

Bit 5 **LSECSSON**: CSS on LSE enable

Set by software to enable the CSS on LSE. 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 OFF

1: CSS on LSE ON

Bits 4:3 **LSEDRV[1:0]**: LSE oscillator drive capability

Set by software to modulate the drive capability of the LSE oscillator. This field can be written only when the external 32 kHz oscillator is disabled (LSEON = 0 and LSERDY = 0).

00: ‘Xtal mode’ lower driving capability

01: ‘Xtal mode’ medium-low driving capability

10: ‘Xtal mode’ medium-high driving capability

11: ‘Xtal mode’ higher driving capability

Note: The oscillator is in ‘Xtal mode’ when it is not in bypass mode.

Bit 2 **LSEBYP**: LSE oscillator bypass

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 six 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

11.8.50 RCC control/status register (RCC_CSR)

Address offset: 0x0F4

Reset value: 0x0C00 4400

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	IWDGRSTF	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
MSISSRANGE[3:0]				MSIKSRANGE[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								

Bit 31 **LPWRRSTF**: Low-power reset flag

Set by hardware when a reset occurs due to Stop, Standby or Shutdown mode entry, whereas the corresponding NRST_STOP, NRST_STBY or NRST_SHDW option bit is cleared.

Cleared by writing to the RMVF bit.

0: No illegal low-power mode reset occurred

1: Illegal low-power mode reset occurred

Bit 30 **WWDGRSTF**: Window watchdog reset flag

Set by hardware when a window watchdog reset occurs.

Cleared by writing to the RMVF bit.

0: No window watchdog reset occurred

1: Window watchdog reset occurred

Bit 29 **IWDGRSTF**: Independent 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**: NRST 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 22:16 Reserved, must be kept at reset value.

Bits 15:12 **MSISSRANGE[3:0]**:MSIS range after Standby mode
Set by software to chose the MSIS frequency at startup. This range is used after exiting Standby mode until MSIRGSEL is set. After a NRST pin or a power-on reset or when exiting Shutdown mode, the range is always 4 MHz. MSISSRANGE can be written only when MSIRGSEL = 1.
0100: range 4 around 4M Hz (reset value)
0101: range 5 around 2 MHz
0110: range 6 around 1.33 MHz
0111: range 7 around 1 MHz
1000: range 8 around 3.072 MHz
others: reserved
Note: Changing the MSISSRANGE does not change the current MSIS frequency.

Bits 11:8 **MSIKSRANGE[3:0]**:MSIK range after Standby mode
Set by software to chose the MSIK frequency at startup. This range is used after exiting Standby mode until MSIRGSEL is set. After a NRST pin or a power-on reset or when exiting Shutdown mode, the range is always 4 MHz. MSIKSRANGE can be written only when MSIRGSEL = 1.
0100: range 4 around 4M Hz (reset value)
0101: range 5 around 2 MHz
0110: range 6 around 1.33 MHz
0111: range 7 around 1 MHz
1000: range 8 around 3.072 MHz
others: reserved
Note: Changing the MSIKSRANGE does not change the current MSIK frequency.

Bits 7:0 Reserved, must be kept at reset value.

11.8.51 RCC secure configuration register (RCC_SECCFGR)

Address offset: 0x110

Reset value: 0x0000 0000

Access: no wait state; word, half-word and byte access

When the system is secure (TZEN = 1), this register can be written only by a secure privileged access if SPRIV = 1, and by a secure privileged or unprivileged access if SPRIV = 0. A non-secure write access generates an illegal access event and data is not written. This register can be read by secure or non-secure, privilege or unprivileged access.

When the system is not secure (TZEN = 0), this register is read as 0 and the register write is ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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	ICLKSEC	PLL3SEC	PLL2SEC	PLL1SEC	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 **ICLKSEC**: intermediate clock source selection security
Set and reset by software.
0: non secure
1: secure

Bit 9 **PLL3SEC**: PLL3 clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

Bit 8 **PLL2SEC**: PLL2 clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

Bit 7 **PLL1SEC**: PLL1 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**: HSI clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

11.8.52 RCC privilege configuration register (RCC_PRIVCFGR)

Address offset: 0x114

Reset value: 0x0000 0000

Access: no wait state; word, half-word and byte access

This register can be written only by a privileged access. It can be read by privileged or unprivileged 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.	NSPRIV	SPRIV
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **NSPRIV**: RCC non-secure functions privilege configuration

Set and reset by software. This bit can be written only by privileged access, secure or non-secure.

0: Read and write to RCC non-secure functions can be done by privileged or unprivileged access.

1: Read and write to RCC non-secure functions can be done by privileged access only.

Bit 0 **SPRIV**: RCC secure functions privilege configuration

Set and reset by software. This bit can be written only by a secure privileged access.

0: Read and write to RCC secure functions can be done by privileged or unprivileged access.

1: Read and write to RCC secure functions can be done by privileged access only.

11.8.53 RCC register map

Table 108. RCC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RCC_CR	Res.	Res.	PLL3RDY	PLL3ON	PLL2RDY	PLL2ON	PLL1RDY	PLL1ON	Res.	Res.	Res.	HSEEXT	CSSON	HSEBYP	HSERDY	HSEON	SHSIRDY	SHSION	HSI48RDY	HSI48ON	Res.	HSIRDY	HSIKERON	HSION	MSIPLLFAST	MSIPLLSEL	MSIKRDY	MSIKON	MSIPLLEN	MSISRDY	MSIKERON	MSISON
	Reset value			0	0	0	0	0	0				0	0	0	0	0	0	0	0	0		0	0	0	0	0	1	1	0	1	0	1
0x004	Reserved	Reserved																															
0x008	RCC_ICSCR1	MSISRANGE[3:0]				MSIKRANGE[3:0]				MSIRGSEL	MSIBIAS	Res.	Res.	MSICAL0[4:0]					MSICAL1[4:0]					MSICAL2[4:0]					MSICAL3[4:0]				
	Reset value	0	1	0	0	0	1	0	0	0	0			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x00C	RCC_ICSCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSITRIM0[4:0]					MSITRIM1[4:0]					MSITRIM2[4:0]					MSITRIM3[4:0]				
	Reset value													1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0
0x010	RCC_ICSCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSITRIM[4:0]					Res.	Res.	Res.	Res.	HSICAL[11:0]											
	Reset value												1	0	0	0	0					X	X	X	X	X	X	X	X	X	X	X	X
0x014	RCC_CRRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSI48CAL[8:0]								
	Reset value																								X	X	X	X	X	X	X	X	X
0x018	Reserved	Reserved																															
0x01C	RCC_CFGR1	Res.	MCOPRE[2:0]			MCOSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOPKERWUCK	STOPWUCK	SWS[1:0]		SW[1:0]	
	Reset value		0	0	0	0	0	0	0																			0	0	0	0	0	0
0x020	RCC_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB2DIS	APB1DIS	AHB2DIS2	AHB2DIS1	AHB1DIS	Res.	Res.	Res.	Res.	Res.	PPRE2[2:0]			Res.	PPRE1[2:0]			HPRE[3:0]			
	Reset value												0	0	0	0	0						0	0	0		0	0	0	0	0	0	0

Table 108. RCC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x024	RCC_CFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB3DIS	AHB3DIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PPRE3[2:0]			Res.	Res.	Res.	Res.
	Reset value															0	0										0	0	0				
0x028	RCC_ PLL1CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL1REN	PLL1QEN	PLL1PEN	PLL1MBOOST[3:0]				PLL1M[3:0]				Res.	Res.	Res.	PLL1FRACEN	PLL1RGE[1:0]		PLL1SRC[1:0]	
	Reset value														0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0
0x02C	RCC_ PLL2CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL2REN	PLL2QEN	PLL2PEN	Res.	Res.	Res.	Res.	PLL2M[3:0]				Res.	Res.	Res.	PLL2FRACEN	PLL2RGE[1:0]		PLL2SRC[1:0]	
	Reset value														0	0	0					0	0	0	0				0	0	0	0	0
0x030	RCC_ PLL3CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL3REN	PLL3QEN	PLL3PEN	Res.	Res.	Res.	Res.	PLL3M[3:0]				Res.	Res.	Res.	PLL3FRACEN	PLL3RGE[1:0]		PLL3SRC[1:0]	
	Reset value														0	0	0					0	0	0	0				0	0	0	0	0
0x034	RCC_ PLL1DIVR	Res.	PLL1R[6:0]							Res.	PLL1Q[6:0]							PLL1P[6:0]							PLL1N[8:0]								
	Reset value		0	0	1	0	0	0	1		0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
0x038	RCC_ PLL1FRACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL1FRACN[12:0]													Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x03C	RCC_ PLL2DIVR	Res.	PLL2R[6:0]							Res.	PLL2Q[6:0]							PLL2P[6:0]							PLL2N[8:0]								
	Reset value		0	0	1	0	0	0	1		0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
0x040	RCC_ PLL2FRACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL2FRACN[12:0]													Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x044	RCC_ PLL3DIVR	Res.	PLL3R[6:0]							Res.	PLL3Q[6:0]							PLL3P[6:0]							PLL3N[8:0]								
	Reset value		0	0	1	0	0	0	1		0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
0x048	RCC_ PLL3FRACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL3FRACN[12:0]													Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x04C	Reserved	Reserved																															
0x050	RCC_CIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SHSIRDYIE	MSIKRDYIE	Res.	Res.	PLL3RDYIE	PLL2RDYIE	PLL1RDYIE	HSI48RDYIE	HSERDYIE	HSIRDYIE	MSISRDYIE	LSERDYIE	LSIRDYIE
	Reset value																				0	0			0	0	0	0	0	0	0	0	0

Table 108. RCC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x054	RCC_CIFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SHSIRDYF	MSIKRDYF	CSSF	Res.	PLL3RDYF	PLL2RDYF	PLL1RDYF	HSI48RDYF	HSERDYF	HSIRDYF	MSISRDYF	LSERDYF	LSIRDYF
	Reset value																				0	0	0		0	0	0	0	0	0	0	0	0
0x058	RCC_CICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SHSIRDYC	MSIKRDYC	CSSC	Res.	PLL3RDYC	PLL2RDYC	PLL1RDYC	HSI48RDYC	HSERDYC	HSIRDYC	MSISRDYC	LSERDYC	LSIRDYC
	Reset value																				0	0	0		0	0	0	0	0	0	0	0	0
0x05C	Reserved	Reserved																															
0x060	RCC_ AHB1RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMA2DRST	RAMCFGRST	TSCRST	Res.	Res.	Res.	CRCRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDF1RST	FMACRST	CORDICRST	GPDMA1RST
	Reset value														0	0	0				0									0	0	0	0
0x064	RCC_ AHB2RSTR1	Res.	Res.	Res.	SDMMC2RST	SDMMC1RST	Res.	Res.	OTFDEC2RST	OTFDEC1RST	Res.	OCTOSPIMRST	SAESRST	PKARST	RNGRST	HASHRST	AESRST	Res.	OTGRST	Res.	DCMI_PSSIRST	Res.	ADC1RST	Res.	GPIOIRST	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
	Reset value				0	0			0	0		0	0	0	0	0	0		0		0		0		0	0	0	0	0	0	0	0	0
0x068	RCC_ AHB2RSTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCTOSPI2RST	Res.	Res.	Res.	OCTOSPI1RST	Res.	Res.	Res.	FSMCRST
	Reset value																								0				0				0
0x06C	RCC_ AHB3RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1RST	LPDMA1RST	Res.	Res.	DAC1RST	ADC4RST	Res.	Res.	Res.	Res.	Res.	LPGPIO1RST
	Reset value																						0	0			0	0						0
0x070	Reserved	Reserved																															
0x074	RCC_ APB1RSTR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRSRST	Res.	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	Res.	Res.	SPI2RST	Res.	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
0x078	RCC_ APB1RSTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCAN1RST	Res.	Res.	Res.	Res.	LPTIM2RST	Res.	Res.	Res.	I2C4RST	Res.
	Reset value									0														0					0				0	
0x07C	RCC_ APB2RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2RST	SAI1RST	Res.	Res.	TIM17RST	TIM16RST	TIM15RST	Res.	USART1RST	TIM8RST	SPI1RST	TIM1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value										0	0			0	0	0		0	0	0	0													

Table 108. RCC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x080	RCC_ APB3RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VREFRST	Res.	Res.	Res.	Res.	COMPRST	OPAMPRST	LPTIM4RST	LPTIM3RST	LPTIM1RST	Res.	Res.	Res.	I2C3RST	LPUART1RST	SPI3RST	Res.	Res.	Res.	SYSCFGRST	Res.
	Reset value												0						0	0	0	0				0	0	0				0	
0x084	Reserved	Reserved																															
0x088	RCC_ AHB1ENR	SRAM1EN	DCACHEEN		BKPSRAMEN				GTZC1EN							RAMCFGEN	TSCEN				CRCEN				FLASHEN				MDF1EN	FMACEN	CORDICEN	GPDMA1EN
	Reset value	1	1		1				0						0	0	0				0				1				0	0	0	0
0x08C	RCC_ AHB2ENR1	SRAM3EN	SRAM2EN		SDMMC2EN	SDMMC1EN			OTFDEC2EN	OTFDEC1EN		OCTOSPIMEN	SAESEN	PKAEN	RNGEN	HASHEN	AESEN		OTGEN		DCMI_PSSIEN		ADC1EN		GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
	Reset value	1	1		0	0			0	0		0	0	0	0	0	0		0		0		0		0	0	0	0	0	0	0	0	0
0x090	RCC_ AHB2ENR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCTOSPI2EN	Res.	Res.	Res.	OCTOSPI1EN	Res.	Res.	Res.	FSMCEN
	Reset value																								0				0				0
0x094	RCC_ AHB3ENR	SRAM4EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GTZC2EN	Res.	ADF1EN	LPDMA1EN	Res.	Res.	DAC1EN	ADC4EN	Res.	Res.	PWREN	Res.	Res.	LPGPIO1EN
	Reset value	1																			0		0	0			0	0			0		0
0x098	Reserved	Reserved																															
0x09C	RCC_ APB1ENR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRSEN	Res.	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Res.	Res.	SPI2EN	Res.	Res.	WWDGEN	Res.	Res.	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
0x0A0	RCC_ APB1ENR2	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res	UCPD1EN	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCAN1EN	Res.	Res.	Res.	Res.	LPTIM2EN	Res.	Res.	Res.	I2C4EN	Res.
	Reset value									0														0					0				0	
0x0A4	RCC_ APB2ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2EN	SAI1EN	Res.	Res.	TIM17EN	TIM16EN	TIM15EN	Res.	USART1EN	TIM8EN	SPI1EN	TIM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value										0	0			0	0	0		0	0	0	0												
0x0A8	RCC_ APB3ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBEN	VREFEN	Res.	Res.	Res.	Res.	COMPEN	OPAMPEN	LPTIM4EN	LPTIM3EN	LPTIM1EN	Res.	Res.	Res.	Res.	I2C3EN	LPUART1EN	SPI3EN	Res.	Res.	Res.	SYSCFGEN	Res.
	Reset value											0	0						0	0	0	0					0	0	0				0	
0x0AC	Reserved	Reserved																															

Table 108. RCC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0B0	RCC_ AHB1SMENR	SRAM1SMEN	DCACHESMEN	ICACHESMEN	BKPSRAMSMEN	Res.	Res.	Res.	GTZC1SMEN	Res.	Res.	Res.	Res.	Res.	DMA2DSMEN	RAMCFGSMEN	TSCSMEN	Res.	Res.	Res.	CRCSMEN	Res.	Res.	Res.	FLASHSMEN	Res.	Res.	Res.	MDF1SMEN	FMACSMEN	CORDICSMEN	GPDMA1SMEN
	Reset value	1	1	1	1				1						1	1	1				1				1				1	1	1	1
0x0B4	RCC_ AHB2SMENR1	SRAM3SMEN	SRAM2SMEN	Res.	SDMMC2SMEN	SDMMC1SMEN	Res.	Res.	OTFDEC2SMEN	OTFDEC1SMEN	Res.	OCTOSPIMSMEN	SAESSMEN	PKASMEN	RNGSMEN	HASHSMEN	AESSMEN	Res.	OTGSMEN	Res.	DCMI_PSSISMEN	Res.	ADC1SMEN	Res.	GPIOISMEN	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	1	1	1	1	1	1	1	1
0x0B8	RCC_ AHB2SMENR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCTOSPI2SMEN	Res.	Res.	Res.	OCTOSPI1SMEN	Res.	Res.	FSMCSMEN
	Reset value																								1				1			1
0x0BC	RCC_ AHB3SMENR	SRAM4SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GTZC2SMEN	Res.	ADF1SMEN	LPDMA1SMEN	Res.	Res.	DAC1SMEN	ADC4SMEN	Res.	Res.	PWRSMEN	Res.	LPGPIO1SMEN
	Reset value	1																			1		1	1			1	1			1		1
0x0C0	Reserved	Reserved																															
0x0C4	RCC_ APB1SMENR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRSSMEN	Res.	I2C2SMEN	I2C1SMEN	UART5SMEN	UART4SMEN	USART3SMEN	USART2SMEN	Res.	Res.	SPI2SMEN	Res.	Res.	Res.	WWDGSMEN	Res.	Res.	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
0x0C8	RCC_ APB1SMENR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCAN1SMEN	Res.	Res.	Res.	LPTIM2SMEN	Res.	Res.	Res.	I2C4SMEN	Res.
	Reset value									1														1				1				1	
0x0CC	RCC_ APB2SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2SMEN	SAI1SMEN	Res.	Res.	TIM17SMEN	TIM16SMEN	TIM15SMEN	Res.	USART1SMEN	TIM8SMEN	SPI1SMEN	TIM1SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value										1	1			1	1	1		1	1	1	1												
0x0D0	RCC_ APB3SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBSMEN	VREFSMEN	Res.	Res.	Res.	Res.	COMPSMEN	OPAMPSMEN	LPTIM4SMEN	LPTIM3SMEN	LPTIM1SMEN	Res.	Res.	Res.	Res.	I2C3SMEN	LPUART1SMEN	SPI3SMEN	Res.	Res.	Res.	SYSCFGSMEN	Res.
	Reset value											1	1					1	1	1	1	1				1	1	1					1	
0x0D4	Reserved	Reserved																															

Table 108. RCC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0D8	RCC_ SRDAMR	SRAM4AMEN	Res.	ADF1AMEN	LPDMA1AMEN	DAC1AMEN	LPGPIO1AMEN	ADC4AMEN	Res.	Res.	Res.	RTCAPBAMEN	VREFAMEN	Res.	Res.	Res.	Res.	COMPAMEN	OPAMPAMEN	LPTIM4AMEN	LPTIM3AMEN	LPTIM1AMEN	Res.	Res.	Res.	I2C3AMEN	LPUART1AMEN	SPI3AMEN	Res.	Res.	Res.	Res.	Res.
	Reset value	0		0	0	0	0	0				0	0					0	0	0	0	0				0	0	0					
0x0E0	RCC_ CCIPR1	TIMICSEL[2:0]			Res.	ICLKSEL[1:0]		FDCANSEL[1:0]		SYSTICKSEL[1:0]		SPI1SEL[1:0]		LPTIM2SEL[1:0]		SPI2SEL[1:0]		I2C4SEL[1:0]		I2C2SEL[1:0]		I2C1SEL[1:0]		UART5SEL[1:0]		UART4SEL[1:0]		USART3SEL[1:0]		USART2SEL[1:0]		USART1SEL[1:0]	
	Reset value	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0E4	RCC_ CCIPR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCTOSPISEL[1:0]		Res.	Res.	Res.	Res.	Res.	SDMMCSEL	RNGSEL[1:0]		SAESSEL	SAI2SEL[2:0]		SAI1SEL[2:0]		Res.	Res.	Res.	MDF1SEL[2:0]		
	Reset value											0	0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0E8	RCC_ CCIPR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1SEL[2:0]			DAC1SEL[1:0]		ADCDACSEL[2:0]		LPTIM1SEL[1:0]		LPTIM34SEL[1:0]		I2C3SEL[1:0]		Res.	SPI3SEL[1:0]		LPUART1SEL[2:0]		
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0
0x0F0	RCC_BDCR	Res.	Res.	Res.	LSIPREDIV	LSIRDY	LSION	LSCOSEL	LSCOEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BDRST	RTCEN	Res.	Res.	LSEGFRDY	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	0	0	0	0	0
0x0F4	RCC_CSR	LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	BORRSTF	PINRSTF	OBLRSTF	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSISSRANGE[3:0]				MSIKSRANGE[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	1	1	0		0								0	1	0	0	0	1	0	0								
0x0F8 to 0x10C	Reserved	Reserved																															
0x110	RCC_ SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RMVFSEC	HSI48SEC	ICLKSEC	PLL3SEC	PLL2SEC	PLL1SEC	PRESCSEC	SYSCLKSEC	LSESEC	LSISEC	MSISEC	HSESEC	HSISEC
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x114	RCC_ 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.	NSPRIV	SPRIV
	Reset value																															0	0

Refer to *Section 2.3* for the register boundary addresses.

12 Clock recovery system (CRS)

12.1 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.

12.2 CRS main features

- Selectable synchronization source with programmable prescaler and polarity:
  - 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)

12.3 CRS implementation

Table 109. CRS features

Feature	CRS1
TRIM width	7 bits

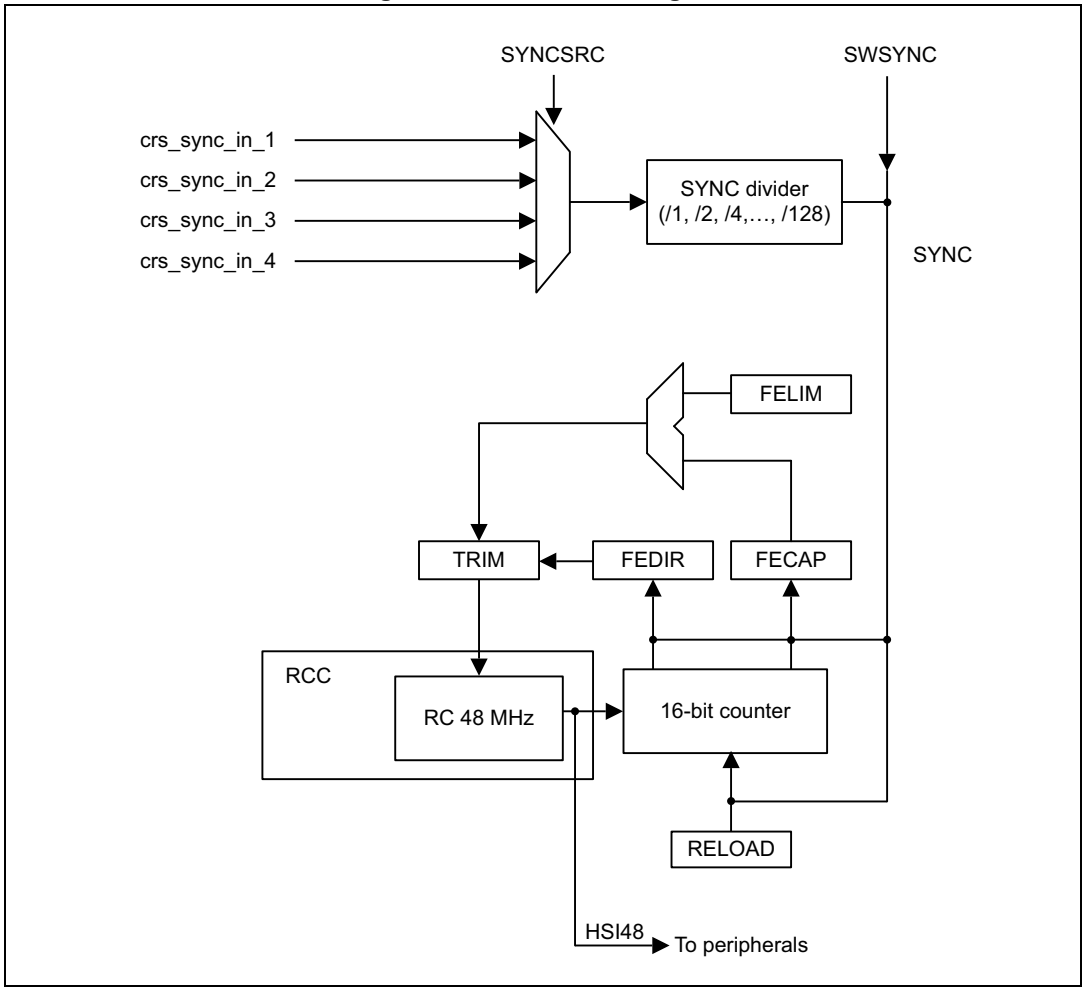
Table 110. CRS internal input/output signals

Internal signal name	Signal type	Description
crs_sync_in_1	Input	00: GPIO AF selected as SYNC signal source
crs_sync_in_2	Input	01: LSE selected as SYNC signal source
crs_sync_in_3	Input	10: USB SOF selected as SYNC signal source (default)
crs_sync_in_4	Input	11: Reserved

12.4 CRS functional description

12.4.1 CRS block diagram

Figure 38. CRS block diagram



12.4.2 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 HSI48 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 12.7.2: CRS configuration register (CRS_CFGR)*.

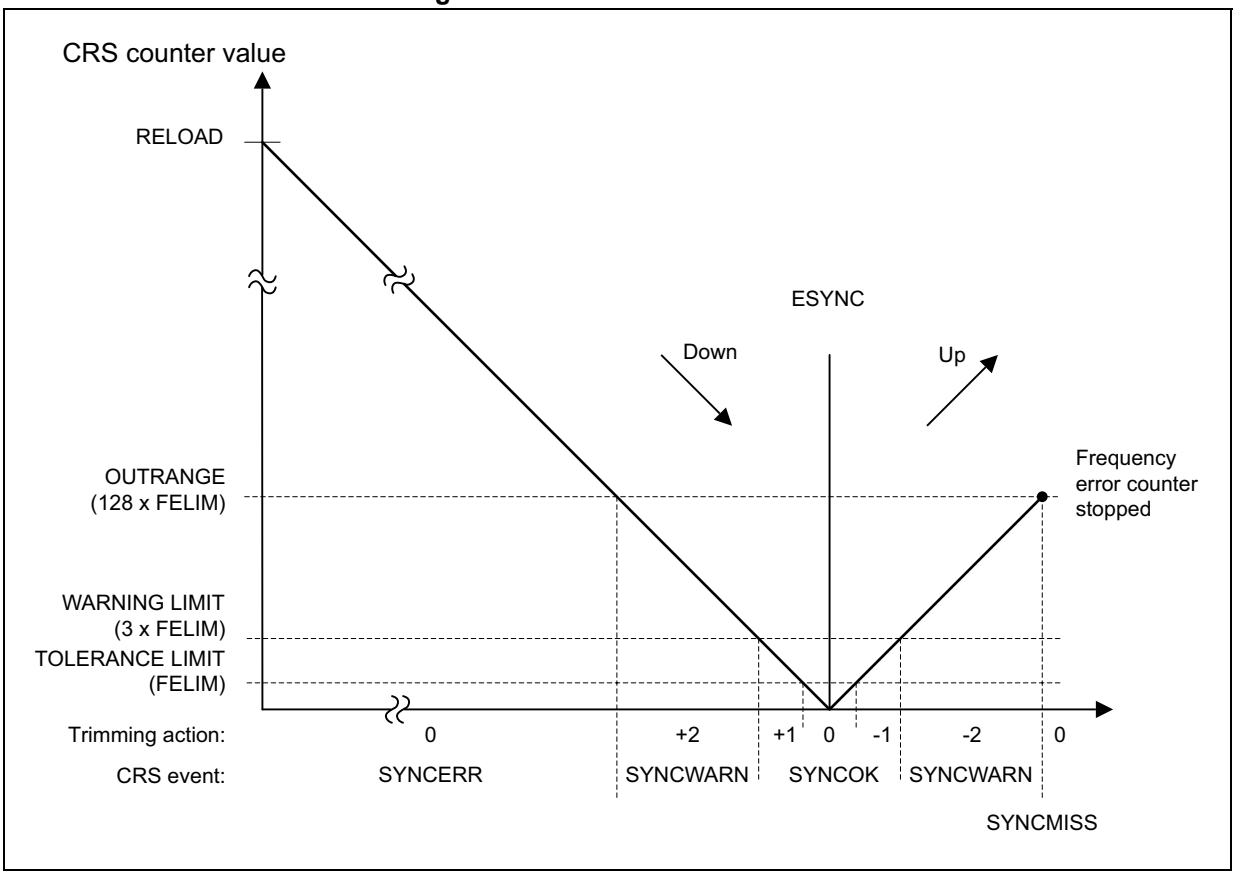
It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS_CR register.

12.4.3 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 39. CRS counter behavior



12.4.4 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.

12.4.5 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%.

Note: The trimming step size depends upon the product, check the datasheet for accurate setting.

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).

12.5 CRS low-power modes

Table 111. 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.

12.6 CRS interrupts

Table 112. 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

12.7 CRS registers

Refer to *Section 1.2 on page 104* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed only by words (32-bit).

12.7.1 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

The default value of the HSI48 oscillator smooth trimming is 64, which corresponds to the middle of the trimming interval.

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 12.4.4* 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

12.7.2 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 (see *Table 110: CRS internal input/output signals*):
00: crs_sync_in_1 selected as SYNC signal source
01: crs_sync_in_2 selected as SYNC signal source
10: crs_sync_in_3 selected as SYNC signal source
11: crs_sync_in_4 selected as SYNC signal source

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 must 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 12.4.4* 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 12.4.3* for more details about counter behavior.

12.7.3 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 12.4.4* 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

12.7.4 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.

12.7.5 CRS register map

Table 113. 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

Table 113. CRS register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C	**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 115* for the register boundary addresses.

13 General-purpose I/Os (GPIO)

13.1 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), a 16 bits reset register (GPIOx_BRR) and a 32-bit set/reset register (GPIOx_BSRR).

In addition, all GPIOs have a 32-bit locking register (GPIOx_LCKR), two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL), a secure configuration register (GPIOx_SECCFGR) and a high-speed low-voltage register (GPIOx_HSLVR).

13.2 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
- Lock 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

13.3 GPIO functional description

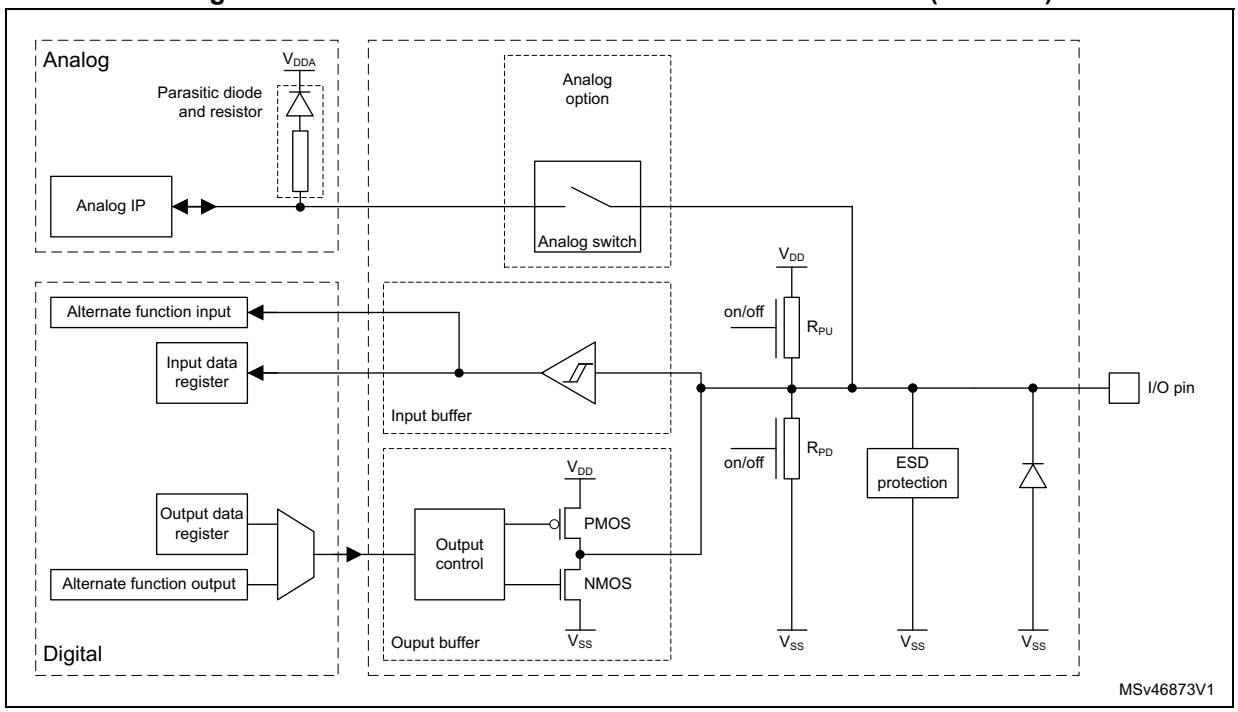
Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

- 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 must be accessed as 32-bit words, half-words or bytes. The GPIOx_BSRR and GPIOx_BRR registers allow atomic read/modify accesses to any of the GPIOx_ODR registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.

The figure below shows the basic structure of a three-volt or five-volt tolerant GPIO (TT or FT). The *Table 114* gives the possible port bit configurations.

Figure 40. Structure of three-volt or five-volt tolerant GPIO (TT or FT)



Note: On a TT GPIO, the analog switch is not present and replaced by a direct connection. The analog bloc parasitic circuitry does not allow five-volt tolerance.

Table 114. Port bit configuration(1)

MODE(i) [1:0]	OTYPE(i)	OSPEED(i) [1:0]	PUPD(i) [1:0]		I/O configuration	
01	0	SPEED[1:0]	0	0	GP output	PP
	0		0	1	GP output	PP + PU
	0		1	0	GP output	PP + PD
	0		1	1	Reserved	
	1		0	0	GP output	OD
	1		0	1	GP output	OD + PU
	1		1	0	GP output	OD + PD
	1		1	1	Reserved (GP output OD)	

Table 114. Port bit configuration[1] (continued)

MODE(i) [1:0]	OTYPE(i)	OSPEED(i) [1:0]		PUPD(i) [1:0]		I/O configuration	
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.

13.3.1 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/SWDIO in pull-up
- PB4: NJTRST in pull-up
- PB3: JTDO/TRACESWO 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 13.3.15: 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, that can be activated or not depending on the value in the GPIOx_PUPDR register.

13.3.2 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 is no conflict between peripherals available on the same I/O pin.

Each I/O pin has a multiplexer with up to 16 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 must proceed as follows:

- **Debug function**: after each device reset these pins are assigned as alternate function pins immediately usable by the debugger host.
- **GPIO**: configure the desired I/O as output, input or analog in the GPIOx_MODER register.
- **Peripheral alternate function**:
  - Connect the I/O to the desired AFx in 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_OSPEEDR registers respectively.
  - Configure the desired I/O as an alternate function in the GPIOx_MODER register.
- **Additional functions:**
  - For the ADC, DAC, OPAMP and COMP, configure the desired I/O in analog mode in the GPIOx_MODER register and configure the required function in the ADC, DAC, OPAMP and COMP registers.
  - 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.

13.3.3 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.

13.3.4 I/O port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (*GPIO port input data register (GPIOx_IDR) (x = A to I)* and *GPIO port output data register (GPIOx_ODR) (x = A to I)*).

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.

13.3.5 I/O data bitwise handling

The bit set reset register (GPIOx_BSRR) is a 32-bit register that allows the application to set and reset each individual bit in the output data register (GPIOx_ODR). 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, BS(i) sets the corresponding ODR(i) bit. When written to 1, BR(i) resets the ODR(i) corresponding bit.

Writing any bit to 0 in GPIOx_BSRR does not have any effect on the corresponding bit in GPIOx_ODR. If there is an attempt to both set and reset a bit in GPIOx_BSRR, the set action takes priority.

Using 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: one or more bits can be modified in a single atomic AHB write access.

13.3.6 GPIO locking mechanism

The GPIO control registers can be frozen 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, GPIOx_AFRH and GPIOx_HSLVR.

To write the GPIOx_LCKR register, a specific write/read sequence must be applied. When the right LOCK sequence is applied to the 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 is applied to a port bit, the value of the port bit can no longer be modified until the next MCU reset or peripheral reset. Each GPIOx_LCKR bit freezes the corresponding bit in the control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

The LOCK sequence can only be performed using a word (32-bit long) access to the GPIOx_LCKR register due to the fact that GPIOx_LCKR bit 16 must be set at the same time as the [15:0] bits.

13.3.7 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.

13.3.8 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 21: Extended interrupts and event controller (EXTI)*.

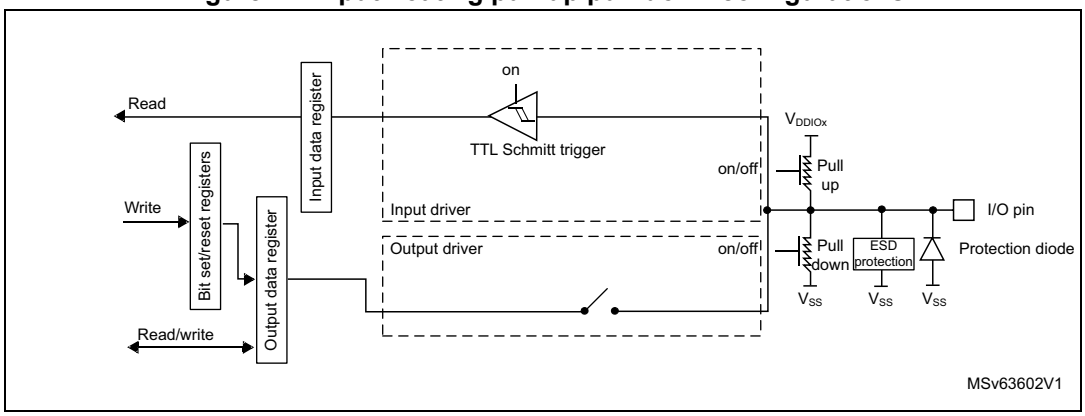
13.3.9 Input configuration

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.

The figure below shows the input configuration of the I/O port bit.

Figure 41. Input floating/pull-up/pull-down configurations



13.3.10 Output configuration

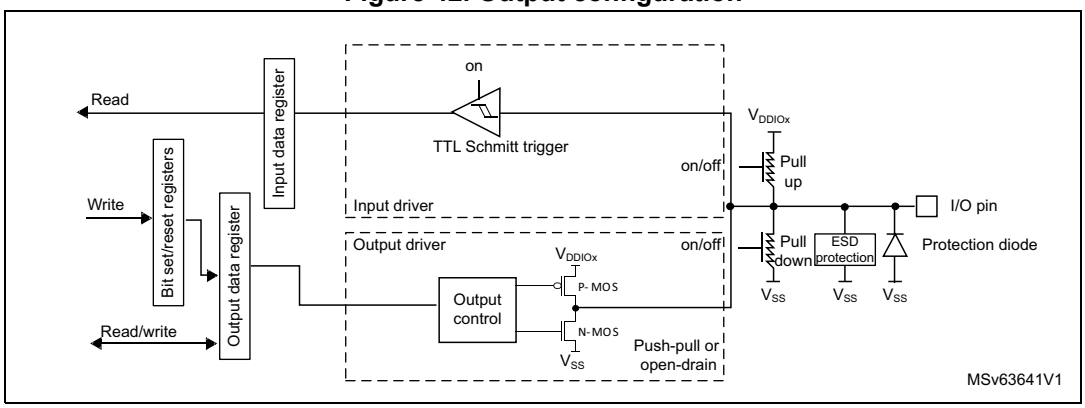
When the I/O port is programmed as output:

- 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.

The figure below shows the output configuration of the I/O port bit.

Figure 42. Output configuration



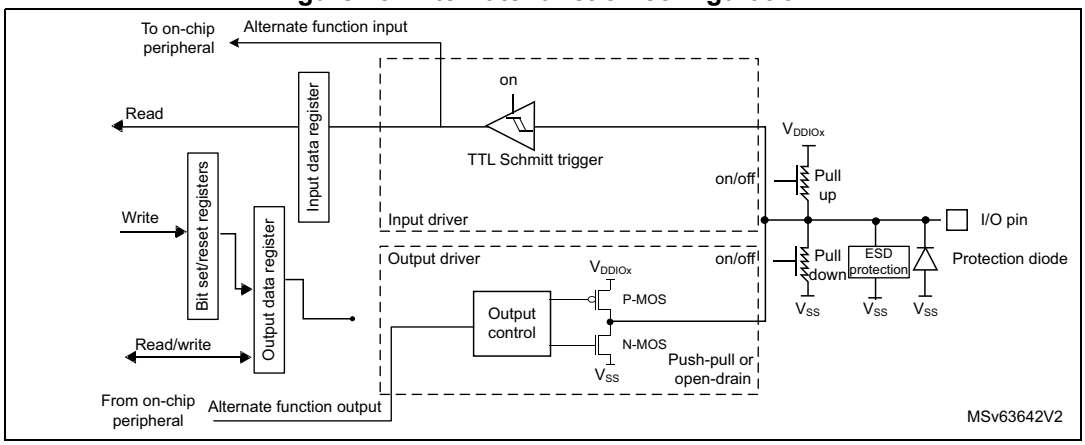
13.3.11 Alternate function configuration

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.

The figure below shows the alternate function configuration of the I/O port bit.

Figure 43. Alternate function configuration



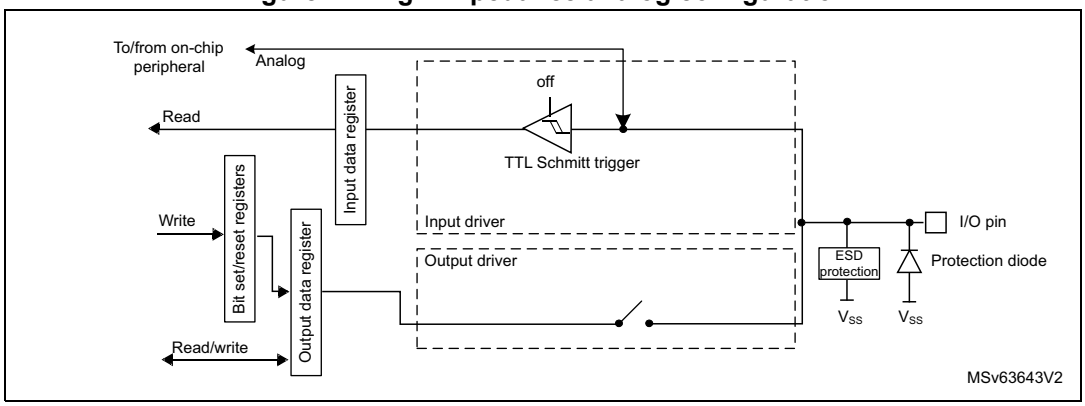
13.3.12 Analog configuration

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.

The figure below shows the high-impedance, analog-input configuration of the I/O port bits.

Figure 44. High-impedance analog configuration



13.3.13 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.

13.3.14 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 54.3: RTC functional description*

13.3.15 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, PH3 switches from the input mode to the analog input mode:

- After the option byte loading phase if nSWBOOT0 = 1.
- After reset if nSWBOOT0 = 0.

13.3.16 OPAMPx_VINM dedicated pins

The OPAMPx_VINM dedicated pins are three-volt tolerant and are supplied by V_{DDA}. These pins do not feature a complete TT structure as shown in *Figure 40*, but a direct connection to OPAMPx. The OPAMPx_VINM dedicated pins are available on specific packages only (refer to the device datasheet for availability of these pins).

13.3.17 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 GPIO clock and reset control bits in the RCC are automatically configured as secure as soon as at least one I/O in the GPIO is secure.

The I/Os with peripherals functions are also conditioned by the peripheral security configuration (see *Section 5: Global TrustZone controller (GTZC)* for more details):

- 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 remains open. This prevent the redirection of secure data to a non-secure peripheral or I/O through analog path. Refer to *Section 3: System security* for more details.
- Some of the paths between I/Os “additional functions” 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. Refer to *Section 3: System security* for the list of concerned peripherals. 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 115 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 115. GPIO secured bits

Secure configuration bit	Secured bit	Register name	Non-secure access on secure bits
SECy = 1 in GPIOx_SECCFGR(1)	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 and BRy	GPIOx_BSRR	
	LCKy	GPIOx_LCKR	
	BRy	GPIOx_BRR	
	AFSELy[3:0]	GPIOx_AFRH	
		GPIOx_AFRL	
	HSLVy	GPIOx_HSLVR	

1. GPIOx, x = A to I. For x = A to H, y = 0 to 15. For x = I, y = 0 to 7.

13.3.18 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).

13.3.19 High-speed low-voltage mode (HSLV)

Some I/Os have the capability to increase their maximum speed at low voltage by configuring them in HSLV mode. The I/O HSLV bit controls whether the I/O output speed is optimized to operate at 3.3 V (default setting) or at 1.8 V (HSLV = 1).

Caution: The I/O HSLV configuration bit must not be set if the I/O supply (V_{DD} or V_{DDIO2}) is above 2.7 V. Setting it while the voltage is higher than 2.7 V can damage the device. The I/O HSLV bit can be set only when the corresponding option bit is activated (IO_VDD_HSLV or IO_VDDIO2_HSLV depending on the I/O supply, refer to *Section 7.4: FLASH option bytes*). There is no hardware protection associated to this feature so it is recommended to use it only as a static configuration for fixed I/O supply.

13.3.20 I/O compensation cell

The I/O commutation slew rate (t_{fall}/t_{rise}) can be adapted by software depending on Process, Voltage and Temperatures conditions, in order to reduce the I/O noise on power supply. Refer to *Section 15: System configuration controller (SYSCFG)* for more details.

13.4 GPIO registers

This section gives a detailed description of the GPIO registers.

The peripheral registers can be written in word, half word or byte mode.

13.4.1 GPIO port mode register (GPIOx_MODER) (x = A to I)

Address offset:0x00

Reset value: 0xABFF FFFF (for port A)

Reset value: 0xFFFF FEBF (for port B)

Reset value: 0xFFFF FFFF (for ports C..H)

Reset value: 0x0000 FFFF (for port I)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MODEy[1:0]:** Port 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)

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to I)

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 **OTy:** 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

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to I)

Address offset: 0x08

Reset value: 0x0C00 0000 (for port A)

Reset value: 0x0000 00C0 (for port B)

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 **OSPEEDy[1:0]**: Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O output speed.

00: Low speed

01: Medium speed

10: High speed

11: Very-high speed

Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed.

The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to I)

Address offset: 0x0C

Reset value: 0x6400 0000 (for port A)

Reset value: 0x0000 0100 (for port B)

Reset value: 0x0000 0000 (for the 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 **PUPDy[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

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.5 GPIO port input data register (GPIOx_IDR) (x = A to I)

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 **IDy:** 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.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.6 GPIO port output data register (GPIOx_ODR) (x = A to I)

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 **ODy:** 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 or GPIOx_BRR registers (x = A to I).

The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to I)

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 **BRy:** 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 ODy bit

1: Resets the corresponding ODy bit

Note: If both BSy and BRy are set, BSy has priority.

The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Bits 15:0 **BSy:** 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 ODy bit

1: Sets the corresponding ODy bit

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A to I)

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]

- LOCK key read

RD LCKR[16] = 1 (this read operation is optional but it confirms that the lock is active)

Note: During the LOCK key write sequence, the value of LCK[15:0] must not change.

Any error in the lock sequence aborts the LOCK.

After the first LOCK sequence on any bit of the port, any read access on the LCKK bit returns 1 until the next MCU reset or peripheral reset.

Bits 15:0 **LCKy:** Port 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

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A to I)

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 **AFSELy[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

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.10 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 **AFSELy[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

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.11 GPIO port bit reset register (GPIOx_BRR) (x = A to I)

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 **BRy:** 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 ODy bit
1: Reset the corresponding ODy bit

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.12 GPIO high-speed low-voltage register (GPIOx_HSLVR) (x = A to I)

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
HSLV 15	HSLV 14	HSLV 13	HSLV 12	HSLV 11	HSLV 10	HSLV9	HSLV8	HSLV7	HSLV6	HSLV5	HSLV4	HSLV3	HSLV2	HSLV1	HSLV0
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 **HSLVy:** Port x high-speed low-voltage configuration (y = 15 to 0)

These bits are written by software to optimize the I/O speed when the I/O supply is low.

Each bit is active only if the corresponding IO_VDD_HSLV/IO_VDDIO2_HSLV user option bit is set. It must be used only if the I/O supply voltage is below 2.7 V.

Setting these bits when the I/O supply (V_{DD} or V_{DDIO2}) is higher than 2.7 V may be destructive.

0: I/O speed optimization disabled

1: I/O speed optimization enabled

Note: Not all I/Os support the HSLV mode. Refer to the I/O structure in the corresponding datasheet for the list of I/Os supporting this feature. Other I/Os HSLV configuration must be kept at reset value.

The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.13 GPIO secure configuration register (GPIOx_SECCFGR) (x = A to I)

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.

When the system is not secure (TZEN = 0), this register is RAZ/WI.

Address offset: 0x30

Reset value: 0x0000 FFFF (for ports A to H)

Reset value: 0x0000 00FF (for port I)

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SECy**: I/O pin of Port x secure bit enable y (y = 15 to 0)

These bits are written by software to enable or disable the I/O port pin security.

0: The I/O pin is non-secure

1: The I/O pin is secure. Refer to *Table 115* for all corresponding secured bits.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.14 GPIO register map

Table 116. GPIO 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	**GPIOx_MODER (x = A to I)**	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value for port A	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value for port B	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1
	Reset value for ports C...H	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
	Reset value for port I	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
0x04	**GPIOx_OTYPER (x = A to I)**	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 (x = A to I)**	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 for port A	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value for port B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
	Reset value for ports C...I	0	0	0	0	0	0	0	0	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	**GPIOx_PUPDR (x = A to I)**	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]		PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset value for port A	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value for port B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	Reset value for ports C...I	0	0	0	0	0	0	0	0	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 (x = A to I)**	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 (x = A to I)**	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 (x = A to I)**	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 (x = A to I)**	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

Table 116. GPIO 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
0x20	**GPIOx_AFRL (x = A to I)**	AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]				AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**GPIOx_AFRH (x = A to H)**	AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]				AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**GPIOx_BRR (x = A to I)**	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
0x2C	**GPIOx_HSLVR (x = A to I)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSLV15	HSLV14	HSLV13	HSLV12	HSLV11	HSLV10	HSLV9	HSLV8	HSLV7	HSLV6	HSLV5	HSLV4	HSLV3	HSLV2	HSLV1	HSLV0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	**GPIOx_ SECCFGR (x = A to I)**	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 to H																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value for port I																	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Refer to *Section 2.3* for the register boundary addresses.

14 Low-power general-purpose I/Os (LPGPIO)

14.1 Introduction

The low-power general-purpose input/output (LPGPIO) allows the I/O control in Stop mode (down to Stop 2 mode), using DMA in memory-to-memory transfer mode. LPGPIO is designed to be used in conjunction with the GPIO.

14.2 LPGPIO main features

- 16 I/Os control in low-power modes down to Stop 2 mode.
- Secure clock and reset management
- Output data from output data register (LPGPIO_ODR)
- Input data to input data register (LPGPIO_IDR)
- Bit set and reset register (LPGPIO_BSRR) for bitwise write access to LPGPIO_ODR
- TrustZone security support

14.3 LPGPIO functional description

14.3.1 LPGPIO and GPIO configuration

The LPGPIO can control in input or output up to 16 I/Os thanks to LPGPIO_Py (y = 0 to 15) pins alternate functions. This control is still functional in Stop 2 mode, thanks to DMA transfers. The I/Os with a LPGPIO function must be configured as LPGPIO alternate function in the *GPIO port mode register (GPIOx_MODER) (x = A to I)*. Then the I/O value can be read in the LPGPIO input data register, and the I/O can be driven by the LPGPIO output data register or thanks to LPGPIO_BSRR/LPGPIO_BRR registers.

14.3.2 LPGPIO control registers

Each of the 16 I/Os controlled by the LPGPIO can be configured as input or output thanks to the *LPGPIO port mode register (LPGPIO_MODER)*.

14.3.3 LPGPIO I/O data registers

The LPGPIO includes the following 16-bit data registers:

- *LPGPIO port output data register (LPGPIO_ODR)* that stores the data to be output (read/write accessible)
- *LPGPIO port input data register (LPGPIO_IDR)*: the data input through the I/O is stored into this input data register (read only)

14.3.4 LPGPIO I/O data bitwise handling

The 32-bit *LPGPIO port bit set/reset register (LPGPIO_BSRR)* is implemented to allow bitwise set and reset in the output data register (LPGPIO_ODR).

Each LPGPIO_ODR bit has the following control bits in LPGPIO_BSRR:

- LPGPIO_BSRR(i): when writing 1 to it, this bit sets the LPGPIO_ODR(i) bit.
- LPGPIO_BSRR(i + 16): when writing 1 to it, this bit resets the LPGPIO_ODR(i) bit.

Note: Writing 0 to these bits has no effect on LPGPIO_ODR corresponding bits.

If there is an attempt to set and reset bits of the same index, the set action takes the priority.

Writing LPGPIO_BSRR register does not lock the LPGPIO_ODR bits, that can be anyway accessed directly. LPGPIO_BSRR provides a way to perform atomic bitwise handling.

The 16-bit *LPGPIO port bit reset register (LPGPIO_BRR)* allows individual bit reset. It is the same as LPGPIO_BSRR but with minimal pattern preparation:

- LPGPIO_BRR(i): when writing 1 to it, this bit resets the LPGPIO_ODR(i) bit.

14.3.5 Security protection

The LPGPIO includes a security mechanism, that allows or locks the access to the I/O configuration and data registers. This system is used to protect the I/O against the data corruption or observation.

The security mechanism within the LPGPIO is directly issued from the *GPIO secure configuration register (GPIOx_SECCFGR) (x = A to I)*. Therefore, no additional configuration is required.

The LPGPIO security means the following:

- When the executed code is secure, all bits can be accessed.
- When the executed code is non-secure, only the bits concerning the non-secure I/Os can be accessed.

A non-secure access to a secure I/O register bit is silent fail:

- A non-secure write to a secure I/O register bit is ignored (WI).
- A non-secure read to a secure I/O register bit returns 0 (RAZ).
- No bus error is generated.

14.3.6 Secure clock and reset management

The LPGPIO clock and reset control bits in the RCC are automatically configured as secured as soon as at least one I/O with LPGPIO alternate function is secure (refer to the corresponding I/Os in GPIO_SECCFGR registers)

14.4 LPGPIO registers

This section gives a detailed description of the LPGPIO registers.

The peripheral registers can be written in word, half-word or byte mode.

14.4.1 LPGPIO port mode register (LPGPIO_MODER)

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
MODE15	MODE14	MODE13	MODE12	MODE11	MODE10	MODE9	MODE8	MODE7	MODE6	MODE5	MODE4	MODE3	MODE2	MODE1	MODE0
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 **MODEy:** Configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O mode.

0: Input mode

1: Output mode

14.4.2 LPGPIO port input data register (LPGPIO_IDR)

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
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **IDy:** 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.

14.4.3 LPGPIO port output data register (LPGPIO_ODR)

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 **ODy:** 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 LPGPIO_BSRR or LPGPIO_BRR registers.

14.4.4 LPGPIO port bit set/reset register (LPGPIO_BSRR)

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 **BRy:** Reset I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns 0.

0: No action on the corresponding ODy bit

1: Resets the corresponding ODy bit

Note: If both BSy and BRy are set, BSy has priority.

Bits 15:0 **BSy:** Port x set I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns 0.

0: No action on the corresponding ODy bit

1: Sets the corresponding ODy bit

14.4.5 LPGPIO port bit reset register (LPGPIO_BRR)

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 **BRy:** Reset IO pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns 0.
0: No action on the corresponding ODy bit
1: Reset the corresponding ODy bit

14.4.6 LPGPIO register map

Table 117. LPGPIO 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	LPGPIO_MODER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE15	MODE14	MODE13	MODE12	MODE11	MODE10	MODE9	MODE8	MODE7	MODE6	MODE5	MODE4	MODE3	MODE2	MODE1	MODE0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04-0x08	Reserved	Reserved																															
0x10	LPGPIO_IDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	LPGPIO_ODR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	LPGPIO_BSRR	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20-0x24	Reserved	Reserved																															
0x28	LPGPIO_BRR		Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

15 System configuration controller (SYSCFG)

15.1 SYSCFG main features

The STM32U575/585 devices feature a set of configuration registers. The main purposes of the system configuration controller are the following:

- Managing robustness feature
- Configuring FPU interrupts
- Enabling/disabling the FMP high-drive mode of some I/Os and voltage booster for I/Os analog switches
- Managing the I/O compensation cell
- Configuring register security access

15.2 SYSCFG functional description

15.2.1 I/O compensation cell management

The I/O compensation cell generates an 8-bit value for the I/O buffer (4 bits for N-MOS and 4 bits for P-MOS), that depends on PVT operating conditions (process, voltage, temperature). These bits are used to control the current slew-rate and output impedance in the I/O buffer. Two compensation cells are embedded, one for the I/Os supplied by V_{DD} and one for the I/Os supplied by V_{DDIO2}.

By default, the compensation cells are disabled, and a fixed code is applied to all the I/Os. The HSI is used by the compensation cells and must be enabled before enabling the compensation cells in the *SYSCFG compensation cell control/status register (SYSCFG_CCCSR)*.

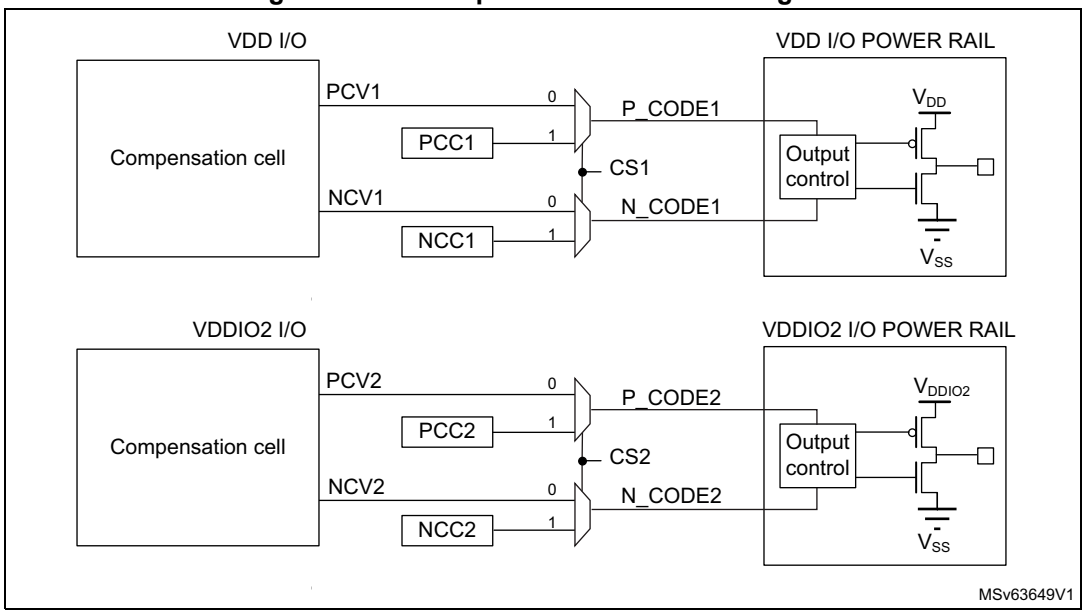
When enabled, the compensation cell tracks the PVT, and the 8-bit code PCVx and NCVx (x = 1 for I/Os supplied by V_{DD}, x = 2 for I/Os supplied by V_{DDIO2}) are available in SYSCFG_CCVR once the RDYx is set. If the CSx bit is cleared, the I/Ox receives the code from SYSCFG_CCVR, resulting from the compensation cell.

To optimize the trimming, the code can be adjusted through SYSCFG_CCCR. Two sets of bits are available: PCC1/NCC1 for the V_{DD} power rail and PCC2/NCC2 for the V_{DDIO2} power rail. They can be selected independently through CS1 and CS2 bits in SYSCFG_CCCSR (see *Figure 45*).

To reduce the power consumption, it is recommended to copy the code from SYSCFG_CCVR to SYSCFG_CCCR. After the result is ready, set the CSx bit and disable the compensation cell.

Note: The compensation cell can be used only when 1.6 V ≤ V_{DDIOx} ≤ 3.6 V.

Figure 45. I/O compensation cell block diagram



15.2.2 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.

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, and the FPUSEC bit in the SYSCFG_SECCFGR.

An unprivileged access to a privileged register is RAZ/WI.

The table below shows the register security overview.

Table 118. TrustZone security and privilege register accesses

SYSCFG register name	Read/write access		Privileged /unprivileged access
TrustZone configuration(1)	**TZEN = 1**	**TZEN = 0**	**Not applicable**
SYSCFG_SECCFGR	Read: no restriction Write: secure access only Non-secure write is WI and generates an illegal access event.	RAZ/WI	Read: no restriction FPUSEC privileged write only Other bits write: no restriction
SYSCFG_CSLCKR	Read/Write: secure access only Non-secure access is RAZ/WI and generates and an illegal access event.	RAZ/WI	Privileged only Unprivileged: RAZ/WI
SYSCFG_FPUIMR	– If FPUSEC = 1: Read/Write: secure access only Non-secure access is RAZ/WI and generates an illegal access event. – If FPUSEC = 0: Read/Write: no restriction	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 GTZC Non-secure access only for non-secure bits, otherwise RAZ/WI	No restriction	No restriction
SYSCFG_CFGR2	– If CLASSBSEC = 1: Read/Write: secure access only Non-secure access is RAZ/WI and generates an illegal access event. – If CLASSBSEC = 0:Read/Write: no restriction	No restriction	No restriction
SYSCFG_MESR	– If SYSCFGSEC = 1: Read/Write: secure access only Non-secure access is RAZ/WI and generates an illegal access event. – If SYSCFGSEC = 0:Read/Write: no restriction	No restriction	No restriction
SYSCFG_CCCSR SYSCFG_CCVR SYSCFG_CCCR	– If SYSCFGSEC = 1: Read/Write: secure access only Non-secure access is RAZ/WI and generates an illegal access event – If SYSCFGSEC = 0: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.

15.3 SYSCFG registers

15.3.1 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, except for FPUSEC that can be written only with privileged 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	Res.	CLASSBSEC	SYSCFGSEC
												rw		rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **FPUSEC**: FPU security
0: SYSCFG_FPUIMR register can be read and written by secure and non-secure access.
1: SYSCFG_FPUIMR register can be read and written by secure access only.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CLASSBSEC**: Class B security
0: SYSCFG_CFGR2 register can be read and written by secure and non-secure access.
1: SYSCFG_CFGR2 register can be read and written by secure access only.

Bit 0 **SYSCFGSEC**: SYSCFG clock control, memory erase status and compensation cell registers security
0: SYSCFG configuration clock in RCC registers, SYSCFG_MESR and SYSCFG_CCCSR, SYSCFG_CCVR and SYSCFG_CCCR can be read and written by secure and non-secure access.
1: SYSCFG configuration clock in RCC registers, SYSCFG_MESR and SYSCFG_CCCSR, SYSCFG_CCVR and SYSCFG_CCCR can be read and written by secure access only.

15.3.2 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 ADC security configuration bit in GTZC peripheral 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.	Res.	Res.	Res.	Res.	PB9_FMP	PB8_FMP	PB7_FMP	PB6_FMP
												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 31:20 Reserved, must be kept at reset value.

Bit 19 **PB9_FMP**: Fast-mode Plus driving capability activation on PB9

This bit can be read and written only with secure access if PB9 is secure in GPIOB. This bit enables the Fm+ driving mode for PB9 when PB9 is not used by I2C peripheral. This can be used to drive a LED for instance.

0: PB9 pin operates in standard mode.

1: Fm+ mode is enabled on PB9 pin and the speed control is bypassed.

Bit 18 **PB8_FMP**: Fast-mode Plus driving capability activation on PB8

This bit can be read and written only with secure access if PB8 is secure in GPIOB. This bit enables the Fm+ driving mode for PB8 when PB8 is not used by I2C peripheral. This can be used to drive a LED for instance.

0: PB8 pin operates in standard mode.

1: Fm+ mode is enabled on PB8 pin and the speed control is bypassed.

Bit 17 **PB7_FMP**: Fast-mode Plus driving capability activation on PB7

This bit can be read and written only with secure access if PB7 is secure in GPIOB. This bit enables the Fm+ driving mode for PB7 when PB7 is not used by I2C peripheral. This can be used to drive a LED for instance.

0: PB7 pin operates in standard mode.

1: Fm+ mode is enabled on PB7 pin and the speed control is bypassed.

Bit 16 **PB6_FMP**: Fast-mode Plus driving capability activation on PB6

This bit can be read and written only with secure access if PB6 is secure in GPIOB. This bit enables the Fm+ driving mode for PB6 when PB6 is not used by I2C peripheral. This can be used to drive a LED for instance.

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
This bit can be read and written only with secure access if ADC1 or ADC4 is secure in GTZC.
0: I/O analog switches are supplied by V_{DDA} or booster when booster is ON.
1: I/O analog switches are supplied by V_{DD}.
Note: Refer to Table 119: BOOSTEN and ANASWVDD set/reset for bit 9 setting.

Bit 8 **BOOSTEN**: I/O analog switch voltage booster enable
This bit can be read and written only with secure access if ADC1 or ADC4 is secure in GTZC.
0: I/O analog switches are supplied by V_{DDA} voltage.
1: I/O analog switches are supplied by a dedicated voltage booster (supplied by V_{DD}).
Note: Refer to Table 119: BOOSTEN and ANASWVDD set/reset for bit 8 setting.

Bits 7:0 Reserved, must be kept at reset value.

the table below describes when the bit 8 (BOOSTEN) and the bit 9 (ANASWVDD) must be set or reset depending on the voltage settings.

Table 119. BOOSTEN and ANASWVDD set/reset

V_{DD}	V_{DDA}	BOOSTEN	ANASWVDD
-	> 2.4 V	0	0
> 2.4 V	< 2.4 V		1
< 2.4 V		1	0

15.3.3 SYSCFG 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	rw	rw	rw	rw	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

15.3.4 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 set, this bit disables write access to non-secure MPU_CTRL_NS, MPU_RNR_NS and MPU_RBAR_NS registers.
0: Non-secure MPU registers write enabled
1: Non-secure MPU registers write 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 enabled
1: VTOR_NS register write disabled

15.3.5 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 set, this bit disables write access to SAU_CTRL, SAU_RNR, SAU_RBAR and SAU_RLAR registers.

0: SAU registers write enabled

1: SAU registers write disabled

Bit 1 **LOCKSMPU**: Secure MPU registers lock

This bit is set by software and cleared only by a system reset. When set, this bit disables write access to secure MPU_CTRL, MPU_RNR and MPU_RBAR registers.

0: Secure MPU registers writes enabled

1: Secure MPU registers writes 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 set, this bit 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 enabled

1: VTOR_S register PRIS and BFHFNMINS bits in the AIRCR register write disabled

15.3.6 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: 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.	Res.	Res.	Res.	Res.	Res.	ECCL	PVDL	SPL	CLL
												rs	rs	rs	rs

Bits 31:4 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 double error signal connection to TIM1/8/15/16/17 break input.

0: ECC double error disconnected from TIM1/8/15/16/17 break input

1: ECC double 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 PVDLS[2:0] in the PWR register.

0: PVD interrupt disconnected from TIM1/8/15/16/17 break input. PVDE and PVDLS[2:0] bits can be programmed by the application.

1: PVD interrupt connected to TIM1/8/15/16/17 break input. PVDE and PVDLS[2:0] bits are read only.

Bit 1 **SPL**: SRAM ECC lock bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the SRAM ECC double error signal connection to TIM1/8/15/16/17 break inputs.

0: SRAM double error disconnected from TIM1/8/15/16/17 break inputs

1: SRAM double error connected to TIM1/8/15/16/17 break inputs

Bit 0 **CLL**: Cortex-M33 LOCKUP (hardfault) output enable

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

15.3.7 SYSCFG memory erase status register (SYSCFG_MESR)

When the system is secure (TZEN =1), this register can be protected against non-secure access by setting the SYSCFGSEC bit in the SYSCFG_SECCFGR register. When SYSCFGSEC bit is set, only secure access is allowed: 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: 0x18

Power-on reset value: 0x0000 0000

System reset value: 0x0000 000X (bit 0 not affected by system reset)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPMEE
															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.	Res.	Res.	Res.	MCLR
															rc_w1

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

Bit 16 **IPMEE**: ICACHE and PKA SRAM erase status

This bit is set by hardware when ICACHE and PKA SRAM erase is completed after potential tamper detection (refer to *Section 55: Tamper and backup registers (TAMP)* for more details). This bit is cleared by software by writing 1 to it.

0: ICACHE and PKA SRAM erase on going if not yet cleared by software

1: ICACHE and PKA SRAM erase done

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **MCLR**: Device memories erase status

This bit is set by hardware when SRAM2, BKPSRAM, ICACHE, DCACHE1, PKA SRAM erase is completed after power-on reset or tamper detection (refer to *Section 55: Tamper and backup registers (TAMP)* for more details). This bit is not reset by system reset and is cleared by software by writing 1 to it.

0: memory erase on going if not yet cleared by software

1: Memory erase done

15.3.8 SYSCFG compensation cell control/status register (SYSCFG_CCCSR)

When the system is secure (TZEN =1), this register can be protected against non-secure access by setting the SYSCFGSEC bit in the SYSCFG_SECCFGR register. When SYSCFGSEC bit is set, only secure access is allowed: 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 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.	RDY2	RDY1	Res.	Res.	Res.	Res.	CS2	EN2	CS1	EN1
						r	r					rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **RDY2:** VDDIO2 I/Os compensation cell ready flag

This bit provides the compensation cell status of the I/Os supplied by V_{DDIO2}.

0: VDDIO2 I/O compensation cell not ready

1: VDDIO2 I/O compensation cell ready

*Note: The HSI clock is required for the compensation cell to work properly. The compensation cell ready bit (*RDY2*) is not set if the HSI clock is not enabled.*

Bit 8 **RDY1:** VDD I/Os compensation cell ready flag

This bit provides the compensation cell status of the I/Os supplied by V_{DD}.

0: VDD I/O compensation cell not ready

1: VDD I/O compensation cell ready

*Note: The HSI clock is required for the compensation cell to work properly. The compensation cell ready bit (*RDY1*) is not set if the HSI clock is not enabled.*

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

Bit 3 **CS2:** VDDIO2 I/Os code selection

This bit selects the code to be applied for the compensation cell of the I/Os supplied by V_{DDIO2}.

0: VDDIO2 I/Os code from the cell (available in the SYSCFG_CCVR)

1: VDDIO2 I/Os code from the SYSCFG compensation cell code register (SYSCFG_CCCR)

Bit 2 **EN2:** VDDIO2 I/Os compensation cell enable
This bit enables the compensation cell of the I/Os supplied by V_{DDIO2}.
0: VDDIO2 I/Os compensation cell disabled
1: VDDIO2 I/Os compensation cell enabled

Bit 1 **CS1:** VDD I/Os code selection
This bit selects the code to be applied for the compensation cell of the I/Os supplied by V_{DD}.
0: VDD I/Os code from the cell (available in the SYSCFG_CCVR)
1: VDD I/Os code from the SYSCFG compensation cell code register (SYSCFG_CCCR)

Bit 0 **EN1:** VDD I/Os compensation cell enable
This bit enables the compensation cell of the I/Os supplied by V_{DD}.
0: VDD I/Os compensation cell disabled
1: VDD I/Os compensation cell enabled

15.3.9 SYSCFG compensation cell value register (SYSCFG_CCVR)

When the system is secure (TZEN =1), this register can be protected against non-secure access by setting the SYSCFGSEC bit in the SYSCFG_SECCFGR register. When SYSCFGSEC bit is set, only secure access is allowed: 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCV2[3:0]				NCV2[3:0]				PCV1[3:0]				NCV1[3:0]			
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 15:12 **PCV2[3:0]:** PMOS compensation value of the I/Os supplied by V_{DDIO2}
This value is provided by the cell and can be used by the CPU to compute an I/O compensation cell code for PMOS transistors. This code is applied to the I/O compensation cell when the CS2 bit of the SYSCFG_CCCSR is reset.

Bits 11:8 **NCV2[3:0]:** NMOS compensation value of the I/Os supplied by V_{DDIO2}
This value is provided by the cell and can be used by the CPU to compute an I/O compensation cell code for NMOS transistors. This code is applied to the I/O compensation cell when the CS2 bit of the SYSCFG_CCCSR is reset.

Bits 7:4 **PCV1[3:0]:** PMOS compensation value of the I/Os supplied by V_{DD}
This value is provided by the cell and can be used by the CPU to compute an I/O compensation cell code for PMOS transistors. This code is applied to the I/O compensation cell when the CS1 bit of the SYSCFG_CCCSR is reset.

Bits 3:0 **NCV1[3:0]:** NMOS compensation value of the I/Os supplied by V_{DD}
This value is provided by the cell and can be used by the CPU to compute an I/O compensation cell code for NMOS transistors. This code is applied to the I/O compensation cell when the CS1 bit of the SYSCFG_CCCSR is reset.

15.3.10 SYSCFG compensation cell code register (SYSCFG_CCCR)

When the system is secure (TZEN =1), this register can be protected against non-secure access by setting the SYSCFGSEC bit in the SYSCFG_SECCFGR register. When SYSCFGSEC bit is set, only secure access is allowed: 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 7878

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCC2[3:0]				NCC2[3:0]				PCC1[3:0]				NCC1[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **PCC2[3:0]:** PMOS compensation code of the I/Os supplied by V_{DDIO2}
These bits are written by software to define an I/O compensation cell code for PMOS transistors. This code is applied to the I/O compensation cell when the CS2 bit of the SYSCFG_CCCSR is set.

Bits 11:8 **NCC2[3:0]:** NMOS compensation code of the I/Os supplied by V_{DDIO2}
These bits are written by software to define an I/O compensation cell code for NMOS transistors. This code is applied to the I/O compensation cell when the CS2 bit of the SYSCFG_CCCSR is set.

Bits 7:4 **PCC1[3:0]:** PMOS compensation code of the I/Os supplied by V_{DD}
These bits are written by software to define an I/O compensation cell code for PMOS transistors. This code is applied to the I/O compensation cell when the CS1 bit of the SYSCFG_CCCSR is set.

Bits 3:0 **NCC1[3:0]:** NMOS compensation code of the I/Os supplied by V_{DD}

These bits are written by software to define an I/O compensation cell code for NMOS transistors. This code is applied to the I/O compensation cell when the CS1 bit of the SYSCFG_CCCSR is set.

15.3.11 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[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 **RSSCMD[15:0]**: RSS commands

This field defines a command to be executed by the RSS.

15.3.12 SYSCFG register map

Table 120. SYSCFG register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**SYSCFG_ 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.	FPUSECC	Res.	CLASSBSEC	SYSCFGSEC
	Reset value																													0		0	0
0x04	**SYSCFG_CFGR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PB9_FMP	PB8_FMP	PB7_FMP	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								
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																											0	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

Table 120. SYSCFG 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
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.	Res.	Res.	Res.	Res.	Res.	ECCL	PVDL	SPL	CLL.
	Reset value																													0	0	0	0
0x18	**SYSCFG_MESR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	IPMEE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCLR
	Reset value																0																0
0x1C	**SYSCFG_CCCSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDY2	RDY1	Res.	Res.	Res.	Res.	CS2	EN2	CS1	EN1
	Reset value																							0	0					1	0	1	0
0x20	**SYSCFG_CCVR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCV2[3:0]				NCV2[3:0]				PCV1[3:0]				NCV1[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**SYSCFG_CCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCC2[3:0]				NCC2[3:0]				PCC1[3:0]				NCC1[3:0]			
	Reset value																	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0
0x28	**Reserved**	Reserved																															
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

Refer to *Section 2.3* for the register boundary addresses.

16 Peripherals interconnect matrix

16.1 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 various power modes: Run, Sleep, Stop 0, Stop 1 and Stop 2 modes.

16.2 Connection summary

Table 121. Peripherals interconnect matrix(1) (2)

Source	Destination																												
	TIM1	TIM8	TIM2	TIM3	TIM4	TIM5	TIM6	TIM7	TIM15	TIM16	TIM17	LPTIM1/2/3	LPTIM4	ADC1	ADC4	MDF1	ADF1	OPAMP1/2	DAC1/2	COMP1/2	GP/LPDMA	IRTIM	U(S)ARTs	LPUART1	I2Cs	SPIs	TAMP	RTC	AES/SAES
TIM1	-	1	1	1	1	1	-	-	1	-	-	-	-	20	20	5	-	-	4	9	-	-	-	-	-	-	-	-	-
TIM8	1	-	1	1	1	1	-	-	1	-	-	-	-	20	-	5	-	-	4	9	-	-	-	-	-	-	-	-	-
TIM2	1	1	-	1	1	1	-	-	1	-	-	-	-	20	20	-	-	-	4	9	16	-	-	-	-	-	-	-	-
TIM3	1	1	1	-	1	1	-	-	1	-	-	-	-	20	-	5	-	-	-	9	16	-	-	-	-	-	-	-	-
TIM4	1	1	1	1	-	1	-	-	1	-	-	-	-	20	-	5	-	-	4	-	16	-	-	-	-	-	-	-	-
TIM5	1	1	1	1	1	-	-	-	1	-	-	-	-	-	-	-	-	-	4	-	16	-	-	-	-	-	-	-	-
TIM6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	20	5	-	-	4	-	-	-	-	-	-	-	-	-	-
TIM7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	5	-	-	4	-	-	-	-	-	-	-	-	-	-
TIM15	1	1	1	1	1	-	-	-	-	-	-	-	-	20	20	-	-	-	4	9	16	-	-	-	-	-	-	-	-
TIM16	1	1	1	1	1	-	-	-	1	-	-	-	-	-	-	5	-	-	-	-	-	14	-	-	-	-	-	-	-
TIM17	1	1	1	1	1	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	14	-	-	-	-	-	-	-
LPTIM1	-	-	-	-	-	-	-	-	-	-	-	-	-	20	20	5	-	-	4	-	16	-	15	15	15	15	-	-	-
LPTIM2	-	-	-	-	-	-	-	-	-	-	-	-	-	20	-	-	-	-	-	-	16	-	15	-	15	15	-	-	-
LPTIM3	-	-	-	-	-	-	-	-	-	-	-	-	-	20	20	-	-	-	4	-	16	-	-	15	15	15	-	-	-
LPTIM4	-	-	-	-	-	-	-	-	-	-	-	-	-	20	-	-	-	-	-	-	16	-	-	-	-	-	-	-	-
ADC1	3	3	-	3	-	-	-	-	-	-	-	-	-	-	-	18	-	-	-	-	16	-	-	-	-	-	-	-	-
ADC4	3	3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	16	-	-	-	-	-	20	-	-
MDF1	6	6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	5	-	-	-	-	-	-	-	-	-	-	-	-
ADF1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	5	-	-	-	-	-	-	-	-	-	-	-	-	-
DAC1/2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	17	-	17	-	-	-	-	-	-	-	-	-
COMP1/2	12	12	12	12	12	12	-	-	12	12	12	8	8	-	-	-	-	-	-	-	16	-	15	15	15	15	-	-	-
GPDMA1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	16	-	15	-	15	15	-	-	-
LPDMA1	-	-	-	-	-	-	-	-	-	-	-	8	8	-	-	-	-	-	-	-	16	-	-	15	15	15	-	-	-
EXTI	-	-	-	-	-	-	-	-	-	-	-	8	8	20	20	5	5	-	4	-	16	-	15	15	15	15	-	-	-
RTC wakeup	-	-	-	-	-	10	-	-	-	10	10	-	-	-	-	-	-	-	-	-	16	-	15	15	15	15	-	-	-

Table 121. Peripherals interconnect matrix(1) (2) (continued)

Source	Destination																												
	TIM1	TIM8	TIM2	TIM3	TIM4	TIM5	TIM6	TIM7	TIM15	TIM16	TIM17	LPTIM1/2/3	LPTIM4	ADC1	ADC4	MDF1	ADF1	OPAMP1/2	DAC1/2	COMP1/2	GP/LPDMA	IRTIM	U(S)ARTs	LPUART1	I2Cs	SPIs	TAMP	RTC	AES/SAES
RTC Alarm	-	-	-	-	-	-	-	-	-	-	-	8	8	-	-	-	-	-	-	-	16	-	15	15	15	15	-	-	-
TAMP	-	-	-	-	-	-	-	-	-	-	-	8	8	-	-	-	-	-	-	-	16	-	-	-	-	-	-	21	22
HSE	-	-	-	-	-	-	-	-	-	7	7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
LSE	-	-	7	-	-	7	-	-	7	7	7	7	-	-	-	-	-	-	19	-	-	-	-	-	-	-	-	-	-
CSS in LSE	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	20	-	-
MSIS/MSIK	7	7	7	7	7	7	-	-	-	7	7	7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
HSI	7	7	7	7	7	7	-	-	-	7	7	7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
LSI	-	-	-	-	-	7	-	-	-	7	7	7	-	-	-	-	-	-	19	-	-	-	-	-	-	-	-	-	-
MCO	-	-	-	-	-	-	-	-	-	7	7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
VCORE	-	-	-	-	-	-	-	-	-	-	-	-	-	-	17	-	-	-	-	-	-	-	-	-	-	-	-	-	-
VREFINT	-	-	-	-	-	-	-	-	-	-	-	-	-	17	17	-	-	-	-	17	-	-	-	-	-	-	-	-	-
T sensor	-	-	-	-	-	-	-	-	-	-	-	-	-	17	17	-	-	-	-	-	-	-	-	-	-	-	-	-	-
VBAT	-	-	-	-	-	-	-	-	-	-	-	-	-	17	17	-	-	-	-	-	-	-	-	-	-	-	-	-	-
VBAT and temp monitor.	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	20	-	-
OPAMP1/2	-	-	-	-	-	-	-	-	-	-	-	-	-	17	17	-	-	-	-	-	-	-	-	-	-	-	-	-	-
System errors	13	13	-	-	-	-	-	-	13	13	13	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
USB OTG	-	-	11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
System Flash	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	22
AES/SAES	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	22

1. Numbers in this table are links to corresponding subsections of *Section 16.3*.
2. "-" means no interconnect.

16.3 Interconnection details

16.3.1 Master to slave interconnection for timers

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 47.4.23: Timer synchronization*.

The synchronization modes are detailed in:

- *Section 46.3.30* for advanced-control timers TIM1/TIM8
- *Section 47.4.22* for general-purpose timers TIM2/TIM3/TIM4/TIM5
- *Section 48.4.23* for the 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. It can be also from signals tim16_oc1 and tim17_oc1 in case of TIM16/TIM17. The input (to slave) is on signals TIMx_ITR0/ITR1/ITR2/ITR3.

The possible master/slave connections are given in:

- *Table 430* for advanced-control timers TIM1/TIM8
- *Table 454* for general-purpose timers TIM2/TIM3/TIM4/TIM5
- *Table 470* for the general-purpose timers TIM15

Active power mode

Timers are optionally active in Run and Sleep modes. The effects of low-power modes on TIMx are given in:

- *Table 443: Effect of low-power modes on TIM1/TIM8*
- *Table 462: Effect of low-power modes on TIM2/TIM3/TIM4/TIM5*
- *Table 477: Effect of low-power modes on TIM15/TIM16/TIM17*

16.3.2 Triggers to ADCs

From EXTI, timers (TIM1/TIM2/TIM3/TIM4/TIM5/TIM8/TIM15/TIM16/TIM17) and LP timers (LPTIM1/ LPTIM2/LPTIM3/LPTIM4) to ADC1

From EXTI, timers (TIM1/TIM2/TIM6/TIM15) and LP timers (LPTIM1/LPTIM3) to ADC4

Purpose

A conversion, or a sequence of conversions, can be triggered either by software or by an external event (such as timer capture or input pins). For ADC1, 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 can trigger a conversion with the selected polarity.

More details in:

- *Section 29.4.19: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN[1:0], JEXTSEL, JEXTEN[1:0])*
- EXTEN[1:0] defined in *ADC configuration register (ADC_CFGR1)*
- JEXTEN[1:0] defined in *ADC injected sequence register (ADC_JSQR)*

General-purpose timers (TIM2/TIM3/TIM4/TIM5), basic timer (TIM6), advanced-control timers (TIM1/TIM8) and general-purpose timer (TIM15/TIM16/TIM17) can be used to generate the ADC triggering event through the timer outputs tim_oc and tim_trgo.

Low-power timers (LPTIM1/ LPTIM2/LPTIM3/LPTIM4) can be used to generate the ADC triggering event through the LPTIM channels (TIMx synchronization described in *Section 46.3.31: ADC triggers* for TIM1/TIM8) in addition to the EXTI on channels 11 and 15.

The ADC4 do not have injected channels. The general-purpose timers (TIM2 and TIM15), basic timers (TIM6), and advanced-control timers (TIM1) can be used to generate the ADC triggering event through the timer outputs tim_oc and tim_trgo. Low-power timers (LPTIM1/LPTIM3) can be used to generate the ADC triggering event through the LPTIM channels in addition to EXTI on channel 11 and 15.

Triggering signals

For ADC1, the input triggering signals and the description of the interconnection between ADC1, and timers, are given in:

- adc_ext_trgy: *Table 233: External triggers for regular channels*
- adc_jext_trgy: *Table 234: External triggers for injected channels*
- *Section 29.4.19: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN[1:0], JEXTSEL, JEXTEN[1:0])*
- *Section 29.4.25: Timing diagrams example (single/continuous modes, hardware/software triggers)*

For ADC4, the input triggering signals list and the description of the interconnection between ADC4 and timers, are given in:

- *Table 253: ADC interconnection*
- *Section 30.4.16: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN)*
- *Section 30.4.21: Example timing diagrams (single/continuous modes hardware/software triggers)*

Active power mode

This interconnection is active in Run and Sleep modes for all ADCs, and under Stop 0, Stop 1 and Stop 2 for ADC4 assuming that its trigger event line is active as well (such as LPTIM). The timers are active in Run and Sleep mode only. The effects of low-power modes are given in:

- *Table 443: Effect of low-power modes on TIM1/TIM8*
- *Table 462: Effect of low-power modes on TIM2/TIM3/TIM4/TIM5*
- *Table 477: Effect of low-power modes on TIM15/TIM16/TIM17*
- *Section 30.5: ADC low-power modes* for ADC4
- *Table 489: STM32U575/585 LPTIM features*
- *Table 504: Effect of low-power modes on the LPTIM*

16.3.3 ADC analog watchdogs as triggers to timers

From ADC1/ADC4 to TIM1/TIM3/TIM8

Purpose

The internal analog watchdog output signals coming from ADC1/ADC4, are connected to on-chip timers. ADC1/ADC4 can provide trigger event through analog watchdog signals to advanced-control timers (TIM1/TIM3/TIM8) in order to reset, start, stop or clock the counter.

Settings description of the ADC analog watchdog and timer trigger, are provided in:

- *Section 46.3.6: External trigger input* for TIM1/TIM8
- *Table 431* for the internal ADC1/ADC4 sources connected to TIM1/TIM8 (tim_etr) input multiplexer
- *Table 455* for the internal ADC1 sources connected to TIM3 (tim_etr) input multiplexer
- *Section 29.4.30* for the ADC1/ADC_AWDy_OUT signal output generation
- *Section 30.4.25* for the ADC4/ADC_AWDy_OUT signal output generation

Triggering signals

The output (from ADC) is on signals ADCn_AWDx_OUT, with n being the ADC instance and x = 1, 2, 3 (three watchdogs per ADC). The input (to timer) is on signal TIMx_ETR (external trigger).

Active power mode

ADC1 and ADC4 are active in Run and Sleep modes.

The ADC4 conversion is functional and autonomous in Stop 0, Stop 1 and Stop 2 modes. This conversion can generate a wakeup interrupt and desired trigger action on timers.

16.3.4 Triggers to DAC

From timer (TIM1/TIM2/TIM4/TIM5/TIM6/TIM7/TIM8/TIM15), Low-power timers (LPTIM1/LPTIM3) and EXTI to DAC (DAC1/DAC2)

Purpose

General-purpose timers (TIM2/TIM4/TIM5/TIM15), basic timers (TIM6/TIM7), advanced-control timers (TIM1/TIM8), LP timers (LPTIM1/LPTIM3) outputs channels (lptim1_ch1 and lptim3_ch1) and EXTI can be used as triggering event to start a DAC conversion.

Triggering signals

The output (from timer) on the TIMx_TRGO signal and from LP timers are directly connected to corresponding DAC inputs.

The selection of input triggers on DAC is provided in:

- *Table 266: DAC interconnection*
- *Section 31.4.8: DAC trigger selection*

Active power mode

This interconnect is active in Run, Sleep, Stop 0, Stop 1 and Stop 2 modes.

16.3.5 Triggers on MDF1 or ADF1

From EXTI, ADF1, timers (TIM1/TIM3/TIM4/TIM6/TIM7/TIM8/TIM16) and LP timer (LPTIM1), to MDF1

From EXTI and MDF1 to ADF1

Purpose

General-purpose timers (TIM3/TIM4/TIM16), basic timers (TIM6/TIM7), advanced-control timers (TIM1/TIM8), low-power timer (LPTIM1), EXTI (EXTI11/EXTI15) and ADF1 can be used to generate a triggering event on MDF1 module and start an A/C conversion. In addition, EXTI15 and MDF1 can trigger ADF1.

A description is given in:

- *Section 35.4.2: MDF pins and internal signals*
- *Section 35.4.11: Start-up sequence examples*

Triggering signals

The mdf_trgi[13:0] trigger inputs are the triggering input signals. The MDF and ADF trigger inputs connections are detailed in:

- *Table 293: MDF trigger connections*
- *Table 314: ADF trigger connections*

Active power mode

This interconnection remains active down to Stop 0 and Stop 1 for MDF1, and Stop 0, Stop 1 and Stop 2 for ADF1, assuming the trigger source remains active.

16.3.6 Timer break from MDF1

From MDF1 to advanced-control timer (TIM1/TIM8)

Purpose

The MDF features an out-off limit detectors (OLD) and a short circuit detector (SCD) functions. When a short-circuit or open-circuit errors (such as over current or over voltage) is detected an interrupt event or/and a break signal can be generated to TIM1 and TIM8. This behavior is described in:

- *Section 35.4.7: Short-circuit detectors (SCD)*
- *Section 35.4.9: Out-of-limit detector (OLD)*
- *Section 46.3.18: Using the break function*
- *Section 46.3.19: Bidirectional break inputs*

Triggering signals

The mdf1_break[0:3] output signals are connected to break1 and break2 inputs signals of TIM1/TIM8. The tables below gives the assignment of break signals:

- *Table 294: MDF break connections*
- *Table 432: Timer break interconnect*
- *Table 433: Timer break2 interconnect*

Active power mode

This interconnection is active under Run and Sleep modes. Refer to:

- *Section 35.4.14: Autonomous mode*
- *Table 305: Effect of low-power modes on MDF*

16.3.7 Clock sources to timers

From HSE, LSE, LSI, MSIK, HSI and MCO to timers (TIM1/TIM8/TIM2/TIM3/TIM4/TIM5/TIM15/TIM16/TIM17) and LP timers (LPTIM1/LPTIM2/LPTIM3)

Purpose

A timer input or timer counter can receive different clock sources and can be used to calibrate internal oscillator on a reference clock for example.

External clocks (HSE, LSE), internal clocks (LSI, MSI, HSI) and microcontroller output clock (MCO) can be used as input to timers:

- MSIK/HSI are assigned to advanced-control timers TIM1 and TIM8 as external trigger signals inputs (tim_etr3/ tim_etr4). MSIK/HSI can be selected as counter clock provided by an external clock source in mode2: external trigger input (tim_etr_in). Inputs assignment and clock selection description are detailed in:
  - *Section 46.3.7: Clock selection* for TIM1/TIM8
  - *Table 431: Interconnect to the tim_etr input multiplexer* for TIM1/TIM8
- MSIK, HSI and LSI are assigned to general purpose timers TIM2/TIM3/TIM4/TIM5 as external inputs signals. MSIK/HSI/LSI can be selected as counter clock provided by an external clock source in mode1 (tim_ti1_in) and mode2 (external trigger input tim_etr_in). Inputs assignment and clock selection description are detailed in:
  - *Section 47.4.5: Clock selection* for TIM2/TIM3/TIM4/TIM5
  - External clock mode1: *Table 450: Interconnect to the tim_ti1 input multiplexer* for TIM5, tim_ti1_in1 (LSI) and tim_ti1_in2 (LSE)
  - External clock mode2: *Table 455: Interconnect to the tim_etr input multiplexer*, tim_etr3 (MSIK), tim_etr4 (HSI) and tim_etr5 (MSIS) for TIM2/TIM3/TIM4/TIM5
- LSE, LSI, MSI and HSI are assigned to general purpose timers TIM15/TIM16/TIM17 as external inputs signals. LSE/LSI/MSI/HSI can be selected as counter clock provided by an external clock source in mode1 (tim_ti1 or tim_ti2 signals). Inputs assignment and clock selection description are detailed in:
  - *Table 48.4.6: Clock selection* for TIM15/TIM16/TIM17. External clock mode1: external input pin (tim_ti1 or tim_ti2, if available)
  - *Table 468: Interconnect to the tim_ti1 input multiplexer*, tim_ti1_in1 (LSE-TIM15), tim_ti1_in5(LSE-TIM16/TIM17), tim_ti1_in6 (LSI- TIM16/TIM17), tim_ti1_in7/_8 (MSI-TIM16/TIM17) and tim_ti1_in9 (HSI-TIM16/TIM17)
- Microcontroller output clock (MCO) is connected as external input to general-purpose timers TIM16/TIM17. This allows the calibration of 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 the low-speed external (LSE) oscillator is used, no additional hardware connections are required. This feature is given in:
  - *Section 11.4.19: Internal/external clock measurement with TIM15/TIM16/TIM17*

  - *Table 468: Interconnect to the tim_ti1 input multiplexer* for TIM15/TIM16/TIM17
- LSI and LSE can be selected as input capture 2 to LPTIM1 as described in *Table 498: LPTIM1 input capture 2 connection*.
- HSI/256 can be selected as input capture 2 to LPTIM2 as described in *Table 499: LPTIM2 input capture 2 connection*.
- MSI/1024 and MSI/4 can be selected as input capture 2 to LPTIM3 as described in *Table 500: LPTIM3 input capture 2 connection*.

Triggering signals

lptim_ic2_mux1 LPTIM input capture selection can be set in the LPTIM configuration register 2 (LPTIM_CFGR2). For timers, the internal clock signal can be selected as counter clock provided by an external clock source in mode1 (tim_ti1_in) and mode2 (external trigger input tim_etr_in).

Active power mode

This feature is available under Run and Sleep modes.

16.3.8 Triggers to low-power timers

From comparators (COMP1/COMP2), EXTI, TAMP and RTC alarm to LP timers (LPTIM1/LPTIM2/LPTIM3/LPTIM4)

Purpose

LPTIM1/LPTIM2/LPTIM3/LPTIM4 counters may be started either by software or after the detection of an active edge on one of the eight trigger inputs (see *Section 50.4.7: Trigger multiplexer*).

GPIO can also be selected as LPTIM input capture selection or LPTIM input selection, according to the LPTIM configuration register 2 (LPTIM_CFGR2).

Triggering signals

This trigger feature is described in *Section 50.4.7: Trigger multiplexer* and the following sections. The input selection is described in *Table 494: LPTIM1/2/3/4 external trigger connection*.

Active power mode

This interconnection remains active down to Stop 2 mode.

16.3.9 Blanking sources to comparators

From timers (TIM1/TIM8/TIM2/TIM3/TIM4/TIM5/TIM15/TIM16/TIM17) to comparators (COMP1/COMP2)

Purpose

Advanced-control timers (TIM1/TIM8) and general-purpose timers (TIM2/TIM3/TIM4/TIM5/TIM15/TIM16/TIM17) can be used as blanking window input to COMP1/COMP2.

The blanking function is described in *Section 33.3.6: Comparator output-blanking function*.

The blanking sources are given in:

- *COMP1 control and status register (COMP1_CSR)*, BLANKSEL[4:0]
- *COMP2 control and status register (COMP2_CSR)*, BLANKSEL[4:0]

Triggering signals

Timer output signal TIMx_Ocx are the inputs to blanking source of COMP1/COMP2.

Active power mode

This feature is available under Run and Sleep modes.

16.3.10 RTC wakeup as inputs to timers

From RTC to timers (TIM5/TIM16/TIM17)

Purpose

RTC wakeup interrupt can be used as input to general-purpose timers (TIM5/TIM16/TIM17) channel 1.

Triggering signals

RTC wakeup signal is connected to tim_ti1_in3 signal as described in *Table 450: Interconnect to the tim_ti1 input multiplexer* for TIM5.

RTC wakeup signal is connected to tim_ti1_in4 signal as described in *Table 468: Interconnect to the tim_ti1 input multiplexer* for TIM16/TIM17.

Active power mode

This interconnection is active down to Stop 3. Timers are not active but the count is performed at wakeup.

16.3.11 USB OTG SOF as trigger to timers

From USB OTG SOF to TIM2

Purpose

USB OTG SOF (start-of-frame) can generate a trigger to the general-purpose timer TIM2. The USB connection to TIM2 is described in *Table 454: TIMx internal trigger connection*.

Triggering signals

The tim_itr11 internal signal is generated by USB SOF.

Active power mode

This interconnection is active in Run and Sleep modes.

16.3.12 Comparators as inputs, trigger or break signals to timers

From comparators to timers (TIM1/TIM2/TIM3/TIM4/TIM5/TIM8/TIM15/TIM16/TIM17)

Purpose

The comparators (COMP1/COMP2) output values can be connected to timers (TIM1/TIM2/TIM3/TIM4/TIM5/TIM8/TIM15/TIM16/TIM17) input captures, TIMx_ETR or timer break signals. The connection to ETR is described in *Section 46.3.6: External trigger input*.

Comparators (COMP1/COMP2) output values can also generate break input signals for timers (such as TIM1 or TIM8). The sources for break (tim_brk) channel are one of the following:

- external: connected to one of the TIMx_BKIN pin (as per selection done in the AFIO controller) with polarity selection and optional digital filtering
- internal: coming from comparators, tim_brk_cmpx input (refer to *Section 46.3.2: TIM1/TIM8 pins and internal signals* for product specific implementation).

Triggering signals

The tim_etr and tim_brk signals connected TIM1 and TIM8 (coming from COMP1 and COMP2) are given in:

- tim_etr (*Table 431: Interconnect to the tim_etr input multiplexer*): external trigger internal input bus

  These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulse width control.
- tim_brk (*Table 432: Timer break interconnect* and *Table 433: Timer break2 interconnect*)
- *Section 46.3.6: External trigger input*
- *Section 46.3.18: Using the break function*

For TIM2/TIM3/TIM4/TIM5, the sources connected to the tim_ti[1..4] input multiplexers coming from comparators and some other peripherals, are given in:

- *Table 450: Interconnect to the tim_ti1 input multiplexer*
- *Table 451: Interconnect to the tim_ti2 input multiplexer*
- *Table 453: Interconnect to the tim_ti4 input multiplexer*
- *Table 431: Interconnect to the tim_etr input multiplexer*
- *Table 456: Interconnect to the tim_ocref_clr input multiplexer*
- *Section 47.4.22: Timers and external trigger synchronization*
- *TIMx timer input selection register (TIMx_TISEL)(x = 2 to 5)*

For TIM15/TIM16/TIM17, the sources connected to timers coming from comparators and other peripherals are given in:

- *Table 469: Interconnect to the tim_ti2 input multiplexer*
- *Table 471: Timer break interconnect*
- *Table 473: Interconnect to the ocref_clr input multiplexer*
- *Section 48.4.15: Using the break function*
- *Section 48.4.23: External trigger synchronization (TIM15 only)*

Active power mode

Run, Sleep and wakeup capability in Stop 0, Stop 1 and Stop 2 modes for trigger sources. Input and break remain active in same low-power modes as timers activity, on Run and Sleep modes.

16.3.13 System errors as break signals to timers

From system errors to timers (TIM1/TIM8/TIM15/TIM16/TIM17)

Purpose

CSS, CPU lockup, SRAM2/SRAM3 ECC double errors, FLASH ECC double-error detection and 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.

Triggering signals

The possible sources of break are described in:

- *Section 46.3.18: Using the break function* for TIM1/TIM8
- *Section 48.4.15: Using the break function* for TIM15/TIM16/TIM17
- *Table 434: System break interconnect* for TIM1/TIM8
- *Table 472: System break interconnect* for TIM15/TIM16/TIM17

Active power mode

Timers are optionally active in Run and Sleep modes. The effects of low-power modes on TIMx are given in:

- *Table 443: Effect of low-power modes on TIM1/TIM8*
- *Table 462: Effect of low-power modes on TIM2/TIM3/TIM4/TIM5*
- *Table 477: Effect of low-power modes on TIM15/TIM16/TIM17*

16.3.14 Timers generating IRTIM signal

From timers (TIM16/TIM17) to IRTIM

The general-purpose timer (TIM16/TIM17) output channels TIMx_OC1, are used to generate the waveform of the infrared signal output.

This functionality is described in *Section 51: Infrared interface (IRTIM)*.

16.3.15 Triggers for communication peripherals

From LP timers (LPTIM1/LPTIM2/LPTIM3), comparators (COMP1/2), GPDMA1 transfer complete, LPDMA1 transfer complete, EXTI, RTC alarm and RTC wakeup to USART1/2/3, UART4/UART5, LPUART1, I2C1/2/3/4, and SPI1/2/3

Purpose

LP timer (LPTIM1/LPTIM2/LPTIM3) output channels (lptim1_ch1, lptim1_ch2 and lptim3_ch1), comparator (COMP1/COMP2) output channels (comp1_out and comp2_out),

EXTI, RTC alarm and RTC wakeup, can be used as trigger to start a communication on the selected USART, UART, LPUART, I2C or SPI peripheral.

A GPDMA1 transfer complete can trigger both the GPDMA1 regular or linked-list new transfers and communication on selected communication peripheral.

A LPDMA1 transfer complete can trigger both the LPDMA1 new transfers and the communication on selected peripheral.

Triggering signals

The outputs from triggers are directly connected to peripheral trigger inputs.

The selection of input triggers is detailed in:

- *Table 559: USART interconnection (USART1/2/3 and UART4/5)*
- *Table 571: LPUART interconnections (LPUART1)*
- *Table 539: I2C1, I2C2, I2C4 interconnection* and *Table 540: I2C3 interconnection*

The outputs (from timer) are directly connected to SPI1/SPI2/SPI3 inputs on signals spi_itrx (x = 6, 7). The selection of input triggers on SPI is provided in:

- *Table 582: SPI interconnection (SPI1 and SPI2)*
- *Table 583: SPI interconnection (SPI3)*

Active power mode

These interconnections remain active in Run, Sleep and Stop modes if both source and communication line are autonomous under the mode. Refer to:

- *Section 57.6: USART in low-power modes*
- *Section 56.5: I2C low-power modes*
- *Section 59.6: SPI low-power modes*

16.3.16 Triggers to GPDMA/LPDMA

From EXTI, RTC (alarm/wakeup), TAMP (TAMP1/TAMP2/TAMP3), timers (TIM2/TIM15), LP timers (LPTIM1/LPTIM3/LPTIM4), comparators (COMP1/COMP2), LP/GPDMA1 transfer complete (gpdma1_chx_tc/lpdma1_chx_tc), ADC1/ADC4, and analog watchdog to LPDMA1 and GPDMA1

Purpose

An LP/GPDMA trigger can be assigned to an LP/GPDMA channel x. A programmed LP/GPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event. The trigger mode can also be programmed to condition the LLI link transfer. More details are given in the sections below:

- *Section 18.3.5: LPDMA triggers* and *Section 17.3.5: GPDMA triggers*
- *Section 18.4.12: LPDMA triggered transfer* and *Section 17.4.12: GPDMA triggered transfer*
- *LPDMA channel x transfer register 2 (LPDMA_CxTR2)* and *GPDMA channel x transfer register 2 (GPDMA_CxTR2)* for more details on:
  - Trigger selection TRIGSEL[5:0] field
  - Trigger mode (LLI) defined by TRIGM[1:0].
  - Trigger polarity as defined by TRIGPOL[1:0]

Triggering signals

LPDMA trigger mapping is specified in *Table 136: Programmed LPDMA1 trigger*, according to LPDMA_CxTR2.TRIGSEL[5:0].

GPDMA trigger mapping is specified in *Table 126: Programmed GPDMA1 trigger*, according to GPDMA_CxTR2.TRIGSEL[5:0].

Active power mode

Assuming sources are active down to Stop modes, this interconnection remains functional in Stop 0 and Stop 1 for GPDMA, and Stop 0, Stop 1 and Stop 2 for LPDMA.

Refer to:

- *Section 18.6: LPDMA in low-power modes*
- *Section 17.6: GPDMA in low-power modes*

16.3.17 Internal analog signals to analog peripherals

From internal analog source to ADC (ADC1/ADC4), comparators (COMP1/COMP2) and OPAMP (OPAMP1/OPAM2)

Purpose

The internal reference voltage (VREFINT), the internal temperature sensor (VTS) and VBAT monitoring channel are connected to ADC (ADC1/ADC4) input channels. In addition, the internal digital core voltage (VCORE) is connected /to ADC1/ADC4 input channels.

DAC channels (DAC1_OUT/DAC2_OUT) and VREFINT are connected to comparators (COMP1/COMP2).

OPAMP1 and OPAMP2 outputs can be connected to ADC1 or ADC4 input channels through the GPIO. DAC1_OUT1 can be connected to OPAMP1_VINP. DAC1_OUT2 can be connected to OPAMP2_VINP.

Refer to *Table 265: DAC internal input/output signals* for:

- dac_out1 analog output DAC channel1, output, for on-chip peripherals
- dac_out2 analog output DAC channel2, output, for on-chip peripherals

This is according:

- *Section 29.2* and *Section 30.2: ADC main features*
- *Section 29.4.11: Channel selection (SQRx, JSQRx)*
- *Table 286: Operational amplifier possible connections*
- *Section 33.3.2: COMP pins and internal signals*

Active power mode

These interconnections remain in Stop modes if the selected peripheral is kept active. Refer to:

- *Section 30.5: ADC low-power modes*
- *Section 33.4: COMP low-power modes*
- *Section 34.4: OPAMP low-power modes*

16.3.18 ADC data filtering by the MDF1

From ADC1 to MDF1

Purpose

The MDF1 allows the connection of up to two ADCs to the filter path. For each filter, the DATSRC[1:0] field in the MDF digital filter configuration register x (MDF_DFLTxCICR) is used to select either data from the ADCs in:

- *Section 29.4.4: ADC connectivity*
- *Table 295: MDF ADC data connections*
  - mdf_adcitf1_dat[15:0] to adc1_dat
- *Table 232: Interconnection*
  - adcx_dat[15:0] (x = 1) to mdf1_adcx_dat[15:0]
- *Section 35.5: MDF low-power modes*
  - Stop0 and Stop1 modes

Active power mode

This feature remains available down to Sleep mode.

16.3.19 Clock source for the DAC sample and hold mode

LSI/LSE to DAC1/DAC2

Purpose

DAC1/DAC2 can run in Stop mode. The sample and hold block and its associated registers use the LSI or LSE clock source (dac_hold_ck) in Stop mode.

Table 265: DAC internal input/output signals:

dac_hold_ck, Input, DAC low-power clock used in sample and hold mode

Active power mode

This feature remains available down to Stop 2 mode.

16.3.20 Internal tamper sources

From internal peripherals, clocks or monitoring to tamper.

Purpose

In order to detect any abnormal activity or tentative to corrupt the device, tampers are introduced and alert the system of such undesired event. Different actions can be taken in consequences.

List of tamper sources can be found in *Table 528: TAMP interconnection*.

Active power mode

This interconnection is active in all power modes if the tamper source is activated.

16.3.21 Output from tamper to RTC

From TAMP to RTC

Purpose

The RTC can timestamp a tamper event in order to retrieve history in time of such detection. The RTC can also control GPIOs and set a signal based on tamp or alarm status outside the MCU.

Refer to section *Section 54.3.3: GPIOs controlled by the RTC and TAMP* for more details.

Active power mode

This interconnection remains active in all power modes.

16.3.22 Encryption keys to AES/SAES

From TAMP backup registers, system Flash memory to and in between SAES and AES

Purpose

The encryption mechanism requires an hardware key that must be stored in a protected non-volatile memory. Different approaches are implemented in order to load them in a non-readable way. Tamper backup registers or system Flash can be used to store respectively BHK or RHUK, and to implement a dedicated bus to pass it to the SAES.

Refer to *Section 42.4.9: SAES operation with wrapped keys* for more details.

The AES encryption mechanism (faster than the SAES) can benefit from the sharing key of the SAES. Refer to *Section 41.4.13: AES operation with shared keys* for more details.

Active power mode

AES and SAES are operating under Run and Sleep modes.

17 General purpose direct memory access controller (GPDMA)

17.1 Introduction

The general purpose direct memory access (GPDMA) controller is a bus master and system peripheral.

The GPDMA is used to perform programmable data transfers between memory-mapped peripherals and/or memories via linked-lists, upon the control of an off-loaded CPU.

17.2 GPDMA main features

- Dual bidirectional AHB master
- Memory-mapped data transfers from a source to a destination:
  - Peripheral-to-memory
  - Memory-to-peripheral
  - Memory-to-memory
  - Peripheral-to-peripheral
- Autonomous data transfers during low-power modes (see *Section 17.3.2*)

  Transfer arbitration based on a 4-grade programmed priority at channel level:
  - One high-priority traffic class, for time-sensitive channels (queue 3)
  - Three low-priority traffic classes, with a weighted round-robin allocation for non time-sensitive channels (queues 0, 1, 2)
- Per channel event generation, on any of the following events: transfer complete, half transfer complete, data transfer error, user setting error, link transfer error, completed suspension, and trigger overrun
- Per channel interrupt generation, with separately programmed interrupt enable per event
- 16 concurrent GPDMA channels:
  - Per channel FIFO for queuing source and destination transfers (see *Section 17.3.1*)
  - Intra-channel GPDMA transfers chaining via programmable linked-list into memory, supporting two execution modes: run-to-completion and link step mode
  - Intra-channel and inter-channel GPDMA transfers chaining via programmable GPDMA input triggers connection to GPDMA task completion events
- Per linked-list item within a channel:
  - Separately programmed source and destination transfers
  - Programmable data handling between source and destination: byte-based reordering, packing or unpacking, padding or truncation, sign extension and left/right realignment
  - Programmable number of data bytes to be transferred from the source, defining the block level

  - Linear source and destination addressing: either fixed or contiguously incremented addressing, programmed at a block level, between successive burst transfers
  - 2D source and destination addressing: programmable signed address offsets between successive burst transfers (non-contiguous addressing within a block, combined with programmable signed address offsets between successive blocks, at a second 2D/repeated block level, for a reduced set of channels (see *Section 17.3.1*)
  - Support for scatter-gather (multi-buffer transfers), data interleaving and deinterleaving via 2D addressing
  - Programmable GPDMA request and trigger selection
  - Programmable GPDMA half transfer and transfer complete events generation
  - Pointer to the next linked-list item and its data structure in memory, with automatic update of the GPDMA linked-list control registers
- Debug:
  - Channel suspend and resume support
  - Channel status reporting, including FIFO level, and event flags
- TrustZone support:
  - Support for secure and non-secure GPDMA transfers, independently at a first channel level, and independently at a source/destination and link sublevels
  - Secure and non-secure interrupts reporting, resulting from any of the respectively secure and non-secure channels
  - TrustZone-aware AHB slave port, protecting any GPDMA secure resource (register, register field) from a non-secure access
- Privileged/unprivileged support:
  - Support for privileged and unprivileged GPDMA transfers, independently at a channel level
  - Privileged-aware AHB slave port

17.3 GPDMA implementation

17.3.1 GPDMA channels

A given GPDMA channel x is implemented with the following features and intended usage. To make the best use of the GPDMA performances, the table below lists some general recommendations, allowing the user to select and allocate a channel, given its implemented FIFO size and the requested GPDMA transfer.

Table 122. GPDMA1 channels implementation

Channel x	Hardware parameters		Features
	dma_fifo_ size[x]	dma_ addressing[x]	
x = 0 to 11	2	0	Channel x (x = 0 to 11) is implemented with: – a FIFO of 8 bytes, 2 words – fixed/contiguously incremented addressing These channels must be typically allocated for GPDMA transfers between an APB or AHB peripheral and SRAM.
x = 12 to 15	4	1	Channel x (x = 12 to 15) is implemented with: – a FIFO of 32 bytes, 8 words – 2D addressing These channels may be also used for GPDMA transfers, between a demanding AHB peripheral and SRAM, or for transfers from/to external memories.

17.3.2 GPDMA autonomous mode in low-power modes

The GPDMA autonomous mode and wakeup feature are implemented in the device low-power modes as per the table below.

Table 123. GPDMA1 autonomous mode and wakeup in low-power modes

Feature	Low-power modes
Autonomous mode and wakeup	GPDMA1 in Sleep, Stop 0 and Stop 1 modes

17.3.3 GPDMA requests

A GPDMA request from a peripheral can be assigned to a GPDMA channel x, via the REQSEL[6:0] field in *GPDMA channel x transfer register 2 (GPDMA_CxTR2)*, provided that SWREQ = 0.

The GPDMA requests mapping is specified in the table below.

Table 124. Programmed GPDMA1 request

GPDMA_CxTR2.REQSEL[6:0]	Selected GPDMA request
0	adc1_dma
1	adc4_dma
2	dac1_ch1_dma
3	dac1_ch2_dma
4	tim6_upd_dma
5	tim7_upd_dma
6	spi1_rx_dma
7	spi1_tx_dma

Table 124. Programmed GPDMA1 request (continued)

GPDMA_CxTR2.REQSEL[6:0]	Selected GPDMA request
8	spi2_rx_dma
9	spi2_tx_dma
10	spi3_rx_dma
11	spi3_tx_dma
12	i2c1_rx_dma
13	i2c1_tx_dma
14	i2c1_evc_dma
15	i2c2_rx_dma
16	i2c2_tx_dma
17	i2c2_evc_dma
18	i2c3_rx_dma
19	i2c3_tx_dma
20	i2c3_evc_dma
21	i2c4_rx_dma
22	i2c4_tx_dma
23	i2c4_evc_dma
24	usart1_rx_dma
25	usart1_tx_dma
26	usart2_rx_dma
27	usart2_tx_dma
28	usart3_rx_dma
29	usart3_tx_dma
30	uart4_rx_dma
31	uart4_tx_dma
32	uart5_rx_dma
33	uart5_tx_dma
34	lpuart1_rx_dma
35	lpuart1_tx_dma
36	sai1_a_dma
37	sai1_b_dma
38	sai2_a_dma
39	sai2_b_dma
40	octospi1_dma
41	octospi2_dma
42	tim1_cc1_dma

Table 124. Programmed GPDMA1 request (continued)

GPDMA_CxTR2.REQSEL[6:0]	Selected GPDMA request
43	tim1_cc2_dma
44	tim1_cc3_dma
45	tim1_cc4_dma
46	tim1_upd_dma
47	tim1_trg_dma
48	tim1_com_dma
49	tim8_cc1_dma
50	tim8_cc2_dma
51	tim8_cc3_dma
52	tim8_cc4_dma
53	tim8_upd_dma
54	tim8_trg_dma
55	tim8_com_dma
56	tim2_cc1_dma
57	tim2_cc2_dma
58	tim2_cc3_dma
59	tim2_cc4_dma
60	tim2_upd_dma
61	tim3_cc1_dma
62	tim3_cc2_dma
63	tim3_cc3_dma
64	tim3_cc4_dma
65	tim3_upd_dma
66	tim3_trg_dma
67	tim4_cc1_dma
68	tim4_cc2_dma
69	tim4_cc3_dma
70	tim4_cc4_dma
71	tim4_upd_dma
72	tim5_cc1_dma
73	tim5_cc2_dma
74	tim5_cc3_dma
75	tim5_cc4_dma
76	tim5_upd_dma
77	tim5_trg_dma

Table 124. Programmed GPDMA1 request (continued)

GPDMA_CxTR2.REQSEL[6:0]	Selected GPDMA request
78	tim15_cc1_dma
79	tim15_upd_dma
80	tim15_trg_dma
81	tim15_com_dma
82	tim16_cc1_dma
83	tim16_upd_dma
84	tim17_cc1_dma
85	tim17_upd_dma
86	dcmi_dma
87	aes_in_dma
88	aes_out_dma
89	hash_in_dma
90	ucpd1_tx_dma
91	ucpd1_rx_dma
92	mdf1_flt0_dma
93	mdf1_flt1_dma
94	mdf1_flt2_dma
95	mdf1_flt3_dma
96	mdf1_flt4_dma
97	mdf1_flt5_dma
98	adf1_flt0_dma
99	fmac_read_dma
100	fmac_write_dma
101	cordic_read_dma
102	cordic_write_dma
103	saes_in_dma
104	saes_out_dma
105	lptim1_ic1_dma
106	lptim1_ic2_dma
107	lptim1_ue_dma
108	lptim2_ic1_dma
109	lptim2_ic2_dma
110	lptim2_ue_dma
111	lptim3_ic1_dma

Table 124. Programmed GPDMA1 request (continued)

GPDMA_CxTR2.REQSEL[6:0]	Selected GPDMA request
112	lptim3_ic2_dma
113	lptim3_ue_dma

17.3.4 GPDMA block requests

Some GPDMA requests must be programmed as a block request, and not as a burst request. Then the BREQ bit in *GPDMA channel x transfer register 2 (GPDMA_CxTR2)* must be set for a correct GPDMA execution of the requested peripheral transfer at the hardware level.

Table 125. Programmed GPDMA1 request as a block request

GPDMA block requests
lptim1_ue
lptim3_ue

17.3.5 GPDMA triggers

A GPDMA trigger can be assigned to a GPDMA channel x, via the TRIGSEL[5:0] field in the *GPDMA channel x transfer register 2 (GPDMA_CxTR2)*, provided that TRIGPOL[1:0] defines a rising or a falling edge of the selected trigger (TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

Table 126.Programmed GPDMA1 trigger

GPDMA_CxTR2.TRIGSEL[5:0]	Selected GPDMA trigger
0	exti0
1	exti1
2	exti2
3	exti3
4	exti4
5	exti5
6	exti6
7	exti7
8	tamp_trg1
9	tamp_trg2
10	tamp_trg3
11	lptim1_ch1
12	lptim1_ch2

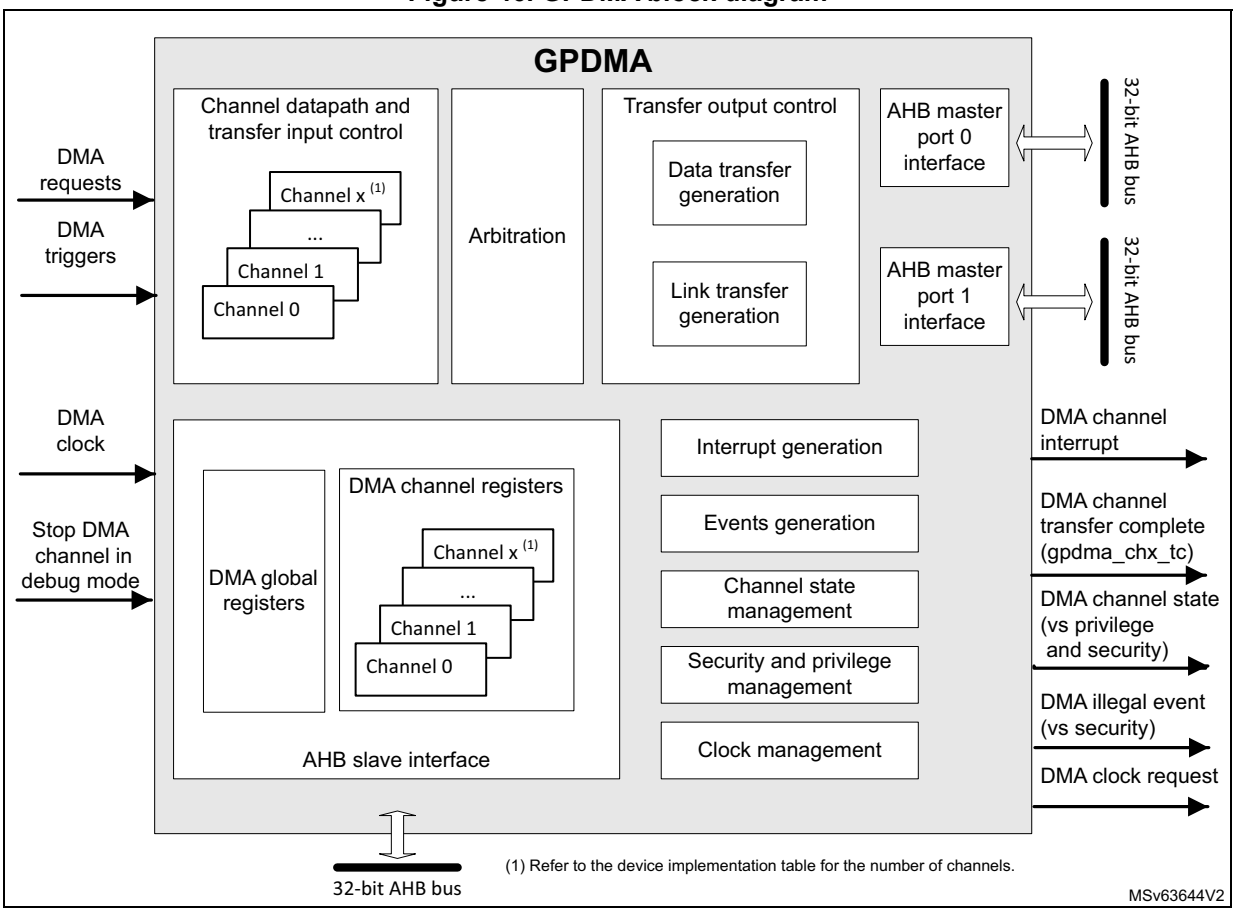
Table 126.Programmed GPDMA1 trigger (continued)

GPDMA_CxTR2.TRIGSEL[5:0]	Selected GPDMA trigger
13	lptim2_ch1
14	lptim2_ch2
15	lptim4_out
16	comp1_out
17	comp2_out
18	rtc_alra_trg
19	rtc_alrb_trg
20	rtc_wut_trg
21	reserved
22	gpdma1_ch0_tc
23	gpdma1_ch1_tc
24	gpdma1_ch2_tc
25	gpdma1_ch3_tc
26	gpdma1_ch4_tc
27	gpdma1_ch5_tc
28	gpdma1_ch6_tc
29	gpdma1_ch7_tc
30	gpdma1_ch8_tc
31	gpdma1_ch9_tc
32	gpdma1_ch10_tc
33	gpdma1_ch11_tc
34	gpdma1_ch12_tc
35	gpdma1_ch13_tc
36	gpdma1_ch14_tc
37	gpdma1_ch15_tc
38	lpdma1_ch0_tc
39	lpdma1_ch1_tc
40	lpdma1_ch2_tc
41	lpdma1_ch3_tc
42	tim2_trgo
43	tim15_trgo
57	adc4_awd1
58	adc1_awd1

17.4 GPDMA functional description

17.4.1 GPDMA block diagram

Figure 46. GPDMA block diagram



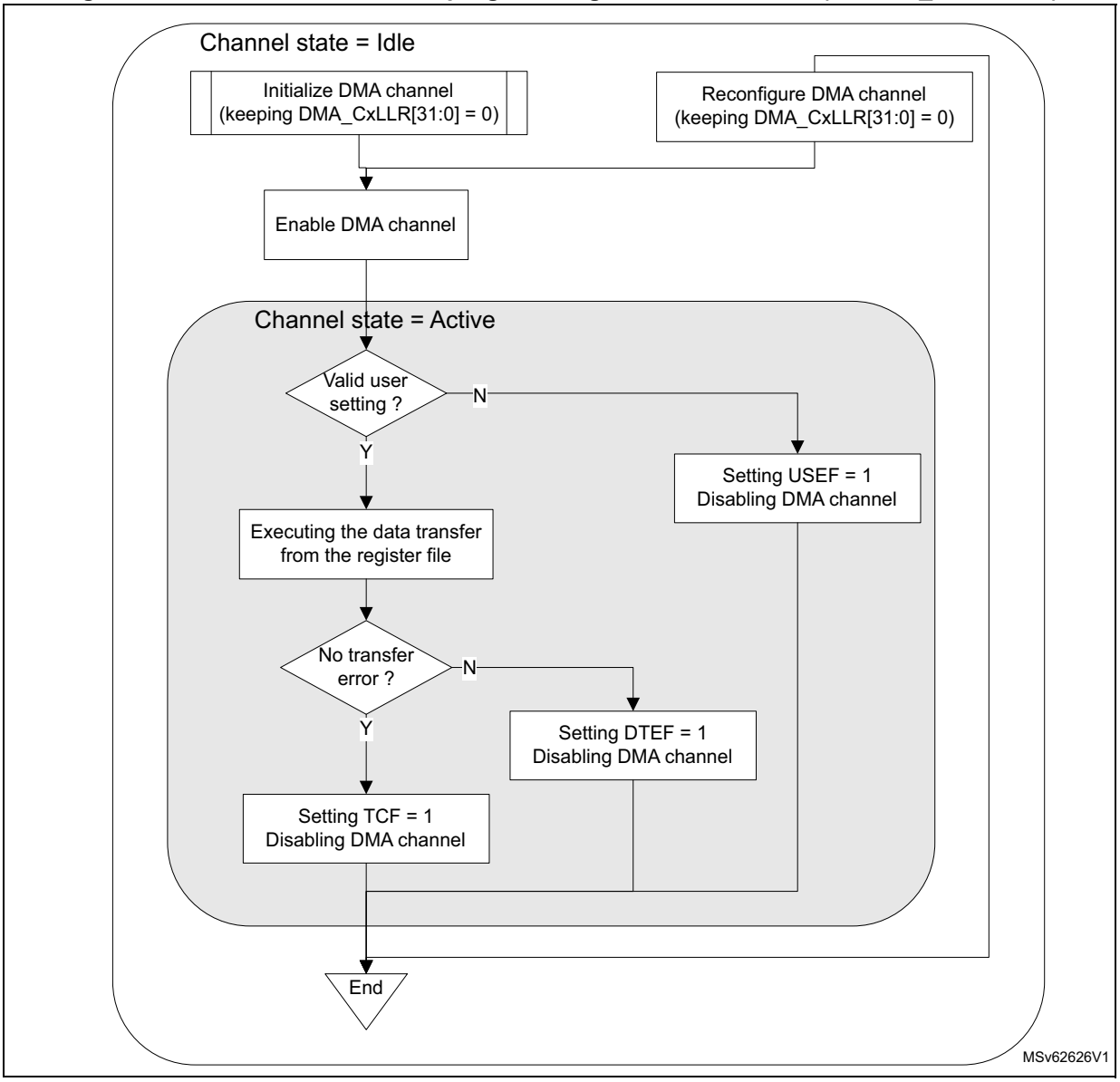
17.4.2 GPDMA channel state and direct programming without any linked-list

After a GPDMA reset, a GPDMA channel x is in idle state. When the software writes 1 into the GPDMA_CxCR.EN enable control bit, the channel takes into account the value of the different channel configuration registers (GPDMA_CxXXX), switches to the active/non-idle state and starts to execute the corresponding requested data transfers.

After enabling/starting a GPDMA channel transfer by writing 1 into the GPDMA_CxCR.EN bit, a GPDMA channel interrupt on a complete transfer notifies the software that the GPDMA channel is back in idle state (EN is then deasserted by hardware) and that the channel is ready to be reconfigured then enabled again.

The figure below illustrates this GPDMA direct programming without any linked-list (GPDMA_CxLLR = 0).

Figure 47. GPDMA channel direct programming without linked-list (GPDMA_CxLLR = 0)



17.4.3 GPDMA channel suspend and resume

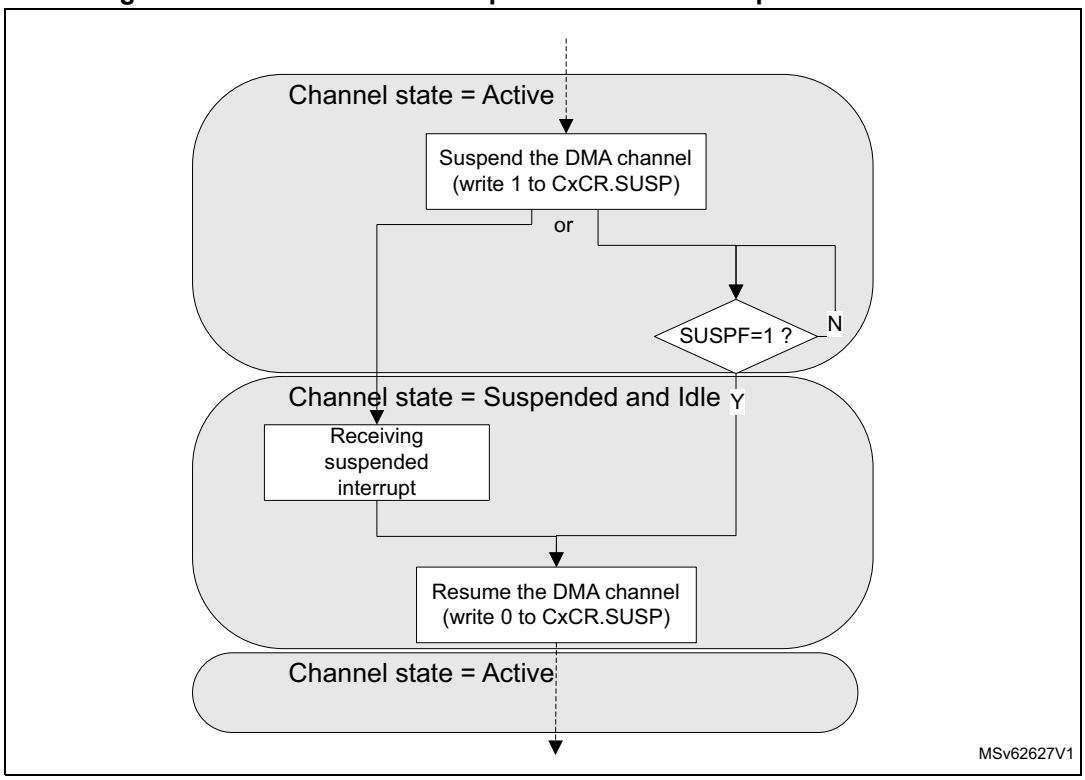
The software can suspend on its own a channel still active, with the following sequence:

1. The software writes 1 into the GPDMA_CxCR.SUSP bit.
2. The software polls the suspended flag GPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing GPDMA transfer over its master ports. Then the software can observe, in a steady state, any read register or register field that is hardware modifiable.

Note that an ongoing GPDMA transfer can be a data transfer (a source/destination burst transfer) or a link transfer for the internal update of the linked-list register file from the next linked-list item.

3. The software safely resumes the suspended channel by writing 0 to GPDMA_CxCR.SUSP.

Figure 48. GPDMA channel suspend and resume sequence



Note: A suspend and resume sequence does not impact the GPDMA_CxCR.EN bit. Suspending a channel (transfer) does not suspend a started trigger detection.

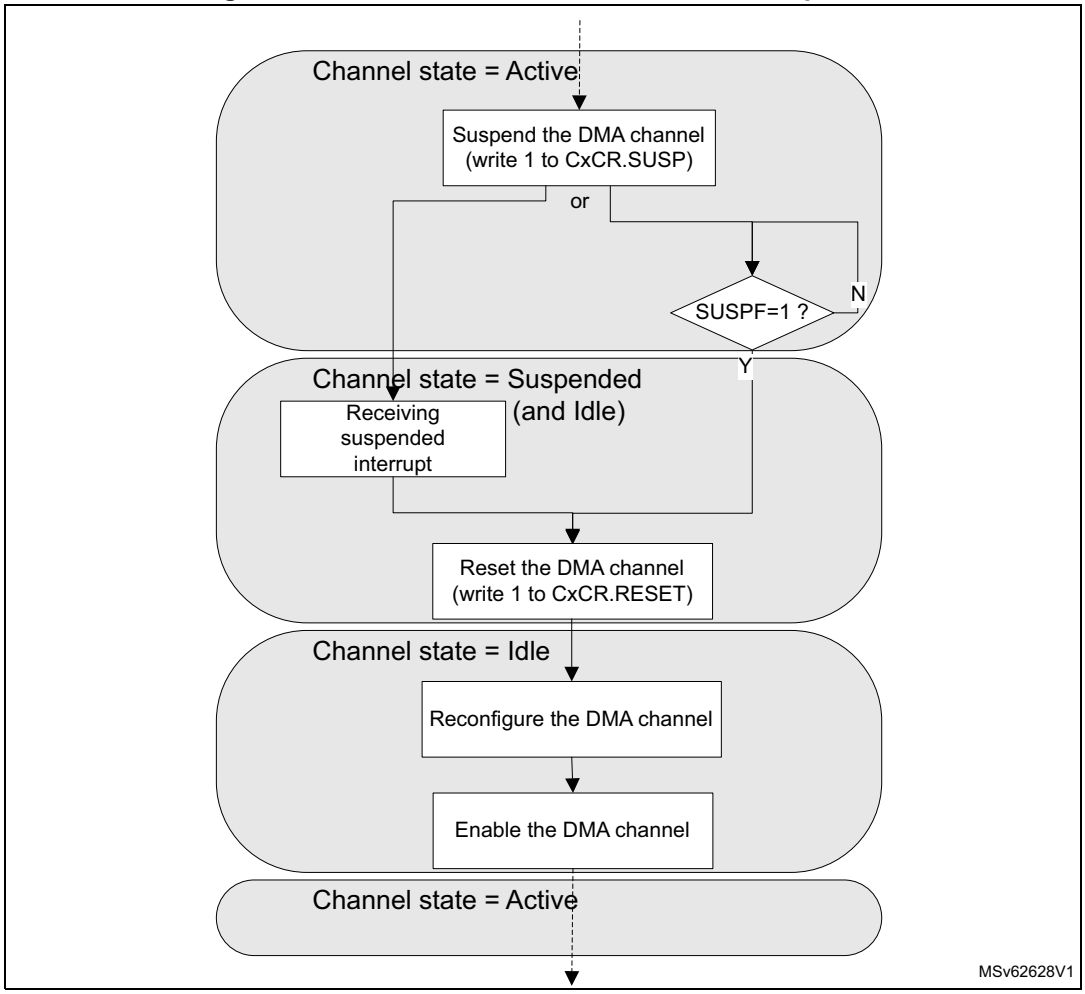
17.4.4 GPDMA channel abort and restart

Alternatively, like for aborting a continuous GPDMA transfer with a circular buffering or a double buffering, the software can abort, on its own, a still active channel with the following sequence:

1. The software writes 1 into the GPDMA_CxCR.SUSP bit.
2. The software polls suspended flag GPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing GPDMA transfer over its master port.
3. The software resets the channel by writing 1 to GPDMA_CxCR.RESET. This causes the reset of the FIFO, the reset of the channel internal state, the reset of the GPDMA_CxCR.EN bit, and the reset of the GPDMA_CxCR.SUSP bit.
4. The software safely reconfigures the channel. The software must reprogram the hardware-modified GPDMA_CxBR1, GPDMA_CxSAR, and GPDMA_CxDAR registers.

5. In order to restart the aborted then reprogrammed channel, the software enables it again by writing 1 to the GPDMA_CxCR.EN bit.

Figure 49. GPDMA channel abort and restart sequence



17.4.5 GPDMA linked-list data structure

Alternatively to the direct programming mode, a channel can be programmed by a list of transfers, known as a list of linked-list items (LLI). Each LLI is defined by its data structure.

The base address in memory of the data structure of a next LLI_{n+1} of a channel x is the sum of the following:

- the link base address of the channel x (in GPDMA_CxLBAR)
- the link address offset (LA[15:2] field in GPDMA_CxLLR). The linked-list register GPDMA_CxLLR is the updated result from the data structure of the previous LLIn of the channel x.

The data structure for each LLI may be specific.

A linked-list data structure is addressed following the value of the UT1, UT2, UB1, USA, UDA and ULL bits, plus UB2 and UT3, in GPDMA_CxLLR.

In linked-list mode, each GPDMA linked-list register (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR or GPDMA_CxLLR, plus GPDMA_CxTR3 or GPDMA_CxBR2) is conditionally and automatically updated from the next linked-list data structure in the memory, following the current value of the GPDMA_CxLLR register that was conditionally updated from the linked-list data structure of the previous LLI.

Static linked-list data structure

For example, when the update bits (UT1, UT2, UB1, USA, UDA and ULL, plus UB2 and UT3) in GPDMA_CxLLR are all asserted, the linked-list data structure in the memory is maximal with:

- channel x (x = 0 to 11) contiguous 32-bit locations, including GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR and GPDMA_CxLLR (see *Figure 50*) and including the first linked-list register file (LLI_0) and the next LLIs (LLI_1, LLI_2,...) in the memory
- channel x (x = 12 to 15) contiguous 32-bit locations, including GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2 (see *Figure 51*), and including the first linked-list register file (LLI_0) and the next LLIs (LLI_1, LLI_2,...) in the memory

Figure 50. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x

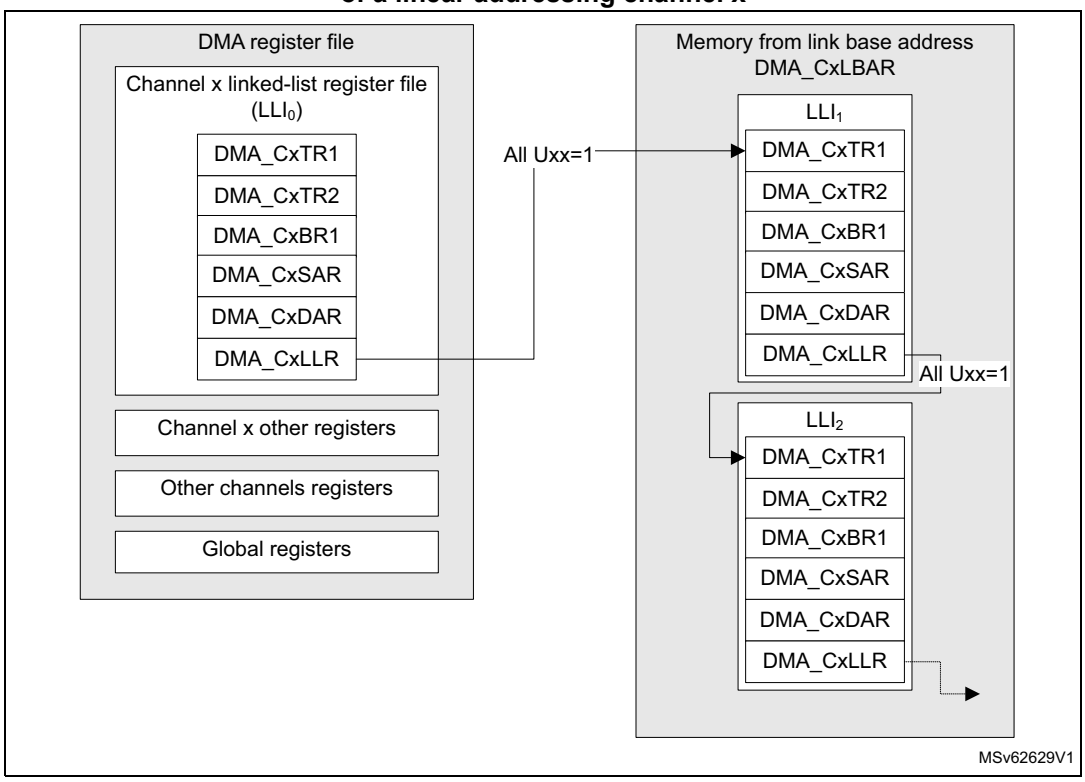
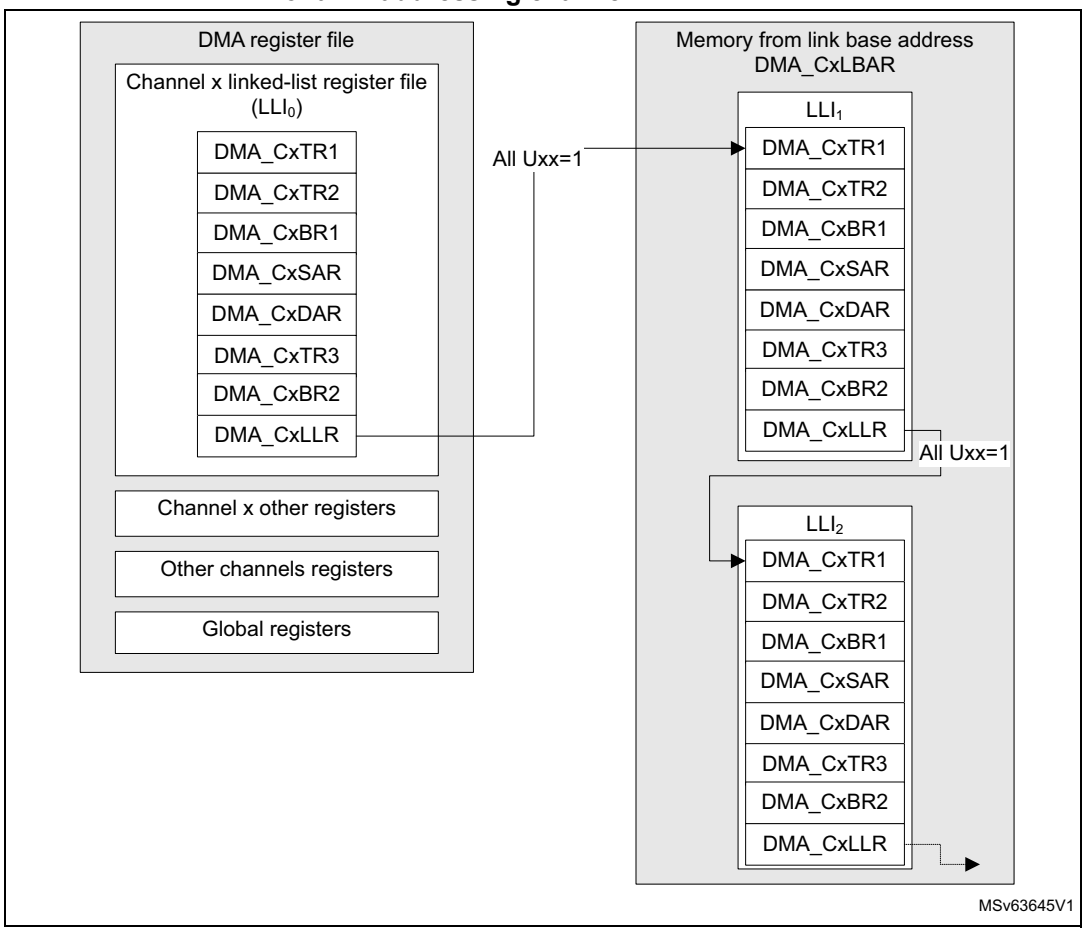


Figure 51. Static linked-list data structure (all Uxx = 1) of a 2D addressing channel x



Dynamic linked-list data structure

Alternatively, the memory organization for the full list of LLIs can be compacted with specific data structure for each LLI.

If UT1 = 0 and UT2 = 1, the link address offset of the register GPDMA_CxLLR is pointing to the updated value of the GPDMA_CxTR2 instead of the GPDMA_CxTR1 which is not to be modified (see *Figure 52*).

Example: if UT1 = UB1 = USA = 0 and if UT3 = UB2 = 0, when channel x is with 2D addressing, and if UT2 = UDA = ULL = 1, the next LLI does not contain an (updated) value for GPDMA_CxTR1, nor GPDMA_CxBR1, nor GPDMA_CxSAR, nor GPDMA_CxTR3, nor GPDMA_CxBR2 when channel x is with 2D addressing. The next LLI contains an updated value for GPDMA_CxTR2, GPDMA_CxDAR, and GPDMA_CxLLR, as shown in *Figure 53*.

Figure 52. GPDMA dynamic linked-list data structure of a linear addressing channel x

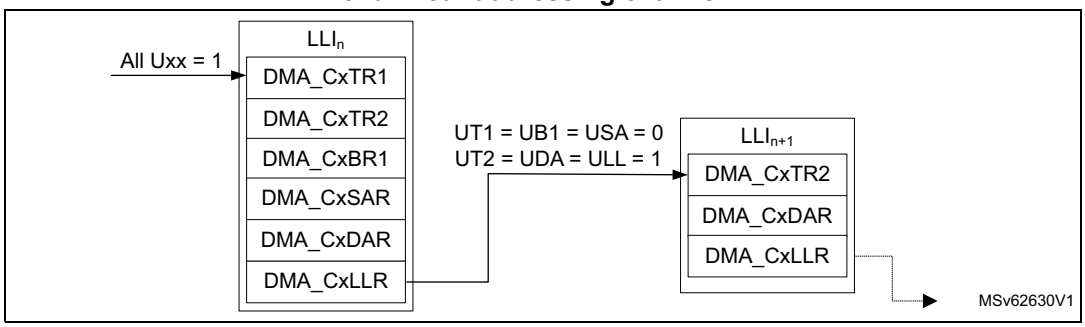
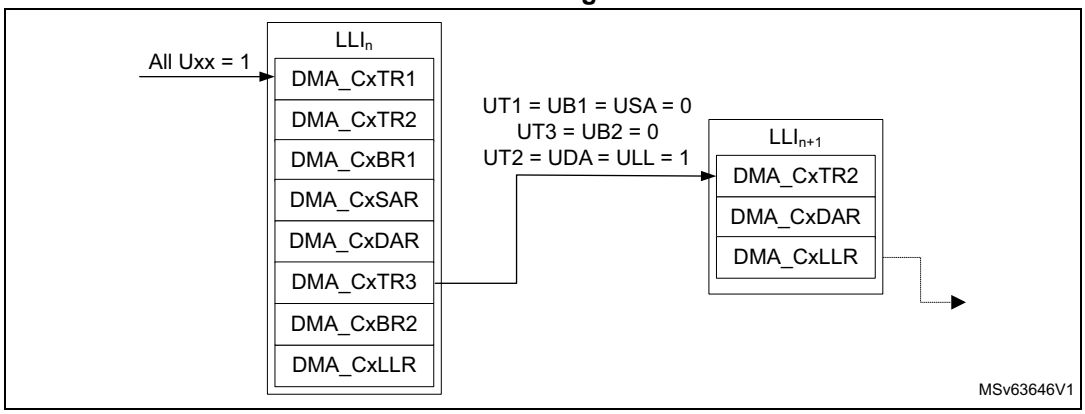


Figure 53. GPDMA dynamic linked-list data structure of a 2D addressing channel x



The user must program GPDMA_CxLLR for each LLI$_n$ to be 32-bit aligned and not to exceed the 64-Kbyte addressable space pointed by GPDMA_CxLBAR.

17.4.6 Linked-list item transfer execution

A LLI$_n$ transfer is the sequence of:

1. a data transfer: GPDMA executes the data transfer as described by the GPDMA internal register file (this data transfer can be void/null for LLI$_0$)
2. a conditional link transfer: GPDMA automatically and conditionally updates its internal register file by the data structure of the next LLI$_{n+1}$, as defined by the GPDMA_CxLLR value of the LLI$_n$.

Note: The initial data transfer as defined by the internal register file (LLI$_0$) can be null (GPDMA_CxBR1.BNDT[15:0] = 0) provided that the conditional update bit UB1 in GPDMA_CxLLR is set (meaning there is a non-null data transfer described by the next LLI$_1$ in the memory to be executed).

Depending on the intended GPDMA usage, a GPDMA channel x can be executed as described by the full linked-list (run-to-completion mode, GPDMA_CxCR.LSM = 0) or a GPDMA channel x can be programmed for a single execution of a LLI (link step mode, GPDMA_CxCR.LSM = 1), as described in the next sections.

17.4.7 GPDMA channel state and linked-list programming in run-to-completion mode

When GPDMA_CxCR.LSM = 0 (in full list execution mode, execution of the full sequence of LLIs, named run-to-completion mode), a GPDMA channel x is initially programmed, started by writing 1 to GPDMA_CxCR.EN, and after completed at channel level. The channel transfer is:

- configured with at least the following:
  - the first LLI_0, internal linked-list register file: GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2
  - the last LLI_N, described by the linked-list data structure in memory, as defined by the GPDMA_CxLLR reflecting the before last LLI_{N-1}
- completed when GPDMA_CxLLR[31:0] = 0, GPDMA_CxBR1.BRC[10:0] = 0, and GPDMA_CxBR1.BNDT[15:0] = 0, at the end of the last LLI_{N-1} transfer

GPDMA_CxLLR[31:0] = 0 is the condition of a linked-list based channel completion and means the following:

- The 16 low significant bits GPDMA_CxLLR.LA[15:0] of the next link address are null.
- All the update bits GPDMA_CxLLR.Uxx are null (UT1, UT2, UB1, USA, UDA and ULL, plus UB2 and UT3).

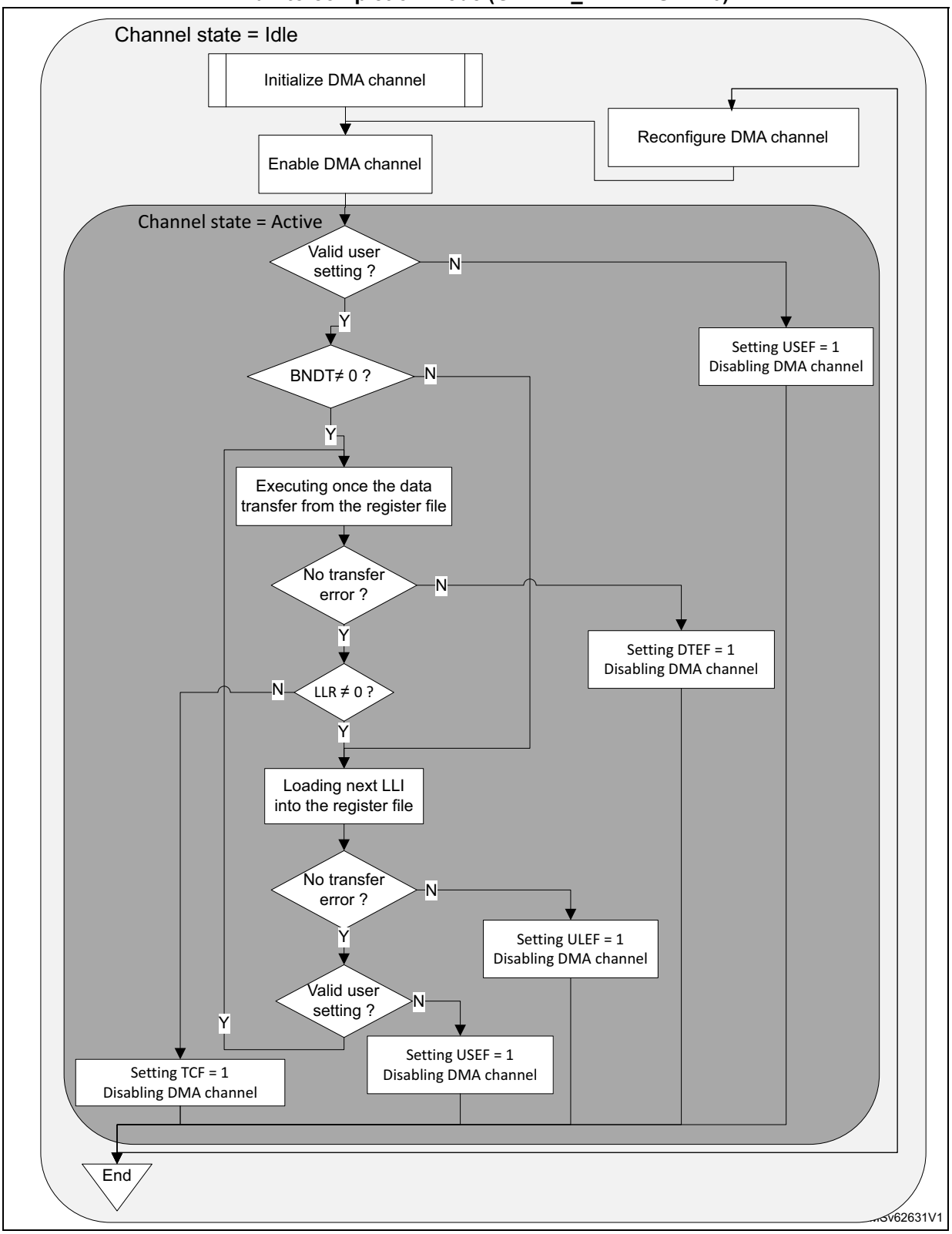
The channel may never be completed when GPDMA_CxLLR.LSM = 0:

- If the last LLI_N is recursive, pointing to itself as a next LLI:
  - either GPDMA_CxLLR.ULL = 1 and GPDMA_CxLLR.LA[15:2] is updated by the same value
  - or GPDMA_CxLLR.ULL = 0
- if LLI_N is pointing to a previous LLI

In the typical run-to-completion mode, the allocation of a GPDMA channel, including its fine programming, is done once during the GPDMA initialization. In order to have a reserved data communication link and GPDMA service during run-time, for continuously repeated transfers (from/to a peripheral respectively to/from memory or for memory-to-memory transfers). This reserved data communication link can consist of a channel, or the channel can be shared and a repeated transfer consists of a sequence of LLIs.

The figure below depicts the GPDMA channel execution and its registers programming in run-to-completion mode.

Figure 54. GPDMA channel execution and linked-list programming in run-to-completion mode (GPDMA_CxCR.LSM = 0)



Run-time inserting a LLI_n via an auxiliary channel, in run-to-completion mode

The start of the link transfer of the LLI_{n-1} (start of the LLI_n loading) can be conditioned by the occurrence of a trigger, when programming the following fields of the GPDMA_CxTR2 in the data structure of the LLI_{n-1}:

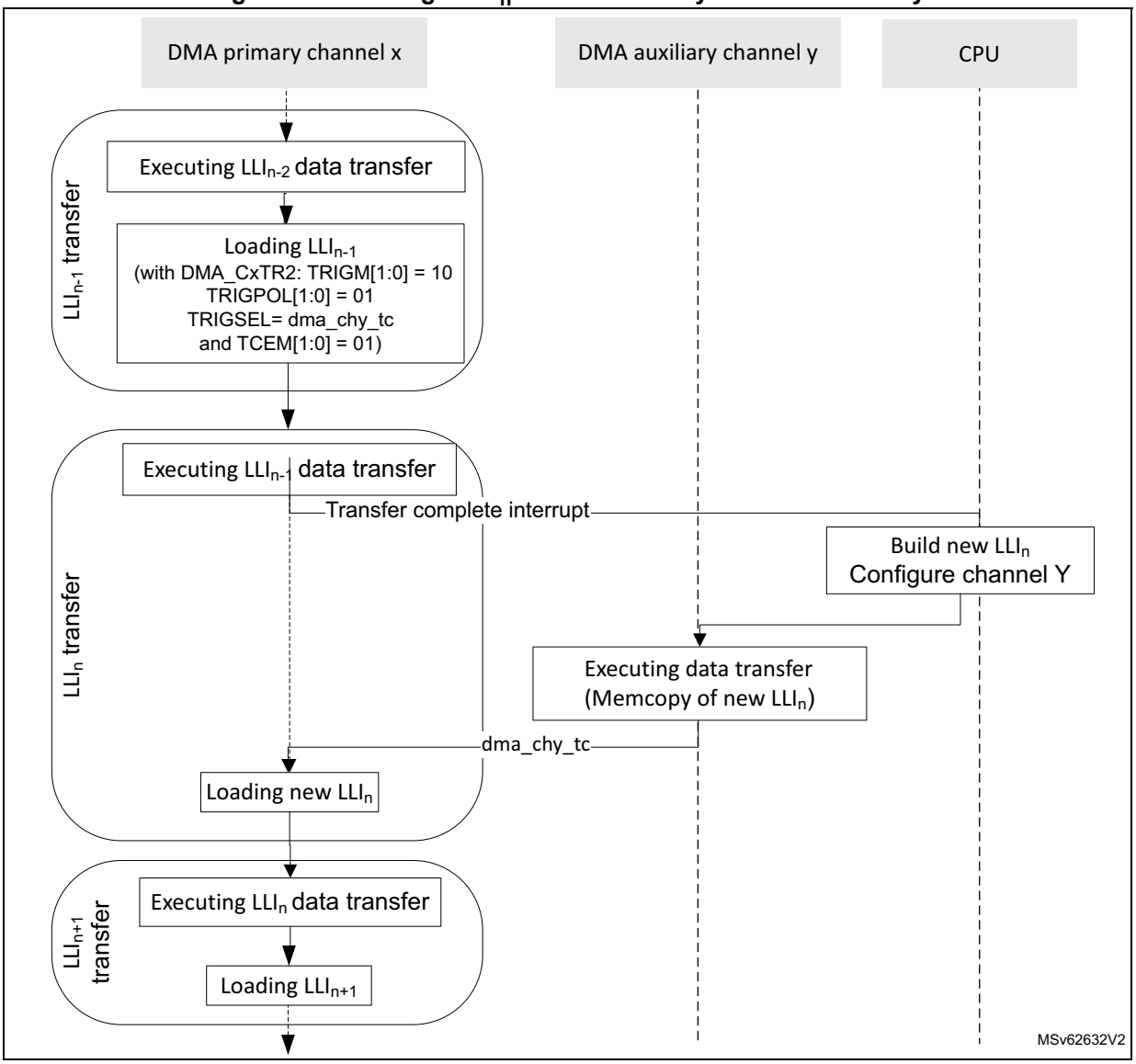
- TRIGM[1:0] = 10 (link transfer triggering mode)
- TRIGPOL[1:0] = 01 or 10 (rising or falling edge)
- TRIGSEL[5:0] (see *Section 17.3.5* for the trigger selection details)

Another auxiliary channel y can be used to store the channel x LLI_n in the memory and to generate a transfer complete event gpdma_chy_tc. By selecting this event as the input trigger of the link transfer of the LLI_{n-1} of the channel x, the software can pause the primary channel x after its LLI_{n-1} data transfer, until it is indeed written the LLI_n.

The figure below depicts such a dynamic elaboration of a linked-list of a primary channel x, via another auxiliary channel y.

Caution: This use case is restricted to an application with a LLI_{n-1} data transfer that does not need a trigger. The triggering mode of this LLI_{n-1} is used to load the next LLI_n.

Figure 55. Inserting a LLI_n with an auxiliary GPDMA channel y



17.4.8 GPDMA channel state and linked-list programming in link step mode

When GPDMA_CxCR.LSM = 1 (in link step execution mode, single execution of one LLI), a channel transfer is executed and completed after each single execution of a LLI, including its (conditional) data transfer and its (conditional) link transfer.

A GPDMA channel transfer can be programmed at LLI level, started by writing 1 into GPDMA_CxCR.EN, and after completed at LLI level:

- The current LLI_n transfer is described with:
  - GPDMA_CxTR1 defines the source/destination elementary single/burst transfers.
  - GPDMA_CxBR1 defines the number of bytes at a block level (BNDT[15:0]) and, for channel x (x = 12 to 15), the number of blocks at a 2D/repeated block level (BRC[10:0]+1) and the incrementing/decrementing mode for address offsets.

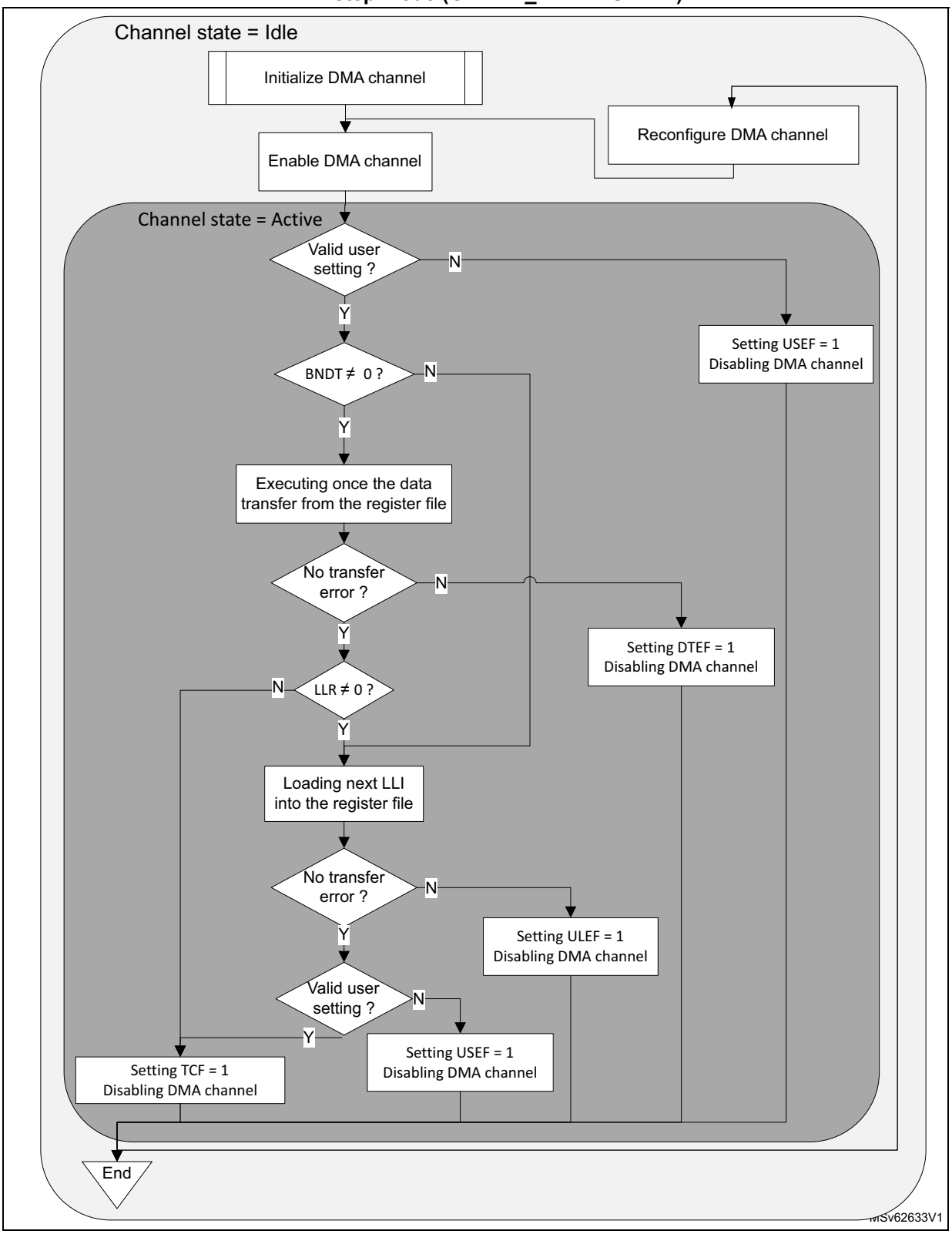
  - GPDMA_CxTR2 defines the input control (request, trigger) and the output control (transfer complete event) of the transfer.
  - GPDMA_CxSAR/GPDMA_CxDAR define the source/destination transfer start address.
  - GPDMA_CxTR3 for channel x (x = 12 to 15) defines the source/destination additional address offset between burst transfers.
  - GPDMA_CxBR2 for channel x (x = 12 to 15) defines the source/destination additional address offset between blocks at a 2D/repeated block level.
  - GPDMA_CxLLR defines the data structure and the address offset of the next LLI_{n+1} in the memory.
- The current LLI_n transfer is completed after the single execution of the current LLI_n:
  - after the (conditional) data transfer completion (when GPDMA_CxBR1.BRC[10:0] = 0, and GPDMA_CxBR1.BNDT[15:0] = 0
  - after the (conditional) update of the GPDMA link register file from the data structure of the next LLI_{n+1} in memory

Note: If a LLI is recursive (pointing to itself as a next LLI, either GPDMA_CxLLR.ULL = 1 and GPDMA_CxLLR.LA[15:2] is updated by the same value, or GPDMA_CxLLR.ULL = 0), a channel in link step mode is completed after each repeated single execution of this LLI.

The link step mode can be used to elaborate dynamically LLIs in memory during run-time. The software can be facilitated by using a static data structure for any LLI_n (all update bits of GPDMA_CxLLR have a static value, LLI_n.LLR.LA = LLI_{n-1}.LLR.LA + constant).

The figure below depicts the GPDMA channel execution mode, and its programming in link step mode.

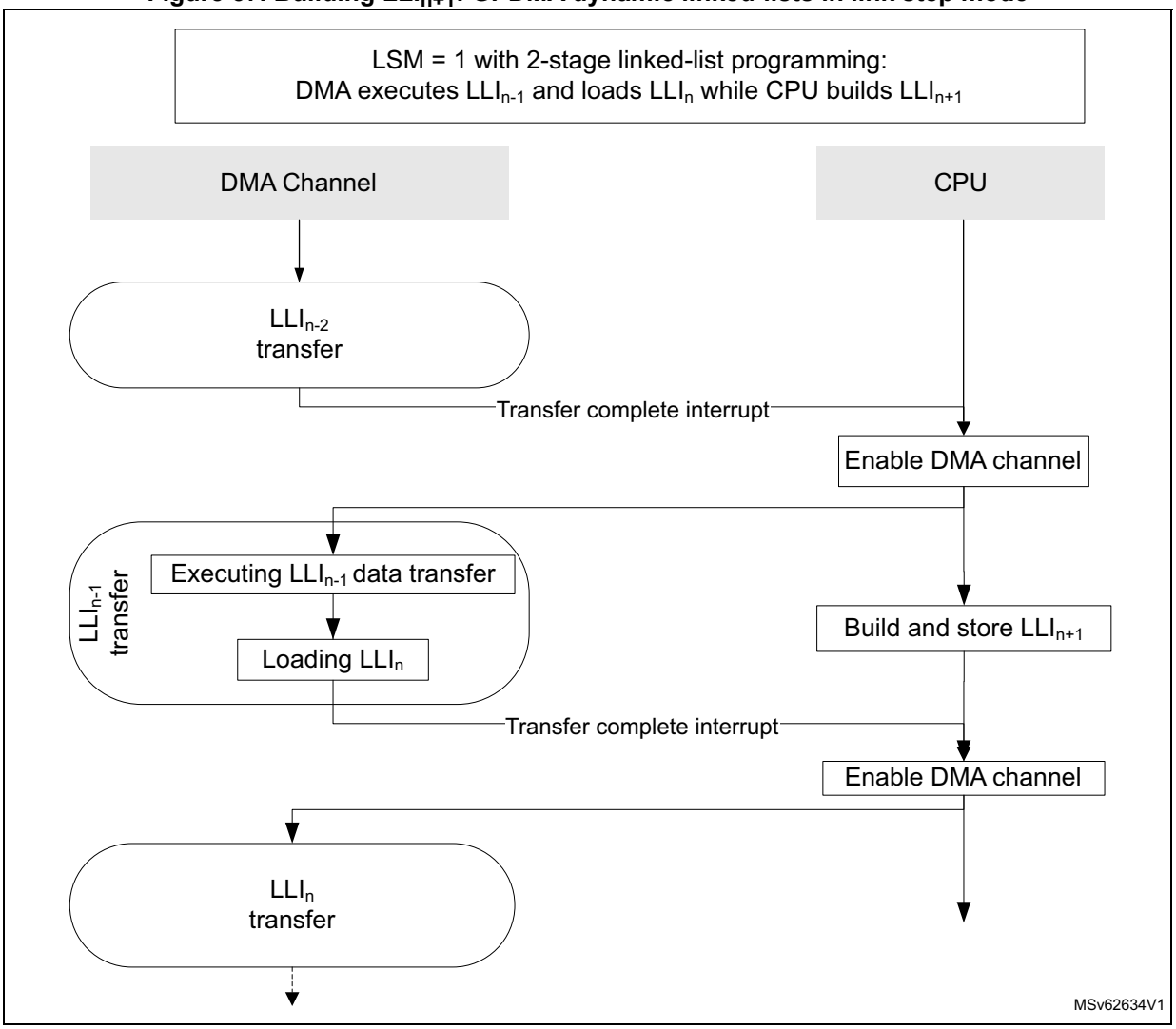
Figure 56. GPDMA channel execution and linked-list programming in link step mode (GPDMA_CxCR.LSM = 1)



Run-time adding a LLI_{n+1} in link step mode

During run-time, the software can defer the elaboration of the LLI_{n+1} (and next LLIs), until/after GPDMA executed the transfer from the LLI_{n-1} and loaded the LLI_n from the memory, as shown in the figure below.

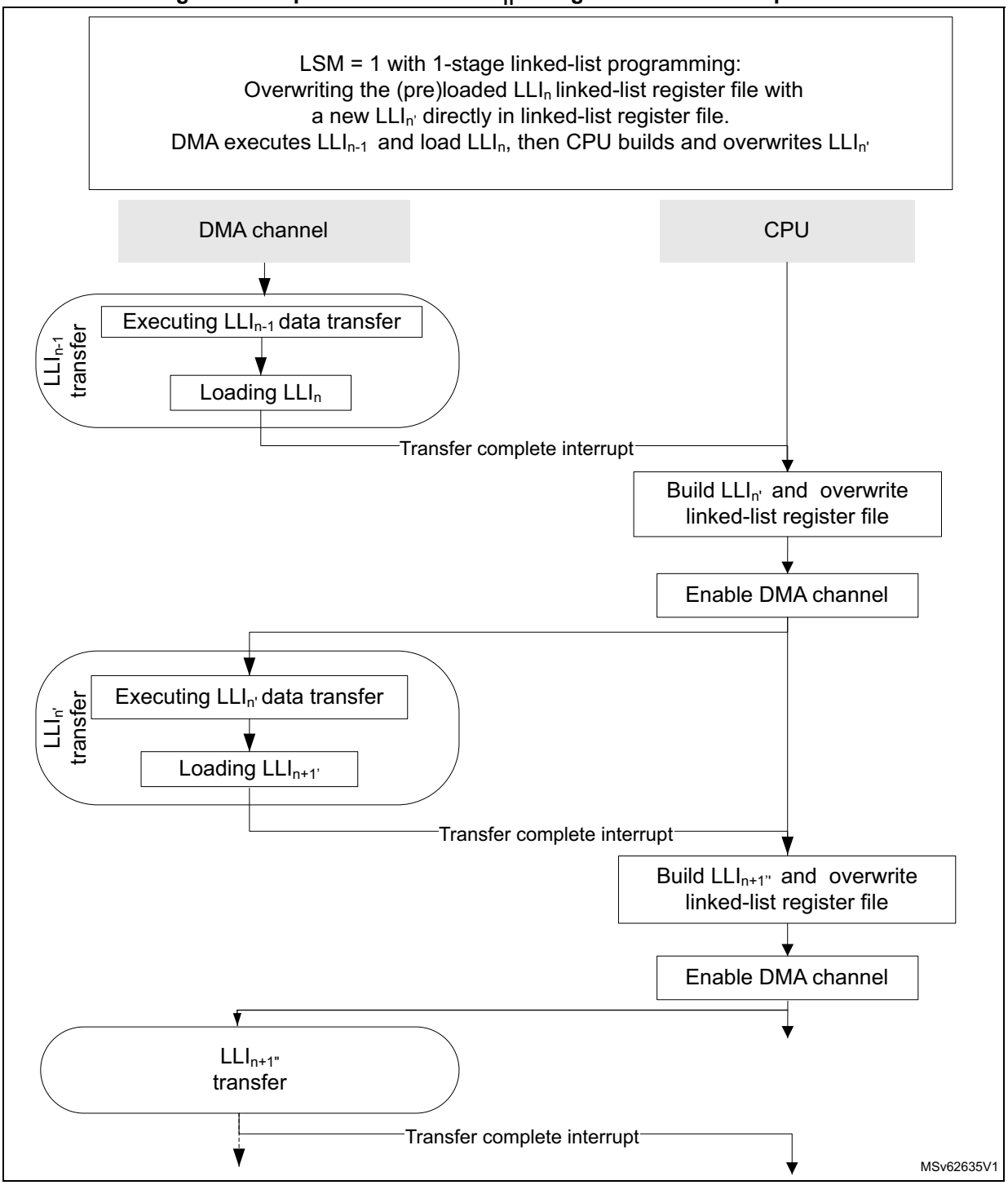
Figure 57. Building LLI_{n+1}: GPDMA dynamic linked-lists in link step mode



Run-time replacing a LLI_n with a new LLI_n', in link step mode (in linked-list register file)

In this link step mode, during run-time, the software can build and insert a new LLI_n', after GPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_n from the memory by overwriting directly the linked-list register file with the new LLI_n', as shown in the figure below.

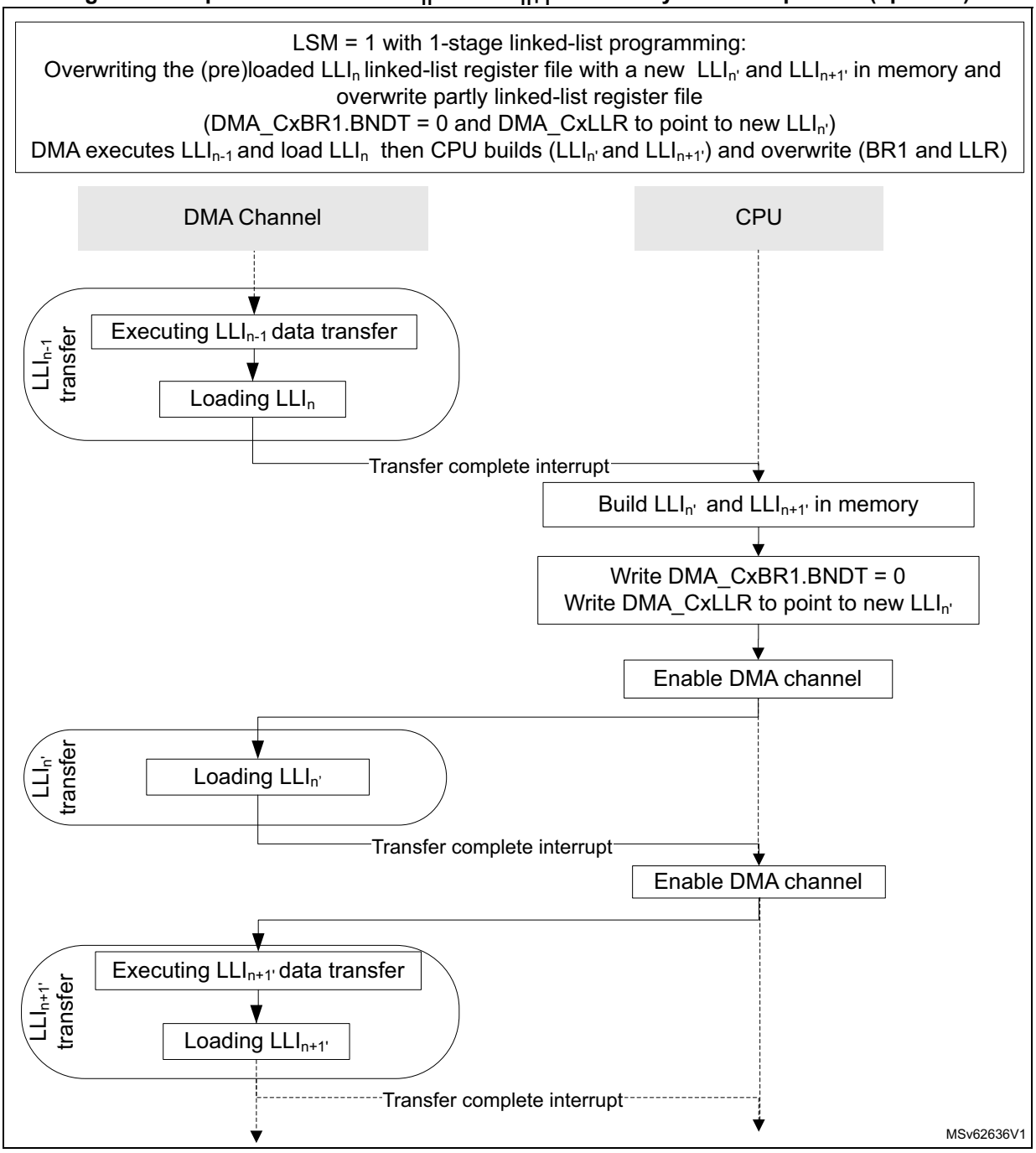
Figure 58. Replace with a new $LLI_{n'}$ in register file in link step mode



Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode (in the memory)

The software can build and insert a new $LLI_{n'}$ and $LLI_{n+1'}$ in the memory, after GPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_n from the memory, by overwriting partly the linked-list register file (GPDMA_CxBR1.BNDT[15:0] to be null and GPDMA_CxLLR to point to new $LLI_{n')}$ as shown in the figure below.

Figure 59. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 1)



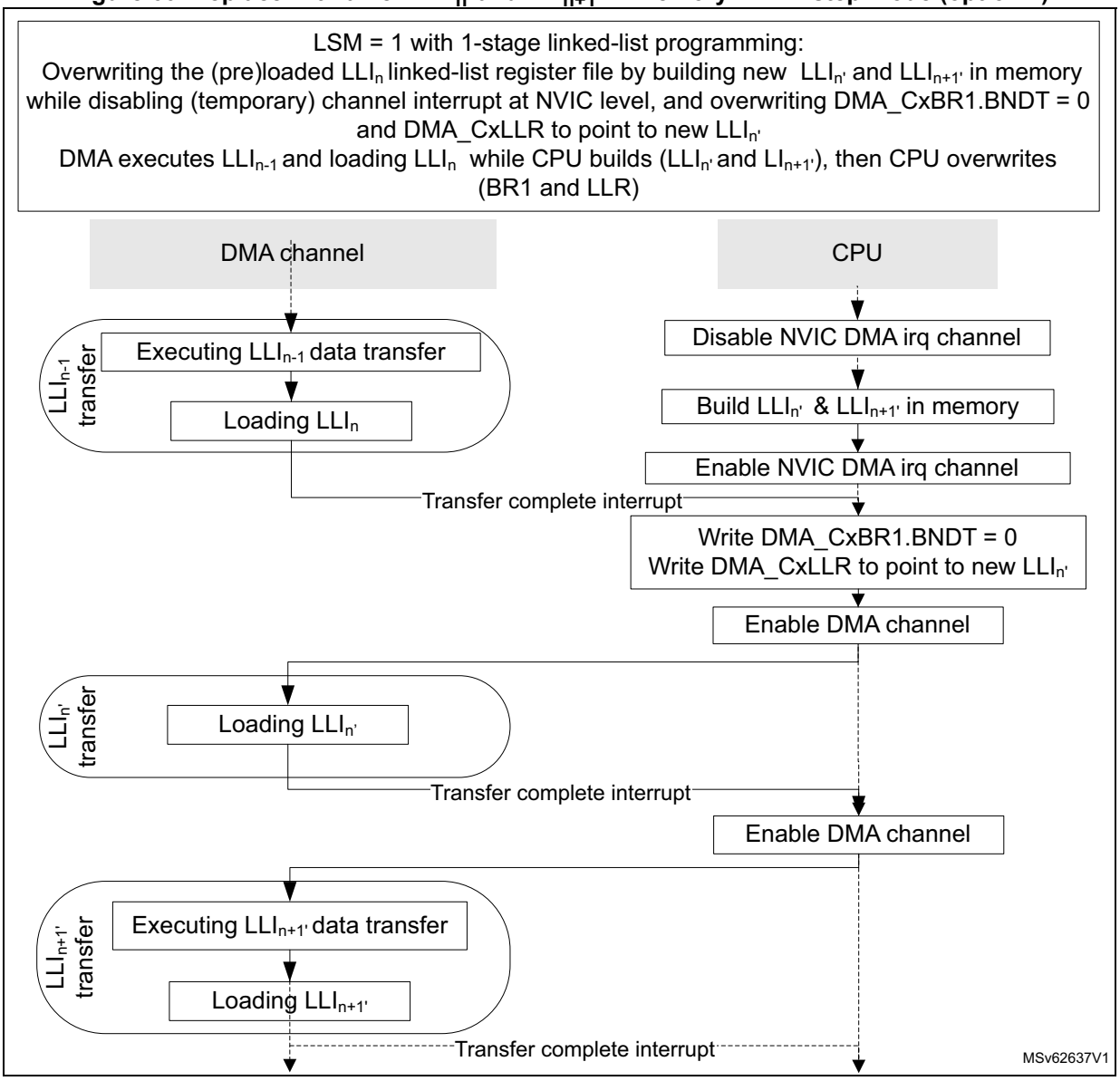
Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode

Other software implementations exist. Meanwhile GPDMA executes the transfer from the LLI_{n-1} and loads a formerly elaborated LLI_n from the memory (or even earlier), the software can do the following:

1. Disable the NVIC for not being interrupted by the interrupt handling.
2. Build a new $LLI_{n'}$ and a new $LLI_{n+1'}$.
3. Enable again the NVIC for the channel interrupt (transfer complete) notification.

The software in the interrupt handler for LLI_{n-1} is then restricted to overwrite GPDMA_CxBR1.BNDT[15:0] to be null and GPDMA_CxLLR to point to new $LLI_{n'}$, as shown in the figure below.

Figure 60. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 2)



17.4.9 GPDMA channel state and linked-list programming

The software can reconfigure a channel when the channel is disabled (GPDMA_CxCR.EN = 0) and update the execution mode (GPDMA_CxCR.LSM) to change from/to run-to-completion mode to/from link step mode.

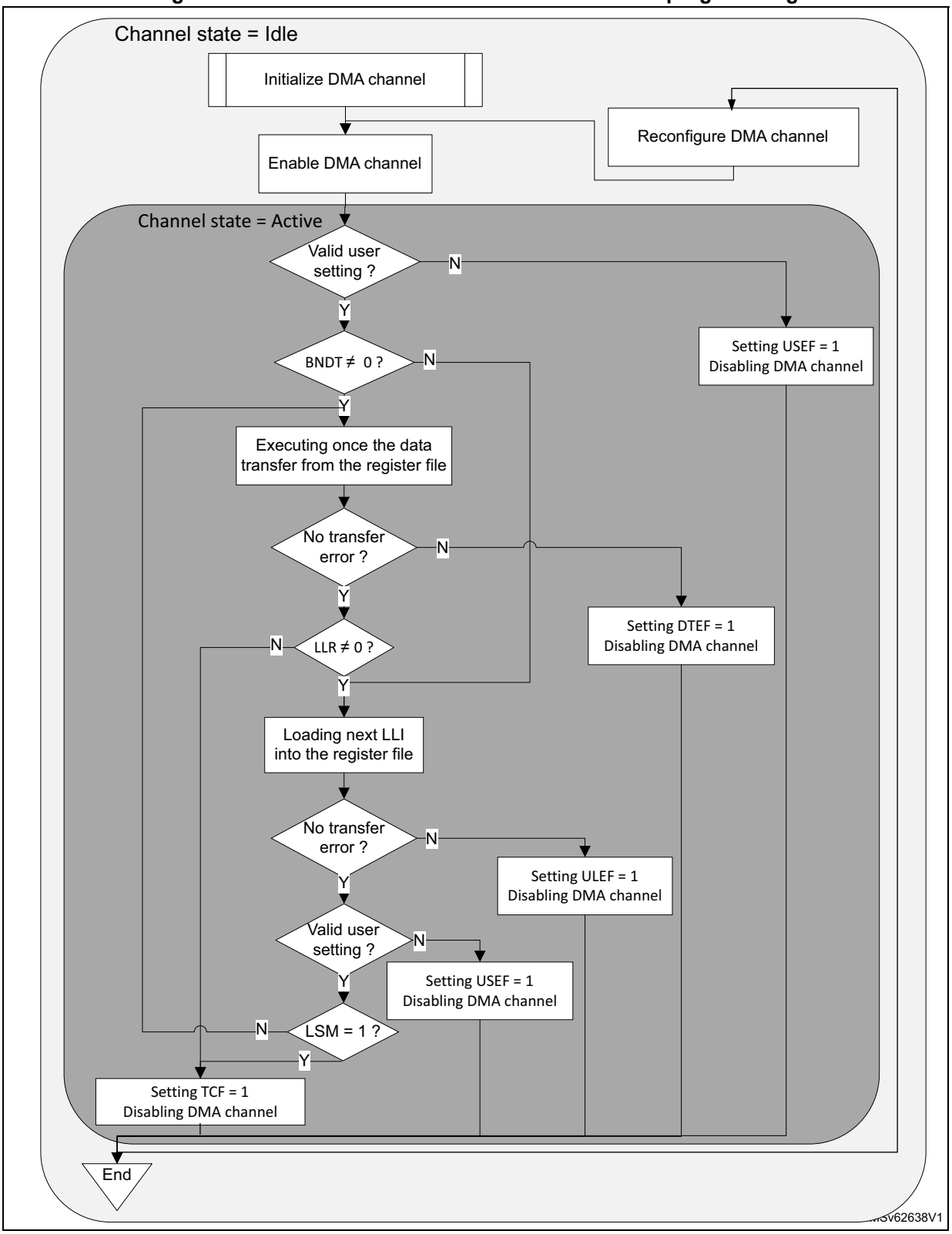
In any execution mode, the software can:

- reprogram LLI_{n+1} in the memory to finally complete the channel by this LLI_{n+1} (clear the GPDMA_CxLLR of this LLI_{n+1}), before that this LLI_{n+1} is loaded/used by the GPDMA channel
- abort and reconfigure the channel with a LSM update (see *Section 17.4.4*.)

In link step mode, the software can clear LSM after each a single execution of any LLI, during LLI_{n-1}.

The figure below shows the overall and unified GPDMA linked-list programming, whatever is the execution mode.

Figure 61. GPDMA channel execution and linked-list programming



17.4.10 GPDMA FIFO-based transfers

There is a single transfer operation mode: the FIFO mode. There are FIFO-based transfers. Any channel x is implemented with a dedicated FIFO whose size is defined by dma_fifo_size[x] (see *Section 17.3.1* for more details).

GPDMA burst

A programmed transfer at the lowest level is a GPDMA burst.

A GPDMA burst is a burst of data received from the source, or a burst of data sent to the destination. A source (and destination) burst is programmed with a burst length by the field SBL_1[5:0] (respectively DBL_1[5:0]), and with a data width defined by the field SDW_LOG2[1:0] (respectively DDW_LOG2[1:0]) in the GPDMA_CxTR1 register.

The addressing mode after each data (named beat) of a GPDMA burst is defined by SINC and DINC in GPDMA_CxTR1, for source and destination respectively: either a fixed addressing or an incremented addressing with contiguous data.

The start and next addresses of a GPDMA source/destination burst (defined by GPDMA_CxSAR and GPDMA_CxDAR) must be aligned with the respective data width.

The table below lists the main characteristics of a GPDMA burst.

Table 127. Programmed GPDMA source/destination burst

SDW_LOG2[1:0] DDW_LOG2[1:0]	Data width (bytes)	SINC/DINC	SBL_1[5:0] DBL_1[5:0]	Burst length (data/beats)	Next data/ beat address	Next burst address	Burst address alignment
00	1	0 (fixed)	n = 0 to 63(1)	n+1	+ 0	+ 0	1
01	2						2
10	4						4
00	1	1 (contiguously incremented)			+ 1	+ (n + 1)	1
01	2				+ 2	+ 2 * (n + 1)	2
10	4				+ 4	+ 4 * (n + 1)	4
11	forbidden user setting, causing USEF generation and none burst to be issued.						

1. When S/DBL_1[5:0] = 0, burst is of length 1. Then burst can be also named as single.

The next burst address in the above table is the next source/destination default address pointed by GPDMA_CxSAR or GPDMA_CxDAR, once the programmed source/destination burst is completed. This default value refers to the fixed/contiguously incremented address.

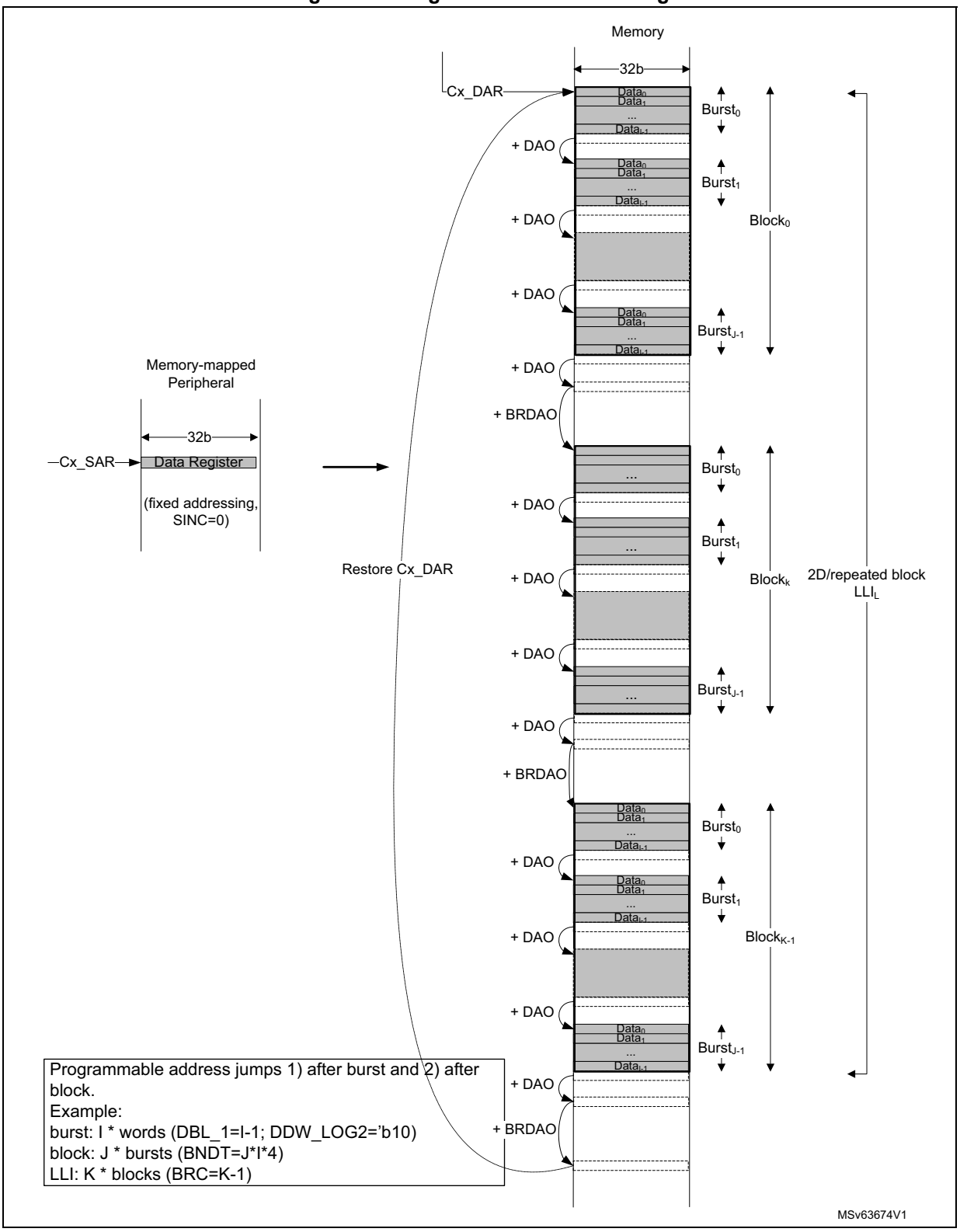
GPDMA burst with 2D addressing (channel x = 12 to 15)

When the channel has additional 2D addressing feature, this default value refers to the value without taking into account the two programmed incremented or decremented offsets. These two additional offsets (with a null default value) are applied:

- after each completed source/destination burst, as defined respectively by GPDMA_CxTR2.SAO[12:0]/DAO[12:0] and GPDMA_CxBR1.SDEC/DDEC
- after each completed block, as defined respectively by GPDMA_CxBR2.BRSAO[15:0]/BRDAO[15:0] and GPDMA_CxBR1.BRSDEC/BRDDEC)

Then, a 2D/repeated block can be addressed with a first programmed address jump after each completed burst, and with a second programmed address jump after each block, as depicted by the figure below with a 2D destination buffer.

Figure 62. Programmed 2D addressing



GPDMA FIFO-based burst

In FIFO-mode, a transfer generally consists of two pipelined and separated burst transfers:

- one burst from the source to the FIFO over the allocated source master port, as defined by GPDMA_CxTR1.SAP
- one burst from the FIFO to the destination over the allocated destination master port, as defined by GPDMA_CxTR1.DAP

GPDMA source burst

The requested source burst transfer to the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to get one new burst from the source):

when FIFO level ≤ $2^{dma\_fifo\_size[x]}$ - (SBL_1[5:0]+1) * $2^{SDW\_LOG2[1:0]}$

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- $2^{dma\_fifo\_size[x]}$ is the half of the FIFO size of the channel x, in bytes (see *Section 17.3.1* for the implementation details and dma_fifo_size[x] value).
- (SBL_1[5:0]+1) * $2^{SDW\_LOG2[1:0]}$ is the size of the programmed source burst transfer, in bytes.

Based on the channel priority (GPDMA_CxCR.PRIO[1:0]), this ready FIFO-based source transfer is internally arbitrated versus the other requested and active channels.

GPDMA destination burst

The requested destination burst transfer from the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to push one new burst to the destination):

when FIFO level ≥ (DBL_1[5:0]+1) * $2^{DDW\_LOG2[1:0]}$

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- (DBL_1[5:0]+1) * $2^{DDW\_LOG2[1:0]}$ is the size of the programmed destination burst transfer, in bytes.

Based on the channel priority, this ready FIFO-based destination transfer is internally arbitrated versus the other requested and active channels.

GPDMA burst vs source block size, 1-Kbyte address boundary and FIFO size

The programmed source/destination GPDMA burst is implemented with an AHB burst as is, unless one of the following conditions is met:

- When half of the FIFO size of the channel x is lower than the programmed source/destination burst size, the programmed source/destination GPDMA burst is implemented with a series of singles or bursts of a lower size, each transfer being of a size that is lower or equal than half of the FIFO size, without any user constraint.
- if the source block size (GPDMA_CxBR1.BNDT[15:0]) is not a multiple of the source burst size but is a multiple of the data width of the source burst (GPDMA_CxTR1.SDW_LOG2[1:0]), the GPDMA modifies and shortens bursts into singles or bursts of lower length, in order to transfer exactly the source block size, without any user constraint.

- if the source/destination burst transfer have crossed the 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol, without any user constraint.
- If the source/destination burst length exceeds 16 on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol, without any user constraint.

In any case, the GPDMA keeps ensuring source/destination data (and address) integrity without any user constraint. The current FIFO level (software readable in GPDMA_CxSR) is compared to and updated with the effective transfer size, and the GPDMA re-arbitrates between each AHB single or burst transfer, possibly modified.

Based on the channel priority, each single or burst of a lower burst size versus the programmed burst, is internally arbitrated versus the other requested and active channels.

Note: In linked-list mode, the GPDMA read transfers related to the update of the linked-list parameters from the memory to the internal GPDMA registers, are scheduled over the link allocated port, as programmed by GPDMA_CxCR.LAP.

GPDMA data handling: byte-based reordering, packing/unpacking, padding/truncation, sign extension and left/right alignment

The data handling is controlled by GPDMA_CxTR1. The source/destination data width of the programmed burst is byte, half-word or word, as per the SDW_LOG2[21:0] and DDW_LOG2[1:0] fields (see *Table 128*).

The user can configure the data handling between transferred data from the source and transfer to the destination. More specifically, programmed data handling is orderly performed with:

1. Byte-based source reordering
   - If SBX = 1 and if source data width is a word, the two bytes of the unaligned half-word at the middle of each source data word are exchanged.
2. Data width conversion by packing, unpacking, padding or truncation, if destination data width is different than the source data width, depending on PAM[1:0]:
   - If destination data width > source data width, the post SBX source data is either right-aligned and padded with 0 s, or sign extended up to the destination data width, or is FIFO queued and packed up to the destination data width.
   - If destination data width < source data width, the post SBX data is either right-aligned and left-truncated down to the destination data width, or is FIFO queued and unpacked and streamed down to the destination data width.
3. Byte-based destination re-ordering:
   - If DBX = 1 and if the destination data width is not a byte, the two bytes are exchanged within the aligned post PAM[1:0] half-words.
   - If DHX = 1 and if the destination data width is neither a byte nor a half-word, the two aligned half-words are exchanged within the aligned post PAM[1:0] words.

Note: Left-alignment with 0s-padding can be achieved by programming both a right-alignment with a 0s-padding and a destination byte-based re-ordering.

The table below lists the possible data handling from the source to the destination.

Table 128. Programmed data handling

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
00	Byte	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$	x	00	Byte	xx	x	x	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, 0P)	0		$0B_3,0B_2,0B_1,0B_0$
							1		B_30,B_20,B_10,B_00
						01 (RA, SE)	0		SB_3,SB_2,SB_1,SB_0
							1		B_3S,B_2S,B_1S,B_0S
						1x (PACK)	0		$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, 0P)	0	0	$000B_1,000B_0$
							1		$00B_10,00B_00$
							0	1	$0B_100,0B_000$
							1		B_1000,B_0000
						01 (RA, SE)	0	0	$SSSB_1,SSSB_0$
							1		SSB_1S,SSB_0S
							0	1	SB_1SS,SB_0SS
							1		B_1SSS,B_0SSS
						1x (PACK)	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$
01	Half-word	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$		00	Byte	00 (RA, LT)	x	x	B_6,B_4,B_2,B_0
						01 (LA, RT)			B_7,B_5,B_3,B_1
						1x (UNPACK)			$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$

Table 128. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
01	Half-word	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$	x	01	Half-word	xx	0	x	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, 0P)	0	0	$00B_3B_2,00B_1B_0$
							1		$00B_2B_3,00B_0B_1$
							0	1	B_3B_200,B_1B_000
							1		B_2B_300,B_0B_100
						01 (RA, SE)	0	0	SSB_3B_2,SSB_1B_0
							1		SSB_2B_3,SSB_0B_1
							0	1	B_3B_2SS,B_1B_0SS
							1		B_2B_3SS,B_0B_1SS
						1x (PACK)	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$
10	Word	$B_7B_6B_5B_4,B_3B_2B_1B_0$	0	00	Byte	00 (RA, LT)	x	x	B_{12},B_8,B_4,B_0
						01 (LA, RT)			B_{15},B_{11},B_7,B_3
						10 (UNPACK)			$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, LT)	0		B_5B_4,B_1B_0
							1		B_4B_5,B_0B_1
						01 (LA, RT)	0		B_7B_6,B_3B_2
							1		B_6B_7,B_2B_3
						1x (UNPACK)	0		$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$

Table 128. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
10	Word	$B_7B_6B_5B_4$, $B_3B_2B_1B_0$	0	10	Word	xx	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$
			1	00	Byte	00 (RA, LT)	x	x	B_{12},B_8,B_4,B_0
						01 (LA, RT)			B_{15},B_{11},B_7,B_3
						1x (UNPACK)			$B_7,B_5,B_6,B_4,B_3,B_1,B_2,B_0$
				01	Half-word	00 (RA, LT)	0		B_6B_4,B_2B_0
							1		B_4B_6,B_0B_2
						01 (LA, RT)	0		B_7B_5,B_3B_1
							1		B_5B_7,B_1B_3
						1x (UNPACK)	0		$B_7B_5,B_6B_4,B_3B_1,B_2B_0$
							1		$B_5B_7,B_4B_6,B_1B_3,B_0B_2$
				10	Word	xx	0	0	$B_7B_5B_6B_4,B_3B_1B_2B_0$
							1		$B_5B_7B_4B_6,B_1B_3B_0B_2$
							0	1	$B_6B_4B_7B_5,B_2B_0B_3B_1$
							1		$B_4B_6B_5B_7,B_0B_2B_1B_3$

1. Data stream is timely ordered starting from the byte with the lowest index (B_0).
2. RA= right aligned, LA = left aligned, RT = right truncated, LT = left truncated, 0P = zero bit padding up to the destination data width, SE = sign bit extended up to the destination data width.

17.4.11 GPDMA transfer request and arbitration

GPDMA transfer request

As defined by GPDMA_CxTR2, a programmed GPDMA data transfer is requested with one of the following:

- a software request if the control bit SWREQ = 1: This is used typically by the CPU for a data transfer from a memory-mapped address to another memory mapped address (memory-to-memory, GPIO to/from memory)
- an input hardware request coming from a peripheral if SWREQ = 0: The selection of the GPDMA hardware peripheral request is driven by the REQSEL[6:0] field (see *Section 17.3.3*). The selected hardware request can be one of the following:
  - an hardware request from a peripheral configured in GPDMA mode (for a transfer from/to the peripheral data register respectively to/from the memory)
  - an hardware request from a peripheral for its control registers update from the memory
  - an hardware request from a peripheral for a read of its status registers transferred to the memory

Caution: The user must not assign a same input hardware peripheral GPDMA request via GPDMA_CxTR.REQSEL[6:0] to two different channels, if at a given time this request is asserted by the peripheral and each channel is ready to execute this requested data transfer. There is no user setting error reporting.

GPDMA transfer request for arbitration

A ready FIFO-based GPDMA source single/burst transfer (from the source address to the FIFO) to be scheduled over the allocated master port (GPDMA_CxTR1.SAP) is arbitrated based on the channel priority (GPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested GPDMA transfers to the same master port.

A ready FIFO-based GPDMA destination single/burst transfer (from the FIFO to the destination address) to be scheduled over the allocated master port (GPDMA_CxTR1.DAP) is arbitrated based on the channel priority (GPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested GPDMA transfers to the same master port.

An arbitrated GPDMA requested link transfer consists of one 32-bit read from the linked-list data structure in memory to one of the linked-list registers (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR or GPDMA_CxLLR, plus GPDMA_CxTR3, GPDMA_CxBR2). Each 32-bit read from memory is arbitrated with the same channel priority as for data transfers, in order to be scheduled over the allocated master port (GPDMA_CxCR.LAP).

Whatever the requested data transfer is programmed with a software request for a memory-to- memory transfer (GPDMA_CxTR2.SWREQ = 1), or with a hardware request (GPDMA_CxTR2.SWREQ = 0) for a memory-to-peripheral transfer or a peripheral-to-memory transfer and whatever is the hardware request type, re-arbitration occurs after each granted single/burst transfer.

When an hardware request is programmed from a destination peripheral (GPDMA_CxTR2.SWREQ = 0 and GPDMA_CxTR2.DREQ = 1), the first memory read of a (possibly 2D/repeated) block (the first ready FIFO-based source burst request), is gated by the occurrence of the corresponding and selected hardware request. This first read request to memory is not taken into account earlier by the arbiter (not as soon as the block transfer is enabled and executable).

GPDMA arbitration

The GPDMA arbitration is directed from the 4-grade assigned channel priority (GPDMA_CxCR.PRIO[1:0]). The arbitration policy, as illustrated in *Figure 63*, is defined by:

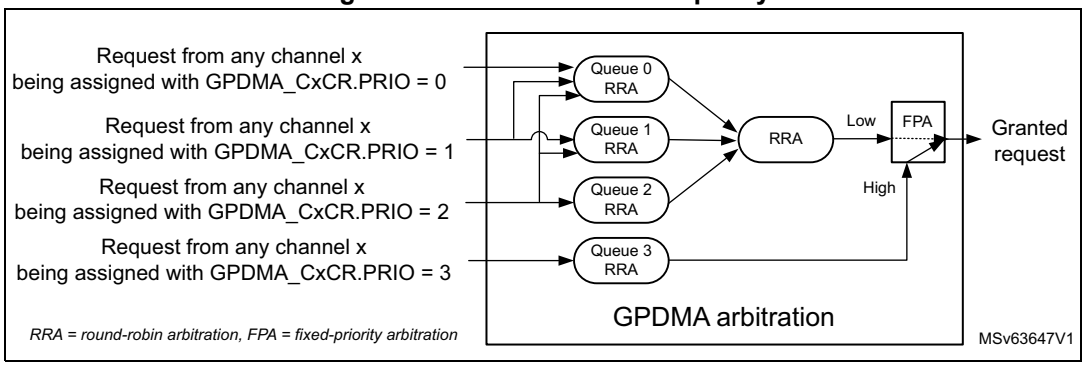
- one high-priority traffic class (queue 3), dedicated to the assigned channels with priority 3, for time-sensitive channels

  This traffic class is granted via a fixed-priority arbitration against any other low-priority traffic class. Within this class, requested single/burst transfers are round-robin arbitrated.
- three low-priority traffic classes (queues 0, 1 or 2) for non time-sensitive channels with priority 0, 1 or 2

  Each requested single/burst transfer within this class is round-robin arbitrated, with a weight that is monotonically driven from the programmed priority:
  - Requests with priority 0 are allocated to the queue 0.
  - Requests with priority 1 are allocated and replicated to the queue 0 and queue 1.

- Requests with priority 2 are allocated and replicated to the queue 0, queue 1, and queue 2.
- Any queue 0, 1 or 2 equally grants any of its active input requests in a round-robin manner, provided there are simultaneous requests.
- Additionally, there is a second stage for the low-traffic with a round-robin arbiter that fairly alternates between simultaneous selected requests from queue 0, queue 1 and queue 2.

Figure 63. GPDMA arbitration policy



GPDMA arbitration and bandwidth

With this arbitration policy, the following is guaranteed:

- Equal maximum bandwidth between requests with same priority
- Reserved bandwidth (noted as B_{Q3}) to the time-sensitive requests (with priority 3)
- Residual weighted bandwidth between different low-priority requests (priority 0 versus priority 1 versus priority 2).

The two following examples highlight that the weighted round-robin arbitration is driven by the programmed priorities:

- **Example 1**: basic application with two non time-sensitive GPDMA requests: req0 and req1. There are the following programming possibilities:
  - If they are assigned with same priority, the allocated bandwidth by the arbiter to req0 (B_{req0}) is **equal** to the allocated bandwidth to req1(B_{req1}).

    $B_{req0} = B_{req1} = 1/2 * (1 - B_{Q3})$
  - If req0 is assigned to priority 0 and req1 to priority 1, the allocated bandwidth to req0 (B_{P0}) is **3 times less** than the allocated bandwidth to req1 (B_{P1}).

    $B_{req0} = B_{P0} = 1/2 * 1/2 * (1 - B_{Q3}) = 1/4 * (1 - B_{Q3})$

    $B_{req1} = B_{P1} = (1/2 + 1) * 1/3 * (1 - B_{Q3}) = 3/4 * (1 - B_{Q3})$
  - If req0 is assigned to priority 0 and req1 to priority 2, the allocated bandwidth to req0 (B_{P0}) is **5 times less** than the allocated bandwidth to req1 (B_{P2}).

    $B_{req0} = B_{P0} = 1/2 * 1/3 * (1 - B_{Q3}) = 1/6 * (1 - B_{Q3})$

    $B_{req1} = B_{P2} = (1/2 + 1 + 1) * 1/3 * (1 - B_{Q3}) = 5/6 * (1 - B_{Q3})$

  The above computed bandwidth calculation is based on a theoretical input request, always active for any GPDMA clock cycle. This computed bandwidth from the arbiter must be weighted by the frequency of the request given by the application, that cannot be always active and may be quite much variable from one GPDMA client (example I2C at 400 kHz) to another one (PWM at 1 kHz) than the above x3 and x5 ratios.

- **Example 2**: application where the user distributes a same non-null N number of GPDMA requests to every non time-sensitive priority 0, 1 and 2. The bandwidth calculation is then the following:
  - The allocated bandwidth to the set of requests of priority 0 (B_{P0}) is
    $B_{P0} = 1/3 * 1/3 * (1 - B_{Q3}) = 1/9 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 1(B_{P1}) is
    $B_{P1} = (1/3 + 1/2) * 1/3 * (1 - B_{Q3}) = 5/18 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 2(B_{P2}) is
    $B_{P2} = (1/3 + 1/2 + 1) * 1/3 * (1 - B_{Q3}) = 11/18 * (1 - B_{Q3})$
  - The allocated bandwidth to any request n ($B_{n)}$ among the N requests of that priority Pi (i = 0 to 2) is $B_n = 1/N * B_{Pi}$
  - The allocated bandwidth to any request n of priority 0i ($B_{n, Pi}$) is
    $B_{n, P0} = 1/N *1/9 * (1 - B_{Q3})$
    $B_{n, P1} = 1/N *5/18 * (1 - B_{Q3})$
    $B_{n, P2} = 1/N *11/18 * (1 - B_{Q3})$

  In this example, when the master port bus bandwidth is not totally consumed by the time-sensitive queue 3, the residual bandwidth is such that 2.5 times less bandwidth is allocated to any request of priority 0 versus priority 1, and 5.5 times less bandwidth is allocated to any request of priority 0 versus priority 2.

More generally, assume that the following requests are present:

- I requests (I ≥ 0) assigned to priority 0
  If I > 0, these requests are noted from i = 0 to I-1.
- J requests (J ≥ 0) assigned to priority 1
  If J > 0, these requests are noted from j = 0 to J-1.
- K requests (K > 0) assigned to priority 2
  These requests are noted from k = 0 to K-1
- L requests (L ≥ 0) assigned to priority 3
  If L > 0, these requests are noted from l = 0 to L-1.

As B_{Q3} is the reserved bandwidth to time-sensitive requests, the bandwidth for each request L with priority 3 is:

- $B_l = B_{Q3} / L$ for L > 0 (else: $B_l = 0$)

The bandwidth for each non-time sensitive queue is:

- $B_{Q0} = 1/3 * (1 - B_{Q3})$
- $B_{Q1} = 1/3 * (1 - B_{Q3})$
- $B_{Q2} = 1/3 * (1 - B_{Q3})$

The bandwidth for the set of requests with priority 0 is:

- $B_{P0} = I / (I + J + K) * B_{Q0}$

The bandwidth for each request i with priority 0 is:

- $B_i = B_{P0} / I$ for L > 0 (else $B_{P0} = 0$)

The bandwidth for the set of requests with priority 1 and routed to queue 0 is:

- $B_{P1,Q0} = J / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 1 and routed to queue 1 is:

- $B_{P1,Q1} = J / (J + K) * B_{Q1}$

The total bandwidth for the set of requests with priority 1 is:

- $B_{P1} = B_{P1,Q0} + B_{P1,Q1}$

The bandwidth for each request j with priority 1 is:

- $B_j = B_{P1} / J$ for $J > 0$ (else $B_j = 0$)

The bandwidth for the set of requests with priority 2 and routed to queue 0 is:

- $B_{P2,Q0} = K / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 2 and routed to queue 1 is:

- $B_{P2,Q1} = K / (J + K) * B_{Q1}$

The bandwidth for the set of requests with priority 2 and routed to queue 2 is:

- $B_{P2,Q2} = B_{Q2}$

The total bandwidth for the set of requests with priority 2 is:

- $B_{P2} = B_{P2,Q0} + B_{P2,Q1} + B_{P2,Q2}$

The bandwidth for each request k with priority 2 is:

- $B_k = B_{P2} / K$ ($K>0$ in the general case)

Thus finally the maximum allocated residual bandwidths for any i, j, k non-time sensitive request are:

- in the general case (when there is at least one request k with a priority 2 ($K > 0$)):
  - $B_i = 1/I * 1/3 * I/(I + J + K) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/3 *[J/(I + J + K) + J/(J + K)] * (1 - B_{Q3})$
  - $B_k = 1/K * 1/3 *[K/(I + J + K) + K/(J + K) + 1] * (1 - B_{Q3})$
- in the specific case (when there is no request k with a priority 2 ($K = 0$)):
  - $B_i = 1/I * 1/2 * I/(I + J) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/2 *[J/(I + J) + 1] * (1 - B_{Q3})$

Consequently, the GPDMA arbiter can be used as a programmable weighted bandwidth limiter, for each queue and more generally for each request/channel. The different weights are monotonically resulting from the programmed channel priorities.

17.4.12 GPDMA triggered transfer

A programmed GPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event, as defined by GPDMA_CxTR2.TRIGPOL[1:0] and GPDMA_CxTR2.TRIGSEL[5:0] (see *Section 17.3.5* for the trigger selection).

The triggered transfer, as defined by the trigger mode in GPDMA_CxTR2.TRIGM[1:0], can be at LLI data transfer level, to condition the first burst read of a block, the first burst read of a 2D/repeated block for channel x (x = 12 to 15), or each programmed single read. The trigger mode can also be programmed to condition the LLI link transfer (see the TRIGM[1:0] description in *GPDMA channel x transfer register 2 (GPDMA_CxTR2)* for more details).

Trigger hit memorization and trigger overrun flag generation

The GPDMA monitoring of a trigger for a channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (respectively TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

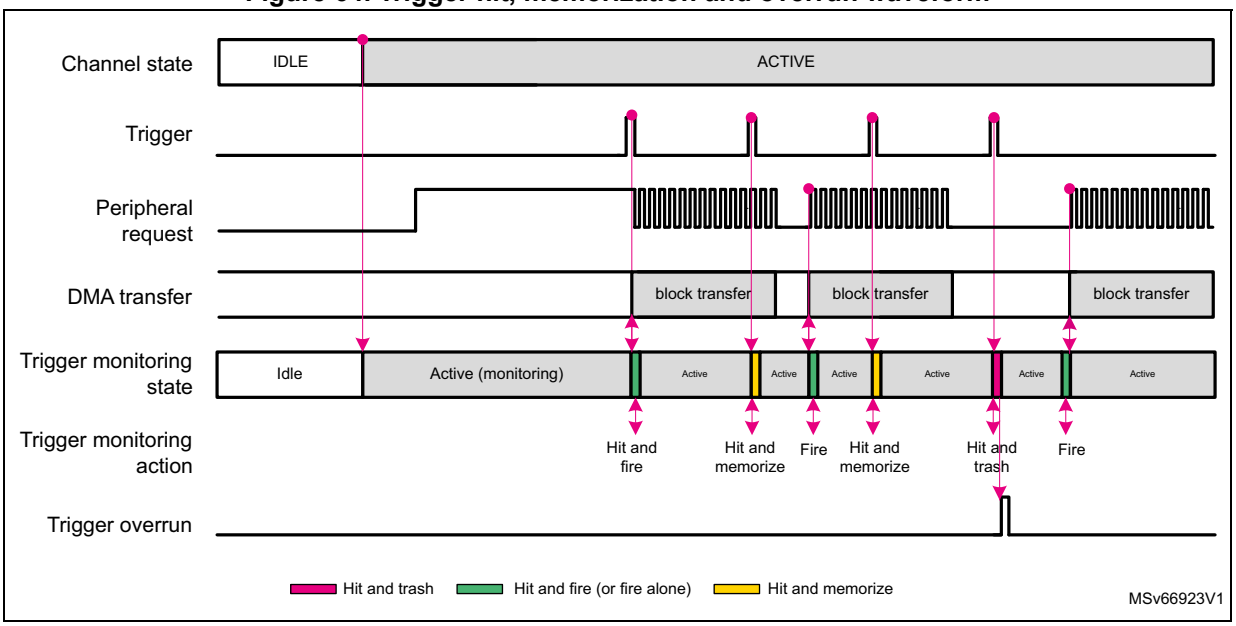
The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer. If a new trigger is detected, this hit is internally memorized to grant the next transfer, as long as the defined rising/falling edge and TRIGSEL[5:0] are not modified, and the channel is enabled.

Transferring a next $LLI_{n+1,}$ that updates the GPDMA_CxTR2 with a new value for any of TRIGSEL[5:0] or TRIGPOL[1:0], resets the monitoring, trashing the possible memorized hit of the formerly defined LLI_n trigger.

Caution: After a first new trigger hit_{n+1} is memorized, if another trigger hit_{n+2} is detected and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized. A trigger overrun flag is reported (GPDMA_CxSR.TOF = 1) and an interrupt is generated if enabled (if GPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

The figure below illustrates the trigger hit, memorization and overrun in the configuration example with a block-level trigger mode and a rising edge trigger polarity.

Figure 64. Trigger hit, memorization and overrun waveform



Note: The user can assign the same input trigger event to different channels. This can be used to trigger different channels on a broadcast trigger event.

17.4.13 GPDMA circular buffering with linked-list programming

GPDMA circular buffering for memory-to-peripheral and peripheral-to-memory transfers, with a linear addressing channel

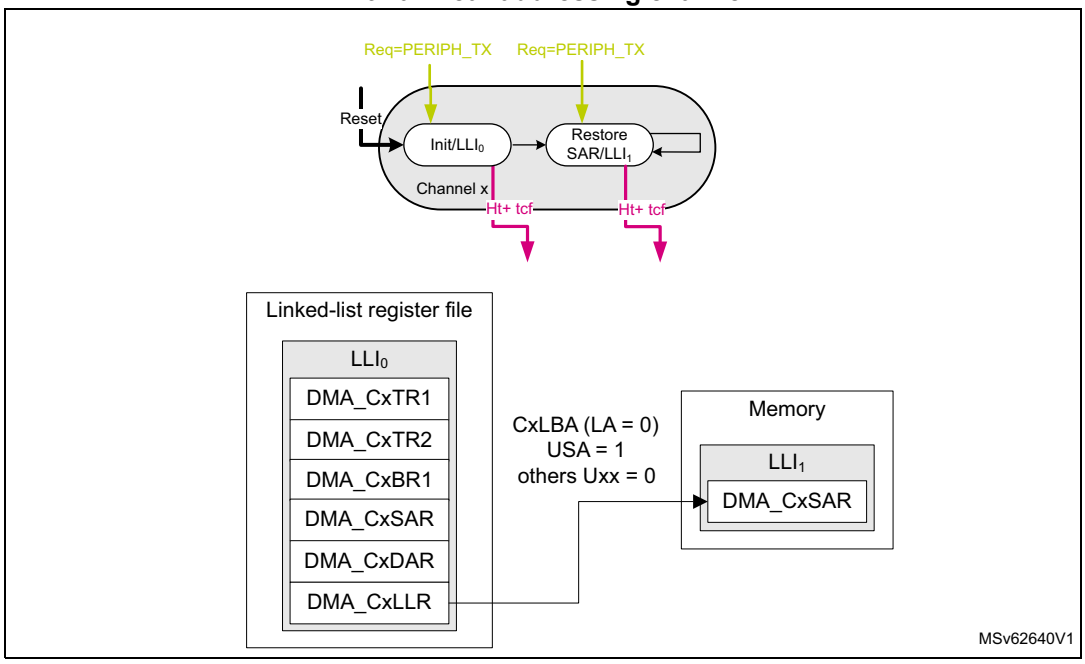
For a circular buffering, with a continuous memory-to-peripheral (or peripheral-to-memory) transfer, the software must set up a channel with half transfer and complete transfer

events/interrupts generation (GPDMA_CxCR.HTIE = 1 and GPDMA_CxCR.TCIE = 1), in order to enable a concurrent buffer software processing.

LLI_0 is configured for the first block transfer with the linear addressing channel. A continuously-executed LLI_1 is needed to restore the memory source (or destination) start address, for the memory-to-peripheral transfer (respectively the peripheral-to-memory transfer). GPDMA automatically reloads the initially programmed GPDMA_CxBR1.BNDT[15:0] when a block transfer is completed, and there is no need to restore GPDMA_CxBR1.

The figure below illustrates this programming with a linear addressing GPDMA channel and a source circular buffer.

Figure 65. GPDMA circular buffer programming: update of the memory start address with a linear addressing channel

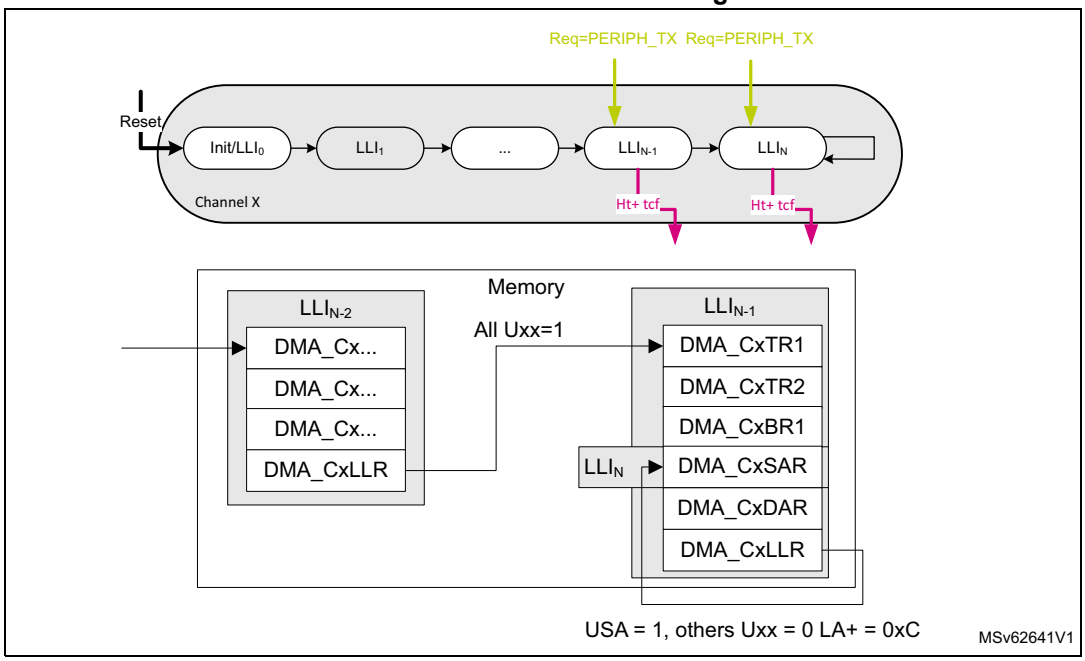


Note: With a 2D addressing channel, the user may use a single LLI with GPDMA_CxBR1.BRC[10:0] = 1, and program a negative memory block address offset with GDMA_CxBR2 and GDMA_CxBR1, in order to jump back to the memory source or the destination start address.

If circular buffering must be executed after some other transfers over the shared GPDMA channel x, the before-last LLI_{N-1} in memory is needed to configure the first block transfer. And the last LLI_N restores the memory source (or destination) start address in memory-to-peripheral transfer (respectively in peripheral-to-memory transfer).

The figure below illustrates this programming with a linear addressing shared GPDMA channel, and a source circular buffer.

Figure 66. Shared GPDMA channel with circular buffering: update of the memory start address with a linear addressing channel



17.4.14 GPDMA secure/non-secure channel

The GPDMA controller is compliant with the TrustZone hardware architecture at channel level, partitioning all its resources so that they exist in one of the secure and non-secure worlds at any given time.

Any channel x is a secure or a non-secure hardware resource, as securely configured by GPDMA_SECCFGR.SECx.

When a channel x is configured in secure state by a secure and privileged agent, the following access control rules are applied:

- A non-secure read access to a register field of this channel is forced to return 0, except for GPDMA_SECCFGR, GPDMA_PRIVCFGR and GPDMA_RCFGLOCKR that are readable by a non-secure agent.
- A non-secure write access to a register field of this channel has no impact.

When a channel x is configured in secure state, a secure agent can configure separately as secure or non-secure the GPDMA data transfer from the source (GPDMA_CxTR1.SSEC) and the GPDMA data transfer to the destination (GPDMA_CxTR1.DSEC).

When a channel x is configured in secure state and in linked-list mode, the loading of the next linked-list data structure from the GPDMA memory into its register file, is automatically performed with secure transfers via the GPDMA_CxCR.LAP allocated master port.

The GPDMA generates a secure bus that reflects GPDMA_SECCFGR, to keep the other peripherals informed of the secure/non-secure state of each GPDMA channel x.

The GPDMA also generates a security illegal access pulse signal on an illegal non-secure access to a secure GPDMA register. This signal is routed to the TrustZone interrupt controller.

When the secure software must switch a channel from a secure state to a non-secure state, the secure software must abort the channel or wait until the secure channel is completed before switching. This is needed to dynamically re-allocate a channel to a next non-secure transfer as a non-secure software is not allowed to do so and must have GPDMA_CxCR.EN = 0 before the non-secure software can reprogram the GPDMA_CxCR for a next transfer. The secure software may reset not only the channel x (GPDMA_CxCR.RESET = 1) but also the full channel x register file to its reset value.

17.4.15 GPDMA privileged/unprivileged channel

Any channel x is a privileged or unprivileged hardware resource, as configured by a privileged agent via GPDMA_PRIVCFGR.PRIVx.

When a channel x is configured in a privileged state by a privileged agent, the following access control rules are applied:

- An unprivileged read access to a register field of this channel is forced to return 0, except for GPDMA_PRIVCFGR, GPDMA_SECCFGR and GPDMA_RCFGLOCKR that are readable by an unprivileged agent.
- An unprivileged write access to a register field of this channel has no impact.

When a channel is configured in a privileged (or unprivileged) state, the source and destination data transfers are privileged (respectively unprivileged) transfers over the AHB master port.

When a channel is configured in a privileged (or unprivileged) state and in linked-list mode, the loading of the next linked-list data structure from the GPDMA memory into its register file, is automatically performed with privileged (respectively unprivileged) transfers, via the GPDMA_CxCR.LAP allocated master port.

The GPDMA generates a privileged bus that reflects GPDMA_PRIVCFGR, to keep the other peripherals informed of the privileged/unprivileged state of each GPDMA channel x.

When the privileged software must switch a channel from a privileged state to an unprivileged state, the privileged software must abort the channel or wait until that the privileged channel is completed before switching. This is needed to dynamically re-allocate a channel to a next unprivileged transfer as an unprivileged software is not allowed to do so, and must have GPDMA_CxCR.EN = 0 before the unprivileged software can reprogram the GPDMA_CxCR for a next transfer. The privileged software may reset not only the channel x (GPDMA_CxCR.RESET = 1) but also the full channel x register file to its reset value.

17.4.16 GPDMA error management

The GPDMA is able to manage and report to the user a transfer error, as follows, depending on the root cause.

Data transfer error

on a bus access (as a AHB single or a burst) to the source or the destination

- The source or destination target reports an AHB error.

- The programmed channel transfer is stopped (GPDMA_CxCR.EN cleared by the GPDMA hardware). The channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and the data error (GPDMA_CxSR.DTEF = 1).
- After a GPDMA data transfer error, the user must perform a debug session, taking care of the product-defined memory mapping of the source and destination, including the protection attributes.
- After a GPDMA data transfer error, the user must issue a channel reset (set GPDMA_CxCR.RESET) to reset the hardware GPDMA channel data path and the content of the FIFO, before the user enables again the same channel for a next transfer.

Link transfer error

on a tentative update of a GPDMA channel register from the programmed LLI in the memory

- The linked-list memory reports an AHB error.
- The programmed channel transfer is stopped (GPDMA_CxCR.EN cleared by the GPDMA hardware), the channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and the link error (GPDMA_CxSR.ULEF = 1).
- After a GPDMA link error, the user must perform a debug session, taking care of the product-defined memory mapping of the linked-list data structure (GPDMA_CxLBAR and GPDMA_CxLLR), including the protection attributes.
- After a GPDMA link error, the user must explicitly write the linked-list register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2), before the user enables again the same channel for a next transfer.

User setting error

on a tentative execution of a GPDMA transfer with an unauthorized user setting:

- The programmed channel transfer is disabled (GPDMA_CxCR.EN forced and cleared by the GPDMA hardware) preventing the next unauthorized programmed data transfer from being executed. The channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and a user setting error (GPDMA_CxSR.USEF = 1).
- After a GPDMA user setting error, the user must perform a debug session, taking care of the GPDMA channel programming. A user setting error can be caused by one of the following:
  - a programmed null source block size without a programmed update of this value from the next LLI_1 (GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxLLR.UB1 = 0)
  - a programmed non-null source block size being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxBR1.BNDT[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - when in packing/unpacking mode (if PAM[1]=1), a programmed non-null source block size being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxBR1.BNDT[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
  - a programmed unaligned source start address, being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxSAR[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])

- for channel x (x = 12 to 15): a programmed unaligned source address offset being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxTR3.SAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
- for channel x (x = 12 to 15): a programmed unaligned block repeated source address offset being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxBR2.BRSAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
- a programmed unaligned destination start address, being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxDAR[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
- for channel x (x = 12 to 15): a programmed unaligned destination address offset being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxTR3.DAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
- for channel x (x = 12 to 15): a programmed unaligned block repeated destination address offset being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxBR2.BRDAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
- a programmed double-word source data width (GPDMA_CxTR1.SDW_LOG2[1:0] = 11)
- a programmed double-word destination data width (GPDMA_CxTR1.DDW_LOG2[1:0] = 11)
- a programmed linked-list item LLI_{n+1} with a null data transfer (GPDMA_CxLLR.UB1 = 1 and GPDMA_CxBR1. BNDT = 0)

17.4.17 GPDMA autonomous mode

To save dynamic power consumption while the GPDMA executes the programmed linked-list transfers, the GPDMA hardware automatically manages its own clock gating and generates a clock request output signal to the RCC, whenever the device is in Run or low-power modes, provided that the RCC is programmed with the corresponding GPDMA enable control bits.

For more details about the RCC programming, refer to the RCC section of the reference manual.

For mode details about the availability of the GPDMA autonomous feature vs the device low-power modes, refer to *Section 17.3.2*.

The user can program and schedule the execution of a given GPDMA transfer at a LLI_n level of a GPDMA channel x, with GPDMA_CxTR2 as follows:

- The software controls and conditions the input of a transfer with TRIGM[1:0], TRIGPOL[1:0], TRIGSEL[5:0], SWREQ and REQSEL[6:0] for the input trigger and request.
- The software controls and signals the output of a transfer with TCEM[1:0] for generating or not a transfer complete event, and generating or not an associated half data transfer event).

See *GPDMA channel x transfer register 2 (GPDMA_CxTR2)* for more details.

When used in low-power modes, this functionality enables a CPU wakeup on a specific transfer completion by the enabled GPDMA transfer complete interrupt (GPDMA_CxCR.TCIE = 1) or/and enables to continue with the autonomous GPDMA for operating another LLI_{n+1} transfer over the same channel.

The output channel x transfer complete event, gpdma_chx_tc, can be programmed as a selected input trigger for a channel if this event is looped-back and connected at the GPDMA level (see *Section 17.3.5*), allowing autonomous and fine GPDMA inter-channel transfer scheduling, without needing a cleared transfer complete flag (TCF).

A given GPDMA channel x asserts its clock request in one of the following conditions:

- if the next transfer to be executed is programmed as conditioned by a trigger (GPDMA_CxTR2.TRIGPOL[1:0] and GPDMA_CxTR2.TRIGM[1:0]), only when the trigger hit occurs.
- if the next transfer to be executed is not conditioned by a trigger:
  - if GPDMA_CxTR2.SWREQ = 0, only when the hardware request is asserted by the selected peripheral
  - if GPDMA_CxTR2.SWREQ = 1 (memory-to-memory, GPIO to/from memory), as soon as the GPDMA is enabled

The GPDMA channel x releases its clock request as soon as all the following conditions are met:

- The transfer to be executed is completed.
- The GPDMA channel x is not immediately ready and requested to execute the next transfer.
- If a channel x interrupt was raised, all the flags of the status register that can cause this interrupt, are cleared by a software agent.

When one channel asserts its clock request, the GPDMA asserts its clock request to the RCC. When none channel asserts its clock request, the GPDMA releases its clock request to the RCC.

17.5 GPDMA in debug mode

When the microcontroller enters debug mode (core halted), any channel x can be individually either continued (default) or suspended, depending on the programmable control bit in the DBGMCU module.

Note: In debug mode, GPDMA_CxSR.SUSPF is not altered by a suspension from the programmable control bit in the DBGMCU module. In this case, GPDMA_CxSR.IDLEF can be checked to know the completion status of the channel suspension.

17.6 GPDMA in low-power modes

Table 129. Effect of low-power modes on GPDMA

Mode	Description
Sleep	No effect. GPDMA interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the GPDMA registers is kept when entering Stop mode. The content of the GPDMA registers can be autonomously updated by a next linked-list item from memory, to perform autonomous data transfers. GPDMA interrupts can cause the device to exit Stop mode(1).
Standby	The GPDMA is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 17.3.2* to know if any Stop mode is supported.

17.7 GPDMA interrupts

There is one GPDMA interrupt line for each channel, and separately for each CPU (if several ones in the devices).

Table 130. GPDMA interrupt requests

Interrupt acronym	Interrupt event	Interrupt enable	Event flag	Event clear method
GPDMA_CHx	Transfer complete	GPDMA_CxCR.TCIE	GPDMA_CxSR.TCF	Write 1 to GPDMA_CxFCR.TCF
	Half transfer	GPDMA_CxCR.HTIE	GPDMA_CxSR.HTF	Write 1 to GPDMA_CxFCR.HTF
	Data transfer error	GPDMA_CxCR.DTEIE	GPDMA_CxSR.DTEF	Write 1 to GPDMA_CxFCR.DTEF
	Update link error	GPDMA_CxCR.ULEIE	GPDMA_CxSR.ULEF	Write 1 to GPDMA_CxFCR.ULEF
	User setting error	GPDMA_CxCR.USEIE	GPDMA_CxSR.USEF	Write 1 to GPDMA_CxFCR.USEF
	Suspended	GPDMA_CxCR.SUSPIE	GPDMA_CxSR.SUSPF	Write 1 to GPDMA_CxFCR.SUSPF
	Trigger overrun	GPDMA_CxCR.TOFIE	GPDMA_CxSR.TOF	Write 1 to GPDMA_CxFCR.TOF

A GPDMA channel x event may be:

- a transfer complete
- a half-transfer complete
- a transfer error, due to either:
  - a data transfer error
  - an update link error
  - a user setting error completed suspension
- a trigger overrun

Note: When a channel x transfer complete event occurs, the output signal gpdma_chx_tc is generated as a high pulse of one clock cycle.

An interrupt is generated following any xx event, provided that both:

- the corresponding interrupt event xx is enabled (GPDMA_CxCR.xxIE = 1)
- the corresponding event flag is cleared (GPDMA_CxSR.xxF = 0). This means that, after a previous same xx event occurrence, a software agent must have written 1 into the corresponding xx flag clear control bit (write 1 into GPDMA_CxFCR.xxF).

TCF (transfer complete) and HTF (half transfer) events generation is controlled by GPDMA_CxTR2.TCEM[1:0] as follows:

- A transfer complete event is a block transfer complete, a 2D/repeated block transfer complete, or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode GPDMA_CxTR2.TCEM[1:0].

- A half transfer event is an half block transfer or a half 2D/repeated block transfer, depending on the transfer complete event mode GPDMA_CxTR2.TCEM[1:0].

  A half-block transfer occurs when half of the source block size bytes (rounded-up integer of GPDMA_CxBR1.BNDT[15:0] / 2) is transferred to the destination.

  A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded-up integer of (GPDMA_CxBR1.BRC[10:0] + 1) / 2) is transferred to the destination.

See *GPDMA channel x transfer register 2 (GPDMA_CxTR2)* for more details.

A transfer error rises in one of the following situations:

- during a single/burst data transfer from the source or to the destination (DTEF)
- during an update of a GPDMA channel register from the programmed LLI in memory (ULEF)
- during a tentative execution of a GPDMA channel with an unauthorized setting (USEF)

  The user must perform a debug session to correct the GPDMA channel programming versus the USEF root causes list (see *Section 17.4.16*).

A trigger overrun is described in *Trigger hit memorization and trigger overrun flag generation*.

17.8 GPDMA registers

The GPDMA registers must be accessed with an aligned 32-bit word data access.

17.8.1 GPDMA secure configuration register (GPDMA_SECCFGR)

Address offset: 0x00

Reset value: 0x0000 0000

A write access to this register must be secure. A read access is secure or non-secure, privileged or unprivileged.

A write access is ignored at bit level if the corresponding channel x is locked (GPDMA_RCFGLOCKR.LOCKx = 1).

This register can mix privileged and unprivileged information. If a channel x is configured as privileged (GPDMA_PRIVCFGR.PRIVx = 1), the SECx bit can be written only by a privileged (and secure) agent.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be programmed at a bit level, at the initialization/closure of a GPDMA channel (when GPDMA_CxCR.EN = 0), to securely allocate individually any channel x to the secure or non-secure world.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SECx**: secure state of channel x (x = 15 to 0)
0: non-secure
1: secure

17.8.2 GPDMA privileged configuration register (GPDMA_PRIVCFGR)

Address offset: 0x04

Reset value: 0x0000 0000

A write access to this register must be privileged. A read access can be privileged or unprivileged, secure or non-secure.

This register can mix secure and non-secure information. If a channel x is configured as secure (GPDMA_SECCFGR.SECx = 1), the PRIVx bit can be written only by a secure (and privileged) agent.

A write access is ignored at bit level if the corresponding channel x is locked (GPDMA_RCFGLOCKR.LOCKx = 1).

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be programmed at a bit level, at the initialization/closure of a GPDMA channel (GPDMA_CxCR.EN = 0), to individually allocate any channel x to the privileged or unprivileged world.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PRIVx**: privileged state of channel x (x = 15 to 0)
0: unprivileged
1: privileged

17.8.3 GPDMA configuration lock register (GPDMA_RCFGLOCKR)

Address offset: 0x08

Reset value: 0x0000 0000

This register can be written by a software agent with secure privileged attributes in order to individually lock at boot time the secure privileged attributes of any GPDMA

channel/resource (to lock the setting of GPDMA_CxSECCFGR and GPDMA_CxPRIVCFGR for any channel x at boot time).

A read access may be privileged or unprivileged, secure or non-secure.

Note: If TZEN = 0, this register cannot be written.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCK15	LOCK14	LOCK13	LOCK12	LOCK11	LOCK10	LOCK9	LOCK8	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **LOCKx**: lock the configuration of GPDMA_SECCFGR.SECx and GPDMA_PRIVCFGR.PRIVx, until a global GPDMA reset (x = 15 to 0)

This bit is cleared after reset and, once set, it cannot be reset until a global GPDMA reset.

0: secure privilege configuration of the channel x is writable.

1: secure privilege configuration of the channel x is not writable.

17.8.4 GPDMA non-secure masked interrupt status register (GPDMA_MISR)

Address offset: 0x0C

Reset value: 0x0000 0000

This register is a read register.

This is a non-secure register, containing the masked interrupt status bit MISx for each non-secure channel x (channel x configured with GPDMA_SECCFGR.SECx = 0). It is a logical OR of all the flags of GPDMA_CxSR, each source flag being enabled by the corresponding interrupt enable bit of GPDMA_CxCR.

Every bit is deasserted by hardware when writing 1 to the corresponding flag clear bit in GPDMA_CxFCR.

If a channel x is in secure state (GPDMA_SECCFGR.SECx = 1), a read access to the masked interrupt status bit MISx of this channel x returns zero.

This register may mix privileged and unprivileged information, depending on the privileged state of each channel GPDMA_PRIVCFGR.PRIVx. A privileged software can read the full non-secure interrupt status. An unprivileged software is restricted to read the status of unprivileged (and non-secure) channels, other privileged bit fields returning zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIS15	MIS14	MIS13	MIS12	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
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 **MISx**: masked interrupt status of channel x (x = 15 to 0)
0: no interrupt occurred on channel x
1: an interrupt occurred on channel x

17.8.5 GPDMA secure masked interrupt status register (GPDMA_SMISR)

Address offset: 0x10

Reset value: 0x0000 0000

This is a secure read register, containing the masked interrupt status bit MISx for each secure channel x (GPDMA_SECCFGR.SECx = 1). It is a logical OR of all the GPDMA_CxSR flags, each source flag being enabled by the corresponding GPDMA_CxCR interrupt enable bit.

Every bit is deasserted by hardware when securely writing 1 to the corresponding GPDMA_CxFCR flag clear bit.

This register does not contain any information about a non-secure channel.

This register can mix privileged and unprivileged information, depending on the privileged state of each channel GPDMA_PRIVCFGR.PRIVx. A privileged software can read the full secure interrupt status. An unprivileged software is restricted to read the status of unprivileged and secure channels, other privileged bit fields returning zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIS15	MIS14	MIS13	MIS12	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
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 **MISx**: masked interrupt status of the secure channel x (x = 15 to 0)
0: no interrupt occurred on the secure channel x
1: an interrupt occurred on the secure channel x

17.8.6 GPDMA channel x linked-list base address register (GPDMA_CxLBAR)

Address offset: 0x50 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register must be written by a privileged software. It is either privileged readable or not, depending on the privileged state of the channel x GPDMA_PRIVCFGR.PRIVx.

This register is either secure or non-secure depending on the secure state of the channel x (GPDMA_SECCFGR.SECx).

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This channel-based register is the linked-list base address of the memory region, for a given channel x, from which the LLIs describing the programmed sequence of the GPDMA transfers, are conditionally and automatically updated.

This 64-Kbyte aligned channel x linked-list base address is offset by the 16-bit GPDMA_CxLLR register that defines the word-aligned address offset for each LLI.

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

Bits 31:16 **LBA[31:16]**: linked-list base address of GPDMA channel x

Bits 15:0 Reserved, must be kept at reset value.

17.8.7 GPDMA channel x flag clear register (GPDMA_CxFCR)

Address offset: 0x5C+ 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This is a write register, secure or non-secure depending on the secure state of channel x (GPDMA_SECCFGR.SECx) and privileged or unprivileged, depending on the privileged state of the channel x (GPDMA_PRIVCFGR.PRIVx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	w	w	w	w	w	w	w								

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

Bit 14 **TOF**: trigger overrun flag clear
0: no effect
1: corresponding TOF flag cleared

Bit 13 **SUSPF**: completed suspension flag clear
0: no effect
1: corresponding SUSPF flag cleared

Bit 12 **USEF**: user setting error flag clear
0: no effect
1: corresponding USEF flag cleared

Bit 11 **ULEF**: update link transfer error flag clear
0: no effect
1: corresponding ULEF flag cleared

Bit 10 **DTEF**: data transfer error flag clear
0: no effect
1: corresponding DTEF flag cleared

Bit 9 **HTF**: half transfer flag clear
0: no effect
1: corresponding HTF flag cleared

Bit 8 **TCF**: transfer complete flag clear
0: no effect
1: corresponding TCF flag cleared

Bits 7:0 Reserved, must be kept at reset value.

17.8.8 GPDMA channel x status register (GPDMA_CxSR)

Address offset: 0x60 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0001

This is a read register, reporting the channel status.

This register is secure or non-secure, depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or non-privileged, depending on the privileged state of the channel (GPDMA_PRIVCFGR.PRIVx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOL[7:0]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDLEF
	r	r	r	r	r	r	r								r

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

Bits 23:16 **FIFOL[7:0]**: monitored FIFO level

Number of available write beats in the FIFO, in units of the programmed destination data width (see GPDMA_CxTR1.DDW_LOG2[1:0], in units of bytes, half-words, or words).

Note: After having suspended an active transfer, the user may need to read FIFOL[7:0], additionally to GPDMA_CxBR1.BDNT[15:0] and GPDMA_CxBR1.BRC[10:0], to know how many data have been transferred to the destination. Before reading, the user may wait for the transfer to be suspended (GPDMA_CxSR.SUSPF = 1).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOF**: trigger overrun flag

0: no trigger overrun event

1: a trigger overrun event occurred

Bit 13 **SUSPF**: completed suspension flag

0: no completed suspension event

1: a completed suspension event occurred

Bit 12 **USEF**: user setting error flag

0: no user setting error event

1: a user setting error event occurred

Bit 11 **ULEF**: update link transfer error flag

0: no update link transfer error event

1: a master bus error event occurred while updating a linked-list register from memory

Bit 10 **DTEF**: data transfer error flag

0: no data transfer error event

1: a master bus error event occurred on a data transfer

Bit 9 **HTF**: half transfer flag

0: no half transfer event

1: a half transfer event occurred

A half transfer event is either a half block transfer or a half 2D/repeated block transfer, depending on the transfer complete event mode (GPDMA_CxTR2.TCEM[1:0]).

A half block transfer occurs when half of the bytes of the source block size (rounded up integer of GPDMA_CxBR1.BNDT[15:0]/2) has been transferred to the destination.

A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded up integer of (GPDMA_CxBR1.BRC[10:0]+1)/2)) has been transferred to the destination.

Bit 8 **TCF**: transfer complete flag

0: no transfer complete event

1: a transfer complete event occurred

A transfer complete event is either a block transfer complete, a 2D/repeated block transfer complete, or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode (GPDMA_CxTR2.TCEM[1:0]).

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

Bit 0 **IDLEF**: idle flag

0: channel not in idle state

1: channel in idle state

This idle flag is deasserted by hardware when the channel is enabled (GPDMA_CxCR.EN = 1) with a valid channel configuration (no USEF to be immediately reported).

This idle flag is asserted after hard reset or by hardware when the channel is back in idle state (in suspended or disabled state).

17.8.9 GPDMA channel x control register (GPDMA_CxCR)

Address offset: 0x64 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is secure or non-secure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of the channel x (GPDMA_PRIVCFGR.PRIVx).

This register is used to control a channel (activate, suspend, abort or disable it).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO[1:0]		Res.	Res.	Res.	Res.	LAP	LSM
								rw	rw					rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	Res.	SUSP	RESET	EN
	rw	rw	rw	rw	rw	rw	rw						rw	w	rw

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

Bits 23:22 **PRIO[1:0]:** priority level of the channel x GPDMA transfer versus others
00: low priority, low weight
01: low priority, mid weight
10: low priority, high weight
11: high priority
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bits 21:18 Reserved, must be kept at reset value.

Bit 17 **LAP**: linked-list allocated port
This bit is used to allocate the master port for the update of the GPDMA linked-list registers from the memory.
0: port 0 (AHB) allocated
1: port 1 (AHB) allocated
Note: This bit must be written when EN=0. This bit is read-only when EN=1.

Bit 16 **LSM**: Link step mode
0: channel executed for the full linked-list and completed at the end of the last LLI (GPDMA_CxLLR = 0). The 16 low-significant bits of the link address are null (LA[15:0] = 0) and all the update bits are null (UT1 = UB1 = UT2 = USA = UDA = ULL = 0 and UT3 = UB2 = 0). Then GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0.
1: channel executed **once** for the current LLI
First the (possible 1D/repeated) block transfer is executed as defined by the current internal register file until GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0. Secondly the next linked-list data structure is conditionally uploaded from memory as defined by GPDMA_CxLLR. Then channel execution is completed.
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOIE**: trigger overrun interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 13 **SUSPIE**: completed suspension interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 12 **USEIE**: user setting error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 11 **ULEIE**: update link transfer error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 10 **DTEIE**: data transfer error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 9 **HTIE**: half transfer complete interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 8 **TCIE**: transfer complete interrupt enable
0: interrupt disabled
1: interrupt enabled

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

Bit 2 **SUSP:** suspend
Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 2. Else:
Software must write 1 in order to suspend an active channel (channel with an ongoing GPDMA transfer over its master ports).
The software must write 0 in order to resume a suspended channel, following the programming sequence detailed in *Figure 48*.
0: write: resume channel, read: channel not suspended
1: write: suspend channel, read: channel suspended.

Bit 1 **RESET:** reset
This bit is write only. Writing 0 has no impact. Writing 1 implies the reset of the following: the FIFO, the channel internal state, SUSP and EN bits (whatever is written receptively in bit 2 and bit 0).
The reset is effective when the channel is in steady state, meaning one of the following:
- active channel in suspended state (GPDMA_CxSR.SUSPF = 1 and GPDMA_CxSR.IDLEF = GPDMA_CxCR.EN = 1)
- channel in disabled state (GPDMA_CxSR.IDLEF = 1 and GPDMA_CxCR.EN = 0).
After writing a RESET, to continue using this channel, the user must explicitly reconfigure the channel including the hardware-modified configuration registers (GPDMA_CxBR1, GPDMA_CxSAR and GPDMA_CxDAR) before enabling again the channel (see the programming sequence in *Figure 49*).
0: no channel reset
1: channel reset

Bit 0 **EN:** enable

Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 0. Else:

this bit is deasserted by hardware when there is a transfer error (master bus error or user setting error) or when there is a channel transfer complete (channel ready to be configured, for example if LSM = 1 at the end of a single execution of the LLI).

Else, this bit can be asserted by software.

Writing 0 into this EN bit is ignored.

0: write: ignored, read: channel disabled

1: write: enable channel, read: channel enabled

17.8.10 GPDMA channel x transfer register 1 (GPDMA_CxTR1)

Address offset: 0x90 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is secure or non-secure depending on the secure state of channel x (GPDMA_SECCFGR.SECx) except for secure DSEC and SSEC, privileged or non-privileged, depending on the privileged state of the channel x in GPDMA_PRIVCFGR.PRIVx.

This register controls the transfer of a channel x.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed. Then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory if GPDMA_CxLLR.UT1 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DSEC	DAP	Res.	Res.	DHX	DBX	DBL_1[5:0]						DINC	Res.	DDW_LOG2[1:0]	
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
SSEC	SAP	SBX	PAM[1:0]		Res.	SBL_1[5:0]						SINC	Res.	SDW_LOG2[1:0]	
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit 31 **DSEC**: security attribute of the GPDMA transfer to the destination

If GPDMA_SECCFGR.SECx = 1 and the access is secure:

0: GPDMA transfer non-secure

1: GPDMA transfer secure

This is a secure register bit. This bit can only be read by a secure software. This bit must be written by a secure software when GPDMA_SECCFGR.SECx = 1. A secure write is ignored when GPDMA_SECCFGR.SECx = 0.

When GPDMA_SECCFGR.SECx is deasserted, this DSEC bit is also deasserted by hardware (on a secure reconfiguration of the channel as non-secure), and the GPDMA transfer to the destination is non-secure.

Bit 30 **DAP**: destination allocated port

This bit is used to allocate the master port for the destination transfer

0: port 0 (AHB) allocated

1: port 1 (AHB) allocated

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

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

Bit 27 **DHX**: destination half-word exchange

If the destination data size is shorter than a word, this bit is ignored.

If the destination data size is a word:

0: no halfword-based exchanged within word

1: the two consecutive (post PAM) half-words are exchanged in each destination word.

Bit 26 **DBX**: destination byte exchange

If the destination data size is a byte, this bit is ignored.

If the destination data size is not a byte:

0:no byte-based exchange within half-word

1: the two consecutive (post PAM) bytes are exchanged in each destination half-word.

Bits 25:20 **DBL_1[5:0]**: destination burst length minus 1, between 0 and 63

The burst length unit is one data named beat within a burst. If DBL_1[5:0] =0 , the burst can be named as single. Each data/beat has a width defined by the destination data width DDW_LOG2[1:0].

Note: If a burst transfer crossed a 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

If a burst transfer is of length greater than the FIFO size of the channel x, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with GPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 19 **DINC**: destination incrementing burst

0: fixed burst

1: contiguously incremented burst

The destination address, pointed by GPDMA_CxDAR, is kept constant after a burst beat/single transfer, or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **DDW_LOG2[1:0]**: binary logarithm of the destination data width of a burst, in bytes

00: byte

01: half-word (2 bytes)

10: word (4 bytes)

11: user setting error reported and no transfer issued

Note: Setting a 8-byte data width causes a user setting error to be reported and none transfer is issued.

A destination burst transfer must have an aligned address with its data width (start address GPDMA_CxDAR[2:0] and address offset GPDMA_CxTR3.DAO[2:0], versus DDW_LOG2[1:0]). Otherwise a user setting error is reported and no transfer is issued.

Bit 15 **SSEC**: security attribute of the GPDMA transfer from the source

If GPDMA_SECCFGR.SECx = 1 and the access is secure:

0: GPDMA transfer non-secure

1: GPDMA transfer secure

This is a secure register bit. This bit can only be read by a secure software. This bit must be written by a secure software when GPDMA_SECCFGR.SECx =1 . A secure write is ignored when GPDMA_SECCFGR.SECx = 0.

When GPDMA_SECCFGR.SECx is deasserted, this SSEC bit is also deasserted by hardware (on a secure reconfiguration of the channel as non-secure), and the GPDMA transfer from the source is non-secure.

Bit 14 **SAP**: source allocated port

This bit is used to allocate the master port for the source transfer

0: port 0 (AHB) allocated

1: port 1 (AHB) allocated

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 13 **SBX**: source byte exchange within the unaligned half-word of each source word

If the source data width is shorter than a word, this bit is ignored.

If the source data width is a word:

0:no byte-based exchange within the unaligned half-word of each source word

1: the two consecutive bytes within the unaligned half-word of each source word are exchanged.

Bits 12:11 **PAM[1:0]**: padding/alignment mode

If DDW_LOG2[1:0] = SDW_LOG2[1:0]: if the data width of a burst destination transfer is equal to the data width of a burst source transfer, these bits are ignored.

Else, in the following enumerated values, the condition PAM_1 is when destination data width is higher that source data width, and the condition PAM_2 is when destination data width is higher than source data width.

Condition: PAM_1

00: source data is transferred as right aligned, padded with 0s up to the destination data width

01: source data is transferred as right aligned, sign extended up to the destination data width

10-11: successive source data are FIFO queued and packed at the destination data width, in a left (LSB) to right (MSB) order (named little endian), before a destination transfer

Condition: PAM_2

00: source data is transferred as right aligned, left-truncated down to the destination data width

01: source data is transferred as left-aligned, right-truncated down to the destination data width

Note: 10-11: source data is FIFO queued and unpacked at the destination data width, to be transferred in a left (LSB) to right (MSB) order (named little endian) to the destination

Bit 10 Reserved, must be kept at reset value.

Bits 9:4 **SBL_1[5:0]**: source burst length minus 1, between 0 and 63

The burst length unit is one data named beat within a burst. If SBL_1[5:0] =0 , the burst can be named as single. Each data/beat has a width defined by the destination data width SDW_LOG2[1:0].

Note: If a burst transfer crossed a 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

If a burst transfer is of length greater than the FIFO size of the channel x, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with GPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 3 **SINC**: source incrementing burst

0: fixed burst

1: contiguously incremented burst

The source address, pointed by GPDMA_CxSAR, is kept constant after a burst beat/single transfer or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **SDW_LOG2[1:0]**: binary logarithm of the source data width of a burst in bytes

00: byte

01: half-word (2 bytes)

10: word (4 bytes)

11: user setting error reported and no transfer issued

Note: Setting a 8-byte data width causes a user setting error to be reported and no transfer is issued.

A source block size must be a multiple of the source data width (GPDMA_CxBR1.BNDT[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and no transfer is issued.

A source burst transfer must have an aligned address with its data width (start address GPDMA_CxSAR[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and none transfer is issued.

17.8.11 GPDMA channel x transfer register 2 (GPDMA_CxTR2)

Address offset: 0x94 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is secure or non-secure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x.

This register must be written when LGPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN =1 .

This register must be written when the channel is completed (the hardware deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory, if GPDMA_CxLLR.UT2 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCEM[1:0]		Res.	Res.	Res.	Res.	TRIGPOL[1:0]		Res.	Res.	TRIGSEL[5:0]					
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
TRIGM[1:0]		Res.	Res.	BREQ	DREQ	SWREQ	Res.	Res.	REQSEL[6:0]						
rw	rw			rw	rw	rw			rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **TCEM[1:0]**: transfer complete event mode

These bits define the transfer granularity for the transfer complete and half transfer complete events generation.

00: at block level (when GPDMA_CxBR1.BNDT[15:0] = 0): the complete (and the half) transfer event is generated at the (respectively half of the) end of a block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half transfer event is generated.

01: channel x (0 to 11), same as 00, channel x (x =12 to 15), at 2D/repeated block level (when GPDMA_CxBR1.BRC[10:0] = 0 and GPDMA_CxBR1.BNDT[15:0] = 0). The complete (and the half) transfer event is generated at the end (respectively half of the end) of the 2D/repeated block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half transfer event is generated.

10: at LLI level: the complete transfer event is generated at the end of the LLI transfer, including the update of the LLI if any. The half transfer event is generated at the half of the LLI data transfer The LLI data transfer is a block transfer or a 2D/repeated block transfer for channel x (x =12 to 15), if any data transfer.

Note: If the initial LLI_0 data transfer is null/void (i.e. directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then the half transfer event is not generated, and the transfer complete event is generated when is completed the loading of the LLI_1.

11: at channel level: the complete transfer event is generated at the end of the last LLI transfer. The half transfer event is generated at the half of the data transfer of the last LLI. The last LLI updates the link address GPDMA_CxLLR.LA[15:2] to zero and clears all the GPDMA_CxLLR update bits (UT1, UT2, UB1, USA, UDA and ULL, plus UT3 and UB2). If the channel transfer is continuous/infinite, no event is generated.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **TRIGPOL[1:0]**: trigger event polarity

These bits define the polarity of the selected trigger event input defined by TRIGSEL[5:0].

00: no trigger (masked trigger event)

01: trigger on the rising edge

10: trigger on the falling edge

11: same as 00

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **TRIGSEL[5:0]**: trigger event input selection

These bits select the trigger event input of the GPDMA transfer (as per *Section 17.3.5*), with an active trigger event if TRIGPOL[1:0] ≠ 00.

Bits 15:14 **TRIGM[1:0]**: trigger mode

These bits define the transfer granularity for its conditioning by the trigger.

If the channel x is enabled (GPDMA_CxCR.EN asserted) with TRIGPOL[1:0] = 00 or 11, these TRIGM[1:0] bits are ignored.

Else, a GPDMA transfer is conditioned by at least one trigger hit:

00: at block level: the first burst read of each block transfer is conditioned by one hit trigger (channel x = 12 to 15, for each block if a 2D/repeated block is configured with GPDMA_CxBR1.BRC[10:0] ≠ 0).

01: channel x = 0 to 11, same as 00; channel x (x =12 to 15), at 2D/repeated block level. The first burst read of a 2D/repeated block transfer is conditioned by one hit trigger.

10: at link level: a LLI link transfer is conditioned by one hit trigger. The LLI data transfer (if any) is not conditioned.

11: at programmed burst level: If SWREQ = 1, each programmed burst read is conditioned by one hit trigger. If SWREQ = 0, each programmed burst that is requested by the selected peripheral, is conditioned by one hit trigger.

– If the peripheral is programmed as a source (DREQ = 0) of the LLI data transfer, each programmed burst read is conditioned.

– If the peripheral is programmed as a destination (DREQ = 1) of the LLI data transfer, each programmed burst write is conditioned. The first memory burst read of a (possibly 2D/repeated) block, also named as the first ready FIFO-based source burst, is gated by the occurrence of both the hardware request and the first trigger hit.

The GPDMA monitoring of a trigger for channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (TRIGPOL[1:0] = 01 or respectively TRIGPOL[1:0] = 10).

The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer; and if a new trigger is detected then, this hit is internally memorized to grant the next transfer, as long as the defined rising or falling edge is not modified, and the TRIGSEL[5:0] is not modified, and the channel is enabled.

Transferring a next LLI_{n+1} that updates the GPDMA_CxTR2 with a new value for any of TRIGSEL[5:0] or TRIGPOL[1:0], resets the monitoring, trashing the memorized hit of the formerly defined LLI_n trigger.

After a first new trigger hit_{n+1} is memorized, if another second trigger hit_{n+2} is detected and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized.memorized. A trigger overrun flag is reported (GPDMA_CxSR.TOF =1), and an interrupt is generated if enabled (GPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

Note: When the source block size is not a multiple of the source burst size and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, if TRIGM[1:0] = 11 and (SWREQ =1 or (SWREQ = 0 and DREQ =0)), the shortened burst transfer (by singles or/and by bursts of lower length) is conditioned once by the trigger.

When the programmed destination burst is internally shortened by singles or/and by bursts of lower length (versus FIFO size, versus block size, 1-Kbyte boundary address crossing): if the trigger is conditioning the programmed destination burst (if TRIGM[1:0] = 11 and SWREQ = 0 and DREQ = 1), this shortened destination burst transfer is conditioned once by the trigger.

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **BREQ**: Block hardware request

If the channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a burst level.

1: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a block level (see *Section 17.3.3*).

Bit 10 **DREQ**: destination hardware request

This bit is ignored if channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer). Else:

0: selected hardware request driven by a source peripheral (request signal taken into account by the GPDMA transfer scheduler over the source/read port)

1: selected hardware request driven by a destination peripheral (request signal taken into account by the GPDMA transfer scheduler over the destination/write port)

Note:

Bit 9 **SWREQ**: software request

This bit is internally taken into account when GPDMA_CxCR.EN is asserted.

0: no software request. The selected hardware request REQSEL[6:0] is taken into account.

1: software request for a memory-to-memory transfer. The default selected hardware request as per REQSEL[6:0] is ignored.

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:0 **REQSEL[6:0]**: GPDMA hardware request selection

These bits are ignored if channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer). Else, the selected hardware request is internally taken into account as per *Section 17.3.3*.

Caution: The user must not assign a same input hardware request (same REQSEL[6:0] value) to different active GPDMA channels (GPDMA_CxCR.EN = 1 and GPDMA_CxTR2.SWREQ = 0 for these channels). GPDMA is not intended to hardware support the case of simultaneous enabled channels incorrectly configured with a same hardware peripheral request signal, and there is no user setting error reporting.

17.8.12 GPDMA channel x block register 1 (GPDMA_CxBR1)

Address offset: 0x98 + 0x80 * x (x = 0 to 11)

Reset value: 0x0000 0000

This register is secure or non-secure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or non-privileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, or LLI or full linked-list.

In linked-list mode, during the link transfer:

- if GPDMA_CxLLR.UB1 = 1, this register is automatically updated by the GPDMA from the next LLI in memory.
- If GPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the field BNDT[15:0].
- If all the update bits GPDMA_CxLLR.Uxx are null and if GPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is automatically and internally restored with the programmed value for BNDT[15:0] after each execution of this final LLI
- If GPDMA_CxLLR = 0, this register and BNDT[15:0] are kept as null, channel x is completed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BNDT[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 **BNDT[15:0]**: block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this field becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):

- if GPDMA_CxLLR.UB1 = 1, this field is updated by the LLI in the memory.
- if GPDMA_CxLLR.UB1 = 0 and if there is at least one non null Uxx update bit, this field is internally restored to the programmed value.
- if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] = 0, this field is internally restored to the programmed value (infinite/continuous last LLI).
- if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI data transfer.

Note: A non-null source block size must be a multiple of the source data width (BNDT[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

When configured in packing mode (GPDMA_CxTR1.PAM[1] = 1 and destination data width different from source data width), a non-null source block size must be a multiple of the destination data width (BNDT[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

17.8.13 GPDMA channel x alternate block register 1 (GPDMA_CxBR1)

Address offset: 0x98 + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is secure or non-secure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or non-privileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, or LLI or full linked-list.

In linked-list mode, during the link transfer:

- if GPDMA_CxLLR.UB1 = 1, this register is automatically updated by the GPDMA from the next LLI in memory.
- If GPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the fields BNDT[15:0] and BRC[10:0].
- If all the update bits GPDMA_CxLLR.Uxx are null and if GPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is

automatically and internally restored with the programmed value for the fields BNDT[15:0] and BRC[10:0] after each execution of this final LLI

- If GPDMA_CxLLR = 0, BNDT[15:0] and BRC[10:0] are kept as null, channel x is completed.

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

Bit 31 **BRDDEC**: Block repeat destination address decrement

0: at the end of a block transfer, the GPDMA_CxDAR register is updated by adding the programmed offset GPDMA_CxBR2.BRDAO to the current GPDMA_CxDAR value (current destination address)

1: at the end of a block transfer, the GPDMA_CxDAR register is updated by subtracting the programmed offset GPDMA_CxBR2.BRDAO from the current GPDMA_CxDAR value (current destination address)

Note: On top of this increment/decrement (depending on BRDDEC), GPDMA_CxDAR is in the same time also updated by the increment/decrement (depending on DDEC) of the GPDMA_CxTR3.DAO value, as it is usually done at the end of each programmed burst transfer.

Bit 30 **BRSDEC**: Block repeat source address decrement

0: at the end of a block transfer, the GPDMA_CxSAR register is updated by adding the programmed offset GPDMA_CxBR2.BRSAO to the current GPDMA_CxSAR value (current source address)

1: at the end of a block transfer, the GPDMA_CxSAR register is updated by subtracting the programmed offset GPDMA_CxBR2.BRSAO from the current GPDMA_CxSAR value (current source address)

Note: On top of this increment/decrement (depending on BRSDEC), GPDMA_CxSAR is in the same time also updated by the increment/decrement (depending on SDEC) of the GPDMA_CxTR3.SAO value, as it is done after any programmed burst transfer.

Bit 29 **DDEC**: destination address decrement

0: At the end of a programmed burst transfer to the destination, the GPDMA_CxDAR register is updated by adding the programmed offset GPDMA_CxTR3.DAO to the current GPDMA_CxDAR value (current destination address)

1: At the end of a programmed burst transfer to the destination, the GPDMA_CxDAR register is updated by subtracting the programmed offset GPDMA_CxTR3.DAO to the current GPDMA_CxDAR value (current destination address)

Bit 28 **SDEC**: source address decrement

0: At the end of a programmed burst transfer from the source, the GPDMA_CxSAR register is updated by adding the programmed offset GPDMA_CxTR3.SAO to the current GPDMA_CxSAR value (current source address)

1: At the end of a programmed burst transfer from the source, the GPDMA_CxSAR register is updated by subtracting the programmed offset GPDMA_CxTR3.SAO to the current GPDMA_CxSAR value (current source address)

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 **BRC[10:0]**: Block repeat counter

This field contains the number of repetitions of the current block (0 to 2047).

When the channel is enabled, this field becomes read-only. After decrements, this field indicates the remaining number of blocks, excluding the current one. This counter is hardware decremented for each completed block transfer.

Once the last block transfer is completed (BRC[10:0] = BNDT[15:0] = 0):

– If GPDMA_CxLLR.UB1 = 1, all GPDMA_CxBR1 fields are updated by the next LLI in the memory.

– If GPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this field is internally restored to the programmed value.

– if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] ≠ 0, this field is internally restored to the programmed value (infinite/continuous last LLI).

– if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI and data transfer.

Bits 15:0 **BNDT[15:0]**: block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this field becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):

- if GPDMA_CxLLR.UB1 = 1, this field is updated by the LLI in the memory.

- if GPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this field is internally restored to the programmed value.

- if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] ≠ 0, this field is internally restored to the programmed value (infinite/continuous last LLI).

- if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI data transfer.

Note: A non-null source block size must be a multiple of the source data width (BNDT[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

When configured in packing mode (GPDMA_CxTR1.PAM[1]=1 and destination data width different from source data width), a non-null source block size must be a multiple of the destination data width (BNDT[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

17.8.14 GPDMA channel x source address register (GPDMA_CxSAR)

Address offset: 0x9C + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is secure or non-secure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the source start address of a transfer.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer from the source.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.USA = 1.

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

Bits 31:0 **SA[31:0]**: source address

This field is the pointer to the address from which the next data is read.

During the channel activity, depending on the source addressing mode (GPDMA_CxTR1.SINC), this field is kept fixed or incremented by the data width (GPDMA_CxTR1.SDW_LOG2[1:0]) after each burst source data, reflecting the next address from which data is read.

During the channel activity, this address is updated after each completed source burst, consequently to:

– the programmed source burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (GPDMA_CxTR1.SINC = 1), then the additional address offset value is the programmed burst size, as defined by GPDMA_CxTR1.SBL_1[5:0] and GPDMA_CxTR1.SDW_LOG2[21:0]

– the additional source incremented/decremented offset value as programmed by GPDMA_CxBR1.SDEC and GPDMA_CxTR3.SAO[12:0].

– once/if completed source block transfer, for a channel x with 2D addressing capability (x = 12 to 15). additional block repeat source incremented/decremented offset value as programmed by GPDMA_CxBR1.BRSDEC and GPDMA_CxBR2.BRSAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by GPDMA from the memory, provided the LLI is set with GPDMA_CxLLR.USA = 1.

Note: A source address must be aligned with the programmed data width of a source burst (SA[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

When the source block size is not a multiple of the source burst size and is a multiple of the source data width, the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional GPDMA_CxTR3.SAO[12:0] is not applied.

17.8.15 GPDMA channel x destination address register (GPDMA_CxDAR)

Address offset: 0xA0 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is secure or non-secure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the destination start address of a transfer.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer to the destination.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory if GPDMA_CxLLR.UDA = 1.

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

Bits 31:0 **DA[31:0]**: destination address

This field is the pointer to the address from which the next data is written.

During the channel activity, depending on the destination addressing mode (GPDMA_CxTR1.DINC), this field is kept fixed or incremented by the data width (GPDMA_CxTR1.DDW_LOG2[21:0]) after each burst destination data, reflecting the next address from which data is written.

During the channel activity, this address is updated after each completed destination burst, consequently to:

– the programmed destination burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (GPDMA_CxTR1.DINC = 1), then the additional address offset value is the programmed burst size, as defined by GPDMA_CxTR1.DBL_1[5:0] and GPDMA_CxTR1.DDW_LOG2[1:0]

– the additional destination incremented/decremented offset value as programmed by GPDMA_CxBR1.DDEC and GPDMA_CxTR3.DAO[12:0].

– once/if completed destination block transfer, for a channel x with 2D addressing capability (x = 12 to 15), the additional block repeat destination incremented/decremented offset value as programmed by GPDMA_CxBR1.BRDDEC and GPDMA_CxBR2.BRDAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by the GPDMA from the memory, provided the LLI is set with GPDMA_CxLLR.UDA = 1.

Note: A destination address must be aligned with the programmed data width of a destination burst (DA[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

17.8.16 GPDMA channel x transfer register 3 (GPDMA_CxTR3)

Address offset: 0xA4 + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is secure or non-secure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.UT3 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DAO[12:0]												
			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.	SAO[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **DAO[12:0]**: destination address offset increment

The destination address, pointed by GPDMA_CxDAR, is incremented or decremented (depending on GPDMA_CxBR1.DDEC) by this offset DAO[12:0] for each programmed destination burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (GPDMA_CxTR1.DINC = 1).

Note: A destination address offset must be aligned with the programmed data width of a destination burst (DAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **SAO[12:0]**: source address offset increment

The source address, pointed by GPDMA_CxSAR, is incremented or decremented (depending on GPDMA_CxBR1.SDEC) by this offset SAO[12:0] for each programmed source burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (GPDMA_CxTR1.SINC = 1).

Note: A source address offset must be aligned with the programmed data width of a source burst (SAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and none transfer is issued.

When the source block size is not a multiple of the destination burst size and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional GPDMA_CxTR3.SAO[12:0] is not applied.

17.8.17 GPDMA channel x block register 2 (GPDMA_CxBR2)

Address offset: 0xA8 + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is secure or non-secure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a 2D/repeated block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.UB2 = 1.

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

Bits 31:16 **BRDAO[15:0]**: Block repeated destination address offset

For a channel with 2D addressing capability, this field is used to update (by addition or subtraction depending on GPDMA_CxBR1.BRDDEC) the current destination address (GPDMA_CxDAR) at the end of a block transfer.

Note: A block repeated destination address offset must be aligned with the programmed data width of a destination burst (BRDAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

Bits 15:0 **BRSAO[15:0]**: Block repeated source address offset

For a channel with 2D addressing capability, this field is used to update (by addition or subtraction depending on GPDMA_CxBR1.BRSDEC) the current source address (GPDMA_CxSAR) at the end of a block transfer.

Note: A block repeated source address offset must be aligned with the programmed data width of a source burst (BRSAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

17.8.18 GPDMA channel x linked-list address register (GPDMA_CxLLR)

Address offset: 0xCC + 0x80 * x (x = 0 to 11)

Reset value: 0x0000 0000

This register is secure or non-secure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	rw	rw	rw											rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LA[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **UT1**: Update GPDMA_CxTR1 from memory

This bit controls the update of GPDMA_CxTR1 from the memory during the link transfer.

0: no GPDMA_CxTR1 update

1: GPDMA_CxTR1 update

Bit 30 **UT2**: Update GPDMA_CxTR2 from memory

This bit controls the update of GPDMA_CxTR2 from the memory during the link transfer.

0: no GPDMA_CxTR2 update

1: GPDMA_CxTR2 update

Bit 29 **UB1**: Update GPDMA_CxBR1 from memory

This bit controls the update of GPDMA_CxBR1 from the memory during the link transfer. If UB1 = 0 and if GPDMA_CxLLR ≠ 0, the linked-list is not completed.
GPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.

0: no GPDMA_CxBR1 update from memory (GPDMA_CxBR1.BNDT[15:0] restored if any link transfer)

1: GPDMA_CxBR1 update

Bit 28 **USA**: update GPDMA_CxSAR from memory

This bit controls the update of GPDMA_CxSAR from the memory during the link transfer.

0: no GPDMA_CxSAR update

1: GPDMA_CxSAR update

Bit 27 **UDA**: Update GPDMA_CxDAR register from memory
This bit is used to control the update of GPDMA_CxDAR from the memory during the link transfer.
0: no GPDMA_CxDAR update
1: GPDMA_CxDAR update

Bits 26:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update GPDMA_CxLLR register from memory
This bit is used to control the update of GPDMA_CxLLR from the memory during the link transfer.
0: no GPDMA_CxLLR update
1: GPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: pointer (16-bit low-significant address) to the next linked-list data structure
If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:20] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list GPDMA register file.
Else, this field is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list GPDMA internal register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR and GPDMA_CxLLR).

Note: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

Bits 1:0 Reserved, must be kept at reset value.

17.8.19 GPDMA channel x alternate linked-list address register (GPDMA_CxLLR)

Address offset: 0xCC + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is secure or non-secure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	UT3	UB2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	rw	rw	rw	rw	rw									rw

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

Bit 31 **UT1**: Update GPDMA_CxTR1 from memory

This bit controls the update of GPDMA_CxTR1 from the memory during the link transfer.

0: no GPDMA_CxTR1 update

1: GPDMA_CxTR1 update

Bit 30 **UT2**: Update GPDMA_CxTR2 from memory

This bit controls the update of GPDMA_CxTR2 from the memory during the link transfer.

0: no GPDMA_CxTR2 update

1: GPDMA_CxTR2 update

Bit 29 **UB1**: Update GPDMA_CxBR1 from memory

This bit controls the update of GPDMA_CxBR1 from the memory during the link transfer. If UB1 = 0 and if GPDMA_CxLLR ≠ 0, the linked-list is not completed. GPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.

0: no GPDMA_CxBR1 update from memory (GPDMA_CxBR1.BNDT[15:0] restored if any link transfer)

1: GPDMA_CxBR1 update

Bit 28 **USA**: update GPDMA_CxSAR from memory

This bit controls the update of GPDMA_CxSAR from the memory during the link transfer.

0: no GPDMA_CxSAR update

1: GPDMA_CxSAR update

Bit 27 **UDA**: Update GPDMA_CxDAR register from memory
This bit is used to control the update of GPDMA_CxDAR from the memory during the link transfer.
0:no GPDMA_CxDAR update
1: GPDMA_CxDAR update

Bit 26 **UT3**: Update GPDMA_CxTR3 from memory
This bit controls the update of GPDMA_CxTR3 from the memory during the link transfer.
0: no GPDMA_CxTR3 update
1: GPDMA_CxTR3 update

Bit 25 **UB2**: Update GPDMA_CxBR2 from memory
This bit controls the update of GPDMA_CxBR2 from the memory during the link transfer.
0: no GPDMA_CxBR2 update
1: GPDMA_CxBR2 update

Bits 24:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update GPDMA_CxLLR register from memory
This bit is used to control the update of GPDMA_CxLLR from the memory during the link transfer.
0: no GPDMA_CxLLR update
1: GPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: pointer (16-bit low-significant address) to the next linked-list data structure
If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:20] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list GPDMA register file.
Else, this field is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list GPDMA internal register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR and GPDMA_CxLLR).
Note: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

Bits 1:0 Reserved, must be kept at reset value.

17.8.20 GPDMA register map

Table 131. GPDMA 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	GPDMA_ SECCFGR	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																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	GPDMA_ PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	GPDMA_ RCFGLOCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK15	LOCK14	LOCK13	LOCK12	LOCK11	LOCK10	LOCK9	LOCK8	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 131. GPDMA register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C	GPDMA_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS15	MIS14	MIS13	MIS12	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	GPDMA_SMISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS15	MIS14	MIS13	MIS12	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14 - 0x4C	Reserved	Reserved																															
0x50 + 0x80 * x (x=0 to 15)	GPDMA_ CxLBAR	LBA[31:16]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x5C + 0x80 * x (x=0 to 15)	GPDMA_CxFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																		0	0	0	0	0	0	0								
0x60 + 0x80 * x (x=0 to 15)	GPDMA_CxSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOL[7:0]								Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDLEF
	Reset value									0	0	0	0	0	0	0	0		0	0	0	0	0	0	0								1
0x64 + 0x80 * x (x=0 to 15)	GPDMA_CxCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO[1:0]		Res.	Res.	Res.	Res.	LAP	LSM	Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	SUSP	RESET	EN
	Reset value									0	0					0	0		0	0	0	0	0	0	0					0	0	0
0x90 + 0x80 * x (x=0 to 15)	GPDMA_CxTR1	DSEC	DAP	Res.	Res.	DHX	DBX	DBL_1[5:0]						DINC	Res.	DDW_LOG2[1:0]		SSEC	SAP	SBX	PAM[1:0]		Res.	SBL_1[5:0]						SINC	Res.	SDW_LOG2[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
0x94 + 0x80 * x (x=0 to 15)	GPDMA_CxTR2	TCEM[1:0]		Res.	Res.	Res.	Res.	TRIGPOL[1:0]		Res.	Res.	TRIGSEL[5:0]						TRIGM[1:0]		Res.	BREQ	DREQ	SWREQ	Res.	Res.	REQSEL[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
0x98 + 0x80 * x (x=0 to 11)	GPDMA_CxBR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BNDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x98 + 0x80 * x (x=12 to 15)	GPDMA_CxBR1	BRDDEC	BRSDEC	DDEC	SDEC	Res.	BRC[10:0]											BNDT[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
0x9C+ 0x80 * x (x=0 to 15)	GPDMA_CxSAR	SA[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
0xA0 + 0x80 * x (x=0 to 15)	GPDMA_CxDAR	DA[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 131. GPDMA register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xA4 + 0x80 * x (x=12 to 15)	GPDMA_CxTR3	Res.	Res.	Res.	DAO[12:0]													Res.	Res.	Res.	SAO[12:0]												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0xA8 + 0x80 * x (x=12 to 15)	GPDMA_CxBR2	BRDAO[15:0]																BRSAO[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
0xCC + 0x80 * x (x=0 to 11)	GPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL	LA[15:2]														Res.	Res.
	Reset value	0	0	0	0	0											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0xCC + 0x80 * x (x=12 to 15)	GPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	UT3	UB2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL	LA[15:2]														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		

Refer to *Section 2.3* for the register boundary addresses.

18 Low-power direct memory access controller (LPDMA)

18.1 Introduction

The low-power direct memory access (LPDMA) controller is a bus master and system peripheral.

The LPDMA is used to perform programmable data transfers between memory-mapped peripherals and/or memories via linked-lists, upon the control of an off-loaded CPU.

18.2 LPDMA main features

- Single bidirectional AHB master
- Memory-mapped data transfers from a source to a destination:
  - Peripheral-to-memory
  - Memory-to-peripheral
  - Memory-to-memory
  - Peripheral-to-peripheral
- Autonomous data transfers during Sleep and Stop modes
- Transfers arbitration based on a 4-grade programmed priority at channel level:
  - One high-priority traffic class, for time-sensitive channels (queue 3)
  - Three low-priority traffic classes, with a weighted round-robin allocation for non time-sensitive channels (queues 0, 1, 2)
- Per channel event generation, on any of the following events: transfer complete, half transfer complete, data transfer error, user setting error, link transfer error, completed suspension and trigger overrun
- Per channel interrupt generation, with separately programmed interrupt enable per event
- 4 concurrent LPDMA channels:
  - Intra-channel LPDMA transfers chaining via programmable linked-list into memory, supporting two execution modes: run-to-completion and link step mode
  - Intra-channel and inter-channel LPDMA transfers chaining via programmable LPDMA input triggers connection to LPDMA task completion events
- Per linked-list item within a channel:
  - Separately programmed source and destination transfers
  - Programmable data handling between source and destination: byte-based padding or truncation, sign extension and left/right realignment
  - Programmable number of data bytes to be transferred from the source, defining the block level
  - Linear source and destination addressing: either fixed or contiguously incremented addressing, programmed at a block level, between successive single transfers
  - Programmable LPDMA request and trigger selection
  - Programmable LPDMA half-transfer and transfer-complete events generation

  - Pointer to the next linked-list item and its data structure in memory, with automatic update of the LPDMA linked-list control registers
- Debug:
  - Channel suspend and resume support
  - Channel status reporting and event flags
- TrustZone support
  - Support for secure and non-secure LPDMA transfers, independently at a first channel level, and independently at a source/destination and link sub-levels
  - Secure and non-secure interrupts reporting, resulting from any of the respectively secure and non-secure channels
  - TrustZone-aware AHB slave port, protecting any LPDMA secure resource (register, register field) from a non-secure access
- Privileged/unprivileged support:
  - Support for privileged and unprivileged LPDMA transfers, independently at a channel level
  - Privileged-aware AHB slave port

18.3 LPDMA implementation

18.3.1 LPDMA channels

A given LPDMA channel x is implemented with the hardware parameters as per the table below.

Table 132.LPDMA1 channels implementation

Channel x	Hardware parameters		Features
	dma_fifo_ size[x]	dma_ addressing[x]	
x = 0 to 3	0	0	Channel x (x = 0 to 3) is implemented with: – no FIFO. Only a single source transfer cell is internally registered. – fixed/contiguously incremented addressing

18.3.2 LPDMA autonomous mode in low-power modes

The LPDMA autonomous mode and wakeup feature is implemented in the device low-power modes as per the table below.

Table 133. LPDMA1 autonomous mode and wakeup in low-power modes

Feature	Low-power modes
Autonomous mode and wakeup	LPDMA1 in Sleep, Stop 0, Stop 1and Stop 2 modes

18.3.3 LPDMA requests

A LPDMA request from a peripheral can be assigned to a LPDMA channel x, via the REQSEL[4:0] field in the *LPDMA channel x transfer register 2 (LPDMA_CxTR2)*, provided that SWREQ = 0.

The LPDMA requests mapping is specified in the table below.

Table 134. Programmed LPDMA1 request

LPDMA_CxTR2.REQSEL[4:0]	Selected LPDMA request
0	lpuart1_rx_dma
1	lpuart1_tx_dma
2	spi3_rx_dma
3	spi3_tx_dma
4	i2c3_rx_dma
5	i2c3_tx_dma
6	i2c3_evc_dma
7	adc4_dma
8	dac1_ch1_dma
9	dac1_ch2_dma
10	adf1_flt0_dma
11	lptim1_ic1_dma
12	lptim1_ic2_dma
13	lptim1_ue_dma
14	lptim3_ic1_dma
15	lptim3_ic2_dma
16	lptim3_ue_dma

18.3.4 LPDMA block requests

Some LPDMA requests must be programmed as a block request, and not as a single request. Then the BREQ bit in *LPDMA channel x transfer register 2 (LPDMA_CxTR2)* must be set for a correct LPDMA execution of the requested peripheral transfer at the hardware level.

The LPDMA block requests are listed in the table below.

Table 135. Programmed LPDMA1 request as a block request

LPDMA block requests
lptim1_ue_dma
lptim3_ue_dma

18.3.5 LPDMA triggers

A LPDMA trigger can be assigned to a LPDMA channel x, via the TRIGSEL[4:0] field in the *LPDMA channel x transfer register 2 (LPDMA_CxTR2)*, provided that TRIGPOL[1:0] defines a rising or a falling edge of the selected trigger (TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

The LPDMA triggers mapping is specified in the table below.

Table 136.Programmed LPDMA1 trigger

LPDMA_CxTR2.TRIGSEL[4:0]	Selected LPDMA trigger
0	exti0
1	exti1
2	exti2
3	exti3
4	exti4
5	tamp_trg1
6	tamp_trg2
7	tamp_trg3
8	lptim1_ch1
9	lptim1_ch2
10	lptim3_ch1
11	lptim4_out
12	comp1_out
13	comp2_out
14	rtc_alra_trg
15	rtc_alrb_trg
16	rtc_wut_trg
17	adc4_awd1
18	lpdma1_ch0_tc
19	lpdma1_ch1_tc
20	lpdma1_ch2_tc
21	lpdma1_ch3_tc
22	gpdma1_ch0_tc
23	gpdma1_ch1_tc
24	gpdma1_ch4_tc
25	gpdma1_ch5_tc
26	gpdma1_ch6_tc
27	gpdma1_ch7_tc
28	gpdma1_ch12_tc
29	gpdma1_ch13_tc

Table 136.Programmed LPDMA1 trigger (continued)

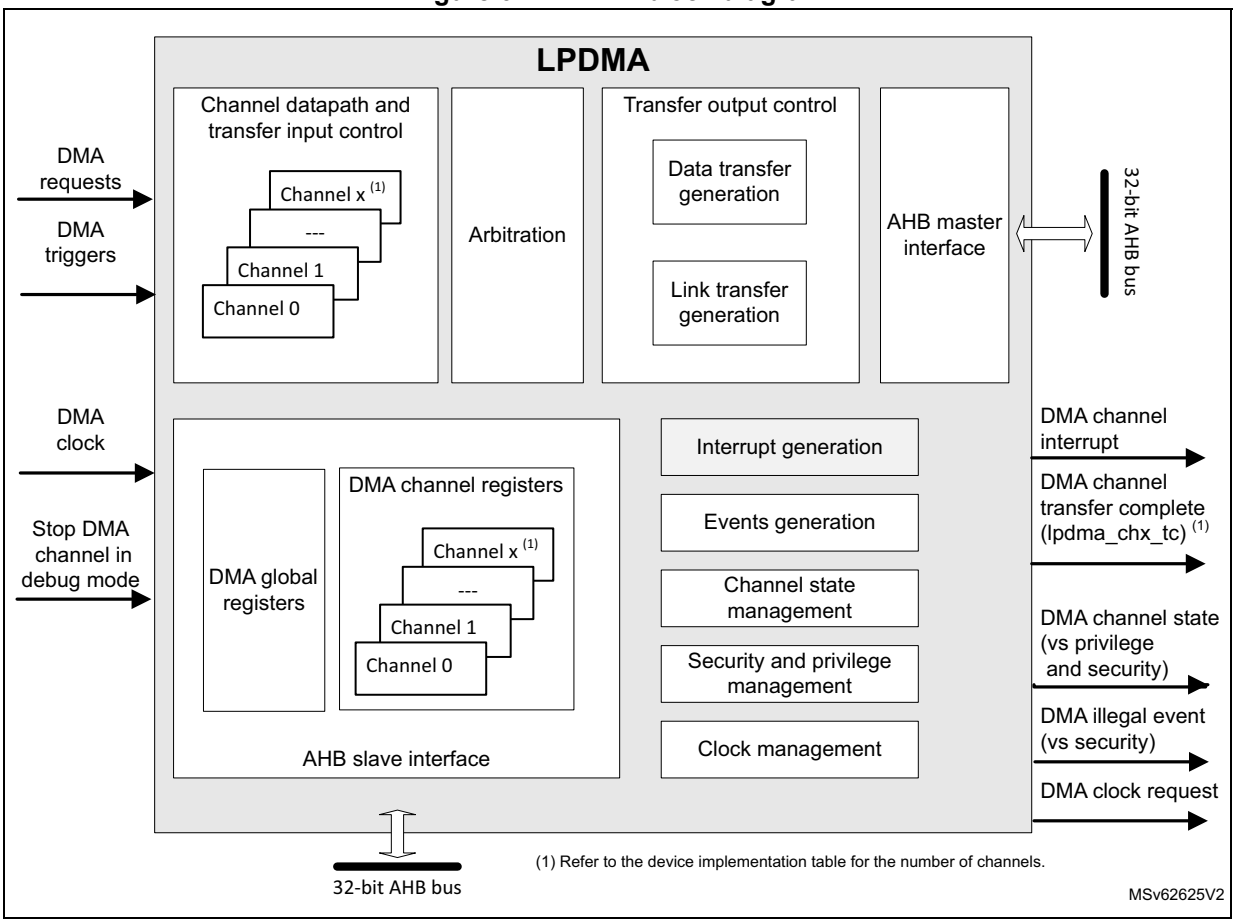
LPDMA_CxTR2.TRIGSEL[4:0]	Selected LPDMA trigger
30	tim2_trgo
31	tim15_trgo

18.4 LPDMA functional description

18.4.1 LPDMA block diagram

The LPDMA block diagram is illustrated in the figure below.

Figure 67. LPDMA block diagram



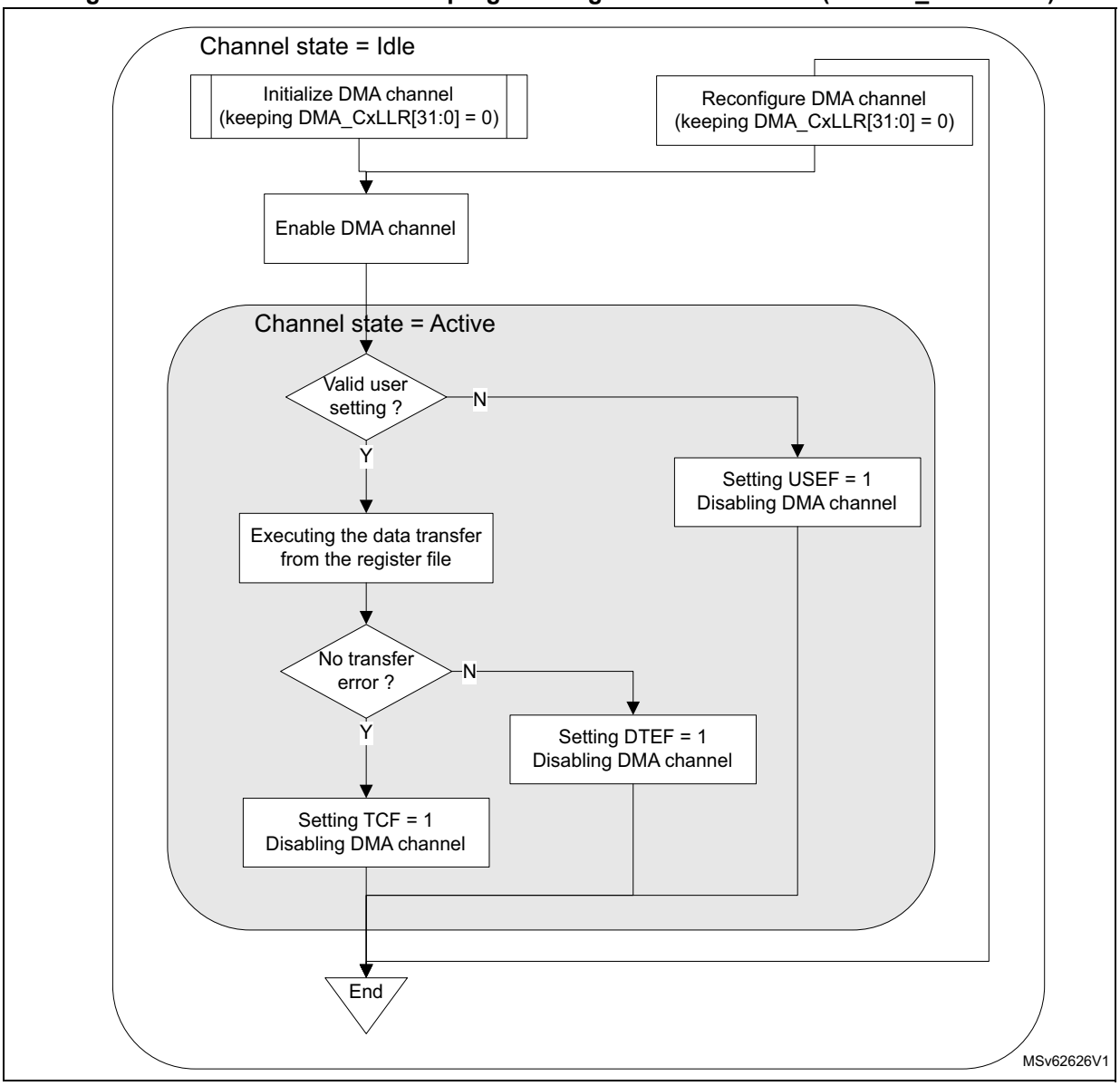
18.4.2 LPDMA channel state and direct programming without any linked-list

After a LPDMA reset, a LPDMA channel x is in idle state. When the software writes 1 into the in LPDMA_CxCR.EN enable control bit, the channel takes into account the value of the different channel configuration registers (LPDMA_CxXXX), switches to the active/non-idle state and starts to execute the corresponding requested data transfers.

After enabling/starting a LPDMA channel transfer by writing 1 into the LPDMA_CxCR.EN bit, a LPDMA channel interrupt on a complete transfer notifies the software that the LPDMA channel is back in idle state (EN is then de-asserted by hardware) and that the channel is ready to be reconfigured then enabled again.

The figure below illustrates this LPDMA direct programming without any linked-list (LPDMA_CxLLR = 0).

Figure 68. LPDMA channel direct programming without linked-list (LPDMA_CxLLR = 0)



18.4.3 LPDMA channel suspend and resume

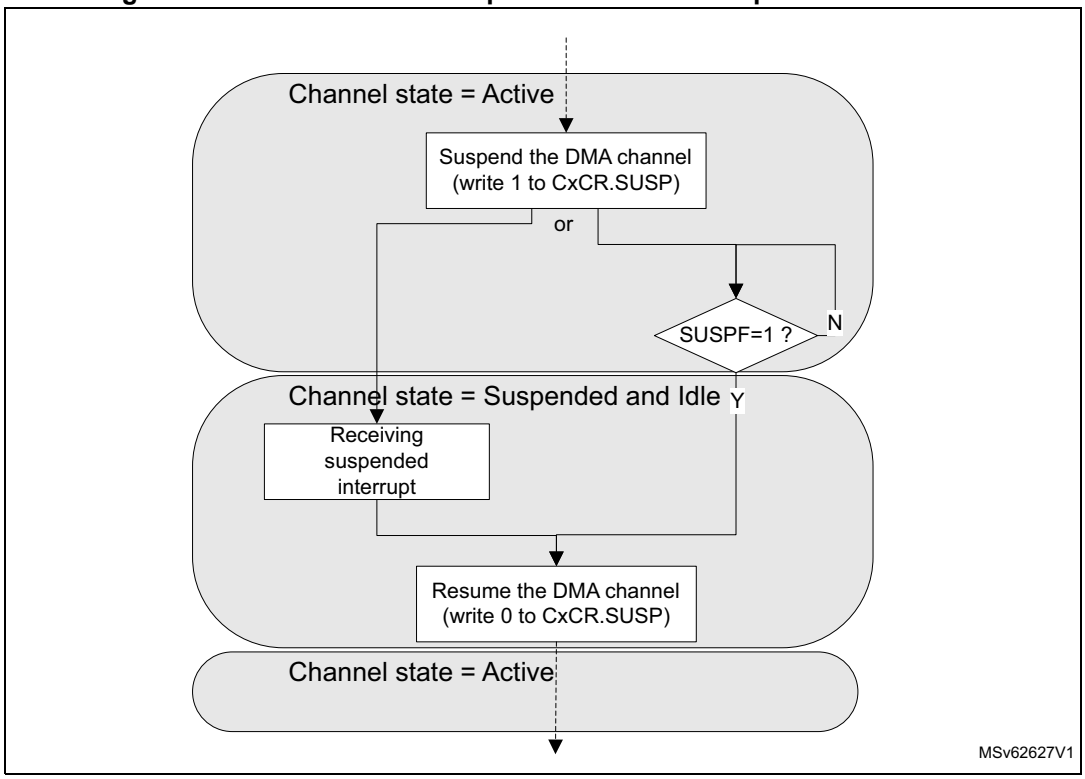
The software can suspend on its own a channel still active, with the following sequence:

1. The software writes 1 into the LPDMA_CxCR.SUSP bit.
2. The software polls the suspended flag LPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to LPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing LPDMA transfer over its master port. Then the software can observe, in a steady state, any read register or register field that is hardware modifiable.

   Note that an ongoing LPDMA transfer can be a data transfer (a read-followed-by-write single transfer) or a link transfer for the internal update of the linked-list register file from the next linked-list item.
3. The software safely resumes the suspended channel by writing 0 to LPDMA_CxCR.SUSP.

The suspend and resume sequence is illustrated in the figure below.

Figure 69. LPDMA channel suspend and resume sequence



Note: A suspend and resume sequence does not impact the LPDMA_CxCR.EN bit. Suspending a channel (transfer) does not suspend a started trigger detection.

18.4.4 LPDMA channel abort and restart

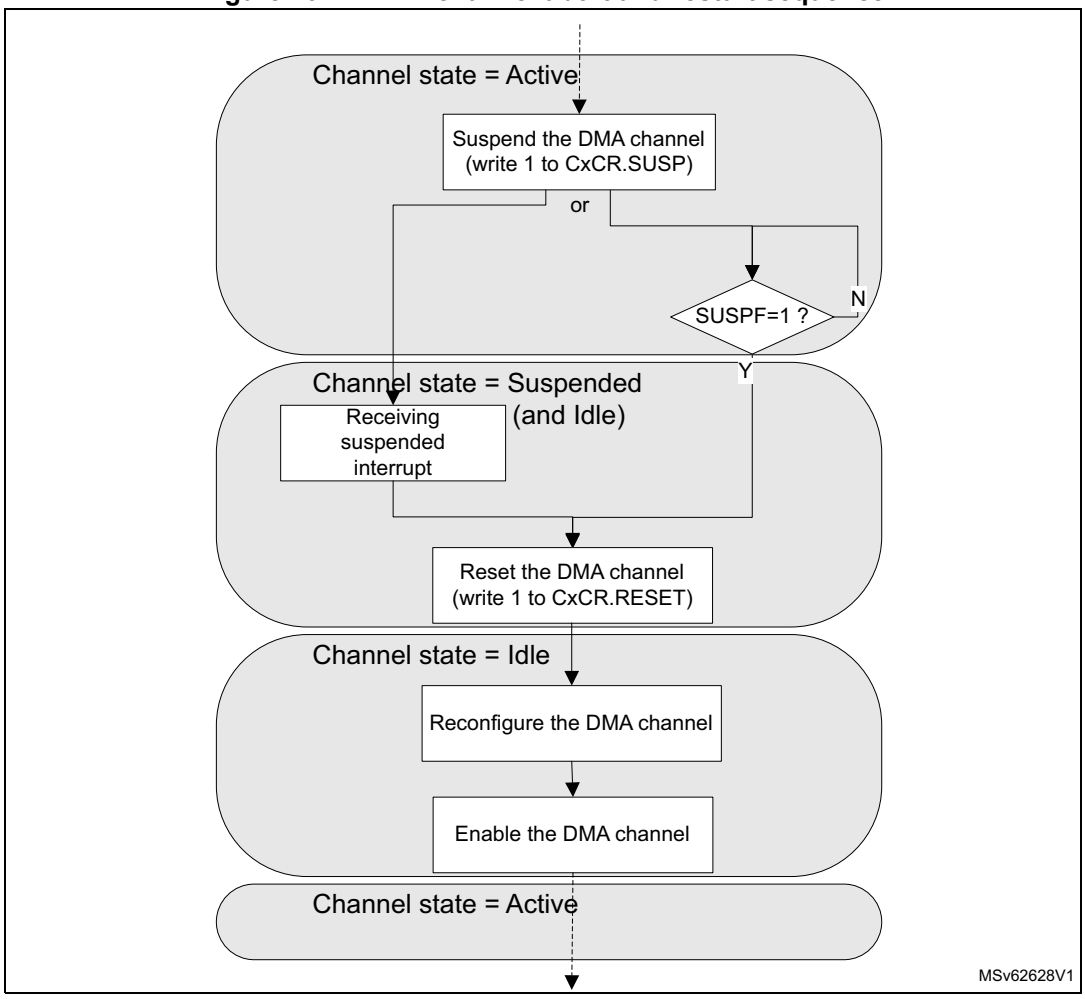
Alternatively, like for aborting a continuous LPDMA transfer with a circular buffering or a double buffering, the software can abort, on its own, a still active channel with the following sequence:

1. The software writes 1 into the LPDMA_CxCR.SUSP bit.

2. The software polls suspended flag LPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to LPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing LPDMA transfer over its master port.
3. The software resets the channel by writing 1 to LPDMA_CxCR.RESET. This causes the reset of the channel internal state, the reset of the LPDMA_CxCR.EN bit, and the reset of the LPDMA_CxCR.SUSP bit.
4. The software safely reconfigures the channel. The software must reprogram the hardware-modified LPDMA_CxBR1, LPDMA_CxSAR, and LPDMA_CxDAR registers.
5. In order to restart the aborted then reprogrammed channel, the software enables it again by writing 1 to the LPDMA_CxCR.EN bit.

The abort and restart sequence is illustrated in the figure below.

Figure 70. LPDMA channel abort and restart sequence



18.4.5 LPDMA linked-list data structure

Alternatively to the direct programming mode, a channel can be programmed by a list of transfers, known as a list of linked-list items (LLI). Each LLI is defined by its data structure.

The base address in memory of the data structure of a next LLI_{n+1} of a channel x is the sum of the following:

- the link base address of the channel x (in LPDMA_CxLBAR)
- the link address offset (LA[15:2] field in LPDMA_CxLLR)

The data structure for each LLI may be specific.

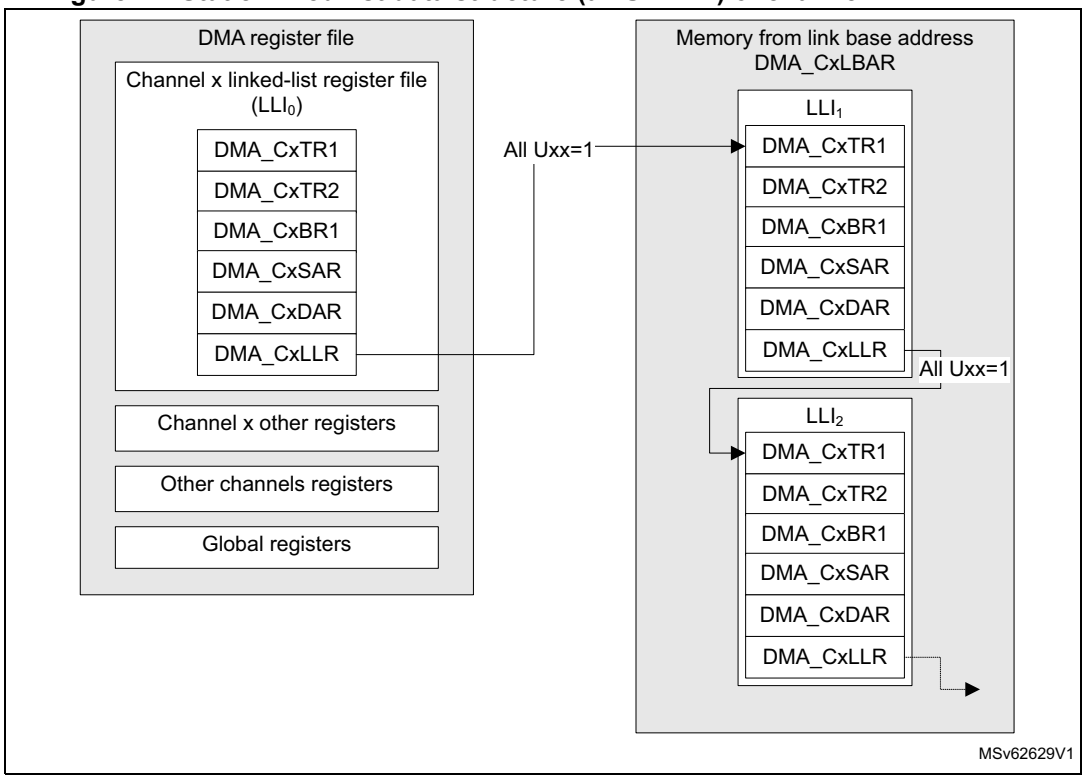
A linked-list data structure is addressed following the value of the UT1, UT2, UB1, USA, UDA and ULL bits of the LPDMA_CxLLR register.

In linked-list mode, each LPDMA linked-list register (LPDMA_CxTR1, LPDMA_CxTR2, LPDMA_CxBR1, LPDMA_CxSAR, LPDMA_CxDAR or LPDMA_CxLLR) is conditionally and automatically updated from the next linked-list data structure in the memory, following the current value of the LPDMA_CxLLR register that was conditionally updated from the linked-list data structure of the previous LLI.

Static linked-list data structure

For example, when the update bits (UT1, UT2, UB1, USA, UDA and ULL) of the LPDMA_CxLLR register are all asserted, the linked-list data structure in memory is maximal with six contiguous 32-bit locations, including LPDMA_CxTR1, LPDMA_CxTR2, LPDMA_CxBR1, LPDMA_CxSAR, LPDMA_CxDAR and LPDMA_CxLLR (see the figure below) and including the first linked-list register file (LLI_0) and the next LLIs (LLI_1, LLI_2,...) in the memory.

Figure 71. Static linked-list data structure (all Uxx = 1) of channel x



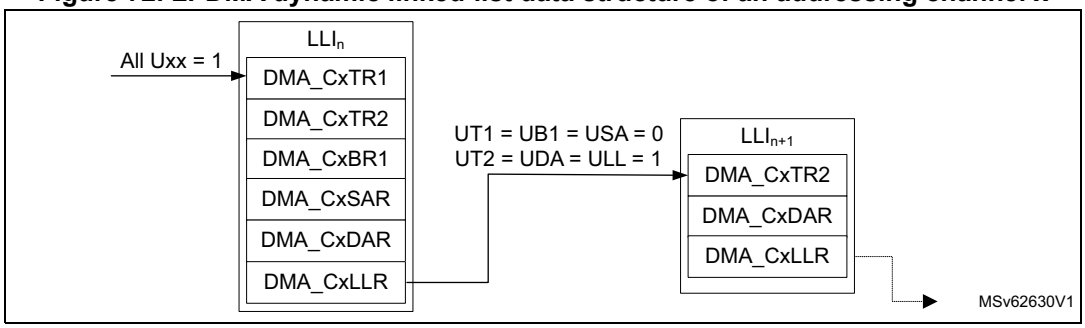
Dynamic linked-list data structure

Alternatively, the memory organization for the full list of LLIs can be compacted with specific data structure for each LLI.

If UT1 = 0 and UT2 = 1, the link address offset of the register LPDMA_CxLLR is pointing to the updated value of the LPDMA_CxTR2 instead of the LPDMA_CxTR1 which is not to be modified.

Example: if UT1 = UB1 = USA = 0, and if UT2 = UDA = ULL = 1, the next LLI does not contain an updated value for LPDMA_CxTR1, nor LPDMA_CxBR1, nor LPDMA_CxSAR. The next LLI contains an updated value for LPDMA_CxTR2, LPDMA_CxDAR, and LPDMA_CxLLR, as shown in the figure below.

Figure 72. LPDMA dynamic linked-list data structure of an addressing channel x



The user must program LPDMA_CxLLR for each LLI_n to be 32-bit aligned and not to exceed the 64-Kbyte addressable space pointed by LPDMA_CxLBAR.

18.4.6 Linked-list item transfer execution

A LLI_n transfer is the sequence of:

1. a data transfer: LPDMA executes the data transfer as described by the LPDMA internal register file (this data transfer can be void/null for LLI_0)
2. a conditional link transfer: LPDMA automatically and conditionally updates its internal register file by the data structure of the next LLI_{n+1}, as defined by the LPDMA_CxLLR value of the LLI_n.

Note: The initial data transfer as defined by the internal register file (LLI_0) can be null (LPDMA_CxBR1.BNDT[15:0] = 0) provided that the conditional update bit UB1 in LPDMA_CxLLR is set (meaning there is a non-null data transfer described by the next LLI_1 in the memory to be executed).

Depending on the intended LPDMA usage, a LPDMA channel x can be executed as described by the full linked-list (run-to-completion mode, LPDMA_CxCR.LSM = 0) or a LPDMA channel x can be programmed for a single execution of a LLI (link step mode, LPDMA_CxCR.LSM = 1), as described in the next paragraphs.

18.4.7 LPDMA channel state and linked-list programming in run-to-completion mode

When LPDMA_CxCR.LSM = 0, a LPDMA channel x is initially programmed, started by writing 1 to LPDMA_CxCR.EN, and after (possibly) completed at channel level. The channel transfer is:

- configured with at least the following:
  - the first LLI_0, internal linked-list register file: LPDMA_CxTR1, LPDMA_CxTR2, LPDMA_CxBR1, LPDMA_CxSAR, LPDMA_CxDAR and LPDMA_CxLLR
  - the last LLI_N, described by the linked-list data structure in memory, as defined by the LPDMA_CxLLR reflecting the before last LLI_{N-1}
- completed when LPDMA_CxLLR[31:0] = 0 and LPDMA_CxBR1.BNDT[15:0] = 0, at the end of the last LLI_{N-1} transfer

LPDMA_CxLLR[31:0] = 0 is the condition of a linked-list based channel completion and means the following:

- The 16 low significant bits LPDMA_CxLLR.LA[15:0] of the next link address are null.
- All the update bits Uxx of LPDMA_CxLLR are null (UT1, UT2, UB1, USA, UDA, ULL).

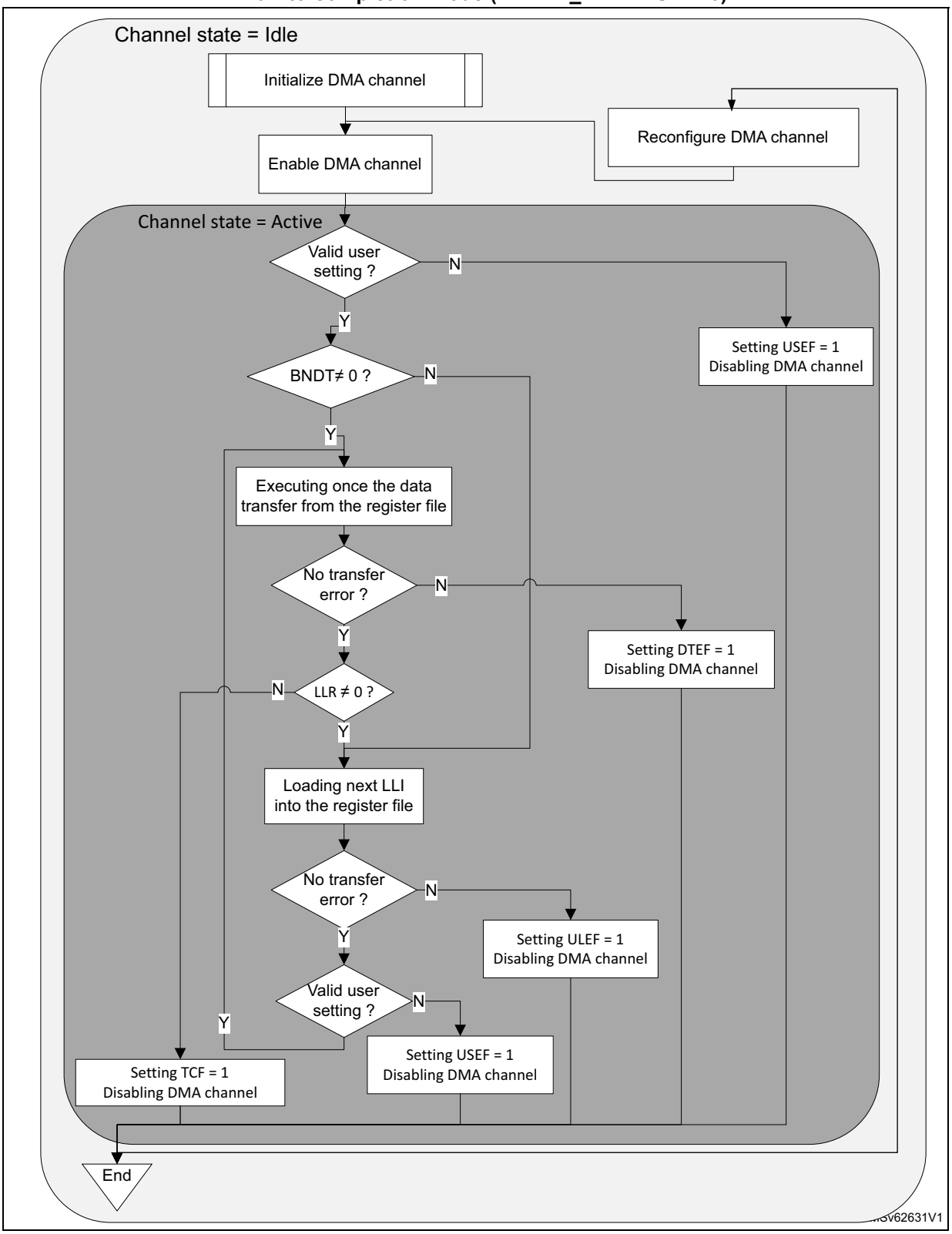
The channel may never be completed when LPDMA_CxLLR.LSM = 0:

- If the last LLI_N is recursive, pointing to itself as a next LLI:
  - either LPDMA_CxLLR.ULL = 1 and LPDMA_CxLLR.LA[15:2] is updated by the same value
  - or LPDMA_CxLLR.ULL = 0
- if LLI_N is pointing to a previous LLI.

In the typical run-to-completion mode, the allocation of a LPDMA channel, including its fine programming, is done once during the LPDMA initialization. In order to have a reserved data communication link and LPDMA service during run-time, for continuously repeated transfers (from/to a peripheral respectively to/from memory or for memory-to-memory transfers). This reserved data communication link can consist of a channel, or the channel can be shared and a repeated transfer consists of a sequence of LLIs.

Figure 73 depicts the LPDMA channel execution and its registers programming in run-to-completion mode.

Figure 73. LPDMA channel execution and linked-list programming in run-to-completion mode (LPDMA_CxCR.LSM = 0)



Run-time inserting a LLI_n via an auxiliary channel, in run-to-completion mode

The start of the link transfer of the LLI_{n-1} (start of the LLI_n loading) can be conditioned by the occurrence of a trigger, when programming the following fields of the LPDMA_CxTR2 in the data structure of the LLI_{n-1}:

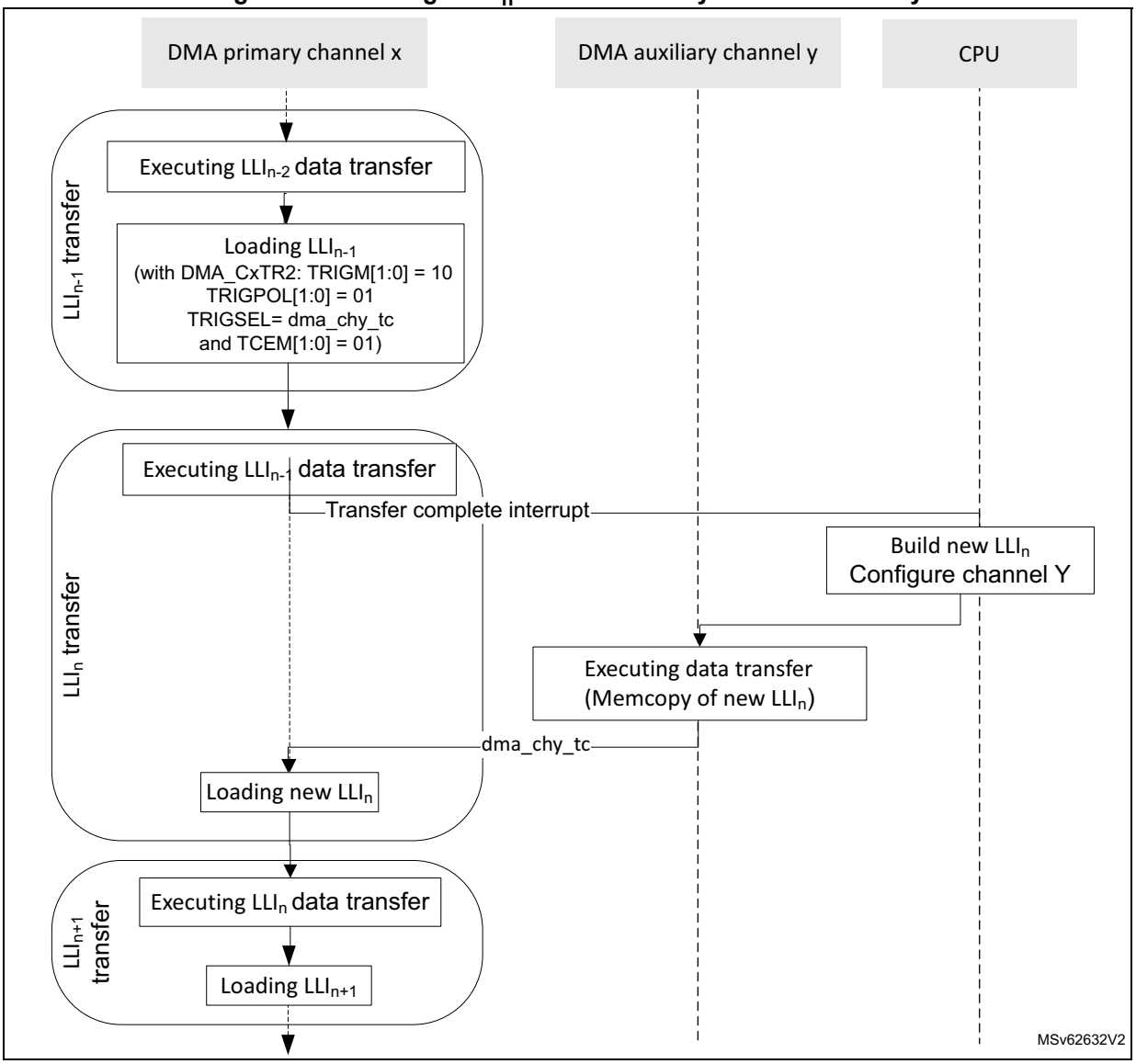
- TRIGM[1:0] = 10 (link transfer triggering mode)
- TRIGPOL[1:0] = 01 or 10 (rising or falling edge)
- TRIGSEL[4:0] (see *Section 18.3.5* for the trigger selection details)

Another auxiliary channel y can be used to store the channel x LLI_n in the memory and to generate a transfer complete event lpdma_chy_tc. By selecting this event as the input trigger of the link transfer of the LLI_{n-1} of the channel x, the software can pause the primary channel x after its LLI_{n-1} data transfer, until it is indeed written the LLI_n.

Figure 74 depicts such a dynamic elaboration of a linked-list of a primary channel x, via another auxiliary channel y.

Caution: This use case is restricted to an application with a LLI_{n-1} data transfer that does not need a trigger. The triggering mode of this LLI_{n-1} is used to load the next LLI_n.

Figure 74. Inserting a LLI_n with an auxiliary LPDMA channel y



18.4.8 LPDMA channel state and linked-list programming in link step mode

When LPDMA_CxCR.LSM = 1, a channel transfer is executed and completed after each single execution of a LLI, including its (conditional) data transfer and its (conditional) link transfer.

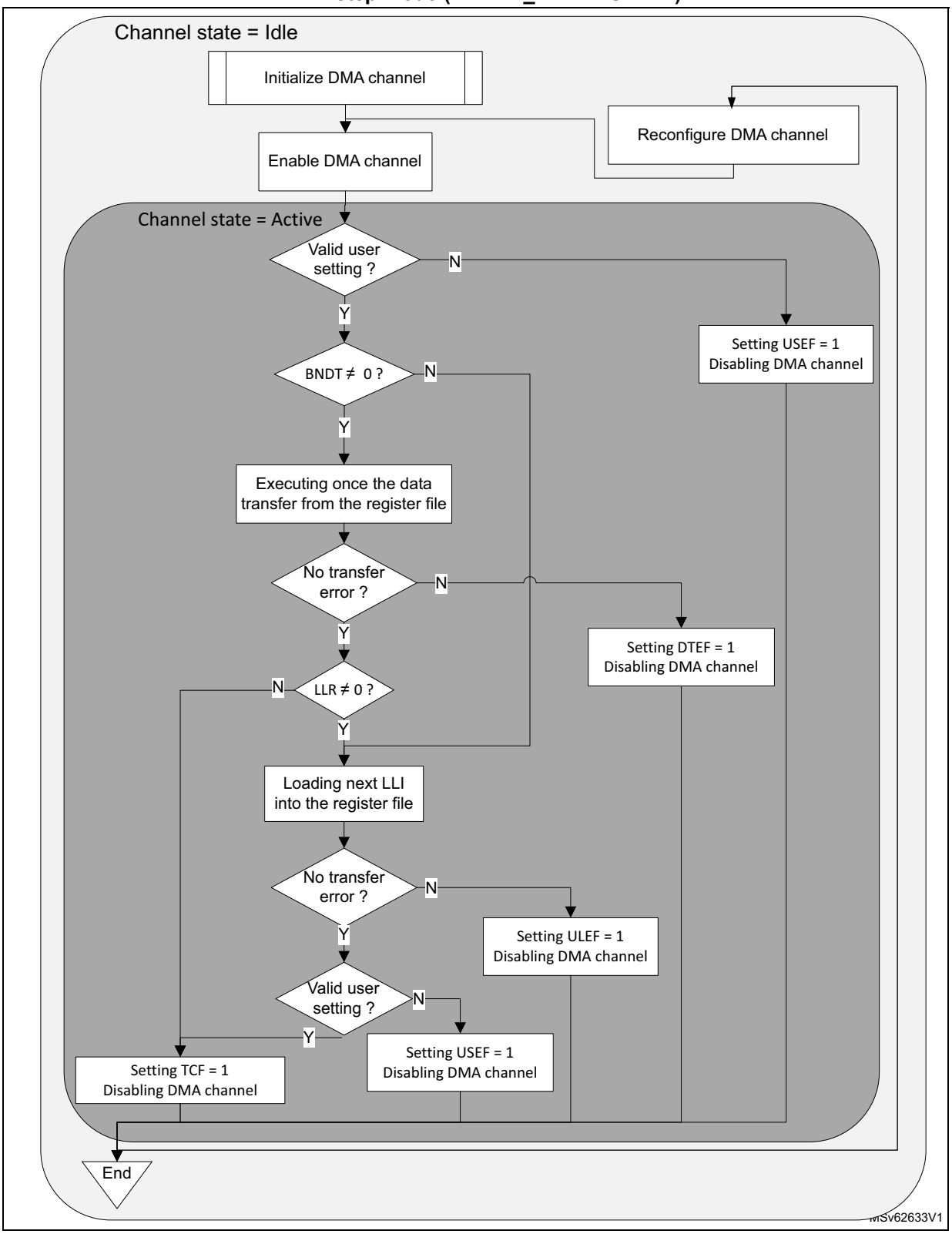
A LPDMA channel transfer can be programmed at LLI level, started by writing 1 into LPDMA_CxCR.EN, and after completed at LLI level:

- The current LLI_n transfer is described with:
  - LPDMA_CxTR1 defines the source/destination elementary single transfers.
  - LPDMA_CxBR1 defines the number of bytes at a block level (BNDT[15:0]).
  - LPDMA_CxTR2 defines the input control (request, trigger) and the output control (transfer complete event) of the transfer.
  - LPDMA_CxSAR/LPDMA_CxDAR defines the source/destination transfer start address.
  - LPDMA_CxLLR defines the data structure and the address offset of the next LLI_{n+1} in the memory.
- The current LLI_n transfer is completed after the single execution of the current LLI_n:
  - after the (conditional) data transfer completion (when LPDMA_CxBR1.BNDT[15:0] = 0)
  - after the (conditional) update of the LPDMA link register file from the data structure of the next LLI_{n+1} in memory

Note: If a LLI is recursive (pointing to itself as a next LLI, either LPDMA_CxLLR.ULL = 1 and LPDMA_CxLLR.LA[15:2] is updated by the same value, or LPDMA_CxLLR.ULL = 0), a channel in link step mode is completed after each repeated single execution of this LLI.

Figure 75 depicts the LPDMA channel execution mode, and its programming in link step mode.

Figure 75. LPDMA channel execution and linked-list programming in link step mode (LPDMA_CxCR.LSM = 1)

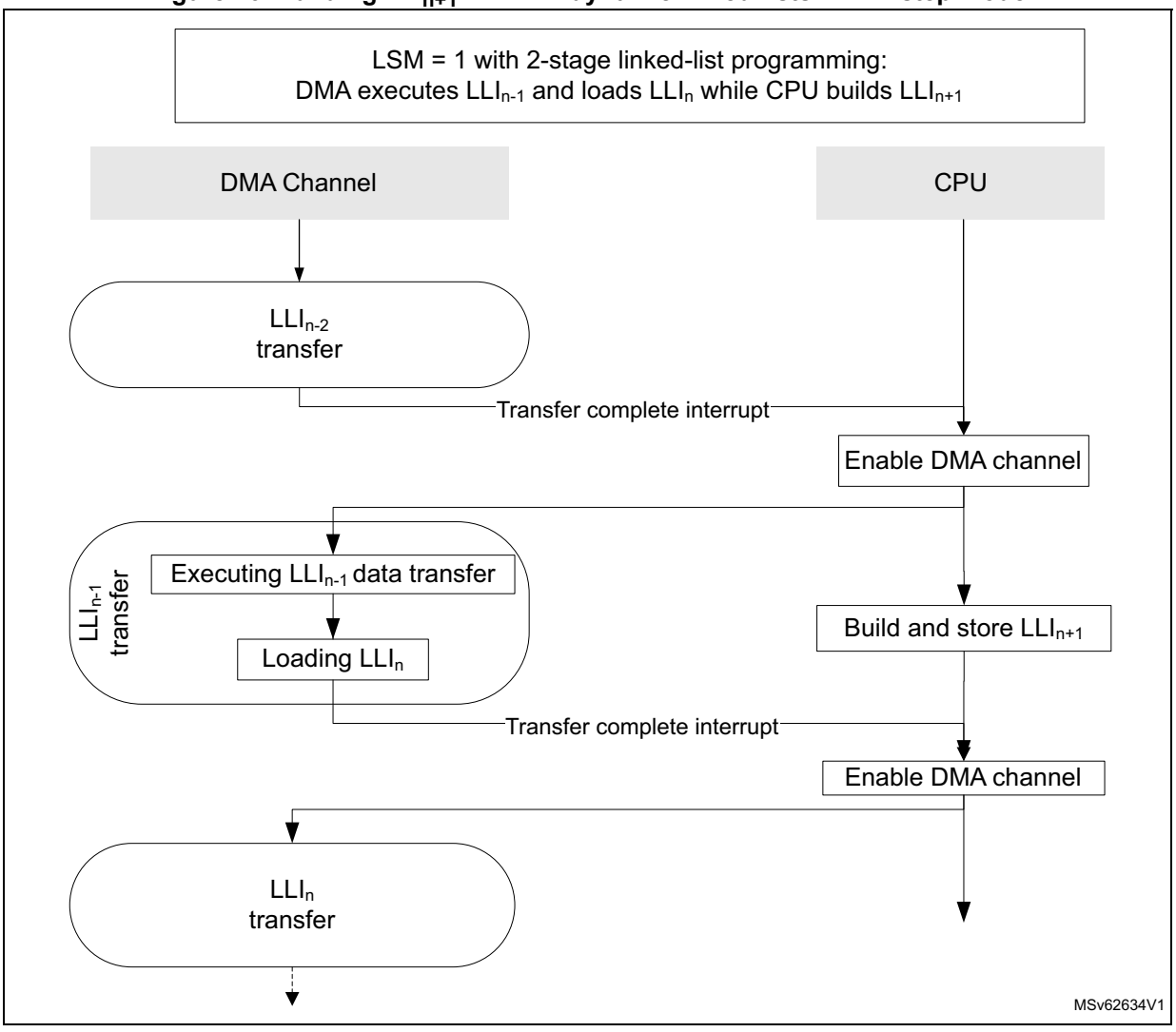


The link step mode can be used to elaborate dynamically LLIs in memory during run-time. The software can be facilitated by using a static data structure for any LLI_n (all update bits of LPDMA_CxLLR have a static value, LLI_n.LLR.LA = LLI_{n-1}.LLR.LA + constant).

Run-time adding a LLI_{n+1} in link step mode

During run-time, the software can defer the elaboration of the LLI_{n+1} (and next LLIs), until/after LPDMA executed the transfer from the LLI_{n-1} and loaded the LLI_n from the memory, as shown in the figure below.

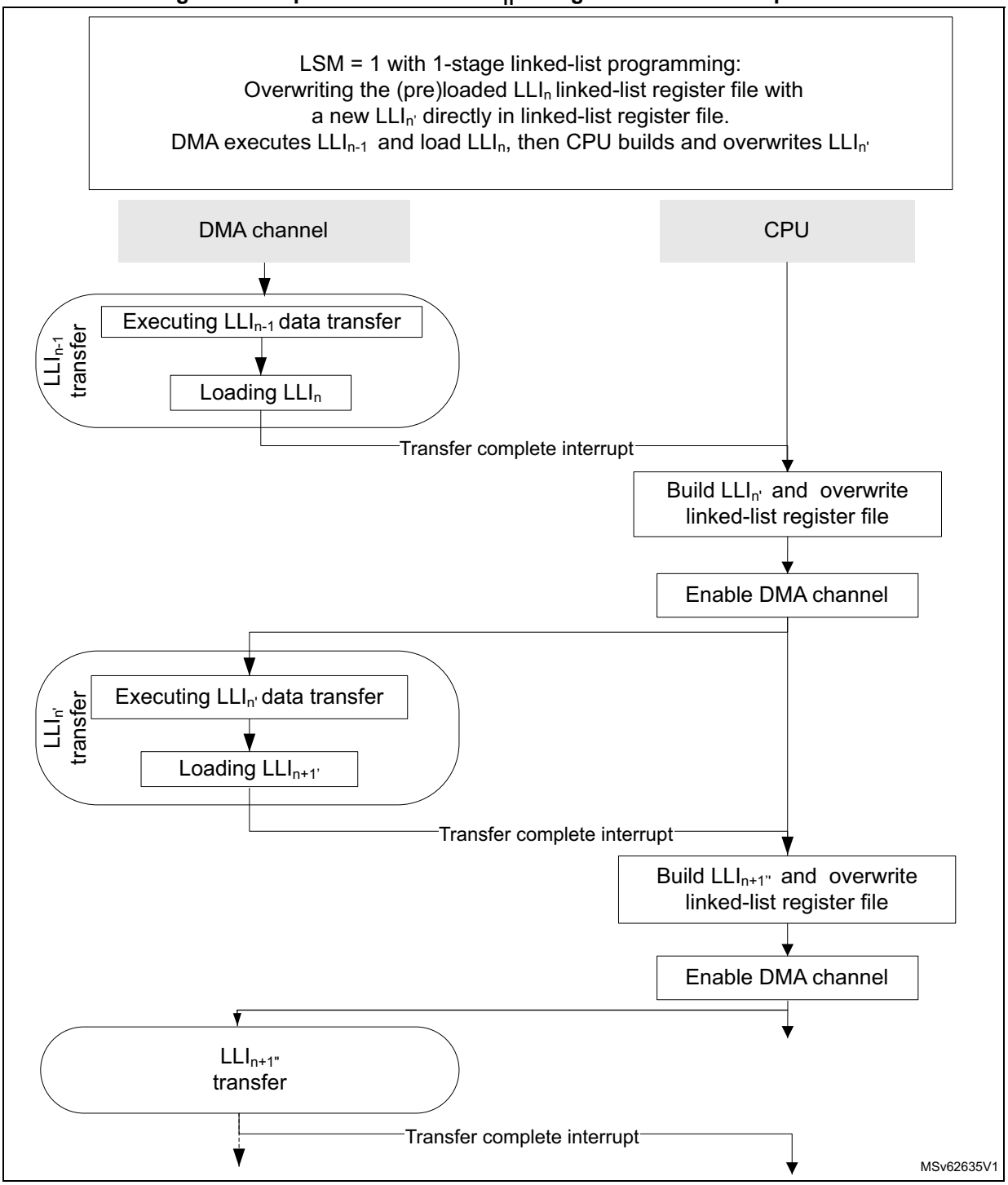
Figure 76. Building LLI_{n+1}: LPDMA dynamic linked-lists in link step mode



Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode (in linked-list register file)

In this link step mode, during run-time, the software can build and insert a new $LLI_{n'}$, after LPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_n from the memory by overwriting directly the linked-list register file with the new $LLI_{n'}$, as shown in *Figure 77*.

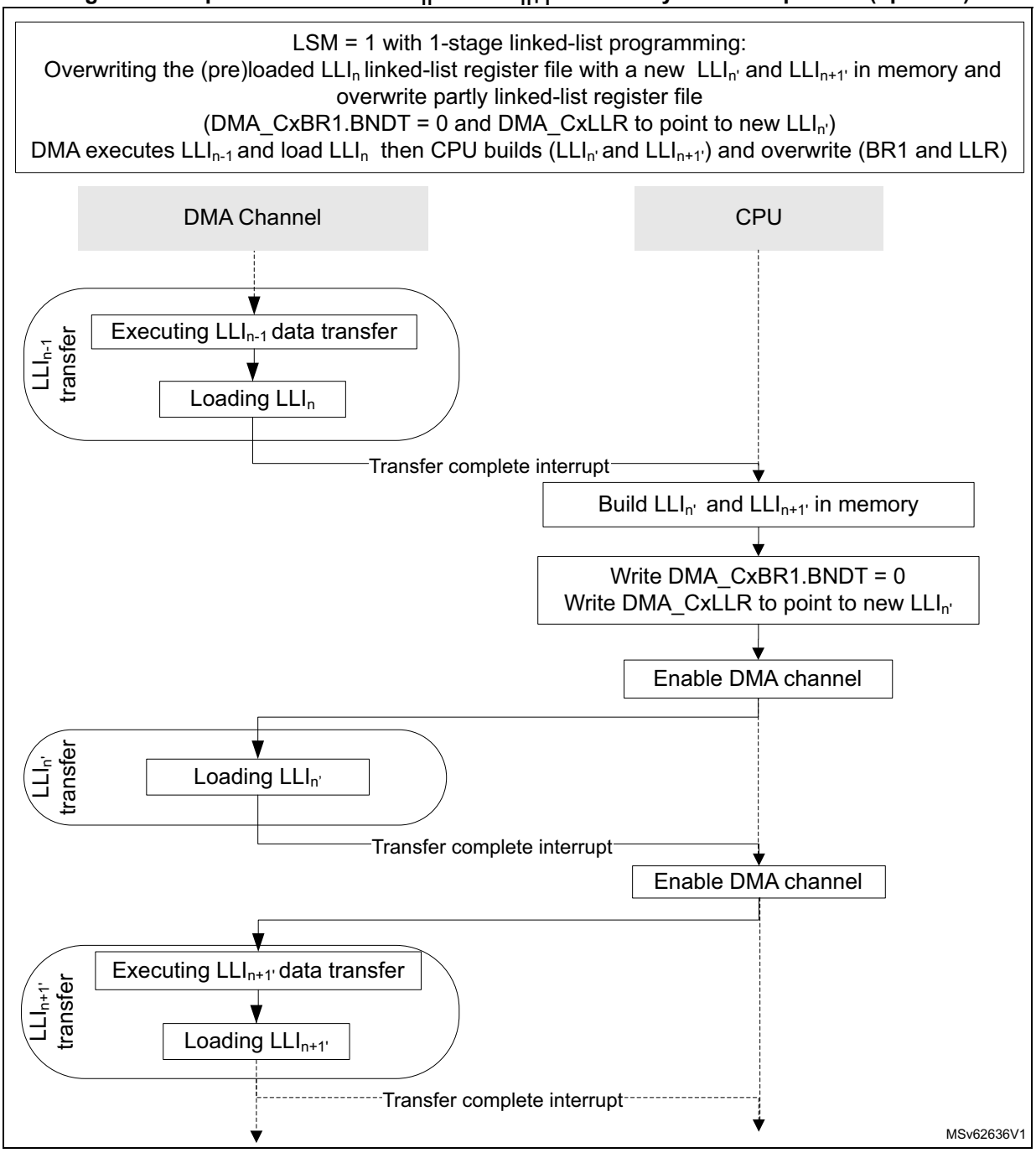
Figure 77. Replace with a new $LLI_{n'}$ in register file in link step mode



Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode (in the memory)

The software can build and insert a new $LLI_{n'}$ and $LLI_{n+1'}$ in the memory, after LPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_n from the memory, by overwriting partly the linked-list register file (LPDMA_CxBR1.BNDT[15:0] to be null and LPDMA_CxLLR to point to new $LLI_{n'}$) as shown in *Figure 78*.

Figure 78. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 1)



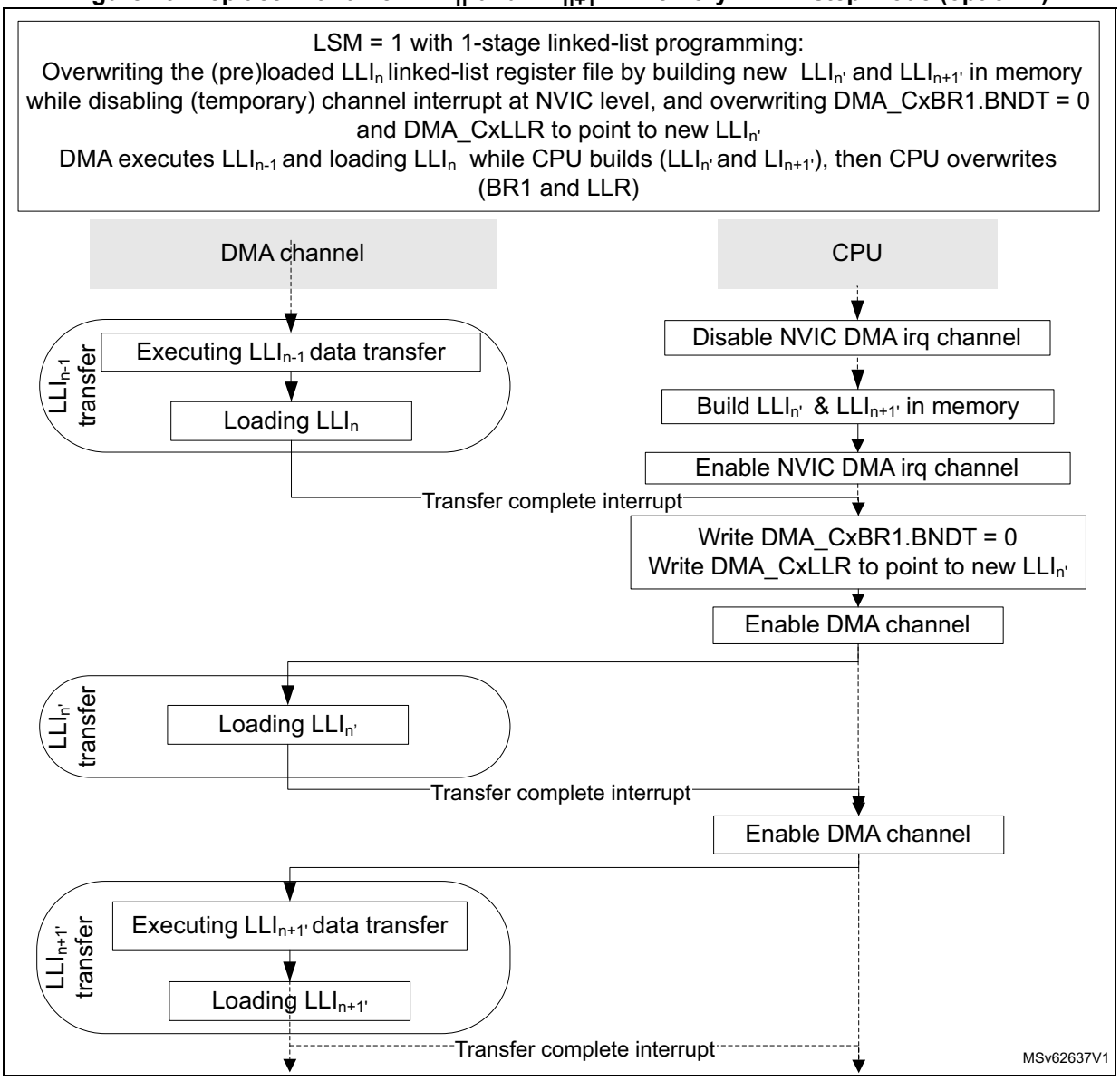
Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode

Other software implementations exist. Meanwhile LPDMA executes the transfer from the LLI_{n-1} and loads a formerly elaborated LLI_n from the memory (or even earlier), the software can do the following:

1. Disable the NVIC for not being interrupted by the interrupt handling.
2. Build a new $LLI_{n'}$ and a new $LLI_{n+1'}$.
3. Enable again the NVIC for the channel interrupt (transfer complete) notification.

The software in the interrupt handler for LLI_{n-1} is then restricted to overwrite LPDMA_CxBR1.BNDT[15:0] to be null and LPDMA_CxLLR to point to new $LLI_{n'}$, as shown in *Figure 79*.

Figure 79. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 2)



18.4.9 LPDMA channel state and linked-list programming

The software can reconfigure a channel when the channel is disabled (LPDMA_CxCR.EN = 0) and update the execution mode (LPDMA_CxCR.LSM) to change from/to run-to-completion mode to/from link step mode.

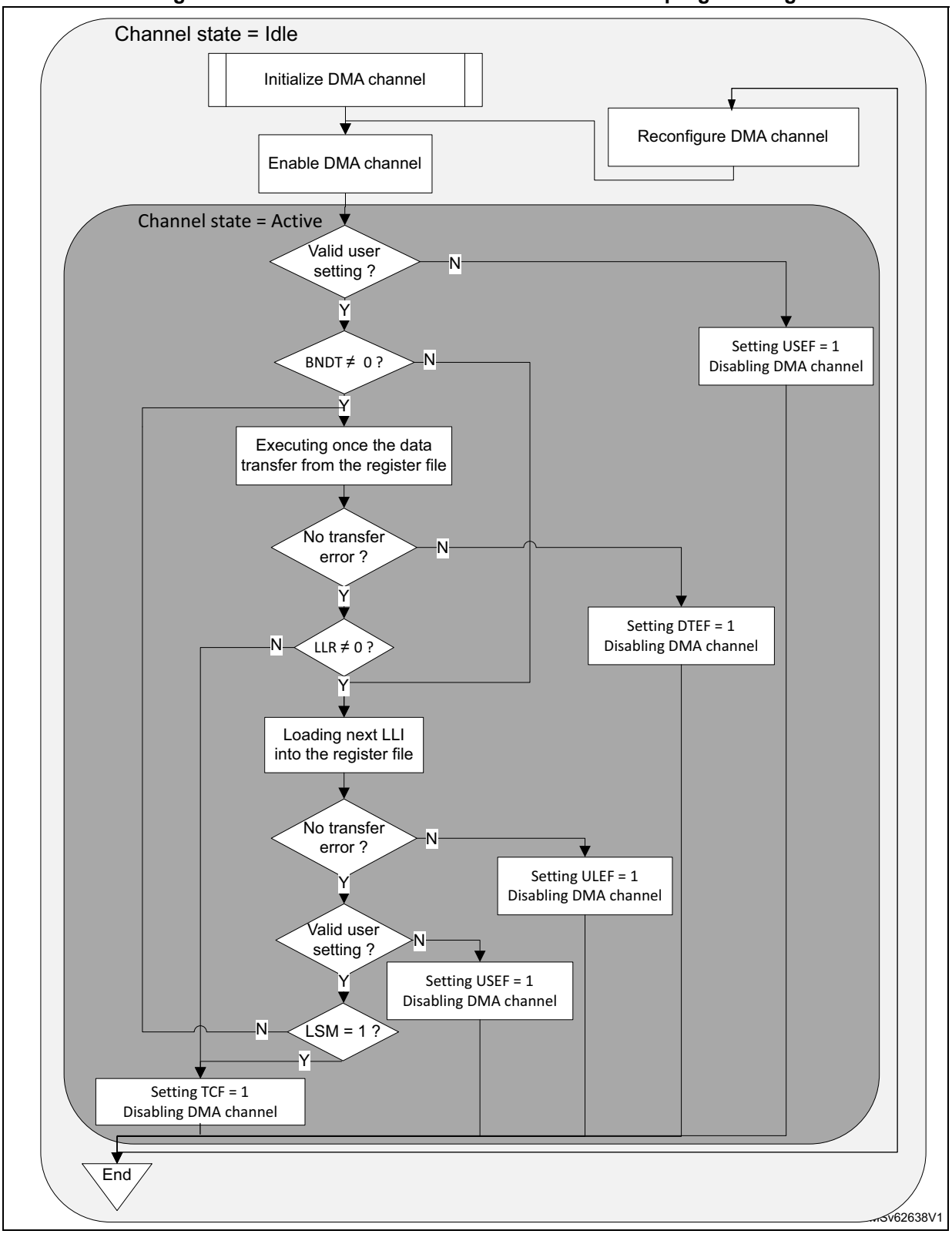
In any execution mode, the software can:

- reprogram LLI_{n+1} in the memory to finally complete the channel by this LLI_{n+1} (clear the LPDMA_CxLLR of this LLI_{n+1}), before that this LLI_{n+1} is loaded/used by the LPDMA channel
- abort and reconfigure the channel with a LSM update (see *Section 18.4.4*)

In link step mode, the software can clear LSM after each a single execution of any LLI, during LLI_{n-1}.

Figure 80 shows the overall and unified LPDMA linked-list programming, whatever is the execution mode.

Figure 80. LPDMA channel execution and linked-list programming



18.4.10 LPDMA direct transfers

There is a single transfer operation mode called the direct mode. Any LPDMA channel is used in direct mode. Any channel is implemented without any FIFO (for every channel x, dma_fifo_size[x] = 0).

LPDMA single

A programmed transfer at the lowest level is called a LPDMA single.

A LPDMA single data width is 1, 2 or 4 bytes, as defined by the SDW_LOG2[1:0] and DDW_LOG2[1:0] fields of LPDMA_CxTR1(respectively for source and destination).

Note: The user must not assign a 8-byte data width (SDW_LOG2[1:0] = 0b11 or DDW_LOG2[1:0] = 0b11) else a user setting is reported and no transfer is issued.

The addressing mode after each data of a LPDMA single is defined by the SINC and DINC bits of LPDMA_CxTR1(respectively for source and destination): either a fixed addressing or an incremented addressing with contiguous data.

The start and next addresses of a LPDMA source/destination single (defined by LPDMA_CxSAR and LPDMA_CxDAR) must be aligned with the respective data width.

The table below lists the main characteristics of a LPDMA single.

The next single address in the table is the next source/destination address, pointed by LPDMA_CxSAR and LPDMA_CxDAR, once the programmed source/destination single is completed.

Table 137. Programmed LPDMA source/destination single

Programmed LPDMA source/destination single	SDW_LOG2[1:0] DDW_LOG2[1:0]	Data width (bytes)	SINC/ DINC	Addressing mode	SAR/DAR next single address	Address alignment
Fixed byte single	00	1	0	Fixed	+0	1
Fixed half-word single	01	2				2
Fixed word single	10	4				4
Incremented byte single	00	1	1	Incremented	+1	1
Incremented half-word single	01	2			+2	2
Incremented word single	10	4			+4	4
Forbidden	11	Causes USEF generation and none single to be issued.				

In direct mode, a LPDMA single is an AHB single transfer.

LPDMA data handling: byte-based padding/truncation, sign extension and left/right alignment

The user can configure some data manipulation between a transferred data from the source and its transfer to the destination. Data handling is controlled by the LPDMA_CxTR1 register:

- If destination data width = source data width (DDW_LOG2[1:0] = SDW_LOG2[1:0]), the source data is copied as is and transferred to the destination.
- Else, depending on PAM:
  - If destination data width > source data width, the source data can be either right-aligned and padded with 0 s, or sign extended up to the destination data width.
  - If destination data width < source data width, the source data can be either right-aligned and left-truncated down to the destination data width, or left-aligned and right-truncated down to the destination data width.

There is no data manipulation between two distinct transferred data from the source, before the generation of the destination transfer.

The table below lists possible data handling from the source to the destination.

Table 138. Programmed data handling

SDW_ LOG2[1:0]	Source data	Source data stream(1)	DDW_ LOG2[1:0]	Destination data	PAM[0]	Destination data stream(1)
00	Byte	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$	00	Byte	x	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
			01	Half-word	0 (RA, 0P)(2)(3)	$0B_3,0B_2,0B_1,0B_0$
					1 (RA, SE)(2)(4)	SB_3,SB_2,SB_1,SB_0
			10	Word	0 (RA, 0P)	$000B_1,000B_0$
					1 (RA, SE)	$SSSB_1,SSSB_0$
01	Half-word	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$	00	Byte	0 (RA, LT)(2)	B_6,B_4,B_2,B_0
					1 (LA, RT)(2)	B_7,B_5,B_3,B_1
			01	Half-word	xx	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
			10	Word	0 (RA, 0P)	$00B_3B_2,00B_1B_0$
					1 (RA, SE)	SSB_3B_2,SSB_1B_0
10	Word	$B_7B_6B_5B_4,B_3B_2B_1B_0$	00	Byte	0 (RA, LT)	B_{12},B_8,B_4,B_0
					1 (LA, RT)	B_{15},B_{11},B_7,B_3
			01	Half-word	0 (RA, LT)	B_5B_4,B_1B_0
				Word	1 (LA, RT)	B_7B_6,B_3B_2
			10		xx	$B_7B_6B_5B_4,B_3B_2B_1B_0$

1. Data stream is timely ordered starting from the byte with the lowest index a.k.a B_0.
2. RA = right aligned. LA = left aligned. RT= right truncated. LT = left truncated.
3. 0P = zero bit padding up to the destination data width.
4. SE = sign bit extended up to the destination data width.

18.4.11 LPDMA transfer request and arbitration

LPDMA transfer request

As defined by LPDMA_CxTR2, a programmed LPDMA data transfer is requested with one of the following:

- a software request if the control bit SWREQ = 1: This is used typically by the CPU for a data transfer from a memory-mapped address to another memory mapped address (memory-to-memory, GPIO to/from memory)
- an input hardware request coming from a peripheral if SWREQ = 0: The selection of the LPDMA hardware peripheral request is driven by the REQSEL[4:0] field (see *Section 18.3.4*). The selected hardware request can be one of the following:
  - a hardware request from a peripheral configured in LPDMA mode (for a transfer from/to the peripheral data register respectively to/from the memory)
  - a hardware request from a peripheral for its control registers update from the memory
  - a hardware request from a peripheral for a read of its status registers transferred to the memory

Caution: The user must not assign a same input hardware peripheral LPDMA request via LPDMA_CxTR.REQSEL[4:0] to two different channel if at a given time this request is asserted by the peripheral and each channel is ready to execute this requested data transfer. There is no user setting error reporting.

LPDMA transfer request for arbitration

For a given channel, a LPDMA requested data transfer from the source address to the destination address is arbitrated versus simultaneous requested LPDMA transfers from other channels, in order to be scheduled over the AHB master port. A LPDMA data transfer is atomic to the LPDMA arbitration: it consists of an AHB read single, immediately followed by an AHB write single. It is granted by the arbiter once for both AHB transfers, based on the channel priority defined by LPDMA_CxCR.PRIO[1:0].

An arbitrated LPDMA requested link transfer consists of one 32-bit read from the linked-list data structure in the memory to one of the linked-list registers (LPDMA_CxTR1, LPDMA_CxTR2, LPDMA_CxBR1, LPDMA_CxSAR, LPDMA_CxDAR or LPDMA_CxLLR). Each 32-bit read from the memory is arbitrated with the same channel priority as for data transfers, in order to be scheduled over the master port.

The re arbitration occurs after each granted single transfer:

- whatever how the requested data transfer is programmed:
  - with a software request for a memory-to-memory transfer (LPDMA_CxTR2.SWREQ = 1)
  - with a hardware request (LPDMA_CxTR2.SWREQ = 0) for a memory-to-peripheral transfer or a peripheral-to-memory transfer
- whatever the hardware request type

When the requested data transfer is programmed with a hardware request from a peripheral (LPDMA_CxTR2.SWREQ = 0), the first memory read of a block is gated by the occurrence of the corresponding and selected hardware request, whatever the peripheral is source or destination of the transfer. This first read request to the memory is not taken into account earlier by the arbiter (not as soon as the block transfer is enabled and executable).

LPDMA arbitration

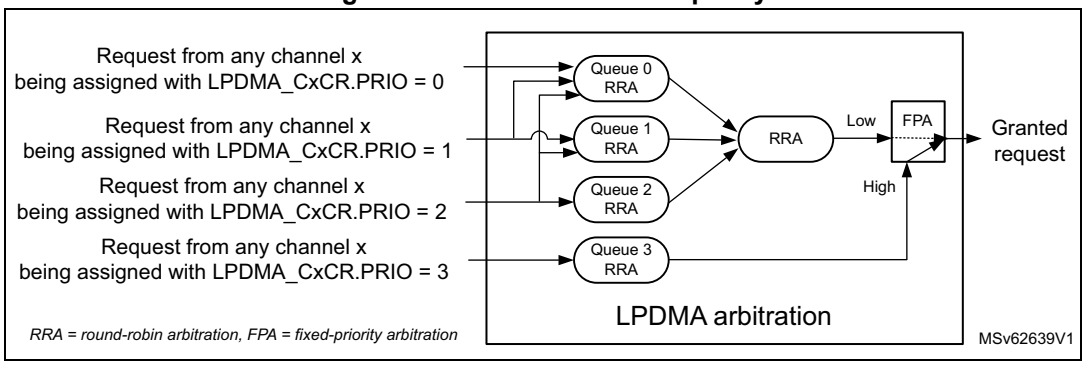
The LPDMA arbitration is directed from the 4-grade assigned channel priority (LPDMA_CxCR.PRIO[1:0]). The arbitration policy, as illustrated in *Figure 81*, is defined by:

- one high-priority traffic class (queue 3), dedicated to the assigned channels with priority 3, for time-sensitive channels

  This traffic class is granted via a fixed-priority arbitration against any other low-priority traffic class. Within this class, requested single transfers are round-robin arbitrated.
- three low-priority traffic classes (queues 0, 1 or 2) for non time-sensitive channels with priority 0, 1 or 2

  Each requested single transfer within this class is round-robin arbitrated, with a weight that is monotonically driven from the programmed priority:
  - Requests with priority 0 are allocated to the queue 0.
  - Requests with priority 1 are allocated and replicated to the queue 0 and queue 1.
  - Requests with priority 2 are allocated and replicated to the queue 0, queue 1, and queue 2.
  - Any queue 0, 1 or 2 equally grants any of its active input requests in a round-robin manner, provided there are simultaneous requests.
  - Additionally, there is a second stage for the low-traffic with a round-robin arbiter that fairly alternates between simultaneous selected requests from queue 0, queue 1 and queue 2.

Figure 81. LPDMA arbitration policy



1. RRA: round-robin arbitration
2. FPA: fixed-priority arbitration

LPDMA arbitration and bandwidth

With this arbitration policy, the following is guaranteed:

- Equal maximum bandwidth between requests with same priority
- Reserved bandwidth (noted as B_{Q3}) to the time-sensitive requests (with priority 3)
- Residual weighted bandwidth between different low-priority requests (priority 0 versus priority 1 versus priority 2).

The two following examples highlight that the weighted round-robin arbitration is driven by the programmed priorities:

- **Example 1**: basic application with two non time-sensitive LPDMA requests: req0 and req1. There are the following programming possibilities:
  - If they are assigned with same priority, the allocated bandwidth by the arbiter to req0 (B_{req0}) is **equal** to the allocated bandwidth to req1(B_{req1}).

    $B_{req0} = B_{req1} = 1/2 * (1 - B_{Q3})$
  - If req0 is assigned to priority 0 and req1 to priority 1, the allocated bandwidth to req0 (B_{P0}) is **3 times less** than the allocated bandwidth to req1 (B_{P1}).

    $B_{req0} = B_{P0} = 1/2 * 1/2 * (1 - B_{Q3}) = 1/4 * (1 - B_{Q3})$

    $B_{req1} = B_{P1} = (1/2 + 1) * 1/3 * (1 - B_{Q3}) = 3/4 * (1 - B_{Q3})$
  - If req0 is assigned to priority 0 and req1 to priority 2, the allocated bandwidth to req0 (B_{P0}) is **5 times less** than the allocated bandwidth to req1 (B_{P2}).

    $B_{req0} = B_{P0} = 1/2 * 1/3 * (1 - B_{Q3}) = 1/6 * (1 - B_{Q3})$

    $B_{req1} = B_{P2} = (1/2 + 1 + 1) * 1/3 * (1 - B_{Q3}) = 5/6 * (1 - B_{Q3})$

  The above computed bandwidth calculation is based on a theoretical input request, always active for any LPDMA clock cycle. This computed bandwidth from the arbiter must be weighted by the frequency of the request given by the application, that cannot be always active and may be quite much variable from one LPDMA client (example I2C at 400 kHz) to another one (PWM at 1 kHz) than the above x3 and x5 ratios.
- **Example 2**: application where the user distributes a same non-null N number of LPDMA requests to every non time-sensitive priority 0, 1 and 2. The bandwidth calculation is then the following:
  - The allocated bandwidth to the set of requests of priority 0 (B_{P0}) is

    $B_{P0} = 1/3 * 1/3 * (1 - B_{Q3}) = 1/9 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 1(B_{P1}) is

    $B_{P1} = (1/3 + 1/2) * 1/3 * (1 - B_{Q3}) = 5/18 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 2(B_{P2}) is

    $B_{P2} = (1/3 + 1/2 + 1) * 1/3 * (1 - B_{Q3}) = 11/18 * (1 - B_{Q3})$
  - The allocated bandwidth to any request n ($B_{n)}$ among the N requests of that priority Pi (i = 0 to 2) is $B_n = 1/N * B_{Pi}$
  - The allocated bandwidth to any request n of priority 0i ($B_{n, Pi}$) is

    $B_{n, P0} = 1/N * 1/9 * (1 - B_{Q3})$

    $B_{n, P1} = 1/N * 5/18 * (1 - B_{Q3})$

    $B_{n, P2} = 1/N * 11/18 * (1 - B_{Q3})$

  In this example, when the master port bus bandwidth is not totally consumed by the time-sensitive queue 3, the residual bandwidth is such that 2.5 times less bandwidth is allocated to any request of priority 0 versus priority 1, and 5.5 times less bandwidth is allocated to any request of priority 0 versus priority 2.

More generally, assume that the following requests are present:

- I requests (I ≥ 0) assigned to priority 0
  If I > 0, these requests are noted from i = 0 to I-1.
- J requests (J ≥ 0) assigned to priority 1
  If J > 0, these requests are noted from j = 0 to J-1.
- K requests (K > 0) assigned to priority 2
  These requests are noted from k = 0 to K-1
- L requests (L ≥ 0) assigned to priority 3
  If L > 0, these requests are noted from l = 0 to L-1.

As B_{Q3} is the reserved bandwidth to time-sensitive requests, the bandwidth for each request L with priority 3 is:

- $B_l = B_{Q3} / L$ for L > 0 (else: $B_l = 0$)

The bandwidth for each non-time sensitive queue is:

- $B_{Q0} = 1/3 * (1 - B_{Q3})$
- $B_{Q1} = 1/3 * (1 - B_{Q3})$
- $B_{Q2} = 1/3 * (1 - B_{Q3})$

The bandwidth for the set of requests with priority 0 is:

- $B_{P0} = I / (I + J + K) * B_{Q0}$

The bandwidth for each request i with priority 0 is:

- $B_i = B_{P0} / I$ for L > 0 (else $B_{P0} = 0$)

The bandwidth for the set of requests with priority 1 and routed to queue 0 is:

- $B_{P1,Q0} = J / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 1 and routed to queue 1 is:

- $B_{P1,Q1} = J / (J + K) * B_{Q1}$

The total bandwidth for the set of requests with priority 1 is:

- $B_{P1} = B_{P1,Q0} + B_{P1,Q1}$

The bandwidth for each request j with priority 1 is:

- $B_j = B_{P1} / J$ for J > 0 (else $B_j = 0$)

The bandwidth for the set of requests with priority 2 and routed to queue 0 is:

- $B_{P2,Q0} = K / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 2 and routed to queue 1 is:

- $B_{P2,Q1} = K / (J + K) * B_{Q1}$

The bandwidth for the set of requests with priority 2 and routed to queue 2 is:

- $B_{P2,Q2} = B_{Q2}$

The total bandwidth for the set of requests with priority 2 is:

- $B_{P2} = B_{P2,Q0} + B_{P2,Q1} + B_{P2,Q2}$

The bandwidth for each request k with priority 2 is:

- $B_k = B_{P2} / K$ (K>0 in the general case)

Thus finally the maximum allocated residual bandwidths for any i, j, k non-time sensitive request are:

- in the general case (when there is at least one request k with a priority 2 (K > 0)):
  - $B_i = 1/I * 1/3 * I/(I + J + K) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/3 *[J/(I + J + K) + J/(J + K)] * (1 - B_{Q3})$
  - $B_k = 1/K * 1/3 *[K/(I + J + K) + K/(J + K) + 1] * (1 - B_{Q3})$
- in the specific case (when there is no request k with a priority 2 (K = 0)):
  - $B_i = 1/I * 1/2 * I/(I + J) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/2 *[J/(I + J) + 1] * (1 - B_{Q3})$

Consequently, the LPDMA arbiter can be used as a programmable weighted bandwidth limiter, for each queue and more generally for each request/channel. The different weights are monotonically resulting from the programmed channel priorities.

18.4.12 LPDMA triggered transfer

A programmed LPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event, as defined by LPDMA_CxTR2.TRIGPOL[1:0] and LPDMA_CxTR2.TRIGSEL[4:0] (see *Section 18.3.5* for the trigger selection).

The triggered transfer, as defined by the trigger mode in LPDMA_CxTR2.TRIGM[1:0], can be at LLI data transfer level, to condition the first single read of a block, or each programmed single read. The trigger mode can also be programmed to condition the LLI link transfer (see the TRIGM[1:0] description in *LPDMA channel x transfer register 2 (LPDMA_CxTR2)* for more details).

Trigger hit memorization and trigger overrun flag generation

The LPDMA monitoring of a trigger for a channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (respectively TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

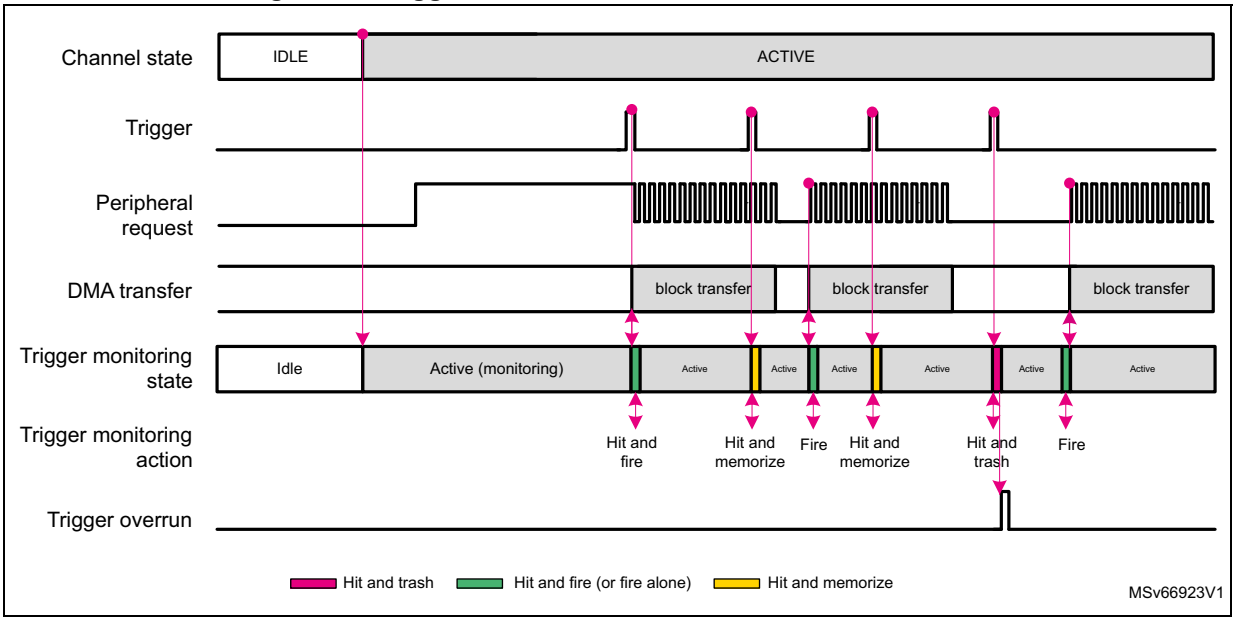
The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer. If a new trigger is detected, this hit is internally memorized to grant the next transfer, as long as the defined rising/falling edge and TRIGSEL[4:0] are not modified, and the channel is enabled.

Transferring a next LLI_{n+1}, that updates the LPDMA_CxTR2 with a new value for any of TRIGSEL[4:0] or TRIGPOL[1:0], resets the monitoring, trashing the possible memorized hit of the formerly defined LLI_n trigger.

Caution: After a first new trigger hit_{n+1} is memorized, if another trigger hit_{n+2} is detected and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized. A trigger overrun flag is reported (LPDMA_CxSR.TOF = 1) and an interrupt is generated if enabled (if LPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

The figure below illustrates the trigger hit, memorization and overrun in the configuration example with a block-level trigger mode and a rising edge trigger polarity.

Figure 82. Trigger hit, memorization and overrun waveform



Note: The user can assign the same input trigger event to different channels. This can be used to trigger different channels on a broadcast trigger event.

18.4.13 LPDMA circular buffering with linked-list programming

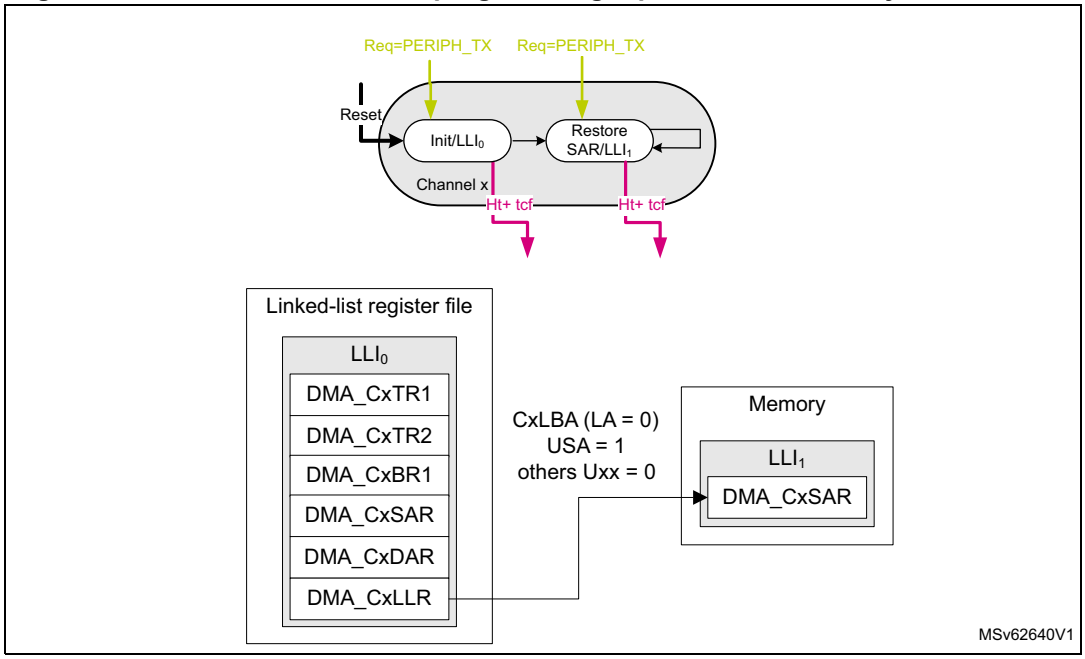
LPDMA circular buffering for memory-to-peripheral and peripheral-to-memory transfers

For a circular buffering, with a continuous memory-to-peripheral (or peripheral-to-memory) transfer, the software must set up a channel with half transfer and complete transfer events/interrupts generation (LPDMA_CxCR.HTIE = 1 and LPDMA_CxCR.TCIE = 1), in order to enable a concurrent buffer software processing.

LLI_0 is configured for the first block transfer. A continuously-executed LLI_1 is needed to restore the memory source (or destination) start address, for the memory-to-peripheral transfer (respectively the peripheral-to-memory transfer). LPDMA automatically reloads the initially programmed LPDMA_CxBR1.BNDT[15:0] when a block transfer is completed, and there is no need to restore LPDMA_CxBR1.

The figure below illustrates this programming with a LPDMA channel and a source circular buffer.

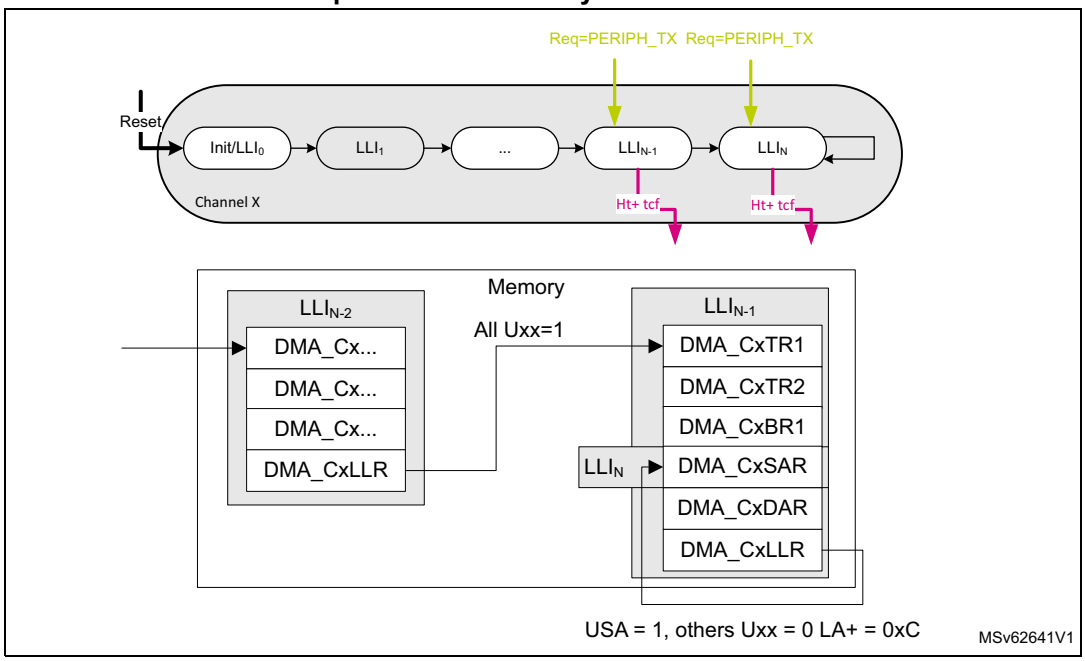
Figure 83. LPDMA circular buffer programming: update of the memory start address



If circular buffering must be executed after some other transfers over the shared LPDMA channel x, the before-last LLI_{N-1} in memory is needed to configure the first block transfer. And the last LLI_N restores the memory source (or destination) start address in memory-to-peripheral transfer (respectively in peripheral-to-memory transfer).

The figure below illustrates this programming with a shared LPDMA channel, and a source circular buffer.

Figure 84. Shared LPDMA channel with circular buffering: update of the memory start address



18.4.14 LPDMA secure/non-secure channel

The LPDMA controller is compliant with the TrustZone hardware architecture at channel level, partitioning all its resources so that they exist in one of the secure and non-secure worlds at any given time.

Any channel x is a secure or a non-secure hardware resource, as configured by LPDMA_SECCFGR.SECx.

When a channel x is configured in secure state by a secure agent, the following access control rules are applied:

- A non-secure read access to a register field of this channel is forced to return 0, except for LPDMA_SECCFGR, LPDMA_PRIVCFGR, LPDMA_RCFGLOCKR that are readable by a non-secure agent.
- A non-secure write access to a register field of this channel has no impact.

When a channel x is configured in secure state, a secure agent can configure separately as secure or non-secure the LPDMA data transfer from the source (LPDMA_CxTR1.SSEC) and the LPDMA data transfer to the destination (LPDMA_CxTR1.DSEC).

When a channel x is configured in secure state and in linked-list mode, the loading of the next linked-list data structure from the LPDMA memory into its register file, is automatically performed with secure transfers via the master port.

LPDMA generates the DMA channel state versus security, reflecting LPDMA_SECCFGR, to keep the other peripherals informed of the secure/non-secure state of each LPDMA channel x.

LPDMA also generates a security illegal access pulse signal on an illegal non-secure access to a secure LPDMA register. This signal is routed to the TrustZone interrupt controller.

When the secure software must switch a channel from a secure state to a non-secure state, the secure software must abort the channel or wait until the secure channel is completed before switching. This is needed to dynamically re-allocate a channel to a next non-secure transfer as a non-secure software is not allowed to do so and must have LPDMA_CxCR.EN = 0 before the non-secure software can reprogram the LPDMA_CxCR for a next transfer. The secure software may reset not only the channel x (LPDMA_CxCR.RESET = 1) but also the full channel x register file to its reset value.

18.4.15 LPDMA privileged/unprivileged channel

Any channel x is a privileged or unprivileged hardware resource, as configured by a privileged agent via LPDMA_PRIVCFGR.PRIVx.

When a channel x is configured in a privileged state by a privileged agent, the following access control rules are applied:

- An unprivileged read access to a register field of this channel is forced to return 0, except for LPDMA_PRIVCFGR, LPDMA_SECCFGR, that are readable by an unprivileged agent.
- An unprivileged write access to a register field of this channel has no impact.

When a channel is configured in a privileged (or unprivileged) state, the source and destination data transfers are privileged (respectively unprivileged) transfers over the AHB master port.

When a channel is configured in a privileged (or unprivileged) state and in linked-list mode, the loading of the next linked-list data structure from the LPDMA memory into its register file, is automatically performed with privileged (respectively unprivileged) transfers, via the master port.

LPDMA generates a DMA channel state versus privilege, reflecting LPDMA_PRIVCFGR, to keep the other peripherals informed of the privileged/unprivileged state of each DMA channel x.

Additionally, the LPDMA generates the privileged illegal access pulse signal on an illegal unprivileged access to a privileged LPDMA register. This signal may be used or not, depending on the product (see the system security section for more details).

When the privileged software must switch a channel from a privileged state to an unprivileged state, the privileged software must abort the channel or wait until that the privileged channel is completed before switching. This is needed to dynamically re-allocate a channel to a next unprivileged transfer as an unprivileged software is not allowed to do so, and must have LPDMA_CxCR.EN = 0 before the unprivileged software can reprogram the LPDMA_CxCR for a next transfer. The privileged software may reset not only the channel x (LPDMA_CxCR.RESET = 1) but also the full channel x register file to its reset value.

18.4.16 LPDMA error management

LPDMA is able to manage and report to the user a transfer error, as follows, depending on the root cause.

Data transfer error

On a bus access (as a AHB single) to the source or the destination:

- The source or destination target reports an AHB error.
- The programmed channel transfer is stopped (LPDMA_CxCR.EN cleared by the LPDMA hardware). The channel status register reports an idle state (LPDMA_CxSR.IDLEF = 1) and the data error (LPDMA_CxSR.DTEF = 1).
- After a LPDMA data transfer error, the user must perform a debug session, taking care of the product-defined memory mapping of the source and destination, including the protection attributes.
- After a LPDMA data transfer error, the user must issue a channel reset (set LPDMA_CxCR.RESET) to reset the hardware LPDMA channel data path before the user enables again the same channel for a next transfer.

Link transfer error

On a tentative update of a LPDMA channel register from the programmed LLI in the memory:

- The linked-list memory reports an AHB error.
- The programmed channel transfer is stopped (LPDMA_CxCR.EN cleared by the LPDMA hardware), the channel status register reports an idle state (LPDMA_CxSR.IDLEF = 1) and the link error (LPDMA_CxSR.ULEF = 1).
- After a LPDMA link error, the user must perform a debug session, taking care of the product-defined memory mapping of the linked-list data structure (LPDMA_CxLBAR and LPDMA_CxLLR), including the protection attributes.
- After a LPDMA link error, the user must explicitly write the linked-list register file (LPDMA_CxTR1, LPDMA_CxTR2, LPDMA_CxBR1, LPDMA_CxSAR, LPDMA_CxDAR and LPDMA_CxLLR), before the user enables again the same channel for a next transfer.

User setting error

On a tentative execution of a LPDMA transfer with an unauthorized user setting:

- The programmed channel transfer is disabled (LPDMA_CxCR.EN forced and cleared by the LPDMA hardware) preventing the next unauthorized programmed data transfer from being executed. The channel status register reports an idle state (LPDMA_CxSR.IDLEF = 1) and a user setting error (LPDMA_CxSR.USEF = 1).
- After a LPDMA user setting error, the user must perform a debug session, taking care of the LPDMA channel programming. A user setting error can be caused by one of the following:
  - a programmed null source block size without a programmed update of this value from the next LLI_1 (LPDMA_CxBR1.BNDT[15:0] = 0 and LPDMA_CxLLR.UB1 = 0)
  - a programmed non-null source block size being not a multiple of the programmed data width of a source single transfer (LPDMA_CxBR1.BNDT[2:0] versus LPDMA_CxTR1.SDW_LOG2[1:0])

- a programmed unaligned source start address, being not a multiple of the programmed data width of a source single transfer (LPDMA_CxSAR[2:0] versus LPDMA_CxTR1.SDW_LOG2[1:0])
- a programmed unaligned destination start address, being not a multiple of the programmed data width of a destination single transfer (LPDMA_CxDAR[2:0] versus LPDMA_CxTR1.DDW_LOG2[1:0])
- a programmed double-word source data width (LPDMA_CxTR1.SDW_LOG2[1:0] = 0b11)
- a programmed double-word destination data width (LPDMA_CxTR1.DDW_LOG2[1:0] = 0b11
- a programmed linked-list item LLI_{n+1} with a null data transfer (LPDMA_CxLLR.UB1 = 1 and LPDMA_CxBR1. BNDT = 0)

18.4.17 LPDMA autonomous mode

To save dynamic power consumption while LPDMA executes the programmed linked-list transfers, LPDMA hardware automatically manages its own clock gating and generates a clock request output signal to the RCC, whenever the device is in Run, Sleep or Stop mode, provided that the RCC is programmed with the corresponding LPDMA enable control bits.

For more details about the RCC programming, refer to the RCC section of the reference manual.

For mode details about the availability of the LPDMA autonomous feature vs the device low-power modes, refer to *Section 18.3.2*.

The user can program and schedule the execution of a given LPDMA transfer at a LLI_n level of a LPDMA channel x, with LPDMA_CxTR2 as follows:

- The software controls and conditions the input of a transfer with TRIGM[1:0], TRIGPOL[1:0], TRIGSEL[4:0], SWREQ and REQSEL[4:0] for the input trigger and request.
- The software controls and signals the output of a transfer with TCEM[1:0] for generating or not a transfer complete event, and generating or not an associated half data transfer event).

See *LPDMA channel x transfer register 2 (LPDMA_CxTR2)* for more details.

The output channel x transfer complete event, lpdma_chx_tc, can be programmed as a selected input trigger for a channel if this event is looped-back and connected at the LPDMA level (see *Section 18.3.5*), allowing autonomous and fine DMA inter-channel transfer scheduling, without needing a cleared transfer complete flag (TCF).

A given LPDMA channel x asserts its clock request in one of the following conditions:

- if the next transfer to be executed is programmed as conditioned by a trigger (LPDMA_CxTR2.TRIGPOL[1:0] and LPDMA_CxTR2.TRIGM[1:0]), only when the trigger hit occurs.
- if the next transfer to be executed is not conditioned by a trigger:
  - if LPDMA_CxTR2.SWREQ = 0, only when the hardware request is asserted by the selected peripheral
  - if LPDMA_CxTR2.SWREQ = 1 (memory-to-memory, GPIO to/from memory), as soon as the GPDMA is enabled

The LPDMA channel x releases its clock request as soon as all the following conditions are met:

- The transfer to be executed is completed.
- The LPDMA channel x is not immediately ready and requested to execute the next transfer.
- If a channel x interrupt was raised, all the flags of the status register that can cause this interrupt, are cleared by a software agent.

When one channel asserts its clock request, the LPDMA asserts its clock request to the RCC. When none channel asserts its clock request, the LPDMA releases its clock request to the RCC.

18.5 LPDMA in debug mode

When the microcontroller enters debug mode (core halted), any channel x can be individually either continued (default) or suspended, depending on the programmable control bit in the DBGMCU module.

Note: In debug mode, LPDMA_CxSR.SUSPF is not altered by a suspension from the programmable control bit in the DBGMCU module. In this case, LPDMA_CxSR.IDLEF can be checked to know the completion status of the channel suspension.

18.6 LPDMA in low-power modes

Table 139. Effect of low-power modes on LPDMA

Mode	Description
Sleep	No effect. LPDMA interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the LPDMA registers is kept when entering Stop mode. The content of the LPDMA registers can be autonomously updated by a next linked-list item from memory, to perform autonomous data transfers. LPDMA interrupts can cause the device to exit Stop mode.
Standby	The LPDMA is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 18.3.2* to know which Stop mode is supported.

18.7 LPDMA interrupts

There is one LPDMA interrupt line for each channel, and separately for each CPU (if several ones in the devices).

Table 140. LPDMA interrupt requests

Interrupt acronym	Interrupt event	Interrupt enable	Event flag	Event clear method
LPDMA_CHx	Transfer complete	LPDMA_CxCR.TCIE	LPDMA_CxSR.TCF	Write 1 to LPDMA_CxFCR.TCF
	Half transfer	LPDMA_CxCR.HTIE	LPDMA_CxSR.HTF	Write 1 to LPDMA_CxFCR.HTF
	Data transfer error	LPDMA_CxCR.DTEIE	LPDMA_CxSR.DTEF	Write 1 to LPDMA_CxFCR.DTEF
	Update link error	LPDMA_CxCR.ULEIE	LPDMA_CxSR.ULEF	Write 1 to LPDMA_CxFCR.ULEF
	User setting error	LPDMA_CxCR.USEIE	LPDMA_CxSR.USEF	Write 1 to LPDMA_CxFCR.USEF
	Suspended	LPDMA_CxCR.SUSPIE	LPDMA_CxSR.SUSPF	Write 1 to LPDMA_CxFCR.SUSPF
	Trigger overrun	LPDMA_CxCR.TOFIE	LPDMA_CxSR.TOF	Write 1 to LPDMA_CxFCR.TOF

A LPDMA channel x event may be:

- a transfer complete
- a half-transfer complete
- a transfer error, due to either:
  - a data transfer error
  - an update link error
  - a user setting error completed suspension
- a trigger overrun

Note: When a channel x transfer complete event occurs, the output signal lpdma_chx_tc is generated as a high pulse of one clock cycle.

An interrupt is generated following any xx event, provided that both:

- the corresponding interrupt event xx is enabled (LPDMA_CxCR.xxIE = 1)
- the corresponding event flag is cleared (LPDMA_CxSR.xxF = 0). This means that, after a previous same xx event occurrence, a software agent must have written 1 into the corresponding xx flag clear control bit (write 1 into LPDMA_CxFCR.xxF).

TCF (transfer complete) and HTF (half transfer) events generation is controlled by LPDMA_CxTR2.TCEM[1:0] as follows:

- A transfer complete event is a block transfer complete or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode LPDMA_CxTR2.TCEM[1:0].
- A half transfer event is a half block transfer. A half-block transfer occurs when half of the source block size bytes (rounded-up integer of LPDMA_CxBR1.BNDT[15:0] / 2) is transferred to the destination.

See *LPDMA channel x transfer register 2 (LPDMA_CxTR2)* for more details.

A transfer error rises in one of the following situations:

- during a single data transfer from the source or to the destination (DTEF)
- during an update of a LPDMA channel register from the programmed LLI in memory (ULEF)
- during a tentative execution of a LPDMA channel with an unauthorized setting (USEF)

  The user must perform a debug session to correct the LPDMA channel programming versus the USEF root causes list (see *Section 18.4.16*).

A trigger overrun is described in *Trigger hit memorization and trigger overrun flag generation*.

18.8 LPDMA registers

The LPDMA registers must be accessed with an aligned 32-bit word data access.

18.8.1 LPDMA secure configuration register (LPDMA_SECCFGR)

Address offset: 0x000

Reset value: 0x0000 0000

A write access is ignored at bit level if the corresponding channel x is locked (LPDMA_RCFGLOCKR.LOCKx = 1).

A write access to this register must be secure and privileged. A read access is secure or non-secure, privileged or unprivileged.

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1.

This register must be programmed at a bit level, at the initialization/closure of a LPDMA channel (when LPDMA_CxCR.EN = 0), to securely allocate individually any channel x to the secure or non-secure world.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	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.	SEC3	SEC2	SEC1	SEC0
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **SECx**: secure state of channel x (x = 3 to 0)

0: non-secure

1: secure

18.8.2 LPDMA privileged configuration register (LPDMA_PRIVCFGR)

Address offset: 0x004

Reset value: 0x0000 0000

A write access to this register must be privileged. A read access can be privileged or unprivileged, secure or non-secure.

A write access is ignored at bit level if the corresponding channel x is locked (GPDMA_RCFGLOCKR.LOCKx = 1).

This register can mix secure and non-secure information. If a channel x is configured as secure (LPDMA_SECCFGR.SECx = 1), the PRIVx bit can be written only by a secure (and privileged) agent.

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1.

This register must be programmed at a bit level, at the initialization/closure of a LPDMA channel (LPDMA_CxCR.EN = 0), to individually allocate any channel x to the privileged or unprivileged world.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	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.	PRIV3	PRIV2	PRIV1	PRIV0
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRIVx**: privileged state of channel x (x = 3 to 0)
0: unprivileged
1: privileged

18.8.3 LPDMA configuration lock register (LPDMA_RCFGLOCKR)

Address offset: 0x008

Reset value: 0x0000 0000

This register can be written by a software agent with secure privileged attribute in order to individually lock at boot time the secure privileged attributes of any DMA channel/resource (to lock the setting of LPDMA_SECCFGR, LPDMA_PRIVCFGR for any channel x at boot time).

A read access may be privileged or unprivileged, secure or non-secure.

Note: If TZEN = 0, this register cannot be written.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	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.	LOCK3	LOCK2	LOCK1	LOCK0
												rs	rs	rs	rs

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **LOCKx**: lock the configuration of LPDMA_SECCFGR.SECx, LPDMA_PRIVCFGR.PRIVx until a global DMA reset (x = 3 to 0)

This bit is cleared after reset and, once set, it cannot be reset until a global DMA reset.

0: secure privilege configuration of the channel x is writable.

1: secure privilege configuration of the channel x is not writable.

18.8.4 LPDMA non-secure masked interrupt status register (LPDMA_MISR)

Address offset: 0x00C

Reset value: 0x0000 0000

This register is a read register.

This is a non-secure register, containing the masked interrupt status bit MISx for each non-secure channel x (channel x configured with LPDMA_SECCFGR.SECx = 0). It is a logical OR of all the flags of LPDMA_CxSR, each source flag being enabled by the corresponding interrupt enable bit of LPDMA_CxCR.

Every bit is de-asserted by hardware when writing 1 to the corresponding flag clear bit in LPDMA_CxFCR.

If a channel x is in secure state (LPDMA_SECCFGR.SECx = 1), a read access to the masked interrupt status bit MISx of this channel x returns zero.

This register may mix privileged and unprivileged information, depending on the privileged state of each channel LPDMA_PRIVCFGR.PRIVx. A privileged software can read the full non-secure interrupt status. An unprivileged software is restricted to read the status of unprivileged (and non-secure) channel(s), other privileged bit fields returning zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	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.	MIS3	MIS2	MIS1	MIS0
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **MISx**: masked interrupt status of channel x (x = 3 to 0)
0: no interrupt occurred on non-secure channel x
1: an interrupt occurred on non-secure channel x

18.8.5 LPDMA secure masked interrupt status register (LPDMA_SMISR)

Address offset: 0x010

Reset value: 0x0000 0000

This is a secure read register, containing the masked interrupt status bit MISx for each secure channel x (LPDMA_SECCFGR.SECx = 1). It is a logical OR of all the LPDMA_CxSR flags, each source flag being enabled by the corresponding LPDMA_CxCR interrupt enable bit.

Every bit is de-asserted by hardware when securely writing 1 to the corresponding LPDMA_CxFCR flag clear bit.

This register does not contain any information about a non-secure channel.

This register can mix privileged and unprivileged information, depending on the privileged state of each channel LPDMA_PRIVCFGR.PRIVx. A privileged software can read the full secure interrupt status. An unprivileged software is restricted to read the status of unprivileged and secure channels, other privileged bit fields returning zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	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.	MIS3	MIS2	MIS1	MIS0
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **MISx**: masked interrupt status of the secure channel x (x = 3 to 0)
0: no interrupt occurred on the secure channel x
1: an interrupt occurred on the secure channel x

18.8.6 LPDMA channel x linked-list base address register (LPDMA_CxLBAR)

Address offset: 0x050 + 0x80 * x (x = 0 to 3)

Reset value: 0x0000 0000

This register must be written by a privileged software. It is either privileged readable or not, depending on the privileged state of the channel x LPDMA_PRIVCFGR.PRIVx.

This register is either secure or non-secure depending on the secure state of the channel x (LPDMA_SECCFGR.SECx).

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1.

This channel-based register is the linked-list base address of the memory region, for a given channel x, from which the LLIs describing the programmed sequence of the LPDMA transfers, are conditionally and automatically updated.

This 64-Kbyte aligned channel x linked-list base address is offset by the 16-bit LPDMA_CxLLR register that defines the word-aligned address offset for each LLI.

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

Bits 31:16 **LBA[31:16]**: linked-list base address of LPDMA channel x

Bits 15:0 Reserved, must be kept at reset value.

18.8.7 LPDMA channel x flag clear register (LPDMA_CxFCR)

Address offset: 0x05C+ 0x80 * x (x = 0 to 3)

Reset value: 0x0000 0000

This is a write register, secure or non-secure depending on the secure state of channel x (LPDMA_SECCFGR.SECx) and privileged or non-privileged, depending on the privileged state of the channel x (LPDMA_PRIVCFGR.PRIVx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	w	w	w	w	w	w	w								

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

Bit 14 **TOF**: trigger overrun flag clear
0: no effect
1: clears the corresponding TOF flag

Bit 13 **SUSPF**: completed suspension flag clear
0: no effect
1: corresponding SUSPF flag cleared

Bit 12 **USEF**: user setting error flag clear
0: no effect
1: corresponding USEF flag cleared

Bit 11 **ULEF**: update link transfer error flag clear
0: no effect
1: corresponding ULEF flag cleared

Bit 10 **DTEF**: data transfer error flag clear
0: no effect
1: corresponding DTEF flag cleared

Bit 9 **HTF**: half transfer flag clear
0: no effect
1: corresponding HTF flag cleared

Bit 8 **TCF**: transfer complete flag clear
0: no effect
1: corresponding TCF flag cleared

Bits 7:0 Reserved, must be kept at reset value.

18.8.8 LPDMA channel x status register (LPDMA_CxSR)

Address offset: 0x060 + 0x80 * x (x = 0 to 3)

Reset value: 0x0000 0001

This is a read register, reporting the channel status.

This register is secure or non-secure, depending on the secure state of channel x (LPDMA_SECCFGR.SECx), and privileged or non-privileged, depending on the privileged state of the channel (LPDMA_PRIVCFGR.PRIVx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDLEF
	r	r	r	r	r	r	r								r

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

Bit 14 **TOF**: trigger overrun flag clear
0: no effect
1: clears the corresponding TOF flag

Bit 13 **SUSPF**: completed suspension flag
0: no completed suspension event
1: a completed suspension event occurred

Bit 12 **USEF**: user setting error flag
0: no user setting error event
1: a user setting error event occurred

Bit 11 **ULEF**: update link transfer error flag
0: no update link transfer error event
1: a master bus error event occurred while updating a linked-list register from memory

Bit 10 **DTEF**: data transfer error flag
0: no data transfer error event
1: a master bus error event occurred on a data transfer

Bit 9 **HTF**: half transfer flag
0: no half transfer event
1: a half transfer event occurred
A half transfer event is a half block transfer that occurs when half of the bytes of the source block size (rounded-up integer of LPDMA_CxBR1.BNDT[15:0] / 2) has been transferred to the destination.

Bit 8 **TCF**: transfer complete flag
0: no transfer complete event
1: a transfer complete event occurred
A transfer complete event is a block transfer complete or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode LPDMA_CxTR2.TCEM[1:0].

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

Bit 0 **IDLEF**: idle flag
0: channel not in idle state
1: channel in idle state
This idle flag is de-asserted by hardware when the channel is enabled (LPDMA_CxCR.EN = 1) with a valid channel configuration (no USEF to be immediately reported).
This idle flag is asserted after hard reset or by hardware when the channel is back in idle state (in suspended or disabled state).

18.8.9 LPDMA channel x control register (LPDMA_CxCR)

Address offset: 0x64 + 0x80 * x (x = 0 to 3)

Reset value: 0x0000 0000

This register is secure or non-secure depending on the secure state of channel x (LPDMA_SECCFGR.SECx), and privileged or non-privileged, depending on the privileged state of the channel x (LPDMA_PRIVCFGR.PRIVx).

This register is used to control a channel (activate, suspend, abort or disable it).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO[1:0]		Res.	Res.	Res.	Res.	Res.	LSM
								rw	rw						rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	Res.	SUSP	RESET	EN
	rw	rw	rw	rw	rw	rw	rw						rw	w	rw

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

Bits 23:22 **PRIO[1:0]:** priority level of the channel x LPDMA transfer versus others
00: low priority, low weight
01: low priority, mid weight
10: low priority, high weight
11: high priority
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bits 21:17 Reserved, must be kept at reset value.

Bit 16 **LSM**: Link step mode
0: channel executed for the full linked-list and completed at the end of the last LLI (LPDMA_CxLLR = 0). The 16 low-significant bits of the link address are null (LA[15:0] = 0) and all the update bits are null (UT1 = UB1 = UT2 = USA = UDA = ULL = 0). Then LPDMA_CxBR1.BNDT[15:0] = 0.
1: channel executed **once** for the current LLI
First the block transfer is executed as defined by the current internal register file until LPDMA_CxBR1.BNDT[15:0] = 0). Secondly the next linked-list data structure is conditionally uploaded from memory as defined by LPDMA_CxLLR. Then channel execution is completed.
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOIE**: trigger overrun interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 13 **SUSPIE**: completed suspension interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 12 **USEIE**: user setting error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 11 **ULEIE**: update link transfer error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 10 **DTEIE**: data transfer error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 9 **HTIE**: half transfer complete interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 8 **TCIE**: transfer complete interrupt enable
0: interrupt disabled
1: interrupt enabled

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

Bit 2 **SUSP:** suspend
Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 2. Else:
Software must write 1 in order to suspend an active channel (with an ongoing DMA transfer over its master ports).
The software must write 0 in order to resume a suspended channel, following the programming sequence detailed in *Figure 69*.
0: write: resume channel, read: channel not suspended
1: write: suspend channel, read: channel suspended.

Bit 1 **RESET:** reset
This bit is write only. Writing 0 has no impact. Writing 1 implies the reset of the following: the channel internal state, SUSP and EN bits (whatever is written receptively in bit 2 and bit 0).
The reset is effective when the channel is in steady state, meaning one of the following:
- active channel in suspended state (LPDMA_CxSR.SUSPF = 1 and LPDMA_CxSR.IDLEF = LPDMA_CxCR.EN = 1)
- channel in disabled state (LPDMA_CxSR.IDLEF = 1 and LPDMA_CxCR.EN = 0).
After writing a RESET, to continue using this channel, the user must explicitly reconfigure the channel including the hardware-modified configuration registers (LPDMA_CxBR1, LPDMA_CxSAR and LPDMA_CxDAR) before enabling again the channel (see the programming sequence in *Figure 70*).
0: no channel reset
1: channel reset

Bit 0 **EN:** enable
Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 0. Else:
this bit is de-asserted by hardware when there is a transfer error (master bus error or user setting error) or when there is a channel transfer complete (channel ready to be configured, e.g. if LSM = 1 at the end of a single execution of the LLI).
Else, this bit can be asserted by software.
Writing 0 into this EN bit is ignored.
0: write: ignored, read: channel disabled
1: write: enable channel, read: channel enabled

18.8.10 LPDMA channel x transfer register 1 (LPDMA_CxTR1)

Address offset: 0x090 + 0x80 * x (x = 0 to 3)

Reset value: 0x0000 0000

This register is secure or non-secure depending on the secure state of channel x (LPDMA_SECCFGR.SECx) except for secure DSEC and SSEC, privileged or non-privileged, depending on the privileged state of the channel x in LPDMA_PRIVCFGR.PRIVx.

This register controls the transfer of a channel x.

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1.

This register must be written when the channel is completed. Then the hardware has de-asserted LPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by LPDMA from the memory if LPDMA_CxLLR.UT1 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DSEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DINC	Res.	DDW_LOG2[1:0]	
rw												rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSEC	Res.	Res.	Res.	PAM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SINC	Res.	SDW_LOG2[1:0]	
rw				rw								rw		rw	rw

Bit 31 **DSEC**: security attribute of the LPDMA transfer to the destination

If LPDMA_SECCFGR.SECx = 1 and the access is secure:

0: LPDMA transfer non-secure

1: LPDMA transfer secure

This is a secure register bit. This bit can only be read by a secure software. This bit must be written by a secure software when LPDMA_SECCFGR.SECx = 1. A secure write is ignored when LPDMA_SECCFGR.SECx = 0.

When LPDMA_SECCFGR.SECx is de-asserted, this DSEC bit is also de-asserted by hardware (on a secure reconfiguration of the channel as non-secure), and the LPDMA transfer to the destination is non-secure.

Bits 30:20 Reserved, must be kept at reset value.

Bit 19 **DINC**: destination incrementing single

0: fixed single

1: contiguously incremented single

The destination address, pointed by LPDMA_CxDAR, is kept constant after a single transfer, or is incremented by the offset value corresponding to a contiguous data after a single transfer.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **DDW_LOG2[1:0]**: binary logarithm of the destination data width of a single in bytes
00: byte
01: half-word (2 bytes)
10: word (4 bytes)
11: user setting error reported and no transfer issued

Note: Setting a 8-byte data width causes a user setting error to be reported and none transfer is issued.
A destination single transfer must have an aligned address with its data width (start address LPDMA_CxDAR[2:0] versus DDW_LOG2[1:0]). Otherwise a user setting error is reported and none transfer is issued.

Bit 15 **SSEC**: security attribute of the LPDMA transfer from the source
If LPDMA_SECCFGR.SECx = 1 and the access is secure:
0: LPDMA transfer non-secure
1: LPDMA transfer secure
This is a secure register bit. This bit can only be read by a secure software. This bit must be written by a secure software when LPDMA_SECCFGR.SECx =1 . A secure write is ignored when LPDMA_SECCFGR.SECx = 0.
When LPDMA_SECCFGR.SECx is de-asserted, this SSEC bit is also de-asserted by hardware (on a secure reconfiguration of the channel as non-secure), and the LPDMA transfer from the source is non-secure.

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

Bit 11 **PAM**: padding/alignment mode
If DDW_LOG2[1:0] = SDW_LOG2[1:0]: if the data width of a single destination transfer is equal to the data width of a single source transfer, this bit is ignored.
Else, in the following enumerated values, the condition PAM_1 is when destination data width is higher that source data width, and the condition PAM_2 is when destination data width is higher than source data width.
Condition: PAM_1
0: source data is transferred as right aligned, padded with 0s up to the destination data width
1: source data is transferred as right aligned, sign extended up to the destination data width
Condition: PAM_2
0: source data is transferred as right aligned, left-truncated down to the destination data width
1: source data is transferred as left-aligned, right-truncated down to the destination data width

Bits 10:4 Reserved, must be kept at reset value.

Bit 3 **SINC**: source incrementing single
0: fixed single
1: contiguously incremented single
The source address, pointed by LPDMA_CxSAR, is kept constant after a single transfer or is incremented by the offset value corresponding to a contiguous data after a single transfer.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **SDW_LOG2[1:0]**: binary logarithm of the source data width of a single in bytes
00: byte
01: half-word (2 bytes)
10: word (4 bytes)
11: user setting error reported and no transfer issued

Note: Setting a 8-byte data width causes a user setting error to be reported and none transfer is issued.
a source block size must be a multiple of the source data width (LPDMA_CxBR1.BNDT[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and no transfer is issued.
A source single transfer must have an aligned address with its data width (start address LPDMA_CxSAR[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and none transfer is issued.

18.8.11 LPDMA channel x transfer register 2 (LPDMA_CxTR2)

Address offset: 0x094 + 0x80 * x (x = 0 to 3)

Reset value: 0x0000 0000

This register is secure or non-secure depending on the secure state of channel x (LPDMA_SECCFGR.SECx), and privileged or non-privileged, depending on the privileged state of channel x (LPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x.

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN =1 .

This register must be written when the channel is completed (the hardware de-asserted LPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by LPDMA from the memory, if LPDMA_CxLLR.UT2 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCEM[1:0]		Res.	Res.	Res.	Res.	TRIGPOL[1:0]		Res.	Res.	Res.	TRIGSEL[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
TRIGM[1:0]		Res.	Res.	BREQ	Res.	SWREQ	Res.	Res.	Res.	Res.	REQSEL[4:0]				
rw	rw			rw		rw					rw	rw	rw	rw	rw

Bits 31:30 **TCEM[1:0]**: transfer complete event mode

These bits define the transfer granularity for the transfer complete and half transfer complete events generation.

00: at block level (when LPDMA_CxBR1.BNDT[15:0] = 0): the complete (and the half) transfer event is generated at the (respectively half of the) end of a block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with LPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half transfer event is generated.

01: same as 00

10: at LLI level: the complete transfer event is generated at the end of the LLI transfer, including the update of the LLI if any. The half transfer event is generated at the half of the LLI data transfer (the LLI data transfer being a block transfer), if any data transfer.

Note: If the initial LLI_0 data transfer is null/void (i.e. directly programmed by the internal register file with LPDMA_CxBR1.BNDT[15:0] = 0), then the half transfer event is not generated, and the transfer complete event is generated when is completed the loading of the LLI_1.

11: at channel level: the complete transfer event is generated at the end of the last LLI transfer. The half transfer event is generated at the half of the data transfer of the last LLI. The last LLI updates the link address LPDMA_CxLLR.LA[15:2] to zero and clears all the LPDMA_CxLLR update bits (UT1, UT2, UB1, USA, UDA and ULL). If the channel transfer is continuous/infinite, no event is generated.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **TRIGPOL[1:0]**: trigger event polarity

These bits define the polarity of the selected trigger event input defined by TRIGSEL[4:0].

00: no trigger (masked trigger event)

01: trigger on the rising edge

10: trigger on the falling edge

11: same as 00

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

Bits 20:16 **TRIGSEL[4:0]**: trigger event input selection

These bits select the trigger event input of the LPDMA transfer (as per *Section 18.3.5*), with an active trigger event if TRIGPOL[1:0] = 00.

Bits 15:14 **TRIGM[1:0]**: trigger mode

These bits define the transfer granularity for its conditioning by the trigger.

If the channel x is enabled (LPDMA_CxCR.EN asserted) with TRIGPOL[1:0] = 0b00 or 0b11, these TRIGM[1:0] bits are ignored.

Else, a DMA transfer is conditioned by at least one trigger hit:

00: at block level: the first single read of each block transfer is conditioned by one hit trigger.

01: same as 00

10: at link level: a LLI link transfer is conditioned by one hit trigger. The LLI data transfer (if any) is not conditioned.

11: at programmed single level: each programmed single read is conditioned by one hit trigger.

The LPDMA monitoring of a trigger for channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (TRIGPOL[1:0] = 0b01 or respectively TRIGPOL[1:0] = 0b10).

The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer; and if a new trigger is detected then, this hit is internally memorized to grant the next transfer, as long as the defined rising or falling edge is not modified, and the TRIGSEL[4:0] is not modified, and the channel is enabled.

Transferring a next LLI_{n+1} that updates the LPDMA_CxTR2 with a new value for any of TRIGSEL[4:0] or TRIGPOL[1:0], resets the monitoring, trashing the memorized hit of the formerly defined LLI_n trigger.

After a first new trigger hit_{n+1} is memorized, if another second trigger hit_{n+2} is detected and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized, and a trigger overrun flag is reported (LPDMA_CxSR.TOF = 1), an interrupt is generated if enabled (LPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **BREQ**: block hardware request

If the channel x is activated (LPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a single level.

1: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a block level (see *Section 18.3.4*).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **SWREQ**: software request

This bit is internally taken into account when LPDMA_CxCR.EN is asserted.

0: no software request. The selected hardware request REQSEL[4:0] is taken into account.

1: software request for a memory-to-memory transfer. The default selected hardware request as per REQSEL[4:0] is ignored.

Bits 8:5 Reserved, must be kept at reset value.

Bits 4:0 **REQSEL[4:0]**: DMA hardware request selection

These bits are ignored if channel x is activated (LPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer). Else, the selected hardware request is internally taken into account as per *Section 18.3.3*.

Caution: The user must not assign a same input hardware request (same REQSEL[4:0] value) to different active DMA channels (LPDMA_CxCR.EN = 1 and LPDMA_CxTR2.SWREQ = 0 for these channels). DMA is not intended to hardware support the case of simultaneous enabled channels incorrectly configured with a same hardware peripheral request signal, and there is no user setting error reporting.

18.8.12 LPDMA channel x block register 1 (LPDMA_CxBR1)

Address offset: 0x098 + 0x80 * x (x = 0 to 3)

Reset value: 0x0000 0000

This register is secure or non-secure depending on the secure state of channel x (LPDMA_SECCFGR.SECx), and privileged or non-privileged, depending on the privileged state of channel x (LPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has de-asserted LPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, or LLI or full linked-list.

In linked-list mode, during the link transfer:

- if LPDMA_CxLLR.UB1 = 1, this register is automatically updated by DMA from the next LLI in memory.
- If LPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed values for the field BNDT[15:0].
- If all the update bits LPDMA_CxLLR.Uxx are null and if LPDMA_CxLLR.LA[15:0] # 0, the current LLI is the last one and is continuously executed: this register is automatically and internally restored with the programmed value for BNDT[15:0] after each execution of this final LLI
- If LPDMA_CxLLR = 0, this register and BNDT[15:0] are kept as null, channel x is completed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BNDT[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 **BNDT[15:0]**: block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this field becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):

- if LPDMA_CxLLR.UB1 = 1, this field is updated by the LLI in the memory.
- if LPDMA_CxLLR.UB1 = 0 and if there is at least one non null Uxx update bit, this field is internally restored to the programmed value.
- if all LPDMA_CxLLR.Uxx = 0 and if LPDMA_CxLLR.LA[15:0] = 0, this field is internally restored to the programmed value (infinite/continuous last LLI).
- if LPDMA_CxLLR = 0, this field is kept as zero following the last LLI data transfer.

Note: A non-null source block size must be a multiple of the source data width (BNDT[2:0] versus LPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and none transfer is issued.

18.8.13 LPDMA channel x source address register (LPDMA_CxSAR)

Address offset: 0x09C + 0x80 * x (x = 0 to 3)

Reset value: 0x0000 0000

This register is secure or non-secure depending on the secure state of channel x (LPDMA_SECCFGR.SECx), and privileged or non-privileged, depending on the privileged state of channel x (LPDMA_PRIVCFGR.PRIVx).

This register configures the source start address of a transfer.

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next single transfer from the source.

This register must be written when the channel is completed (then the hardware has de-asserted LPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by LPDMA from the memory if LPDMA_CxLLR.USA = 1.

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

Bits 31:0 **SA[31:0]**: source address

This field is the pointer to the address from which the next data is read.

During the channel activity, depending on the source addressing mode (LPDMA_CxTR1.SINC), this field is either kept fixed or incremented by the data width (LPDMA_CxTR1.SDW_LOG2[1:0]) after each single source data, reflecting the next address from which data is read.

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by LPDMA from the memory, provided the LLI is set with LPDMA_CxLLR.USA = 1.

Note: A source address must be aligned with the programmed data width of a source single (SA[32:0] versus LPDMA_CxTR1.SDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

18.8.14 LPDMA channel x destination address register (LPDMA_CxDAR)

Address offset: 0x0A0 + 0x80 * x (x = 0 to 3)

Reset value: 0x0000 0000

This register is secure or non-secure depending on the secure state of channel x (LPDMA_SECCFGR.SECx), and privileged or non-privileged, depending on the privileged state of channel x (LPDMA_PRIVCFGR.PRIVx).

This register configures the destination start address of a transfer.

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next single transfer to the destination.

This register must be written when the channel is completed (then the hardware has de-asserted LPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by LPDMA from the memory if LPDMA_CxLLR.UDA = 1.

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

Bits 31:0 **DA[31:0]**: destination address

This field is the pointer to the address from which the next data is written.

During the channel activity, depending on the destination addressing mode (LPDMA_CxTR1.DINC), this field is kept fixed or incremented by the data width (LPDMA_CxTR1.DDW_LOG2[21:0]) after each single destination data, reflecting the next address from which data is written.

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by DMA from the memory, provided the LLI is set with LPDMA_CxLLR.UDA = 1.

Note: A destination address must be aligned with the programmed data width of a destination single (DA[2:0] versus LPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

18.8.15 LPDMA channel x linked-list address register (LPDMA_CxLLR)

Address offset: 0x0CC + 0x80 * x (x = 0 to 3)

Reset value: 0x0000 0000

This register is secure or non-secure depending on the secure state of channel x (LPDMA_SECCFGR.SECx), and privileged or non-privileged, depending on the privileged state of channel x (LPDMA_PRIVCFGR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when LPDMA_CxCR.EN = 0.

This register is read-only when LPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has de-asserted LPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by LPDMA from the memory if LPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	rw	rw	rw											rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LA[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **UT1**: Update LPDMA_CxTR1 from memory

This bit controls the update of the LPDMA_CxTR1 register from the memory during the link transfer.

0: no LPDMA_CxTR1 update

1: LPDMA_CxTR1 update

Bit 30 **UT2**: Update LPDMA_CxTR2 from memory

This bit controls the update of the LPDMA_CxTR2 register from the memory during the link transfer.

0: no LPDMA_CxTR2 update

1: LPDMA_CxTR2 update

Bit 29 **UB1**: Update LPDMA_CxBR1 from memory

This bit controls the update of the LPDMA_CxBR1 register from the memory during the link transfer.

0: no LPDMA_CxBR1 update from memory and internally restored to the previous programmed value

1: LPDMA_CxBR1 update

Bit 28 **USA**: update LPDMA_CxSAR from memory

This bit controls the update of the LPDMA_CxSAR register from the memory during the link transfer.

0: no LPDMA_CxSAR update

1: LPDMA_CxSAR update

Bit 27 **UDA**: Update LPDMA_CxDAR register from memory

This bit is used to control the update of the LPDMA_CxDAR register from the memory during the link transfer.

0: no LPDMA_CxDAR update

1: LPDMA_CxDAR update

Bits 26:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update LPDMA_CxLLR register from memory

This bit is used to control the update of the LPDMA_CxLLR register from the memory during the link transfer.

0: no LPDMA_CxLLR update

1: LPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: pointer (16-bit low-significant address) to the next linked-list data structure

If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:20] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list DMA register file.

Else, this field is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list DMA internal register file (LPDMA_CxTR1, LPDMA_CxTR2, LPDMA_CxBR1, LPDMA_CxSAR, LPDMA_CxDAR and LPDMA_CxLLR).

Note: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

Bits 1:0 Reserved, must be kept at reset value.

18.8.16 LPDMA register map

Table 141. LPDMA 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	LPDMA_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.	SEC3	SEC2	SEC1	SEC0
	Reset value																													0	0	0	0
0x004	LPDMA_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.	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value																													0	0	0	0
0x008	LPDMA_ RCFGLOCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK3	LOCK2	LOCK1	LOCK0
	Reset value																													0	0	0	0
0x00C	LPDMA_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.	Res.	Res.	MIS3	MIS2	MIS1	MIS0
	Reset value																													0	0	0	0
0x010	LPDMA_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.	Res.	Res.	MIS3	MIS2	MIS1	MIS0
	Reset value																													0	0	0	0
0x014 - 0x04C	Reserved	Res.																															
0x050+ 0x080 * x (x = 0 to 3)	LPDMA_CxLBAR	LBA[31:16]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x05C+ 0x080 * x (x = 0 to 3)	LPDMA_CxFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																		0	0	0	0	0	0	0								

Table 141. LPDMA 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+ 0x080 * x (x = 0 to 3)	LPDMA_CxSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDLEF
	Reset value																		0	0	0	0	0	0	0								1
0x064+ 0x080 * x (x = 0 to 3)	LPDMA_CxCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO[1:0]		Res.	Res.	Res.	Res.	Res.	LSM	Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	Res.	SUSP	RESET	EN
	Reset value									0	0						0		0	0	0	0	0	0	0						0	0	0
0x090+ 0x080 * x (x = 0 to 3)	LPDMA_CxTR1	DSEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DINC	Res.	DDW_LOG2[1:0]		SSEC	Res.	Res.	Res.	PAM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SINC	Res.	SDW_LOG2[1:0]	
	Reset value	0												0		0	0	0				0								0		0	0
0x094+ 0x080 * x (x = 0 to 3)	LPDMA_CxTR2	TCEM [1:0]		Res.	Res.	Res.	Res.	TRIGPOL [1:0]		Res.	Res.	Res.	TRIGSEL [4:0]				Res.	TRIGM [1:0]		Res.	Res.	Res.	BREQ	SWREQ	Res.	Res.	Res.	Res.	REQSEL [4:0]				
	Reset value	0	0					0	0				0	0	0	0		0	0				0	0					0	0	0	0	0
0x098+ 0x080 * x (x = 0 to 3)	LPDMA_CxBR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BNDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x09C+ 0x080 * x (x = 0 to 3)	LPDMA_CxSAR	SA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A0+ 0x080 * x (x = 0 to 3)	LPDMA_CxDAR	DA[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
0x0CC+ 0x080 * x (x = 0 to 3)	LPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL	LA[15:2]														Res.	Res.
	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* for the register boundary addresses.

19 Chrom-ART Accelerator controller (DMA2D)

19.1 DMA2D introduction

The Chrom-ART Accelerator (DMA2D) is a specialized DMA dedicated to image manipulation. It can perform the following operations:

- fill a part or the whole of a destination image with a specific color
- copy a part or the whole of a source image into a part or the whole of a destination image
- copy a part or the whole of a source image into a part or the whole of a destination image with a pixel format conversion
- blend a part and/or two complete source images with different pixel format and copy the result into a part or the whole of a destination image with a different color format

All the classical color coding schemes are supported from 4-bit up to 32-bit per pixel with indexed or direct color mode. The DMA2D has its own dedicated memories for CLUTs (color look-up tables).

19.2 DMA2D main features

The main DMA2D features are:

- Single AHB master bus architecture
- AHB slave programming interface supporting 8/16/32-bit accesses (except for CLUT accesses which are 32-bit)
- User programmable working area size
- User programmable offset for sources and destination areas expressed in pixels or bytes
- User programmable sources and destination addresses on the whole memory space
- Up to 2 sources with blending operation
- Alpha value can be modified (source value, fixed value or modulated value)
- User programmable source and destination color format
- Up to 11 color formats supported from 4-bit up to 32-bit per pixel with indirect or direct color coding
- 2 internal memories for CLUT storage in indirect color mode
- Automatic CLUT loading or CLUT programming via the CPU
- User programmable CLUT size
- Internal timer to control AHB bandwidth
- 6 operating modes: register-to-memory, memory-to-memory, memory-to-memory with pixel format conversion, memory-to-memory with pixel format conversion and blending, memory-to memory with pixel format conversion, blending and fixed color foreground, and memory-to memory with pixel format conversion, blending and fixed color background.
- Area filling with a fixed color
- Copy from an area to another
- Copy with pixel format conversion between source and destination images

- Copy from two sources with independent color format and blending
- Output buffer byte swapping to support refresh of displays through parallel interface
- Abort and suspend of DMA2D operations
- Watermark interrupt on a user programmable destination line
- Interrupt generation on bus error or access conflict
- Interrupt generation on process completion

19.3 DMA2D functional description

19.3.1 DMA2D block diagram

The DMA2D controller performs direct memory transfer. As an AHB master, it can take the control of the AHB bus matrix to initiate AHB transactions.

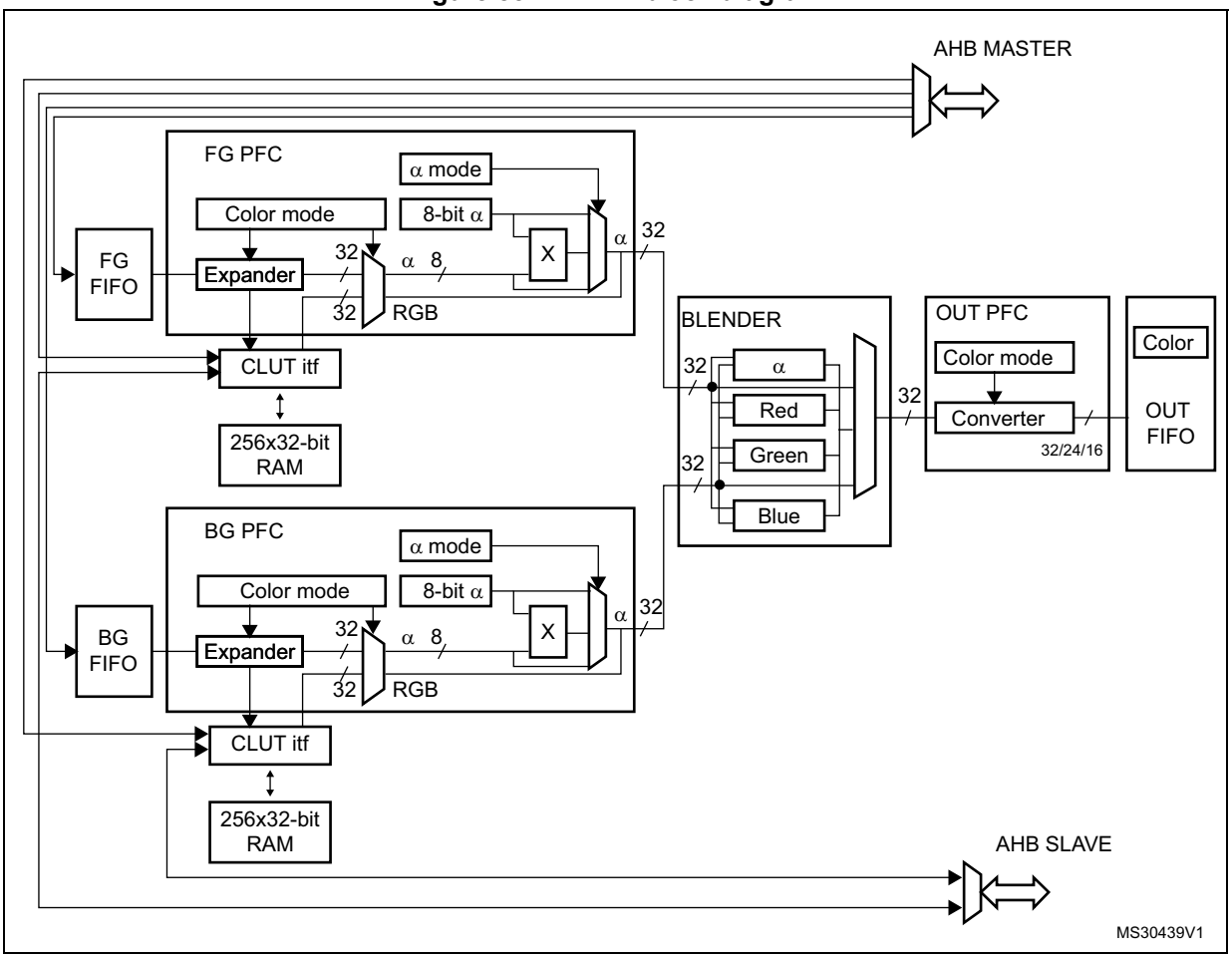
The DMA2D can operate in the following modes:

- Register-to-memory
- Memory-to-memory
- Memory-to-memory with pixel format conversion
- Memory-to-memory with pixel format conversion and blending
- Memory-to memory with pixel format conversion, blending and fixed color foreground
- Memory-to memory with pixel format conversion, blending and fixed color background

The AHB slave port is used to program the DMA2D controller.

The block diagram of the DMA2D is shown in the figure below.

Figure 85. DMA2D block diagram



19.3.2 DMA2D control

The DMA2D controller is configured through the DMA2D control register (DMA2D_CR).

The user application can perform the following operations:

- Select the operating mode.
- Enable/disable the DMA2D interrupt.
- Start/suspend/abort ongoing data transfers.

19.3.3 DMA2D foreground and background FIFOs

The DMA2D foreground (FG) FG FIFO and background (BG) FIFO fetch the input data to be copied and/or processed.

These FIFOs fetch the pixels according to the color format defined in their respective pixel format converter (PFC). They are programmed through a set of control registers:

- DMA2D foreground memory address register (DMA2D_FGMAR)
- DMA2D foreground offset register (DMA2D_FGOR)

- DMA2D background memory address register (DMA2D_BGMAR)
- DMA2D background offset register (DMA2D_BGBOR)
- DMA2D number of lines register (number of lines and pixel per lines) (DMA2D_NLR)

When the DMA2D operates in register-to-memory mode, none of the FIFOs is activated.

When the DMA2D operates in memory-to-memory mode (no pixel format conversion nor blending operation), only the FG FIFO is activated and acts as a buffer.

When the DMA2D operates in memory-to-memory operation with pixel format conversion (no blending operation), the BG FIFO is not activated.

19.3.4 DMA2D foreground and background pixel format converter (PFC)

DMA2D foreground pixel format converter (PFC) and background pixel format converter perform the pixel format conversion to generate a 32-bit per pixel value. The PFC can also modify the alpha channel.

The first stage of the converter converts the color format. The original color format of the foreground pixel and background pixels are configured through the CM[3:0] bits of the DMA2D_FGPFCCR and DMA2D_BGPFCCR, respectively.

The supported input formats are given in the table below.

Table 142. Supported color mode in input

CM[3:0]	Color mode
0000	ARGB8888
0001	RGB888
0010	RGB565
0011	ARGB1555
0100	ARGB4444
0101	L8
0110	AL44
0111	AL88
1000	L4
1001	A8
1010	A4

The color format is coded as follows:

- Alpha value field: transparency
  0xFF value corresponds to an opaque pixel and 0x00 to a transparent one.
- R field for Red
- G field for Green
- B field for Blue
- L field: luminance
  This field is the index to a CLUT to retrieve the three/four RGB/ARGB components.

If the original format is direct color mode (ARGB/RGB), the extension to 8 bits per channel is performed by copying the MSBs into the LSBs. This ensures a perfect linearity of the conversion.

If the original format is indirect color mode (L/AL), a CLUT is required and each pixel format converter is associated with a 256 entry 32-bit CLUT.

If the original format does not include an alpha channel, the alpha value is automatically set to 0xFF (opaque).

For the specific alpha mode A4 and A8, no color information is stored nor indexed. The color to be used for the image generation is fixed and is defined in the DMA2D_FGCOLR for foreground pixels and in the DMA2D_BGCOLR register for background pixels.

The order of the fields in the system memory is defined in the table below.

Table 143. Data order in memory

Color Mode	@ + 3	@ + 2	@ + 1	@ + 0
ARGB8888	A_0[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
RGB888	B_1[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
	G_2[7:0]	B_2[7:0]	R_1[7:0]	G_1[7:0]
	R_3[7:0]	G_3[7:0]	B_3[7:0]	R_2[7:0]
RGB565	R_1[4:0]G_1[5:3]	G_1[2:0]B_1[4:0]	R_0[4:0]G_0[5:3]	G_0[2:0]B_0[4:0]
ARGB1555	A_1[0]R_1[4:0]G_1[4:3]	G_1[2:0]B_1[4:0]	A_0[0]R_0[4:0]G_0[4:3]	G_0[2:0]B_0[4:0]
ARGB4444	A_1[3:0]R_1[3:0]	G_1[3:0]B_1[3:0]	A_0[3:0]R_0[3:0]	G_0[3:0]B_0[3:0]
L8	L_3[7:0]	L_2[7:0]	L_1[7:0]	L_0[7:0]
AL44	A_3[3:0]L_3[3:0]	A_2[3:0]L_2[3:0]	A_1[3:0]L_1[3:0]	A_0[3:0]L_0[3:0]
AL88	A_1[7:0]	L_1[7:0]	A_0[7:0]	L_0[7:0]
L4	L_7[3:0]L_6[3:0]	L_5[3:0]L_4[3:0]	L_3[3:0]L_2[3:0]	L_1[3:0]L_0[3:0]
A8	A_3[7:0]	A_2[7:0]	A_1[7:0]	A_0[7:0]
A4	A_7[3:0]A_6[3:0]	A_5[3:0]A_4[3:0]	A_3[3:0]A_2[3:0]	A_1[3:0]A_0[3:0]

The 24-bit RGB888 aligned on 32 bits is supported through the ARGB8888 mode.

Once the 32-bit value is generated, the alpha channel can be modified according to AM[1:0] in DMA2D_FGPFCCR/DMA2D_BGPFCCR registers as shown in the table below.

The alpha channel can be:

- kept as it is (no modification)
- replaced by the ALPHA[7:0] value of DMA2D_FGPFCCR/DMA2D_BGPFCCR
- replaced by the original alpha value multiplied by the ALPHA[7:0] value of DMA2D_FGPFCCR/DMA2D_BGPFCCR divided by 255.

Table 144. Alpha mode configuration

AM[1:0]	Alpha mode
00	No modification
01	Replaced by value in DMA2D_xxPFCCR
10	Replaced by original value multiplied by the value in DMA2D_xxPFCCR / 255
11	Reserved

Note: To support the alternate format, the incoming alpha value can be inverted setting AI in DMA2D_FGPFCCR/DMA2D_BGPFCCR registers. This applies also to the Alpha value stored in the DMA2D_FGPFCCR/DMA2D_BGPFCCR and in the CLUT.

The R and B fields can also be swapped setting RBS in DMA2D_FGPFCCR/DMA2D_BGPFCCR registers. This applies also to the RGB order used in the CLUT and in the DMA2D_FGCOLR/DMA2D_BGCOLR registers.

19.3.5 DMA2D foreground and background CLUT interface

The CLUT interface manages the CLUT memory access and the automatic loading of the CLUT.

Three access types are possible:

- CLUT read by the PFC during pixel format conversion operation
- CLUT accessed through the AHB slave port when the CPU is reading or writing data into the CLUT
- CLUT written through the AHB master port when an automatic loading of the CLUT is performed

The CLUT memory loading can be done in two different ways:

- Automatic loading

  The following sequence must be followed to load the CLUT:

  a) Program the CLUT address in DMA2D_FGCMAR (foreground CLUT) or DMA2D_BGCMAR (background CLUT).

  b) Program the CLUT size with CS[7:0] in DMA2D_FGPFCCR (foreground CLUT) or DMA2D_BGPFCCR (background CLUT).

  c) Set the START bit in DMA2D_FGPFCCR (foreground CLUT) or DMA2D_BGPFCCR (background CLUT) to start the transfer. During this automatic loading process, the CLUT is not accessible by the CPU. If a conflict occurs, a CLUT access error interrupt is raised assuming CAEIE = 1 in DMA2D_CR.

- Manual loading

  The application has to program the CLUT manually through the DMA2D AHB slave port to which the local CLUT memory is mapped.The foreground CLUT is located at address offset 0x0400 and the background CLUT at address offset 0x0800.

The CLUT format is 24 or 32 bits. It is configured through the CCM bit in DMA2D_FGPFCCR (foreground CLUT) or DMA2D_BGPFCCR (background CLUT) as shown in the table below.

Table 145. Supported CLUT color mode

CCM	CLUT color mode
0	32-bit ARGB8888
1	24-bit RGB888

The way the CLUT data are organized in the system memory is specified in the table below.

Table 146. CLUT data order in system memory

CLUT color mode	@ + 3	@ + 2	@ + 1	@ + 0
ARGB8888	A_0[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
RGB888	B_1[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
	G_2[7:0]	B_2[7:0]	R_1[7:0]	G_1[7:0]
	R_3[7:0]	G_3[7:0]	B_3[7:0]	R_2[7:0]

19.3.6 DMA2D blender

The DMA2D blender blends the source pixels by pair to compute the resulting pixel.

The blending is performed according to the following equation:

$$\text{with } \alpha_{Mult} = \frac{\alpha_{FG} \cdot \alpha_{BG}}{255}$$

$$\alpha_{OUT} = \alpha_{FG} + \alpha_{BG} - \alpha_{Mult}$$

$$C_{OUT} = \frac{C_{FG}.\alpha_{FG} + C_{BG}.\alpha_{BG} - C_{BG}.\alpha_{Mult}}{\alpha_{OUT}} \quad \text{with C = R or G or B}$$

Division is rounded to the nearest lower integer

No configuration register is required by the blender. The blender use depends on the DMA2D operating mode defined by MODE[2:0] in DMA2D_CR.

19.3.7 DMA2D output PFC

The output PFC performs the pixel format conversion from 32 bits to the output format defined by CM[2:0]in DMA2D_OPFCCR.

The supported output formats are given in the table below.

Table 147. Supported color mode in output

CM[2:0]	Color mode
000	ARGB8888
001	RGB888
010	RGB565
011	ARGB1555
100	ARGB4444

Note: To support the alternate format, the calculated alpha value is inverted setting AI bit in DMA2D_OPFCCR. This applies also to the alpha value used in DMA2D_OCOLR.

The R and B fields can also be swapped setting RBS in DMA2D_OPFCCR. This applies also to the RGB order used in DMA2D_OCOLR.

19.3.8 DMA2D output FIFO

The output FIFO programs the pixels according to the color format defined in the output PFC.

The destination area is defined through a set of control registers:

- DMA2D output memory address register (DMA2D_OMAR)
- DMA2D output offset register (DMA2D_OOR**)**
- DMA2D number of lines register (number of lines and pixel per lines) (DMA2D_NLR**)**

If the DMA2D operates in register-to-memory mode, the configured output rectangle is filled by the color specified in DMA2D_OCOLR which contains a fixed 32-, 24-, or 16-bit value. The format is selected by CM[2:0] in DMA2D_OPFCCR.

The data are stored into the memory in the order defined in the table below.

Table 148. Data order in memory

Color mode	@ + 3	@ + 2	@ + 1	@ + 0
ARGB8888	A_0[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
RGB888	B_1[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
	G_2[7:0]	B_2[7:0]	R_1[7:0]	G_1[7:0]
	R_3[7:0]	G_3[7:0]	B_3[7:0]	R_2[7:0]
RGB565	R_1[4:0]G_1[5:3]	G_1[2:0]B_1[4:0]	R_0[4:0]G_0[5:3]	G_0[2:0]B_0[4:0]
ARGB1555	A_1[0]R_1[4:0]G_1[4:3]	G_1[2:0]B_1[4:0]	A_0[0]R_0[4:0]G_0[4:3]	G_0[2:0]B_0[4:0]
ARGB4444	A_1[3:0]R_1[3:0]	G_1[3:0]B_1[3:0]	A_0[3:0]R_0[3:0]	G_0[3:0]B_0[3:0]

The RGB888 aligned on 32 bits is supported through the ARGB8888 mode.

19.3.9 DMA2D output FIFO byte reordering

The output FIFO bytes are reordered to support display frame buffer update through a parallel interface (F(S)MC) directly from the DMA2D.

The reordering of bytes can be done using:

- RBS bit to swap Red and Blue component
- SB bit to swap byte two-by-two in the output FIFO

When the byte swapping is activated (SB = 1 in DMA2D_OPFCR), the number of pixel per line (PL field in DMA2D_NLR) must be even, the output memory address (MA field in DMA2D_OMAR) must be even, and the output line offset computed in bytes (resulting from LOM field in DMA2D_CR and LO field in DMA2D_OOR values) must be even.
If not, a configuration error is detected.

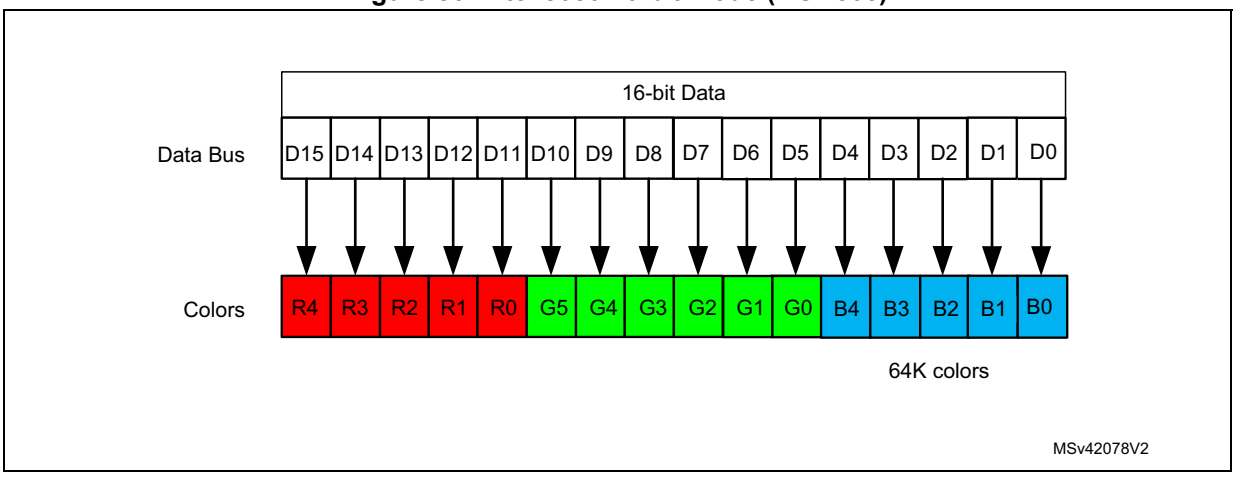
Table 149. Standard data order in memory

Color Mode	@ + 3	@ + 2	@ + 1	@ + 0
RGB888	B_1[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
	G_2[7:0]	B_2[7:0]	R_1[7:0]	G_1[7:0]
	R_3[7:0]	G_3[7:0]	B_3[7:0]	R_2[7:0]
RGB565	R_1[4:0]G_1[5:3]	G_1[2:0]B_1[4:0]	R_0[4:0]G_0[5:3]	G_0[2:0]B_0[4:0]

16-bit mode (RGB565)

This mode is supported without byte reordering by the DMA2D.

Figure 86. Intel 8080 16-bit mode (RGB565)



18/24-bit mode (RGB888)

This mode needs data reordering.

1. Red and the Blue have to be swapped (setting the RBS bit).
2. MSB and the LSB bytes of a half-word have to be swapped (setting the SB bit).

Figure 87. Intel 8080 18/24-bit mode (RGB888)

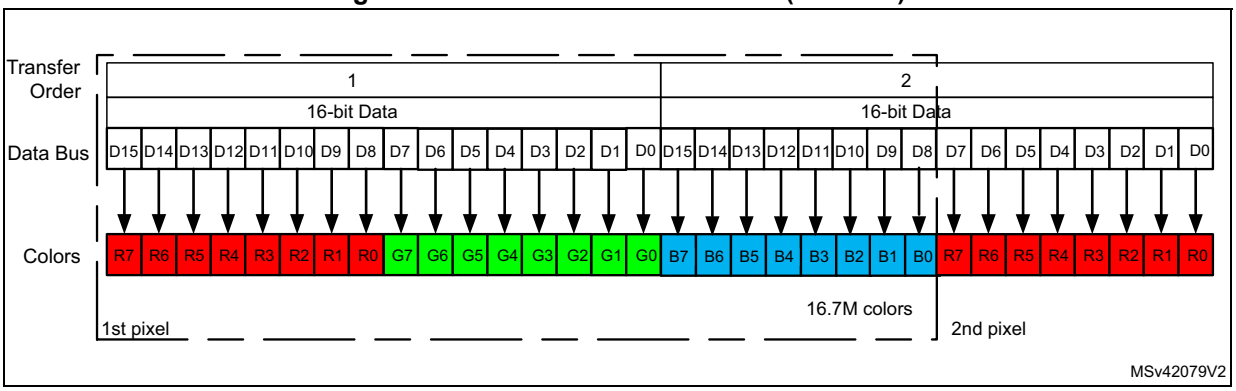


Table 150. Output FIFO byte reordering steps

Steps	@ + 3	@ + 2	@ + 1	@ + 0
Original data ordering	B_1[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
	G_2[7:0]	B_2[7:0]	R_1[7:0]	G_1[7:0]
	R_3[7:0]	G_3[7:0]	B_3[7:0]	R_2[7:0]
Setting the RBS bit				
Data ordering after red and blue swap (RBS set)	R_1[7:0]	B_0[7:0]	G_0[7:0]	R_0[7:0]
	G_2[7:0]	R_2[7:0]	B_1[7:0]	G_1[7:0]
	B_3[7:0]	G_3[7:0]	R_3[7:0]	B_2[7:0]
Setting the SB bit				
Data ordering after byte swapping (SB set)	B_0[7:0]	R_1[7:0]	R_0[7:0]	G_0[7:0]
	R_2[7:0]	G_2[7:0]	G_1[7:0]	B_1[7:0]
	G_3[7:0]	B_3[7:0]	B_2[7:0]	R_3[7:0]

19.3.10 DMA2D AHB master port timer

An 8-bit timer is embedded into the AHB master port to provide an optional limitation of the bandwidth on the crossbar.

This timer is clocked by the AHB clock and counts a dead time between two consecutive accesses. This limits the bandwidth availability.

The timer enabling and the dead time value are configured through the AHB master port timer configuration register (DMA2D_AMPTCR).

19.3.11 DMA2D transactions

DMA2D transactions consist of a sequence of a given number of data transfers.
The number of data and the width can be programmed by software.

Each DMA2D data transfer is composed of up to four steps:

1. Data loading from the memory location pointed by DMA2D_FGMAR and pixel format conversion as defined in DMA2D_FGCR

2. Data loading from a memory location pointed by DMA2D_BGMAR and pixel format conversion as defined in DMA2D_BGCR
3. Blending of all retrieved pixels according to the alpha channels resulting of the PFC operation on alpha values
4. Pixel format conversion of the resulting pixels according to DMA2D_OCR and programming of the data to the memory location addressed through DMA2D_OMAR

19.3.12 DMA2D configuration

Both source and destination data transfers can target peripherals and memories in the whole 4 Gbyte memory area, at addresses ranging between 0x0000 0000 and 0xFFFF FFFF.

The DMA2D can operate in any of the following modes selected through MODE[2:0] in DMA2D_CR:

- Register-to-memory
- Memory-to-memory
- Memory-to-memory with PFC
- Memory-to-memory with PFC and blending
- Memory-to-memory with PFC, blending and fixed FG color
- Memory-to-memory with PFC, blending and fixed BG color

Register-to-memory

The register-to-memory mode is used to fill a user defined area with a predefined color.

The color format is set in DMA2D_OPFCCR.

The DMA2D does not perform any data fetching from any source. It just writes the color defined in DMA2D_OCOLR to the area located at the address pointed by DMA2D_OMAR, and defined in DMA2D_NLR and DMA2D_OOR.

Memory-to-memory

In memory-to-memory mode, the DMA2D does not perform any graphical data transformation. The foreground input FIFO acts as a buffer and the data are transferred from the source memory location defined in DMA2D_FGMAR to the destination memory location pointed by DMA2D_OMAR.

The color mode programmed by CM[3:0] in DMA2D_FGPFCCR defines the number of bits per pixel for both input and output.

The size of the area to be transferred is defined by DMA2D_NLR and DMA2D_FGOR for the source, and by DMA2D_NLR and DMA2D_OOR for the destination.

Memory-to-memory with PFC

In this mode, the DMA2D performs a pixel format conversion of the source data and stores them in the destination memory location.

The size of the areas to be transferred are defined by DMA2D_NLR and DMA2D_FGOR for the source, and by DMA2D_NLR and DMA2D_OOR for the destination.

Data are fetched from the location defined in DMA2D_FGMAR and processed by the foreground PFC. The original pixel format is configured through DMA2D_FGPFCCR.

If the original pixel format is direct color mode, then the color channels are all expanded to 8 bits.

If the pixel format is indirect color mode, the associated CLUT has to be loaded into the CLUT memory.

The CLUT loading can be done automatically by following the sequence below:

1. Set the CLUT address into DMA2D_FGCMAR.
2. Set the CLUT size with CS[7:0] bits in DMA2D_FGPFCCR.
3. Set the CLUT format (24 or 32 bits) with CCM in DMA2D_FGPFCCR.
4. Start the CLUT loading by setting START i nDMA2D_FGPFCCR.

Once the CLUT loading is complete, the CTCIF flag in DMA2D_IFR is raised, and an interrupt is generated if CTCIE = 1 in DMA2D_CR. The automatic CLUT loading process can not work in parallel with classical DMA2D transfers.

The CLUT can also be filled by the CPU or by any other master through the APB port. The access to the CLUT is not possible when a DMA2D transfer is ongoing and uses the CLUT (indirect color format).

In parallel to the color conversion process, the alpha value is added or changed depending on the value programmed in DMA2D_FGPFCCR. If the original image does not have an alpha channel, a default alpha value of 0xFF is automatically added to obtain a fully opaque pixel. The alpha value is modified according to AM[1:0] in DMA2D_FGPFCCR:

- It can be unchanged.
- It can be replaced by the value defined by ALPHA[7:0] in DMA2D_FGPFCCR.
- It can be replaced by the original value multiplied by ALPHA[7:0] divided by 255.

The resulting 32-bit data are encoded by the OUT PFC into the format specified by CM[2:0] in DMA2D_OPFCCR. The output pixel format cannot be the indirect mode since no CLUT generation process is supported.

The processed data are written into the destination memory location pointed by DMA2D_OMAR.

Memory-to-memory with PFC and blending

In this mode, two sources are fetched in the foreground and background FIFOs from the memory locations defined by DMA2D_FGMAR and DMA2D_BGMAR.

The two pixel format converters have to be configured as described in the memory-to-memory mode. Their configurations can be different as each pixel format converter is independent and has its own CLUT memory.

Once each pixel has been converted into 32 bits by its respective PFC, all pixels are blended according to the equation below:

$$\text{with } \alpha_{Mult} = \frac{\alpha_{FG} \cdot \alpha_{BG}}{255}$$

$$\alpha_{OUT} = \alpha_{FG} + \alpha_{BG} - \alpha_{Mult}$$

$$C_{OUT} = \frac{C_{FG}.\alpha_{FG} + C_{BG}.\alpha_{BG} - C_{BG}.\alpha_{Mult}}{\alpha_{OUT}} \quad \text{with C = R or G or B}$$

Division are rounded to the nearest lower integer

The resulting 32-bit pixel value is encoded by the output PFC according to the specified output format, and the data are written into the destination memory location pointed by DMA2D_OMAR.

Memory-to-memory with PFC, blending and fixed color FG

In this mode, only one source is fetched in the background FIFO from the memory location defined by DMA2D_BGMAR.

The value of the foreground color is given by DMA2D_FGCOLR and the alpha value is set to 0xFF (opaque).

The alpha value can be replaced or modified according to AM[1:0] and ALPHA[7:0] in DMA2D_FGPFCR.

The two pixel format converters have to be configured as described in the memory-to-memory mode. Their configurations can be different as each pixel format converter is independent and has its own CLUT memory

Once each pixel has been converted into 32 bits by its respective PFC, all pixels are blended together, and the resulting 32-bit pixel value is encoded by the output PFC according to the specified output format. Data are written into the destination memory location pointed by DMA2D_OMAR.

Memory-to-memory with PFC, blending and fixed color BG

In this mode, only one source is fetched in the foreground FIFO from the memory location defined by DMA2D_FGMAR.

The value of the background color is given by DMA2D_BGCOLR and the alpha value is set to 0xFF (opaque).

The alpha value can be replaced or modified according to AM[1:0] and ALPHA[7:0] in DMA2D_BGPFCR.

The two pixel format converters have to be configured as described in the memory-to-memory mode. Their configurations can be different as each pixel format converter is independent and has its own CLUT memory

Once each pixel has been converted into 32 bits by its respective PFC, all pixels are blended together, and the resulting 32-bit pixel value is encoded by the output PFC

according to the specified output format, and the data are written into the destination memory location pointed by DMA2D_OMAR.

Configuration error detection

The DMA2D checks that the configuration is correct before any transfer. The configuration error interrupt flag is set by hardware when a wrong configuration is detected when a new transfer/automatic loading starts. An interrupt is then generated if CEIE = 1 in DMA2D_CR.

The wrong configurations that can be detected are listed below:

- Foreground CLUT automatic loading: MA bits in DMA2D_FGCMAR are not aligned with CCM in DMA2D_FGPFCCR.
- Background CLUT automatic loading: MA bits in DMA2D_BGCMAR are not aligned with CCM in DMA2D_BGPFCCR.
- Memory transfer (except in register-to-memory mode and except in memory-to-memory mode with blending and fixed color FG): MA bits in DMA2D_FGMAR are not aligned with CM in DMA2D_FGPFCCR.
- Memory transfer (except in register-to-memory mode and except in memory-to-memory mode with blending and fixed color FG): CM bits in DMA2D_FGPFCCR are invalid
- Memory transfer (except in register-to-memory mode and except in memory-to-memory mode with blending and fixed color FG): PL bits in DMA2D_NLR are odd while CM in DMA2D_FGPFCCR is A4 or L4.
- Memory transfer (except in register-to-memory mode and except in memory-to-memory mode with blending and fixed color FG): LO bits in DMA2D_FGOR are odd while CM in DMA2D_FGPFCCR is A4 or L4, and LOM bit in DMA2D_CR is pixel mode.
- Memory transfer (only in blending mode and except in memory-to-memory mode with blending and fixed color FG): MA bits in DMA2D_BGMAR are not aligned with CM in DMA2D_BGPFCCR.
- Memory transfer: (only in blending mode and in blending with fixed color FG mode): CM bits in DMA2D_BGPFCCR are invalid
- Memory transfer (only in blending mode and in blending with fixed color FG mode): PL bits in DMA2D_NLR odd while CM in DMA2D_BGPFCCR is A4 or L4.
- Memory transfer (only in blending mode and in blending with fixed color FG mode): LO bits in DMA2D_BGOR are odd while CM in DMA2D_BGPFCCR is A4 or L4, and LOM bit in DMA2D_CR is pixel mode.
- Memory transfer (except in memory-to-memory mode): MA bits in DMA2D_OMAR are not aligned with CM bits in DMA2D_OPFCCR.
- Memory transfer (except in memory to memory mode): CM bits in DMA2D_OPFCCR are invalid.
- Memory transfer with byte swapping: PL bits in DMA2D_NLR are odd or MA bits in DMA2D_OMAR are odd, or LO in bytes (resulting from LOM in DMA2D_CR and LO bits in DMA2D_OOR) are odd while SB = 1 in DMA2D_OPFCCR.
- Memory transfer: NL = 0x0 in DMA2D_NLR
- Memory transfer: PL = 0x0 i nDMA2D_NLR
- Memory transfer: MODE bits in DMA2D_CR are invalid.

19.3.13 DMA2D transfer control (start, suspend, abort and completion)

Once the DMA2D is configured, the transfer can be launched by setting START in DMA2D_CR. Once the transfer is completed, START is automatically reset and TCIF flag in DMA2D_ISR is raised. An interrupt can be generated if TCIE is set in DMA2D_CR.

The user application can suspend the DMA2D at any time by setting SUSP in DMA2D_CR. The transaction can then be aborted by setting ABORT in DMA2D_CR, or can be restarted by resetting SUSP in DMA2D_CR.

The user application can abort at any time an ongoing transaction by setting ABORT in DMA2D_CR. In this case, the TCIF flag is not raised.

Automatic CLUT transfers can also be aborted or suspended by using ABORT or SUSP in DMA2D_CR.

19.3.14 Watermark

A watermark can be programmed to generate an interrupt when the last pixel of a given line has been written to the destination memory area.

The line number is defined by LW[15:0] in DMA2D_LWR.

When the last pixel of this line has been transferred, the TWIF flag in DMA2D_ISR is raised, and an interrupt is generated if TWIE is set in DMA2D_CR.

19.3.15 Error management

Two kind of errors can be triggered:

- AHB master port errors signaled by TEIF in DMA2D_ISR
- conflicts caused by a CLUT access (CPU trying to access the CLUT while a CLUT loading or a DMA2D transfer is ongoing) signaled by CAEIF in DMA2D_ISR.

Both flags are associated to their own interrupt enable flag in DMA2D_CR to generate an interrupt if need be (TEIE and CAEIE).

19.3.16 AHB dead time

To limit the AHB bandwidth use, a dead time between two consecutive AHB accesses can be programmed.

This feature can be enabled by setting EN in DMA2D_AMTCR.

The dead time value is stored into DT[7:0] in DMA2D_AMTCR. This value represents the guaranteed minimum number of cycles between two consecutive transactions on the AHB bus.

The update of the dead time value while the DMA2D runs is taken into account for the next AHB transfer.

19.4 DMA2D interrupts

An interrupt can be generated on the following events:

- Configuration error
- CLUT transfer complete
- CLUT access error
- Transfer watermark reached
- Transfer complete
- Transfer error

Separate interrupt enable bits are available for flexibility.

Table 151. DMA2D interrupt requests

Interrupt event	Event flag	Enable control bit
Configuration error	CEIF	CEIE
CLUT transfer complete	CTCIF	CTCIE
CLUT access error	CAEIF	CAEIE
Transfer watermark	TWF	TWIE
Transfer complete	TCIF	TCIE
Transfer error	TEIF	TEIE

19.5 DMA2D registers

19.5.1 DMA2D control register (DMA2D_CR)

Address offset: 0x0000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CEIE	CTCIE	CAEIE	TWIE	TCIE	TEIE	Res.	LOM	Res.	Res.	Res.	ABORT	SUSP	START
		rw	rw	rw	rw	rw	rw		rw				rs	rw	rs

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

Bits 18:16 **MODE[2:0]**: DMA2D mode
These bits are set and cleared by software. They cannot be modified while a transfer is ongoing.
000: Memory-to-memory (FG fetch only)
001: Memory-to-memory with PFC (FG fetch only with FG PFC active)
010: Memory-to-memory with blending (FG and BG fetch with PFC and blending)
011: Register-to-memory (no FG nor BG, only output stage active)
100: Memory-to-memory with Blending and fixed color FG (BG fetch only with FG and BG PFC active)
101: Memory-to-memory with Blending and fixed color BG (BG fetch only with FG and BG PFC active)
others: Reserved

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **CEIE**: Configuration error interrupt enable
This bit is set and cleared by software.
0: CE interrupt disabled
1: CE interrupt enabled

Bit 12 **CTCIE**: CLUT transfer complete interrupt enable
This bit is set and cleared by software.
0: CTC interrupt disabled
1: CTC interrupt enabled

Bit 11 **CAEIE**: CLUT access error interrupt enable
This bit is set and cleared by software.
0: CAE interrupt disabled
1: CAE interrupt enabled

Bit 10 **TWIE**: Transfer watermark interrupt enable
This bit is set and cleared by software.
0: TW interrupt disabled
1: TW interrupt enabled

Bit 9 **TCIE**: Transfer complete interrupt enable
This bit is set and cleared by software.
0: TC interrupt disabled
1: TC interrupt enabled

Bit 8 **TEIE**: Transfer error interrupt enable
This bit is set and cleared by software.
0: TE interrupt disabled
1: TE interrupt enabled

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LOM**: Line offset mode
This bit configures how is expressed the line offset (pixels or bytes) for the foreground, background and output.
This bit is set and cleared by software. It can not be modified while a transfer is on going.
0: Line offsets expressed in pixels
1: Line offsets expressed in bytes

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **ABORT**: Abort

This bit can be used to abort the current transfer. This bit is set by software and is automatically reset by hardware when the START bit is reset.

0: No transfer abort requested

1: Transfer abort requested

Bit 1 **SUSP**: Suspend

This bit can be used to suspend the current transfer. This bit is set and reset by software. It is automatically reset by hardware when the START bit is reset.

0: Transfer not suspended

1: Transfer suspended

Bit 0 **START**: Start

This bit can be used to launch the DMA2D according to the parameters loaded in the various configuration registers. This bit is automatically reset by the following events:

- at the end of the transfer
- when the data transfer is aborted by the user application (setting ABORT in DMA2D_CR)
- when a data transfer error occurs
- when the data transfer has not started due to a configuration error or another transfer operation already ongoing (automatic CLUT loading).

19.5.2 DMA2D interrupt status register (DMA2D_ISR)

Address offset: 0x0004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEIF	CTCIF	CAEIF	TWIF	TCIF	TEIF
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **CEIF**: Configuration error interrupt flag
This bit is set when START in DMA2D_CR, DMA2DFGPFCCR or DMA2D_BGPFCCR is set and a wrong configuration has been programmed.

Bit 4 **CTCIF**: CLUT transfer complete interrupt flag
This bit is set when the CLUT copy from a system memory area to the internal DMA2D memory is complete.

Bit 3 **CAEIF**: CLUT access error interrupt flag
This bit is set when the CPU accesses the CLUT while the CLUT is being automatically copied from a system memory to the internal DMA2D.

Bit 2 **TWIF**: Transfer watermark interrupt flag
This bit is set when the last pixel of the watermarked line has been transferred.

Bit 1 **TCIF**: Transfer complete interrupt flag
This bit is set when a DMA2D transfer operation is complete (data transfer only).

Bit 0 **TEIF**: Transfer error interrupt flag
This bit is set when an error occurs during a DMA transfer (data transfer or automatic CLUT loading).

19.5.3 DMA2D interrupt flag clear register (DMA2D_IFCR)

Address offset: 0x0008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCEIF	CCTCIF	CAECIF	CTWIF	CTCIF	CTEIF
										rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **CCEIF**: Clear configuration error interrupt flag
Programming this bit to 1 clears CEIF in DMA2D_ISR.

Bit 4 **CCTCIF**: Clear CLUT transfer complete interrupt flag
Programming this bit to 1 clears CTCIF in DMA2D_ISR.

Bit 3 **CAECIF**: Clear CLUT access error interrupt flag
Programming this bit to 1 clears CAEIF in DMA2D_ISR.

Bit 2 **CTWIF**: Clear transfer watermark interrupt flag
Programming this bit to 1 clears TWIF in DMA2D_ISR.

Bit 1 **CTCIF**: Clear transfer complete interrupt flag
Programming this bit to 1 clears TCIF in DMA2D_ISR.

Bit 0 **CTEIF**: Clear Transfer error interrupt flag
Programming this bit to 1 clears TEIF in DMA2D_ISR.

19.5.4 DMA2D foreground memory address register (DMA2D_FGMAR)

Address offset: 0x000C

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]**: Memory address

This field contains the address of the data used for the foreground image. This register can only be written when data transfers are disabled. Once the data transfer has started, this register is read-only.

The address alignment must match the image format selected: for example, a 32-bit per pixel format must be 32-bit aligned, a 16-bit per pixel format must be 16-bit aligned and a 4-bit per pixel format must be 8-bit aligned.

19.5.5 DMA2D foreground offset register (DMA2D_FGOR)

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
LO[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 **LO[15:0]**: Line offset

The line offset used for the foreground image, expressed in pixel when the LOM bit is reset and in byte when the LOM bit is set.

When expressed in pixels, only LO[13:0] is considered, LO[15:14] are ignored.

This value is used for the address generation. It is added at the end of each line to determine the starting address of the next line.

These bits can only be written when data transfers are disabled. Once data transfer has started, they become read-only.

If the image format is 4-bit per pixel, the line offset must be even.

19.5.6 DMA2D background memory address register (DMA2D_BGMAR)

Address offset: 0x0014

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]**: Memory address

This field contains the address of the data used for the background image. This register can only be written when data transfers are disabled. Once a data transfer has started, this register is read-only.

The address alignment must match the image format selected e.g. a 32-bit per pixel format must be 32-bit aligned, a 16-bit per pixel format must be 16-bit aligned and a 4-bit per pixel format must be 8-bit aligned.

19.5.7 DMA2D background offset register (DMA2D_BGOR)

Address offset: 0x0018

Reset value: 0x0000 0000

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

This field contains the line offset used for the background image, expressed in pixel when the LOM bit is reset, and in byte when the LOM bit is set.

When expressed in pixels, only LO[13:0] is considered, LO[15:14] are ignored.

This value is used for the address generation. It is added at the end of each line to determine the starting address of the next line.

These bits can only be written when data transfers are disabled. Once data transfer has started, they become read-only.

If the image format is 4-bit per pixel, the line offset must be even.

19.5.8 DMA2D foreground PFC control register (DMA2D_FGPFCCR)

Address offset: 0x001C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALPHA[7:0]								Res.	Res.	RBS	AI	Res.	Res.	AM[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CS[7:0]								Res.	Res.	START	CCM	CM[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rs	rw	rw	rw	rw	rw

Bits 31:24 **ALPHA[7:0]**: Alpha value

These bits define a fixed alpha channel value which can replace the original alpha value or be multiplied by the original alpha value according to the alpha mode selected through the AM[1:0] bits.

These bits can only be written when data transfers are disabled. Once a transfer has started, they become read-only.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **RBS**: Red Blue swap

This bit is used to swap the R and B to support BGR or ABGR color formats. Once the transfer has started, this bit is read-only.

0: Regular mode (RGB or ARGB)

1: Swap mode (BGR or ABGR)

Bit 20 **AI**: Alpha Inverted

This bit inverts the alpha value. Once the transfer has started, this bit is read-only.

0: Regular alpha

1: Inverted alpha

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **AM[1:0]**: Alpha mode

These bits select the alpha channel value to be used for the foreground image. They can only be written data the transfer are disabled. Once the transfer has started, they become read-only.

00: No modification of the foreground image alpha channel value

01: Replace original foreground image alpha channel value by ALPHA[7:0]

10: Replace original foreground image alpha channel value by ALPHA[7:0] multiplied with original alpha channel value

Other: Reserved

Bits 15:8 **CS[7:0]**: CLUT size

These bits define the size of the CLUT used for the foreground image. Once the CLUT transfer has started, this field is read-only.

The number of CLUT entries is equal to CS[7:0] + 1.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **START**: Start

This bit can be set to start the automatic loading of the CLUT. It is automatically reset:
- at the end of the transfer
- when the transfer is aborted by the user application by setting ABORT in DMA2D_CR
- when a transfer error occurs
- when the transfer has not started due to a configuration error or another transfer operation already ongoing (data transfer or automatic background CLUT transfer)

Bit 4 **CCM**: CLUT color mode

This bit defines the color format of the CLUT. It can only be written when the transfer is disabled. Once the CLUT transfer has started, this bit is read-only.
0: ARGB8888
1: RGB888
Other: Reserved

Bits 3:0 **CM[3:0]**: Color mode

These bits defines the color format of the foreground image. They can only be written when data transfers are disabled. Once the transfer has started, they are read-only.
0000: ARGB8888
0001: RGB888
0010: RGB565
0011: ARGB1555
0100: ARGB4444
0101: L8
0110: AL44
0111: AL88
1000: L4
1001: A8
1010: A4
Other: Reserved

19.5.9 DMA2D foreground color register (DMA2D_FGCOLR)

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.	RED[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
GREEN[7:0]								BLUE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:16 **RED[7:0]**: Red value

These bits define the Red value for the A4 or A8 mode of the foreground image. They can only be written when data transfers are disabled. Once the transfer has started, they are read-only.
These bits can also be used for fixed color FG in memory-to-memory with blending and fixed color FG (BG fetch only with FG and BG PFC active) mode.

Bits 15:8 **GREEN[7:0]**: Green value

These bits defines the Green value for the A4 or A8 mode of the foreground image. They can only be written when data transfers are disabled. Once the transfer has started, They are read-only.

These bits can also be used for fixed color FG in memory-to-memory with blending and fixed color FG (BG fetch only with FG and BG PFC active) mode.

Bits 7:0 **BLUE[7:0]**: Blue value

These bits defines the Blue value for the A4 or A8 mode of the foreground image. They can only be written when data transfers are disabled. Once the transfer has started, They are read-only.

These bits can also be used for fixed color FG in memory-to-memory with blending and fixed color FG (BG fetch only with FG and BG PFC active) mode.

19.5.10 DMA2D background PFC control register (DMA2D_BGPFCCR)

Address offset: 0x0024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALPHA[7:0]								Res.	Res.	RBS	AI	Res.	Res.	AM[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CS[7:0]								Res.	Res.	START	CCM	CM[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rs	rw	rw	rw	rw	rw

Bits 31:24 **ALPHA[7:0]**: Alpha value

These bits define a fixed alpha channel value which can replace the original alpha value or be multiplied with the original alpha value according to the alpha mode selected with AM[1:0]. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **RBS**: Red Blue swap

This bit is used to swap the R and B to support BGR or ABGR color formats. Once the transfer has started, this bit is read-only.

0: Regular mode (RGB or ARGB)

1: Swap mode (BGR or ABGR)

Bit 20 **AI**: Alpha inverted

This bit inverts the alpha value. Once the transfer has started, this bit is read-only.

0: Regular alpha

1: Inverted alpha

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **AM[1:0]**: Alpha mode

These bits define which alpha channel value to be used for the background image. They can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

00: No modification of the foreground image alpha channel value

01: Replace original background image alpha channel value by ALPHA[7:0]

10: Replace original background image alpha channel value by ALPHA[7:0] multiplied with original alpha channel value

Other: Reserved

Bits 15:8 **CS[7:0]**: CLUT size

These bits define the size of the CLUT used for the BG. Once the CLUT transfer has started, this field is read-only.

The number of CLUT entries is equal to CS[7:0] + 1.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **START**: Start

This bit is set to start the automatic loading of the CLUT. This bit is automatically reset:

- at the end of the transfer
- when the transfer is aborted by the user application by setting ABORT in DMA2D_CR
- when a transfer error occurs
- when the transfer has not started due to a configuration error or another transfer operation already on going (data transfer or automatic foreground CLUT transfer).

Bit 4 **CCM**: CLUT color mode

These bits define the color format of the CLUT. This register can only be written when the transfer is disabled. Once the CLUT transfer has started, this bit is read-only.

0: ARGB8888

1: RGB888

Other: Reserved

Bits 3:0 **CM[3:0]**: Color mode

These bits define the color format of the foreground image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

0000: ARGB8888

0001: RGB888

0010: RGB565

0011: ARGB1555

0100: ARGB4444

0101: L8

0110: AL44

0111: AL88

1000: L4

1001: A8

1010: A4

Other: Reserved

19.5.11 DMA2D background color register (DMA2D_BGCOLR)

Address offset: 0x0028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RED[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
GREEN[7:0]								BLUE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:16 **RED[7:0]**: Red value

These bits define the Red value for the A4 or A8 mode of the background. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

These bits are also used for fixed color FG in memory-to-memory with blending and fixed color FG (BG fetch only with FG and BG PFC active) mode.

Bits 15:8 **GREEN[7:0]**: Green value

These bits define the green value for the A4 or A8 mode of the background. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

These bits are also used for fixed color FG in memory-to-memory with blending and fixed color FG (BG fetch only with FG and BG PFC active) mode.

Bits 7:0 **BLUE[7:0]**: Blue value

These bits define the blue value for the A4 or A8 mode of the background. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

These bits are also used for fixed color FG in memory-to-memory with blending and fixed color FG (BG fetch only with FG and BG PFC active) mode.

19.5.12 DMA2D foreground CLUT memory address register (DMA2D_FGCMAR)

Address offset: 0x002C

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]**: Memory address

This field contains the address of the data used for the CLUT address dedicated to the foreground image. This register can only be written when no transfer is ongoing. Once the CLUT transfer has started, this register is read-only.

If the foreground CLUT format is 32-bit, the address must be 32-bit aligned.

19.5.13 DMA2D background CLUT memory address register (DMA2D_BGCMAR)

Address offset: 0x0030

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]**: Memory address

This field contains the address of the data used for the CLUT address dedicated to the background image. This register can only be written when no transfer is on going. Once the CLUT transfer has started, this register is read-only.

If the background CLUT format is 32-bit, the address must be 32-bit aligned.

19.5.14 DMA2D output PFC control register (DMA2D_OPFCCR)

Address offset: 0x0034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RBS	AI	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.	SB	Res.	Res.	Res.	Res.	Res.	Res.	CM[2:0]		
						rw							rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **RBS**: Red Blue swap

This bit is used to swap the R and B to support BGR or ABGR color formats. Once the transfer has started, this bit is read-only.

0: Regular mode (RGB or ARGB)

1: Swap mode (BGR or ABGR)

Bit 20 **AI**: Alpha Inverted

This bit inverts the alpha value. Once the transfer has started, this bit is read-only.

0: Regular alpha

1: Inverted alpha

Bits 19:10 Reserved, must be kept at reset value.

Bit 9 **SB**: Swap bytes

When this bit is set, the bytes in the output FIFO are swapped two by two, the number of pixel per line (PL) must be even, and the output memory address (OMAR) must be even.

This register can only be written when the transfer is disabled. Once the transfer has started, this register is read-only.

0: Bytes in regular order in the output FIFO

1: Bytes are swapped two by two in the output FIFO

Bits 8:3 Reserved, must be kept at reset value.

Bits 2:0 **CM[2: 0]**: Color mode

These bits define the color format of the output image. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

000: ARGB8888

001: RGB888

010: RGB565

011: ARGB1555

100: ARGB4444

Other: Reserved

19.5.15 DMA2D output color register (DMA2D_OCOLR)

Address offset: 0x0038

Reset value: 0x0000 0000

The same register is used to show the color values, with different formats depending on the color mode.

This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.

ARGB8888 or RGB888 color mode

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALPHA[7:0]								RED[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
GREEN[7:0]								BLUE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **ALPHA[7:0]**: Alpha channel value of the output color in ARGB8888 mode (otherwise reserved)

Bits 23:16 **RED[7:0]**: Red value of the output image in ARGB8888 or RGB888 mode

Bits 15:8 **GREEN[7:0]**: Green value of the output image in ARGB8888 or RGB888

Bits 7:0 **BLUE[7:0]**: Blue value of the output image in ARGB8888 or RGB888

19.5.16 DMA2D output color register [alternate] (DMA2D_OCOLR)

Address offset: 0x0038

Reset value: 0x0000 0000

The same register is used to show the color values, with different formats depending on the color mode.

This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.

RGB565 color 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
RED[4:0]					GREEN[5:0]						BLUE[4:0]				
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 15:11 **RED[4:0]**: Red value of the output image in RGB565 mode

Bits 10:5 **GREEN[5:0]**: Green value of the output image in RGB565 mode

Bits 4:0 **BLUE[4:0]**: Blue value of the output image in RGB565 mode

19.5.17 DMA2D output color register [alternate] (DMA2D_OCOLR)

Address offset: 0x0038

Reset value: 0x0000 0000

The same register is used to show the color values, with different formats depending on the color mode.

This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.

ARGB1555 color 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
A	RED[4:0]					GREEN[4:0]					BLUE[4:0]				
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.

Bit 15 **A**: Alpha channel value of the output color in ARGB1555 mode

Bits 14:10 **RED[4:0]**: Red value of the output image in ARGB1555 mode

Bits 9:5 **GREEN[4:0]**: Green value of the output image in ARGB1555 mode

Bits 4:0 **BLUE[4:0]**: Blue value of the output image in ARGB1555 mode

19.5.18 DMA2D output color register [alternate] (DMA2D_OCOLR)

Address offset: 0x0038

Reset value: 0x0000 0000

The same register is used to show the color values, with different formats depending on the color mode.

This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.

ARGB4444 color 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
ALPHA[3:0]				RED[3:0]				GREEN[3:0]				BLUE[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **ALPHA[3:0]**: Alpha channel of the output color value in ARGB4444

Bits 11:8 **RED[3:0]**: Red value of the output image in ARGB4444 mode

Bits 7:4 **GREEN[3:0]**: Green value of the output image in ARGB4444 mode

Bits 3:0 **BLUE[3:0]**: Blue value of the output image in ARGB4444 mode

19.5.19 DMA2D output memory address register (DMA2D_OMAR)

Address offset: 0x003C

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]**: Memory Address

This field contains the address of the data used for the output FIFO. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only. The address alignment must match the image format selected e.g. a 32-bit per pixel format must be 32-bit aligned and a 16-bit per pixel format must be 16-bit aligned.

19.5.20 DMA2D output offset register (DMA2D_OOR)

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LO[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 **LO[15:0]**: Line offset

This field contains the line offset used for the output expressed in pixel when the LOM bit is reset and in byte when the LOM bit is set.

When expressed in pixels, only LO[13:0] is considered, LO[15:14] are ignored.

This value is used for the address generation. It is added at the end of each line to determine the starting address of the next line.

These bits can only be written when data transfers are disabled. Once data transfer has started, they become read-only.

19.5.21 DMA2D number of line register (DMA2D_NLR)

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.	PL[13:0]													
		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
NL[15:0]															
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 29:16 **PL[13:0]**: Pixel per lines

Number of pixels per lines of the area to be transferred. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

If any of the input image format is 4-bit per pixel, pixel per lines must be even.

Bits 15:0 **NL[15:0]**: Number of lines

Number of lines of the area to be transferred. These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

19.5.22 DMA2D line watermark register (DMA2D_LWR)

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
LW[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 **LW[15:0]**: Line watermark

These bits allow the configuration of the line watermark for interrupt generation.
An interrupt is raised when the last pixel of the watermarked line has been transferred.
These bits can only be written when data transfers are disabled. Once the transfer has started, they are read-only.

19.5.23 DMA2D AHB master timer configuration register (DMA2D_AMTCR)

Address offset: 0x004C

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DT[7:0]**: Dead time

Dead time value in the AHB clock cycle inserted between two consecutive accesses on the AHB master port. These bits represent the minimum guaranteed number of cycles between two consecutive AHB accesses.

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

Bit 0 **EN**: Enable

Enables the dead time functionality.

19.5.24 DMA2D foreground CLUT (DMA2D_FGCLUTx)

Address offset: 0x0400 + 0x4 * x, (x = 0 to 255)

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALPHA[7:0]								RED[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
GREEN[7:0]								BLUE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **ALPHA[7:0]**: Alpha
Alpha value for index {x} for the foreground

Bits 23:16 **RED[7:0]**: Red
Red value for index {x} for the foreground

Bits 15:8 **GREEN[7:0]**: Green
Green value for index {x} for the foreground

Bits 7:0 **BLUE[7:0]**: Blue
Blue value for index {x} for the foreground

19.5.25 DMA2D background CLUT (DMA2D_BGCLUTx)

Address offset: 0x0800 + 0x4 * x, (x = 0 to 255)

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALPHA[7:0]								RED[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
GREEN[7:0]								BLUE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **ALPHA[7:0]**:Alpha
Alpha value for index {x} for the background

Bits 23:16 **RED[7:0]**: Red
Red value for index {x} for the background

Bits 15:8 **GREEN[7:0]**: Green
Green value for index {x} for the background

Bits 7:0 **BLUE[7:0]**: Blue
Blue value for index {x} for the background

19.5.26 DMA2D register map

Table 152. DMA2D 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	DMA2D_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE [2:0]			Res.	Res.	CEIE	CTCIE	CAEIE	TWIE	TCIE	TEIE	Res.	LOM	Res.	Res.	Res.	ABORT	SUSP	START
	Reset value														0	0	0			0	0	0	0	0	0		0				0	0	0
0x0004	DMA2D_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.	CEIF	CTCIF	CAEIF	TWIF	TCIF	TEIF
	Reset value																											0	0	0	0	0	0
0x0008	DMA2D_IFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCEIF	CCTCIF	CAECIF	CTWIF	CTCIF	CTEIF
	Reset value																											0	0	0	0	0	0
0x000C	DMA2D_FGMAR	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
0x0010	DMA2D_FGOR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LO[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0014	DMA2D_BGMAR	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
0x0018	DMA2D_BGOR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LO[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x001C	DMA2D_FGPFCCR	ALPHA[7:0]								Res.	Res.	RBS	AI	Res.	Res.	AM[1:0]		CS[7:0]								Res.	Res.	START	CCM	CM[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
0x0020	DMA2D_FGCOLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RED[7:0]								GREEN[7:0]								BLUE[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
0x0024	DMA2D_BGPFCCR	ALPHA[7:0]								Res.	Res.	RBS	AI	Res.	Res.	AM[1:0]		CS[7:0]								Res.	Res.	START	CCM	CM[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
0x0028	DMA2D_BGCOLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RED[7:0]								GREEN[7:0]								BLUE[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
0x002C	DMA2D_FGCMAR	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
0x0030	DMA2D_BGCMAR	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
0x0034	DMA2D_OPFCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RBS	AI	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SB	Res.	Res.	Res.	Res.	Res.	Res.	CM[2:0]		
	Reset value											0	0											0							0	0	0
0x0038	DMA2D_OCOLR ARGB8888 or RGB888 color mode	ALPHA[7:0]								RED[7:0]								GREEN[7:0]								BLUE[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
0x0038	DMA2D_OCOLR RGB565 color mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RED[4:0]					GREEN[6:0]							BLUE[4:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 152. DMA2D 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
0x0038	DMA2D_OCOLR ARGB1555 color mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	A	RED[4:0]					GREEN[4:0]					BLUE[4:0]				
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0038	DMA2D_OCOLR ARGB4444 color mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALPHA[3:0]				RED[3:0]				GREEN[3:0]				BLUE[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x003C	DMA2D_OMAR	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
0x0040	DMA2D_OOR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LO[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	DMA2D_NLR	Res.	Res.	PL[13:0]														NL[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
0x0048	DMA2D_LWR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LW[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004C	DMA2D_AMTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
	Reset value																	0	0	0	0	0	0	0	0								0
0x0050-0x03FC	Reserved	Reserved																															
0x0400 + 0x4*x, (x=0 to 255)	DMA2D_FGCLUTx	ALPHA[7:0]								RED[7:0]								GREEN[7:0]								BLUE[7:0]							
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0800 + 0x4*x, (x=0 to 255)	DMA2D_BGCLUTx	ALPHA[7:0]								RED[7:0]								GREEN[7:0]								BLUE[7:0]							
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Refer to *Section 2.3* for the register boundary addresses.

20 Nested vectored interrupt controller (NVIC)

20.1 NVIC main features

- 125 maskable interrupt channels (not including the 16 Cortex-M33 with FPU interrupt lines)
- 16 programmable priority levels (4 bits of interrupt priority 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, enabling 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.

20.2 SysTick calibration value register

The Cortex-M33 with TrustZone mainline security extension embeds two SysTick timers.

When TrustZone is activated, the following SysTick timers are available:

- SysTick, secure instance
- SysTick, non-secure instance

When TrustZone is disabled, 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:

- When SysTick clock source is CPU clock HCLK
  reload value = (HCLK x STCALIB) - 1
- When SysTick clock source is external clock (HCLK/8)
  reload value = ((HCLK/8) x STCALIB) - 1

The HCLK refers to the 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

20.3 Interrupt and exception vectors

The grey rows in the table below describe the vectors without specific position.

Table 153. STM32U575/585 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. The RCC clock security system (CSS) is linked to the NMI vector.	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	PVD_PVM	Programmable voltage detector/peripheral voltage monitor	0x0000 0044
2	10	Settable	RTC	RTC global non-secure interrupts	0x0000 0048
3	11	Settable	RTC_S	RTC global secure interrupts	0x0000 004C
4	12	Settable	TAMP	Tamper global interrupts	0x0000 0050
5	13	Settable	RAMCFG	RAM configuration global interrupt	0x0000 0054
6	14	Settable	FLASH	Flash memory non-secure global interrupt	0x0000 0058
7	15	Settable	FLASH_S	Flash memory secure global interrupt	0x0000 005C
8	16	Settable	GTZC	GTZC1/GTZC2 global interrupt	0x0000 0060
9	17	Settable	RCC	RCC secure global interrupt	0x0000 0064
10	18	Settable	RCC_S	RCC secure global interrupt	0x0000 0068
11	19	Settable	EXTI0	EXTI Line0 interrupt	0x0000 006C

Table 153. STM32U575/585 vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
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 00A0
25	33	Settable	EXTI14	EXTI Line14 interrupt	0x0000 00A4
26	34	Settable	EXTI15	EXTI Line15 interrupt	0x0000 00A8
27	35	Settable	IWDG	Independent watchdog interrupt	0x0000 00AC
28	36	Settable	SAES	Secure AES	0x0000 00B0
29	37	Settable	GPDMA1_CH0	GPDMA1 channel 0 global interrupt	0x0000 00B4
30	38	Settable	GPDMA1_CH1	GPDMA1 channel 1 global interrupt	0x0000 00B8
31	39	Settable	GPDMA1_CH2	GPDMA1 channel 2 global interrupt	0x0000 00BC
32	40	Settable	GPDMA1_CH3	GPDMA1 channel 3 global interrupt	0x0000 00C0
33	41	Settable	GPDMA1_CH4	GPDMA1 channel 4 global interrupt	0x0000 00C4
34	42	Settable	GPDMA1_CH5	GPDMA1 channel 5 global interrupt	0x0000 00C8
35	43	Settable	GPDMA1_CH6	GPDMA1 channel 6 global interrupt	0x0000 00CC
36	44	Settable	GPDMA1_CH7	GPDMA1 channel 7 global interrupt	0x0000 00D0
37	45	Settable	ADC1	ADC1 (14 bits) global interrupt	0x0000 00D4
38	46	Settable	DAC1	DAC1 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_TERR TIM1_IERR	TIM1 break TIM1 transition error TIM1 index error	0x0000 00E4
42	50	Settable	TIM1_UP	TIM1 update	0x0000 00E8

Table 153. STM32U575/585 vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
43	51	Settable	TIM1_TRG_COM TIM1_DIR TIM1_IDX	TIM1 trigger and commutation TIM1 direction change interrupt TIM1 index	0x0000 00EC
44	52	Settable	TIM1_CC	TIM1 capture compare interrupt	0x0000 00F0
45	53	Settable	TIM2	TIM2 global interrupt	0x0000 00F4
46	54	Settable	TIM3	TIM3 global interrupt	0x0000 00F8
47	55	Settable	TIM4	TIM4 global interrupt	0x0000 00FC
48	56	Settable	TIM5	TIM5 global interrupt	0x0000 0100
49	57	Settable	TIM6	TIM6 global interrupt	0x0000 0104
50	58	Settable	TIM7	TIM7 global interrupt	0x0000 0108
51	59	Settable	TIM8_BRK TIM8_TERR TIM8_IERR	TIM8 break interrupt TIM8 transition error TIM8 index error	0x0000 010C
52	60	Settable	TIM8_UP	TIM8 update interrupt	0x0000 0110
53	61	Settable	TIM8_TRG_COM TIM8_DIR TIM8_IDX	TIM8 trigger and commutation interrupt TIM8 direction change interrupt TIM8 Index	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

Table 153. STM32U575/585 vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
72	80	Settable	COMP	COMP1/COMP2	0x0000 0160
73	81	Settable	OTG_FS	USB OTG FS global interrupt	0x0000 0164
74	82	Settable	CRS	Clock recovery system global interrupt	0x0000 0168
75	83	Settable	FMC	FSMC global interrupt	0x0000 016C
76	84	Settable	OCTOSPI1	OCTOSPI1 global interrupt	0x0000 0170
77	85	Settable	PWR_S3WU	PWR wakeup from Stop 3 interrupt	0x0000 0174
78	86	Settable	SDMMC1	SDMMC1 global interrupt	0x0000 0178
79	87	Settable	SDMMC2	SDMMC2 global interrupt	0x0000 017C
80	88	Settable	GPDMA1_CH8	GPDMA1 channel 8 interrupt	0x0000 0180
81	89	Settable	GPDMA1_CH9	GPDMA1 channel 9 interrupt	0x0000 0184
82	90	Settable	GPDMA1_CH10	GPDMA1 channel 10 interrupt	0x0000 0188
83	91	Settable	GPDMA1_CH11	GPDMA1 channel 11 interrupt	0x0000 018C
84	92	Settable	GPDMA1_CH12	GPDMA1 channel 12 interrupt	0x0000 0190
85	93	Settable	GPDMA1_CH13	GPDMA1 channel 13 interrupt	0x0000 0194
86	94	Settable	GPDMA1_CH14	GPDMA1 channel 14 interrupt	0x0000 0198
87	95	Settable	GPDMA1_CH15	GPDMA1 channel 15 interrupt	0x0000 019C
88	96	Settable	I2C3_EV	I2C3 event interrupt	0x0000 01A0
89	97	Settable	I2C3_ER	I2C3 error interrupt	0x0000 01A4
90	98	Settable	SAI1	SAI1 global interrupt	0x0000 01A8
91	99	Settable	SAI2	SAI2 global interrupt	0x0000 01AC
92	100	Settable	TSC	TSC global interrupt	0x0000 01B0
93	101	Settable	AES	AES global interrupt	0x0000 01B4
94	102	Settable	RNG	RNG global interrupt	0x0000 01B8
95	103	Settable	FPU	Floating point interrupt	0x0000 01BC
96	104	Settable	HASH	HASH interrupt	0x0000 01C0
97	105	Settable	PKA	PKA global interrupt	0x0000 01C4
98	106	Settable	LPTIM3	LPTIM3 global interrupt	0x0000 01C8
99	107	Settable	SPI3	SPI3 global interrupt	0x0000 01CC
100	108	Settable	I2C4_ER	I2C4 error interrupt	0x0000 01D0
101	109	Settable	I2C4_EV	I2C4 event interrupt	0x0000 01D4
102	110	Settable	MDF1_FLT0	MDF1 filter 0 global interrupt	0x0000 01D8
103	111	Settable	MDF1_FLT1	MDF1 filter 1 global interrupt	0x0000 01DC
104	112	Settable	MDF1_FLT2	MDF1 filter 2 global interrupt	0x0000 01E0

Table 153. STM32U575/585 vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
105	113	Settable	MDF1_FLT3	MDF1 filter 3 global interrupt	0x0000 01E4
106	114	Settable	UCPD1	UCPD1 global interrupt	0x0000 01E8
107	115	Settable	ICACHE	Instruction cache global interrupt	0x0000 01EC
108	116	Settable	OTFDEC1	OTFDEC1 secure global interrupt	0x0000 01F0
109	117	Settable	OTFDEC2	OTFDEC2 secure global interrupt	0x0000 01F4
110	118	Settable	LPTIM4	LPTIM4 global interrupt	0x0000 01F8
111	119	Settable	DCACHE1	Data cache global interrupt	0x0000 01FC
112	120	Settable	ADF1_FLT0	ADF1 filter 0 global interrupt	0x0000 0200
113	121	Settable	ADC4	ADC4 (12 bits) global interrupt	0x0000 0204
114	122	Settable	LPDMA1_CH0	LPDMA1 SmartRun channel 0 global interrupt	0x0000 0208
115	123	Settable	LPDMA1_CH1	LPDMA1 SmartRun channel 1 global interrupt	0x0000 020C
116	124	Settable	LPDMA1_CH2	LPDMA1 SmartRun channel 2 global interrupt	0x0000 0210
117	125	Settable	LPDMA1_CH3	LPDMA1 SmartRun channel 3 global interrupt	0x0000 0214
118	126	Settable	DMA2D	DMA2D global interrupt	0x0000 0218
119	127	Settable	DCMI_PSSI	DCMI/PSSI global interrupt	0x0000 021C
120	128	Settable	OCTOSPI2	OCTOSPI2 global interrupt	0x0000 0220
121	129	Settable	MDF1_FLT4	MDF1 filter 4 global interrupt	0x0000 0224
122	130	Settable	MDF1_FLT5	MDF1 filter 5 global interrupt	0x0000 0228
123	131	Settable	CORDIC	CORDIC interrupt	0x0000 022C
124	132	Settable	FMAC	FMAC interrupt	0x0000 0230
125	133	Settable	LSECSS(1) MSI_PLL_UNLOCK(1)	LSE CSS failure interrupt MSI PLL-mode unlock interrupt	0x0000 0234

1. Not available in STM32U575/585 rev. X devices. Available in all other STM32U575/585 revisions.

21 Extended interrupts and event controller (EXTI)

The extended interrupts and event controller (EXTI) manages the individual CPU and system wakeup through configurable 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 IO port selection.

21.1 EXTI main features

The EXTI main features are the following:

- Up to 24 input events supported
- All event inputs allow the possibility to wake up the system.
- Events that 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 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
- Software 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

21.2 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 88*.

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 88. EXTI block diagram

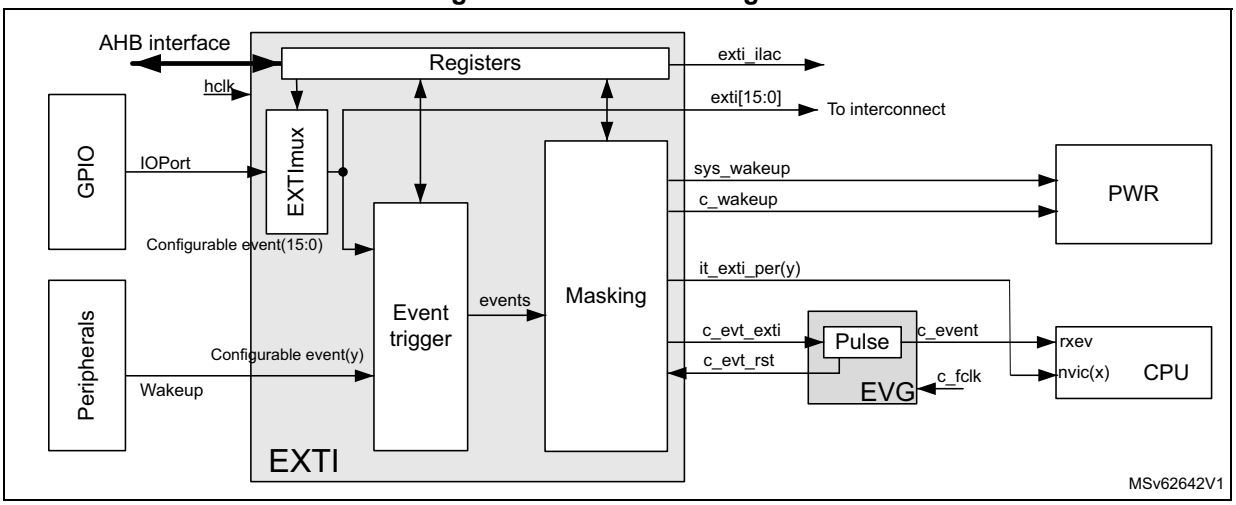


Table 154. EXTI signals

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 that do not have an associated interrupt and flag in the peripheral
exti_ilac	O	Illegal access event
IOPort(n)	I	GPIOs block IO ports[15:0]
exti[15:0]	O	EXTI GPIO output port to trigger other peripherals
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
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 155. EVG signals

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

21.2.1 EXTI connections between peripherals and CPU

Some peripherals able to generate wakeup or interrupt events when the system is in Stop mode, are connected to the EXTI.

- Peripheral wakeup signals that generate a pulse or 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 that requires to be cleared. It is the EXTI interrupt, associated with the status bit, that interrupts the CPU.
- All GPIO ports input to the EXTI multiplexer allow 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.

The dedicated EXTI/EVG CPU event is connected to the CPU rxev input.

The EXTI CPU wakeup signals are connected to the PWR and are used to wake up the system and the CPU sub-system bus clocks.

21.2.2 EXTI interrupt/event mapping

The EXTI lines are connected as shown in the table below.

Table 156. EXTI line connections

EXTI line	Line source	Line type
0-15	GPIO	Configurable
16	PVD output	Configurable
17	COMP1 output	Configurable
18	COMP2 output	Configurable
19	V_{DDUSB} voltage monitor	Configurable
20	V_{DDIO2} voltage monitor	Configurable
21	V_{DDA} voltage monitor 1	Configurable
22	V_{DDA} voltage monitor 2	Configurable
23(1)	LSECSSD(1) OR MSI_PLL_UNLOCK(1)	Configurable

1. Not available in STM32U575/585 rev. X devices. Available in all other STM32U575/585 revisions.

21.3 EXTI functional description

The events features are controlled from register bits as follows:

- Active trigger edge enable
  - by rising edge selection in the *EXTI rising trigger selection register (EXTI_RTSR1)*
  - by falling edge selection in the *EXTI falling trigger selection register (EXTI_FTSR1)*
- Software trigger in the *EXTI software interrupt event register (EXTI_SWIER1)*
- Interrupt pending flag in the

  EXTI rising edge pending register (EXTI_RPR1)

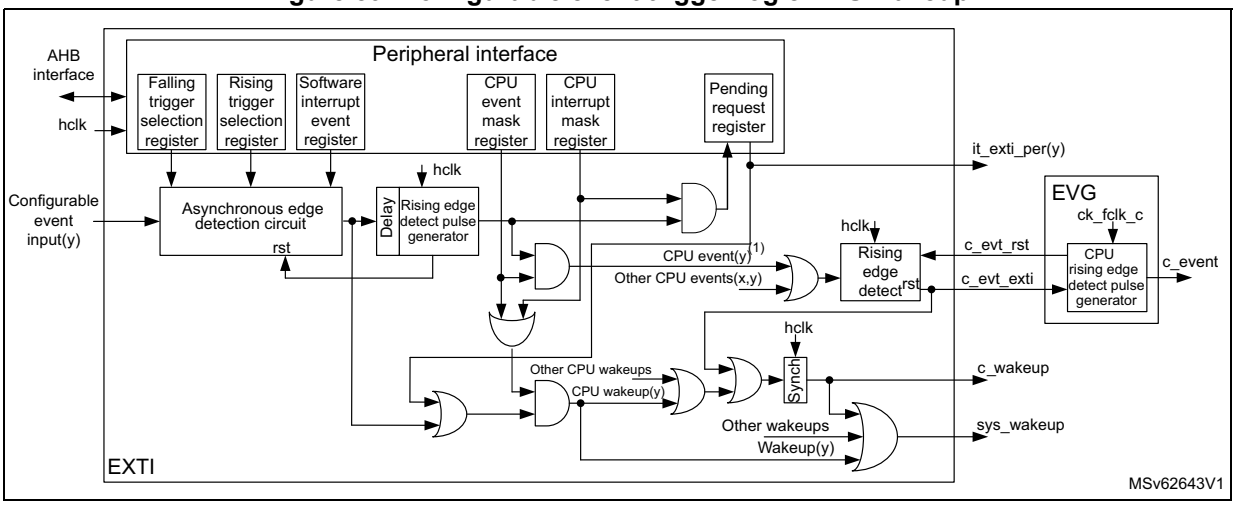
  EXTI falling edge pending register (EXTI_FPR1)

- CPU wakeup and interrupt enable in the
  EXTI CPU wakeup with interrupt mask register (EXTI_IMR1)
- CPU wakeup and event enable
  EXTI CPU wakeup with interrupt mask register (EXTI_IMR1)

21.3.1 EXTI configurable event input wakeup

The figure below is a detailed representation of the logic associated with configurable event inputs that wake up the CPU sub-system bus clocks and generate an EXTI pending flag and interrupt to the CPU, and/or a CPU wakeup event.

Figure 89. 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 configurable events to be triggered by software, writing the corresponding register bit, whatever the edge selection setting.

The configurable event active trigger edge (or both edges) is selected and enabled in the rising/falling edge selection registers.

The CPU has its dedicated wakeup (interrupt) mask register and a dedicated event mask registers. When the event is enabled, it is generated to the CPU. All events for the CPU are ORed together into a single CPU event signal. The event pending registers (EXTI_RPR and EXTI_FPR) are not set for an unmasked CPU event.

The configurable events have unique interrupt pending request registers. The pending register is only set for an unmasked interrupt. Each configurable event provides a common interrupt to the CPU. The configurable event interrupts must 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.

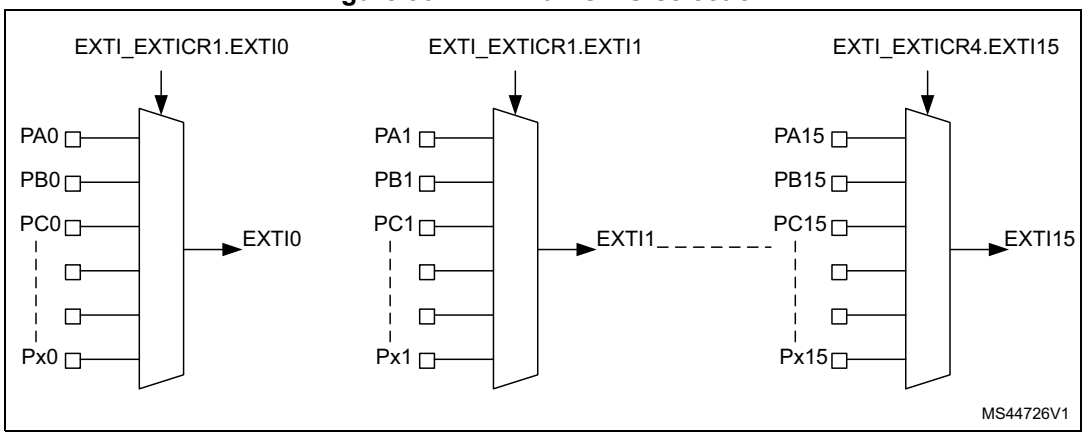
Note: A detected configurable event interrupt pending request can be cleared by the CPU with the correct access permission. The system is not able to enter into low-power modes as long as an interrupt pending request is active.

21.3.2 EXTI mux selection

The EXTI mux allows the selection of GPIOs as interrupts and wakeup. 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 in the *EXTI external interrupt selection register (EXTI_EXTICRm)*.

Figure 90. EXTI mux GPIO selection



The EXTI mux outputs are available as output signals from the EXTI to trigger other peripherals, whatever the masking in EXTI_IMR and EXTI_EMR registers.

21.4 EXTI functional behavior

The configurable events are enabled by enabling at least one of the trigger edges.

Once an event input is enabled, the CPU wakeup generation is conditioned by the CPU interrupt mask and CPU event mask.

Table 157. Masking functionality

CPU interrupt enable (in EXTI_IMR.IMn)	CPU event enable (in EXTI_EMR.EMn)	Configurable event inputs (in EXTI_RPR.RPIFn and 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 goes to the CPU. If no interrupt is required for CPU(m), the exti(n) interrupt must be masked in the CPU NVIC.
2. Only if CPU interrupt is enabled in EXTI_IMR.IMn.

For configurable event inputs, when the enabled edges occur on the event input, an event request is generated. When the associated CPU interrupt is unmasked, the corresponding pending bits EXTI_RPR.RPIFn and/or EXTI_FPR.FPIFn is/are set: the CPU sub-system is woken up and the CPU interrupt signal is activated. The EXTI_RPR.RPIFn and/or

EXTI_FPR.FPIFn pending bits must be cleared by software writing it to 1. This action clears the CPU interrupt.

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, whatever the setting in EXTI_RTSR.

21.5 EXTI event protection

The EXTI is able to protect event register bits from being modified by non-secure and unprivileged accesses. The protection is individually activated per input event via the register bits in EXTI_SECCFGR and EXTI_PRIVCFGR. At EXTI level, the protection consists in preventing the following unauthorized write access:

- 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 158. 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. Privilege can be bit-wise enabled in EXTI_PRIVCFGR.
EXTI_PRIVCFGR	RW	Always privilege. 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_SECCFGR register).
2. Privilege is enabled with the individual Input event (EXTI_PRIVCFGR register).

21.5.1 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.

21.5.2 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 privileged 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 privileged access and unprivileged access.

The privileged configuration in registers EXTI_PRIVCFGR can be globally locked after reset by EXTI_LOCKR.LOCK.

21.6 EXTI registers

The EXTI register map is divided in the following sections:

Table 159. EXTI register map sections

Address offset	Description
0x000 - 0x01C	General configurable event [22:0] configuration
0x060 - 0x06C	EXTI IO port 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.

21.6.1 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.	RT23	RT22	RT21	RT20	RT19	RT18	RT17	RT16
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:0 **RTx:** Rising trigger event configuration bit of configurable event input x[(1)] (x = 23 to 0)

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 and non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, RTx can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, RTx can only be accessed with privileged 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.

21.6.2 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.	FT23	FT22	FT21	FT20	FT19	FT18	FT17	FT16
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:0 **FTx:** Falling trigger event configuration bit of configurable event input x [(1)] (x = 23 to 0)

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 privileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, FTx can only be accessed with privileged 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.

21.6.3 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.	SWI23	SWI22	SWI21	SWI20	SWI19	SWI18	SWI17	SWI16
								rw	rw	rw	rw	rw	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 31:24 Reserved, must be kept at reset value.

Bits 23:0 **SWIx:** Software interrupt on event x (x = 23 to 0)

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 SWI x is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, SWIx can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, SWIx can only be accessed with privileged 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 1 triggers a rising edge event on event x. This bit is auto cleared by hardware.

21.6.4 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.	RPIF23	RPIF22	RPIF21	RPIF20	RPIF19	RPIF18	RPIF17	RPIF16
								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
RPIF15	RPIF14	RPIF13	RPIF12	RPIF11	RPIF10	RPIF9	RPIF8	RPIF7	RPIF6	RPIF5	RPIF4	RPIF3	RPIF2	RPIF1	RPIF0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:0 **RPIFx:** configurable event inputs x rising edge pending bit (x = 23 to 0)
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 privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, RPIFx can only be accessed with privileged 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 1 to it.

21.6.5 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.	FPIF23	FPIF22	FPIF21	FPIF20	FPIF19	FPIF18	FPIF17	FPIF16
								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
FPIF15	FPIF14	FPIF13	FPIF12	FPIF11	FPIF10	FPIF9	FPIF8	FPIF7	FPIF6	FPIF5	FPIF4	FPIF3	FPIF2	FPIF1	FPIF0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:0 **FPIFx:** configurable event inputs x falling edge pending bit (x = 23 to 0)
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 privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, FPIFx can only be accessed with privileged access. Unprivileged write to this FPIFx is discarded, unprivileged read returns 0.
0: No falling edge trigger request occurred
1: Falling 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 1 to it.

21.6.6 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16
								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 31:24 Reserved, must be kept at reset value.

Bits 23:0 **SECx:** Security enable on event input x (x = 23 to 0)

When EXTI_PRIVCFGR.PRIVx is disabled, SECx can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, SECx can only be written with privileged access. Unprivileged write to this SECx is discarded.
0: Event security disabled (non-secure)
1: Event security enabled (secure)

21.6.7 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16
								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 31:24 Reserved, must be kept at reset value.

Bits 23:0 **PRIVx:** Security enable on event input x (x = 23 to 0)
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)

21.6.8 EXTI external interrupt selection register (EXTI_EXTICRm)

Address offset: 0x060 + 0x004 * (m - 1) (m = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI{4*(m-1)+3}[7:0]								EXTI{4*(m-1)+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
EXTI{4*(m-1)+1}[7:0]								EXTI{4*(m-1)}[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI{4*(m-1)+3}[7:0]:** EXTI{4*(m-1)+3} GPIO port selection
These bits are written by software to select the source input for EXTI{4*(m-1)+3} external interrupt.
When EXTI_SECCFGR.SEC{4*(m-1)+3} is disabled, this field can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SEC{4*(m-1)+3} is enabled, this field can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIV{4*(m-1)+3} is disabled, this field can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV{4*(m-1)+3} is enabled, this field can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA[{4*(m-1)+3}] pin
0x01: PB[{4*(m-1)+3}] pin
0x02: PC[{4*(m-1)+3}] pin
0x03: PD[{4*(m-1)+3}] pin
0x04: PE[{4*(m-1)+3}] pin
0x05: PF[{4*(m-1)+3}] pin
0x06: PG[{4*(m-1)+3}] pin
0x07: PH[{4*(m-1)+3}] pin
0x08: PI[{4*(m-1)+3}] pin
Others: reserved

Bits 23:16 **EXTI{4*(m-1)+2}[7:0]:** EXTI{4*(m-1)+2} GPIO port selection
These bits are written by software to select the source input for EXTI{4*(m-1)+2} external interrupt.
When EXTI_SECCFGR.SEC{4*(m-1)+2} is disabled, this field can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SEC{4*(m-1)+2} is enabled, this field can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIV{4*(m-1)+2} is disabled, this field can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV{4*(m-1)+2} is enabled, this field can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA[{4*(m-1)+2}] pin
0x01: PB[{4*(m-1)+2}] pin
0x02: PC[{4*(m-1)+2}] pin
0x03: PD[{4*(m-1)+2}] pin
0x04: PE[{4*(m-1)+2}] pin
0x05: PF[{4*(m-1)+2}] pin
0x06: PG[{4*(m-1)+2}] pin
0x07: PH[{4*(m-1)+2}] pin
0x08: PI[{4*(m-1)+2}] pin
Others: reserved

Bits 15:8 **EXTI{4*(m-1)+1}[7:0]:** EXTI{4*(m-1)+1} GPIO port selection
These bits are written by software to select the source input for EXTI{4*(m-1)+1} external interrupt.
When EXTI_SECCFGR.SEC{4*(m-1)+1} is disabled, this field can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SEC{4*(m-1)+1} is enabled, this field can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIV{4*(m-1)+1} is disabled, this field can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV{4*(m-1)+1} is enabled, this field can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA[{4*(m-1)+1] pin
0x01: PB[{4*(m-1)+1] pin
0x02: PC[{4*(m-1)+1] pin
0x03: PD[{4*(m-1)+1] pin
0x04: PE[{4*(m-1)+1] pin
0x05: PF[{4*(m-1)+1] pin
0x06: PG[{4*(m-1)+1] pin
0x07: PH[{4*(m-1)+1] pin
0x08: PI[{4*(m-1)+1] pin
Others: reserved

Bits 7:0 **EXTI{4*(m-1)}[7:0]:** EXTI{4*(m-1)} GPIO port selection

These bits are written by software to select the source input for EXTI{4*(m-1)} external interrupt.

When EXTI_SECCFGR.SEC{4*(m-1)} is disabled, this field can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SEC{4*(m-1)} is enabled, this field can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.

When EXTI_PRIVCFGR.PRIV{4*(m-1)} is disabled, this field can be accessed with privileged and unprivileged access.

When EXTI_PRIVCFGR.PRIV{4*(m-1)} is enabled, this field can only be accessed with privilege access. Unprivileged write to this bit is discarded.

0x00: PA[{4*(m-1)}] pin
0x01: PB[{4*(m-1)}] pin
0x02: PC[{4*(m-1)}] pin
0x03: PD[{4*(m-1)}] pin
0x04: PE[{4*(m-1)}] pin
0x05: PF[{4*(m-1)}] pin
0x06: PG[{4*(m-1)}] pin
0x07: PH[{4*(m-1)}] pin
0x08: PI[{4*(m-1)}] pin
Others: reserved

21.6.9 EXTI lock register (EXTI_LOCKR)

Address offset: 0x070

Reset value: 0x0000 0000

This register provides write access security: a non-secure write access is ignored and a read access returns zero data, and both generates 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 bit is written once after reset.

0: Security and privilege configuration open, can be modified.

1: Security and privilege configuration locked, can no longer be modified.

21.6.10 EXTI CPU wakeup with interrupt mask register (EXTI_IMR1)

Address offset: 0x080

Reset value: 0x0000 0000

Contains 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.	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
								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 31:24 Reserved, must be kept at reset value.

Bits 23:0 **IMx:** CPU wakeup with interrupt mask on event input x [(1)] (x = 23 to 0)

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 and non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, IMx can be accessed with privileged and unprivileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, IMx can only be accessed with privileged 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.

21.6.11 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM23	EM22	EM21	EM20	EM19	EM18	EM17	EM16
								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 31:24 Reserved, must be kept at reset value.

Bits 23:0 **EMx:** CPU wakeup with event generation mask on event input x (x = 23 to 0)

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 and non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, EMx can be accessed with privileged and unprivileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, EMx can only be accessed with privileged 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.

21.6.12 EXTI register map

Table 160. EXTI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	EXTI_RTSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT23	RT22	RT21	RT20	RT19	RT18	RT17	RT16	RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	EXTI_FTSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT23	FT22	FT21	FT20	FT19	FT18	FT17	FT16	FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	EXTI_SWIER1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI23	SWI22	SWI21	SWI20	SWI19	SWI18	SWI17	SWI16	SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	EXTI_RPR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF23	RPIF22	RPIF21	RPIF20	RPIF19	RPIF18	RPIF17	RPIF16	RPIF15	RPIF14	RPIF13	RPIF12	RPIF11	RPIF10	RPIF9	RPIF8	RPIF7	RPIF6	RPIF5	RPIF4	RPIF3	RPIF2	RPIF1	RPIF0
	Reset value									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	EXTI_FPR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF23	FPIF22	FPIF21	FPIF20	FPIF19	FPIF18	FPIF17	FPIF16	FPIF15	FPIF14	FPIF13	FPIF12	FPIF11	FPIF10	FPIF9	FPIF8	FPIF7	FPIF6	FPIF5	FPIF4	FPIF3	FPIF2	FPIF1	FPIF0
	Reset value									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	EXTI_ SECCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value									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_ PRIVCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV22	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020-0x05C	Reserved	Reserved																															
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

Table 160. EXTI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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
0x074-0x07C	Reserved	Reserved																															
0x080	EXTI_IMR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16	IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	EXTI_EMR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM23	EM22	EM21	EM20	EM19	EM18	EM17	EM16	EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

22 Cyclic redundancy check calculation unit (CRC)

22.1 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.

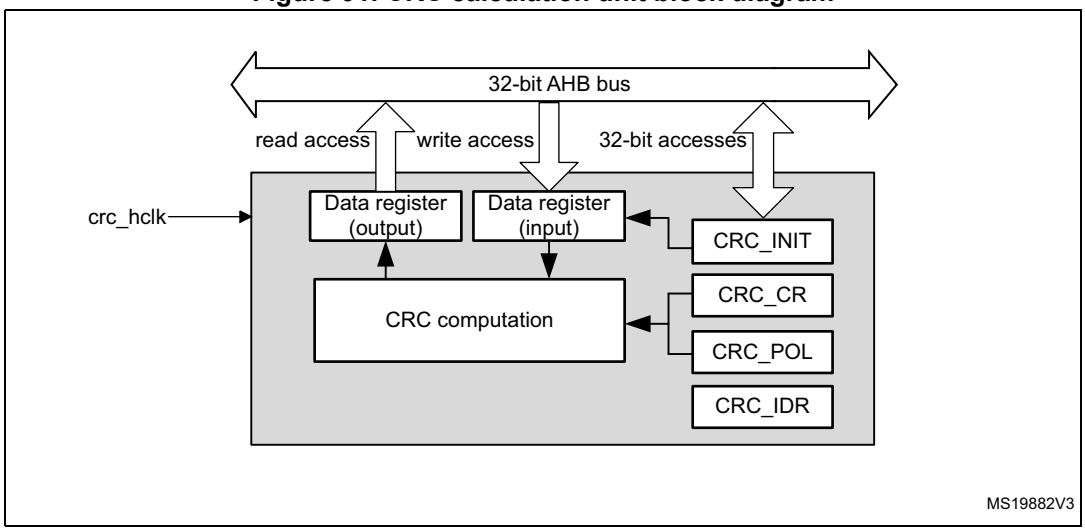
22.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
  $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Alternatively, uses fully programmable polynomial with programmable size (7, 8, 16, 32 bits)
- Handles 8-,16-, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (can be used for temporary storage)
- Reversibility option on I/O data
- Accessed through AHB slave peripheral by 32-bit words only, with the exception of CRC_DR register that can be accessed by words, right-aligned half-words and right-aligned bytes

22.3 CRC functional description

22.3.1 CRC block diagram

Figure 91. CRC calculation unit block diagram



22.3.2 CRC internal signals

Table 161. CRC internal input/output signals

Signal name	Signal type	Description
crc_hclk	Digital input	AHB clock

22.3.3 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 bits
- 2 AHB clock cycles for 16 bits
- 1 AHB clock cycles for 8 bits

An input buffer allows a second data to be immediately written without waiting for any wait states due to the previous CRC calculation.

The data size can be dynamically adjusted to minimize the number of write accesses for a given number of bytes. For instance, a CRC for 5 bytes can be computed with a word write followed by a byte write.

The input data can be reversed, to manage the various endianness schemes. The reversing operation can be performed on 8 bits, 16 bits and 32 bits depending on the REV_IN[1:0] bits in the CRC_CR register.

For example: 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.

Note: The type of an even polynomial is $X+X^2+..+X^n$, while the type of an odd polynomial is $1+X+X^2+..+X^n$.

If the CRC data is less than 32-bit, its value can be read from the least significant bits of the CRC_DR register.

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size can not be performed during a CRC calculation. As a result, if a CRC calculation is ongoing, the application must either reset it or perform a CRC_DR read before changing the polynomial.

The default polynomial value is the CRC-32 (Ethernet) polynomial: 0x4C11DB7.

22.4 CRC registers

The CRC_DR register can be accessed by words, right-aligned half-words and right-aligned bytes. For the other registers only 32-bit accesses are allowed.

22.4.1 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.

22.4.2 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

22.4.3 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

22.4.4 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.

22.4.5 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.

22.4.6 CRC register map

Table 162. 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 115* for the register boundary addresses.

23 CORDIC co-processor (CORDIC)

23.1 CORDIC introduction

The CORDIC co-processor provides hardware acceleration of mathematical functions (mainly trigonometric ones) commonly used in motor control, metering, signal processing and many other applications.

It speeds up the calculation of these functions compared to a software implementation, making it possible the use of a lower operating frequency, or freeing up processor cycles in order to perform other tasks.

23.2 CORDIC main features

- 24-bit CORDIC rotation engine
- Circular and Hyperbolic modes
- Rotation and Vectoring modes
- Functions: sine, cosine, sinh, cosh, atan, atan2, atanh, modulus, square root, natural logarithm
- Programmable precision
- Low latency AHB slave interface
- Results can be read as soon as ready, without polling or interrupt
- DMA read and write channels
- Multiple register read/write by DMA

23.3 CORDIC functional description

23.3.1 General description

The CORDIC is a cost-efficient successive approximation algorithm for evaluating trigonometric and hyperbolic functions.

In trigonometric (circular) mode, the sine and cosine of an angle θ are determined by rotating the unit vector [1, 0] through decreasing angles until the cumulative sum of the rotation angles equals the input angle θ. The x and y cartesian components of the rotated vector then correspond, respectively, to the cosine and sine of θ. Inversely, the angle of a vector [x, y] corresponding to arctangent (y / x), is determined by rotating [x, y] through successively decreasing angles to obtain the unit vector [1, 0]. The cumulative sum of the rotation angles gives the angle of the original vector.

The CORDIC algorithm can also be used for calculating hyperbolic functions (sinh, cosh, atanh), by replacing the successive circular rotations by steps along a hyperbole.

Other functions can be derived from the basic functions described above.

23.3.2 CORDIC functions

The first step when using the co-processor is to select the required function, by programming the FUNC field of the CORDIC_CR register accordingly.

Table 163 lists the functions supported by the CORDIC co-processor.

Table 163. CORDIC functions

Function	Primary argument (ARG1)	Secondary argument (ARG2)	Primary result (RES1)	Secondary result (RES2)
Cosine	angle θ	modulus m	$m \cdot \cos\theta$	$m \cdot \sin\theta$
Sine	angle θ	modulus m	$m \cdot \sin\theta$	$m \cdot \cos\theta$
Phase	x	y	atan2(y,x)	$\sqrt{x^2+y^2}$
Modulus	x	y	$\sqrt{x^2+y^2}$	atan2(y,x)
Arctangent	x	none	$\tan^{-1} x$	none
Hyperbolic cosine	x	none	cosh x	sinh x
Hyperbolic sine	x	none	sinh x	cosh x
Hyperbolic arctangent	x	none	$\tanh^{-1} x$	none
Natural logarithm	x	none	ln x	none
Square root	x	none	$\sqrt{x}$	none

Several functions take two input arguments (ARG1 and ARG2) and some generate two results (RES1 and RES2) simultaneously. This is a side-effect of the algorithm and means that only one operation is needed to obtain two values. This is the case, for example, when performing polar-to-rectangular conversion: sin θ also generates cos θ, cos θ also generates sin θ. Similarly for rectangular-to-polar conversion (phase(x,y), modulus(x,y)) and for hyperbolic functions (cosh θ, sinh θ).

Note: The exponential function, exp x, can be obtained as the sum of sinh x and cosh x. Furthermore, base N logarithms, $\log_N x$, can be derived by multiplying ln x by a constant K, where K = 1/ln N.

For certain functions (atan, log, sqrt) a scaling factor (see *Section 23.3.4*) can be applied to extend the range of the function beyond the maximum [-1, 1] supported by the q1.31 fixed point format. The scaling factor must be set to 0 for all other circular functions, and to 1 for hyperbolic functions.

Cosine

Table 164. Cosine parameters

Parameter	Description	Range
ARG1	Angle θ in radians, divided by π	[-1, 1]
ARG2	Modulus m	[0, 1]
RES1	m · cos θ	[-1, 1]

Table 164. Cosine parameters (continued)

Parameter	Description	Range
RES2	m · sin θ	[-1, 1]
SCALE	Not applicable	0

This function calculates the cosine of an angle in the range -π to π. It can also be used to perform polar to rectangular conversion.

The primary argument is the angle θ in radians. It must be divided by π before programming ARG1.

The secondary argument is the modulus m. If m is greater than 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the cosine of the angle, multiplied by the modulus.

The secondary result, RES2, is the sine of the angle, multiplied by the modulus.

Sine

Table 165. Sine parameters

Parameter	Description	Range
ARG1	Angle θ in radians, divided by π	[-1, 1]
ARG2	Modulus m	[0, 1]
RES1	m · sin θ	[-1, 1]
RES2	m · cos θ	[-1, 1]
SCALE	Not applicable	0

This function calculates the sine of an angle in the range -π to π. It can also be used to perform polar to rectangular conversion.

The primary argument is the angle θ in radians. It must be divided by π before programming ARG1.

The secondary argument is the modulus m. If m is greater than 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the sine of the angle, multiplied by the modulus.

The secondary result, RES2, is the cosine of the angle, multiplied by the modulus.

Phase

Table 166. Phase parameters

Parameter	Description	Range
ARG1	x coordinate	[-1, 1]
ARG2	y coordinate	[-1, 1]
RES1	Phase angle θ in radians, divided by π	[-1, 1]

Table 166. Phase parameters (continued)

Parameter	Description	Range
RES2	Modulus m	[0, 1]
SCALE	Not applicable	0

This function calculates the phase angle in the range -π to π of a vector **v** = [x y] (also known as atan2(y,x). It can also be used to perform rectangular to polar conversion.

The primary argument is the x coordinate, that is, the magnitude of the vector in the direction of the x axis. If |x| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG1.

The secondary argument is the y coordinate, that is, the magnitude of the vector in the direction of the y axis. If |y| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the phase angle θ of the vector **v**. RES1 must be multiplied by π to obtain the angle in radians. Note that values close to π may sometimes wrap to -π due to the circular nature of the phase angle.

The secondary result, RES2, is the modulus, given by: $|v| = \sqrt{x^2 + y^2}$. If |**v**| > 1 the result in RES2 is saturated to 1.

Modulus

Table 167. Modulus parameters

Parameter	Description	Range
ARG1	x coordinate	[-1, 1]
ARG2	y coordinate	[-1, 1]
RES1	Modulus m	[0, 1]
RES2	Phase angle θ	[-1, 1]
SCALE	Not applicable	0

This function calculates the magnitude, or modulus, of a vector **v** = [x y]. It can also be used to perform rectangular to polar conversion.

The primary argument is the x coordinate, that is, the magnitude of the vector in the direction of the x axis. If |x| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG1.

The secondary argument is the y coordinate, that is, the magnitude of the vector in the direction of the y axis. If |y| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the modulus, given by: $|v| = \sqrt{x^2 + y^2}$. If |**v**| > 1 the result in RES1 is saturated to 1.

The secondary result, RES2, is the phase angle θ of the vector **v**. RES2 must be multiplied by π to obtain the angle in radians. Note that values close to π may sometimes wrap to -π due to the circular nature of the phase angle.

Arctangent

Table 168. Arctangent parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-1, 1]
ARG2	Not applicable	-
RES1	$2^{-n} \cdot \tan^{-1} x$, in radians, divided by p	[-1, 1]
RES2	Not applicable	-
SCALE	n	[0 7]

This function calculates the arctangent, or inverse tangent, of the input argument x.

The primary argument, ARG1, is the input value, $x = \tan \theta$. If $|x| > 1$, a scaling factor of 2^{-n} must be applied in software such that $-1 < x \cdot 2^{-n} < 1$. The scaled value $x \cdot 2^{-n}$ is programmed in ARG1 and the scale factor n must be programmed in the SCALE parameter.

Note that the maximum input value allowed is $\tan \theta = 128$, which corresponds to an angle $\theta = 89.55$ degrees. For $|x| > 128$, a software method must be used to find $\tan^{-1} x$.

The secondary argument, ARG2, is unused.

The primary result, RES1, is the angle $\theta = \tan^{-1} x$. RES1 must be multiplied by $2^n \cdot \pi$ to obtain the angle in radians.

The secondary result, RES2, is unused.

Hyperbolic cosine

Table 169. Hyperbolic cosine parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-0.559 0.559]
ARG2	Not applicable	-
RES1	$2^{-n} \cdot \cosh x$	[0.5 0.846]
RES2	$2^{-n} \cdot \sinh x$	[-0.683 0.683]
SCALE	n	1

This function calculates the hyperbolic cosine of a hyperbolic angle x. It can also be used to calculate the exponential functions $e^x = \cosh x + \sinh x$, and $e^{-x} = \cosh x - \sinh x$.

The primary argument is the hyperbolic angle x. Only values of x in the range -1.118 to +1.118 are supported. Since the minimum value of cosh x is 1, which is beyond the range of the q1.31 format, a scaling factor of 2^{-n} must be applied in software. The factor n = 1 must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result, RES1, is the hyperbolic cosine, cosh x. RES1 must be multiplied by 2 to obtain the correct result.

The secondary result, RES2, is the hyperbolic sine, sinh x. RES2 must be multiplied by 2 to obtain the correct result.

Hyperbolic sine

Table 170. Hyperbolic sine parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-0.559, 0.559]
ARG2	Not applicable	-
RES1	$2^{-n} \cdot \sinh x$	[-0.683, 0.683]
RES2	$2^{-n} \cdot \cosh x$	[0.5, 0.846]
SCALE	n	1

This function calculates the hyperbolic sine of a hyperbolic angle x. It can also be used to calculate the exponential functions e^x = cosh x + sinh x, and e^{-x} = cosh x- sinh x.

The primary argument is the hyperbolic angle x. Only values of x in the range -1.118 to +1.118 are supported. For all input values, a scaling factor of 2^{-n} must be applied in software, where n = 1. The scaled value x · 0.5 is programmed in ARG1 and the factor n = 1 must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result, RES1, is the hyperbolic sine, sinh x. RES1 must be multiplied by 2 to obtain the correct result.

The secondary result, RES2, is the hyperbolic cosine, cosh x. RES2 must be multiplied by 2 to obtain the correct result.

Hyperbolic arctangent

Table 171. Hyperbolic arctangent parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-0.403 0.403]
ARG2	Not applicable	-
RES1	$2^{-n} \cdot \text{atanh } x$	[-0.559 0.559]
RES2	Not applicable	-
SCALE	n	1

This function calculates the hyperbolic arctangent of the input argument x.

The primary argument is the input value x. Only values of x in the range -0.806 to +0.806 are supported. The value x must be scaled by a factor 2^{-n}, where n = 1. The scaled value

$x \cdot 0.5$ is programmed in ARG1 and the factor n = 1 must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result is the hyperbolic arctangent, atanh x. RES1 must be multiplied by 2 to obtain the correct value.

The secondary result is not used.

Natural logarithm

Table 172. Natural logarithm parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[0.054 0.875]
ARG2	Not applicable	-
RES1	$2^{-(n+1)}.\ln x$	[-0.279 0.137]
RES2	Not applicable	-
SCALE	n	[1 4]

This function calculates the natural logarithm of the input argument x.

The primary argument is the input value x. Only values of x in the range 0.107 to 9.35 are supported. The value x must be scaled by a factor 2^{-n}, such that $x \cdot 2^{-n} < 1-2^{-n}$. The scaled value $x \cdot 2^{-n}$ is programmed in ARG1 and the factor n must be programmed in the SCALE parameter.

Table 173 lists the valid scaling factors, n, and the corresponding ranges of x and ARG1.

Table 173. Natural log scaling factors and corresponding ranges

n	x range	ARG1 range
1	$0.107 \leq x < 1$	$0.0535 \leq ARG1 < 0.5$
2	$1 \leq x < 3$	$0.25 \leq ARG1 < 0.75$
3	$3 \leq x < 7$	$0.375 \leq ARG1 < 0.875$
4	$7 \leq x \leq 9.35$	$0.4375 \leq ARG1 < 0.584$

The secondary argument is not used.

The primary result is the natural logarithm, ln x. RES1 must be multiplied by $2^{(n+1)}$ to obtain the correct value.

The secondary result is not used.

Square root

Table 174. Square root parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[0.027 0.875]
ARG2	Not applicable	-
RES1	$2^{-n}\sqrt{x}$	[0.04 1]
RES2	Not applicable	-
SCALE	n	[0 2]

This function calculates the square root of the input argument x.

The primary argument is the input value x. Only values of x in the range 0.027 to 2.34 are supported. The value x must be scaled by a factor 2^{-n}, such that $x \cdot 2^{-n} < (1 - 2^{(-n-2)})$.

The scaled value $x \cdot 2^{-n}$ is programmed in ARG1 and the factor n must be programmed in the SCALE parameter.

Table 175 lists the valid scaling factors, n, and the corresponding ranges of x and ARG1.

Table 175. Square root scaling factors and corresponding ranges

n	x range	ARG1 range
0	$0.027 \leq x < 0.75$	0.027 ≤ ARG1 < 0.75
1	$0.75 \leq x < 1.75$	0.375 ≤ ARG1 < 0.875
2	$1.75 \leq x \leq 2.341$	0.4375 ≤ ARG1 ≤ 0.585

The secondary argument is not used.

The primary result is the square root of x. RES1 must be multiplied by 2^n to obtain the correct value.

The secondary result is not used.

23.3.3 Fixed point representation

The CORDIC operates in fixed point signed integer format. Input and output values can be either q1.31 or q1.15.

In q1.31 format, numbers are represented by one sign bit and 31 fractional bits (binary decimal places). The numeric range is therefore -1 (0x80000000) to $1 - 2^{-31}$ (0x7FFFFFFF).

In q1.15 format, the numeric range is 1 (0x8000) to $1 - 2^{-15}$ (0x7FFF). This format has the advantage that two input arguments can be packed into a single 32-bit write, and two results can be fetched in one 32-bit read.

23.3.4 Scaling factor

Several of the functions listed in *Section 23.3.2* specify a scaling factor, SCALE. This allows the function input range to be extended to cover the full range of values supported by the CORDIC, without saturating the input, output or internal registers. If the scaling factor is

required, it has to be calculated in software and programmed into the SCALE field of the CORDIC_CSR register. The input arguments must be scaled accordingly before programming the scaled values in the CORDIC_WDATA register. The scaling must also be undone on the results read from the CORDIC_RDATA register.

Note: The scaling factor entails a loss of precision due to truncation of the scaled value.

23.3.5 Precision

The precision of the result is dependent on the number of CORDIC iterations. The algorithm converges at a constant rate of one binary digit per iteration for trigonometric functions (sine, cosine, phase, modulus), see *Figure 92*.

For hyperbolic functions (hyperbolic sine, hyperbolic cosine, natural logarithm), the convergence rate is less constant due to the peculiarities of the CORDIC algorithm (see *Figure 93*). The square root function converges at roughly twice the speed of the hyperbolic functions (see *Figure 94*).

Figure 92. CORDIC convergence for trigonometric functions

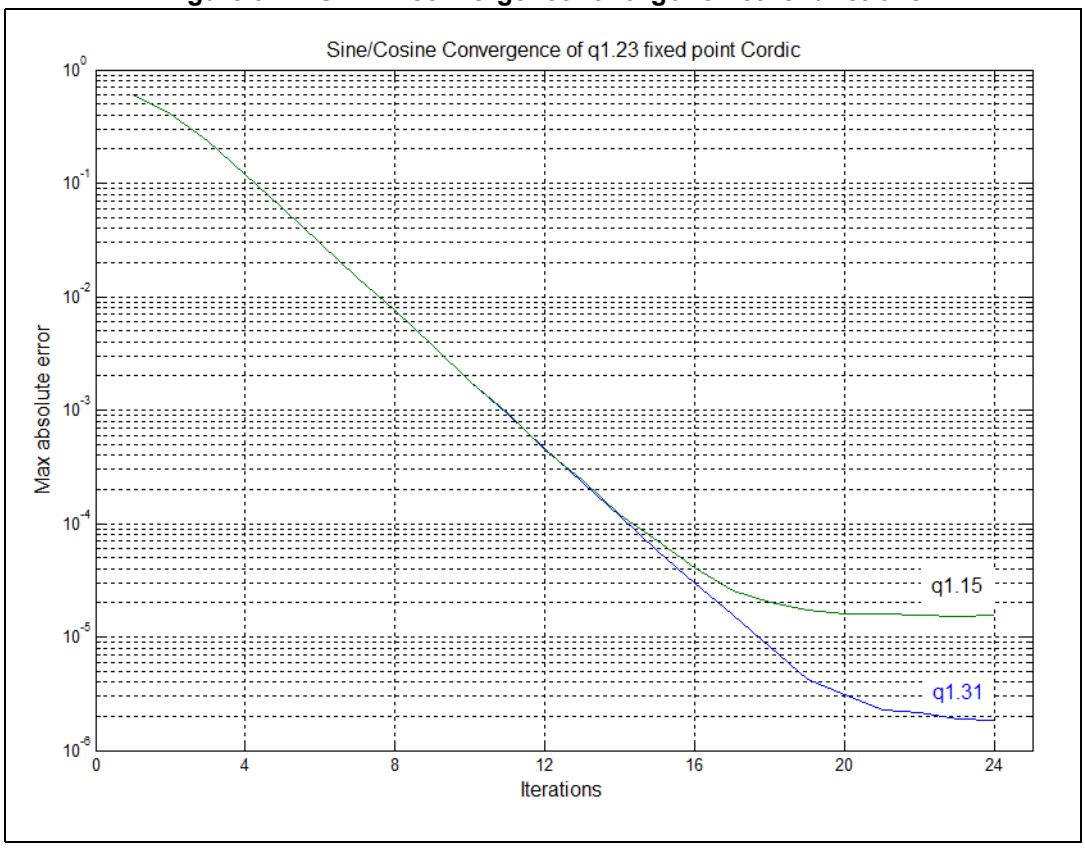


Figure 93. CORDIC convergence for hyperbolic functions

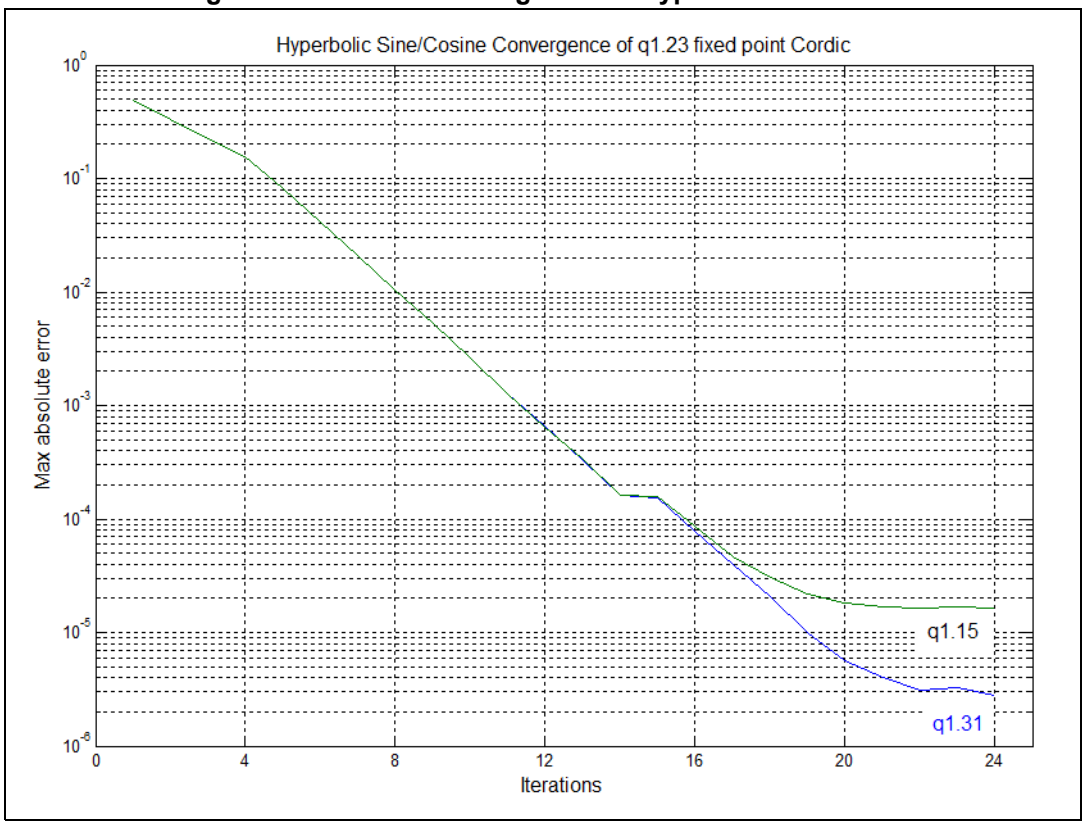
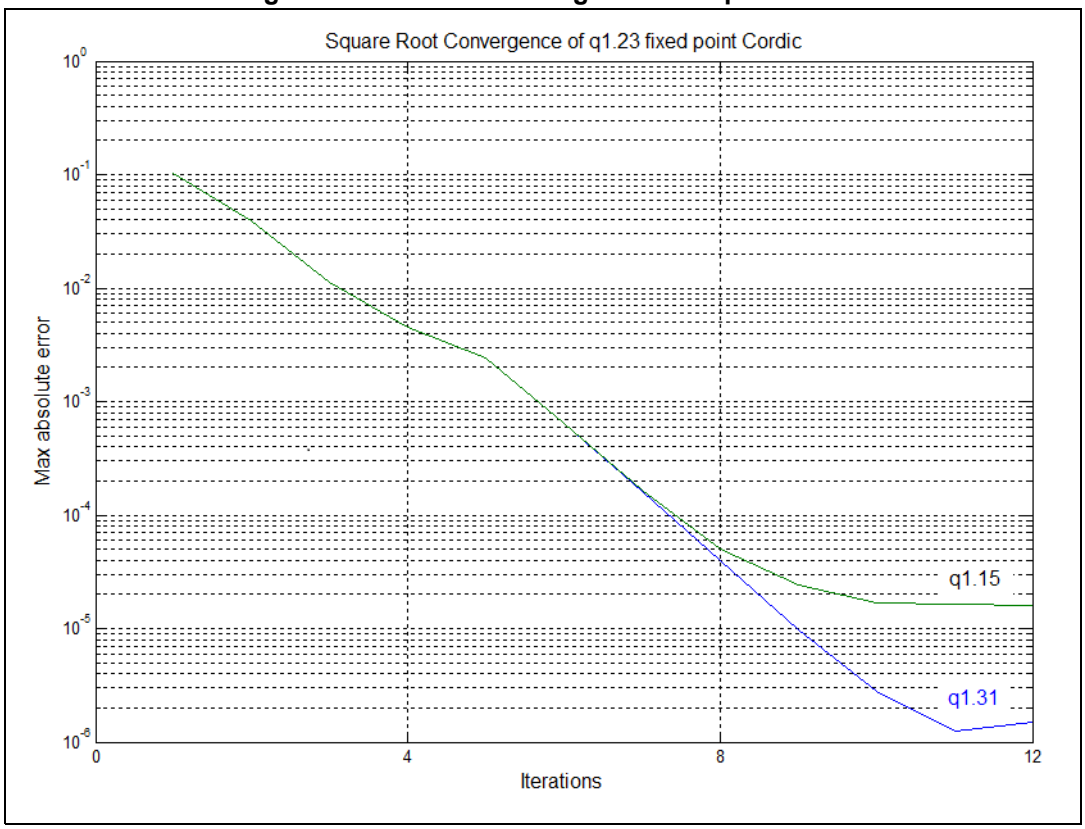


Figure 94. CORDIC convergence for square root



Note: The convergence rate decreases as the quantization error starts to become significant.

The CORDIC can perform four iterations per clock cycle. For each function, the maximum error remaining after every four iterations is shown in *Table 176*, together with the number of clock cycles required to reach that precision. From this table, the desired number of cycles can be determined and programmed in the PRECISION field of the CORDIC_CR register. The co-processor stops as soon as the programmed number of iterations has completed, and the result can be read immediately.

Table 176. Precision vs. number of iterations

Function	Number of iterations	Number of cycles	Max residual error(1)	
			q1.31 format	q1.15 format
Sin, Cos, Phase(2), Mod, Atan(4)	4	1	2^{-3}	2^{-3}
	8	2	2^{-7}	2^{-7}
	12	3	2^{-11}	2^{-11}
	16	4	2^{-15}	2^{-15}
	20	5	2^{-18}	2^{-16}
	24	6	2^{-19}	2^{-16}

Table 176. Precision vs. number of iterations (continued)

Function	Number of iterations	Number of cycles	Max residual error(1)	
			q1.31 format	q1.15 format
Sinh, Cosh, Atanh, Ln(3)	4	1	2^{-2}	2^{-2}
	8	2	2^{-6}	2^{-6}
	12	3	2^{-10}	2^{-10}
	16	4	2^{-13}	2^{-13}
	20	5	2^{-17}	2^{-15}
	24	6	2^{-18}	2^{-15}
Sqrt(4)	4	1	2^{-7}	2^{-7}
	8	2	2^{-14}	2^{-14}
	12	3	2^{-19}	2^{-15}

1. Max residual error is the maximum error remaining after the given number of iterations, compared to the identical calculation performed in double precision floating point. An additional rounding error may be incurred, of up to 2^{-16} for q15 format or 2^{-20} for q31 format.
2. For modulus > 0.5. The achievable precision reduces proportionally to the magnitude of the modulus, as quantization error becomes significant.
3. SCALE = 1. If a higher scaling factor is used, the achievable precision is reduced proportionally.
4. SCALE = 0. If a higher scaling factor is used, the achievable precision is reduced proportionally.

23.3.6 Zero-overhead mode

The fastest way to use the co-processor is to pre-program the CORDIC_CSR register with the function to be performed (FUNC), the desired number of clock cycles (PRECISION), the size of the input and output values (ARGSIZE, RESSIZE), the number of input arguments (NARGS) and/or results (NRES), and the scaling factor (SCALE), if applicable.

Subsequently, a calculation is triggered by writing the input arguments to the CORDIC_WDATA register. As soon as the correct number of input arguments has been written (and any ongoing calculation has finished) a new calculation is launched using these input arguments and the current CORDIC_CSR settings. There is no need to re-program the CORDIC_CSR register if there is no change.

If a dual 32-bit input argument is needed (ARGSIZE = 0, NARGS = 1), the primary input argument, ARG1, must be written first, followed by the secondary argument, ARG2. If the secondary argument remains unchanged for a series of calculations, the second write can be avoided, by reprogramming the number of arguments to one (NARGS = 0), once the first calculation has started. The secondary argument retains its programmed value as long as the function is not changed.

Note: ARG2 is set to +1 (0x7FFFFFFF) after a reset.

If two 16-bit arguments are used (ARGSIZE = 1) they must be packed into a 32-bit word, with ARG1 in the least significant half-word and ARG2 in the most significant half-word. The packed 32-bit word is then written to the CORDIC_WDATA register. Only one write is needed in this case (NARGS = 0).

For functions taking only one input argument, ARG1, it is recommended to set NARGS = 0. If NARGS = 1, a second write to CORDIC_WDATA must be performed to trigger the calculation. The ARG2 data in this case is not used.

Once the calculation starts, any attempt to read the CORDIC_RDATA register inserts bus wait states until the calculation is completed, before returning the result. Hence it is possible for the software to write the input and immediately read the result without polling to see if it is valid. Alternatively, the processor can wait for the appropriate number of clock cycles before reading the result. This time can be used to program the CORDIC_CSR register for the next calculation and prepare the next input data, if needed. The CORDIC_CSR register can be re-programmed while a calculation is in progress, without affecting the result of the ongoing calculation. In the same way, the CORDIC_WDATA register can be updated with the next argument(s) once the previous arguments have been taken into account. The next arguments and settings remain pending until the previous calculation has completed.

When a calculation is finished, the result(s) can be read from the CORDIC_RDATA register. If two 32-bit results are expected (NRES = 1, RESSIZE = 0), the primary result (RES1) is read out first, followed by the secondary result (RES2). If only one 32-bit result is expected (NRES = 0, RESSIZE = 0), then RES1 is output on the first read.

If 16-bit results are expected (RESSIZE = 1), a single read to CORDIC_RDATA fetches both results packed into a 32-bit word. RES1 is in the lower half-word, and RES2 in the upper half-word. In this case, it is recommended to program NRES = 0. IF NRES = 1, a second read of CORDIC_RDATA must be performed in order to free up the CORDIC for the next operation. The data from this second read must be discarded.

The next calculation starts when the expected number of results has been read, provided the expected number of arguments has been written. This means that at any time, there can be one calculation in progress, or waiting for the results to be read, and one operation pending. Any further access to CORDIC_WDATA while an operation is pending, cancels the pending operation and overwrite the data.

The following sequence summarizes the use of the CORDIC_IP in zero-overhead mode:

1. Program the CORDIC_CSR register with the appropriate settings
2. Program the argument(s) for the first calculation in the CORDIC_WDATA register. This launches the first calculation.
3. If needed, update the CORDIC_CSR register settings for the next calculation.
4. Program the argument(s) for the next calculation in the CORDIC_WDATA register.
5. Read the result(s) from the CORDIC_RDATA register. This triggers the next calculation.
6. Go to step *3*.

23.3.7 Polling mode

When a new result is available in the CORDIC_RDATA register, the RRDY flag is set in the CORDIC_CSR register. The flag can be polled by reading the register. It is reset by reading the CORDIC_RDATA register (once or twice depending on the NRES field of the CORDIC_CSR register).

Polling the RRDY flag takes slightly longer than reading the CORDIC_RDATA register directly, since the result is not read as soon as it is available. However the processor and bus interface are not stalled while reading the CORDIC_CSR register, so this mode may be of interest if stalling the processor is not acceptable (e.g. if low latency interrupts must be serviced).

23.3.8 Interrupt mode

By setting the interrupt enable (IE) bit in the CORDIC_CSR register, an interrupt is generated whenever the RRDY flag is set. The interrupt is cleared when the flag is reset.

This mode allows the result of the calculation to be read under interrupt service routine, and hence given a priority relative to other tasks. However it is slower than directly reading the result, or polling the flag, due to the interrupt handling delays.

23.3.9 DMA mode

If the DMA write enable (DMAWEN) bit is set in the CORDIC_CSR register, and no operation is pending, a DMA write channel request is generated. The DMA controller can transfer a primary input argument (ARG1) from memory into the CORDIC_WDATA register. Writing into the register deasserts the DMA request. If NARGS = 1 in the CORDIC_CSR register, a second DMA write channel request is generated to transfer the secondary input argument (ARG2) into the CORDIC_WDATA register. When all input arguments have been written, and any ongoing calculation has been completed (by reading the results), a new calculation is started and another DMA write channel request is generated.

If the DMA read enable (DMAREN) bit is set in the CORDIC_CSR register, the RRDY flag going active generates a DMA read channel request. The DMA controller can then transfer the primary result (RES1) from the CORDIC_RDATA register to memory. Reading the register deasserts the DMA request. If NRES = 1 in the CORDIC_CSR register, a second DMA request is generated to read out the secondary result (RES2). When all results have been read, the RRDY flag is deasserted.

The DMA read and write channels can be enabled separately. If both channels are enabled, the CORDIC can autonomously perform repeated calculations on a buffer of data without processor intervention.This allows the processor to perform other tasks. The DMA controller is operating in memory-to-peripheral mode for the write channel, and peripheral-to-memory mode for the read channel. Note that the sequence is started by the processor setting the DMAWEN flag. Thereafter the DMA read and write requests are generated as fast as the CORDIC can process the data.

In some cases, the input data may be stored in memory, and the output is transferred at regular intervals to another peripheral, such as a digital-to-analog converter. In this case, the destination peripheral generates a DMA request each time it needs a new data. The DMA controller can directly fetch the next sample from the CORDIC_RDATA register (in this case the DMA controller is operating in memory-to-peripheral mode, even though the source is a peripheral register). The act of reading the result allows the CORDIC to start a new calculation, which in turn generates a DMA write channel request, and the DMA controller transfers the next input value to the CORDIC_WDATA register. The DMA write channel is enabled (DMAWEN = 1), but the read channel must not be enabled.

In a similar way, data coming from another peripheral, such as an ADC, can be transferred directly to the CORDIC_WDATA register (in peripheral-to-memory mode). The DMA write channel must not be enabled. The CORDIC processes the input data and generate a DMA read request when complete, if DMAREN = 1. The DMA controller then transfers the result from CORDIC_RDATA register to memory (peripheral-to-memory mode).

Note: No DMA request is generated to program the CORDIC_CSR register. DMA mode is therefore only useful when repeatedly performing the same function with the same settings. The scale factor cannot be changed during a series of DMA transfers.

Note: Each DMA request must be acknowledged, as a result of the DMA performing an access to the CORDIC_WDATA or CORDIC_RDATA register. If an extraneous access to the relevant register occurs before this, the acknowledge is asserted prematurely, and could block the DMA channel. Therefore, when the DMA read channel is enabled, CPU access to the CORDIC_RDATA register must be avoided. Similarly, the processor must avoid accessing the CORDIC_WDATA register when the DMA write channel is enabled.

23.4 CORDIC registers

The CORDIC registers can only be accessed in 32-bit word format

23.4.1 CORDIC control/status register (CORDIC_CSR)

Address offset: 0x00

Reset value: 0x0000 0050

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARG SIZE	RES SIZE	NARG S	NRES	DMA WEN	DMA REN	IEN
r									rw	rw	rw	rw	rw	rw	rw

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

Bit 31 **RRDY**: Result ready flag

0: No new data in output register

1: CORDIC_RDATA register contains new data.

This bit is set by hardware when a CORDIC operation completes. It is reset by hardware when the CORDIC_RDATA register is read (NRES+1) times.

When this bit is set, if the IEN bit is also set, the CORDIC interrupt is asserted. If the DMAREN bit is set, a DMA read channel request is generated. While this bit is set, no new calculation is started.

Bits 30:23 Reserved, must be kept at reset value.

Bit 22 **ARGSIZE**: Width of input data

0: 32-bit

1: 16-bit

ARGSIZE selects the number of bits used to represent input data.

If 32-bit data is selected, the CORDIC_WDATA register expects arguments in q1.31 format.

If 16-bit data is selected, the CORDIC_WDATA register expects arguments in q1.15 format. The primary argument (ARG1) is written to the least significant half-word, and the secondary argument (ARG2) to the most significant half-word.

Bit 21 **RESSIZE**: Width of output data

0: 32-bit

1: 16-bit

RESSIZE selects the number of bits used to represent output data.

If 32-bit data is selected, the CORDIC_RDATA register contains results in q1.31 format.

If 16-bit data is selected, the least significant half-word of CORDIC_RDATA contains the primary result (RES1) in q1.15 format, and the most significant half-word contains the secondary result (RES2), also in q1.15 format.

Bit 20 **NARGS**: Number of arguments expected by the CORDIC_WDATA register
0: Only one 32-bit write (or two 16-bit values if ARGSIZE = 1) is needed for the next calculation.
1: Two 32-bit values must be written to the CORDIC_WDATA register to trigger the next calculation.
Reads return the current state of the bit.

Bit 19 **NRES**: Number of results in the CORDIC_RDATA register
0: Only one 32-bit value (or two 16-bit values if RESSIZE = 1) is transferred to the CORDIC_RDATA register on completion of the next calculation. One read from CORDIC_RDATA resets the RRDY flag.
1: Two 32-bit values are transferred to the CORDIC_RDATA register on completion of the next calculation. Two reads from CORDIC_RDATA are necessary to reset the RRDY flag.
Reads return the current state of the bit.

Bit 18 **DMAWEN**: Enable DMA write channel
0: Disabled. No DMA write requests are generated.
1: Enabled. Requests are generated on the DMA write channel whenever no operation is pending
This bit is set and cleared by software. A read returns the current state of the bit.

Bit 17 **DMAREN**: Enable DMA read channel
0: Disabled. No DMA read requests are generated.
1: Enabled. Requests are generated on the DMA read channel whenever the RRDY flag is set.
This bit is set and cleared by software. A read returns the current state of the bit.

Bit 16 **IEN**: Enable interrupt.
0: Disabled. No interrupt requests are generated.
1: Enabled. An interrupt request is generated whenever the RRDY flag is set.
This bit is set and cleared by software. A read returns the current state of the bit.

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

Bits 10:8 **SCALE[2:0]**: Scaling factor
The value of this field indicates the scaling factor applied to the arguments and/or results. A value n implies that the arguments have been multiplied by a factor 2^{-n}, and/or the results need to be multiplied by 2^{n}. Refer to *Section 23.3.2* for the applicability of the scaling factor for each function and the appropriate range.

Bits 7:4 **PRECISION[3:0]**: Precision required (number of iterations)
0: reserved
1 to 15: (Number of iterations)/4
To determine the number of iterations needed for a given accuracy refer to *Table 176*.
Note that for most functions, the recommended range for this field is 3 to 6.

Bits 3:0 **FUNC[3:0]**: Function

0: Cosine
1: Sine
2: Phase
3: Modulus
4: Arctangent
5: Hyperbolic cosine
6: Hyperbolic sine
7: Arctanh
8: Natural logarithm
9: Square Root
10 to 15: reserved

23.4.2 CORDIC argument register (CORDIC_WDATA)

Address offset: 0x04

Reset value: 0xXXXX XXXX

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

Bits 31:0 **ARG[31:0]**: Function input arguments

This register is programmed with the input arguments for the function selected in the CORDIC_CSR register FUNC field.

If 32-bit format is selected (CORDIC_CSR.ARGSIZE = 0) and two input arguments are required (CORDIC_CSR.NARGS = 1), two successive writes are required to this register. The first writes the primary argument (ARG1), the second writes the secondary argument (ARG2).

If 32-bit format is selected and only one input argument is required (NARGS = 0), only one write is required to this register, containing the primary argument (ARG1).

If 16-bit format is selected (CORDIC_CSR.ARGSIZE = 1), one write to this register contains both arguments. The primary argument (ARG1) is in the lower half, ARG[15:0], and the secondary argument (ARG2) is in the upper half, ARG[31:16]. In this case, NARGS must be set to 0.

Refer to *Section 23.3.2* for the arguments required by each function, and their permitted range.

When the required number of arguments has been written, the CORDIC evaluates the function designated by CORDIC_CSR.FUNC using the supplied input arguments, provided any previous calculation has completed. If a calculation is ongoing, the ARG1 and ARG 2 values are held pending until the calculation is completed and the results read. During this time, a write to the register cancels the pending operation and overwrite the argument data.

23.4.3 CORDIC result register (CORDIC_RDATA)

Address offset: 0x8

Reset value: 0x0000 0000

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

Bits 31:0 **RES[31:0]**: Function result

If 32-bit format is selected (CORDIC_CSR.RESSIZE = 0) and two output values are expected (CORDIC_CSR.NRES = 1), this register must be read twice when the RRDY flag is set. The first read fetches the primary result (RES1). The second read fetches the secondary result (RES2) and resets RRDY.

If 32-bit format is selected and only one output value is expected (NRES = 0), only one read of this register is required to fetch the primary result (RES1) and reset the RRDY flag.

If 16-bit format is selected (CORDIC_CSR.RESSIZE = 1), this register contains the primary result (RES1) in the lower half, RES[15:0], and the secondary result (RES2) in the upper half, RES[31:16]. In this case, NRES must be set to 0, and only one read performed.

A read from this register resets the RRDY flag in the CORDIC_CSR register.

23.4.4 CORDIC register map

Table 177. CORDIC register map and reset value

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CORDIC_CSR	RRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARGSIZE	RESSIZE	NARGS	NRES	DMAWEN	DMAREN	IEN	Res.	Res.	Res.	Res.	SCALE [2:0]				PRECISION [3:0]				FUNC [3:0]			
	Reset value	0									0	0	0	0	0	0	0					0	0	0	0	0	1	0	1	0	0	0	0
0x04	CORDIC_WDATA	ARG[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x08	CORDIC_RDATA	RES[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

24 Filter math accelerator (FMAC)

24.1 FMAC introduction

The filter math accelerator unit performs arithmetic operations on vectors. It comprises a multiplier/accumulator (MAC) unit, together with address generation logic which allows it to index vector elements held in local memory.

The unit includes support for circular buffers on input and output, which allows digital filters to be implemented. Both finite and infinite impulse response filters can be realized.

The unit allows frequent or lengthy filtering operations to be offloaded from the CPU, freeing up the processor for other tasks. In many cases it can accelerate such calculations compared to a software implementation, resulting in a speed-up of time critical tasks.

24.2 FMAC main features

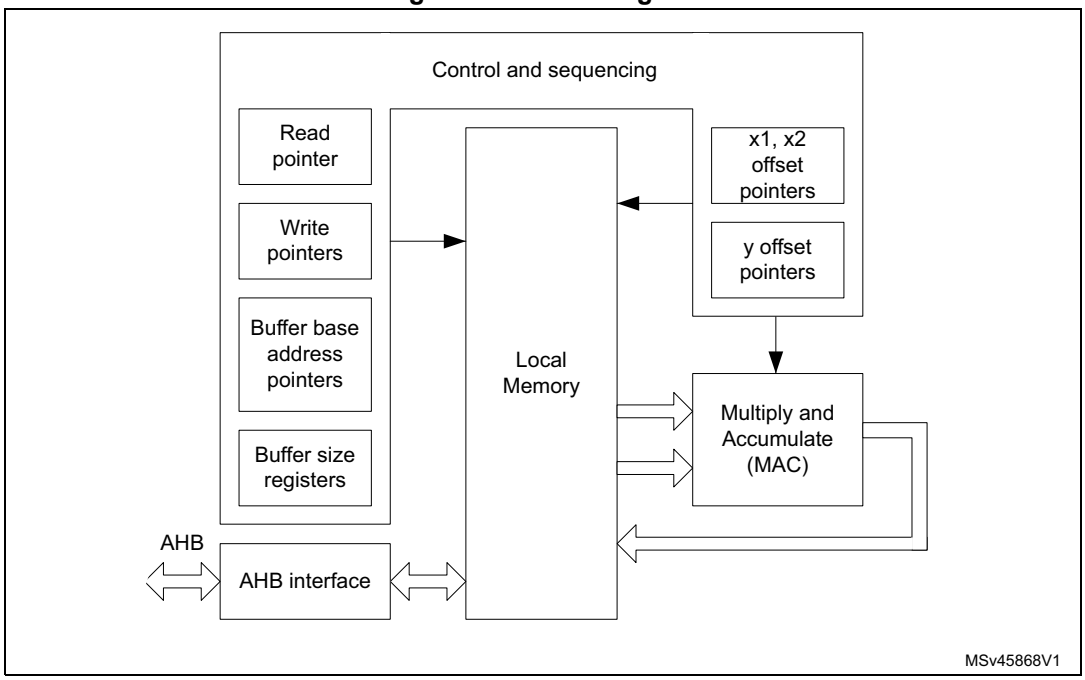
- 16 x 16-bit multiplier
- 24 + 2-bit accumulator with addition and subtraction
- 16-bit input and output data
- 256 x 16-bit local memory
- Up to three areas can be defined in memory for data buffers (two input, one output), defined by programmable base address pointers and associated size registers
- Input and output buffers can be circular
- Filter functions: FIR, IIR (direct form 1)
- Vector functions: Dot product, convolution, correlation
- AHB slave interface
- DMA read and write data channels

24.3 FMAC functional description

24.3.1 General description

The FMAC is shown in *Figure 95*.

Figure 95. Block diagram



The unit is built around a fixed point multiplier and accumulator (MAC). The MAC can take two 16-bit input signed values from memory, multiply them together and add them to the contents of the accumulator. The address of the input values in memory is determined using a set of pointers. These pointers can be loaded, incremented, decremented or reset by the internal hardware. The pointer and MAC operations are controlled by a built-in sequencer in order to execute the requested operation.

To calculate a dot product, the two input vectors are loaded into the local memory by the processor or DMA controller, and the requested operation is selected and started. Each pair of input vector elements is fetched from memory, multiplied together and accumulated. When all the vector elements have been processed, the contents of the accumulator are stored in the local memory, from where they can be read out by the processor or DMA.

The finite impulse response (FIR) filter operation (also known as convolution) consists in repeatedly calculating the dot product of the coefficient vector and a vector of input samples, the latter being shifted by one sample delay, with the least recent sample being discarded and a new sample added, at each repetition.

The infinite impulse response (IIR) filter operation is the convolution of the feedback coefficients with the previous output samples, added to the result of the FIR convolution.

A more detailed description of the filter operations is given in *Section 24.3.6: Filter functions*.

24.3.2 Local memory and buffers

The unit contains a 256 x 16-bit read/write memory which is used for local storage:

- Input values (the elements of the input vectors) are stored in two buffers, X1 and X2.
- Output values (the results of the operations) are stored in another buffer, Y.
- The locations and sizes of the buffers are designated as follows:
  - x1_base: the base address of the X1 buffer
  - x2_base: the base address of the X2 buffer
  - y_base: the base address of the Y buffer
  - x1_buf_size: the number of 16-bit addresses allocated to the X1 buffer
  - x2_buf_size: the number of 16-bit addresses allocated to the X2 buffer
  - y_buf_size: the number of 16-bit addresses allocated to the Y buffer.

These parameters are programmed in the corresponding registers when configuring the unit.

The CPU (or DMA controller) can initialize the contents of each buffer using the Initialization functions (*Section 24.3.5: Initialization functions*) and writing to the write data register. The data is transferred to the location within the target buffer indicated by a write pointer. After each new write, the write pointer is incremented. When the write pointer reaches the end of the allocated buffer space, it wraps back to the base address. This feature is used to load the elements of a vector prior to an operation, or to initialize a filter and load filter coefficients.

Buffer configuration

The buffer sizes and base address offsets must be configured in the X1, X2 and Y buffer configuration registers. For each function, the required buffer size is specified in the function description in *Section 24.3.6: Filter functions*. The base addresses can be chosen anywhere in internal memory, provided that all buffers fit within the internal memory address range (0x00 to 0xFF), that is, base address + buffer size must be less than 256.

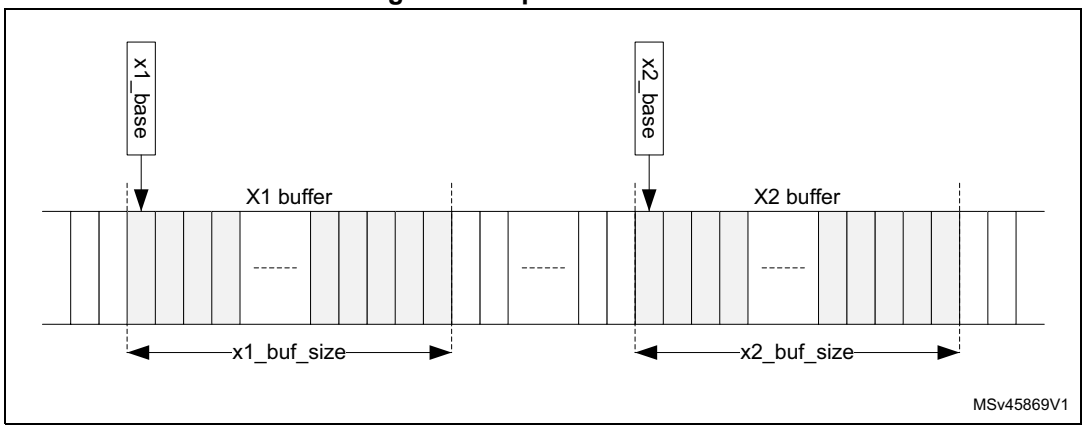
There is no constraint on the size and location of the buffers (they can overlap or even coincide exactly). For filter functions it is recommended not to overlap buffers as this can lead to erroneous behavior.

When circular buffer operation is required, an optional “headroom”, d, can be added to the buffer size. Furthermore, a watermark level can be set, to regulate the CPU or DMA activity. The value of d and the watermark level should be chosen according to the application performance requirements. For maximum throughput, the input buffer should never go empty, so d should be somewhat greater than the watermark level, allowing for any interrupt or DMA latency. On the other hand, if the input data can not be provided as fast as the unit can process them, the buffer can be allowed to empty waiting for the next data to be written, so d can be equal to the watermark level (to ensure that no overflow occurs on the input).

24.3.3 Input buffers

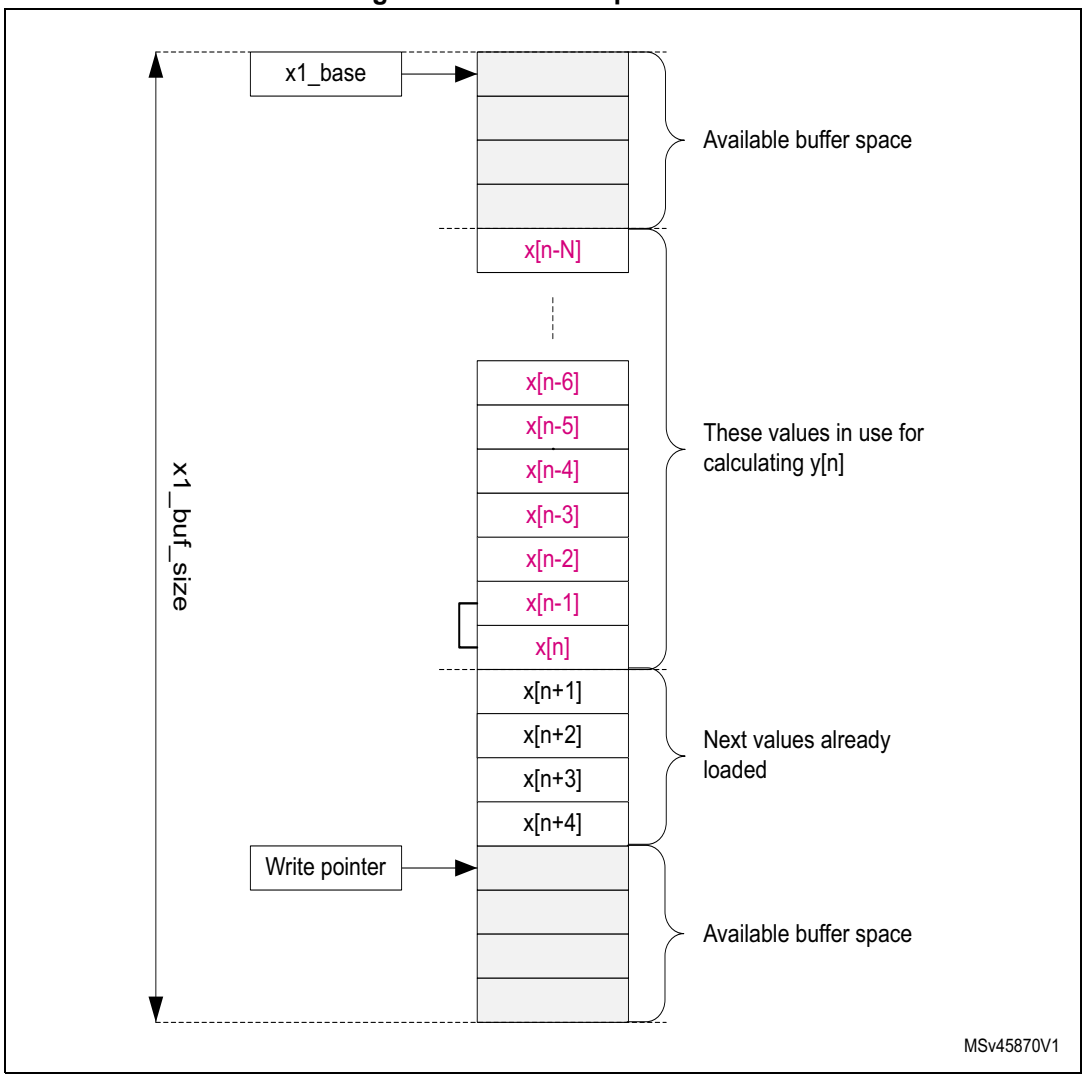
The X1 and X2 buffers are used to store data for input to the MAC. Each multiplication takes a value from the X1 buffer and a value from the X2 buffer and multiplies them together. A pointer in the control unit generates the read address offset (relative to the buffer base address) for each value. The pointers are managed by hardware according to the current function.

Figure 96. Input buffer areas



The X1 buffer can be used as a circular buffer, in which case new data are continually transferred into the input buffer whenever space is available. Pre-loading this buffer is optional for digital filters, since if no input samples have been written in the buffer when the operation is started, it is flagged as empty, which triggers the CPU or DMA to load new samples until there are enough to begin operation. Pre-loading is nevertheless useful in the case of a vector operation, that is, the input data is already available in system memory and circular operation is not required.

Figure 97. Circular input buffer



The X2 buffer can only be used in vector mode (that is not circular), and needs to be pre-loaded, except if the contents of the buffer do not change from one operation to the next. For filter functions, the X2 buffer is used to store the filter coefficients.

When operating as a circular buffer, the space allocated to the buffer (x1_buf_size) should generally be bigger than the number of elements in use for the current calculation, so that there are always new values available in the buffer. *Figure 97* illustrates the layout of the buffer for a filter operation. While calculating an output sample y[n], the unit uses a set of N+1 input samples, x[n-N] to x[n]. When this is finished, the unit starts the calculation of y[n+1], using the set of input samples x[n-N+1] to x[n+1]. The least-recent input sample, x[n-N], drops out of the input set, and a new sample, x[n+1], is added to it.

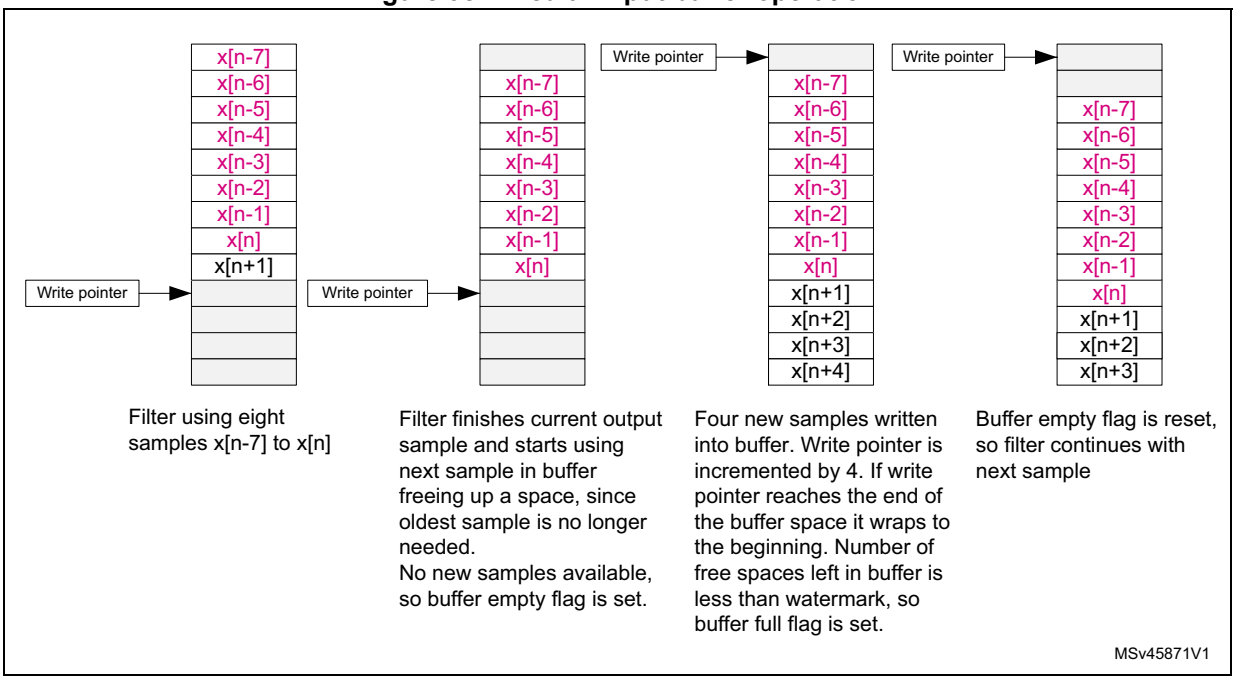
The processor, or DMA controller, must ensure that the new sample x[n+1] is available in the buffer space when required. If not, the buffer is flagged as empty, which stalls the execution of the unit until a new sample is added. No underflow condition is signaled on the X1 buffer.

Note: If the flow of samples is controlled by a timer or other peripheral such as an ADC, the buffer regularly goes empty, since the filter processes each new sample faster than the source can provide it. This is an essential feature of filter operation.

If the number of free spaces in the buffer is less than the watermark threshold programmed in the FULL_WM bitfield of the FMAC_X1BUFCFG register, the buffer is flagged as full. As long as the full flag is not set, interrupts are generated, if enabled, to request more data for the buffer. The watermark allows several data to be transferred under one interrupt, without danger of overflow. Nevertheless, if an overflow does occur, the OVFL error flag is set and the write data is ignored. The write pointer is not incremented in the event of an overflow.

The operation of the X1 buffer during a filtering operation is illustrated in *Figure 98*. This example shows an 8-tap FIR filter with a watermark set to four.

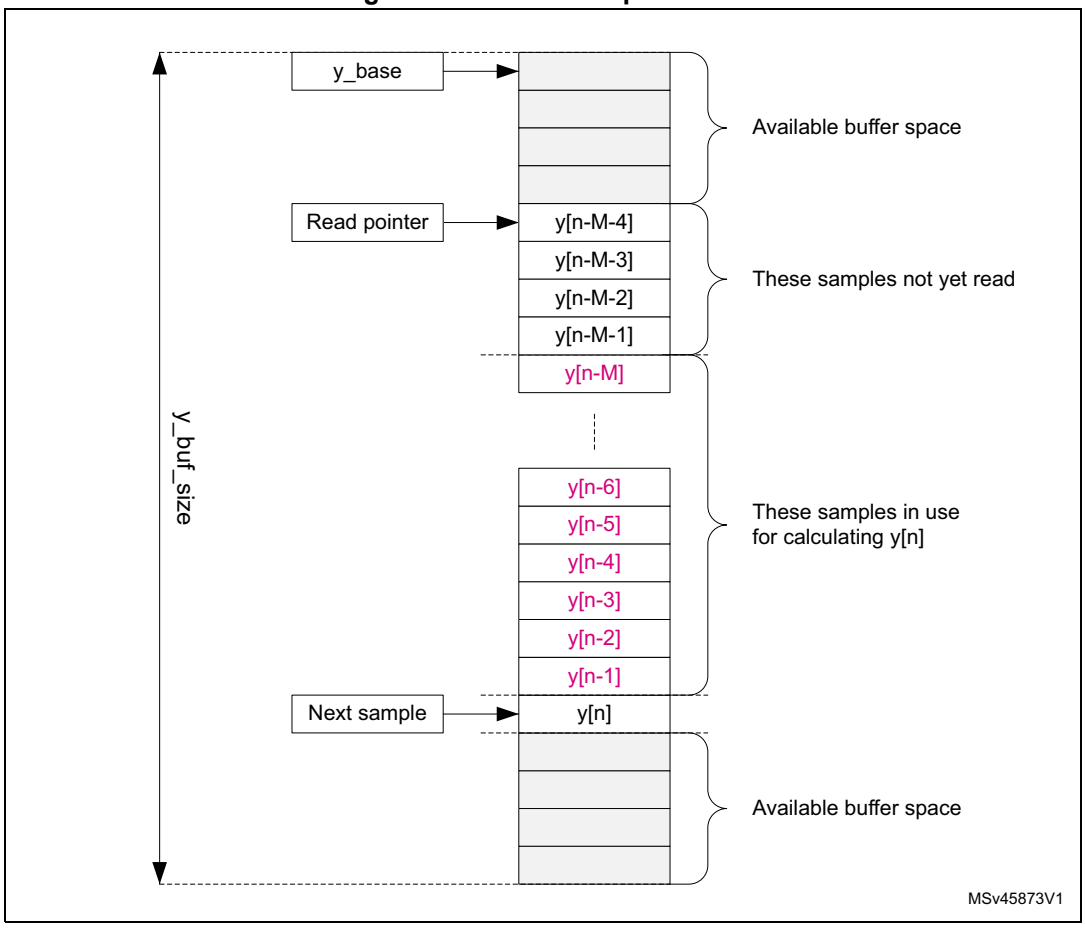
Figure 98. Circular input buffer operation



24.3.4 Output buffer

The Y (output) buffer is used to store the output of an accumulation. Each new output value is stored in the buffer until it is read by the processor or DMA controller. Each time a read access is made to the read data register, the read data is fetched from the address indicated by the read pointer. This pointer is incremented after each read, and wraps back to the base address when it reaches the end of the allocated Y buffer space.

Figure 99. Circular output buffer



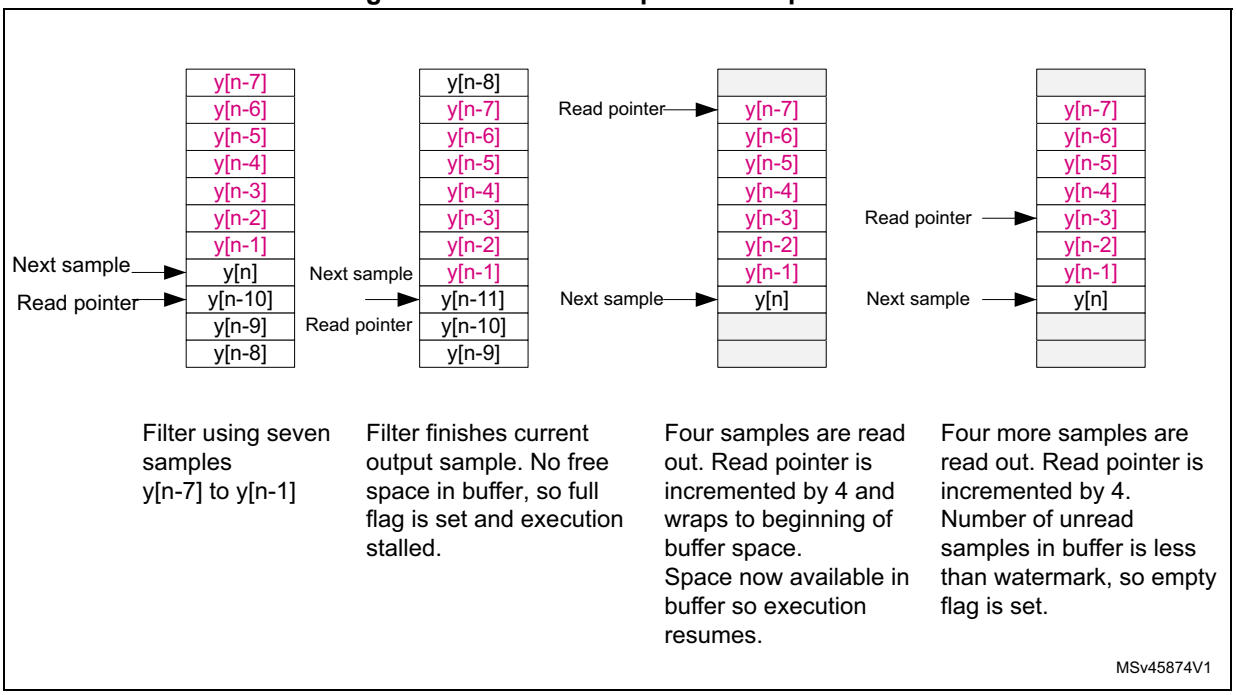
The Y buffer can also operate as a circular buffer. If the address for the next output value is the same as that indicated by the read pointer (an unread sample), then the buffer is flagged as full and execution stalled until the sample is read.

In the case of IIR filters, the Y buffer is used to store the set of M previous output samples, y[n-M] to y[n-1], used for calculating the next output sample y[n]. Each time a new sample is added to the set, the least recent sample y[n-M] drops out.

If the number of unread data in the buffer is less than the watermark threshold programmed in the EMPTY_WM bitfield of the FMAC_YBUFCFG register, the buffer is flagged as empty. As long as the empty flag is not set, interrupts or DMA requests are generated, if enabled, to request reads from the buffer. The watermark allows several data to be transferred under one interrupt, without danger of underflow. Nevertheless, if an underflow does occur, the UNFL error flag is set. In this case, the read pointer is not incremented and the read operation returns the content of the memory at the read pointer address.

The operation of the Y buffer in circular mode is illustrated in *Figure 100*. This example shows a 7-tap IIR filter with a watermark set to four.

Figure 100. Circular output buffer operation



24.3.5 Initialization functions

The following functions initialize the FMAC unit. They are triggered by writing the appropriate value in the FUNC bitfield of the FMAC_PARAM register, with the START bit set. The P and Q bitfields must also contain the appropriate parameter values for each function as detailed below. The R bitfield is not used. When the function completes, the START bit is automatically reset by hardware.

During initialization, it is recommended that the DMA requests and interrupts be disabled. The transfer of data into the FMAC memory can be done by software or by memory-to-memory DMA transfers, since no flow control is required.

Load X1 buffer

This function pre-loads the X1 buffer with N values, starting from the address in X1_BASE. Successive writes to the FMAC_WDATA register load the write data into the X1 buffer and increment the write address. The write pointer points to the address X1_BASE + N when the function completes.

The function can be used to pre-load the buffer with the elements of a vector, or to initialize the input storage elements of a filter.

Parameters

- The parameter P contains the number of values, N, to be loaded into the X1 buffer.
- The parameters Q and R are not used.

The function completes when N writes have been performed to the FMAC_WDATA register.

Load X2 buffer

This function pre-loads the X2 buffer with N + M values, starting from the address in X2_BASE. Successive writes to the FMAC_WDATA register load the write data into the X2 buffer and increment the write address.

The function can be used to pre-load the buffer with the elements of a vector, or the coefficients of a filter. In the case of an IIR, the N feed-forward and M feed-back coefficients are concatenated and loaded together into the X2 buffer. The total number of coefficients is equal to N + M. For an FIR, there are no feedback coefficients, so M = 0.

Parameters

- The parameter P contains the number of values, N, to be loaded into the X2 buffer starting from address X2_BASE.
- The parameter Q contains the number of values, M, to be loaded into the X2 buffer starting from address X2_BASE + N.
- The parameter R is not used.

The function completes when N + M writes have been performed to the FMAC_WDATA register.

Load Y buffer

This function pre-loads the Y buffer with N values, starting from the address in Y_BASE. Successive writes to the FMAC_WDATA register load the write data into the Y buffer and increment the write address. The read pointer points to the address Y_BASE + N when the function completes.

The function can be used to pre-load the feedback storage elements of an IIR filter.

Parameters

- The parameter P contains the number of values to be loaded into the Y buffer.
- The parameters Q and R are not used.

The function completes when N writes have been performed to the FMAC_WDATA register.

24.3.6 Filter functions

The following filter functions are supported by the FMAC unit. These functions are triggered by writing the corresponding value in the FUNC bitfield of the FMAC_PARAM register with the START bit set. The P, Q and R bitfields must also contain the appropriate parameter values for each function as detailed below. The filter functions continue to run until the START bit is reset by software.

Convolution (FIR filter)

Y = **B***__X__

$$y_n = 2^R \cdot \sum_{k=0}^{N} b_k x_{n-k}$$

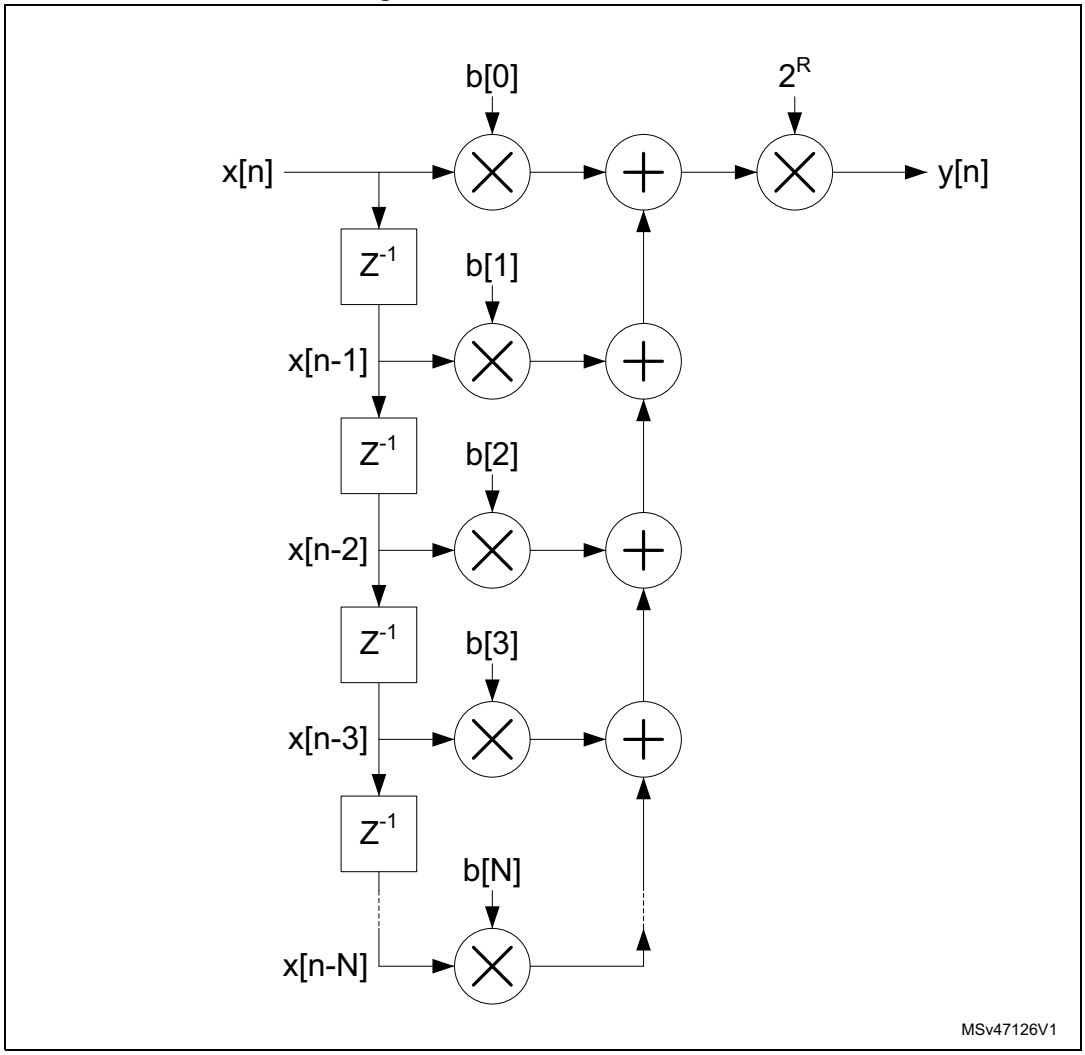
This function performs a convolution of a vector **B** of length N+1 and a vector **X** of indefinite length. The elements of **Y** for incrementing values of n are calculated as the dot product,

$y_n = \underline{\mathbf{B}}.\underline{\mathbf{X}}_{\mathbf{n}}$, where $\underline{\mathbf{X}}_{\mathbf{n}} = [x_{n-N},...,x_n]$ is composed of the N+1 elements of $\underline{\mathbf{X}}$ at indexes n - N to n.

This function corresponds to a finite impulse response (FIR) filter, where vector **B** contains the filter coefficients and vector **X** the sampled data.

The structure of the filter (direct form) is shown in *Figure 101*.

Figure 101. FIR filter structure



Note that the cross correlation vector can be calculated by reversing the order of the coefficient vector **B**.

Input:

- X1 buffer contains the elements of vector **X**. It is a circular buffer of length N + 1 + d.
- X2 buffer contains the elements of vector **B**. It is a fixed buffer of length N + 1.

Output:

- Y buffer contains the output values, y_n. It is a circular buffer of length d.

Parameters:

- The parameter P contains the length, N+1, of the coefficient vector **B** in the range [2:127].
- The parameter R contains the gain to be applied to the accumulator output. The value output to the Y buffer is multiplied by 2^R, where R is in the range [0:7]
- The parameter Q is not used.

The function completes when the START bit in the FMAC_PARAM register is reset by software.

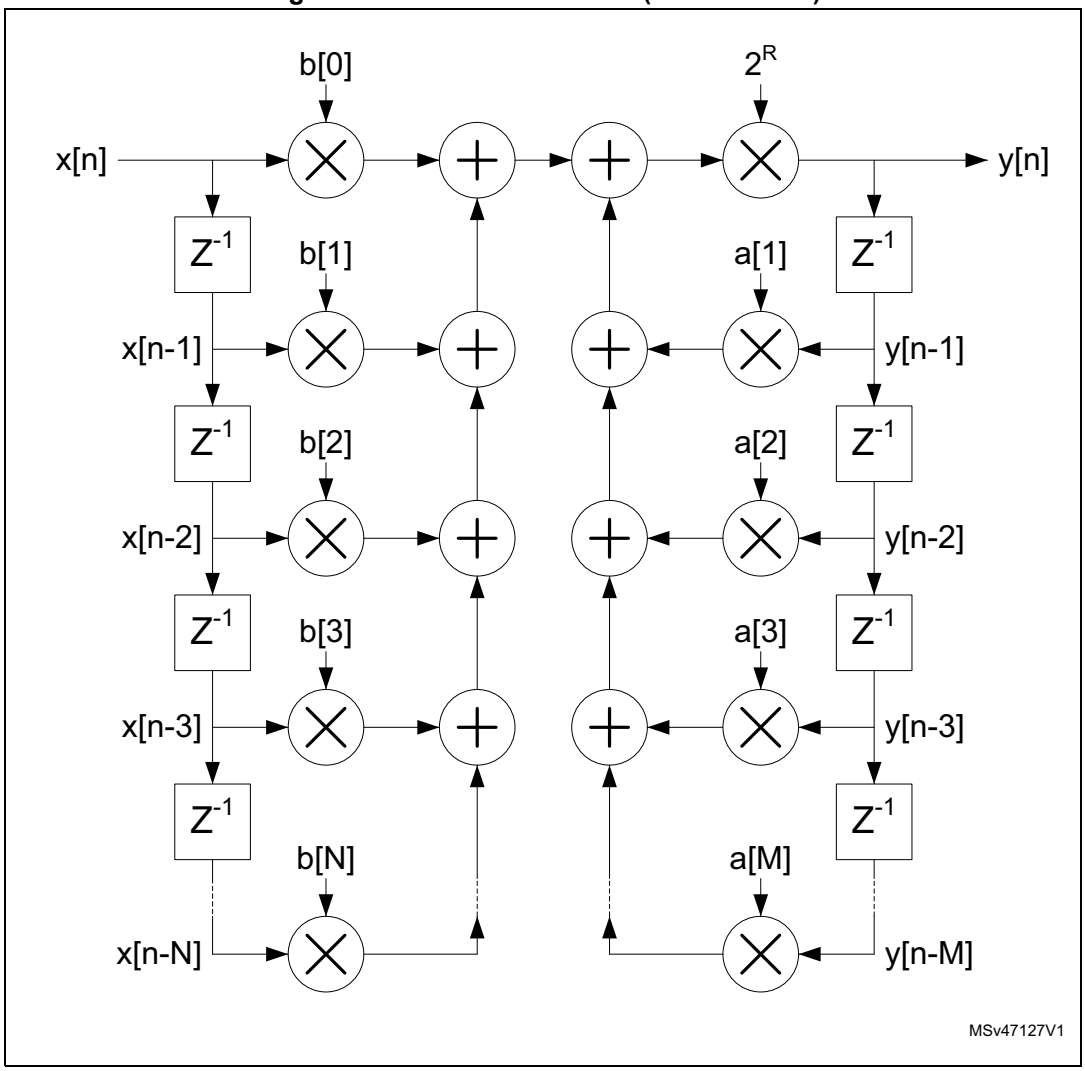
IIR filter

Y = **B*****X**+**A*****Y'**

$$y_n = 2^R \cdot \left(\sum_{k=0}^{N} b_k x_{n-k} + \sum_{k=1}^{M} a_k y_{n-k} \right)$$

This function implements an infinite impulse response (IIR) filter. The filter output vector **Y** is the convolution of a coefficient vector **B** of length N+1 and a vector **X** of indefinite length, plus the convolution of the delayed output vector **Y'** with a second coefficient vector **A**, of length M. The elements of **Y** for incrementing values of n are calculated as y_n = **B**.$\mathbf{X_n}$ + **A**.$\mathbf{Y_{n-1}}$, where $\mathbf{X_n}$ = $[x_{n-N},...,x_n]$ comprises the N+1 elements of **X** at indexes n - N to n, while $\mathbf{Y_{n-1}}$ = $[y_{n-M},...,y_{n-1}]$ comprises the M elements of **Y** at indexes n - M to n - 1. The structure of the filter (direct form 1) is shown in *Figure 102*.

Figure 102. IIR filter structure (direct form 1)



Input:

- X1 buffer contains the elements of vector **X**. It is a circular buffer of length N + 1+ d.
- X2 buffer contains the elements of coefficient vectors **B** and **A** concatenated (b_0, b_1, b_2..., b_N, a_1, a_2, ..., a_M). It is a fixed buffer of length M+N+1.

Output:

- Y buffer contains the output values, y_n. It is a circular buffer of length M + d.

Parameters

- The parameter P contains the length, N + 1, of the coefficient vector **B** in the range [2:64].
- The parameter Q contains the length, M, of the coefficient vector **A** in the range [1:63].
- The parameter R contains the gain to be applied to the accumulator output. The value output to the Y buffer is multiplied by 2^R, where R is in the range [0:7].

The function completes when the START bit in the FMAC_PARAM register is reset by software.

24.3.7 Fixed point representation

The FMAC operates in fixed point signed integer format. Input and output values are q1.15.

In q1.15 format, numbers are represented by one sign bit and 15 fractional bits (binary decimal places). The numeric range is therefore -1 (0x8000) to 1 - 2^{-15} (0x7FFF).

The accumulator has 26 bits, of which 22 are fractional and 4 are integer/sign (q4.22). This allows it to support partial accumulation sums in the range -8 (0x2000000) to +7.99999976 (0x1FFFFFF). A programmable gain from 0dB to 42dB in steps of 6dB can be applied at the output of the accumulator.

Note that the content of the accumulator is not saturated if the numeric range is exceeded. Partial sums whose value is greater than +7.99999976 or less than -8, wrap but this is harmless provided subsequent accumulations undo the wrapping. Nevertheless, the SAT flag in the FMAC_SR register is set if wrapping occurs, and generates an interrupt if the SATIEN bit is set in the FMAC_CR register. This helps in debugging the filter.

The data output by the accumulator can optionally be saturated, after application of the programmable gain, by setting the CLIPEN bit in the FMAC_CR register. If this bit is set, then any value which exceeds the numeric range of the q1.15 output, is set to 1 - 2^{-15} or -1, according to the sign. If clipping is not enabled, the unused accumulator bits after applying the gain is simply truncated.

24.3.8 Implementing FIR filters with the FMAC

The FMAC supports FIR filters of length N, where N is the number of taps or coefficients. The minimum local memory requirement for a FIR filter of length N is 2N + 1:

- N coefficients
- N input samples
- 1 output sample

Since the local memory size is 256, the maximum value for N is 127.

If maximum throughput is required, it may be necessary to allocate a small amount of extra space, d1 and d2, to the input and output sample buffers respectively, to ensure that the filter never stalls waiting for a new input sample, or waiting for the output sample to be read. In this case, the local memory requirement is 2N + d1 + d2.

The buffers should be configured as follows:

- X1_BUF_SIZE = N + d1;
- X2_BUF_SIZE = N;
- Y_BUF_SIZE = d2 (or 1 if no extra space is required)

The buffer base addresses can be allocated anywhere, but the X2 buffer must not overlap with the others, or else the coefficients are overwritten. An example configuration could be:

- X2_BASE = 0;
- X1_BASE = N;
- Y_BASE = 2N + d1

However, if the memory space is limited, the X1 and Y buffer areas can be overlapped, such that each output sample takes the place of the oldest input sample, which is no longer required:

- X2_BASE = 0;
- X1_BASE = N;
- Y_BASE = N

In this case, Y_BUF_SIZE = X1_BUF_SIZE = N + d1, so that the buffers remain in sync.

Note: The FULL_WM bitfield of X1 buffer configuration register must be programmed with a value less than or equal to $log_2(d1)$, otherwise the buffer is flagged full before N input samples have been written, and no more samples are requested. Similarly, the EMPTY_WM bitfield of the Y buffer configuration register must be less than or equal to $log_2(d2)$.

The filter coefficients **must** be pre-loaded into the X2 buffer, using the Load X2 Buffer function. The X1 buffer can optionally be pre-loaded with any number of samples up to a maximum of N. There is no point in pre-loading the Y buffer, since for the FIR filter there is no feedback path.

After configuring and initializing the buffers, the FMAC_CR register should be programmed according to the method used for writing and reading data to and from the FMAC memory.

Three methods are supported:

- Polling: No DMA request or Interrupt request is generated. Software must check that the X1_FULL flag is low before writing to WDATA, or that the Y_EMPTY flag is low before reading from RDATA.
- Interrupt: The interrupt request is asserted while the X1_FULL flag is low, for writes, or when the Y_EMPTY flag is low, for reads.
- DMA: DMA requests are asserted on the DMA write channel while the X1_FULL flag is low, and on the read channel while the Y_EMPTY flag is low.

Different methods can be used for read and for write. However it is not recommended to use both interrupts and DMA requests for the same operation[(a)]. The valid combinations are listed in *Table 178*.

Table 178. Valid combinations for read and write methods

WIEN	RIEN	DMAWEN	DMAREN	Write	Read
0	0	0	0	Polling	Polling
0	1	0	0	Polling	Interrupt
1	0	0	0	Interrupt	Polling
1	1	0	0	Interrupt	Interrupt
0	0	0	1	Polling	DMA
0	0	1	0	DMA	Polling
0	0	1	1	DMA	DMA
0	1	1	0	DMA	Interrupt
1	0	0	1	Interrupt	DMA

a. If both interrupts and DMA requests are enabled then only DMA should perform the transfer.

The filter is started by writing to the FMAC_PARAM register with the following bitfield values:

- FUNC = 8 (FIR filter);
- P = N (number of coefficients);
- Q = “Don’t care”;
- R = Gain;
- START = 1;

If less than N + d - 2^{FULL_WM} values have been pre-loaded in the X1 buffer, the X1FULL flag remains low. If the WIEN bit is set in the FMAC_CR register, then the interrupt request is asserted immediately to request the processor to write 2^{FULL_WM} additional samples into the buffer, via the FMAC_WDATA register. It remains asserted until the X1FULL flag goes high in the FMAC_SR register. The interrupt service routine should check the X1FULL flag after every 2^{FULL_WM} writes to the FMAC_WDATA register, and repeat the transfer until the flag goes high. Similarly, if the DMAWEN bit is set in the FMAC_CR register, DMA write channel requests are generated until the X1FULL flag goes high.

The filter calculates the first output sample when at least N samples have been written into the X1 buffer (including any pre-loaded samples).

When 2^{EMPTY_WM} output samples have been written into the Y buffer, the YEMPTY flag in the FMAC_SR register goes low. If the RIEN bit is set in the FMAC_CR register, the interrupt request is asserted to request the processor to read 2^{EMPTY_WM} samples from the buffer, via the FMAC_RDATA register. It remains asserted until the YEMPTY flag goes high. The interrupt service routine should check the YEMPTY flag after every 2^{EMPTY_WM} reads from the FMAC_RDATA register, and repeat the transfer until the flag goes high. If the DMAREN bit is set in the FMAC_CR, DMA read channel requests are generated until the YEMPTY flag goes high.

The filter continues to operate in this fashion until it is stopped by the software resetting the START bit.

24.3.9 Implementing IIR filters with the FMAC

The FMAC supports IIR filters of length N, where N is the number of feed-forward taps or coefficients. The number of feedback coefficients, M, can be any value from 1 to N-1. Only direct form 1 implementations can be realized, so filters designed for other forms need to be converted.

The minimum memory requirement for an IIR filter with N feed-forward coefficients and M feed-back coefficients is 2N + 2M:

- N + M coefficients
- N input samples
- M output samples

If M = N-1, then the maximum filter length that can be implemented is N = 64.

As for the FIR, for maximum throughput a small amount of additional space, d1 and d2, should be allowed in the input and output buffer size respectively, making the total memory requirement 2M + 2N + d1 + d2.

The buffers must be configured as follows:

- X1_BUF_SIZE = N + d1;
- X2_BUF_SIZE = N + M;
- Y_BUF_SIZE = M + d2;

The buffer base addresses can be allocated anywhere, but must not overlap. An example configuration is given below:

- X2_BASE = 0;
- X1_BASE = N + M;
- Y_BASE = 2N + M + d1;

Note: The FULL_WM bitfield of X1 buffer configuration register must be programmed with a value less than or equal to $log_2(d1)$, otherwise the buffer is flagged full before N input samples have been written, and no more samples are requested. Similarly, the EMPTY_WM bitfield of the Y buffer configuration register must be less than or equal to $log_2(d2)$.

The filter coefficients (N feed-forward followed by M feedback) **must** be pre-loaded into the X2 buffer, using the Load X2 Buffer function. The X1 buffer can optionally be pre-loaded with any number of samples up to a maximum of N. The Y buffer can optionally be pre-loaded with any number of values up to a maximum of M. This has the effect of initializing the feedback delay line.

After configuring the buffers, the FMAC_CR register should be programmed in the same way as for the FIR filter (see *Section 24.3.8: Implementing FIR filters with the FMAC*).

The filter is started by writing to the FMAC_PARAM register with the following bitfield values:

- FUNC = 9 (IIR filter);
- P = N (number of feed-forward coefficients);
- Q = M (number of feed-back coefficients);
- R = Gain;
- START = 1;

If less than $N + d - 2^{FULL\_WM}$ values have been pre-loaded in the X1 buffer, the X1FULL flag remains low. If the WIEN bit is set in the FMAC_CR register, then the interrupt request is asserted immediately to request the processor to write 2^{FULL_WM} additional samples into the buffer, via the FMAC_WDATA register. It remains asserted until the X1FULL flag goes high in the FMAC_SR register. The interrupt service routine should check the X1FULL flag after every 2^{FULL_WM} writes to the FMAC_WDATA register, and repeat the transfer until the flag goes high. Similarly, if the DMAWEN bit is set in the FMAC_CR register, DMA write channel requests are generated until the X1FULL flag goes high.

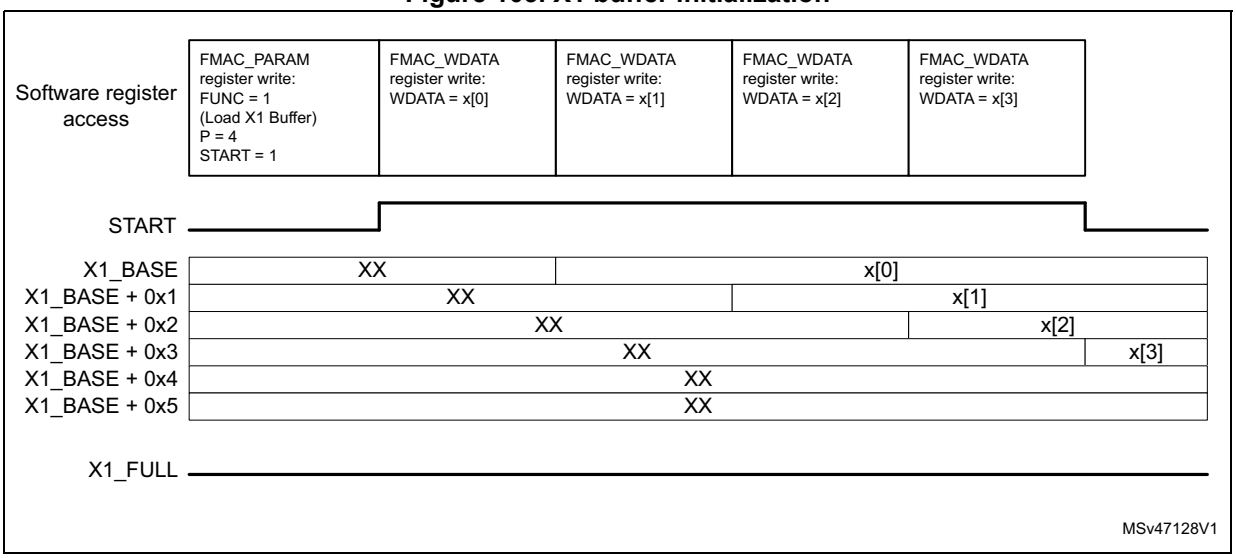
The filter calculates the first output sample when at least N samples have been written into the X1 buffer (including any pre-loaded samples). The first sample is calculated using the first N samples in the X1 buffer, and the first M samples in the Y buffer (whether or not they are preloaded. The first output sample is written into the Y buffer at Y_BASE + M.

When 2^{EMPTY_WM} new output samples have been written into the Y buffer, the YEMPTY flag in the FMAC_SR register goes low. If the RIEN bit is set in the FMAC_CR register, the interrupt request is asserted to request the processor to read 2^{EMPTY_WM} samples from the buffer, via the FMAC_RDATA register. It remains asserted until the YEMPTY flag goes high. The interrupt service routine should check the YEMPTY flag after every 2^{EMPTY_WM} reads from the FMAC_RDATA register, and repeat the transfer until the flag goes high. If the DMAREN bit is set in the FMAC_CR, DMA read channel requests are generated until the YEMPTY flag goes high

The filter continues to operate in this fashion until it is stopped by the software resetting the START bit.

24.3.10 Examples of filter initialization

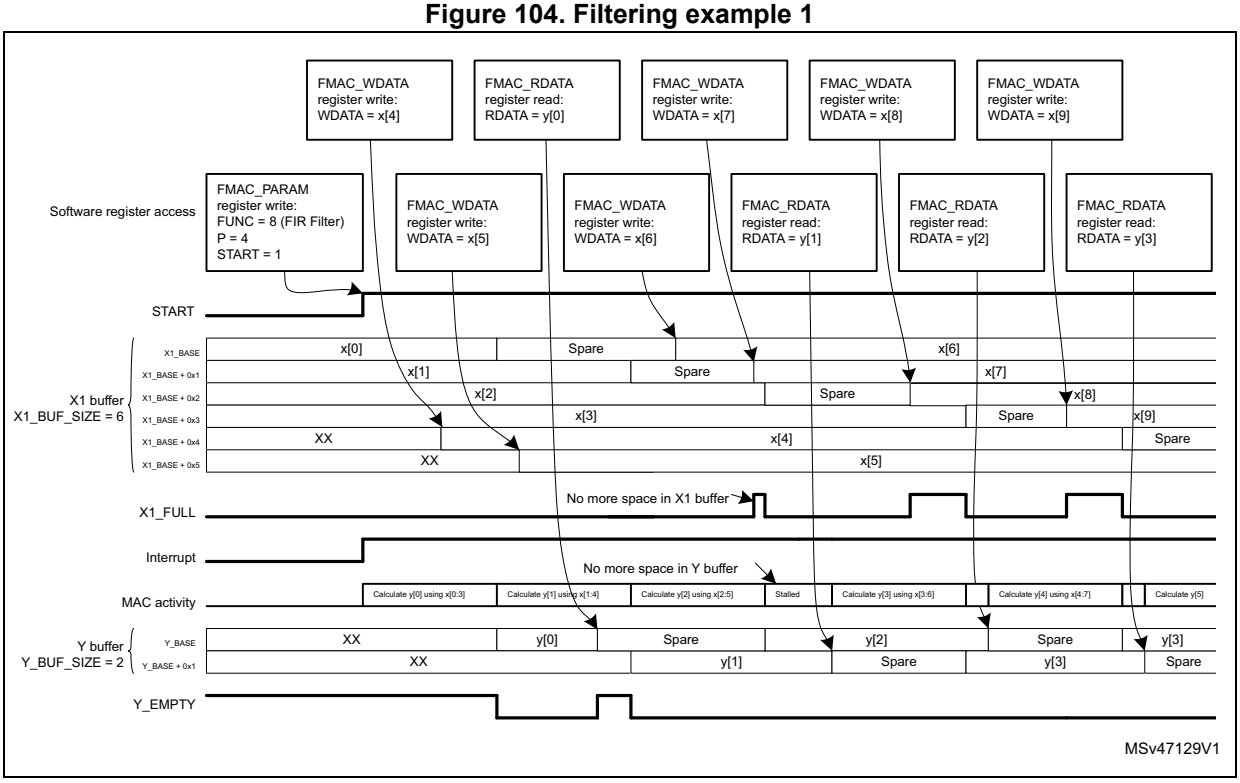
Figure 103. X1 buffer initialization



The example in *Figure 103* illustrates a X1 buffer pre-load with four samples (P = 4). The buffer size is six (X1_BUF_SIZE = 6). The initialization is launched by programming the FMAC_PARAM register with the START bit set. The four samples are then written to FMAC_WDATA, and transferred into local memory from X1_BASE onwards. The START bit resets after the fourth sample has been written. At this point, the X1 buffer contains the four samples, in order of writing, and the write pointer (next empty space) is at X1_BASE + 0x4.

24.3.11 Examples of filter operation

Figure 104. Filtering example 1



The example in *Figure 104* illustrates the beginning of a filter operation. The filter has four taps (P=4). The X1 buffer size is six and the Y buffer size is two. The FULL_WM and EMPTY_WM bitfields are both set to 0. Prior to starting the filter, the X1 buffer has been pre-loaded with four samples, x[0:3] as in *Figure 103*. So the filter starts calculating the first output sample, y[0], immediately after the START bit is set. Since the X1FULL flag is not set (due to two uninitialized spaces in the X1 buffer), the interrupt is asserted straight away, to request new data. The processor writes two new samples, x[4] and x[5], to the FMAC_WDATA register, which are transferred to the empty locations in the X1 buffer.

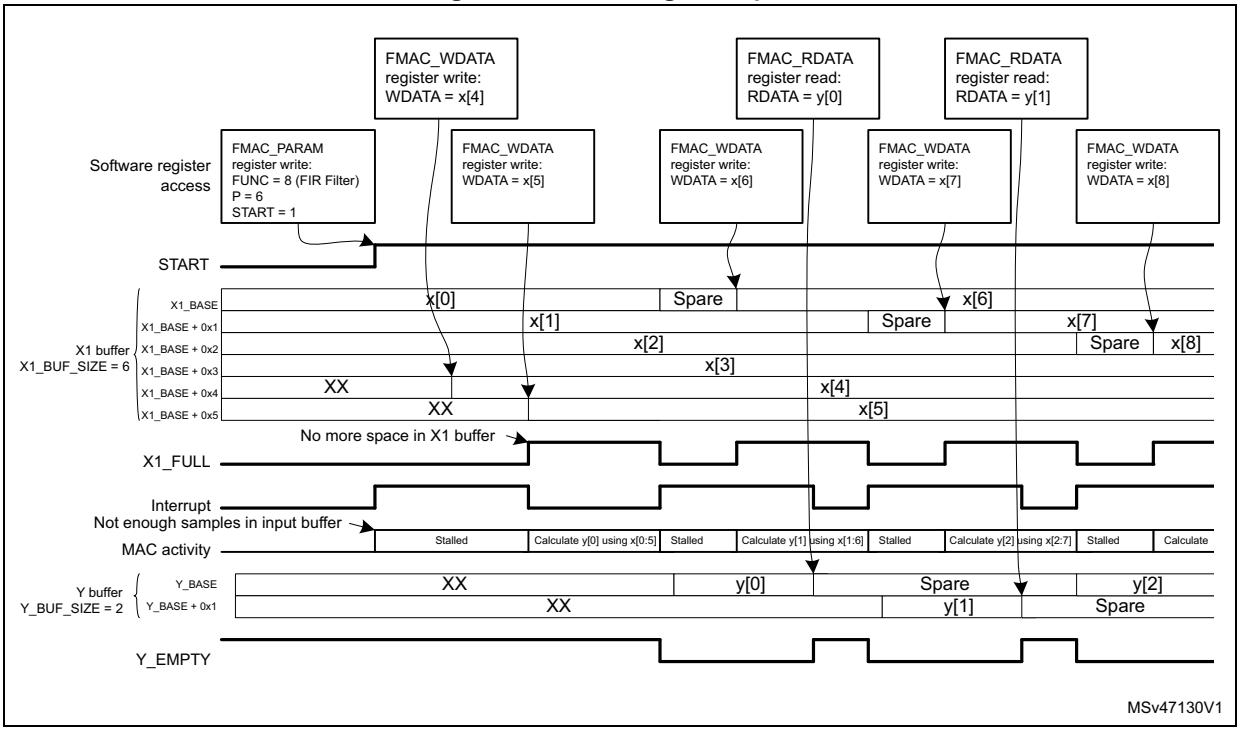
In the mean time, the FMAC finishes calculating the first output sample, y[0], and writes it into the Y buffer, causing the Y_EMPTY flag to go low. At the same time, the x[0] sample is discarded, as it is no longer required, freeing up its location in memory (at X1_BASE). The FMAC can immediately start work on the second output sample, y[1], since all the required input samples x[1:5] are present in the X1 buffer.

Since the Y_EMPTY flag is low, the interrupt remains active after the processor finishes writing x[5]. The processor reads y[0] from the FMAC_RDATA register, freeing up its location in the Y buffer. There are now no samples in the output buffer since y[1] is still being calculated, so the Y_EMPTY flag goes high. Nevertheless, the interrupt remains active, because there is still free space in the X1 buffer, which the processor next fills with x[6], and so on.

Note: In this example, the processor can fill the input buffer more quickly than the FMAC can process them, so the X1_full flag regularly goes active. However, it struggles to read the Y buffer fast enough, so the FMAC stalls regularly waiting for space to be freed up in the Y buffer. This means the filter is not executing at maximum throughput. The reason is that the

filter length is small and the processor relatively slow, in this example. So increasing the Y buffer size would not help.

Figure 105. Filtering example 2



The example in *Figure 105* illustrates the beginning of the same filter operation, but this time the filter has six taps (P=6). The X1 buffer size is six and the Y buffer size is two. The FULL_WM and EMPTY_WM bitfields are both set to 0. Prior to starting the filter, the X1 buffer has been pre-loaded with four samples, x[0:3] as in *Figure 103*. Because there are not enough samples in the input buffer, the X1FULL flag is not set, so the interrupt is asserted straight away, to request new data. The FMAC is stalled.

The processor writes two new samples, x[4] and x[5], to the FMAC_WDATA register, which are transferred to the empty locations in the X1 buffer. As soon as there are six unused samples in the X1 buffer, the X1_FULL flag goes active (since the buffer size is six), causing the interrupt to go inactive. The FMAC starts calculating the first output sample, y[0]. Since this requires all six input samples, there are no free spaces in the X1 buffer and so the X1_FULL flag remains active. Only when the FMAC finishes calculating y[0] and writes it into the Y buffer, can x[0] be discarded, freeing up a space in the X1 buffer, and deasserting X1_FULL. At the same time, the Y_EMPTY flag goes inactive. Both these flag states cause the interrupt to be asserted, requesting the processor to write a new input sample, first of all, and then read the output sample just calculated. The FMAC remains stalled until a new input sample is written.

In this example, the processor has to wait for the FMAC to finish calculating the current output sample, before it can write a new input sample, and therefore the X1 buffer regularly goes empty, stalling the FMAC. This can be avoided by allowing some extra space in the input buffer.

24.3.12 Filter design tips

The FMAC architecture imposes some constraints detailed below, on the design of digital filters.

1. Implementation of direct form 2, or transposed forms, is not efficient. Filters which have been designed for such forms should be converted to direct form 1.
2. Cascaded filters must either be combined into a single stage, or implemented as separate filters. In the latter case, multiple sets of filter coefficients can be pre-loaded into the memory, one set per stage, and only the X2_BASE address changed to select which set is used. The most efficient method of implementing a multi-stage filter is to pre-load a large X1 buffer with input samples, run the IIR filter function on it using the first stage coefficients, and store the output samples back in memory. Then change the X2_BASE pointer to point to the 2nd stage coefficients, and reload the input buffer with the output of the first stage (with a gain if required), before running the IIR function again. The procedure is repeated for all stages. Once the final stage samples have been transferred back into system memory, the input buffer can be loaded with the next set of input samples, and a new round of calculations started. Note that the N sample input buffer of each stage must be pre-loaded first of all with the N-1 last inputs from the previous round, plus one new sample, in order to keep continuity between each round. Similarly, the output buffer of each stage must be loaded with the last M samples from the previous round, for the same reason.
3. The use of direct form 1 for IIR designs can lead to large positive or negative partial sums in the accumulator, if for example a large step occurs on the input, or some of the filter coefficients' absolute values are >1. Since the accumulator is limited to 26 bits, the biggest value that it can handle without wrapping (changing sign) is 0x1FFFFFF positive or 0x2000000 negative. This corresponds to 3.99999988 and -4 respectively in q3.23 fixed point format. Wrapping does not represent a problem provided the wrapping is "undone" before the end of the accumulation. However this is not always the case when a filter is starting up and can lead to unexpected results. Consider pre-loading the output buffer with suitable values to avoid this.
4. The IIR filter has feed-forward (numerator) coefficients $[b_0, b_1, ..., b_{N-1}]$, and feed-back (denominator) coefficients $[1, a_1, ..., a_M]$. Many IIR filters require some of the denominator coefficients to have an absolute value greater than 1 to achieve a steep roll-off in the frequency response. Given that the coefficients are coded in fixed point q1.15 format, this is not possible. Nevertheless, by scaling the denominator coefficients by a factor 2^{-R}, such that $2^{-R}.[1, a_1, ..., a_M]$ are all less than 1, such filters can be implemented. However an inverse gain of 2^R must be applied at the output of the accumulator to compensate the scaling. This has an adverse effect on the signal-to-noise ratio.

24.4 FMAC registers

24.4.1 FMAC X1 buffer configuration register (FMAC_X1BUFCFG)

Address offset: 0x00

Reset value: 0x0000 0000

Access: word access

This register can only be modified if START = 0 in the FMAC_PARAM register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	FULL_WM[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw	rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X1_BUF_SIZE[7:0]								X1_BASE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **FULL_WM[1:0]**: Watermark for buffer full flag

Defines the threshold for setting the X1 buffer full flag when operating in circular mode. The flag is set if the number of free spaces in the buffer is less than 2^{FULL_WM}.

0: Threshold = 1

1: Threshold = 2

2: Threshold = 4

3: Threshold = 8

Setting a threshold greater than 1 allows several data to be transferred into the buffer under one interrupt.

Threshold should be set to 1 if DMA write requests are enabled (DMAWEN = 1 in FMAC_CR register).

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:8 **X1_BUF_SIZE[7:0]**: Allocated size of X1 buffer in 16-bit words

The minimum buffer size is the number of feed-forward taps in the filter (+ the watermark threshold - 1).

Bits 7:0 **X1_BASE[7:0]**: Base address of X1 buffer

24.4.2 FMAC X2 buffer configuration register (FMAC_X2BUFCFG)

Address offset: 0x04

Reset value: 0x0000 0000

Access: word access

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X2_BUF_SIZE[7:0]								X2_BASE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **X2_BUF_SIZE[7:0]**: Size of X2 buffer in 16-bit words
This bitfield can not be modified when a function is ongoing (START = 1).

Bits 7:0 **X2_BASE[7:0]**: Base address of X2 buffer
The X2 buffer base address can be modified while START=1, for example to change coefficient values. The filter should be stalled when doing this, since changing the coefficients while a calculation is ongoing affects the result.

24.4.3 FMAC Y buffer configuration register (FMAC_YBUFCFG)

Address offset: 0x08

Reset value: 0x0000 0000

Access: word access

This register can only be modified if START = 0 in the FMAC_PARAM register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	EMPTY_WM[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw	rw								

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

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **EMPTY_WM[1:0]**: Watermark for buffer empty flag
Defines the threshold for setting the Y buffer empty flag when operating in circular mode. The flag is set if the number of unread values in the buffer is less than 2^{EMPTY_WM}.
0: Threshold = 1
1: Threshold = 2
2: Threshold = 4
3: Threshold = 8
Setting a threshold greater than 1 allows several data to be transferred from the buffer under one interrupt.
Threshold should be set to 1 if DMA read requests are enabled (DMAREN = 1 in FMAC_CR register).

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:8 **Y_BUF_SIZE[7:0]**: Size of Y buffer in 16-bit words
For FIR filters, the minimum buffer size is 1 (+ the watermark threshold). For IIR filters the minimum buffer size is the number of feedback taps (+ the watermark threshold).

Bits 7:0 **Y_BASE[7:0]**: Base address of Y buffer

24.4.4 FMAC parameter register (FMAC_PARAM)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: word access

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

Bit 31 **START**: Enable execution
0: Stop execution
1: Start execution
Setting this bit triggers the execution of the function selected in the FUNC bitfield. Resetting it by software stops any ongoing function. For initialization functions, this bit is reset by hardware.

Bits 30:24 **FUNC[6:0]**: Function
0: Reserved
1: Load X1 buffer
2: Load X2 buffer
3: Load Y buffer
4 to 7: Reserved
8: Convolution (FIR filter)
9: IIR filter (direct form 1)
10 to 127: Reserved
This bitfield can not be modified when a function is ongoing (START = 1)

Bits 23:16 **R[7:0]**: Input parameter R.
The value of this parameter is dependent on the function.
This bitfield can not be modified when a function is ongoing (START = 1)

Bits 15:8 **Q[7:0]**: Input parameter Q.
The value of this parameter is dependent on the function.
This bitfield can not be modified when a function is ongoing (START = 1)

Bits 7:0 **P[7:0]**: Input parameter P.
The value of this parameter is dependent on the function
This bitfield can not be modified when a function is ongoing (START = 1)

24.4.5 FMAC control register (FMAC_CR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESET
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLIP EN	Res.	Res.	Res.	Res.	Res.	DMA WEN	DMA REN	Res.	Res.	Res.	SAT IEN	UNFL IEN	OVFL IEN	WIEN	RIEN
rw						rw	rw				rw	rw	rw	rw	rw

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

Bit 16 **RESET**: Reset FMAC unit

This resets the write and read pointers, the internal control logic, the FMAC_SR register and the FMAC_PARAM register, including the START bit if active. Other register settings are not affected. This bit is reset by hardware.

0: Reset inactive

1: Reset active

Bit 15 **CLIPEN**: Enable clipping

0: Clipping disabled. Values at the output of the accumulator which exceed the q1.15 range, wrap.

1: Clipping enabled. Values at the output of the accumulator which exceed the q1.15 range are saturated to the maximum positive or negative value (+1 or -1) according to the sign.

Bits 14:10 Reserved, must be kept at reset value.

Bit 9 **DMAWEN**: Enable DMA write channel requests

0: Disable. No DMA requests are generated

1: Enable. DMA requests are generated while the X1 buffer is not full.

This bit can only be modified when START= 0 in the FMAC_PARAM register. A read returns the current state of the bit.

Bit 8 **DMAREN**: Enable DMA read channel requests

0: Disable. No DMA requests are generated

1: Enable. DMA requests are generated while the Y buffer is not empty.

This bit can only be modified when START= 0 in the FMAC_PARAM register. A read returns the current state of the bit.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SATIEN**: Enable saturation error interrupts

0: Disabled. No interrupts are generated upon saturation detection.

1: Enabled. An interrupt request is generated if the SAT flag is set

This bit is set and cleared by software. A read returns the current state of the bit.

Bit 3 **UNFLIEN**: Enable underflow error interrupts

0: Disabled. No interrupts are generated upon underflow detection.

1: Enabled. An interrupt request is generated if the UNFL flag is set

This bit is set and cleared by software. A read returns the current state of the bit.

Bit 2 **OVFLIEN**: Enable overflow error interrupts
0: Disabled. No interrupts are generated upon overflow detection.
1: Enabled. An interrupt request is generated if the OVFL flag is set
This bit is set and cleared by software. A read returns the current state of the bit.

Bit 1 **WIEN**: Enable write interrupt
0: Disabled. No write interrupt requests are generated.
1: Enabled. An interrupt request is generated while the X1 buffer FULL flag is not set.
This bit is set and cleared by software. A read returns the current state of the bit.

Bit 0 **RIEN**: Enable read interrupt
0: Disabled. No read interrupt requests are generated.
1: Enabled. An interrupt request is generated while the Y buffer EMPTY flag is not set.
This bit is set and cleared by software. A read returns the current state of the bit.

24.4.6 FMAC status register (FMAC_SR)

Address offset: 0x14

Reset value: 0x0000 0001

Access: word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SAT	UNFL	OVFL	Res.	Res.	Res.	Res.	Res.	Res.	X1 FULL	Y EMPTY
					r	r	r							r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **SAT**: Saturation error flag
Saturation occurs when the result of an accumulation exceeds the numeric range of the accumulator.
0: No saturation detected
1: Saturation detected. If the SATIEN bit is set, an interrupt is generated.
This flag is cleared by a reset of the unit.

Bit 9 **UNFL**: Underflow error flag
An underflow occurs when a read is made from FMAC_RDATA when no valid data is available in the Y buffer.
0: No underflow detected
1: Underflow detected. If the UNFLIEN bit is set, an interrupt is generated.
This flag is cleared by a reset of the unit.

Bit 8 **OVFL**: Overflow error flag
An overflow occurs when a write is made to FMAC_WDATA when no free space is available in the X1 buffer.
0: No overflow detected
1: Overflow detected. If the OVFLIEN bit is set, an interrupt is generated.
This flag is cleared by a reset of the unit.

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

Bit 1 **X1FULL**: X1 buffer full flag

The buffer is flagged as full if the number of available spaces is less than the FULL_WM threshold. The number of available spaces is the difference between the write pointer and the least recent sample currently in use.

0: X1 buffer not full. If the WIEN bit is set, the interrupt request is asserted until the flag is set. If DMAWEN is set, DMA write channel requests are generated until the flag is set.

1: X1 buffer full.

This flag is set and cleared by hardware, or by a reset.

Note: after the last available space in the X1 buffer is filled there is a delay of 3 clock cycles before the X1FULL flag goes high. To avoid any risk of overflow it is recommended to insert a software delay after writing to the X1 buffer before reading the FMAC_SR. Alternatively, a FULL_WM threshold of 2 can be used.

Bit 0 **YEMPTY**: Y buffer empty flag

The buffer is flagged as empty if the number of unread data is less than the EMPTY_WM threshold. The number of unread data is the difference between the read pointer and the current output destination address.

0: Y buffer not empty. If the RIEN bit is set, the interrupt request is asserted until the flag is set. If DMAREN is set, DMA read channel requests are generated until the flag is set.

1: Y buffer empty.

This flag is set and cleared by hardware, or by a reset.

Note: after the last sample is read from the Y buffer there is a delay of 3 clock cycles before the YEMPTY flag goes high. To avoid any risk of underflow it is recommended to insert a software delay after reading from the Y buffer before reading the FMAC_SR. Alternatively, an EMPTY_WM threshold of 2 can be used.

24.4.7 FMAC write data register (FMAC_WDATA)

Address offset: 0x18

Reset value: 0x0000 0000

Access: word and half-word access

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **WDATA[15:0]**: Write data

When a write access to this register occurs, the write data are transferred to the address offset indicated by the write pointer. The pointer address is automatically incremented after each write access.

24.4.8 FMAC read data register (FMAC_RDATA)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: word and half-word access

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RDATA[15:0]**: Read data

When a read access to this register occurs, the read data are the contents of the Y output buffer at the address offset indicated by the READ pointer. The pointer address is automatically incremented after each read access.

24.4.9 FMAC register map

Table 179.FMAC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	FMAC_X1BUFCFG	Res.	Res.	Res.	Res.	Res.	Res.	FULL_WM[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	X1_BUF_SIZE[7:0]								X1_BASE[7:0]							
	Reset value							0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	FMAC_X2BUFCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	X2_BUF_SIZE[7:0]								X2_BASE[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	FMAC_YBUFCFG	Res.	Res.	Res.	Res.	Res.	Res.	EMPTY_WM[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Y_BUF_SIZE[7:0]								Y_BASE[7:0]							
	Reset value							0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	FMAC_PARAM	START	FUNC[6:0]							R[7:0]								Q[7:0]								P[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	FMAC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESET	CLIPEN	Res.	Res.	Res.	Res.	Res.	DMAWEN	DMAREN	Res.	Res.	Res.	SATIEN	UNFLIEN	OVFLIEN	WIEN	RIEN
	Reset value																0	0						0	0				0	0	0	0	0
0x14	FMAC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAT	UNFL	OVFL	Res.	Res.	Res.	Res.	Res.	Res.	X1FULL	YEMPTY
	Reset value																						0	0	0							0	1
0x18	FMAC_WDATA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	FMAC_RDATA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

25 Flexible static memory controller (FSMC)

25.1 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).

25.2 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, only a few bits can be changed on-the-fly:

- MBKEN, FMCEN, WEN bits in FMC_BCRx register
- ECCEN and PBKEN bits in the FMC_PCR register
- IFS, IRS and ILS bits in the FMC_SR register

Follow the below sequence to modify parameters while the FMC is enabled:

1. First disable the FMC controller to prevent further accesses to any memory controller while the register is modified.
2. Update all required configurations.
3. Enable the FMC controller again.

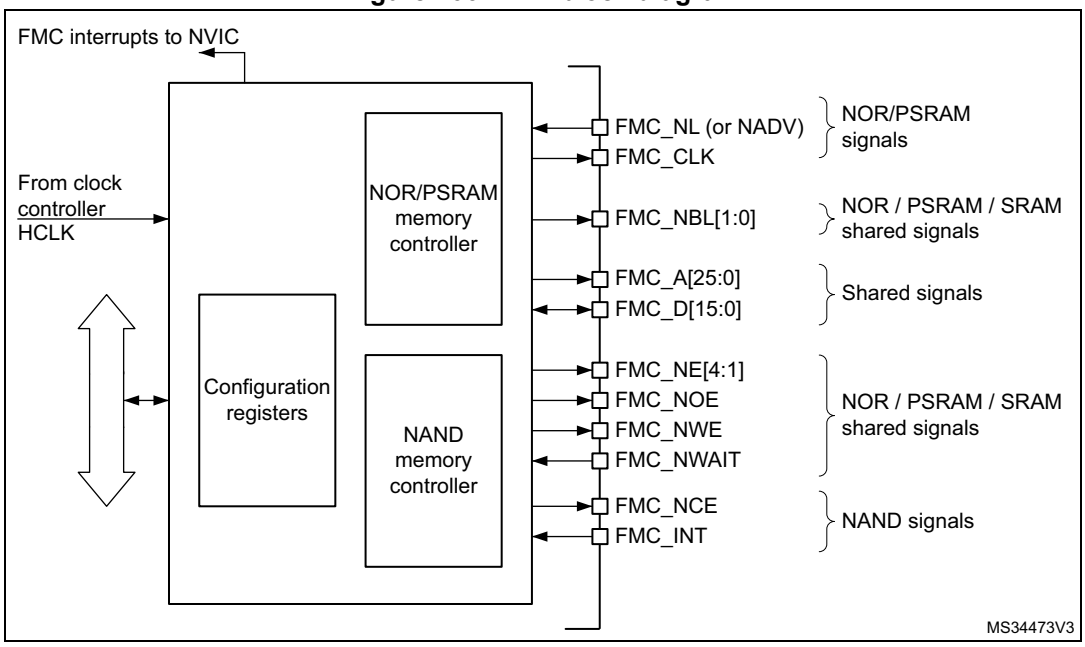
25.3 FMC block diagram

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 106. FMC block diagram



25.4 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 a 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.

25.4.1 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 25.6.6*, for a detailed description of the NOR Flash/PSRAM controller registers. Refer to *Section 25.7.7*, for a detailed description of the NAND Flash registers.

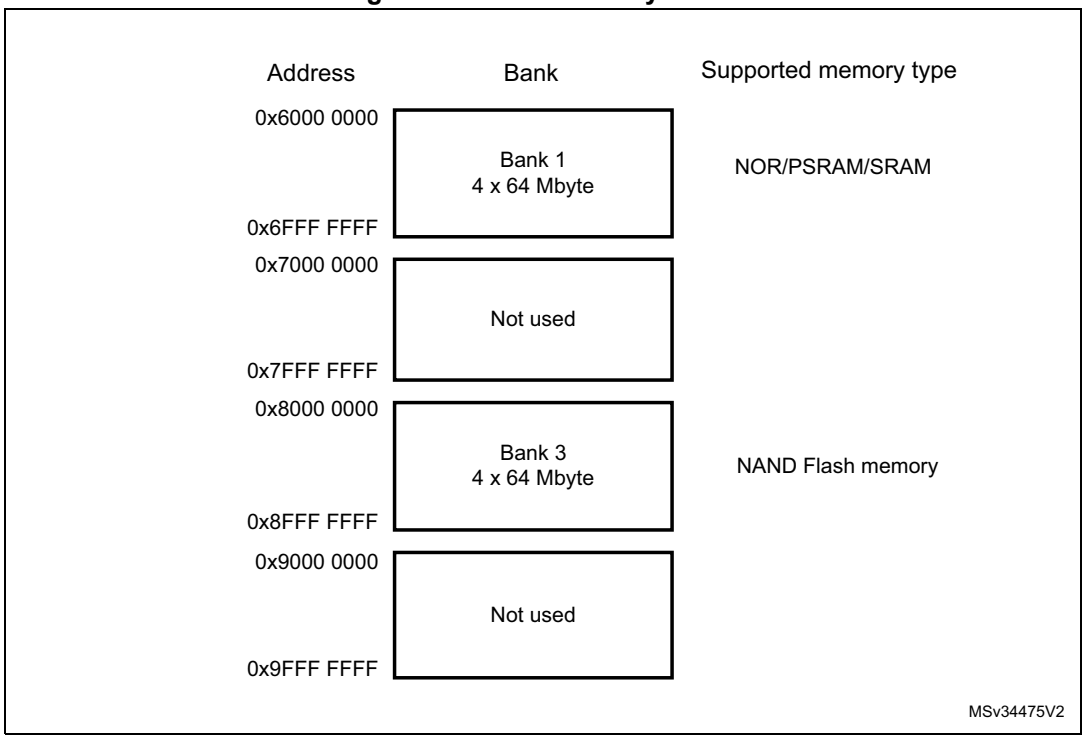
25.5 External device address mapping

From the FMC point of view, the external memory is divided into fixed-size banks of 256 Mbytes each (see *Figure 107*):

- 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 107. FMC memory banks



25.5.1 NOR/PSRAM address mapping

HADDR[27:26] bits are used to select one of the four memory banks as shown in *Table 180*.

Table 180. 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 181. 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 Mbits
16-bit	HADDR[25:1] >> 1	64 Mbytes/2 x 16 = 512 Mbits

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].

25.5.2 NAND Flash memory address mapping

The NAND bank is divided into memory areas as indicated in *Table 182*.

Table 182. 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 183* 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 183. 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.

25.6 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 25.6.6: 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 25.6.5: 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 25.6.6: NOR/PSRAM controller registers*).

The programmable memory parameters include access times (see *Table 184*) and support for wait management (for PSRAM and NOR Flash accessed in Burst mode).

Table 184. 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

Table 184. Programmable NOR/PSRAM access parameters (continued)

Parameter	Function	Access mode	Unit	Min.	Max.
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

25.6.1 External memory interface signals

Table 185, *Table 186* and *Table 187* 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 185. 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 186. 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 187. 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 188. 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 188. 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).

25.6.2 Supported memories and transactions

Table 189 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 189. 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 189. 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]

25.6.3 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.

25.6.4 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 108. Mode 1 read access waveforms

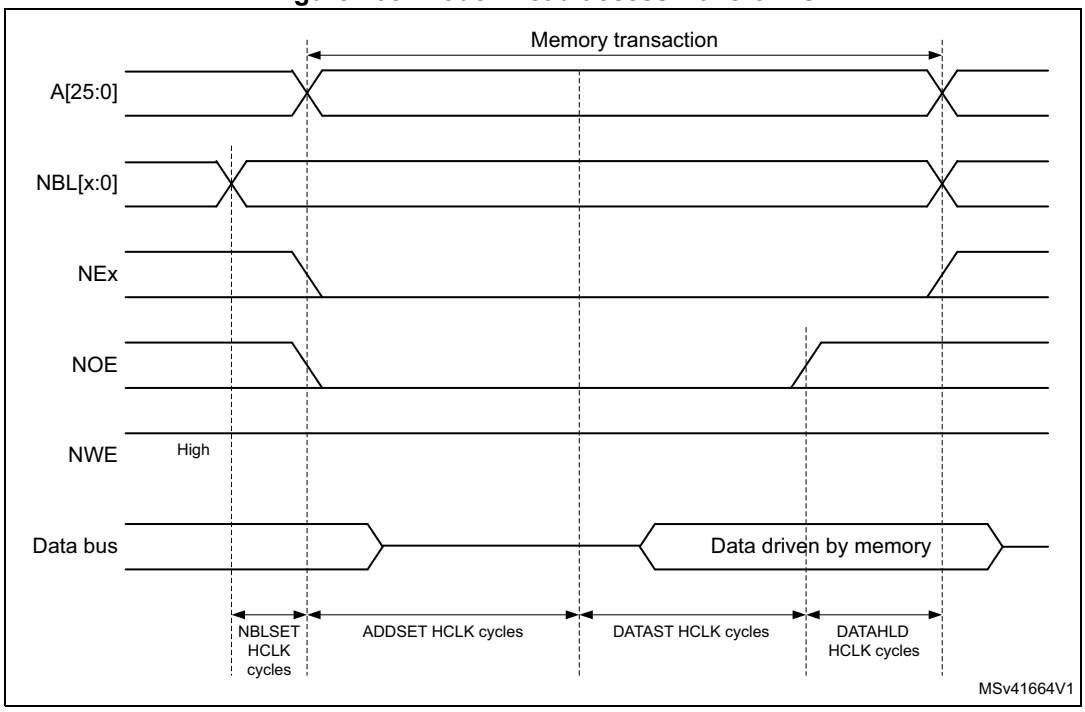
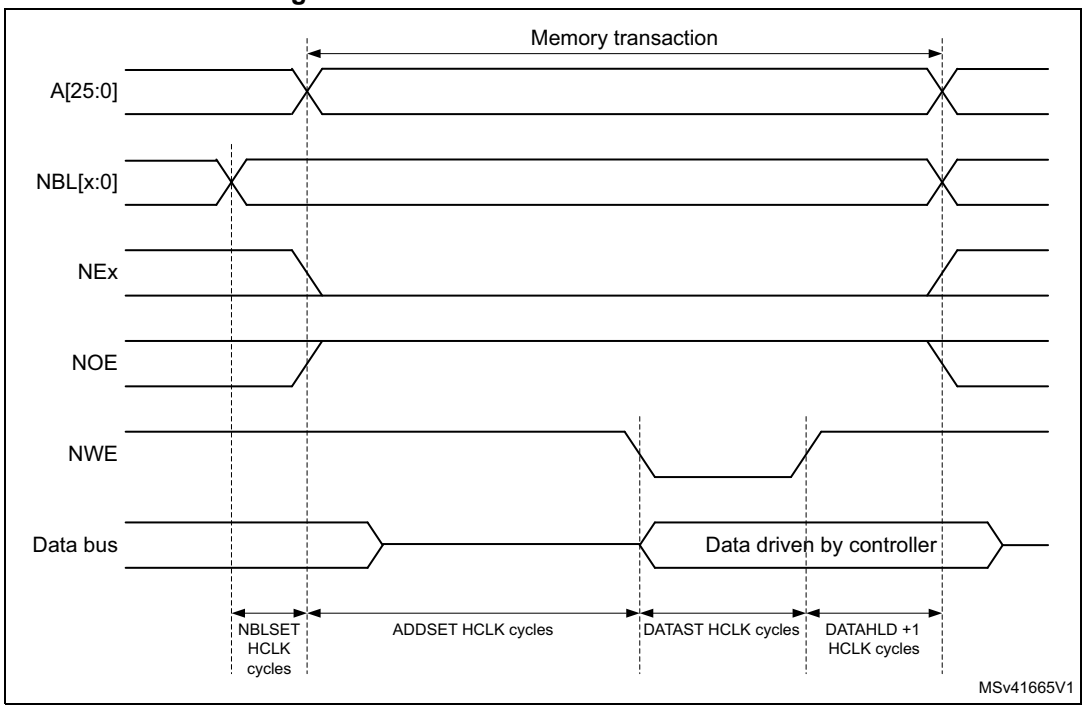


Figure 109. Mode 1 write access waveforms



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 190. FMC_BCRx bitfields (mode 1)

Bit number	Bit name	Value to set
31	FMCEN	0x1
30:24	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

Table 190. FMC_BCRx bitfields (mode 1) (continued)

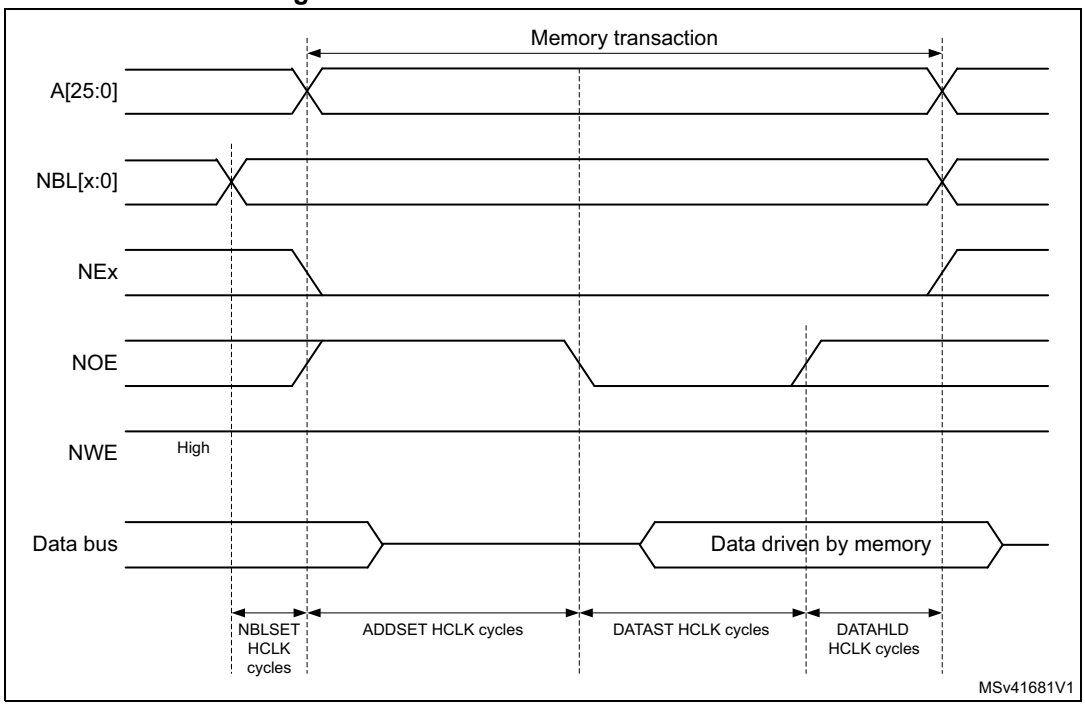
Bit number	Bit name	Value to set
5:4	MWID	As needed
3:2	MTYP	As needed, exclude 0x2 (NOR Flash memory)
1	MUXE	0x0
0	MBKEN	0x1

Table 191. 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.

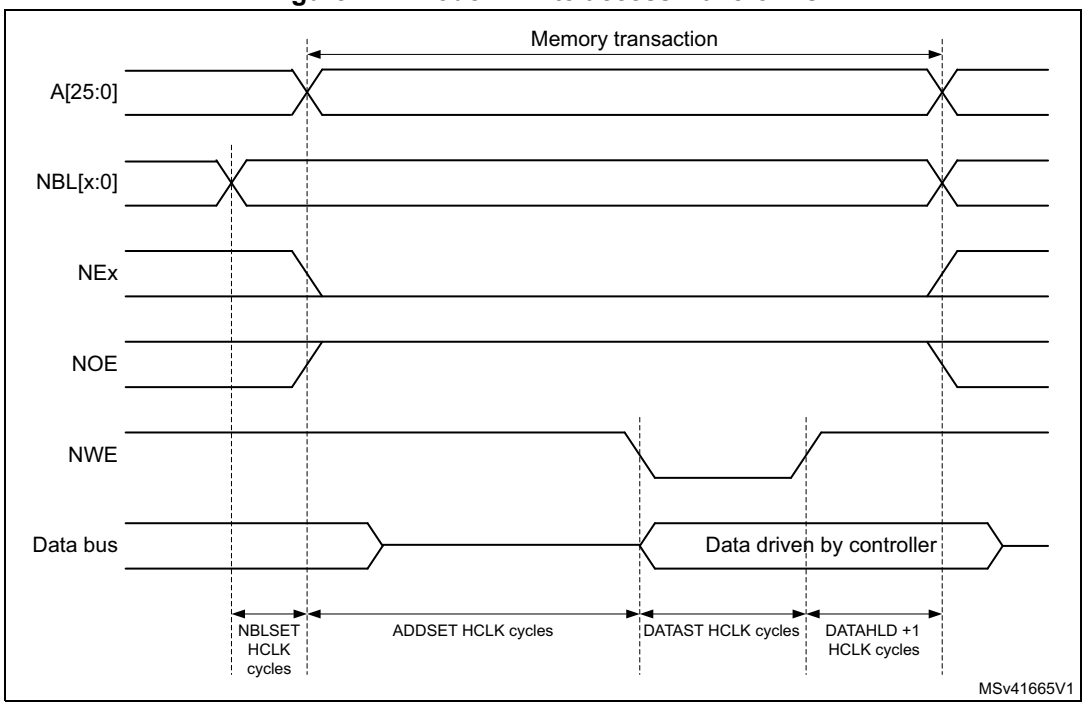
Mode A - SRAM/FRAM/PSRAM (CRAM) OE toggling

Figure 110. Mode A read access waveforms



1. NBL[1:0] are driven low during the read access

Figure 111. Mode A write access waveforms



The differences compared with Mode 1 are the toggling of NOE and the independent read and write timings.

Table 192. FMC_BCRx bitfields (mode A)

Bit number	Bit name	Value to set
31	FMCEN	0x1
30:24	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
3:2	MTYP	As needed, exclude 0x2 (NOR Flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 193. 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 194. 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 112. Mode 2 and mode B read access waveforms

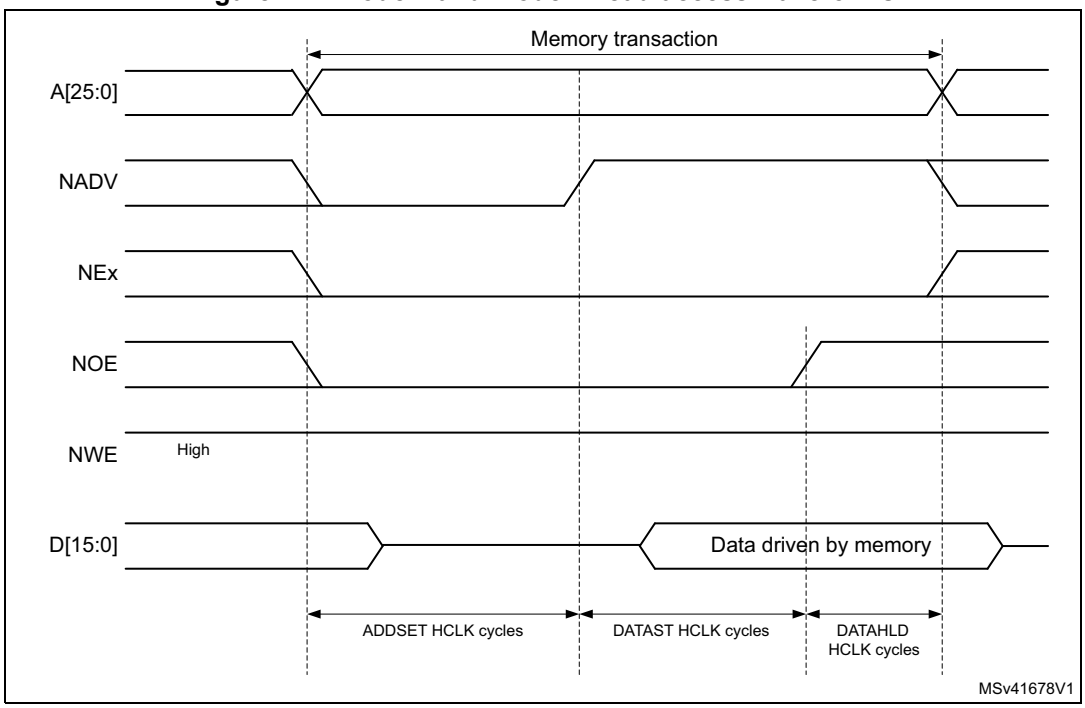


Figure 113. Mode 2 write access waveforms

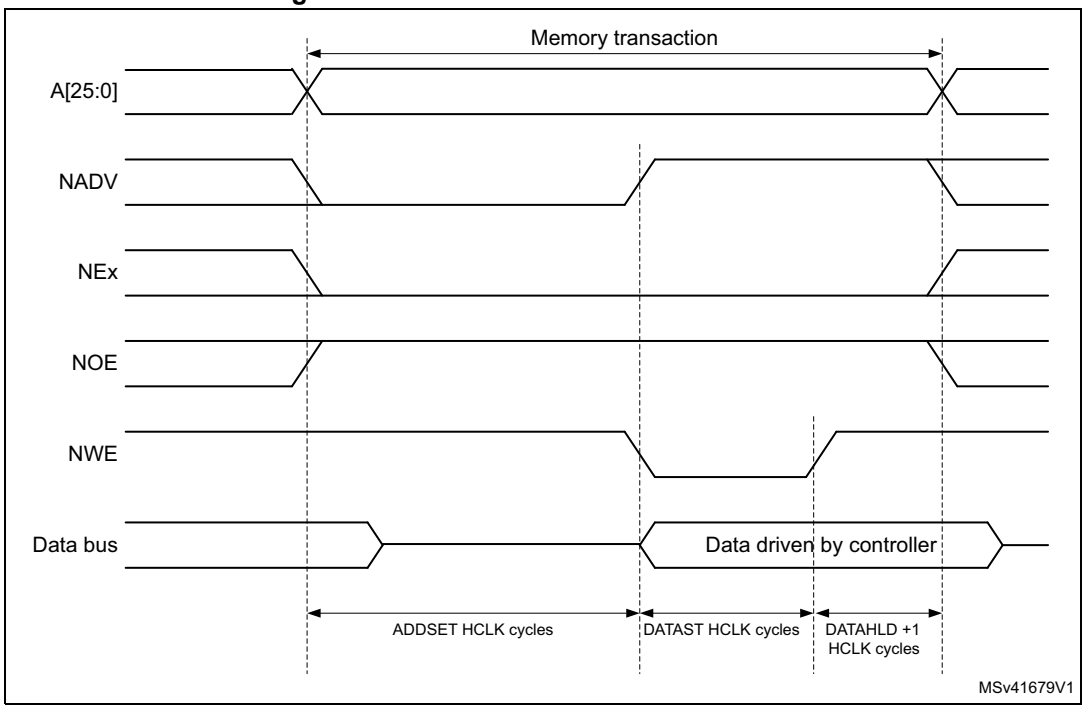
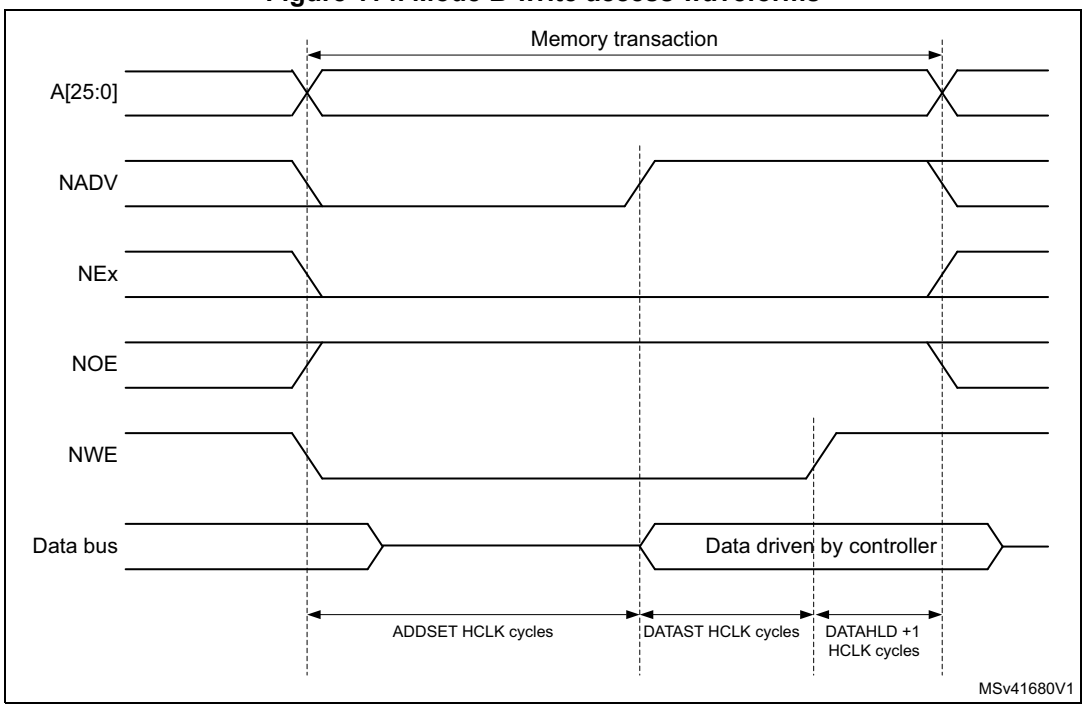


Figure 114. Mode B write access waveforms



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 195. FMC_BCRx bitfields (mode 2/B)

Bit number	Bit name	Value to set
31	FMCEN	0x1
30:24	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
3:2	MTYP	0x2 (NOR Flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 196. 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 197. 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 115. Mode C read access waveforms

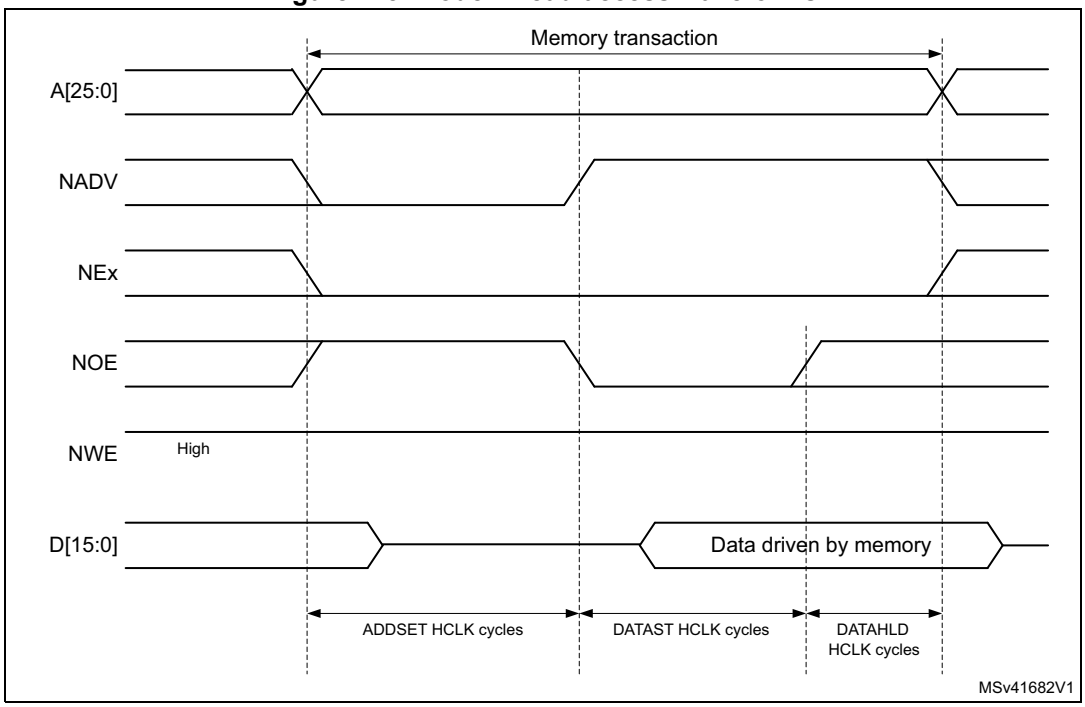
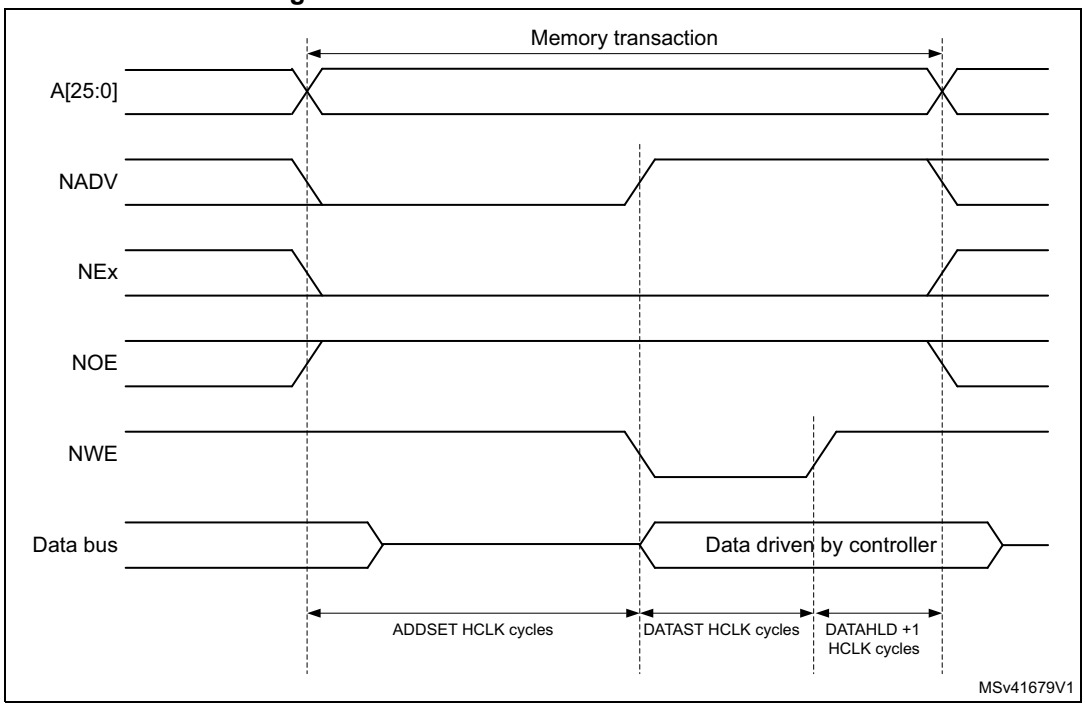


Figure 116. Mode C write access waveforms



The differences compared with mode 1 are the toggling of NOE and the independent read and write timings.

Table 198. FMC_BCRx bitfields (mode C)

Bit number	Bit name	Value to set
31	FMCEN	0x1
30:24	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

Table 198. FMC_BCRx bitfields (mode C) (continued)

Bit number	Bit name	Value to set
5:4	MWID	As needed
3:2	MTYP	0x02 (NOR Flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 199. 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 200. 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 117. Mode D read access waveforms

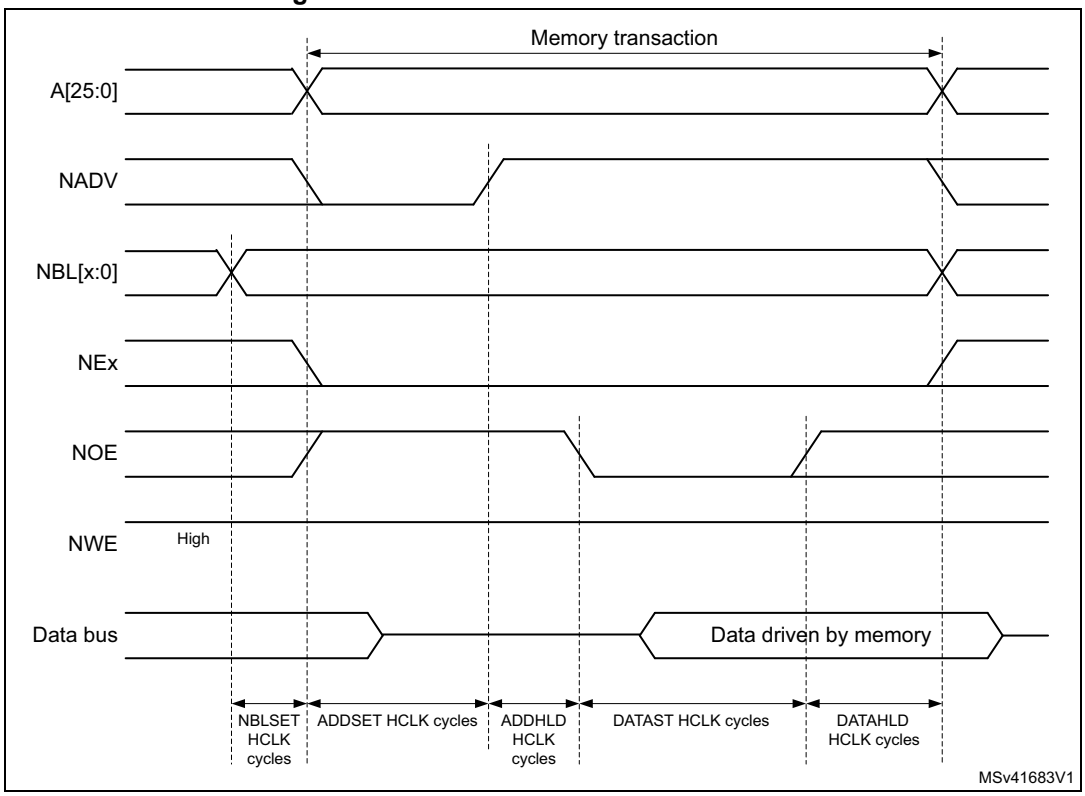
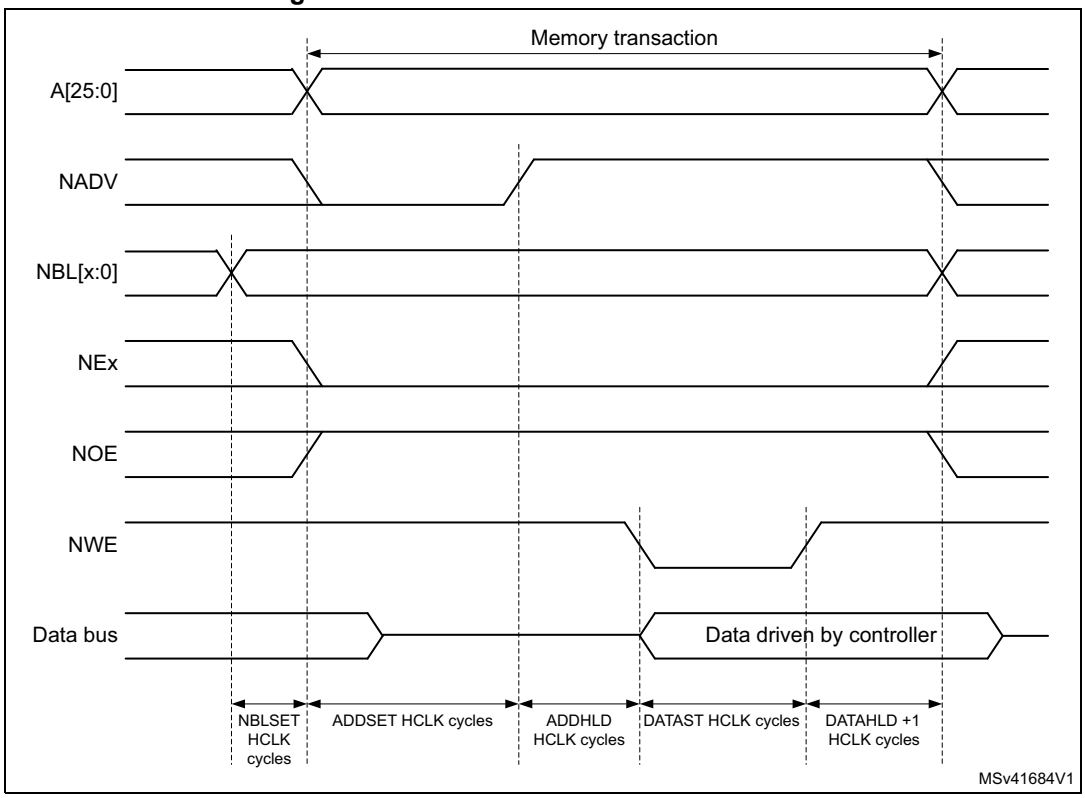


Figure 118. Mode D write access waveforms



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 201. FMC_BCRx bitfields (mode D)

Bit number	Bit name	Value to set
31	FMCEN	0x1
30:24	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

Table 201. FMC_BCRx bitfields (mode D) (continued)

Bit number	Bit name	Value to set
7	Reserved	0x1
6	FACCEN	Set according to memory support
5:4	MWID	As needed
3:2	MTYP	As needed
1	MUXEN	0x0
0	MBKEN	0x1

Table 202. 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 203. 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 119. Muxed read access waveforms

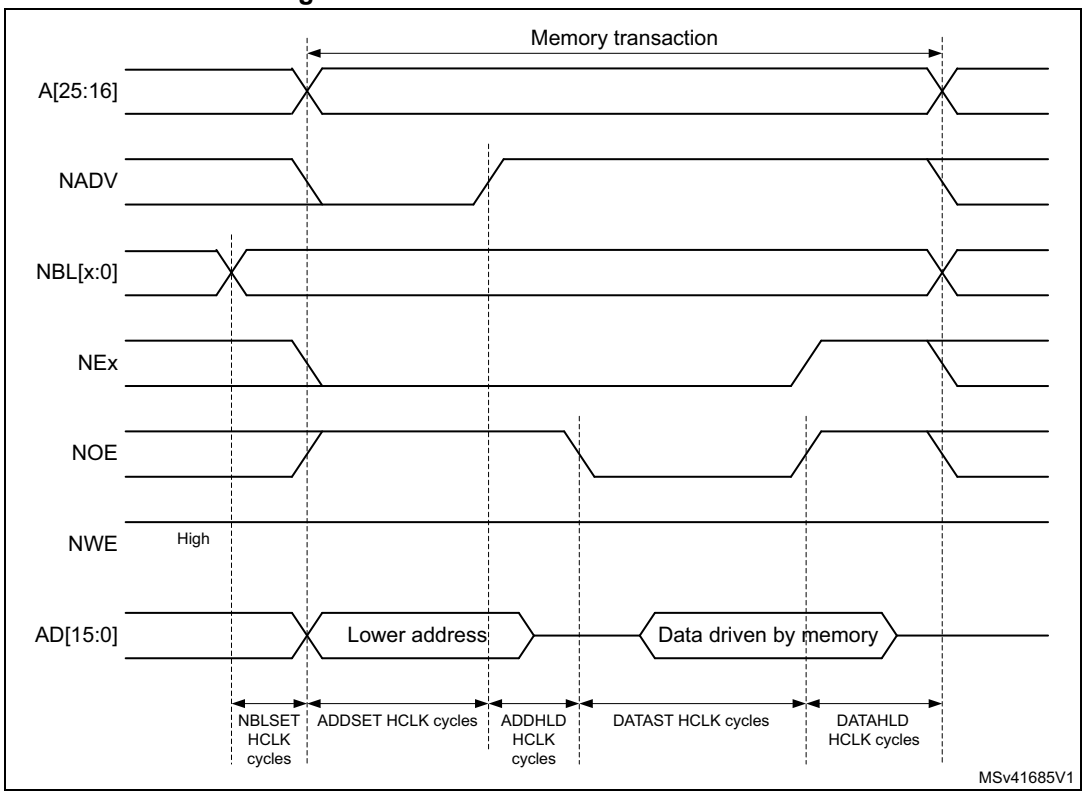
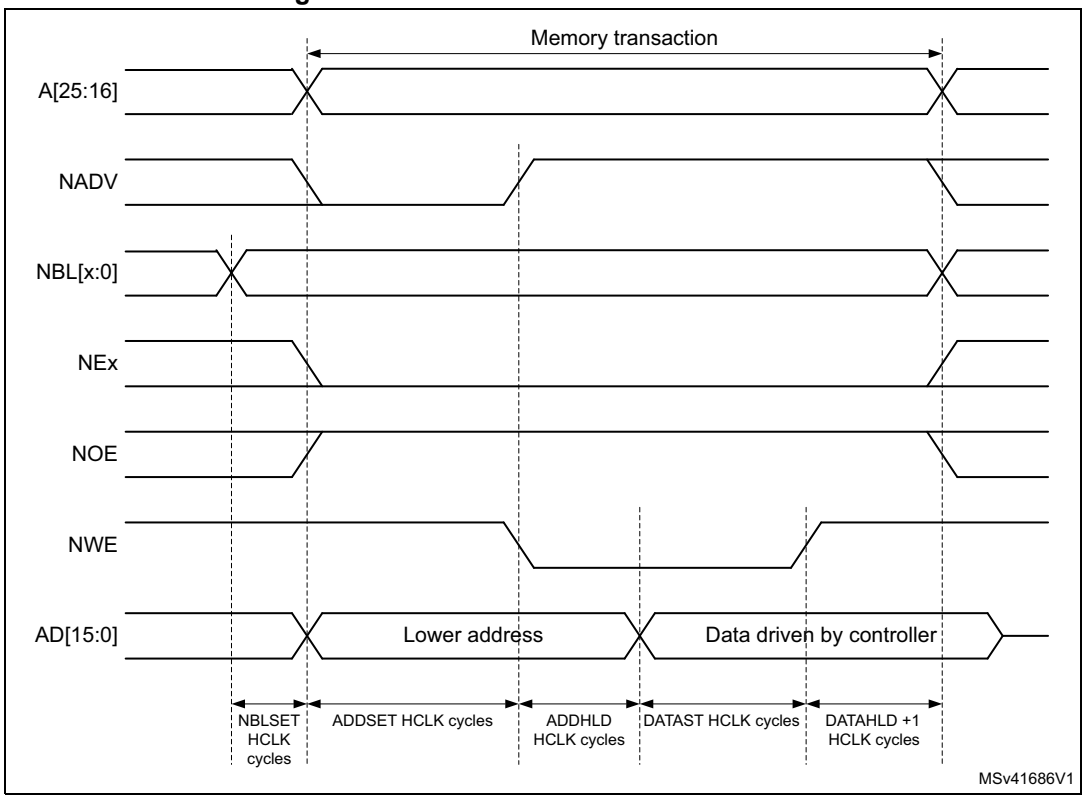


Figure 120. Muxed write access waveforms



The difference with mode D is the drive of the lower address byte(s) on the data bus.

Table 204. FMC_BCRx bitfields (Muxed mode)

Bit number	Bit name	Value to set
31	FMCEN	0x1
30:24	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

Table 204. FMC_BCRx bitfields (Muxed mode) (continued)

Bit number	Bit name	Value to set
6	FACCEN	0x1
5:4	MWID	As needed
3:2	MTYP	0x2 (NOR Flash memory) or 0x1(PSRAM)
1	MUXEN	0x1
0	MBKEN	0x1

Table 205. 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:

$$DATAST \geq (4 \times HCLK) + (max\_wait\_assertion\_time - address\_phase - hold\_phase)$$

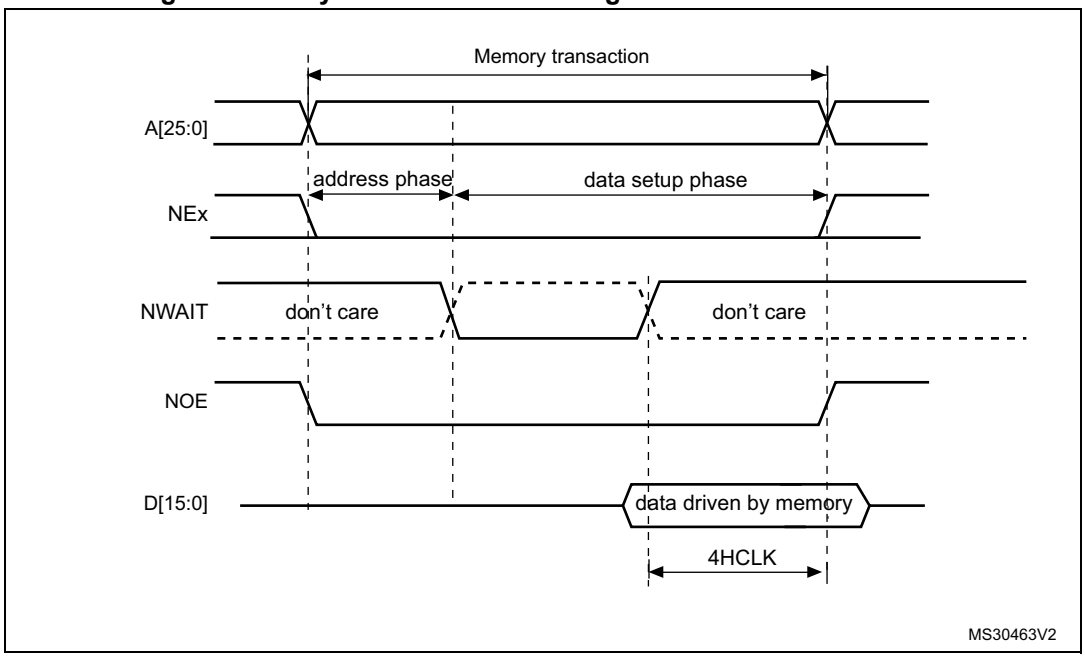
   otherwise

$$DATAST \geq 4 \times HCLK$$

   where max_wait_assertion_time is the maximum time taken by the memory to assert the WAIT signal once NEx/NOE/NWE is low.

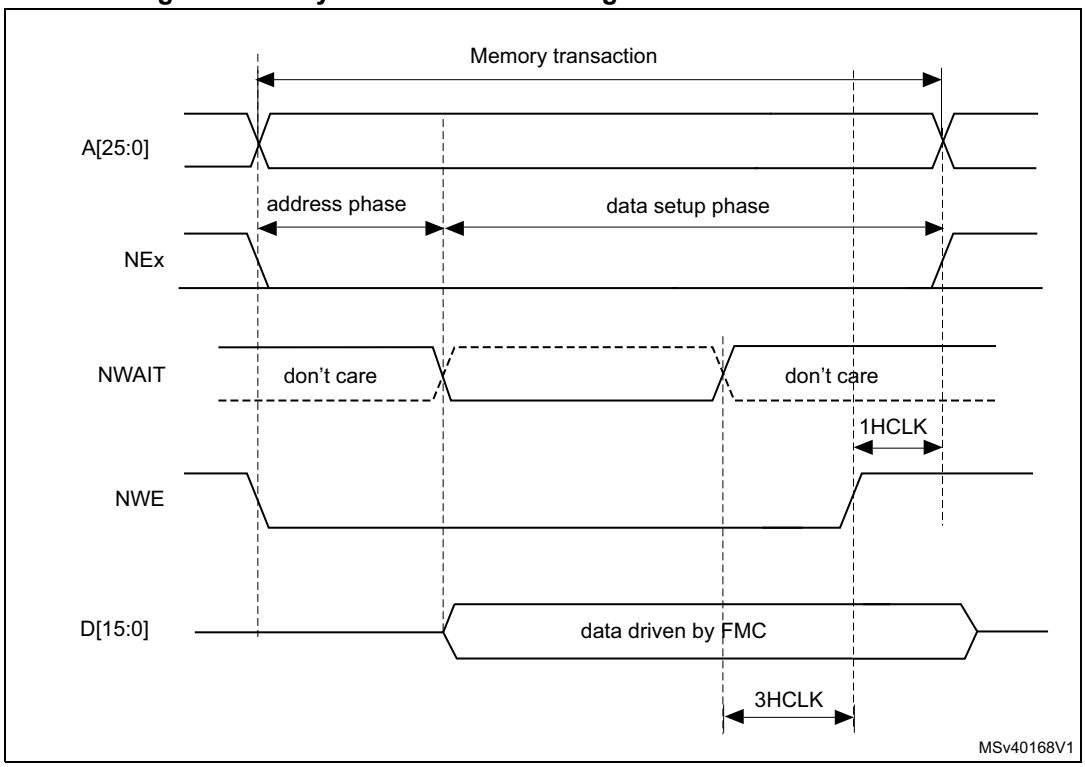
Figure 121 and *Figure 122* show the number of HCLK clock cycles that are added to the memory access phase after WAIT is released by the asynchronous memory (independently of the above cases).

Figure 121. Asynchronous wait during a read access waveforms



1. NWAIT polarity depends on WAITPOL bit setting in FMC_BCRx register.

Figure 122. 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

25.6.5 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 is used 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 123. Wait configuration waveforms

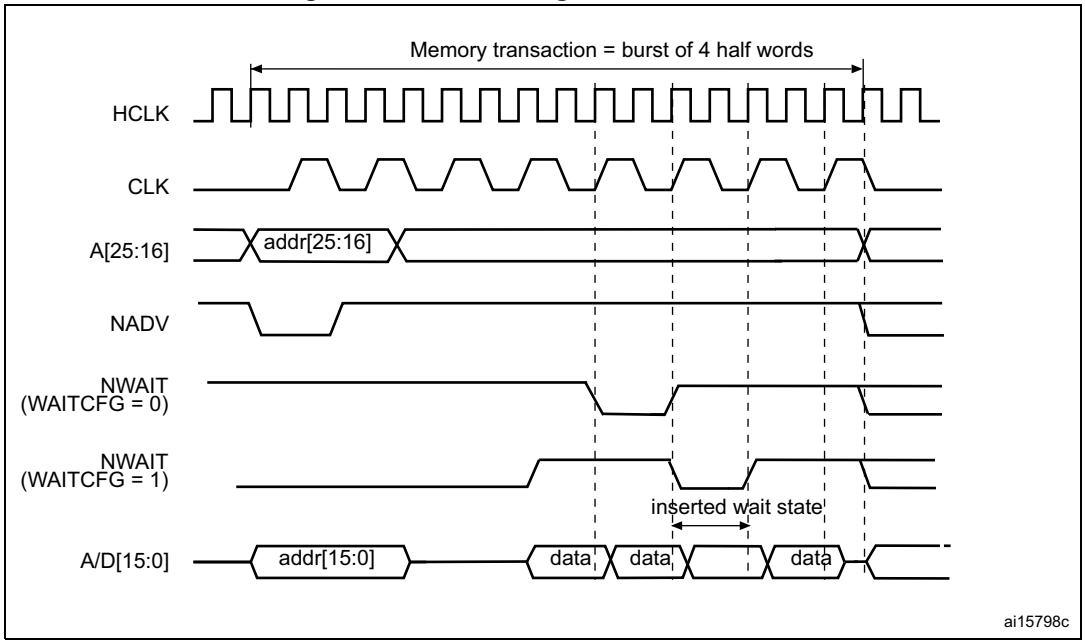
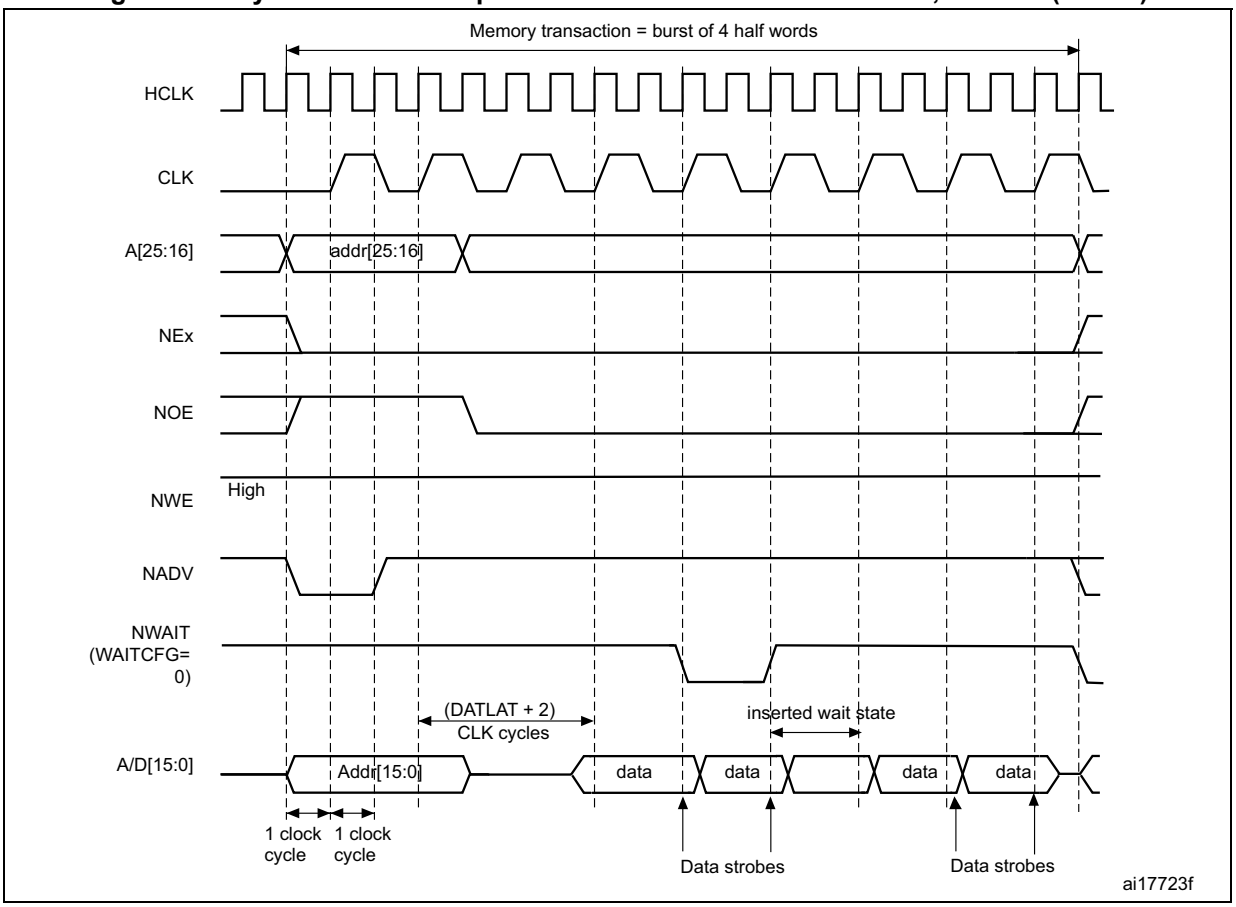


Figure 124. Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM)



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 206. FMC_BCRx bitfields (Synchronous multiplexed read mode)

Bit number	Bit name	Value to set
31	FMCEN	0x1
30:24	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

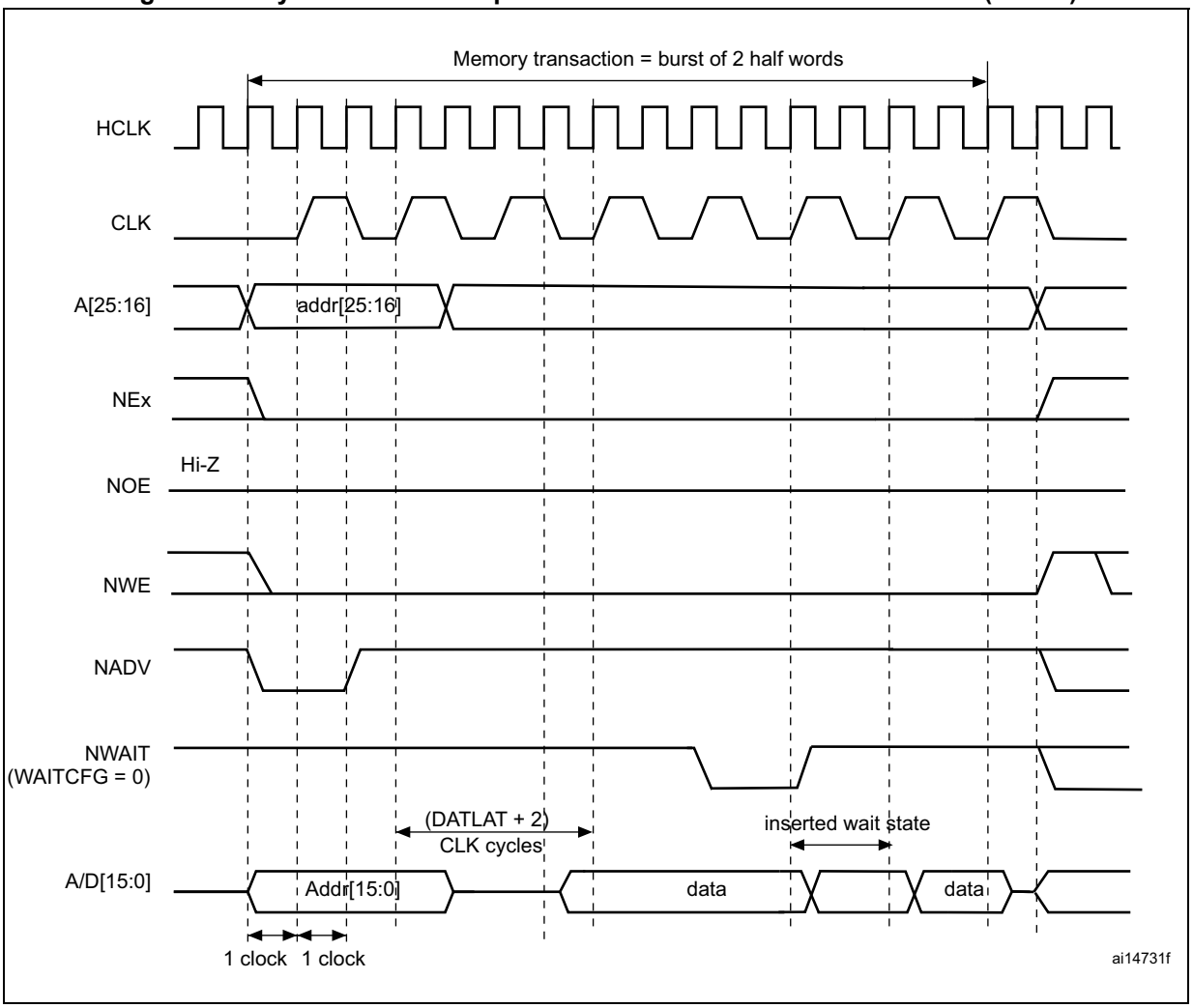
Table 206. FMC_BCRx bitfields (Synchronous multiplexed read mode) (continued)

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 207. 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 125. 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 208. FMC_BCRx bitfields (Synchronous multiplexed write mode)

Bit number	Bit name	Value to set
31	FMCEN	0x1
30:24	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

Table 208. FMC_BCRx bitfields (Synchronous multiplexed write mode) (continued)

Bit number	Bit name	Value to set
13	WAITEN	To be set to 1 if the memory supports this feature, to be kept at 0 otherwise.
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 209. 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

25.6.6 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
FMCEN	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	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

Bit 31 **FMCEN**: FMC controller enable

This bit enables or disables the FMC controller.

0: Disable the FMC controller

1: Enable the FMC controller

Note: The FMCEN bit of the FMC_BCR2..4 registers is don't care. It is only enabled through the FMC_BCR1 register.

Bits 30:24 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 108* to *Figure 120*), 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 25.6.5: 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 is used 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 108* to *Figure 120*), 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 108* to *Figure 120*).

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 108* to *Figure 120*), 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 108* to *Figure 120*).

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 108* to *Figure 120*), 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 108* to *Figure 120*).

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 108* to *Figure 120*), 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 108* to *Figure 120*), 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 117* to *Figure 120*), 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 108* to *Figure 120*), 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.

25.7 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 25.7.7*). The programmable memory parameters include access timings (shown in *Table 210*) and ECC configuration.

Table 210. 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
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

25.7.1 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 211. 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)

Table 211. 8-bit NAND Flash (continued)

FMC signal name	I/O	Function
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 212. 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.

25.7.2 NAND Flash supported memories and transactions

Table 213 shows the supported devices, access modes and transactions. Transactions not allowed (or not supported) by the NAND Flash controller are shown in gray.

Table 213. 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

Table 213. Supported memories and transactions (continued)

Device	Mode	R/W	AHB data size	Memory data size	Allowed/ not allowed	Comments
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

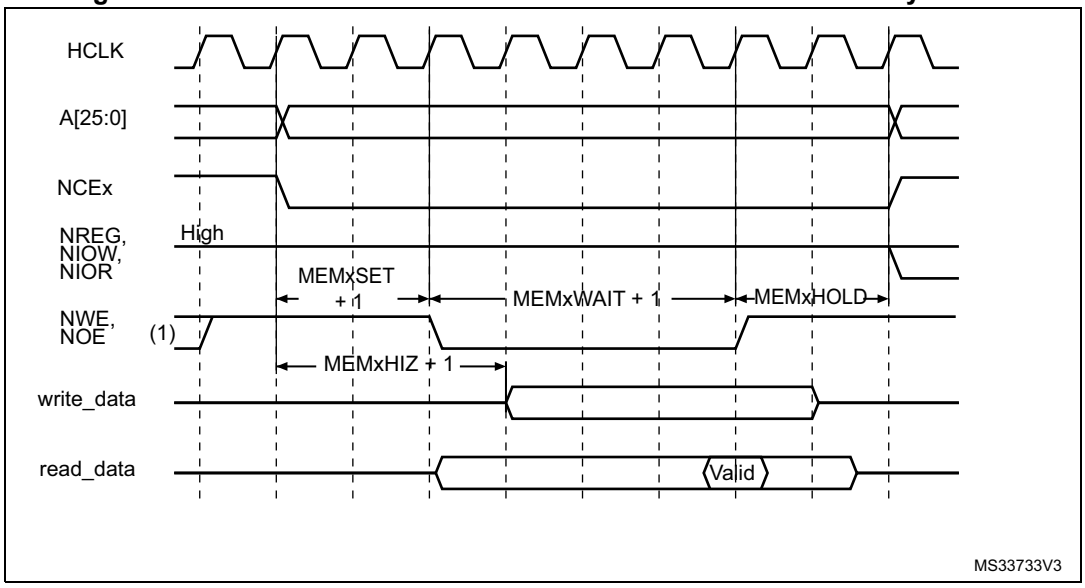
25.7.3 Timing diagrams for NAND Flash memory

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 126* shows the timing parameter definitions for common memory accesses, knowing that Attribute memory space access timings are similar.

Figure 126. 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.

25.7.4 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.

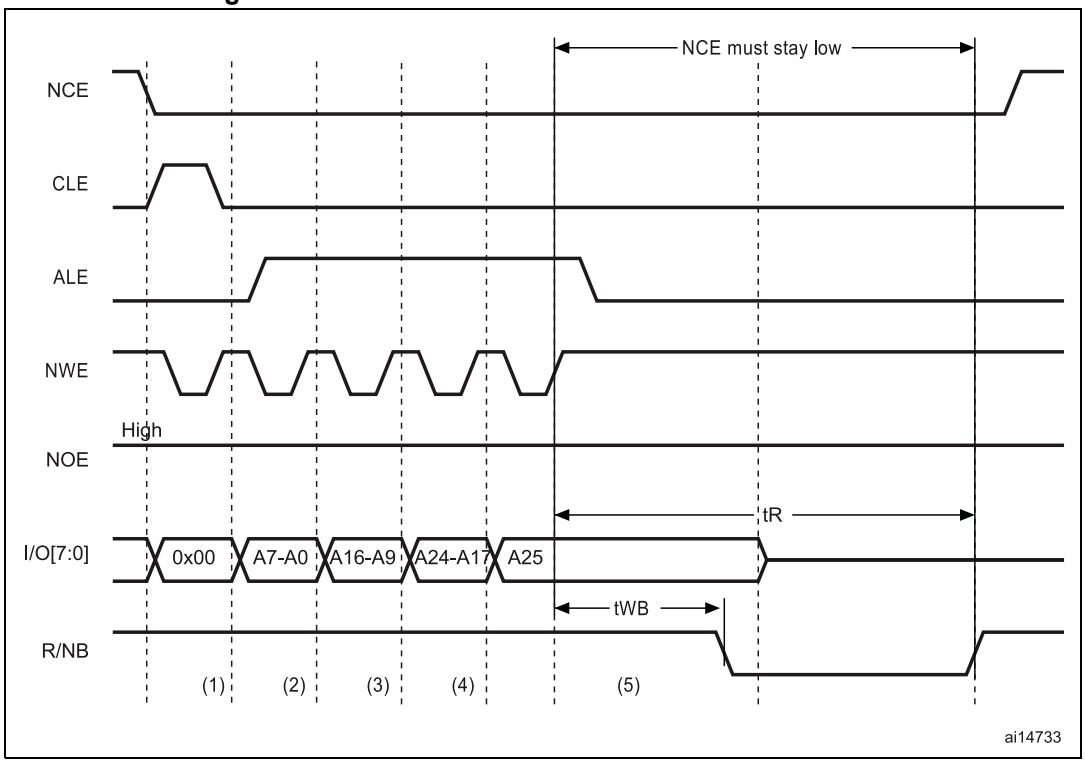
A typical page read operation from the NAND Flash device requires the following steps:

1. Program and enable the corresponding memory bank by configuring the FMC_PCR and FMC_PMEM (and for some devices, FMC_PATT, see *Section 25.7.5: 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 25.5.2: 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 25.7.5: 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

25.7.5 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 127*).

Figure 127. 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.

25.7.6 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.

25.7.7 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 25.7.5: 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 25.7.6: 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 214* describes the contents of these bitfields.

Table 214. 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]

25.7.8 FMC register map

Table 215. 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**	FMCEN	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	0	1	1	0		0	0		1	0	1	1	0	1	1
0x08	**FMC_BCR2**	FMCEN	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	0	1	1	0		0	0		1	0	1	0	0	1	0
0x10	**FMC_BCR3**	FMCEN	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	0	1	1	0		0	0		1	0	1	0	0	1	0
0x18	**FMC_BCR4**	FMCEN	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	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 215. 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 115* for the register boundary addresses.

26 Octo-SPI interface (OCTOSPI)

26.1 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

26.2 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 memory 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 protocol support
- 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.

26.3 OCTOSPI implementation

Table 216. OCTOSPI implementation

OCTOSPI feature	OCTOSPI1/2
HyperBus standard compliant	X
Xcella standard compliant	X
XSPI (JEDEC251ES) standard compliant	X
AMBA® AHB compliant data interface	X
Dual AHB 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
Compliant with dual-OCTOSPI arbiter (communication regulation)	X
Extended CSHT timeout	X
Memory-mapped write	X
Refresh counter	X
GPDMA interface	X

26.4 OCTOSPI functional description

26.4.1 OCTOSPI block diagram

Figure 128. OCTOSPI block diagram in octal configuration

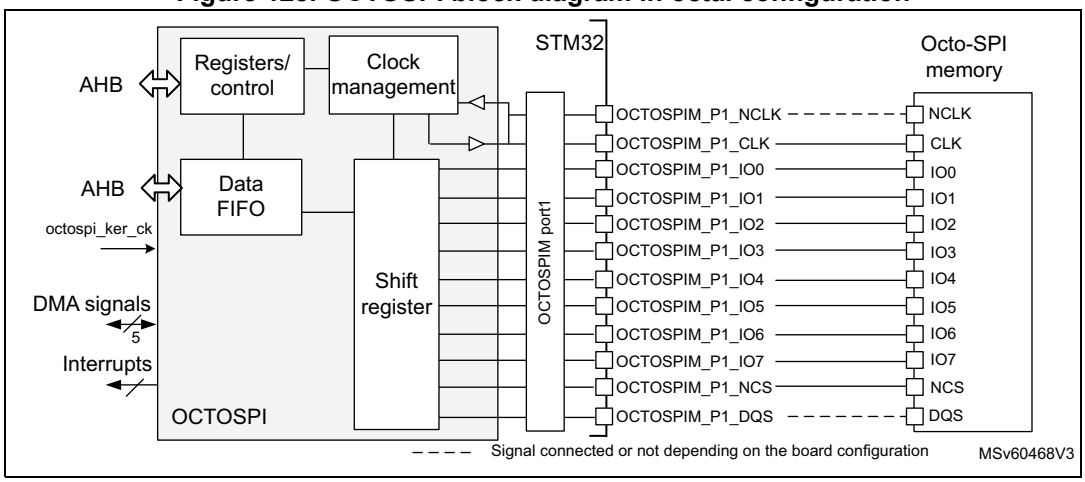


Figure 129. OCTOSPI block diagram in quad configuration

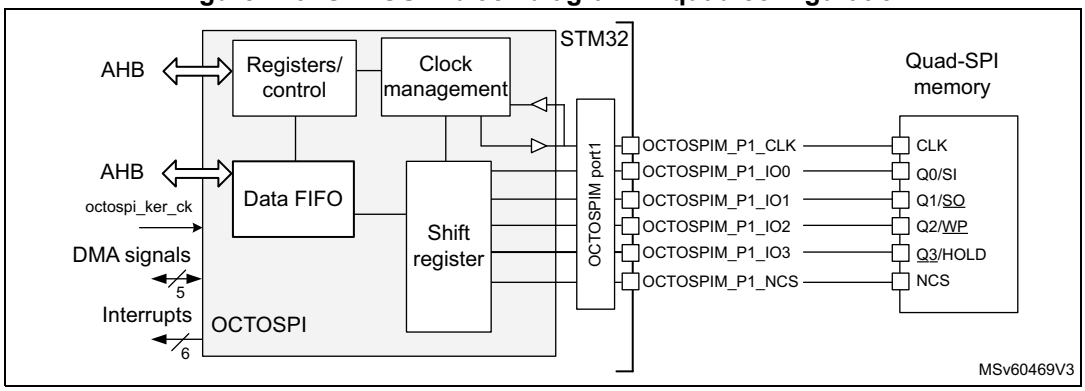
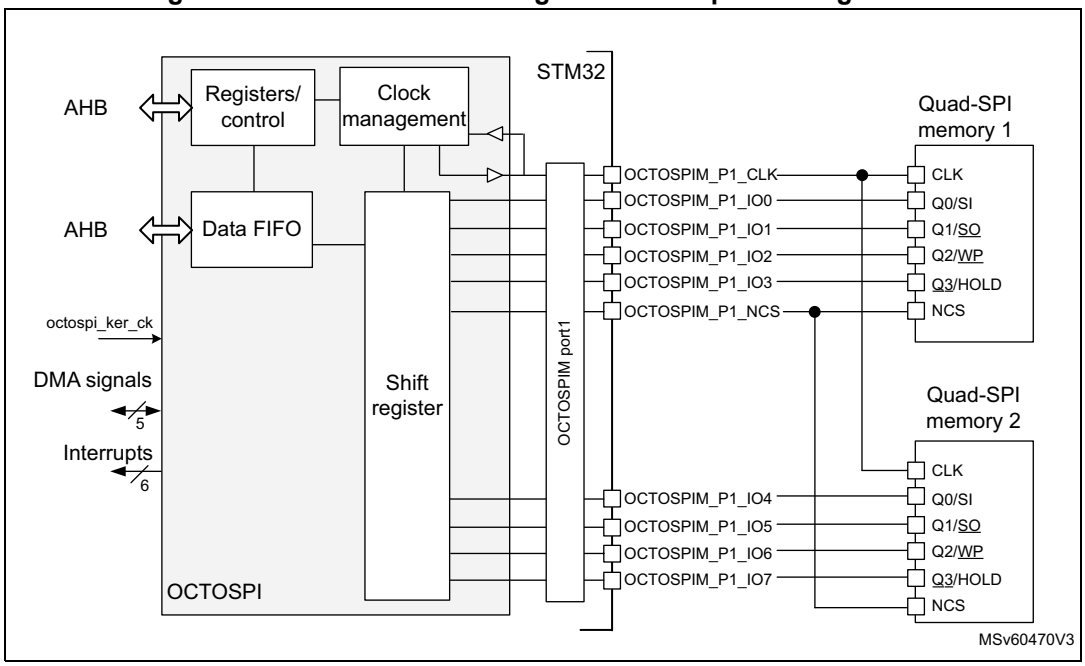


Figure 130. OCTOSPI block diagram in dual-quad configuration



26.4.2 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
  - data bus can be used as possible remap for quad-SPI mode

26.4.3 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:

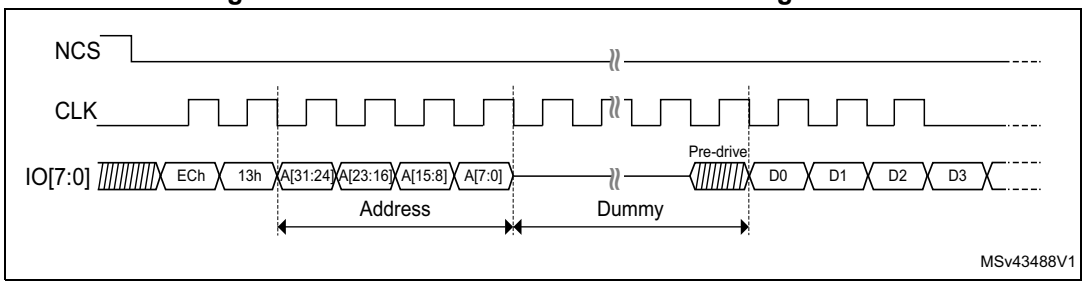
- Instruction phase
- Address phase
- Alternate-byte phase
- Dummy-cycle phase
- Data phase

Any of these phases can be configured to be skipped but, in case of single-phase command, the only use case supported is instruction-phase-only.

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 131. 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 (over IO0, SO single 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 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 IO0, SO single 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 IO0, SO single 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 5:4 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 IO0, SO single 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.

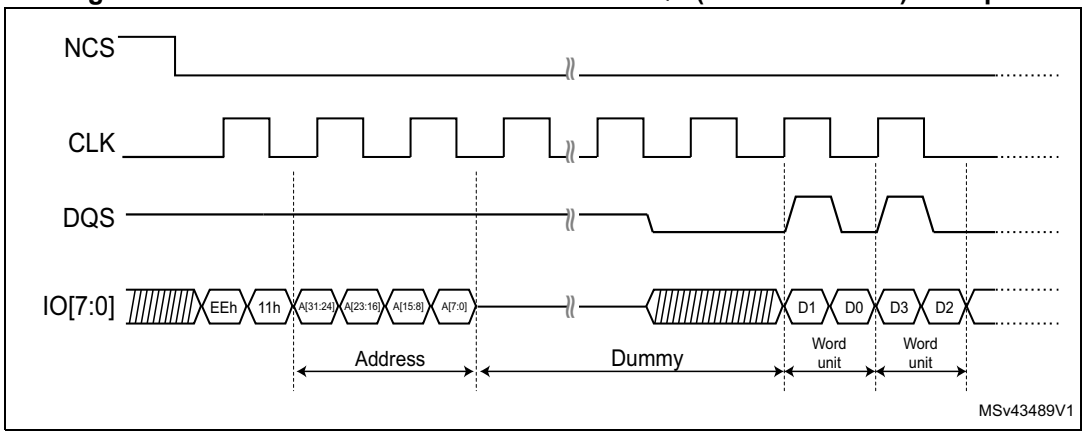
In Memory-mapped mode, the data format for the write operation is specified in OCTOSPI_WCCR. The data format for the read operation is specified in OCTOSPI_CCR.

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 132. DTR read in octal-SPI mode with DQS (Macronix mode) example



26.4.4 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-SPI 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.

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.

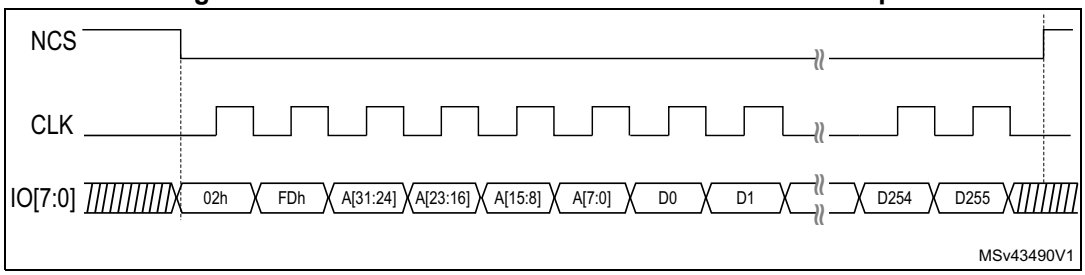
Single-data rate (SDR) mode

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 133. SDR write command in octo-SPI mode example



Note: Due to internal synchronization, up to six extra dummy clock cycles may be generated by the Octo-SPI interface after the last data is read.

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.

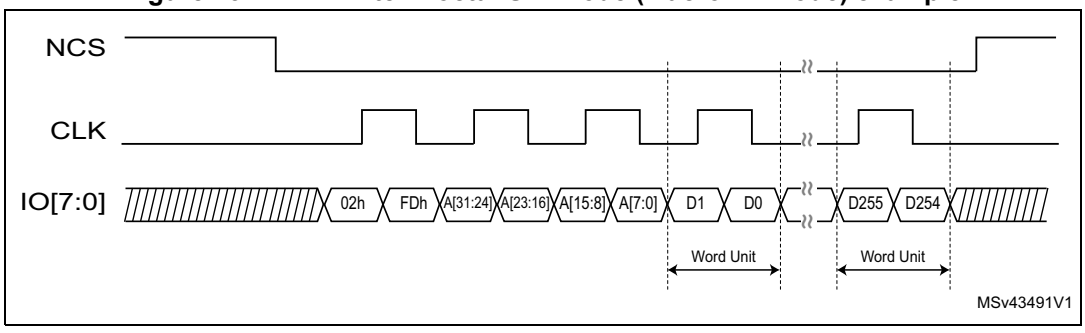
In DTR mode, when the OCTOSPI drives the IO0/SO and IO1to IO7 signals in the instruction, address, and alternate-byte phases, a bit is sent or received on each of the falling and rising edges of CLK.

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 134. DTR write in octal-SPI mode (Macronix mode) example



Note: Due to internal synchronization, up to six extra dummy clock cycles may be generated by the Octo-SPI interface after the last data is read.

Dual-quad configuration

When DMM = 1 in OCTOSPI_CR, the OCTOSPI is in dual-memory configuration: if DMODE = 011, 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.

26.4.5 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 (to support this mode, the device must be powered with 1.8 V)

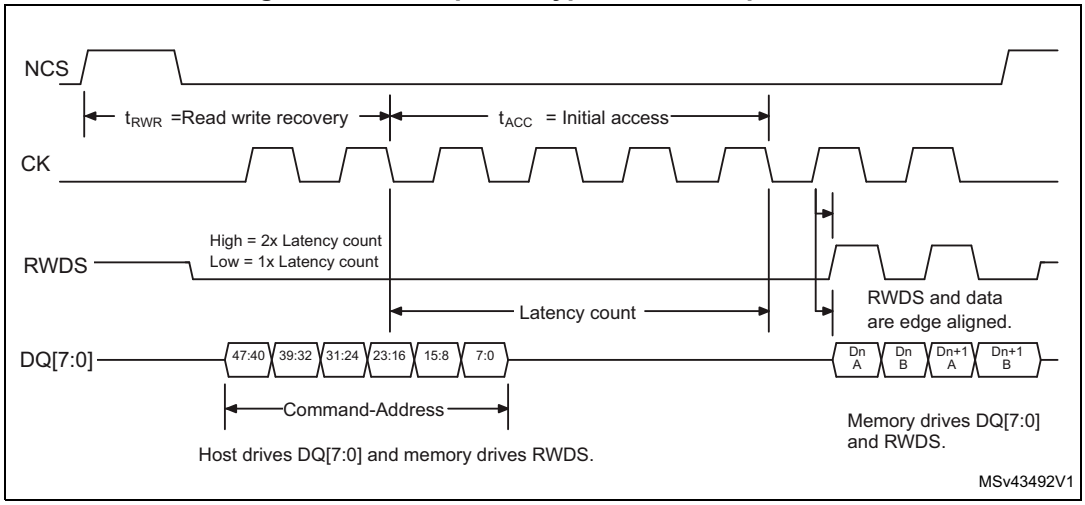
The HyperBus does not require any command specification nor any alternate bytes. As a consequence, a separate register set is used to define the timing of the transaction.

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 135. Example of HyperBus read operation



Note: Due to internal synchronization, up to six extra dummy clock cycles may be generated by the Octo-SPI interface after the last data is read.

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 217. 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)

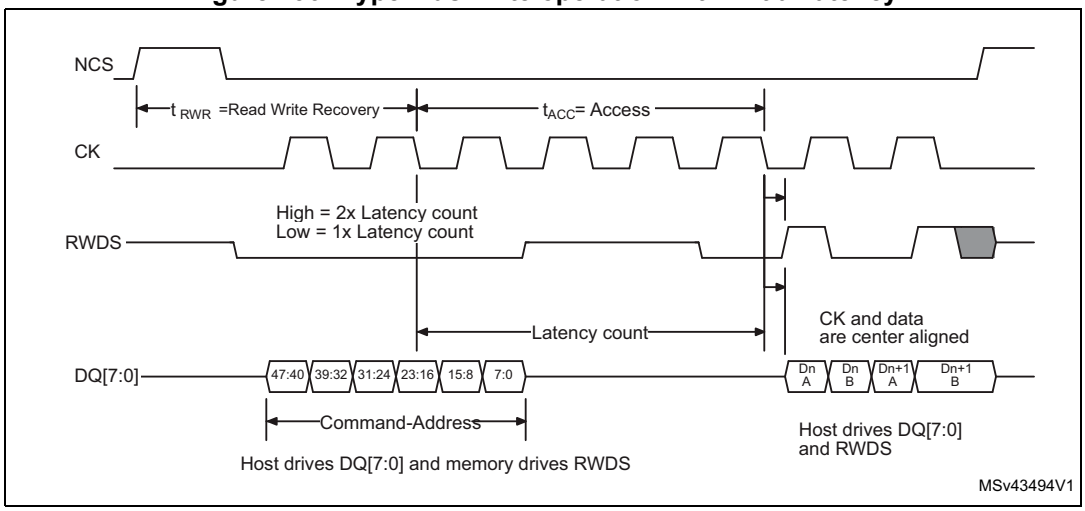
During the read operation, the RWDS is used by the device, in two ways (see *Figure 135*):

- 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 136. 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 137. HyperBus read operation with additional latency

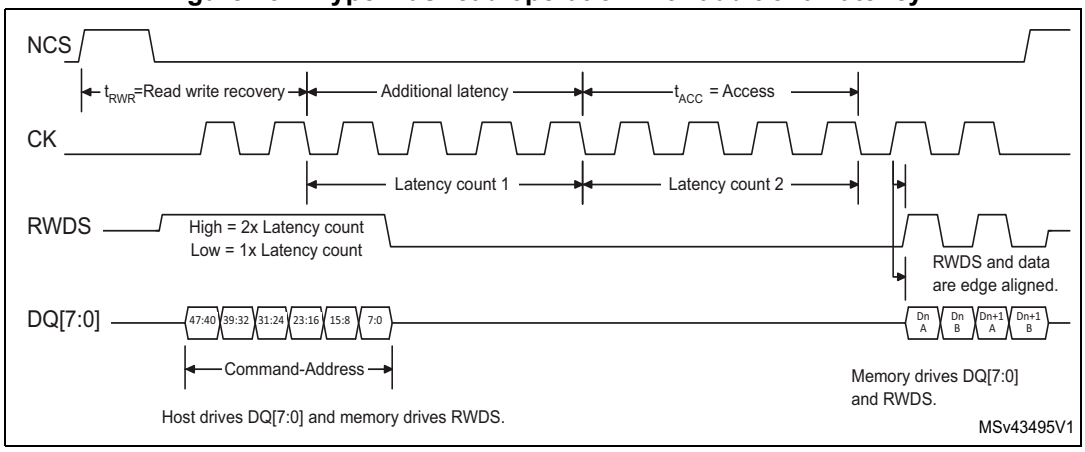
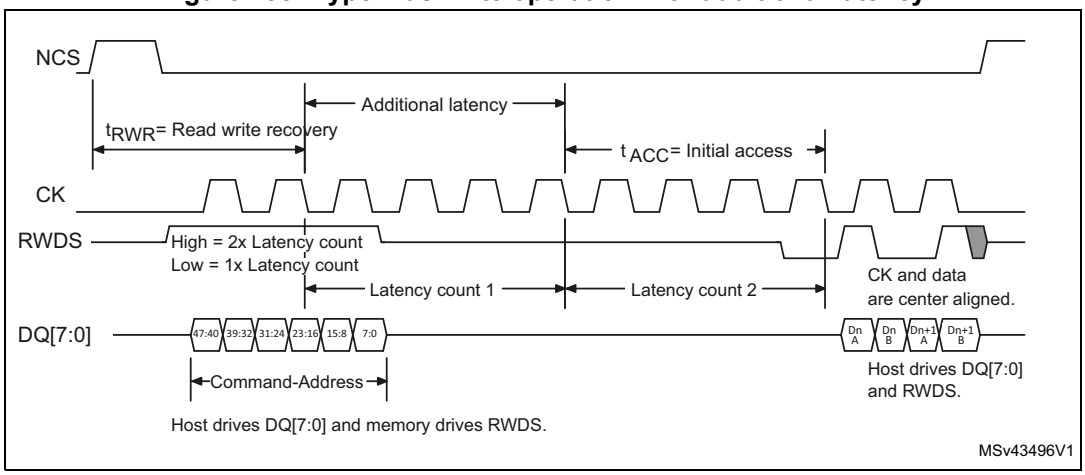


Figure 138. HyperBus write operation with additional latency



Fixed-latency mode

Some devices or some applications may not want to operate with a variable latency time as described above.

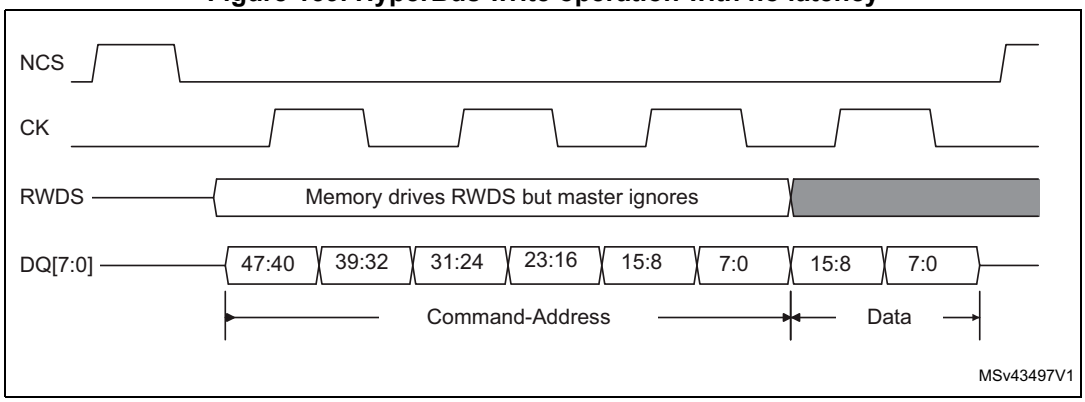
The latency can be forced to 2 x t_{ACC} by setting LM of OCTOSPI_HLCR.

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 139. HyperBus write operation with no latency

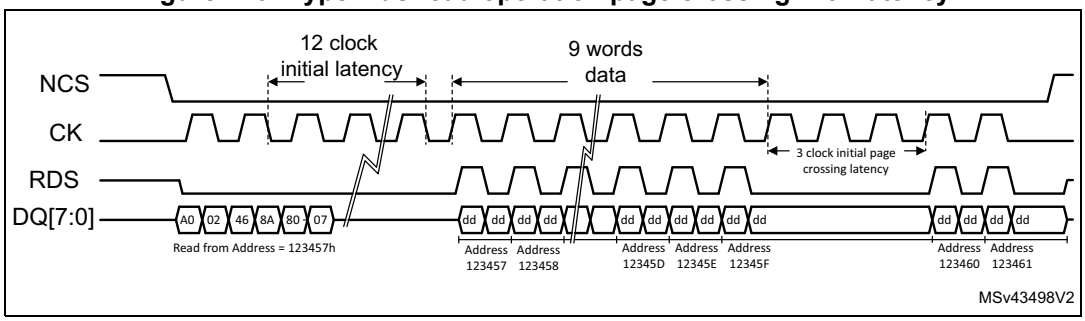


Latency on page-crossing during the read operations

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 140. HyperBus read operation page crossing with latency



26.4.6 Specific features

The OCTOSPI supports some specific features, such as:

- Wrap support
- NCS boundary and refresh
- Communication regulation

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. 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.

Communication regulation

The communication regulation feature limits the maximum length of a transaction to the value programmed in MAXTRAN[7:0] of OCTOSPI_DCR3.

If the number of clock cycles reach the MAXTRAN + 1 value, and if the second OCTOSPI requests an access, the NCS is released and a new transaction is issued to address the next data. If the second OCTOSPI does not request an access, the transaction is not stopped and the NCS is not released.

If MAXTRAN[7:0] = 0, no limitation occurs.

The MAXTRAN[7:0] value must be greater than the minimal transaction size in terms of number of cycles including the command, address, alternate, and dummy phases.

Note: The communication regulation feature cannot be used in write mode for the Flash memory devices that require extra command to re-enable the write operation after the NCS is active again.

If NCS boundary, refresh and communication regulation 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.

26.4.7 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

26.4.8 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] = 00 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, a 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.

26.4.9 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.

26.4.10 OCTOSPI Memory-mapped mode

When configured in Memory-mapped mode, the external SPI device is seen as an internal memory.

Note: No 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.

26.4.11 OCTOSPI configuration introduction

The OCTOSPI configuration is done in three steps:

1. OCTOSPI system configuration
2. OCTOSPI device configuration
3. OCTOSPI mode configuration

26.4.12 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-memory configuration, if needed, with DMM

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.

26.4.13 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[5:0]
- 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[5:0] 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.

26.4.14 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 configuration operations must be completed (ensured by checking BUSY = 0) before the first access to the memory area: any register write operation when BUSY = 1 have no effect and is not signaled with an error response. On the first access, the OCTOSPI becomes busy, and no further configuration is allowed. Then, the only way to get BUSY low is to clear the ENABLE bit or to abort by setting the ABORT bit.

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 configurations where delay does not need to be compensated, the external delay block 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.

26.4.15 OCTOSPI HyperBus protocol configuration

Indirect mode configuration (HyperBus)

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. Make a write operation in OCTOSPI_IR and enable the DMA channel if needed.
3. Specify the targeted address in OCTOSPI_AR.
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 (HyperBus)

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 (HyperBus)

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. Then, the only way to get BUSY low is to clear the ENABLE bit or to abort by setting the ABORT bit.

26.4.16 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 in octal DTR mode or in dual-memory configuration.
- 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).

26.4.17 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.

26.4.18 OCTOSPI reconfiguration or deactivation

Prior to any OCTOSPI reconfiguration, the software must ensure that all the transactions are completed:

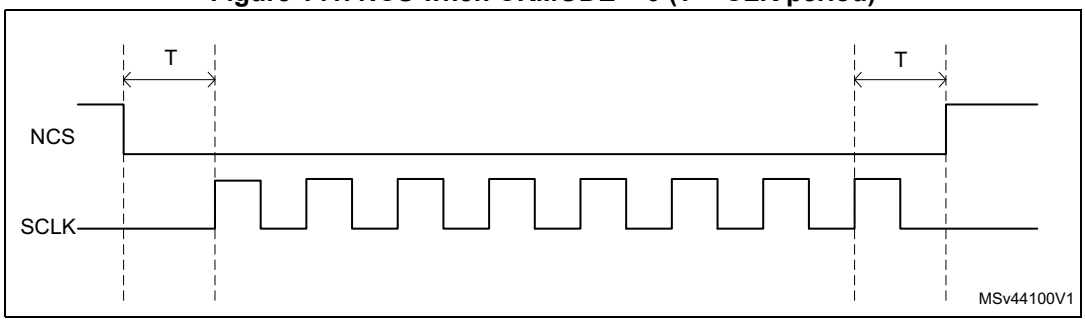
- After a Memory-mapped write, the software must perform a dummy read followed by a synchronization barrier, then an abort.
- After a Memory-mapped read, the software must perform a synchronization barrier then an abort.

26.4.19 NCS behavior

By default, NCS is high, deselecting the external device. NCS falls before an operation begins and rises as soon as it finishes.

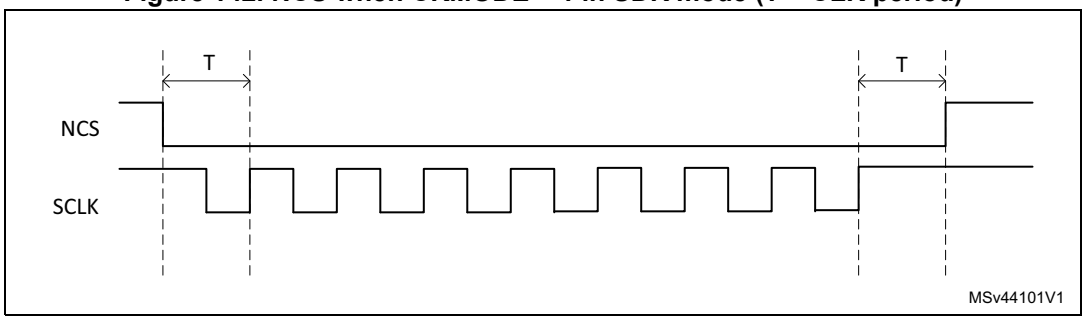
When CKMODE = 0 (clock mode 0: CLK stays low when no operation is in progress), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge (see the figure below).

Figure 141. NCS when CKMODE = 0 (T = CLK period)



When CKMODE = 1 (clock 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).

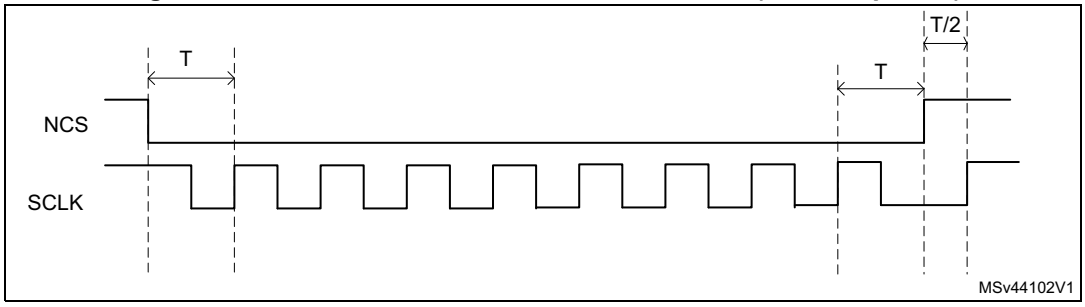
Figure 142. NCS when CKMODE = 1 in SDR mode (T = CLK period)



When the CKMODE = 1 (clock 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

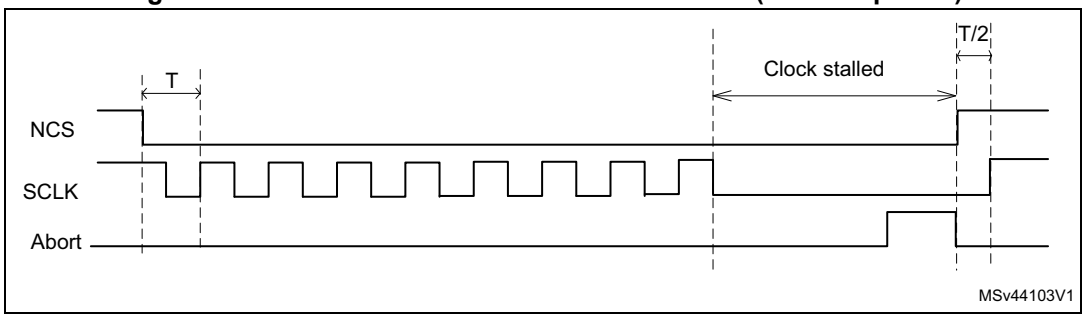
must finish with a falling edge, CLK is low when NCS rises, and CLK rises back up one half of a CLK cycle afterwards.

Figure 143. 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 144. NCS when CKMODE = 1 with an abort (T = CLK period)



26.5 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 218. 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				

Table 218. Address alignment cases (continued)

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 = 1)	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	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 memory in DTR mode without WDM(6)	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM write	Even	Slave error	Even	Last byte is lost.
Octal Flash or RAM in DTR mode with WDM	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write				
	MM write				
HyperBus	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	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. Extra data at transfer start.
5. Extra data at transfer end.
6. WDM = write data mask.

26.6 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 219. 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

26.7 OCTOSPI registers

26.7.1 OCTOSPI control register (OCTOSPI_CR)

Address offset: 0x0000

Reset value: 0x0000 0000

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.

Note: This field can be modified only when BUSY = 0.

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.

Note: This bit can be modified only when BUSY = 0.

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.
Note: This bit can be modified only when BUSY = 0.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **TOIE**: Timeout interrupt enable
This bit enables the timeout interrupt.
0: interrupt 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**: External memory select
This bit selects the external memory to be addressed in single-, dual-, quad-SPI mode in single-memory configuration (when DMM = 0).
0: external memory 1 selected (data exchanged over IO[3:0])
1: external memory 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-memory configuration disabled

1: dual-memory configuration enabled

Note: This bit can be modified only when BUSY = 0.

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.

Note: This bit can be modified only when BUSY = 0.

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 ongoing 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.

26.7.2 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.	CSHT[5:0]						Res.	Res.	Res.	Res.	DLY BYP	Res.	FRCK	CKMODE
		rw	rw	rw	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 capacity of one of the two external devices.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **CSHT[5:0]**: 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.

0x0: NCS stays high for at least 1 cycle between external device commands.

0x1: NCS stays high for at least 2 cycles between external device commands.

...

0x3F: NCS stays high for at least 64 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).

Note: Free running clock mode is intended for delay calibration only. No memory or other device access is possible when FRCK is set.

Bit 0 **CKMODE**: Clock 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 clock mode 0.

1: CLK must stay high while NCS is high (chip-select released). This is referred to as clock mode 3.

26.7.3 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.

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.

Others: 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.

26.7.4 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.	MAXTRAN[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **MAXTRAN[7:0]**: Maximum transfer

This field enables the communication regulation feature.

The NCS is released every MAXTRAN+1 clock cycles when the other OCTOSPI request the access to the bus.

0: maximum communication disabled

Others: maximum communication is set to MAXTRAN + 1 bytes.

26.7.5 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.

Note: REFRESH count is based on the divided clock period: if OCTOSPI_DCR2 PRESCALER field is changed, the REFRESH field must be updated accordingly.

26.7.6 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.

26.7.7 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.

26.7.8 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.

26.7.9 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 0.
Writes to this field are ignored when BUSY = 1 or when FMODE = 11 (Memory-mapped mode).

26.7.10 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].

26.7.11 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.

26.7.12 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

26.7.13 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

26.7.14 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
Others: 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
Others: 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
Others: 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
Others: reserved

26.7.15 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.

26.7.16 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

26.7.17 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.

26.7.18 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.

26.7.19 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
Others: 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
Others: 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
Others: 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
Others: reserved

26.7.20 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.

26.7.21 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

26.7.22 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

26.7.23 OCTOSPI write communication configuration register (OCTOSPI_WCCR)

Address offset: 0x0180

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

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
Others: 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
Others: 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
Others: 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
Others: reserved

26.7.24 OCTOSPI write timing configuration register (OCTOSPI_WTCR)

Address offset: 0x0188

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped 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.	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.

26.7.25 OCTOSPI write instruction register (OCTOSPI_WIR)

Address offset: 0x0190

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

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

26.7.26 OCTOSPI write alternate bytes register (OCTOSPI_WABR)

Address offset: 0x01A0

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

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

26.7.27 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

26.7.28 OCTOSPI register map

Table 220. 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.	FMDOE[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.	CSHT[5:0]						Res.	Res.	Res.	Res.	DLYBYP	Res.	FRCK	CKMODE
	Reset value						0	0	0				0	0	0	0	0			0	0	0	0	0	0					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.	MAXTRAN[7:0]							
	Reset value												0	0	0	0	0									0	0	0	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 220. OCTOSPI 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
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 220. OCTOSPI 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
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.

27 OCTOSPI I/O manager (OCTOSPIM)

27.1 Introduction

The OCTOSPI I/O manager is a low-level interface that enables an efficient OCTOSPI pin assignment with a full I/O matrix (before alternate function map) and multiplex of single/dual/quad/octal SPI interfaces over the same bus.

27.2 OCTOSPIM main features

- Supports up to two single/dual/quad/octal SPI interfaces
- Supports up to two ports for pin assignment
- Fully programmable I/O matrix for pin assignment by function (data/control/clock)

27.3 OCTOSPIM implementation

The table below describes the OCTOSPIM implementation.

Table 221. OCTOSPIM implementation

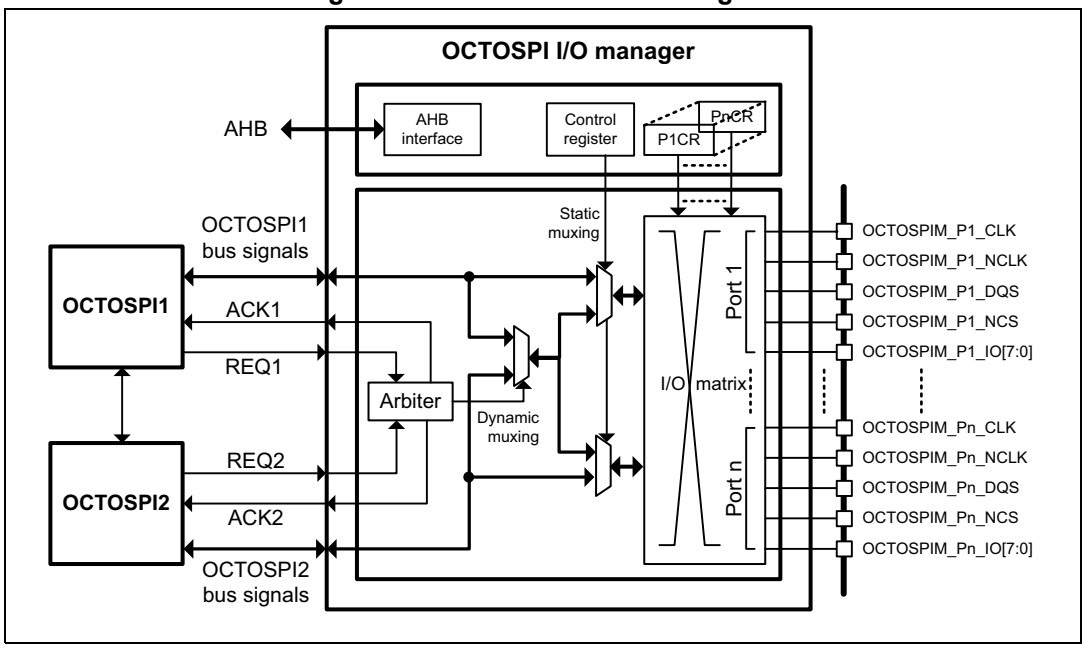
OCTOSPI feature	Available on the devices
Supports up to two single/dual/quad interfaces	X
Fully I/O multiplexing capability	X
Supports time-multiplexed mode	X
Supports high-speed interface	-
Chip select selection if OCTOSPI provides dual chip select	-
Supports 16-bit data interface and dual-octal mode	-

27.4 OCTOSPIM functional description

27.4.1 OCTOSPIM block diagram

The block diagram of the OCTOSPI I/O manager is shown in *Figure 145*.

Figure 145. OCTOSPIM block diagram



1. The number of ports (n) is 2.
2. Arbitration is possible for both I/O matrix input ports.

27.4.2 OCTOSPIM matrix

The OCTOSPI I/O manager matrix allows the user to set a fully programmable pre-mapping of functions:

- Any OCTOSPIM_Pn_CLK / OCTOSPIM_Pn_NCLK pair can be mapped independently to OCTOSPI1_CLK/OCTOSPI1_NCLK or OCTOSPI2_CLK/OCTOSPI2_NCLK
- Any OCTOSPIM_Pn_DQS can be mapped independently to OCTOSPI1_DQS or OCTOSPI2_DQS
- Any OCTOSPIM_Pn_NCS can be mapped independently to OCTOSPI1_NCS or OCTOSPI2_NCS
- Any OCTOSPIM_Pn_IO[3:0] and OCTOSPIM_Pn_IO[7:4] can be mapped independently to OCTOSPI1_IO[3:0], OCTOSPI1_IO[7:4], OCTOSPI2_IO1[3:0] or OCTOSPI2_IO[7:4]

For each OCTOSPI I/O manager port, individual signal enables and mapping are configured through the corresponding OCTOSPI I/O manager Port n configuration register (OCTOSPIM_PnCR).

When several I/O pins have the same configuration and are enabled at the same time, the result can be unpredictable.

In the default out-of-reset configuration, all the OCTOSPI1 and OCTOSPI2 signals are mapped, respectively, on Port 1 and on Port 2.

The OCTOSPIM configuration can be changed only when all OCTOSPIs are disabled.

27.4.3 OCTOSPIM multiplexed mode

When this mode is set the OCTOSPIs are time-multiplexed over the same bus. They get the ownership of the bus (in turn) through a request/acknowledge protocol with REQ/ACK signals.

The time-multiplexing is enabled by setting the MUXEN bit of the configuration register OCTOSPIM_CR.

The fairness counter (MAXTRAN) of each OCTOSPI can be used to accurately manage the maximum duration for which a given OCTOSPI takes the bus: this feature ensures a maximum bus access latency for the other OCTOSPI(s). When the bus is released by one OCTOSPI, an arbitration phase occurs, which is round-robin: when another OCTOSPI requests the bus, it gets it.

When the multiplexed mode is enabled, either the fairness counter or the refresh timeout counter of both OCTOSPI interfaces must be activated.

OCTOSPIn_NCS are not part of the multiplexing.Only OCTOSPIn_IOs, OCTOSPIn_DQS and OCTOSPIn_CLK / OCTOSPIn_NCLK are multiplexed.

When the multiplexed mode is used, only clock mode 0 is supported on the OCTOSPIs.

Due to arbitration and bus sharing, the auto polling interval time of the OCTOSPI, when used, may be increased.

Minimum switching duration

The minimum number of cycles needed to switch from an OCTOSPI to another can be configured.

This internal timer guarantees a latency between the falling edge of the REQ signal of the active OCTOSPI (the active one releases the bus), and the rising edge of the ACK signal to the requesting OCTOSPI (the bus is granted to the requesting one).

The duration is defined by the REQ2ACK_TIME field of the configuration register OCTOSPIM_CR.

Pin mapping in Multiplexed mode

In Multiplexed mode, the mapping of the bus is done as described below:

- OCTOSPI1_NCS and OCTOSPI2_NCS work in the same way, then in Non-multiplexed mode they have to be assigned to their respective OCTOSPIM_Pn_NCS.
- All the other signals are seen by the I/O matrix as if they were seen from OCTOSPI1.

27.5 OCTOSPIM registers

27.5.1 OCTOSPIM control register (OCTOSPIM_CR)

Address offset: 0x0000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REQ2ACK_TIME[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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUXEN
															rw

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

Bits 23:16 **REQ2ACK_TIME[7:0]**: REQ to ACK time

In Multiplexed mode (MUXEN = 1), this field defines the time between two transactions. The value is the number of OCTOSPI clock cycles - 1

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **MUXEN**: Multiplexed mode enable

This bit enables the multiplexing of the two OCTOSPIs.

0: No multiplexing, hence no arbitration

1: OCTOSPI1 and OCTOSPI2 are multiplexed over the same bus.

27.5.2 OCTOSPIM Port n configuration register (OCTOSPIM_PnCR)

Address offset: 0x0000 + 0x04*n (n=1 to 2)

Reset value: 0x0301 0111, 0x0705 0333

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	IOHSRC[1:0]		IOHEN	Res.	Res.	Res.	Res.	Res.	IOLSRC[1:0]		IOLEN
					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.	NCSSRC	NCSEN	Res.	Res.	DQSSRC	DQSEN	Res.	Res.	CLKSRC	CLKEN
						rw	rw			rw	rw			rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:25 **IOHSRC[1:0]**: IO[7:4] source for Port n

This bits select the source of Port n IO[7:4].

00: OCTOSPI1_IO[3:0] in non multiplexed mode / multiplexed_IO[3:0] in multiplexed mode

01: OCTOSPI1_IO[7:4] in non multiplexed mode / multiplexed_IO[7:4] in multiplexed mode

10: OCTOSPI2_IO[3:0] in non multiplexed mode / unused in multiplexed mode

11: OCTOSPI2_IO[7:4] in non multiplexed mode / unused in multiplexed mode

Bit 24 **IOHEN**: IO[7:4] enable for Port n
This bit enables the Port n IO[7:4].
0: IO[7:4] for Port n disabled
1: IO[7:4] for Port n enabled

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

Bits 18:17 **IOLSRC[1:0]**: IO[3:0] source for Port n
This bits select the source of Port n IO[3:0].
00: OCTOSPI1_IO[3:0] in non multiplexed mode / multiplexed_IO[3:0] in multiplexed mode
01: OCTOSPI1_IO[7:4] in non multiplexed mode / multiplexed_IO[7:4] in multiplexed mode
10: OCTOSPI2_IO[3:0] in non multiplexed mode / unused in multiplexed mode
11: OCTOSPI2_IO[7:4] in non multiplexed mode / unused in multiplexed mode

Bit 16 **IOLEN**: IO[3:0] enable for Port n
This bit enables the Port n IO[3:0].
0: IO[3:0] for Port n disabled
1: IO[3:0] for Port n enabled

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

Bit 9 **NCSSRC**: NCS source for Port n
This bit selects the source of Port n NCS.
0: OCTOSPI1_NCS
1: OCTOSPI2_NCS

Bit 8 **NCSEN**: NCS enable for Port n
This bit enables the Port n NCS.
0: NCS for Port n is disabled
1: NCS for Port n is enabled

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **DQSSRC**: DQS source for Port n
This bit selects the source of Port n DQS.
0: OCTOSPI1_DQS in non multiplexed mode / multiplexed_DQS in multiplexed mode
1: OCTOSPI2_DQS in non multiplexed mode / unused port in multiplexed mode

Bit 4 **DQSEN**: DQS enable for Port n
This bit enables the Port n DQS.
0: DQS for Port n is disabled
1: DQS for Port n is enabled

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **CLKSRC**: CLK/NCLK source for Port n
This bit selects the source of Port n CLK/NCLK.
0: OCTOSPI1_CLK/NCLK in non multiplexed mode / multiplexed_CLK/CLKn in multiplexed mode
1: OCTOSPI2_CLK/NCLK in non multiplexed mode / unused port in multiplexed mode

Bit 0 **CLKEN**: CLK/NCLK enable for Port n
This bit enables the Port n CLK/NCLK.
0: CLK/NCLK for Port n is disabled
1: CLK/NCLK for Port n is enabled

27.5.3 OCTOSPIM register map

Table 222. OCTOSPIM 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	**OCTOSPIM _CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REQ2ACK_TIME[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MUXEN
	Reset value									0	0	0	0	0	0	0	0																0
0x0004	**OCTOSPIM _P1CR**	Res.	Res.	Res.	Res.	Res.	IOHSRC [1:0]		IOHEN	Res.	Res.	Res.	Res.	Res.	IOLSRC [1:0]		IOLEN	Res.	Res.	Res.	Res.	Res.	Res.	NCSSRC	NCSEN	Res.	Res.	DQSSRC	DQSEN	Res.	Res.	CLKSRC	CLKEN
	Reset value						0	1	1						0	0	1							0	1			0	1			0	1
0x0008	**OCTOSPIM _P2CR**	Res.	Res.	Res.	Res.	Res.	IOHSRC [1:0]		IOHEN	Res.	Res.	Res.	Res.	Res.	IOLSRC [1:0]		IOLEN	Res.	Res.	Res.	Res.	Res.	Res.	NCSSRC	NCSEN	Res.	Res.	DQSSRC	DQSEN	Res.	Res.	CLKSRC	CLKEN
	Reset value						1	1	1						1	0	1							1	1			1	1			1	1

Refer to *Section 2.3 on page 115* for the register boundary addresses.

28 Delay block (DLYB)

28.1 Introduction

The delay block (DLYB) is used to generate an output clock that is dephased from the input clock. The phase of the output clock must be programmed by the user application. The output clock is then used to clock the data received by another peripheral such as an SDMMC or Octo-SPI interface.

The delay is voltage- and temperature-dependent, that may require the application to re-configure and recenter the output clock phase with the receive data.

28.2 DLYB main features

The delay block has the following features:

- Input clock frequency ranging from 25 MHz to the maximum frequency supported by the communication interface (see datasheet)
- Up to 12 oversampling phases.

28.3 DLYB implementation

Table 223. STM32U575/585 features

DLYB associated peripheral	DLYBOS1	DLYBOS2	DLYBSD1	DLYBSD2
OCTOSPI1	X	-	-	-
OCTOSPI2	-	X	-	-
SDMMC1	-	-	X	
SDMMC2	-	-	-	X

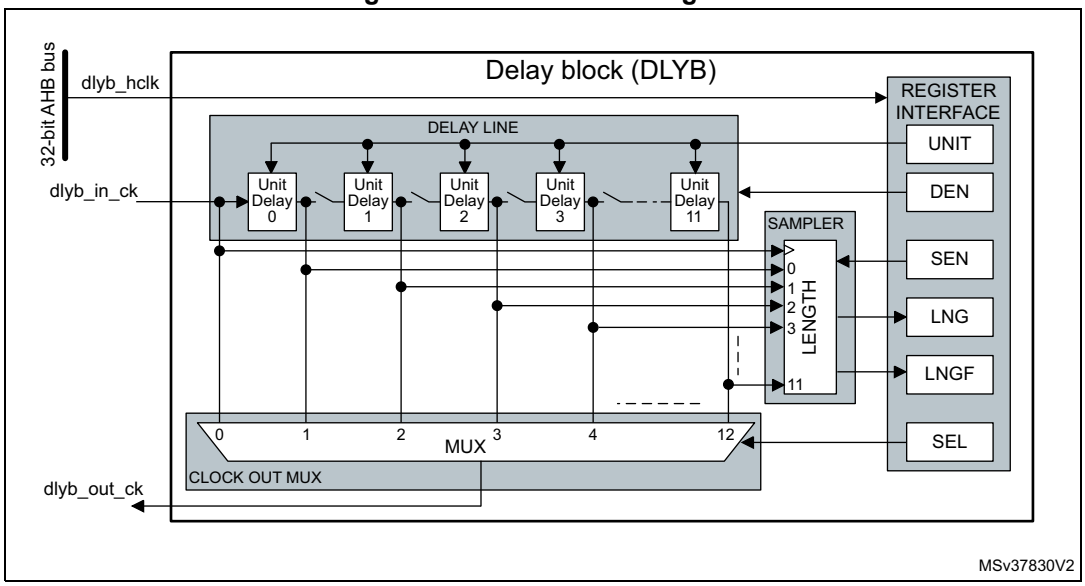
28.4 DLYB functional description

28.4.1 DLYB diagram

The delay block includes the following sub-blocks (shown in the figure below):

- register interface block providing AHB access to the DLYB registers
- delay line supporting the unit delays
- delay line length sampling
- output clock selection multiplexer

Figure 146. DLYB block diagram



28.4.2 DLYB pins and internal signals

Table 224 lists the DLYB internal signals.

Table 224. DLYB internal input/output signals

Signal name	Signal type	Description
dlyb_hclk	Digital input	Delay block register interface clock
dlyb_in_ck	Digital input	Delay block input clock
dlyb_out_ck	Digital output	Delay block output clock

28.4.3 General description

The delay block is enabled by setting the DEN bit in the DLYB control register (DLYB_CR). The length sampler is enabled through the SEN bit in DLYB_CR register.

When the delay block is enabled, the delay added by a unit delay is defined by the UNIT[6:0] field in the DLYB configuration register (DLYB_CFGR).

Note: UNIT[6:0] can be programmed only when the output clock is disabled (SEN = 1).

When the delay block is enabled, the output clock phase is selected through the SEL[3:0] field in DLYB_CFGR register.

Note: SEL can be programmed only when the output clock is disabled (SEN = 1).

The output clock can be de-phased over one input clock period by configuring the delay line length to span one period. The delay line length can be configured by enabling the length sampler through the SEN bit, that gives access to the delay line length (LNG[11:0]) and length valid flag (LNGF) in DLYB_CFGR.

If an output clock delay smaller than one input clock period is needed the delay line length can be reduced. This allows a smaller unit delay providing higher resolution.

Once the delay line length is configured, a dephased output clock can be selected by the output clock multiplexer. This is done through SEL[3:0]. The output clock is only available on the selected phase when SEN is set to 0.

The table below gives a summary of the delay block control.

Table 225. Delay block control

DEN	SEN	UNIT	SEL	LNG	LNGF	Output clock
0	0	Don't care	Don't care	Don't care	Don't care	Enabled (= Input clock)
x	1	Unit delay	Output clock phase	Length	Length flag	Disabled
1	0	Unit delay(1)	Output clock phase(2)	Don't care	Don't care	Enabled (= selected phase)

1. The unit delay can only be changed when SEN = 1.
2. The output clock phase can only be changed when SEN = 1.

28.4.4 Delay line length configuration procedure

LNG[11:0] is used to determine the delay line length with respect to the input clock period. The length must be configured so that one full input clock period is covered by the delay line length.

Note that despite the delay line has 12 unit delay elements, the following procedure description returns a length between 0 and 10, as the upper delay output value is used to ensure that the delay is calibrated over one full input clock cycle. Depending on the clock frequency and UNIT value, unit delay element 10 may also be truncated from the clock cycle length.

A clock input (free running clock) must be present during the whole tuning procedure.

To configure the delay line length to one period of the Input clock, follow the sequence below:

1. Enable the delay block by setting DEN bit to 1.
2. Enable the length sampling by setting SEN bit to 1.
3. Enable all delay cells by setting SEL[3:0] to 12.
4. For UNIT[6:0] = 0 to 127 (this step must be repeated until the delay line length is configured):
   a) Update the UNIT[6:0] value and wait till the length flag LNGF is set to 1.
   b) Read LNG[11:0].

   If (LNG[10:0] > 0) and (LNG[11] or LNG[10] = 0), the delay line length is configured to one input clock period.
5. Determine how many unit delays (N) span one input clock period: for N = 0 to 10, if LNG[N] = 1, the number of unit delays spanning the input clock period = N.
6. Disable the length sampling by clearing SEN to 0.

If an output clock delay smaller than one input clock period is needed the delay line length can be reduced smaller than one input clock period. This allows a smaller unit delay, providing a higher resolution spanning a shorter time interval.

28.4.5 Output clock phase configuration procedure

When the delay line length is configured to one input clock period, the output clock phase can be selected between the unit delays spanning one Input clock period.

Follow the steps below to select the output clock phase:

1. Disable the output clock and enable the access to the phase selection SEL[3:0] bits by setting SEN bit to 1.
2. Program SEL[3:0] with the desired output clock phase value.
3. Enable the output clock on the selected phase by clearing SEN to 0.

28.5 DLYB registers

All registers can be accessed in word, half-word and byte access.

28.5.1 DLYB control register (DLYB_CR)

Address offset: 0x000

Reset value: 0x0000 0000

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SEN:** Sampler length enable bit

0: Sampler length and register access to UNIT[6:0] and SEL[3:0] disabled, output clock enabled.

1: Sampler length and register access to UNIT[6:0] and SEL[3:0] enabled, output clock disabled.

Bit 0 **DEN:** Delay block enable bit

0: DLYB disabled.

1: DLYB enabled.

28.5.2 DLYB configuration register (DLYB_CFGR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LNGF	Res.	Res.	Res.	LNG[11:0]											
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.	UNIT[6:0]							Res.	Res.	Res.	Res.	SEL[3:0]			
	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

Bit 31 **LNGF:** Length valid flag

This flag indicates when the delay line length value contained in LNG[11:0] is valid after UNIT[6:0] bits changed.

0: Length value in LNG is not valid.

1: Length value in LNG is valid.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **LNG[11:0]:** Delay line length value

These bits reflect the 12 unit delay values sampled at the rising edge of the input clock. The value is only valid when LNGF = 1.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **UNIT[6:0]:** Delay of a unit delay cell.

These bits can only be written when SEN = 1.

Unit delay = initial delay + UNIT[6:0] x delay step

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

Bits 3:0 **SEL[3:0]:** Phase for the output clock.

These bits can only be written when SEN = 1.

Output clock phase = input clock + SEL[3:0] x unit delay

28.5.3 DLYB register map

Table 226. DLYB 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	**DLYB_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.	SEN	DEN
	Reset value																															0	0
0x004	**DLYB_CFGR**	LNGF	Res.	Res.	Res.	LNG												Res.	UNIT							Res.	Res.	Res.	Res.	SEL			
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0					0	0	0	0

Refer to *Section 2.3 on page 115* for the register boundary addresses.

29 Analog-to-digital converter (ADC1)

29.1 Introduction

This section describes the implementation of the 14-bit ADC successive approximation analog-to-digital converter.

The ADC features up to 20 multiplexed channels. Channel A/D conversion can be performed in Single, Continuous, Scan or Discontinuous mode. The result of the ADC can be stored in a left-aligned or right-aligned 32-bit data register.

The ADC is mapped on the AHB bus to enable fast data handling.

In addition, the analog watchdog features enable the application to detect if the input voltage goes outside user-defined high or low thresholds.

The ADC features a built-in hardware oversampler that improves analog performances while off-loading the related computational burden from the CPU.

An efficient low-power mode is also implemented to achieve very low consumption at low frequency.

29.2 ADC main features

- High-performance features
  - 14-, 12-, 10- or 8-bit configurable resolution
  - ADC conversion time independent from the AHB bus clock frequency
  - Faster conversion time by lowering resolution
  - Management of single-ended or differential inputs (programmable per channels)
  - AHB slave bus interface for fast data handling
  - Self-calibration (both offset and linearity)
  - Channel-wise programmable sampling time
  - Flexible sampling time control
  - Up to four injected channels (fully configurable analog input assignment to regular or injected channels)
  - Hardware assistant to prepare the injected channel context and enable fast context switching
  - Data alignment with in-built data coherency
  - Data management by general-purpose DMA for regular channel conversions with FIFO
  - Data routing to MDF for post processing
  - Four dedicated data registers for injected channels
- Oversampler
  - 32-bit data register
  - Oversampling ratio adjustable from 2 to 1024
  - Programmable data right and left shift

- Data preconditioning
  - Gain compensation
  - Offset compensation
- Low-power features
  - Speed adaptive low-power mode to reduce ADC consumption when operating at low frequency
  - Support of slow bus frequency applications while keeping optimum ADC performance
  - Automatic control to avoid ADC overrun in AHB bus clock low-frequency application (auto-delayed mode)
- Up to 17 external analog input channels connected to dedicated GPIO pads
- 3 internal dedicated channels
  - One channel for internal reference voltage (V_{REFINT})
  - One channel for internal temperature sensor (V_{SENSE})
  - One channel for V_{BAT} monitoring channel ($V_{BAT}/4$)
- Start-of-conversion can be initiated:
  - by software for both regular and injected conversions or
  - by hardware triggers with configurable polarity (internal timers events or GPIO input events) for both regular and injected conversions
- Conversion modes
  - Single mode: the ADC converts a single channel. The conversion is triggered by a special event.
  - Scan mode: the ADC scans and converts a sequence of channels.
  - Continuous mode: the ADC converts continuously selected inputs.
  - Discontinuous mode: the ADC converts a subset of the conversion sequence.
- Interrupt generation when the ADC is ready, at end of sampling, end of conversion (regular or injected), end of sequence conversion (regular or injected), analog watchdog 1, 2 or 3 or when an overrun event occurs
- Three analog watchdogs

  The watchdogs can perform filtering to ignore out-of-range data.
- ADC input range: $V_{REF-} \leq V_{IN} \leq V_{REF+}$

Figure 147 shows the block diagram of one ADC.

29.3 ADC implementation

The tables below describe ADC implementation on STM32U575/585 devices. It also includes ADC4 for comparison.

Table 227. ADC features(1)

ADC modes/features	ADC1	ADC4
Resolution	14 bits	12 bits
Maximum sampling speed for 14-bit resolution	2.5 Msps	2.5 Msps
Hardware offset calibration	X	X

Table 227. ADC features(1) (continued)

ADC modes/features	ADC1	ADC4
Hardware linearity calibration	X	-
Extended calibration mode	X(2)	-
Single-ended inputs	X	X
Differential inputs	X	-
Injected channel conversion	X	-
Oversampling	up to x1024	up to x256
Data register	32 bits	16 bits
DMA support	X	X
Parallel data output to MDF	X	-
Dual mode	-	-
Autonomous mode	-	X
Offset compensation	X	-
Gain compensation	X	-
Number of analog watchdogs	3	3
Wakeup from Stop mode	-	X(3)

1. Note: 'X' = supported, '-'= not supported.
2. The extended calibration mode is not supported on devices revision X.
3. Wake up supported from Stop 0, Stop 1 and Stop 2 modes.

Table 228. Memory location of the temperature sensor calibration values

Name	Description	Memory address
TS_CAL1	Temperature sensor 14-bit raw data acquired by ADC1 at 30 °C (± 5 °C), $V_{DDA} = V_{REF+}$ = 3.0 V (±10 mV)	0x0BFA 0710 - 0x0BFA 0711
TS_CAL2	Temperature sensor 14-bit raw data acquired by ADC1 at 130 °C (± 5 °C), $V_{DDA} = V_{REF+}$ = 3.0 V (±10 mV)	0x0BFA 0742 - 0x0BFA 0743

Table 229. Memory location of the internal reference voltage sensor calibration value

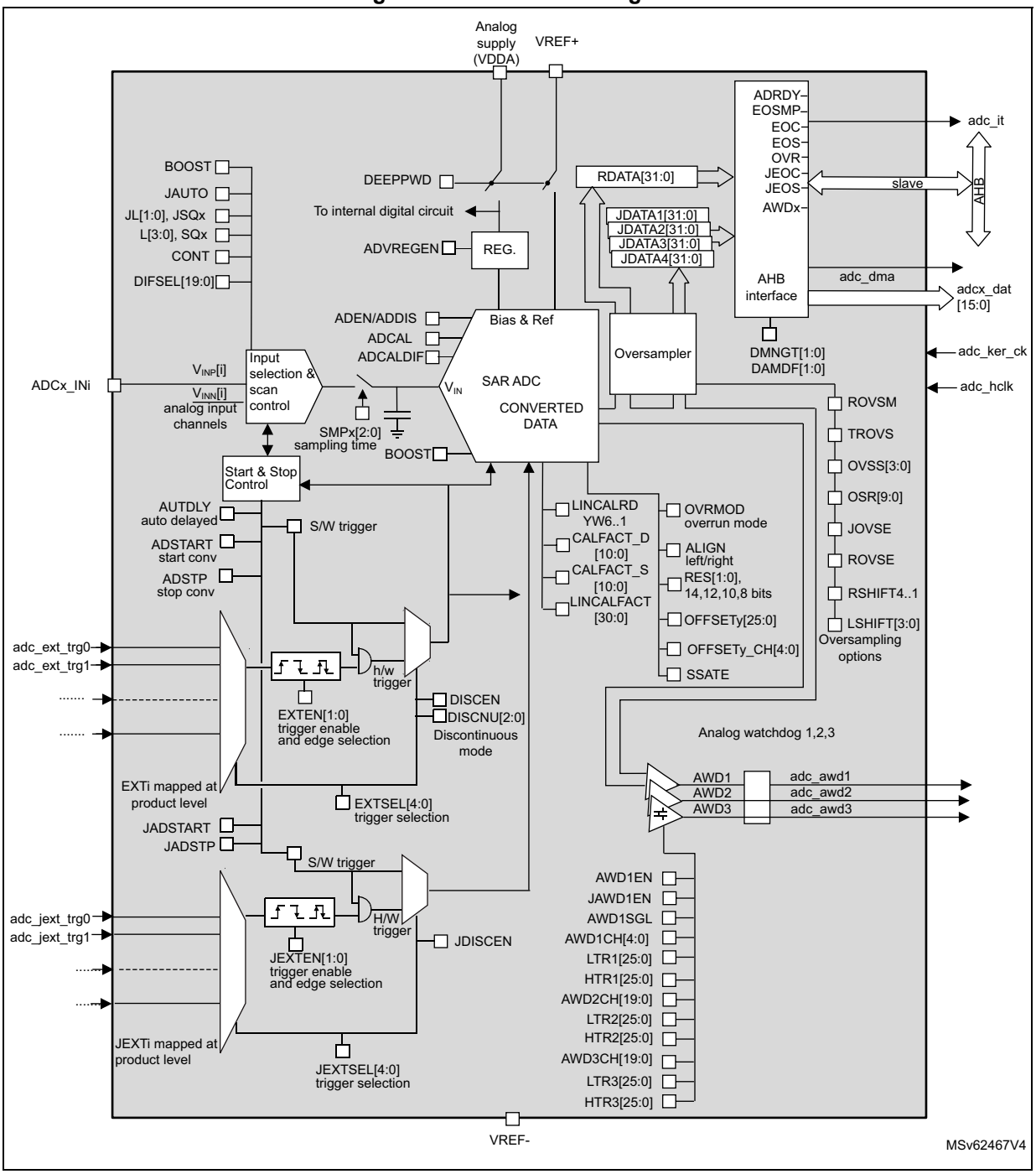
Name	Description	Memory address
VREFINT_CAL	14-bit raw data acquired by ADC1 at 30 °C (± 5 °C), $V_{DDA} = V_{REF+}$ = 3.0 V (±10 mV)	0x0BFA 07A5 - 0x0BFA 07A6

29.4 ADC functional description

29.4.1 ADC block diagram

Figure 147 shows the ADC block diagram and *Table 230* gives the ADC pin description.

Figure 147. ADC block diagram



29.4.2 ADC pins and internal signals

Table 230. ADC input/output pins

Name	Signal type	Description
VREF+	Input, analog reference positive	Higher/positive reference voltage for the ADC
VDDA	Input, analog supply	Analog power supply equal V_{DDA}
VREF–_	Input, analog reference negative	Lower/negative reference voltage for the ADC, $V_{REF-} = V_{SSA}$
VSSA	Input, analog supply ground	Ground for analog power supply equal to V_{SS}
ADCx_INy	External analog input signals	Up to 17 external analog input channels

Table 231. ADC internal input/output signals

Internal signal name	Signal type	Description
V_{INP}[i]	Analog inputs	Positive input analog channels for each ADC
V_{INN}[i]	Analog inputs	Negative input analog channels for each ADC
adc_ext_trgy	Inputs	External trigger inputs for the regular conversions (can be connected to on-chip timers).
adc_jext_trgy	Inputs	External trigger inputs for the injected conversions (can be connected to on-chip timers).
adc_awd1 adc_awd2 adc_awd3	Outputs	Internal analog watchdog output signal connected to on-chip timers.
adc_it	Output	ADC interrupt
adc_hclk	Input	AHB clock
adc_ker_ck	Input	ADC kernel clock
adc_dma	Output	ADC DMA requests
adcx_dat[15:0]	Output	ADC data outputs (regular data register)

Table 232. Interconnection

Signal name	Source/destination
ADCx V_{INP}[19] (x = 1)	V_{SENSE}
ADCx V_{INP}[0] (x = 1)	V_{REFINT} buffered voltage
ADCx V_{INP}[18] (x = 1)	V_{BAT}/4
adcx_dat[15:0] (x = 1)	mdf1_adcx_dat[15:0]

Table 233. External triggers for regular channels

Name	Source
adc_ext_trg0	tim1_oc1
adc_ext_trg1	tim1_oc2
adc_ext_trg2	tim1_oc3
adc_ext_trg3	tim2_oc2
adc_ext_trg4	tim3_trgo
adc_ext_trg5	tim4_oc4
adc_ext_trg6	exti11
adc_ext_trg7	tim8_trgo
adc_ext_trg8	tim8_trgo2
adc_ext_trg9	tim1_trgo
adc_ext_trg10	tim1_trgo2
adc_ext_trg11	tim2_trgo
adc_ext_trg12	tim4_trgo
adc_ext_trg13	tim6_trgo
adc_ext_trg14	tim15_trgo
adc_ext_trg15	tim3_oc4
adc_ext_trg16	exti15
adc_ext_trg18	lptim1_ch1
adc_ext_trg19	lptim2_ch1
adc_ext_trg20	lptim3_ch1
adc_ext_trg21	lptim4_out

Table 234. External triggers for injected channels

Name	Source
adc_jext_trg0	tim1_trgo
adc_jext_trg1	tim1_oc4
adc_jext_trg2	tim2_trgo
adc_jext_trg3	tim2_oc1
adc_jext_trg4	tim3_oc4
adc_jext_trg5	tim4_trgo
adc_jext_trg6	exti15
adc_jext_trg7	tim8_oc4
adc_jext_trg8	tim1_trgo2
adc_jext_trg9	tim8_trgo
adc_jext_trg10	tim8_trgo2

Table 234. External triggers for injected channels (continued)

Name	Source
adc_jext_trg11	tim3_oc3
adc_jext_trg12	tim3_trgo
adc_jext_trg13	tim3_oc1
adc_jext_trg14	tim6_trgo
adc_jext_trg15	tim15_trgo
adc_jext_trg18	lptim1_ch2
adc_jext_trg19	lptim2_ch2
adc_jext_trg20	lptim3_ch1
adc_jext_trg21	lptim4_out1

29.4.3 ADC clocks

Dual clock domain architecture

Dual clock-domain architecture means that the ADC kernel clock is independent from the AHB bus clock that is used to access ADC registers.

The adc_ker_ck input clock can be selected between different clock sources (see *Figure 148: ADC clock scheme*). This selection is done in the RCC (refer to th RCC section for more information):

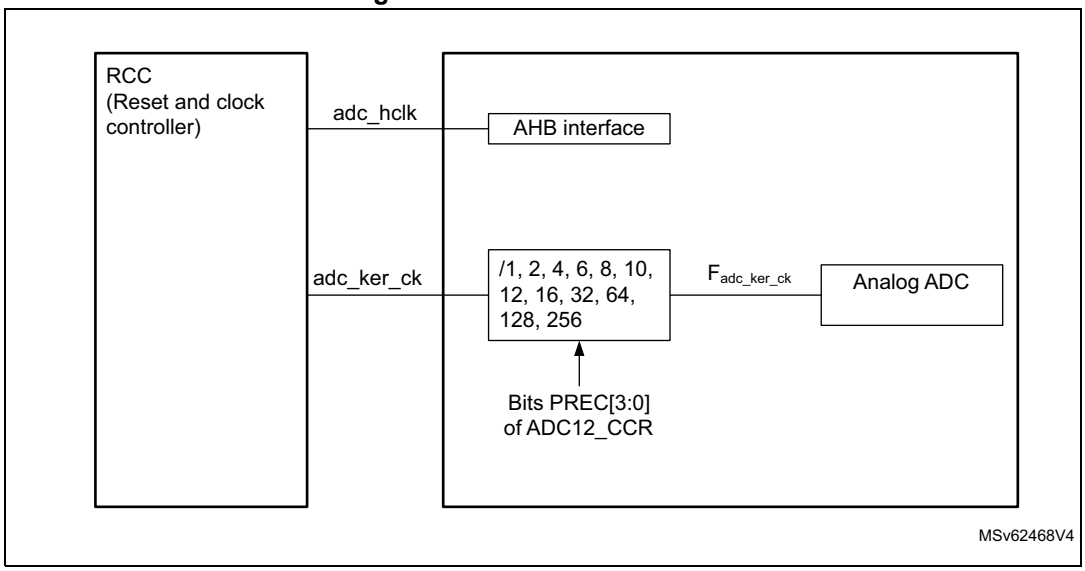
1. The ADC clock can be provided by an internal or external clock source, which is independent and asynchronous with the AHB clock.
2. The ADC clock can be derived from the AHB clock.

Option **1** has the advantage of achieving the maximum ADC clock frequency whatever the AHB clock scheme selected. The ADC clock can eventually be divided by a ratio of 1, 2, 4, 6, 8, 10, 12, 16, 32, 64, 128 or 256, using the prescaler configured through PRESC[3:0] bits in the ADC12_CCR register.

Option **2** enables to bypass the clock domain resynchronizations. This can be useful when the ADC is triggered by a timer and the application requires that the ADC is accurately triggered without any uncertainty (otherwise, an uncertainty of the trigger instant is added by the resynchronizations between the two clock domains).

The clock is configured through the RCC. It must be compliant with the operating frequency specified in the device datasheet.

Figure 148. ADC clock scheme



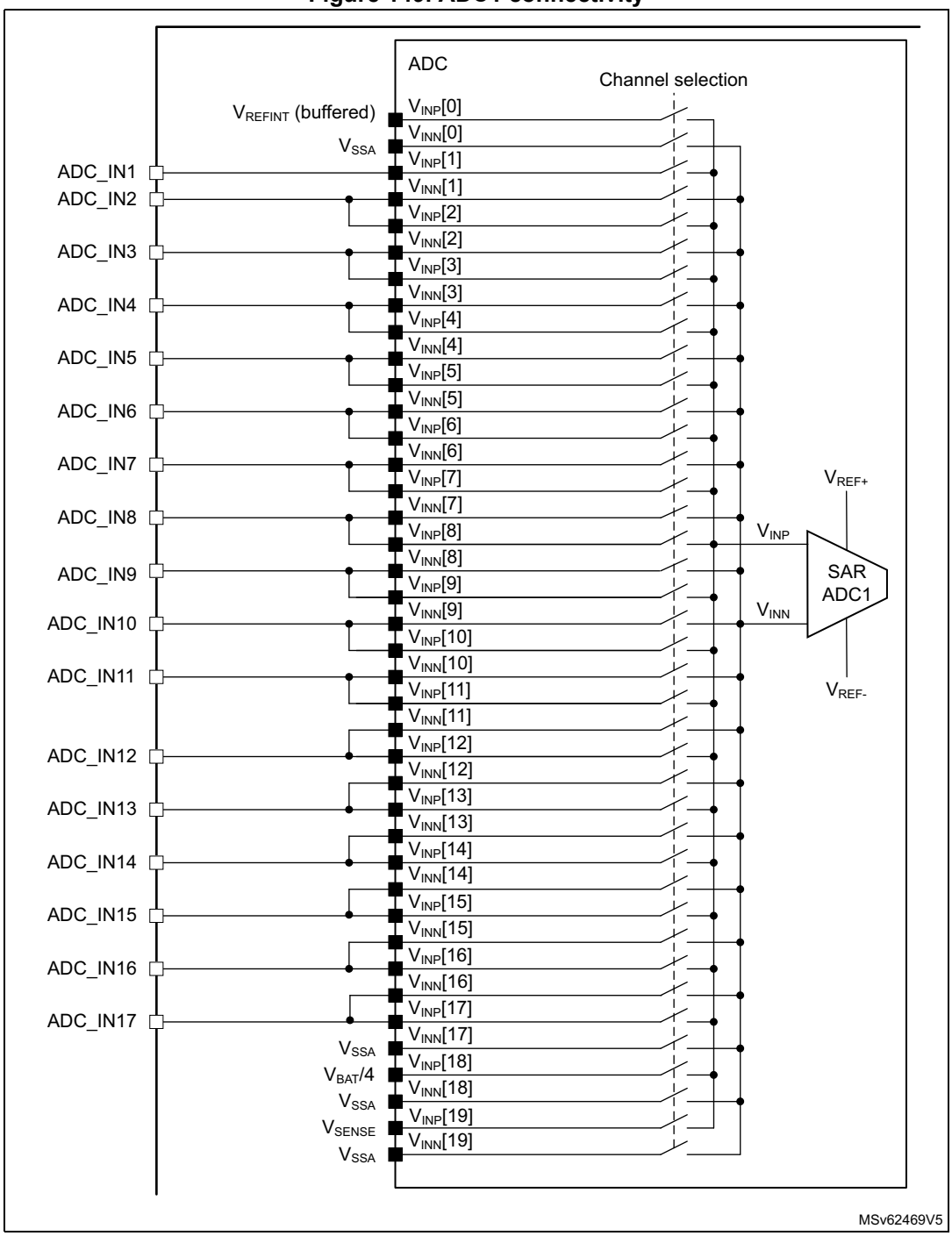
1. Refer to the RCC section for information on adc_hclk and adc_ker_ck generation.

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. However the ratio must be carefully chosen to avoid any overrun especially if the clock AHB is much slower than the ADC clock.

29.4.4 ADC connectivity

Figure 149. ADC1 connectivity



1. $V_{INN}[i]$ signal can only be used when the corresponding ADC input channel is configured as differential mode.

29.4.5 Slave AHB interface

The ADC implements 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.

29.4.6 ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)

By default, the ADC is in Deep-power-down mode where its supply voltage 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, follow the sequence below:

1. First exit Deep-power-down mode by clearing DEEPPWD bit.
2. Then, enable the ADC internal voltage regulator by setting ADVREGEN bit in the ADC_CR register. The software must wait for the startup time of the ADC voltage regulator ($T_{ADCVREG\_STUP}$) before launching a calibration or enabling the ADC. This can be done by software by polling the LDORDY bit of ADC_ISR register.

After 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 clearing ADVREGEN bit. Power consumption can be further reduced by reducing the leakage currents. In addition, it is possible to enter again in ADC Deep-power-down mode by setting DEEPPWD bit in ADC_CR register. This is particularly interesting before entering Stop mode.

Note: Setting DEEPPWD automatically disables the ADC voltage regulator and ADVREGEN bit is automatically cleared.

When the internal voltage regulator is disabled (ADVREGEN = 0), the internal analog calibration factor *is kept.*

In ADC Deep-power-down mode (DEEPPWD = 1), the internal analog calibration is lost and it is necessary either to relaunch a calibration or apply again the calibration factor which was previously saved (refer to *Section 29.4.8: Calibration (ADCAL, ADCALLIN, ADC_CALFACT)*).

29.4.7 Single-ended and differential input channels

ADC channels can be configured either as single-ended input or as differential input. This is done by writing DIFSEL[19:0] bits in the ADC_DIFSEL register. This configuration must be performed while the ADC is disabled (ADEN = 0).

In Single-ended input mode, the analog voltage to be converted for channel “i” is the difference between the external voltage $V_{INP}[i]$ (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 external voltage $V_{INP}[i]$ (positive input) and $V_{INN}[i]$ (negative input).

The output data in Differential mode is an unsigned data:

- When $V_{INP}[i] = V_{REF-}$: $V_{INN}[i] = V_{REF+}$and output data = 0x0000 (14-bit resolution mode),
- When $V_{INP}[i] = V_{REF+}$: $V_{INN}[i] = V_{REF-}$ and output data = 0x3FFF.

$$\text{Converted value} = \frac{\text{ADC\_Full\_Scale}}{2} \times \left[1 + \frac{V_{INP} - V_{INN}}{V_{REF+}}\right]$$

When ADC is configured as Differential mode, both inputs must be biased at V_{REF+} / 2 voltage.

For a complete description of how the input channels are connected, refer to *Section 29.4.4: ADC connectivity*

Caution: When channel “i” is configured in Differential input mode, its negative input voltage is connected to $V_{INN}[i-1]$.

29.4.8 Calibration (ADCAL, ADCALLIN, ADC_CALFACT)

The ADC provides an automatic calibration procedure that controls the whole calibration sequence including the ADC power-on/off. During the procedure, the ADC calculates an offset calibration factor for Single-ended and Differential mode. This factor includes the internal offset and the linearity that are 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 the calibration is complete.

The calibration is a prerequisite to any ADC operation. It removes the systematic errors that may vary from chip to chip and enables to compensate offset and linearity deviation.

The offset calibration is the same for single-ended or differential channels.

The linearity correction must be done only once, regardless of single / differential configuration:

- Set ADCALLIN in ADC_CR before launching a calibration that runs the linearity calibration simultaneously with the offset calibration or
- Clear ADCALLIN in ADC_CR before launching a calibration that does not run the linearity calibration but only the offset calibration.

The calibration is then initiated by software by setting ADCAL bit. The calibration can only be initiated when the ADC is disabled (ADEN = 0). ADCAL bit remains at 1 during all the calibration sequence. It is cleared by hardware as soon the calibration completes. At this time, the associated calibration factor is stored internally in the analog ADC.

The internal analog calibration is kept if the ADC is disabled (ADEN = 0). However, if the ADC is disabled for extended periods of time, the temperature changes, or the supply voltage is modified of more than 10%, it is recommended that a new offset calibration cycle is run before enabling the ADC again.

The internal analog calibration is lost each time the ADC power is switched off (for example, when the device enters Standby or VBAT mode). In this case, to avoid spending time recalibrating the ADC, the calibration factor can be written again to the ADC analog block without recalibrating, assuming that the software has previously saved the calibration factor generated during the previous calibration.

The calibration factor obtained during factory tests is programmed in a specific device option byte. To reduce the calibration time, this value can be copied to the analog ADC.

The calibration factor can be written if the ADC is enabled and no calibration is ongoing (ADEN = 1 and ADSTART = 0 and JADSTART = 0). Then, at the next start of conversion, the calibration factor is automatically injected into the analog ADC. This operation is transparent and does not add any cycle latency to the start of the conversion. It is recommended to recalibrate the ADC offset when V_{REF+} changes of more than 10%.

Refer to the device datasheet for the offset and linearity calibration time requirements.

Software procedure to calibrate the ADC

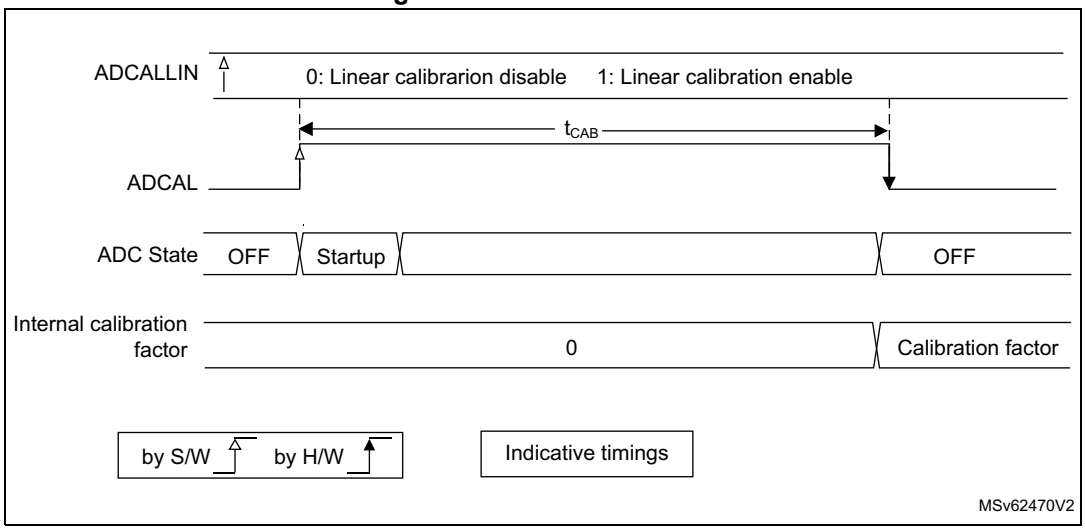
1. Make sure DEEPPWD = 0, ADVREGEN = 1 and check that the ADC voltage regulator startup time has elapsed (LDORDY = 1).
2. Make sure ADEN = 0.
3. Either enable the linearity calibration (ADCALLIN = 1) or disable it (ADCALLIN = 0).
4. Make sure CAPTURE_COEF and LATCH_COEF in ADC_CALFACT are cleared.
5. Set ADCAL in ADC_CR register and wait until ADCAL = 0.

Figure 150 shows the ADC calibration timing diagram.

Note: The software can launch the calibration only when ADEN = 0, LDORDY = 1 *(ADC disabled and LDO ready).*

Single-ended and differential channels cannot be calibrated separately. The offset calibration must be performed within the same sequence.

Figure 150. ADC calibration



Reading calibration factor procedure

Once the calibration is complete (ADCAL bit cleared by hardware), the calibration factor can be read using the ADC_CALFACT2 register. Nine read accesses are required to perform this operation:

1. Make sure DEEPPWD = 0, ADVREGEN = 1, and check that the ADC voltage regulator startup time has elapsed (LDORDY = 1).
2. Make sure that ADEN = 1.
3. Set CAPTURE_COEF and clear LATCH_COEF in ADC_CALFAC register.
4. Select the calibration factor by setting CALLINDEX[0:3] in *ADC control register (ADC_CR)*.
5. Read the calibration factor from ADC_CALFACT2 register.
6. Repeat steps **4** and **5** for each required calibration factor.
7. Clear CAPTURE_COEF bit in ADC_CALFACT register.

Note: The software can access the calibration factor only when ADEN = 1, ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing).

Software procedure to reinject the calibration factor into the ADC

1. Make sure ADEN = 1, ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing).
2. Clear CAPTURE_COEF and LATCH_COEF bits in ADC_CALFACT register
3. Set CALINDEX[3:0] to the targeted calibration index to be updated, then write the calibration factor to CALFACT[31:0] in ADC_CALFACT2 register.
4. Repeat step 3 for the necessary calibration index.
5. Set LATCH_COEF bit in ADC_CALFACT register.
6. Clear LATCH_COEF bit in ADC_CALFACT register.

Note: The software is allowed to update the calibration factor only when ADEN = 1 and ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing).

Calibration factor index

The calibration factors are stored in the analog block in an indexed way. Index 0b0000 and 0b1000 contains the offset calibration factor while index 0b0001 to 0b0110 contain the linearity factor. The lower two bytes of index 0b0111 contain the linearity calibration factor. The internal offset calibration factor must be programmed into the second byte of index 0b0111. However, it is read at byte 3 of index 0b1000. When programming or reinjecting the calibration factor, make sure to use the correct indexes for read and write operations.

Refer to *Table 235* for a summary.

Table 235. Calibration factor index

CALINDEX[3:0] values	Calibration factor			
	Byte location			
	Byte 3	Byte 2	Byte 1	Byte 0
0b0000	Differential offset		Single-end offset	
0b0001	Linearity factor 1			

Table 235. Calibration factor index (continued)

CALINDEX[3:0] values	Calibration factor			
	Byte location			
	Byte 3	Byte 2	Byte 1	Byte 0
0b0010	Linearity factor 2			
0b0011	Linearity factor 3			
0b0100	Linearity factor 4			
0b0101	Linearity factor 5			
0b0110	Linearity factor 6			
0b0111 (read)	Reserved	Reserved	Linearity factor 7	
0b0111 (write)	Reserved	Internal offset		
0b1000 (read)	Internal offset	Reserved	Reserved	Reserved
0b1001	Reserved	Calibration mode	Reserved	Reserved
0b1000 (write)	Reserved	Reserved	Reserved	Reserved
Other values	Reserved			

To run an offset calibration when the linearity calibration factor is programmed by software, first program the linearity calibration factor, then run the offset calibration. Running the offset calibration first would cause the offset calibration factor to be overwritten when the LATCH_COEF bit is programmed.

It is impossible to program the offset calibration alone since byte 1 and byte 0 (linearity factor 7) are overwritten when programming the offset calibration factor into byte 2 of index 0b0111. To avoid this, read the linearity calibration factor then program index 0b0111 with the read value.

Extended calibration mode

To enhance the ADC performance, the extended calibration mode is implemented on some product versions (see *Section 29.3: ADC implementation*).

Below the procedure to enable the extended calibration mode:

1. Make sure that DEEPPWD = 0 and ADVREGEN = 1, and check that the ADC voltage regulator startup time has elapsed (LDORDY = 1).
2. Set ADCCALLIN in ADC_CR register.
3. Make sure CAPTURE_COEF and LATCH_COEF of ADC_CALFACT are cleared.
4. Set ADEN and wait until ADRDY is set.
5. Set CALINDEX[3:0] = 0b1001, then write CALFACT[31:0] = 0x0002 0000 in ADC_CALFACT2 register.
6. Set LATCH_COEF bit in ADC_CALFACT register
7. Clear LATCH_COEF bit in ADC_CALFACT register
8. Set ADCAL in ADC_CR register and wait until both ADCAL and ADEN are cleared.

Note: Only this procedure is allowed to set ADCAL when ADEN is set (exception).

Once the calibration is complete, the value of CALFACT[31:0] at CALINDEX[3:0] = 0b1001 is reset.

29.4.9 ADC on-off control (ADEN, ADDIS, ADRDY)

First of all, follow the procedure described in *Section 29.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)*).

Once DEEPPWD bit is cleared and ADVREGEN bit is set, the ADC can be enabled. It requires a stabilization time of t_{STAB} before staring converting accurately (see *Figure 151*). Two control bits enable or disable the ADC:

- When ADEN = 1: the ADC is enabled. The ADRDY is set as soon as the ADC is ready for operation.
- When ADDIS = 1: the ADC is disabled.

ADEN and ADDIS bits are automatically cleared by hardware as soon as the analog ADC is effectively disabled.

Regular conversions can then start either by setting ADSTART (refer to *Section 29.4.19: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN[1:0], JEXTSEL, JEXTEN[1:0])*) or when an external trigger event occurs if triggers are enabled.

Injected conversions start by setting JADSTART or when an external injected trigger event occurs if injected triggers are enabled.

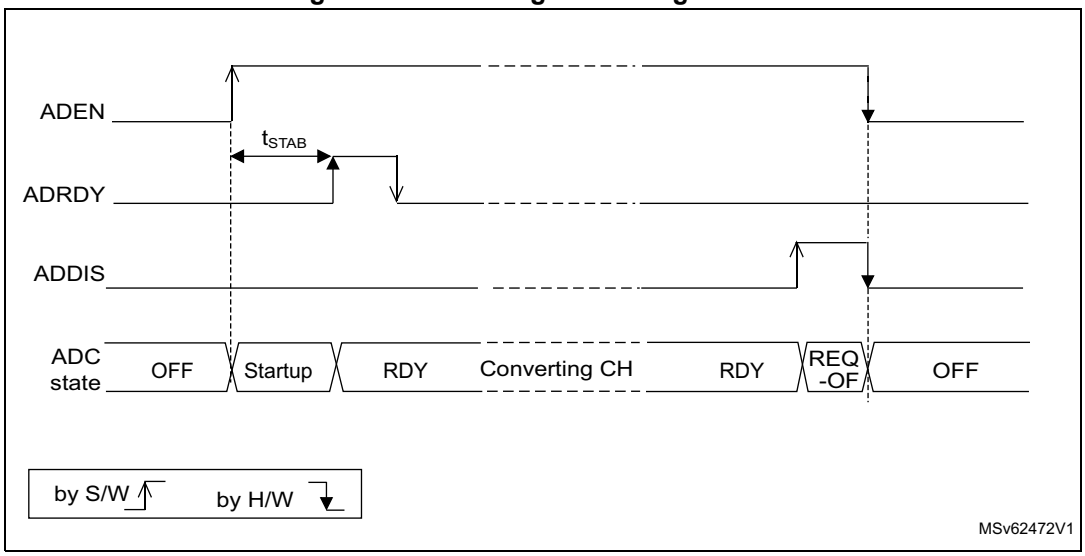
Software procedure to enable the ADC

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 by using the associated interrupt (ADRDYIE = 1).
4. Clear the ADRDY bit in the ADC_ISR register by writing 1 (optional).

Software procedure to disable the ADC

1. Check that both ADSTART = 0 and JADSTART = 0 to make sure that no conversion is ongoing. If required, stop any ongoing regular and injected conversion by setting ADSTP = 1 and JADSTP = 1 and then wait until ADSTP = 0 and JADSTP = 0.
2. Set ADDIS.
3. If required by the application, wait until ADEN = 0, until the analog ADC is effectively disabled (ADDIS is automatically reset once ADEN = 0).

Figure 151. Enabling / Disabling the ADC



29.4.10 Constraints when writing the ADC control bits

The software can program the RCC control bits to configure and enable the ADC clock (refer to the Reset and clock control section), the control DIFSEL bits in the ADC_DIFSEL register, the ADC12_CCR register and the ADCAL and ADEN control bits in the ADC_CR register, only if the ADC is disabled.

The software can program the ADCAL bit to launch the calibration when ADEN is cleared to 0. It can read or update the calibration factor if ADEN is set to 1 and no conversion is ongoing (ADSTART and JADSTART both cleared to 0).

The software is then allowed to write the ADSTART, JADSTART and ADDIS control bits in the ADC_CR register only if the ADC is enabled and there is no pending request to disable it (ADEN must be equal to 1 and ADDIS to 0).

The following constraints apply to all the other control bits of the ADC_CFGRx, ADC_SMPRy, ADC_LTRy, ADC_HTRy, ADC_SQRy, ADC_OFRy and ADC_IER registers:

- Control bits related to configuration of regular conversions: the software is allowed to write them only if the ADC is enabled (ADEN = 1) and no regular conversion is ongoing (ADSTART must be equal to 0).
- Control bits related to configuration of injected conversions: the software is allowed to write them only if the ADC is enabled (ADEN = 1) and no injected conversion is ongoing (JADSTART must be equal to 0).
- ADC_LTRy, ADC_HTRy registers can be modified when an analog-to-digital conversion is ongoing (refer to *Section 29.4.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)*for details).

The software can write ADSTP or JADSTP control bits in the ADC_CR register only if the ADC is enabled, a conversion is ongoing and there is no pending request to disable it (ADSTART or JADSTART must be equal to 1 and ADDIS to 0).

Note: There is no hardware protection to prevent these forbidden write accesses which may cause the ADC to enter an unknown state. To recover from this situation, the ADC must be disabled (clear ADEN as well as all the bits of ADC_CR register).

29.4.11 Channel selection (SQRx, JSQRx)

The ADC features up to 20 multiplexed channels:

- Up to 17 analog inputs coming from GPIO pads
- The ADC is connected to 3 internal analog inputs:
  - the internal temperature sensor (V_{SENSE})
  - the internal reference voltage (V_{REFINT})
  - the V_{BAT} monitoring channel ($V_{BAT}/4$)

Refer to *Table ADC interconnection* in *Section 29.4.2: ADC pins and internal signals* for the connection of the above internal analog inputs to external ADC pins or internal signals.

The conversions can be organized 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: ADC_IN3, ADC_IN8, ADC_IN2, ADC_IN2, ADC_IN0, ADC_IN2, ADC_IN2, ADC_IN15.

- 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 four 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 are ongoing. To modify ADC_SQRy registers, the ADC regular conversions must first be stopped by setting ADSTP to 1 (refer to *Section 29.4.18: Stopping an ongoing conversion (ADSTP, JADSTP)*).

Note: To convert one of the internal analog channels, the corresponding analog sources must first be enabled by programming VBATEN, VSENSESEL or VREFEN bits in the ADC12_CCR registers.

29.4.12 Channel preselection register (ADC_PCSEL)

For each channel selected through SQRx or JSQRx bits, the corresponding ADC_PCSEL bit must be configured in advance.

This ADC_PCSEL bit controls the analog switch integrated in the I/O level. The ADC input multiplexer selects the ADC input according to SQRx and JSQRx configuration with very high speed and the analog switch integrated in the I/O cannot react as fast as ADC multiplexer does. To avoid the delay due to on analog switch control on the I/O, it is necessary to preselect the input channels which is selected through the SQRx and JSQRx. The selection is based on the $V_{INP}[i]$ of each ADC input. For example, if the ADC converts ADC_IN1, PCSEL1 bit must also be set in ADC_PCSEL.

29.4.13 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 that is programmable using the SMP[2:0] bits in the ADC_SMPR1 and ADC_SMPR2 registers. It is therefore possible to select among the following sampling time values:

- SMP = 000: 5 ADC clock cycles
- SMP = 001: 6 ADC clock cycles
- SMP = 010: 12 ADC clock cycles
- SMP = 011: 20 ADC clock cycles
- SMP = 100: 36 ADC clock cycles
- SMP = 101: 68 ADC clock cycles
- SMP = 110: 391 ADC clock cycles
- SMP = 111: 814 ADC clock cycles

The total conversion time is calculated as follows:

$$T_{CONV} = \text{Sampling time} + \text{Conversion time}$$

Example

When converting a single data, the sampling time is 5 cycles and the conversion time is 17 cycles for 14-bit mode. With an $F_{adc\_ker\_ck}$ of 55 MHz:

T_{CONV} = (5 + 17) ADC clock cycles = 22 ADC clock cycles = 0.40 µs

The above result assumes that Rain << 1 KΩ (refer to the datasheet for additional sampling time to be added depending on the external resistance).

The ADC notifies the end of the sampling phase by setting the status bit EOSMP (only for regular conversion).

I/O analog switches voltage booster

The I/O analog switches resistance increases when the V_{DDA} voltage is too low. This requires to adapt the sampling time accordingly (see the device datasheet for electrical characteristics). This resistance can be minimized at low V_{DDA} by enabling an internal voltage booster (refer to the SYSCFG section for more details).

Bulb sampling mode

When the BULB bit is set in ADC_CFGR2 register, the sampling period starts immediately after the last ADC conversion. A hardware or software trigger starts the conversion after the sampling time has been programmed in ADC_SMPR1 register. The very first ADC conversion, after the ADC is enabled, is performed with the sampling time programmed in SMP bits. The Bulb mode is effective starting from the second conversion.

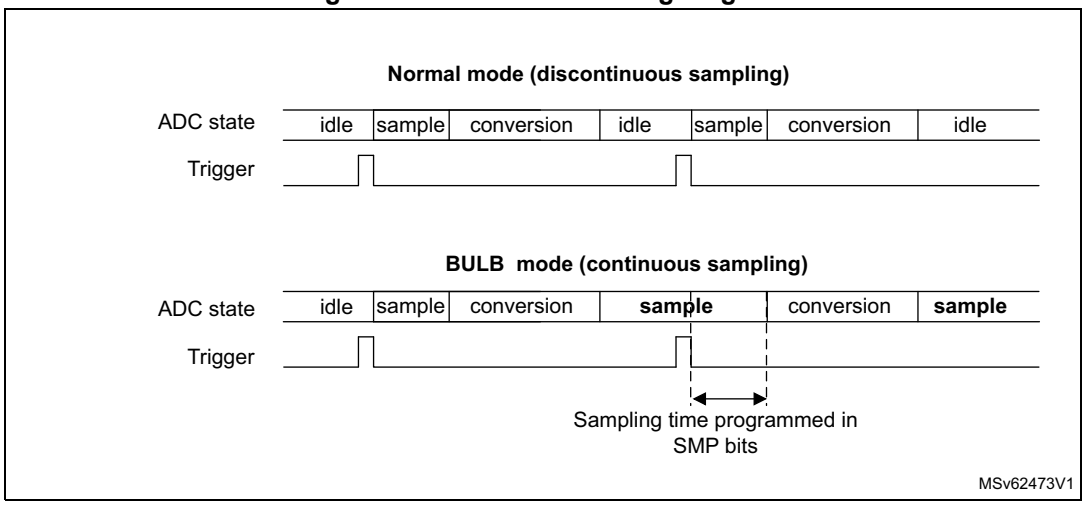
The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

The Bulb mode could not be used in continuous conversion mode or with injected channel conversion.

When the BULB bit is set, it is not allowed to set SMPTRIG bit in ADC_CFGR2.

When conversions in Bulb mode are stopped by setting ADSTP bit or when the DMA transfers are complete, the ADC must be disabled by setting ADDIS bit.

Figure 152. Bulb mode timing diagram



Sampling time control trigger mode

When the SMPTRIG bit is set, the sampling time programmed though SMPx bits is not applicable. The sampling time is controlled by the trigger signal edge.

When a hardware trigger is selected, each rising edge of the trigger signal starts the sampling period. A falling edge ends the sampling period and starts the conversion.

When a software trigger is selected, the software trigger is not the ADSTART bit in ADC_CR but the SWTRIG bit. SWTRIG bit has to be set to start the sampling period, and the SWTRIG bit has to be cleared to end the sampling period and start the conversion.

The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

This mode is not compatible with the continuous conversion mode and injected channel conversion.

When SMPTRIG bit is set, it is not allowed to set BULB bit.

29.4.14 Single conversion mode (CONT = 0)

In Single conversion mode, the ADC performs once the conversions of all 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, with software trigger selected)
- Setting the JADSTART bit in the ADC_CR register (for an injected channel, with software trigger selected)
- External hardware trigger event (for a regular or injected channel)

  ADSTART bit or JADSTART bit must be set before triggering an external event.

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 32-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 32-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.

29.4.15 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 re-starts 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 32-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 Section : Auto-injection mode).

29.4.16 Starting conversions (ADSTART, JADSTART)

The software starts ADC regular conversions by setting ADSTART to 1. When ADSTART is set, the conversion starts:

- immediately if EXTEN[1:0] = 0x0 (software trigger) or
- at the next active edge of the selected regular hardware trigger, if EXTEN[1:0] is not equal to 0x0

The software starts ADC injected conversions by setting JADSTART to 1. When JADSTART is set, the conversion starts:

- immediately, if JEXTEN[1:0] = 0x0 (software trigger) or
- at the next active edge of the selected injected hardware trigger if JEXTEN[1:0] is not equal to 0x0

Note: In Auto-injection mode (JAUTO = 1), use ADSTART bit to start regular conversions followed by auto-injected conversions (JADSTART must be kept cleared).

ADSTART and JADSTART also provide information on whether any ADC operation is ongoing. The ADC can be reconfigured while ADSTART and JADSTART are both set to 0, (the ADC is idle).

ADSTART is cleared by hardware:

- In Single mode with software trigger (CONT = 0, EXTEN[1:0] = 0x0): at any end of conversion sequence (EOS = 1)
- In Discontinuous mode with software trigger (CONT = 0, DISCEN = 1, EXTEN[1:0] = 0x0): at end of conversion (EOC = 1)
- In all other cases (CONT = x, EXTEN[1:0] = x): after executing the ADSTP assertion procedure by 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 EXTEN[1:0] ≠ *0x0), 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 (JEXTEN[1:0] = 0x0): at any end of injected conversion sequence (JEOS assertion) or at any end of sub-group processing if JDISCEN = 1
- In all other cases (JEXTEN[1:0]=x): after executing the JADSTP assertion procedure by software.

Note: When the software trigger is selected, the ADSTART bit must not be set if the EOC flag is still high.

29.4.17 Timing

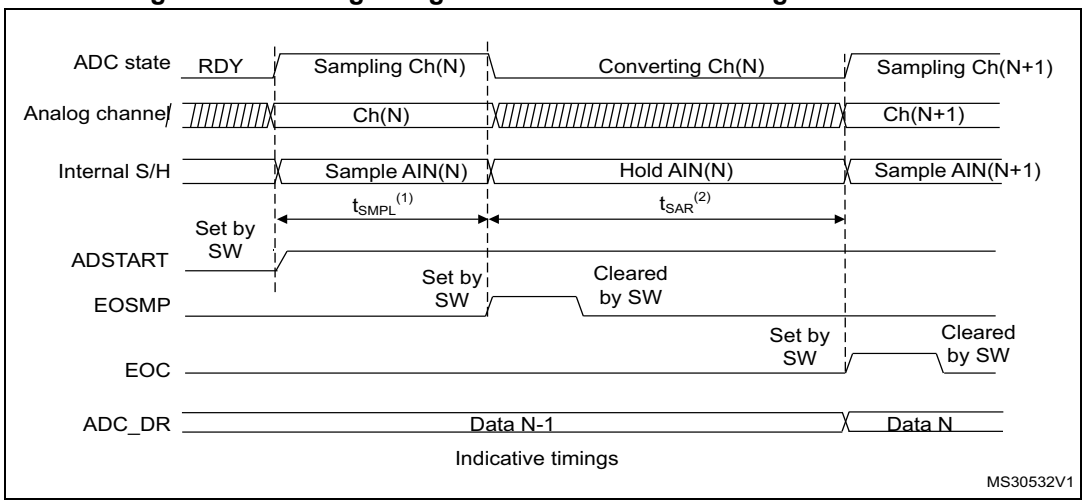
The elapsed time between the start of a conversion and the end of conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution for single conversion and minus the overlap time between the sampling and the previous SAR for continuous conversion.

In Single conversion mode:

$$T_{CONV} = T_{SMPL} + T_{SAR} = [5_{min} + 17_{14bits}] \times T_{adc\_ker\_ck}$$

$$T_{CONV} = T_{SMPL} + T_{SAR} = 22 \times T_{adc\_ker\_ck\ min\ for\ 14bits} = 400.0\ ns\ (for\ F_{adc\_ker\_ck} = 55\ MHz)$$

Figure 153. Analog to digital conversion time in single conversion



1. T_{SMPL} depends on SMP[2:0]
2. T_{SAR} depends on RES[1:0]

29.4.18 Stopping an ongoing conversion (ADSTP, JADSTP)

The software can decide to stop regular conversions ongoing by setting ADSTP to 1, and injected conversions ongoing by setting JADSTP to 1.

Stopping conversions resets the ongoing ADC operation. The ADC can then be reconfigured (for example by changing the channel selection or the trigger). It is then ready for a new operation.

Injected conversions can be stopped while regular conversions are still ongoing and vice-versa. This enables, for instance, to reconfigure the injected conversion sequence and triggers while regular conversions are still ongoing (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 re-start a new sequence).

Once this procedure is complete, ADSTP/ADSTART bits (in case of regular conversion), or JADSTP/JADSTART bits (in case of injected conversion) are cleared by hardware. 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 154. Stopping ongoing regular conversions

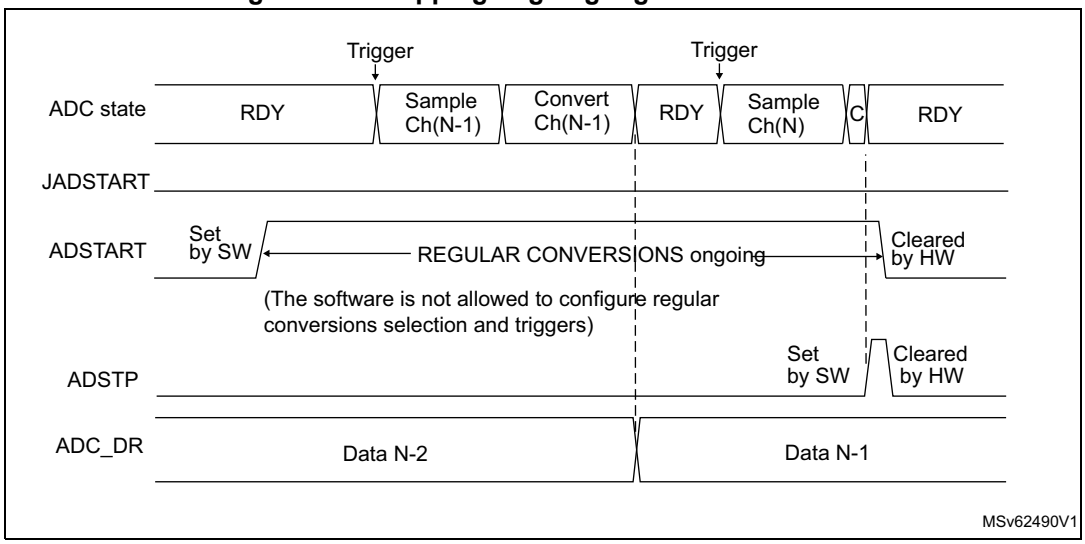
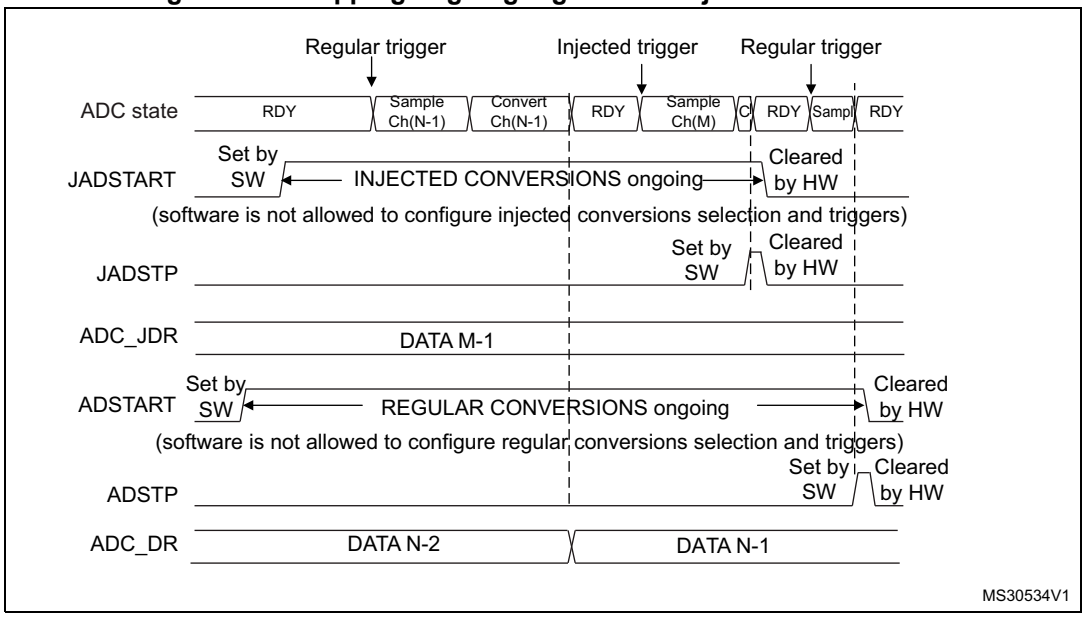


Figure 155. Stopping ongoing regular and injected conversions



29.4.19 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN[1:0], JEXTSEL, JEXTEN[1:0])

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 00, then external events are able to trigger a conversion with the selected polarity.

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 trigger which occurs while a conversion is ongoing are ignored.

- If ADSTART = 0, regular hardware triggers are ignored.
- If JADSTART = 0, injected hardware triggers are ignored.

Table 236 provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

Table 236. 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 237. Configuring the trigger polarity for injected external triggers

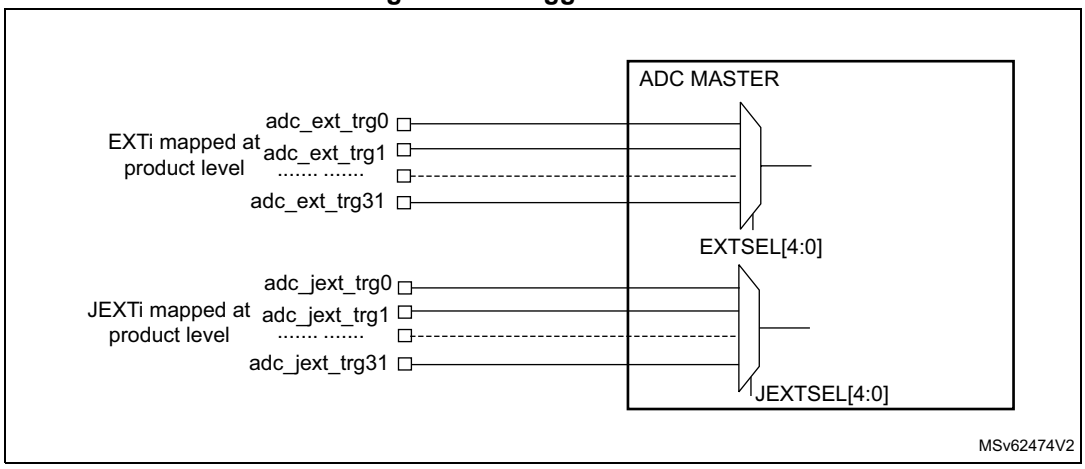
JEXTEN[1:0]	Source
00	Hardware trigger detection disabled, software trigger detection enabled
01	Hardware trigger with detection on the rising edge
10	Hardware trigger with detection on the falling edge
11	Hardware trigger with detection on both the rising and falling edges

The EXTSEL[4:0] and JEXTSEL[4:0] control bits select which events can trigger regular and injected groups conversion, out of 21 possibilities.

A regular group conversion can be interrupted by an injected trigger.

Note: The trigger selection cannot be changed on-the-fly.

Figure 156. Trigger selection



Refer to *Table ADC external triggers for regular channels and Table ADC external triggers for injected channels* in *Section 29.4.2: ADC pins and internal signals* for the connection of the above internal analog inputs to external ADC pins or internal signals.

29.4.20 Injected channel management

Triggered injection mode

To use triggered injection, the JAUTO bit must be cleared in the ADC_CFGR1 register:

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 157* 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 44 ADC clock cycles (that is two conversions with a minimum sampling time), the minimum interval between triggers must be 45 ADC clock cycles.

Auto-injection mode

If the JAUTO bit is set in the ADC_CFGR1 register, 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).

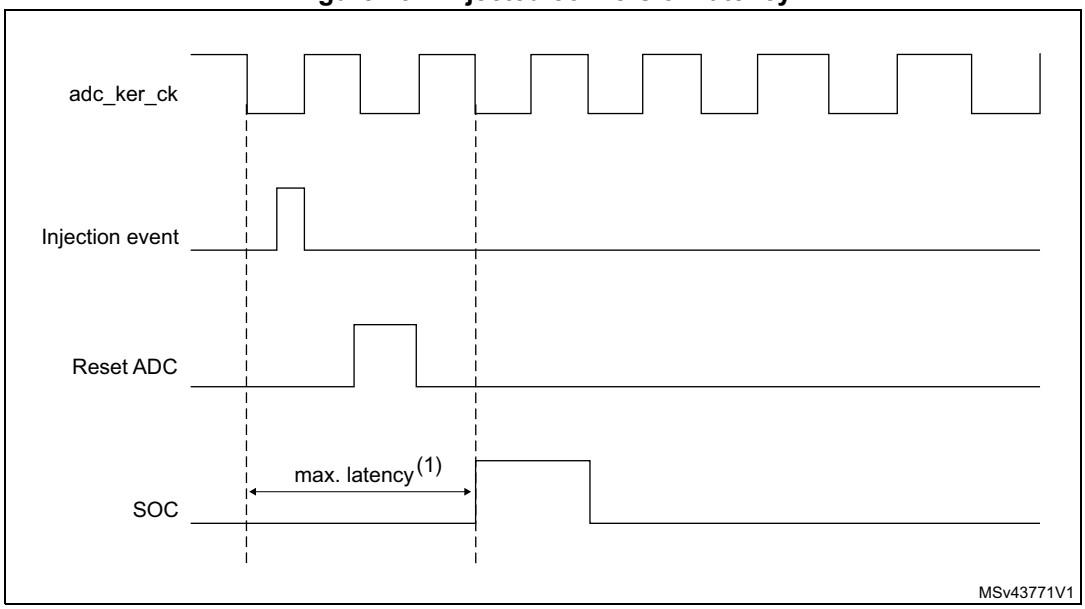
In this mode, external trigger on injected channels must be disabled.

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 is stopped upon DMA Transfer Complete event.

Figure 157. Injected conversion latency



1. The maximum latency value can be found in the electrical characteristics of the device datasheet.

29.4.21 Discontinuous mode (DISCEN, DISCNUM, JDISCEN)

Regular group mode

This mode is enabled by setting the DISCEN bit in the ADC_CFGR1 register.

It is used to convert a short sequence (sub-group) 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_CFGR1 register.

When an external trigger occurs, it starts the next n conversions selected in the ADC_SQRx 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
  - First 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 relaunch the complete sequence.

Note: 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_CFGR1 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: 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.

29.4.22 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 14, 12, 10, 8 bits by programming the control bits RES[1:0]. *Figure 162*, *Figure 163*, *Figure 164* and *Figure 165* show the conversion result format with respect to the resolution as well as to the data alignment (in continuous mode assuming no added extra sampling cycle for high input resistance).

Lower resolution enables 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 238*.

Table 238. T_{SAR} timings depending on resolution

RES	T_{SAR} (ADC clock cycles)	T_{SAR} (ns) at $F_{adc\_ker\_ck}$ = 55 MHz	$T_{adc\_ker\_ck}$ (ADC clock cycles) with Sampling time=5 ADC clock cycles	$T_{adc\_ker\_ck}$ (ns) at $F_{adc\_ker\_ck}$ = 55 MHz
14	17 ADC clock cycles	309.1	22 ADC clock cycles	400.0
12	15 ADC clock cycles	272.7	20 ADC clock cycles	363.6
10	13 ADC clock cycles	236.4	18 ADC clock cycles	327.3
8	11 ADC clock cycles	200.0	16 ADC clock cycles	290.9

29.4.23 End of conversion and 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.

29.4.24 End of conversion sequence (EOS, JEOS)

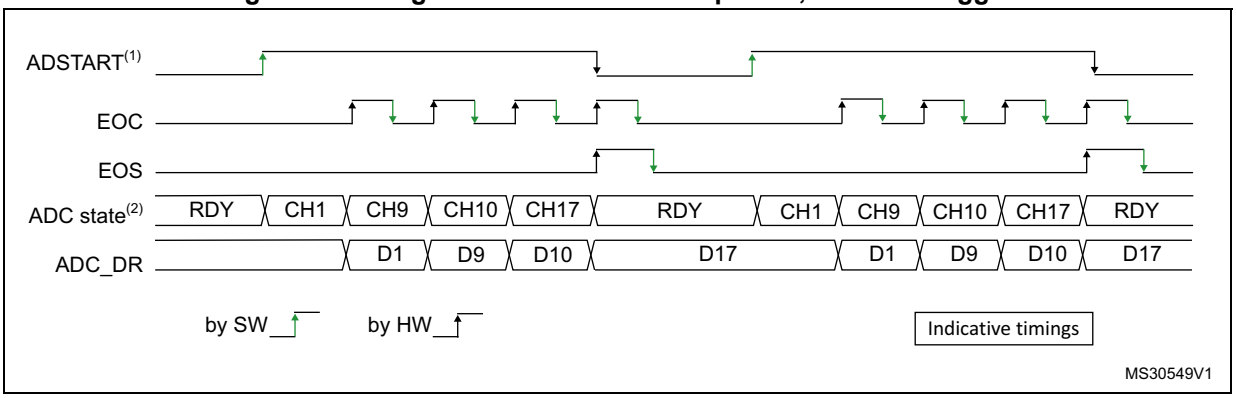
The ADC notifies the application for each end of regular sequence (EOS) and for each end of injected sequence (JEOS) event.

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.

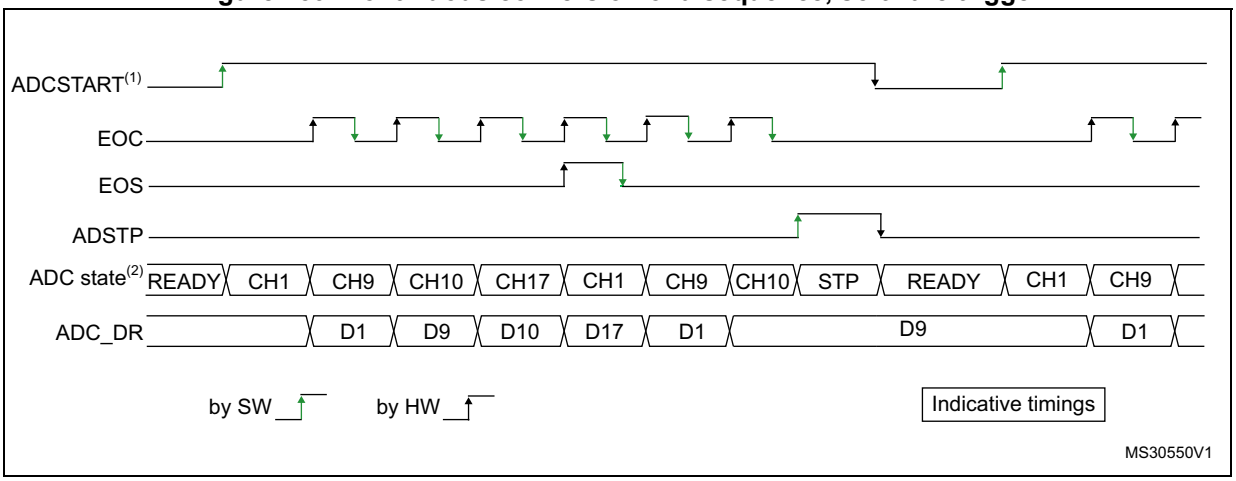
29.4.25 Timing diagrams example (single/continuous modes, hardware/software triggers)

Figure 158. Single conversions of a sequence, software trigger



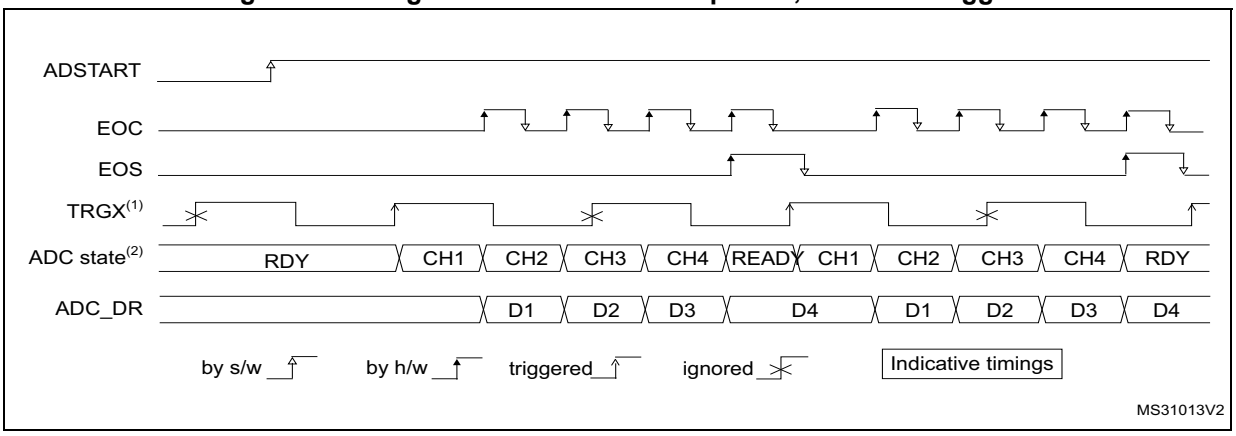
1. EXTEN[1:0] = 0x0, CONT = 0
2. Channels selected = 1,9, 10, 17; AUTDLY = 0.

Figure 159. Continuous conversion of a sequence, software trigger



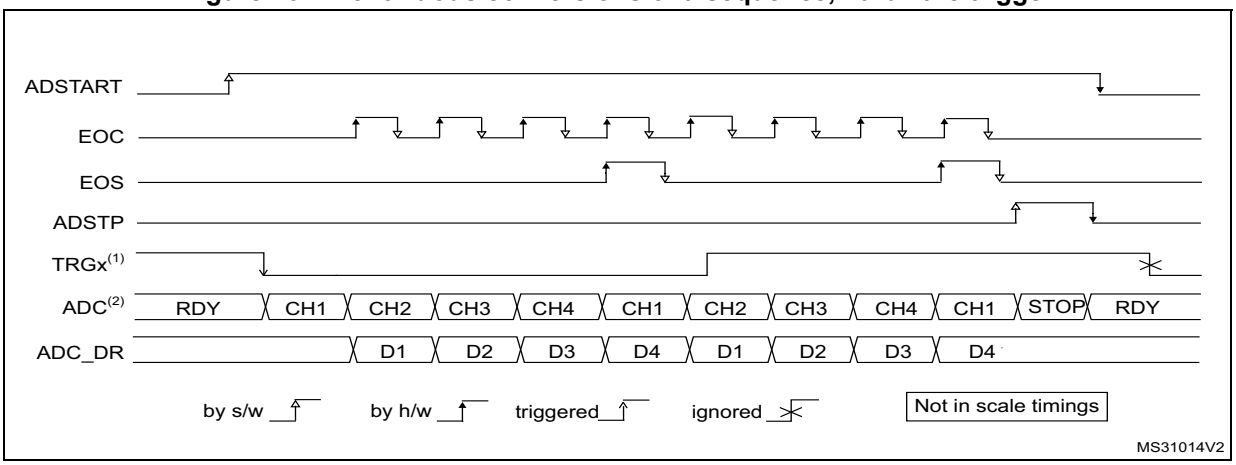
1. EXTEN[1:0] = 0x0, CONT = 1
2. Channels selected = 1,9, 10, 17; AUTDLY = 0.

Figure 160. 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 161. 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.

29.4.26 Low-frequency trigger mode (LFTRIG)

Once the ADC is enabled or when the last ADC conversion is complete, the ADC is ready to start the next conversion. The ADC needs to be started at a defined time (T_{idle}) otherwise the converted data can be corrupted due to transistor leakage (refer to the datasheet for the maximum value of T_{idle}).

If the user application requires a time higher than T_{idle} maximum value between one trigger to another for single conversion mode, or between ADC enable and the first conversion, then the ADC converter input multiplexer must be re initialized. This operation is selected by setting LFTRIG bit in ADC_CFGR2.

29.4.27 Data management

Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, USAT, SSAT)

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 32 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 32 bits wide.

The OVSS[3:0] and LSHIFT[3:0] bitfields in the ADC_CFGR2 register selects the alignment of the data stored after conversion. By default, data are right-aligned. Refer to *Figure 162*, *Figure 163*, *Figure 164* and *Figure 165* for examples of data alignment.

Note: The data can be re-aligned in normal and in oversampling mode.

Offset

An offset y (y = 1,2,3,4) can be applied to a channel by programming a value different from 0 in OFFSETy[23:0] bit field into ADC_OFRy register. The channel to which the offset is applied is programmed into the bits OFFSETy_CH[4:0] of ADC_OFRy register. The offset can be positive or negative depending on the value of POSOFF bit. When POSOFF is set to 0, the converted value is subtracted by the user-defined offset written in OFFSETy[23:0] bits. The result can be a negative value. The read data is consequently signed and the SEXT bit represents the extended sign value.

The offset value must be lower than the maximum conversion value (eg in 14-bit mode, the maximum offset value is 0x3FFF).

The offset can be used to convert unsigned data to signed data (e.g. in 14-bit mode, the offset value is equal to 0x2000).

The offset correction is also supported in Oversampling mode. For the oversampling mode, offset is subtracted before OVSS right shift applied.

Table 239 describes how the comparison is performed for all the possible resolutions for analog watchdog 1, 2, 3.

Table 239. Offset computation versus data resolution

Resolution (bits RES[1:0])	Subtraction/addition between raw converted data and offset		Result	Comments
	Raw - converted left -aligned data	Offset		
00: 14-bit	DATA[13:0]	OFFSET[13:0]	Signed or unsigned 24-bit data, right aligned to [13:0]	-
01: 12-bit	DATA[13:2],00	OFFSET[13:2]	Signed or unsigned 24-bit data, right aligned to [11:0]	The user must configure OFFSET[11:0] to 0b0000 0000 0000.

Table 239. Offset computation versus data resolution (continued)

Resolution (bits RES[1:0])	Subtraction/addition between raw converted data and offset		Result	Comments
	Raw - converted left -aligned data	Offset		
10: 10-bit	DATA[13:4],0000	OFFSET[13:4]	Signed or unsigned 24-bit data, right aligned to [9:0]	The user must configure OFFSET[3:0] to 0b0000.
11: 8-bit	DATA[13:6],000000	OFFSET[13:6]	Signed or unsigned 24-bit data, right aligned to [7:0]	The user must configure OFFSET[5:0] to 0b000000.

Figure 162, *Figure 163*, *Figure 164* and *Figure 165* show alignments for signed and unsigned data together with corresponding OVSS and LSHIFT values.

Figure 162. Right alignment (offset disabled, unsigned value)

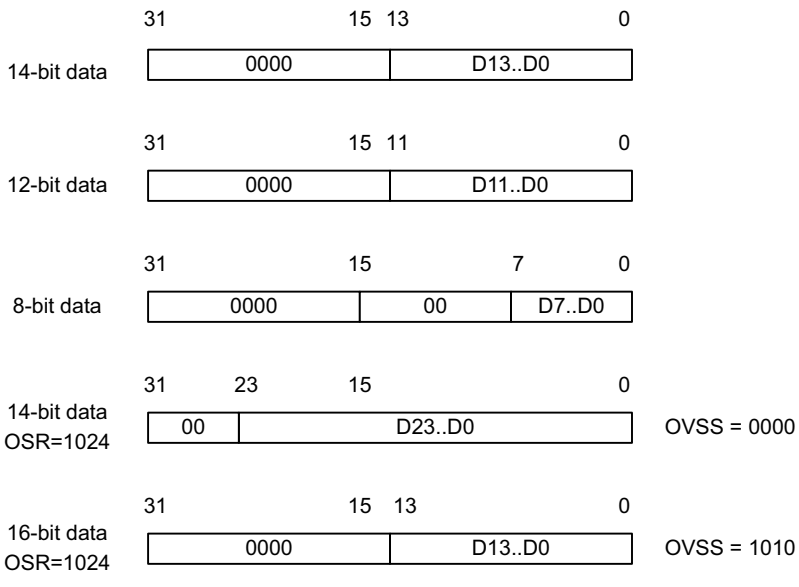


Figure 163. Right alignment (offset enabled, signed value)

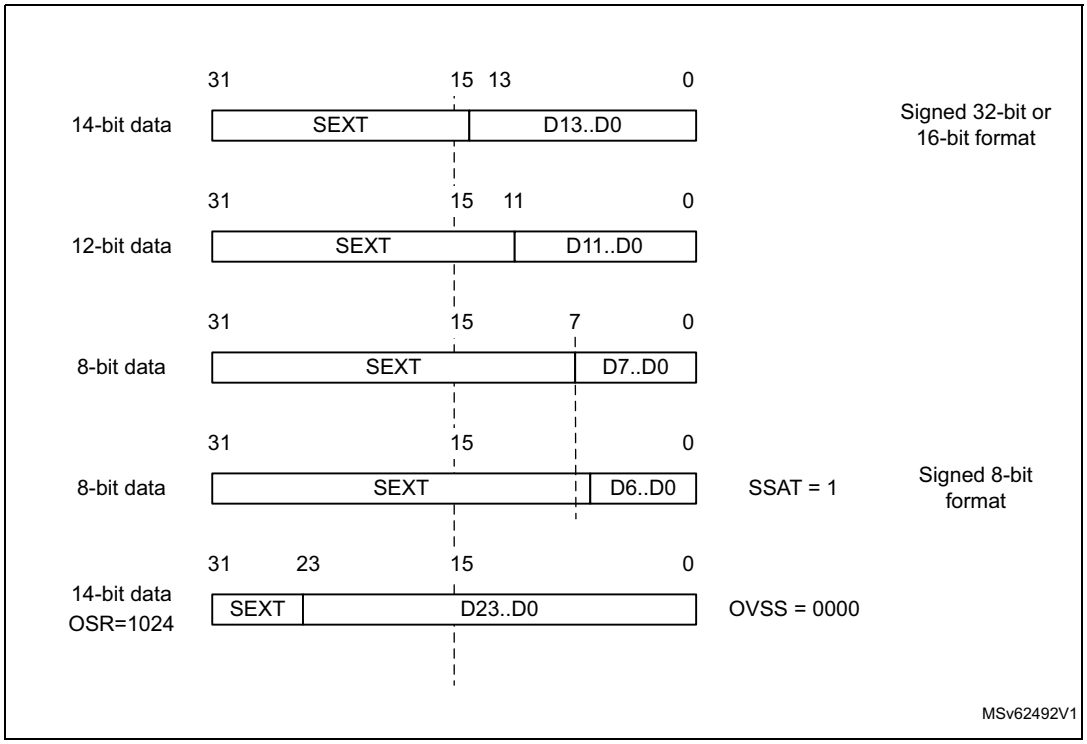


Figure 164. Left alignment (offset disabled, unsigned value)

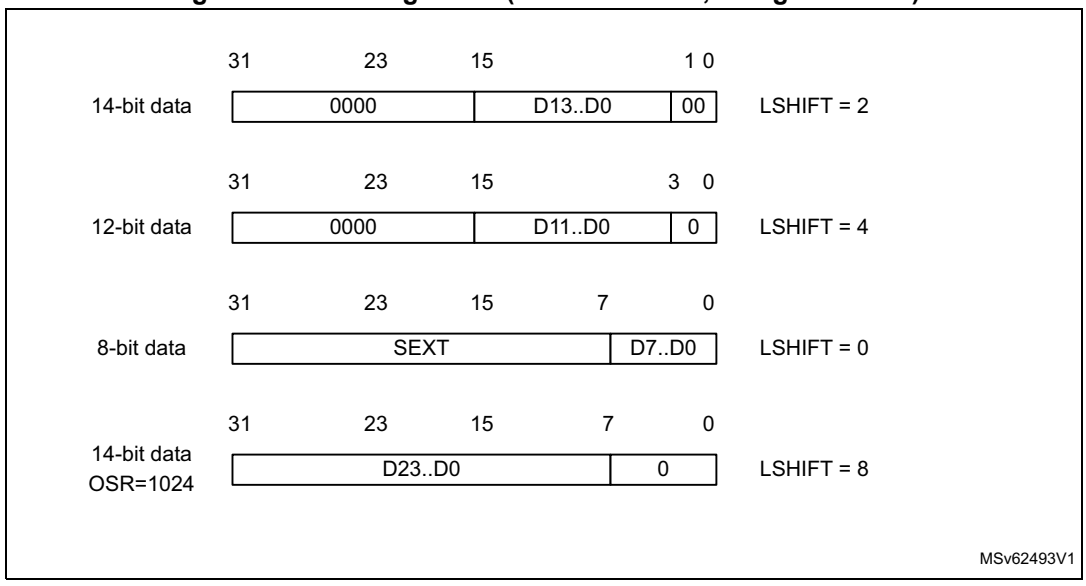
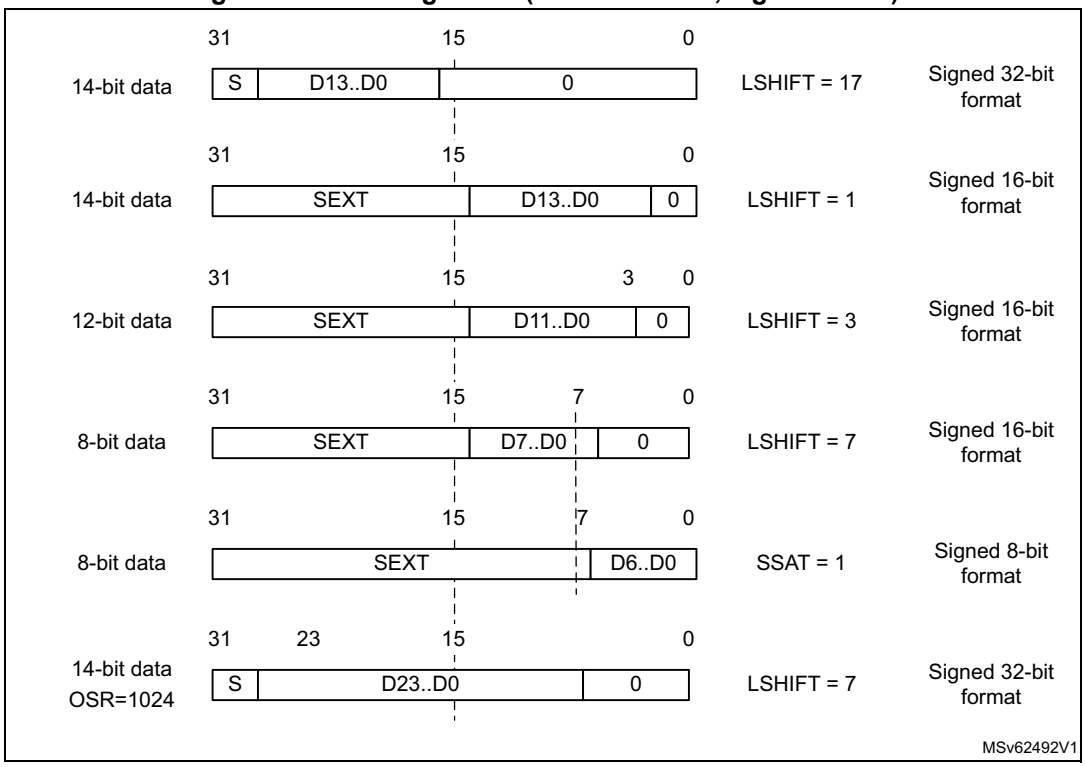


Figure 165. Left alignment (offset enabled, signed value)



Management of signed and unsigned saturation format (SSAT, USAT)

The offset correction might results in the data width to be wider than the original data.

To limit the original width, the data saturation can be enabled through SSAT and USAT bits of ADC_OFRy register.

Unsigned 14-bit data can be extended to 15-bit signed data by using an offset value different from 0x2000.

The original data width can be preserved by setting SSAT bit to 1 to limit the data width to 14 bits.

Unsigned data can be saturated to the original data width by setting USAT bit to 1.

Table 240 shows the sign-extended data format corresponding to different resolutions.

Table 240. 14-bit data formats

SSAT	USAT	Format	Data range (offset = 0x2000)
0	0	Sign-extended 15-bit significant data: SEXT[31:14] DATA[13:0]	0x0000 1FFF - 0xFFFF E000
1	0	Sign-extended 14-bit significant data: SEXT[31:13] DATA[12:0]	0x1FFF - 0xE000

Table 240. 14-bit data formats (continued)

SSAT	USAT	Format	Data range (offset = 0x2000)
0	1	Unsigned saturation14-bit significant data: DATA[13:0]	0x3FFF - 0x0000
1	1	Reserved	-

Table 241 provides numerical examples for three different offset values.

Table 241. Numerical examples for 16-bit format

Raw conversion result	Offset value	Result SSAT = 0 USAT = 0	Result SSAT = 0 USAT = 1	Result SSAT = 1 USAT = 0
0x3FFF	0x2000	0x0000 1FFF	1FFF	1FFF
0x2000		0x0000 0000	0000	0000
0x0000		0xFFFF E000	E000	E000
0x3FFF	0x2020	0x0000 1FDF	1FDF	1FDF
0x2000		0xFFFF FFE0	0000	FFE0
0x0000		0xFFFF DFE0	0000	DFE0
0x3FFF	0x1FE0	0x0000 201F	201F	1FFF
0x2000		0x0000 0020	0020	0020
0x0000		0xFFFF E020	0000	E020
0x3FFF	0x20	0x0000 3FDF	3FDF	3FDF
0x2000		0x0000 1FE0	1FE0	1FE0
0x0000		0xFFFF FFE0	0000	FFE0

Caution: SSAT must not be used in conjunction with USAT. No hardware check is performed to ensure this recommendation is respected.

Gain compensation

When the GCOMP bit is set in ADC_CFGR2 register, the gain compensation is activated on all converted data. After each conversion, data is calculated using the following formula.

$$DATA = DATA(\text{adc result}) \times (GCOMPCOEFF) / 4096$$

As GCOMPCOEFF can be programmed from 0 to 16383, the actual gain compensation factor can range from 0 to 3.999756.

Before storing the resulting data in RDATA or JDATAx registers, the LSB−1 value is evaluated to round up the data and minimize the error.

The gain compensation is also effective for the oversampling. When the gain compensation is used for the oversampling mode, the gain calculation is performed after the accumulation and right-shift operations to minimize the power consumption (the gain calculation is done

only once instead of at each conversion). The internal multiplier width is 32 bits and the input data width for the gain compensation must be less than 18 bits. When using oversampling with injected and regular conversion mode the bit ADC_CFGR2.ROVSM bit must be set to resume the pending conversion with the correct value.

ADC overrun (OVRMOD)

The overrun flag (OVR) notifies of that a buffer overrun event occurred when the regular converted data has not been read (by the CPU or the DMA) before ADC_DR FIFO (eight stages) is overflowed.

The OVR flag is set when a new conversion completes while ADC_CR register FIFO was full. An interrupt is generated if OVRIE bit is set to 1.

When an overrun event occurs, the ADC is still operating and can continue converting unless the software decides to stop and reset the sequence by setting ADSTP bit to 1.

OVR flag is cleared by software by writing 1 to it.

Data can be configured to be preserved or overwritten when an overrun event occurs by programming the OVRMOD control bit of the ADC_CFGR1 register:

- OVRMOD = 0

  The overrun event preserves the data register from being overwritten: the old data is maintained up to ADC_DR FIFO depth (8 data) and the new conversion is discarded and lost. If OVR remains at 1, further conversions occur but the result data are also discarded.
- OVRMOD = 1

  The data register is overwritten with the last conversion result and the previous unread data is lost. In this mode, ADC_DR FIFO is disabled. If OVR remains at 1, further conversions operate normally and the ADC_DR register always contains the latest converted data.

Figure 166. Example of overrun (OVRMOD = 0)

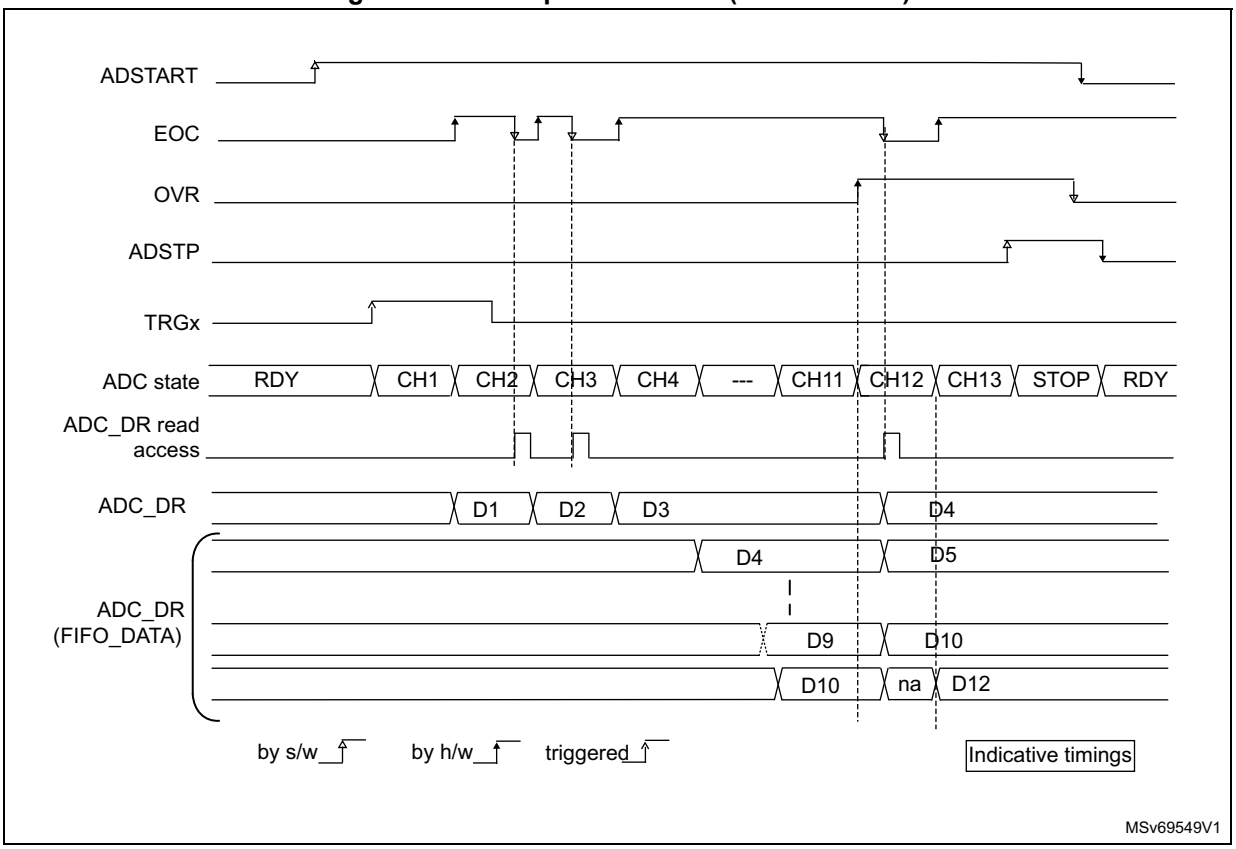
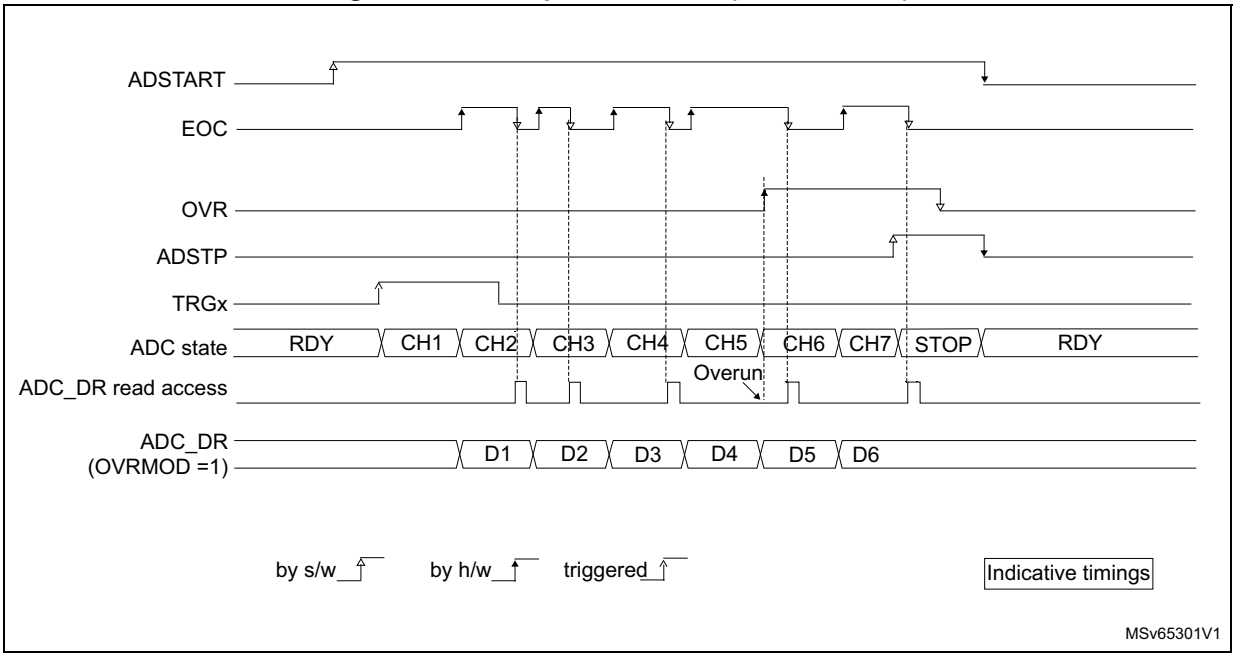


Figure 167. Example of overrun (OVRMOD = 1)



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 must 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 (for example if the device features an analog watchdog). In this case, the OVRMOD bit must be configured to 1 and the OVR flag must be ignored by the software. An overrun event does not prevent the ADC from continuing to convert and the ADC_DR register always contains the latest conversion.

Managing conversions using the DMA

Since converted channel values are stored in a unique data register, it is useful to use DMA to convert more than one channel. This avoids the loss of the data already stored in the ADC_DR register.

When the DMA mode is enabled (DMNGT[1:0] = 01 or 11 in the ADC_CFGR register in Single 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 cannot 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 the OVRMOD bit, the data is either preserved or overwritten.

The DMA transfer requests are blocked until the software clears the OSV bit.

Two different DMA modes are proposed depending on the application. They can be configured through the DMNGT[1:0] bitfield of the ADC_CFGR1 register, in Single ADC mode and in dual ADC mode, respectively:

- DMA one shot mode (DMNGT[1:0] = 01).
  This mode is suitable when the DMA is programmed to transfer a fixed number of data.
- DMA circular mode (DMNGT[1:0] = 11)
  This mode is suitable when programming the DMA in Circular mode.

DMA one shot mode (DMNGT[1:0] = 01)

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 a transfer complete interrupt occurs - refer to DMA section) 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 (DMNGT[1:0] = 11)

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.

DMA with FIFO

The output data register features an eight-stage FIFO. Two different DMA requests are generated in parallel. When a data is available, an “SREQ single request” is generated. When four data are available, a “BREQ burst request” is generated. DMA can be programmed either single transfer mode or incremental burst mode(4 beats), according to this mode, correct request line is selected by the DMA. Refer to the DMA chapter for further information.

29.4.28 Managing conversions using the MDF

The ADC conversion results can be transferred directly to the MDF.

In this case, the DMNGT[1:0] bits must be set to 10.

The ADC transfers the 16 least significant bits of the regular data register to the MDF through adcx_dat[15:0] bus, which in turn resets the EOC flag once the transfer is effective.

The data format must be in 16-bit signed format:

ADC_DR[31:16] = don’t care

ADC_DR[15] = sign

ADC_DR[14:0] = data

Any value above 16-bit signed format is truncated.

29.4.29 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 168*).
- For an injected conversion: when the JEOS bit has been cleared (see *Figure 169*).

The Auto-delayed conversion mode enables to automatically adapt the speed of the ADC to the speed of the system which reads 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 re-start 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 169*).
- 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 171*).

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 172*).

To stop a conversion in continuous auto-injection mode combined with Auto-delay mode (JAUTO = 1, CONT = 1 and AUTDLY = 1), follow the following procedure:

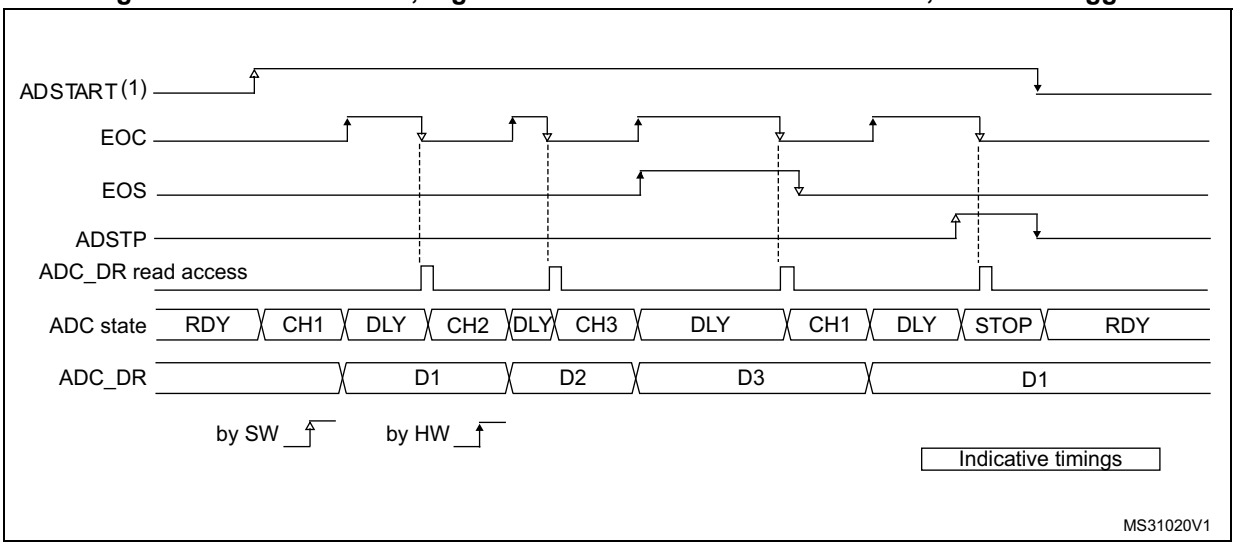
1. Wait until JEOS = 1 (no more conversions are restarted).
2. Clear JEOS.
3. Set ADSTP = 1.
4. Read the regular data.

If this procedure is not respected, a new regular sequence can re-start 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.

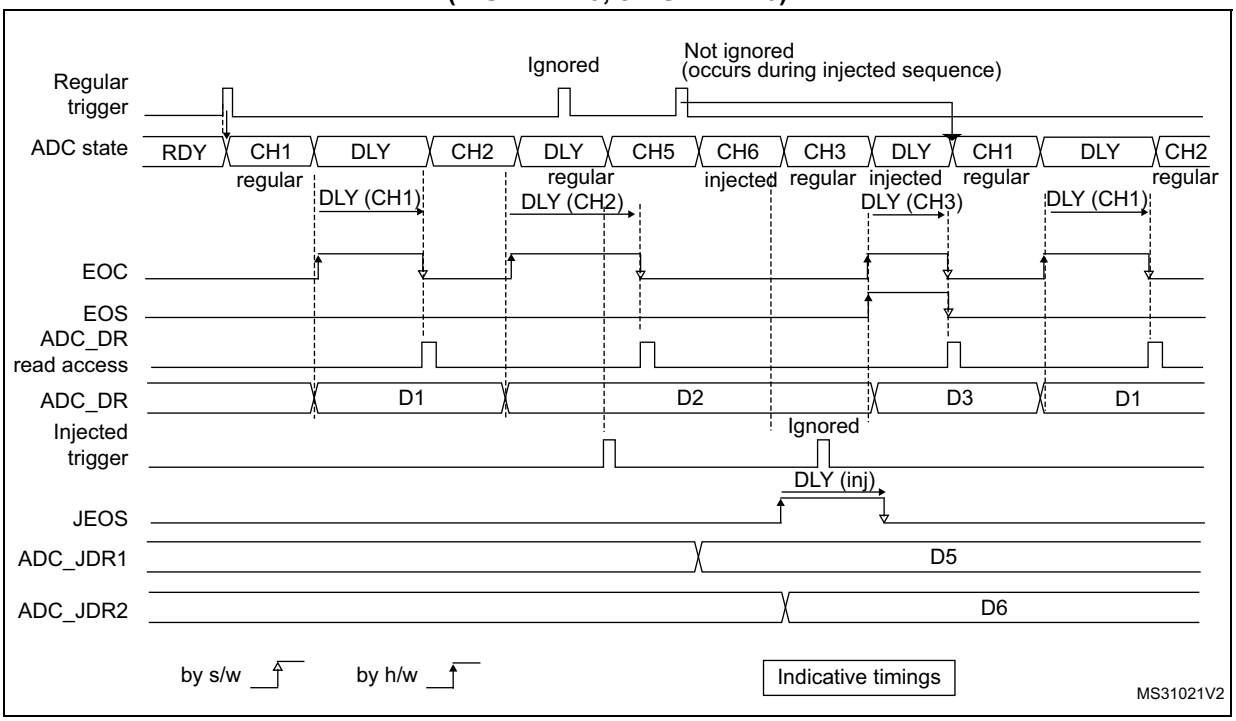
In AUTDLY mode, a hardware injected trigger event is ignored if it occurs during an already ongoing injected sequence or during the delay that follows the last injected conversion of the sequence.

Figure 168. AUTODLY = 1, regular conversion in continuous mode, software trigger



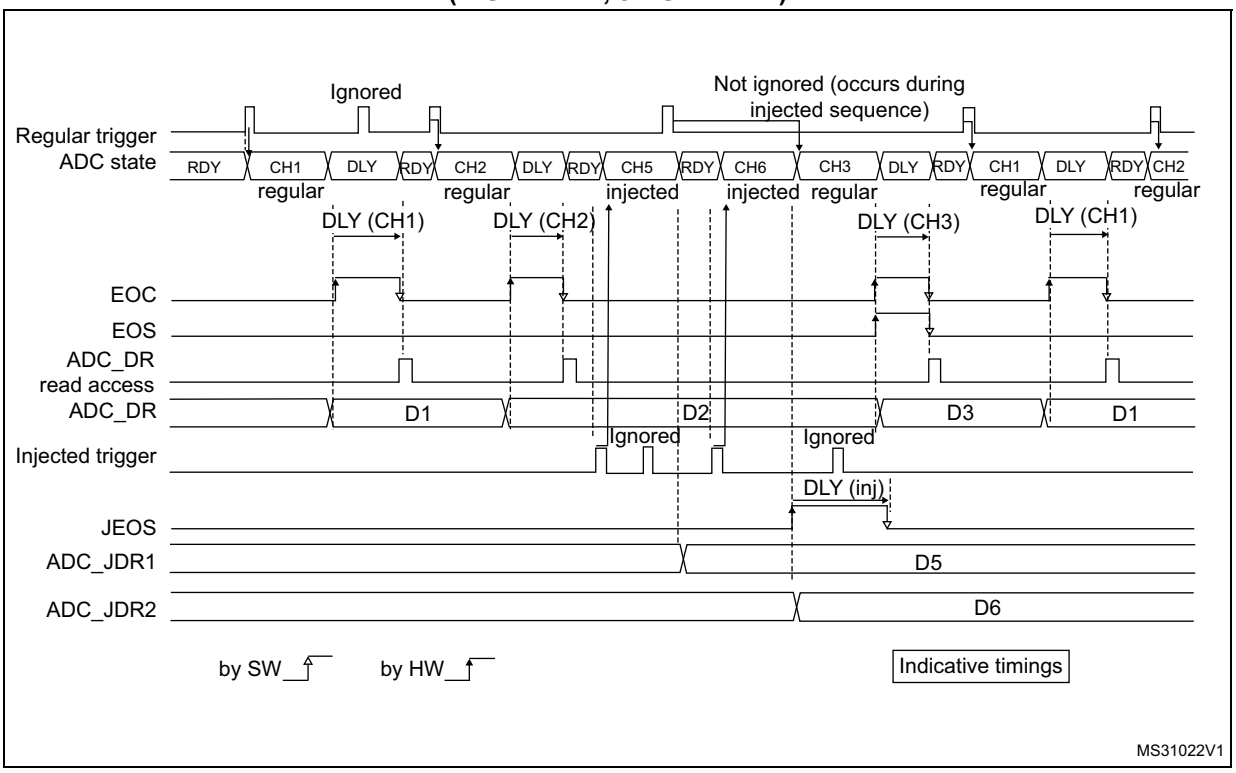
1. AUTDLY = 1.
2. Regular configuration: EXTEN[1:0] = 0x0 (SW trigger), CONT = 1, CHANNELS = 1,2,3.
3. Injected configuration DISABLED.

Figure 169. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0)



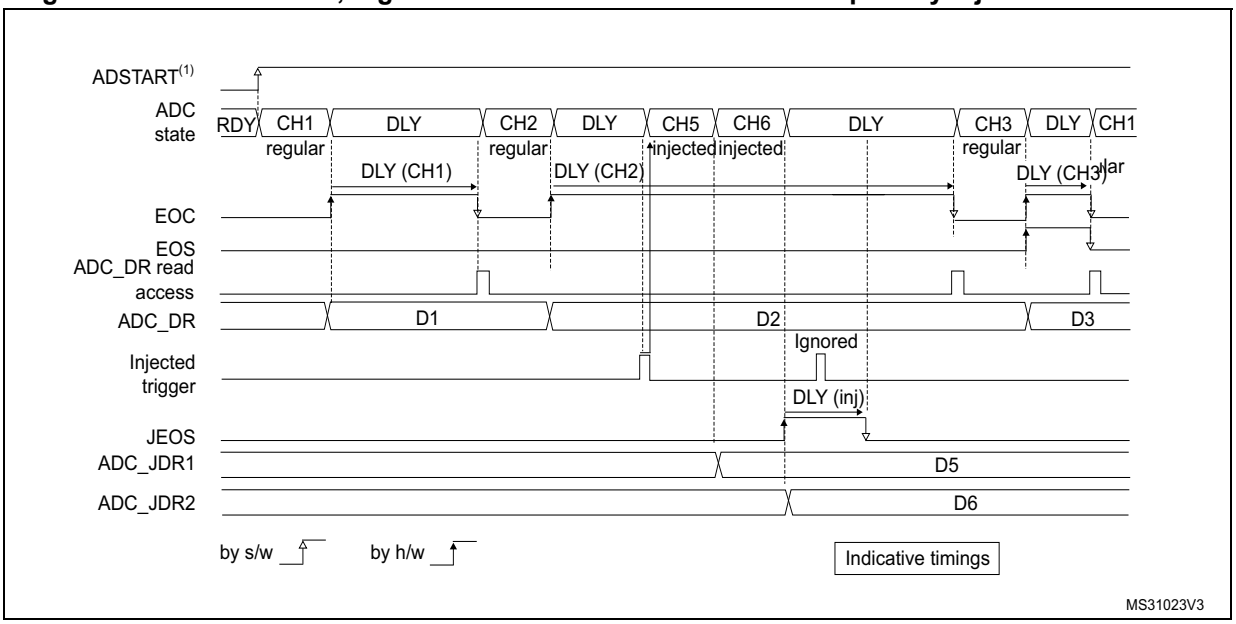
1. AUTDLY = 1.
2. Regular configuration: EXTEN[1:0] = 0x1 (HW trigger), CONT = 0, DISCEN = 0, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 0x1 (HW Trigger), JDISCEN = 0, CHANNELS = 5,6.

Figure 170. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 1)



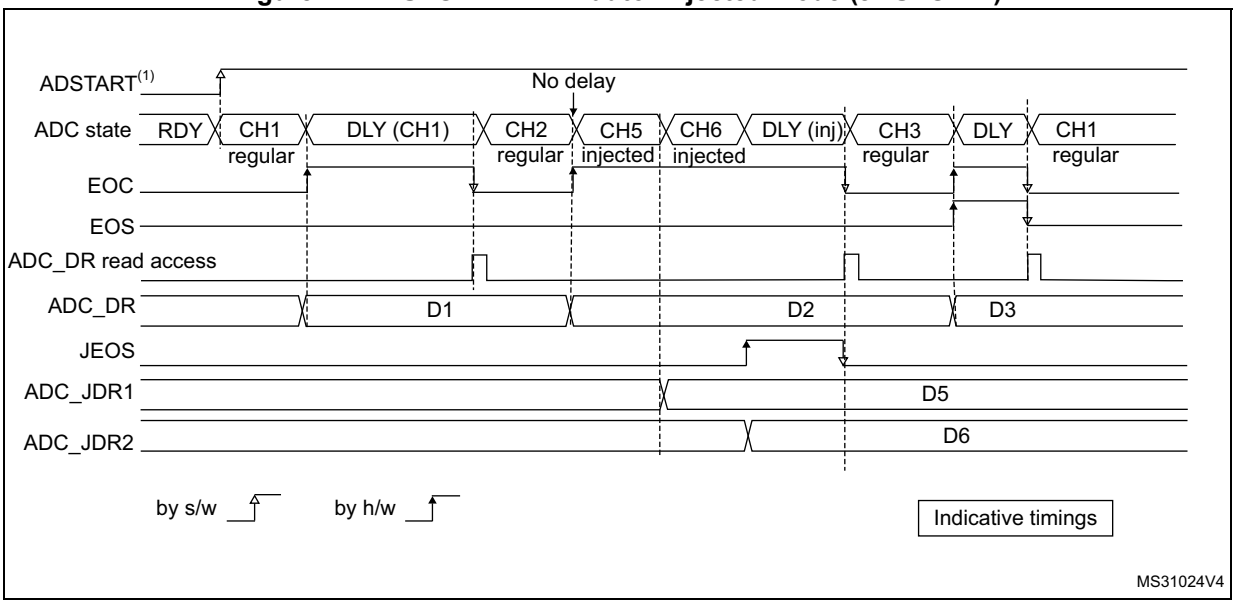
1. AUTDLY = 1.
2. Regular configuration: EXTEN[1:0] = 0x1 (HW trigger), CONT = 0, DISCEN = 1, DISCNUM = 1, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 0x1 (HW Trigger), JDISCEN = 1, CHANNELS = 5,6.

Figure 171. AUTODLY = 1, regular continuous conversions interrupted by injected conversions



1. AUTDLY = 1.
2. Regular configuration: EXTEN[1:0] = 0x0 (SW trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 0x1 (HW Trigger), JDISCEN = 0, CHANNELS = 5,6.

Figure 172. AUTODLY = 1 in auto- injected mode (JAUTO = 1)

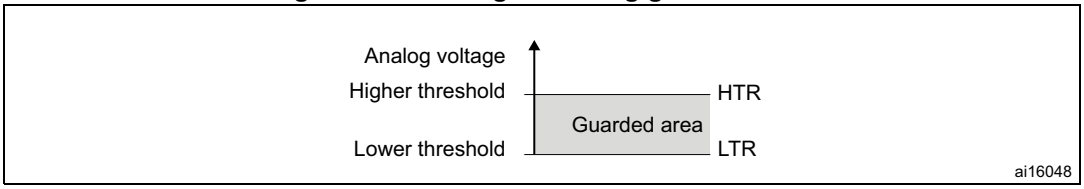


1. AUTDLY = 1.
2. Regular configuration: EXTEN[1:0] = 0x0 (SW trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2.
3. Injected configuration: JAUTO = 1, CHANNELS = 5,6.

29.4.30 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

Figure 173. Analog watchdog guarded area



AWDx flag and interrupt

An interrupt can be enabled for each of the 3 analog watchdogs by setting AWDyIE in the ADC_IER register (y = 1, 2, 3).

AWDy (y = 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_CFGR1 register. This watchdog monitors whether either one selected channel or all enabled channels remain within a configured voltage range (window).

Table 242 shows how the ADC_CFGR1 registers must be configured to enable the analog watchdog on one or more channels.

Table 242. 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 AWDyCH[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 the HTR1[24:0] bits of the ADC_HTR1 register and LTR1[24:0] of the ADC_LTR1 register for the analog watchdog 1.

The threshold can be up to 25-bits (14-bit resolution with oversampling, OSR=256).

When converting data with a resolution of less than 14 bits (according to bits RES[1:0]), the LSBs of the programmed thresholds must be kept cleared, the internal comparison being performed on the full 14-bit converted data (left aligned to the half-word boundary).

Table 243 describes how the comparison is performed for all the possible resolutions for analog watchdog 1,2,3.

Table 243. Analog watchdog 1,2,3 comparison

Resolution (bit RES[1:0])	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned(1)	Thresholds	
00: 14-bit	DATA[13:0]	LTR1[24:0] and HTR1[24:0]	-
01: 12-bit	DATA[13:2],00	LTR1[24:0] and HTR1[24:0]	User must configure LTR1[1:0] and HTR1[1:0] to 00
10: 10-bit	DATA[13:4],0000	LTR1[24:0] and HTR1[24:0]	User must configure LTR1[3:0] and HTR1[3:0] to 0000
11: 8-bit	DATA[13:6],000000	LTR1[24:0] and HTR1[24:0]	User must configure LTR1[5:0] and HTR1[5:0] to 000000

1. The watchdog comparison is performed when the oversampling, the gain compensation and the offset compensation are complete (the data that are compared can be either signed or unsigned).

Analog watchdog filter for watchdog 1

When the ADC is configured with only one input channel (selecting several channels in Scan mode not allowed), a valid ADC conversion data range can be configured through the ADC_LTR1 and ADC_HTR1 register:

- When converted data belong to the range defined in ADC_LTR1 and ADC_HTR1, a DMA request is generated.
- Otherwise, no DMA request is issued. RDATA register is updated at each conversion. If data are out-of-range a number of times higher than the value specified in AWDFILT bit of ADC_HTR1, the AWDx flag is set an the corresponding interrupt is issued.

Description of analog watchdog 2 and 3

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the corresponding bits in AWDCHy[19:0] (y=2,3).

The corresponding watchdog is enabled when any bit of AWDCHy[19:0] (y=2,3) is set.

The threshold can be up to 25 bits (14-bit resolution with oversampling, OSR = 1024, and offset conversion in signed format) and are programmed with the ADC_HTR2, ADC_LTR2, ADC_LTR3, and ADC_HTR3 registers.

When converting data with a resolution of less than 14 bits (according to bits RES[1:0]), the LSBs of the programmed thresholds must be kept cleared, the internal comparison being performed on the full 14-bit converted data (left aligned).

ADC_AWDy_OUT signal output generation

Each analog watchdog is associated to an internal hardware signal ADC_AWDy_OUT (y being the 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 ADC_AWDy_OUT signal as ETR.

ADC_AWDy_OUT is activated when the associated analog watchdog is enabled:

- ADC_AWDy_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADC_AWDy_OUT is reset after the end of the next guarded conversion which is inside the programmed thresholds (It remains at 1 if the next guarded conversions are still outside the programmed thresholds).
- ADC_AWDy_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: AWDy flag has no influence on the generation of ADC_AWDy_OUT (ex: ADCy_AWDy_OUT can toggle while AWDx flag remains at 1 if the software did not clear the flag).

Figure 174. ADCy_AWDx_OUT signal generation (on all regular channels)

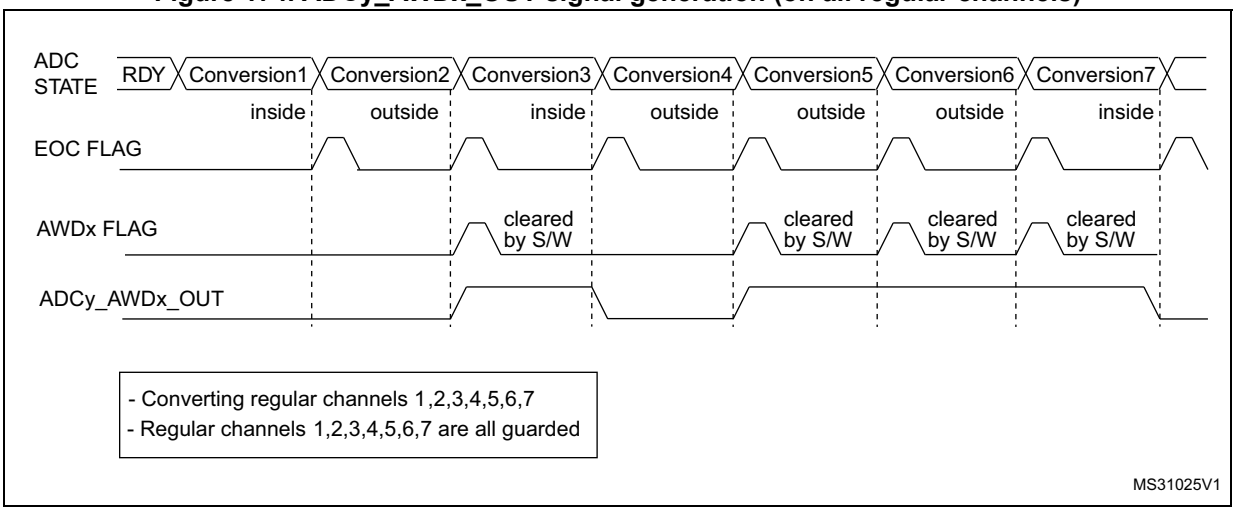


Figure 175. ADC_AWDx_OUT signal generation (AWDx flag not cleared by SW)

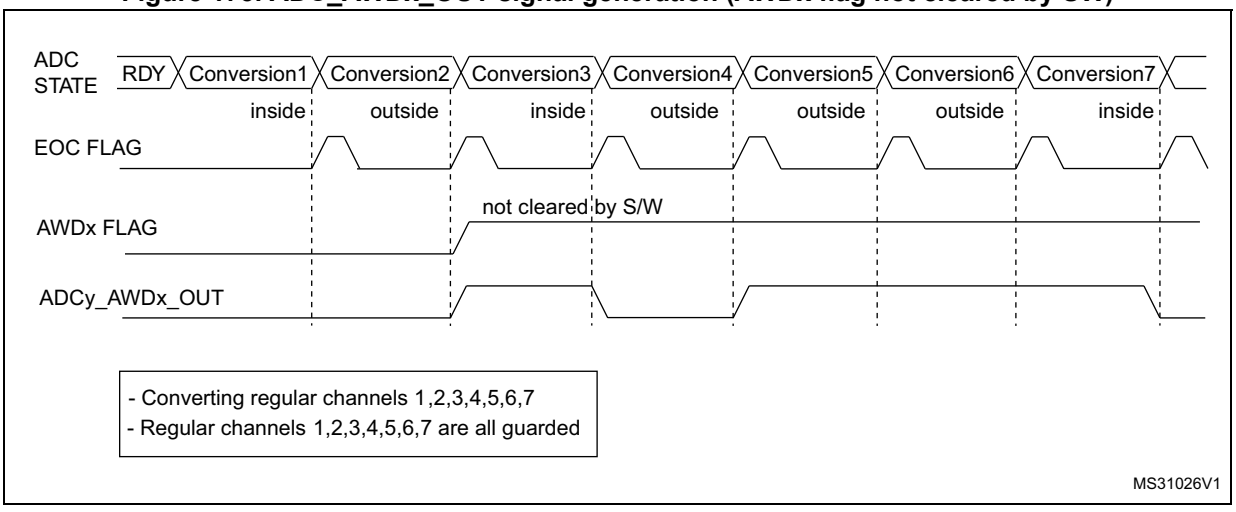


Figure 176. ADC_AWDx_OUT signal generation (on a single regular channel)

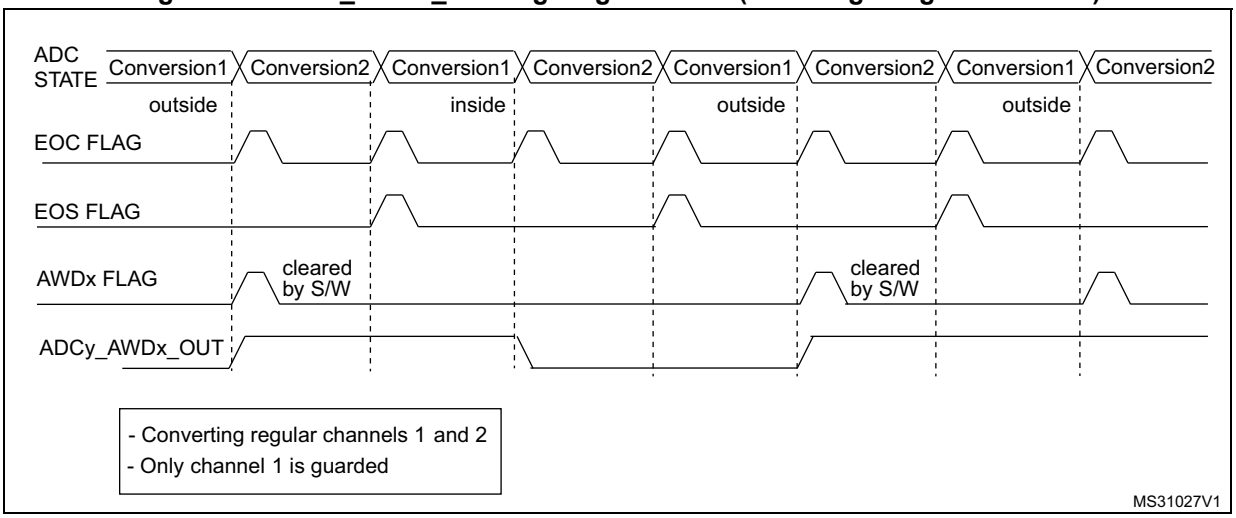
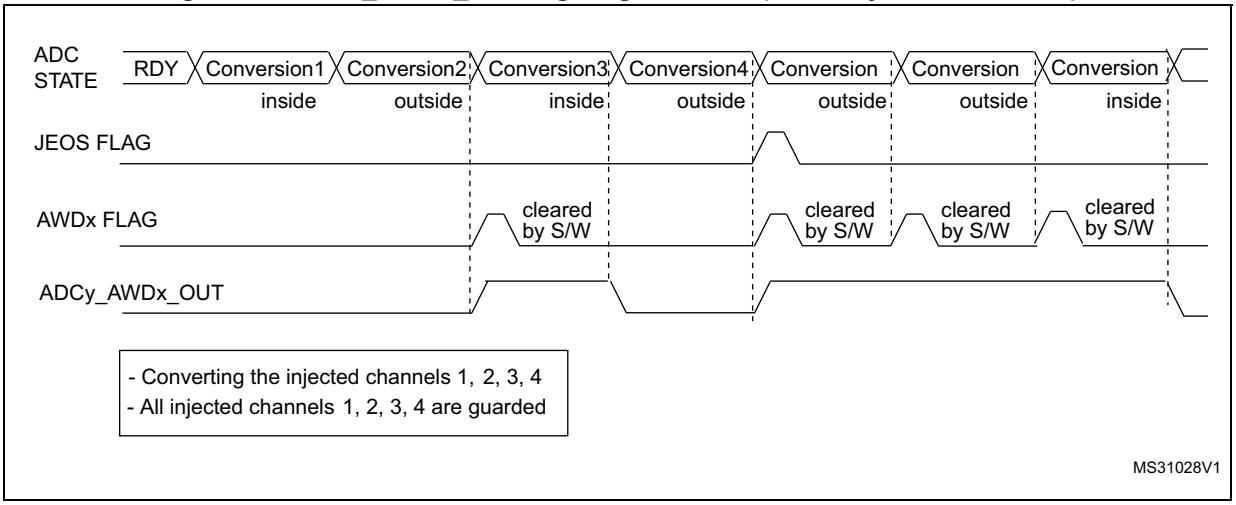


Figure 177. ADC_AWDx_OUT signal generation (on all injected channels)



Analog watchdog threshold control

LTRx[24:0] and HTRx[24:0] can be changed when an analog-to-digital conversion is ongoing (that is between the start of conversion and the end of conversion of the ADC internal state). If LTRx[24:0] and HTRx[24:0] are updated during the ADC conversion of the ADC guarded channel, the watchdog function is masked for this conversion. This masking is removed at the next start of conversion, resulting in a analog watchdog thresholds to be applied from the next ADC conversion. The analog watchdog comparison is performed at each end of conversion. If the current ADC data is out of the new interval, no interrupt and AWDx_OUT signal are issued. The Interrupt and the AWD generation only happen at the end of the conversion which started after the threshold update. If AWD_xOUT is already asserted, programming the new thresholds does not deassert the AWD_OUT signal.

Analog watchdog with gain and offset compensation

When gain and offset compensation are enabled, the analog watchdog compares the threshold after the compensated data.

29.4.31 Oversampler

The oversampling unit performs data preprocessing to offload the CPU. It is able to handle multiple conversions and average them into a single data with increased data width, up to 24-bit (14-bit values and OSR = 1024).

It provides a result with the following form, where N and M can be adjusted:

$$\text{Result} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{Conversion}(t_n)$$

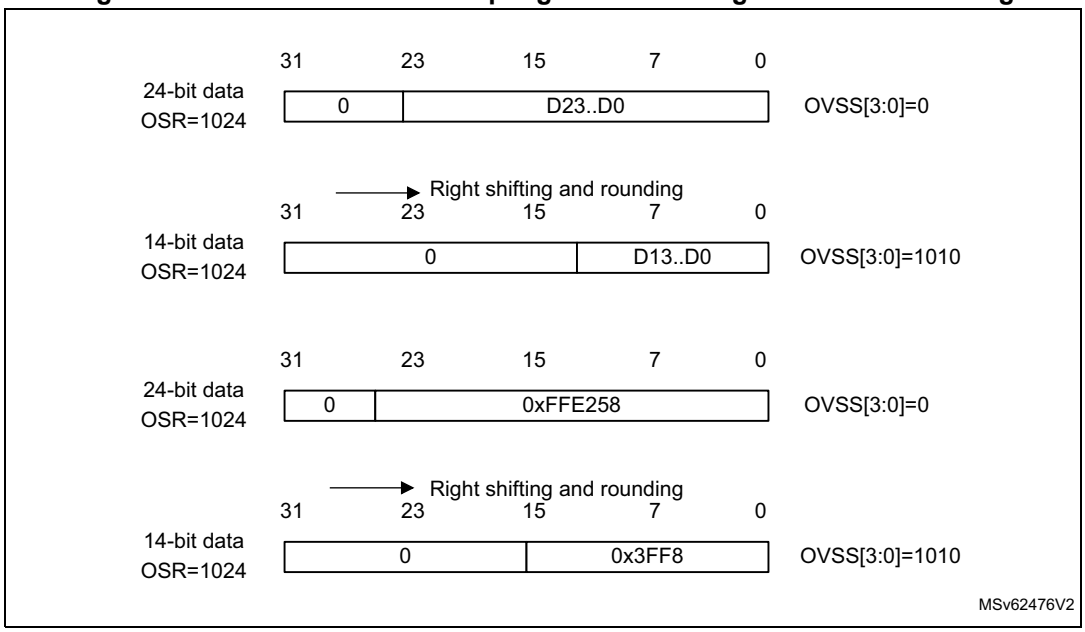
It enables the following functions to be performed by hardware: averaging, data rate reduction, SNR improvement, basic filtering.

The oversampling ratio N is defined using the OSR[9:0] bits in the ADC_CFGR2 register, and can range from 2x to 1024x. The division coefficient M consists of a right bit shift up to 10 bits, and is defined using the OVSS[3:0] bits in the ADC_CFGR2 register.

The summation unit can yield a result up to 24 bits (1024 x 14-bit results), which can be left or right shifted. When right shifting is selected, it is rounded to the nearest value using the least significant bits left apart by the shifting, before being transferred into the ADC_DR data register.

Figure 178 gives a numerical example of the processing, from a raw 24-bit accumulated data to the final 14-bit result.

Figure 178. 14-bit result oversampling with 10-bits right shift and rounding



The conversion timings in Oversampled mode do not change: the sample time is maintained 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).

Operating modes supported during oversampling

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 (configured through RES[1:0] bits in ADC_CFGR1 register) are accumulated, truncated, rounded and shifted in the same way as 14-bit conversions.

Note: The Alignment mode is not available when working with oversampled data. The ALIGN bit in ADC_CFGR 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 differences:

- The RES[1:0] bits are ignored, comparison is always done using the full 25-bit values, HTRx[24:0] and LTRx[24:0].
- The comparison is performed on the oversampled accumulated value before shifting.

Note: Care must be taken when using high shifting values, since this reduces the comparison range. For instance, if the oversampled result is shifted by 4 bits, thus yielding an 8-bit right-aligned data, the effective analog watchdog comparison can only be performed on 8 bits. The comparison is done between ADC_DR[11:4] and HT[7:0]/LT[[7:0]. HT[11:8]/LT[11:8] must be kept reset.

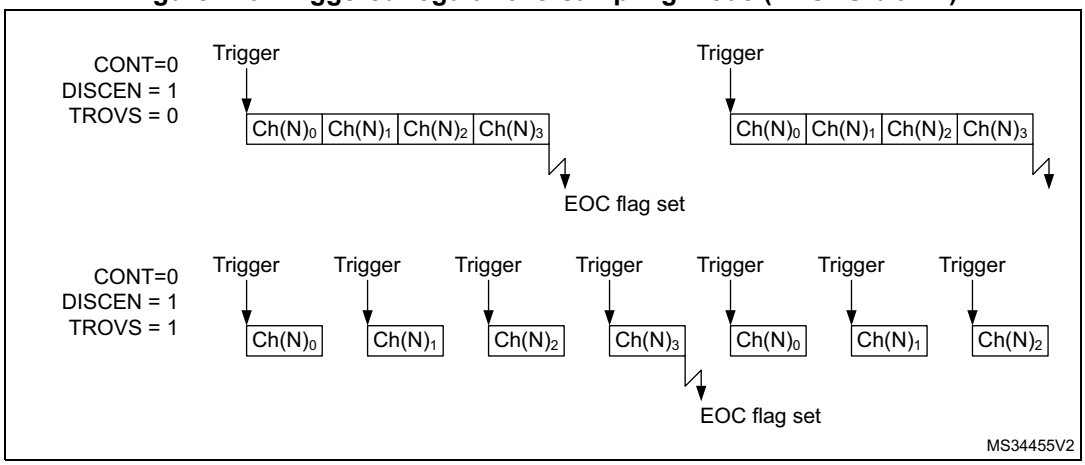
Triggered mode

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.

The *Figure 179* 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 179. Triggered regular oversampling mode (TROVS bit = 1)



Injected and regular sequencer management when oversampling

In Oversampling mode, injected and regular sequencers can have different behaviors. 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).

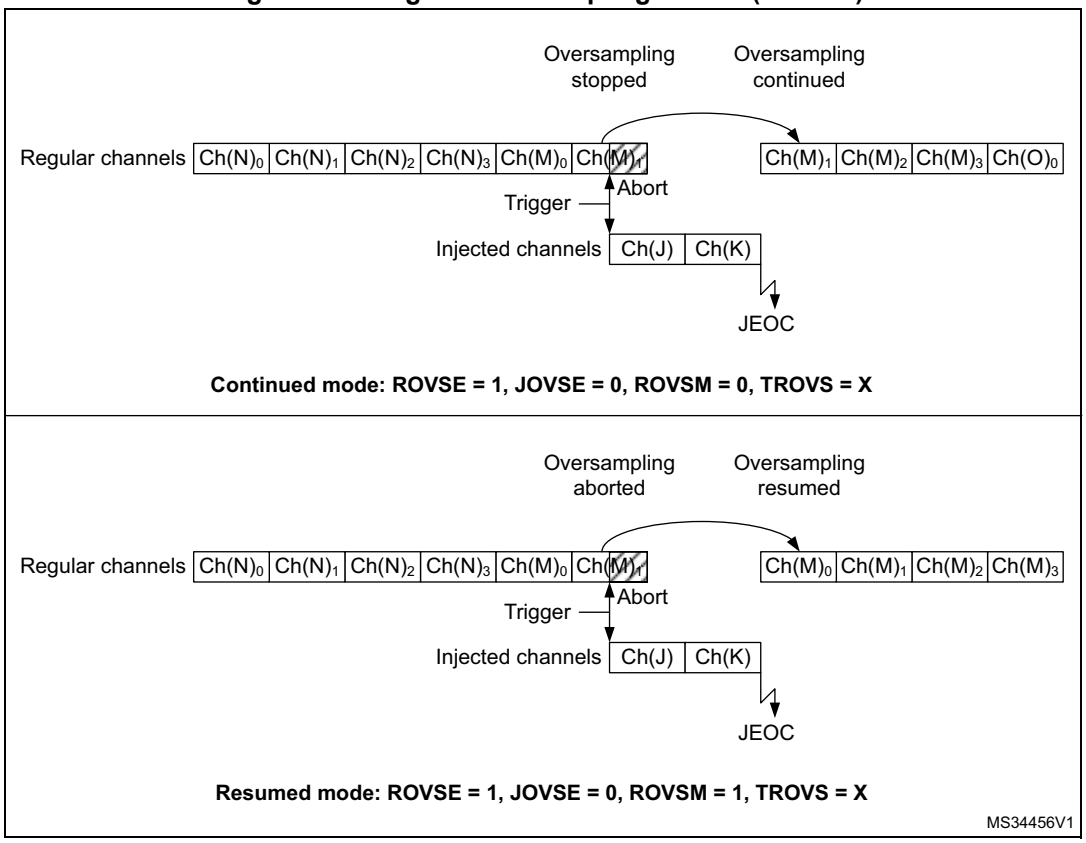
Oversampling regular channels only

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 is complete whatever the injection frequency (providing at least one regular conversion can be completed between triggers);
- In Resumed mode, the accumulation restarts from 0 (previous conversions results are ignored). This mode guarantees that all data used for oversampling were converted back-to-back within a single time-slot. 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 completed and the regular sequencer is blocked.

Figure 180 gives examples for an oversampling ratio of 4x.

Figure 180. Regular oversampling modes (4x ratio)



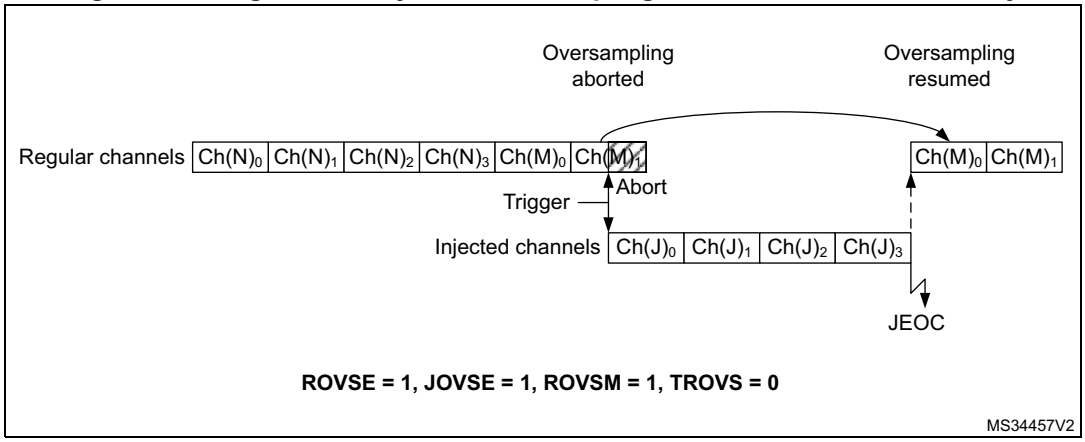
Oversampling Injected channels only

The injected oversampling mode bit, JOVSE, enables oversampling solely for conversions in the injected sequencer.

Oversampling regular and Injected channels

Both ROVSE and JOVSE bits can be set simultaneously. In this case, the Regular oversampling mode is forced to Resumed mode (ROVSM bit ignored), as shown in *Figure 181*.

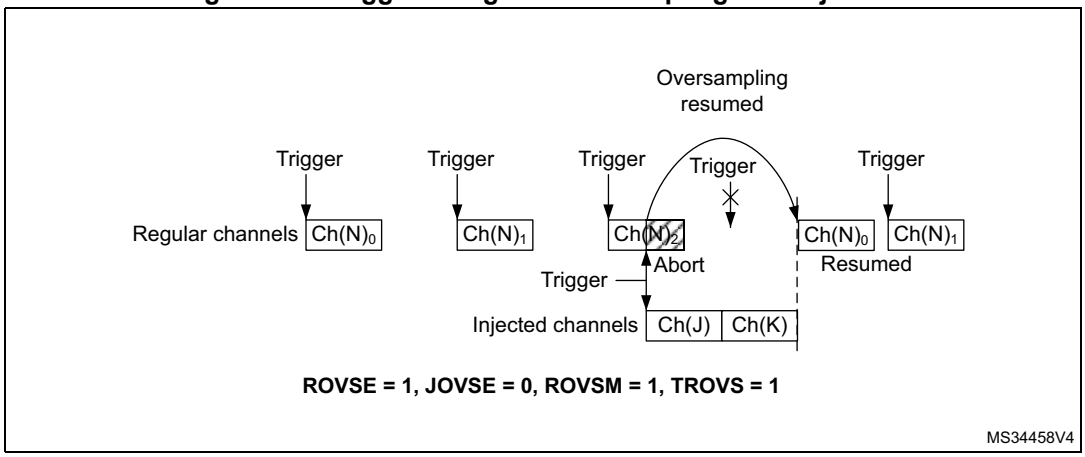
Figure 181. Regular and injected oversampling modes used simultaneously



Triggered regular oversampling with injected conversions

Injected conversions can be performed in Triggered regular mode. In this case, the Injected mode oversampling mode must be disabled, and the ROVSM bit is ignored (Resumed mode forced). The JOVSE bit must be reset. The behavior is shown in *Figure 182*.

Figure 182. Triggered regular oversampling with injection

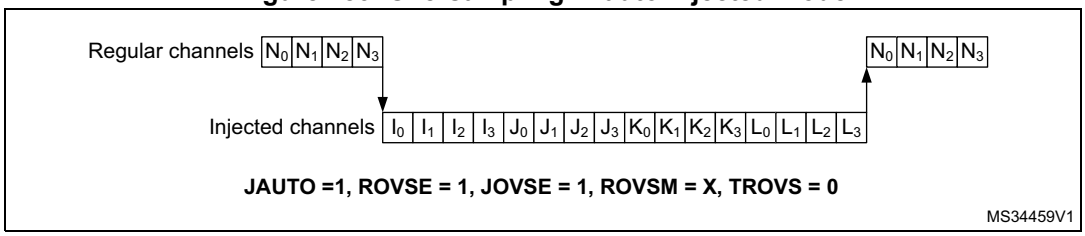


Auto-injected mode

It is possible to oversample auto-injected sequences and have all conversions results stored in registers. This enables to save a DMA resource. This mode is available only when 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.

Figure 183 shows how the conversions are sequenced.

Figure 183. 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.

Summary of combined modes

Table 244 summarizes all mode combinations, including non-supported modes.

Table 244. Summary of oversampler operating modes

Regular over-sampling: ROVSE bit	Injected over-sampling: JOVSE bit	Oversampler mode: ROVSM bit 0 = continued 1 = resumed	Triggered Regular mode: TROVS bit	Comment
1	0	0 (continued)	0	Regular continued mode
1	0	0 (continued)	1	Not supported
1	0	1 (resumed)	0	Regular resumed mode
1	0	1 (resumed)	1	Triggered regular resumed mode
1	1	0 (continued)	X	Not supported
1	1	1 (resumed)	0	Injected and regular resumed mode
1	1	1 (resumed)	1	Not supported
0	1	X	X	Injected oversampling

29.4.32 Temperature sensor

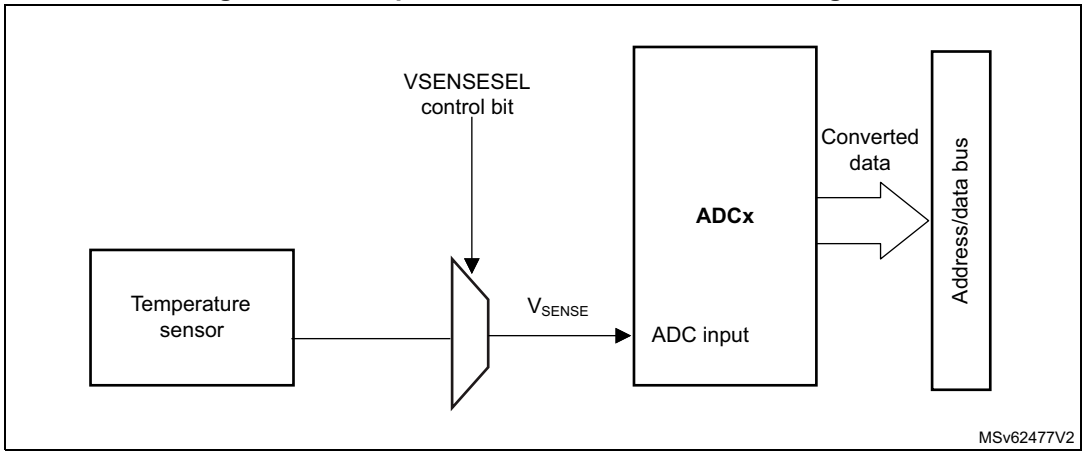
The temperature sensor can measure the device junction temperature (T_J) in the –40 to 125 °C temperature range.

The temperature sensor is internally connected ADC input channels that are used to convert the sensor output voltage to a digital value (see *Section 29.4.4: ADC connectivity* for more details). The sampling time for the temperature sensor analog pin must be greater than the stabilization time specified in the device datasheet.

When it is not in use, the sensor can be placed in power-down mode.

Figure 184 shows the block diagram of the temperature sensor.

Figure 184. Temperature sensor channel block diagram



Reading the temperature

To use the sensor:

1. Select the input channels to which the temperature sensor is connected (with the appropriate sampling time).
2. Program with the appropriate sampling time (refer to electrical characteristics section of the device datasheet).
3. Set the VSENSESEL bit in the ADC12_CCR register to wake up the temperature sensor from power-down mode.
4. Start the ADC conversion.
5. Read the resulting 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 *Section 29.3: ADC implementation* for more information on TS_CAL1 and TS_CAL2 calibration points.

Note: The sensor has a startup time after waking from power-down mode and before it can output at the correct level. The ADC also has a startup time after power-on. As a result, to minimize the delay, the ADEN and VSENSESEL bits must be set simultaneously.

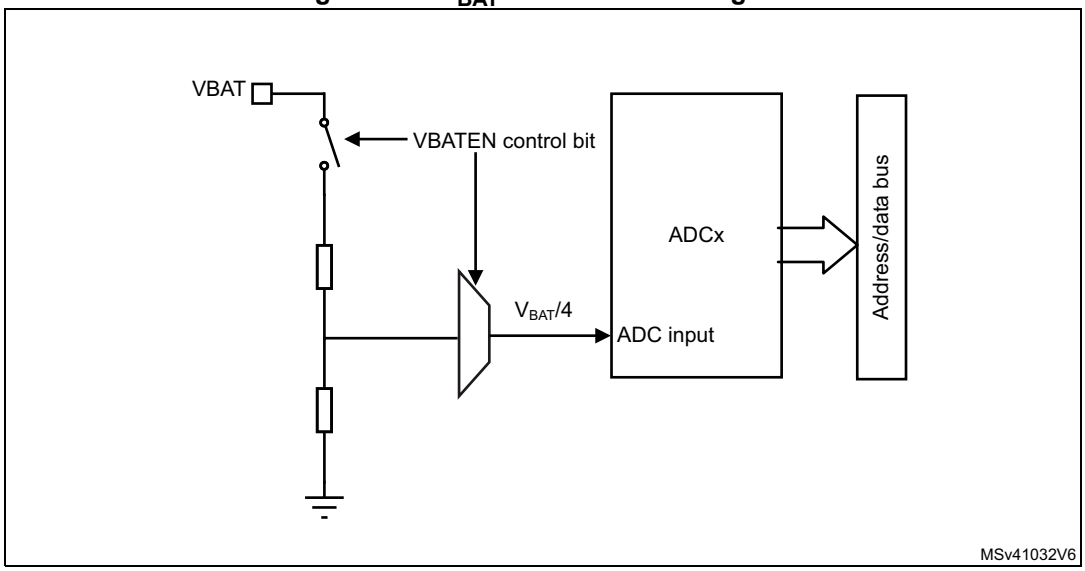
29.4.33 V_{BAT} supply monitoring

The VBATEN bit in the ADC12_CCR register is used to switch to the battery voltage. As the V_{BAT} voltage could be higher than V_{DDA}, the VBAT pin is internally connected to a bridge divider by 4 to ensure the correct operation of the ADC. This bridge is automatically enabled when VBATEN is set, to connect $V_{BAT}/4$ to the corresponding ADC input channels (see *Section 29.4.4: ADC connectivity* for more details). As a consequence, the converted digital value is one fourth 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 $V_{BAT}/4$ voltage.

Figure 185 shows the block diagram of the V_{BAT} sensing feature.

Figure 185. V_{BAT} channel block diagram



Note: The VBATEN bit of the ADC12_CCR register must be set to enable the conversion of the ADC internal channels to which VBAT is connected (see Section 29.4.4: ADC connectivity for more details).

29.4.34 Monitoring the internal voltage reference

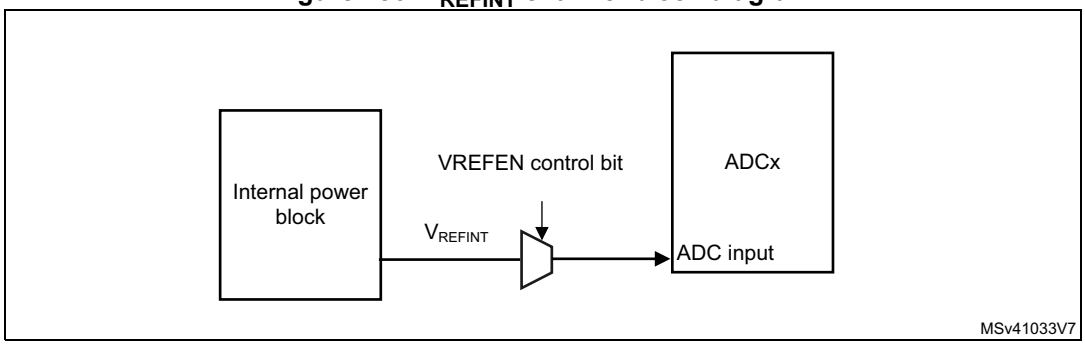
The internal voltage reference can be monitored to have a reference point for evaluating the ADC V_{REF+} voltage level.

Refer to *Section 29.4.4: ADC connectivity* for details on the ADC input channels to which the internal voltage reference is internally connected.

The sampling time for this channel must be greater than the stabilization time specified in the device datasheet.

Figure 185 shows the block diagram of the V_{REFINT} sensing feature.

Figure 186. V_{REFINT} channel block diagram



Note: The VREFEN bit of the ADC12_CCR register must be set to enable the conversion of the ADC internal channels to which VREFINT is connected (see Section 29.4.4: ADC connectivity for more details).

Calculating the actual V_{REF+} voltage using the internal reference voltage

The V_{DDA} power supply voltage applied to the microcontroller may be subject to variation or not precisely known. The embedded internal voltage reference (V_{REFINT}) and its calibration data acquired by the ADC during the manufacturing process at V_{REF+} = 3.0 V can be used to evaluate the actual V_{REF+} voltage level, if VREF+ pin is connected to a variable V_{DDA} power supply.

The following formula gives the actual V_{REF+} voltage supplying the device:

$$V_{REF+} = 3.0\ V \times VREFINT\_CAL / VREFINT\_DATA$$

Where:

- VREFINT_CAL is the VREFINT calibration value (refer to *Section 29.3: ADC implementation* for the value of VREFINT_CAL).
- VREFINT_DATA is the actual VREFINT output value converted by ADC.

Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between the voltage reference V_{REF+} and the voltage applied on the converted channel. For most application use cases, it is necessary to convert this ratio into a voltage independent of V_{DDA}. For applications where V_{DDA} is known and ADC converted values are right-aligned, the following formula can be used to calculate this absolute value:

$$V_{CHANNELx} = \frac{V_{REF+}}{FULL\_SCALE} \times ADC\_DATA$$

For applications where V_{DDA} value is not known, the internal voltage reference and V_{DDA} can be replaced by the expression provided in *Section : Calculating the actual V_{REF+} voltage using the internal reference voltage*, resulting in the following formula:

$$V_{CHANNELx} = \frac{3.0\ V \times VREFINT\_CAL \times ADC\_DATA}{VREFINT\_DATA \times FULL\_SCALE}$$

Where:

VREFINT_CAL is the VREFINT calibration value (refer to *Section 29.3: ADC implementation* for the value of VREFINT_CAL).

ADC_DATA is the value measured by the ADC on channel x (right-aligned)

VREFINT_DATA is the actual VREFINT output value converted by the ADC

FULL_SCALE is the maximum digital value of the ADC output. For example with 14-bit resolution, it is 2^{14} - 1 = 16383 or with 8-bit resolution, 2^8 - 1 = 255.

Note: If ADC measurements are done using an output format other than 14-bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.

29.5 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 (OVR flag)

Separate interrupt enable bits are available for flexibility.

Table 245. ADC interrupts

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop 0, Stop 1 and Stop 2 modes	Exit from Stop 3, Standby modes
ADC	ADC ready	ADRDY	ADRDYIE	Set by hardware and cleared by software	Yes	No	No
	End of conversion of a regular group	EOC	EOCIE				
	End of conversion sequence of a regular group	EOS	EOSIE				
	End of conversion of an injected group	JEOC	JEOCIE				
	End of conversion sequence 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				

29.6 ADC registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

29.6.1 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.	LDO RDY	Res.	Res.	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
			r			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LDORDY**: ADC voltage regulator ready

This bit is set by hardware. It indicates that the ADC internal supply is ready. The ADC is available after $t_{ADCVREG\_SETUP}$ time.
0: ADC voltage regulator disabled
1: ADC voltage regulator enabled

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **AWD3**: Analog watchdog 3 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT3[7:0] and HT3[7:0] of ADC_LTR3 & ADC_HTR3 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_LTR2 & ADC_HTR2 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_LTR1, & ADC_HTR1 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

29.6.2 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.	Res.	AWD3 IE	AWD2 IE	AWD1 IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMP IE	ADRDY IE
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 3 interrupt disabled

1: Analog watchdog 3 interrupt enabled

Note: 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: 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: 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: 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: Software is allowed to write this bit only when JADSTART = 0 (which ensures that no regular 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: 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: 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: 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: 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: Software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

29.6.3 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	Res.	DEEPPWD	ADVREGEN	CALINDEX[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADCALLIN
rs		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.	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 ADC calibration.
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.

Bit 30 Reserved, must be kept at reset value.

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 29.4.6: 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 27:24 **CALINDEX[3:0]**: Calibration factor

This bitfield controls the calibration factor to be read or written.
Calibration index 0 is dedicated to single-ended and differential offsets, calibration index 1 to 7 to the linearity calibration factors, and index 8 to the internal offset:
0000: Offset calibration factor
0001: linearity calibration factor 1
0010: linearity calibration factor 2
0011: linearity calibration factor 3
0100: linearity calibration factor 4
0101: linearity calibration factor 5
0110: linearity calibration factor 6
0111: linearity calibration factor 7 and internal offset (write access only)
1000: internal offset (read access only)
1001: Calibration mode selection
Others: Reserved, must not be used

Note: ADC_CALFACT2[31:0] correspond to the location of CALINDEX[3:0] calibration factor data (see Section 29.4.8: Calibration (ADCAL, ADCALLIN, ADC_CALFACT) for details).

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **ADCALLIN**: Linearity calibration

This bit is set and cleared by software to enable the linearity calibration.
0: Writing ADCAL launches a calibration without the linearity calibration.
1: Writing ADCAL launches a calibration with he linearity calibration.

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).

Bits 15:6 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 starts 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 as 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 starts immediately (software trigger configuration) or once a regular hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

– in Single conversion mode (CONT = 0, DISCEN = 0) when software trigger is selected (EXTEN[1:0] = 0x0): at the assertion of the end of regular conversion sequence (EOS) flag.

– In Discontinuous conversion mode (CONT = 0, DISCEN = 1), when the software trigger is selected (EXTEN[1:0] = 0x0): at the end of conversion (EOC) flag.

– in all other 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 is 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)

29.6.4 ADC configuration register (ADC_CFGR1)

Address offset: 0x0C

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	Res.	JDISCEN	DISCNUM[2:0]			DISCEN
	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.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL[4:0]					Res.	RES[1:0]		DMNGT[1:0]	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:26 **AWD1CH[4:0]**: Analog watchdog 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

00001: ADC analog input channel-1 monitored by AWD1

.....

10011: ADC analog input channel-19 monitored by AWD1

Others: Reserved, must not be used

Note: The channel selected by AWD1CH must be also selected into the SQRi or JSQRi registers.

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).

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 Reserved, must be kept at reset value.

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.

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).

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).

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).

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).

Bit 12 **OVRMOD**: Overrun Mode

This bit is set and cleared by software and configure the way data overrun is managed.

0: ADC_DR register is preserved with the old data when an overrun is detected.

1: ADC_DR register is overwritten with the last conversion result when an overrun is detected.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection for regular channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.

00: Hardware trigger detection disabled (conversions can be launched by software)
01: Hardware trigger detection on the rising edge
10: Hardware trigger detection on the falling edge
11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 9:5 **EXTSEL[4:0]**: External trigger selection for regular group

These bits select the external event used to trigger the start of conversion of a regular group:
00000: adc_ext_trg0
00001: adc_ext_trg1
...
Refer to the ADC external trigger for regular channels in *Section 29.4.2: ADC pins and internal signals* for details on trigger mapping.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **RES[1:0]**: Data resolution

These bits are written by software to select the resolution of the conversion.
00: 14 bits
01: 12 bits
10: 10 bits
11: 8bits

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 1:0 **DMNGT[1:0]**: Data management configuration

This bit is set and cleared by software to select how the ADC interface output data are managed.
00: Regular conversion data stored in DR only
01: DMA One -shot mode selected
10: MDF mode selected
11: DMA Circular mode selected

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

29.6.5 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
LSHIFT[3:0]				LFTRIG	Res.	OSR[9:0]									
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMPTRIG	SWTRIG	BULB	Res.	Res.	ROVSM	TROVS	OVSS[3:0]				Res.	Res.	Res.	JOVSE	ROVSE
rw	rw	rw			rw	rw	rw	rw	rw	rw				rw	rw

Bits 31:28 **LSHIFT[3:0]**: Left shift factor

This bitfield is set and cleared by software to define the left shifting applied to the final result with or without oversampling.

0000: No left shift
0001: 1-bit left shift
0010: 2-bit left shift
0011: 3-bit left shift
0100: 4-bit left shift
0101: 5-bit left shift
0110: 6-bit left shift
0111: 7-bit left shift
1000: 8-bit left shift
1001: 9-bit left shift
1010: 10-bit left shift
1011: 11-bit left shift
1100: 12-bit left shift
1101: 13-bit left shift
1110: 14-bit left shift
1111: 15-bit left shift

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 27 **LFTRIG**: Low-frequency trigger

This bit is set and cleared by software
0: Low-frequency trigger mode disabled
1: Low-frequency trigger mode enabled

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 26 Reserved, must be kept at reset value.

Bits 25:16 **OSR[9:0]**: Oversampling ratio

This bitfield is set and cleared by software to define the oversampling ratio.
0: 1x (no oversampling)
1: 2x
2: 3x
...
1023: 1024x

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 15 **SMPTRIG**: Sampling time control trigger mode

This bit is set and cleared by software to enable the sampling time control trigger mode.
0: Sampling time control trigger mode disabled
1: Sampling time control trigger mode enabled
The sampling time starts on the trigger rising edge, and the conversion on the trigger falling edge.
EXTEN[1:0] bits must be set to 01. BULB bit must not be set when the SMPTRIG bit is set.
When EXTEN[1:0] bits is set to 00, set SWTRIG to start the sampling and clear SWTRIG bit to start the conversion.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 14 **SWTRIG**: Software trigger bit for sampling time control trigger mode

This bit is set and cleared by software to enable the bulb sampling mode.

0: Software trigger starts the conversion for sampling time control trigger mode

1: Software trigger starts the sampling for sampling time control trigger mode.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 13 **BULB**: Bulb sampling mode

This bit is set and cleared by software to select the bulb sampling mode.

0: Bulb sampling mode disabled

1: Bulb sampling mode enabled. The sampling period starts just after the previous end of the conversion.

SMPTRIG bit must not be set when the BULB bit is set.

The very first ADC conversion is performed with the sampling time specified in SMPx bits.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 12:11 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 right shift

This bit field is set and cleared by software to define the right shifting applied to the raw oversampling result.

0000: No right shift

0001: 1-bit right shift

0010: 2-bit right shift

0011: 3-bit right shift

0100: 4-bit right shift

0101: 5-bit right shift

0110: 6-bit right shift

0111: 7-bit right shift

1000: 8-bit right shift

1001: 9-bit right shift

1010: 10-bit right shift

1011: 11-bit right shift

Others: Reserved, must not be used.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 4:2 Reserved, must be kept at reset value.

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)

29.6.6 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 **SMPx[2:0]:** Channel x sampling time selection (x = 9 to 0)

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: 5 ADC clock cycles

001: 6 ADC clock cycles

010: 12 ADC clock cycles

011: 20 ADC clock cycles

100: 36 ADC clock cycles

101: 68 ADC clock cycles

110: 391 ADC clock cycles

111: 814 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).

29.6.7 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.	SMP19[2:0]			SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[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
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 31:30 Reserved, must be kept at reset value.

Bits 29:0 **SMPx[2:0]:** Channel x sampling time selection (x = 19 to 10)

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: 5 ADC clock cycles
001: 6 ADC clock cycles
010: 12 ADC clock cycles
011: 20 ADC clock cycles
100: 36 ADC clock cycles
101: 68 ADC clock cycles
110: 391 ADC clock cycles
111: 814 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).

29.6.8 ADC channel preselection register (ADC_PCSEL)

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.	PCSEL19	PCSEL18	PCSEL17	PCSEL16
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCSEL15	PCSEL14	PCSEL13	PCSEL12	PCSEL11	PCSEL10	PCSEL9	PCSEL8	PCSEL7	PCSEL6	PCSEL5	PCSEL4	PCSEL3	PCSEL2	PCSEL1	PCSEL0
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 **PCSEL[19:0]**: Channel i (V_{INP}[i]) preselection

These bits are written by software to preselect the input channel I/O instance to be converted.

0: Input channel i (V_{INP}[i]) is not preselected for conversion, the ADC conversion of this channel shows a wrong result.

1: Input channel i (V_{INP}[i]) is preselected for conversion

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

29.6.9 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..19) assigned as the 4th in the regular conversion sequence.

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..19) 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..19) 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..19) 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).

29.6.10 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..19) 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..19) 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..19) 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..19) 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..19) 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).

29.6.11 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..19) 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..19) 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..19) 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..19) 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..19) 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).

29.6.12 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..19) 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..19) 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).

29.6.13 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
RDATA[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
RDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDATA[31:0]**: 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 29.4.27: Data management*.

29.6.14 ADC injected sequence register (ADC_JSQR)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:1]			
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ2[0]	Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL[4:0]					JL[1:0]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 **JSQ4[4:0]:** 4th conversion in the injected sequence

These bits are written by software with the channel number (0..19) 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 26 Reserved, must be kept at reset value.

Bits 25:21 **JSQ3[4:0]:** 3rd conversion in the injected sequence

These bits are written by software with the channel number (0..19) assigned as the 3rd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing.

Bit 20 Reserved, must be kept at reset value.

Bits 19:15 **JSQ2[4:0]:** 2nd conversion in the injected sequence

These bits are written by software with the channel number (0..19) assigned as the 2nd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing.

Bit 14 Reserved, must be kept at reset value.

Bits 13:9 **JSQ1[4:0]:** 1st conversion in the injected sequence

These bits are written by software with the channel number (0..19) assigned as the 1st in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing.

Bits 8:7 **JEXTEN[1:0]**: External trigger enable and polarity selection for injected channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.

00: 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.

Bits 6:2 **JEXTSEL[4:0]**: External trigger selection for injected group

These bits select the external event used to trigger the start of conversion of an injected group:

00000: adc_jext_trg0

00001: adc_jext_trg1

...

Refer to the ADC external trigger for injected channels in *Section 29.4.2: ADC pins and internal signals* for details on trigger mapping.

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.

29.6.15 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
OFFSET_CH[4:0]					SSAT	USAT	POSOFF	OFFSET[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
OFFSET[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 **OFFSET_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 OFFSETy[25:0] bits applies.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

If OFFSETy_EN bit is set, it is not allowed to select the same channel in different ADC_OFRy registers.

Bit 26 **SSAT:** Signed saturation enable

This bit is written by software to enable or disable the Signed saturation feature.

(see *Section : Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, USAT, SSAT)* for details).

0: Offset is subtracted maintaining data integrity and extending converted data size (9-bit and 15-bit signed format).

1: Offset is subtracted and result is saturated to maintain converted data size.

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 25 **USAT**: Unsigned saturation enable

This bit is written by software to enable or disable the unsigned saturation feature.

0: Offset is subtracted maintaining data integrity and keeping converted data size

1: Offset is subtracted and result is saturated to maintain converted data size.

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 24 **POSOFF**: offset sign

This bit is set and cleared by software to enable the positive offset.

0: Negative offset

1: Positive offset

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 23:0 **OFFSET[23:0]:** Data offset y for the channel programmed into OFFSETy_CH[4:0] bits

These bits are written by software to define the offset y to be subtracted from the raw converted data when converting a channel (regular or injected). The channel to which the data offset y applies must be programmed to the OFFSETy_CH[4:0] bits. The conversion result can be read from in the ADC_DR (regular conversion) or from in the ADC_JDRyi registers (injected conversion).

When OFFSETy[21:0] bitfield is reset, the offset compensation is disabled.

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 offsets (OFFSETy) point to the same channel, only the offset with the lowest y value is considered for the subtraction.

For example, if OFFSET1_CH[4:0] = 4 and OFFSET2_CH[4:0] = 4, this is OFFSET1[25:0] that is subtracted when converting channel 4.

29.6.16 ADC gain compensation register (ADC_GCOMP)

Address offset: 0x70

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GCOMP	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.	GCOMPCOEFF[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **GCOMP**: Gain compensation mode

This bit is set and cleared by software to enable the gain compensation mode.

0: Regular ADC operating mode

1: Gain compensation enabled and applied on all channels

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 30:14 Reserved, must be kept at reset value.

Bits 13:0 **GCOMPCOEFF[13:0]**: Gain compensation coefficient

These bits are set and cleared by software to program the gain compensation coefficient.
00 1000 0000 0000: gain factor of 0.5
...
01 0000 0000 0000: gain factor of 1
10 0000 0000 0000: gain factor of 2
11 0000 0000 0000: gain factor of 3
...
The coefficient is divided by 4096 to get the gain factor ranging from 0 to 3.999756.

Note: This gain compensation is only applied when GCOMP bit of ADCx_CFGR2 register is 1.

29.6.17 ADC injected 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
JDATA[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
JDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **JDATA[31:0]:** 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 29.4.27: Data management*.

29.6.18 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.	AWD2CH[19:16]			
												rw	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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **AWD2CH[19:0]**: 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[19:0] = 000..0, the analog Watchdog 2 is disabled

Note: The channels selected by AWD2CH must be also selected into the SQRi or JSQRi registers. Software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

29.6.19 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.	AWD3CH[19:16]			
												rw	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 31:20 Reserved, must be kept at reset value.

Bits 19:0 **AWD3CH[19:0]**: 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[19:0] = 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).

29.6.20 ADC watchdog threshold register 1 (ADC_LTR1)

Address offset: 0xA8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR1[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
LTR1[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 **LTR1[24:0]**: Analog watchdog 1 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 1.

Refer to *Section 29.4.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)*.

29.6.21 ADC watchdog threshold register 1 (ADC_HTR1)

Address offset: 0xAC

Reset value: 0x01FF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AWDFILT1[2:0]			Res.	Res.	Res.	Res.	HTR1[24:16]								
rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **AWDFILT1[2:0]**: Analog watchdog filtering parameter

This bit is set and cleared by software.

000: No filtering

001: two consecutive detection generates an AWDx flag or an interrupt

...

111: Eight consecutive detection generates an AWDx flag or an interrupt

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 28:25 Reserved, must be kept at reset value.

Bits 24:0 **HTR1[24:0]**: Analog watchdog 1 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 1.

Refer to *Section 29.4.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)*.

29.6.22 ADC watchdog lower threshold register 2 (ADC_LTR2)

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.	LTR2[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
LTR2[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 **LTR2[24:0]**: Analog watchdog 2 lower threshold
These bits are written by software to define the lower threshold for the analog watchdog 2.
Refer to *Section 29.4.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)*.

29.6.23 ADC watchdog higher threshold register 2 (ADC_HTR2)

Address offset: 0xB4

Reset value: 0x01FF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	HTR2[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
HTR2[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 **HTR2[24:0]**: Analog watchdog 2 higher threshold
These bits are written by software to define the higher threshold for the analog watchdog 2.
Refer to *Section 29.4.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)*.

29.6.24 ADC watchdog lower threshold register 3 (ADC_LTR3)

Address offset: 0xB8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR3[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
LTR3[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 **LTR3[24:0]**: Analog watchdog 3 lower threshold
These bits are written by software to define the lower threshold for the analog watchdog 3.
Refer to *Section 29.4.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)*.

29.6.25 ADC watchdog higher threshold register 3 (ADC_HTR3)

Address offset: 0xBC

Reset value: 0x01FF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	HTR3[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
HTR3[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 **HTR3[24:0]**: Analog watchdog 3 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 3.
Refer to *Section 29.4.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)*.

29.6.26 ADC differential mode selection register (ADC_DIFSEL)

Address offset: 0xC0

Reset value: 0x0000 0000

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

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

Bits 19:0 **DIFSEL[19:0]**: Differential mode for channels 19 to 0

These bits are set and cleared by software. They allow selecting if a channel is configured as single ended or differential mode.
DIFSEL[i] = 0: ADC analog input channel-i is configured in single ended mode
DIFSEL[i] = 1: ADC analog input channel-i is configured in differential mode

Note: 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).

29.6.27 ADC user control register (ADC_CALFACT)

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.	CAPTURE_COEF	LATCH_COEF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VALIDITY
						rw	rw								r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I_APB_DATA[7:0]								I_APB_ADDR[7:0]							
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 **CAPTURE_COEF**: Calibration factor capture enable bit
This bit enables the internal calibration factor capture.
0: Calibration factor not captured
1: Calibration factor available in CALFACT[31:0] bits, the calibration factor index being defined by CALINDEX[3:0] bits

Bit 24 **LATCH_COEF**: Calibration factor latch enable bit
This bit latches the calibration factor in the CALFACT[31:0] bits.
0: No effect
1: Calibration factor latched in the analog block on LATCH_COEF bit transition from 0 to 1. Prior to latching the calibration factor, CALFACT[31:0] bits must be programmed with the content of CALINDEX[3:0] bits.

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **VALIDITY**: Delayed write access status bit
This bit indicates the communication status between the ADC digital and analog blocks.
0: Operation still in progress
1: Operation complete

Bits 15:8 **I_APB_DATA[7:0]**: Delayed write access data
This bitfield contains the data that are being written during delayed write accesses.

Bits 7:0 **I_APB_ADDR[7:0]**: Delayed write access address
This bitfield contains the address that is being written during delayed write accesses.

29.6.28 ADC calibration factor register (ADC_CALFACT2)

Address offset: 0xC8

Reset value: 0x0000 0000

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

Bits 31:0 **CALFACT[31:0]**: Linearity or offset calibration factor

These bits can be written either by hardware or by software.

They contain the 32-bit offset or linearity calibration factor.

When CAPTURE_COEF is set to 1, the calibration factor of the analog block is read back and stored in CALFACT[31:0], indexed by CALINDEX[3:0] bits.

When LATCH_COEF is set to 1, the calibration factor of the analog block is updated with the value programmed in CALFACT[31:0], indexed by CALINDEX[3:0] bits.

To read all calibration factors, perform nine accesses to the ADC_CALFACT2 register.

To write all calibration factors, perform eight accesses to the ADC_CALFACT2 register.

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).

29.7 ADC common registers

These registers define the control and status registers common to master and slave ADCs.

29.7.1 ADC system control register (ADC12_CCR)

Address offset: 0x08

Reset value: 0x0000 0000

The address offset is relative to the master ADC base address + 0x300.

ADC12_CCR is common to ADC1 and ADC2. ADC2 is not available on all devices (refer to *Section 29.3: ADC implementation*).

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

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **VBATEN**: VBAT enable

This bit is set and cleared by software to control the V_{BAT} channel.

0: V_{BAT} channel disabled

1: V_{BAT} channel enabled

Note: The software is allowed to write this bit only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bit 23 **VSENSESEL**: Temperature sensor voltage selection

This bit is set and cleared by software to control the temperature sensor channel.

0: Temperature sensor channel disabled

1: Temperature sensor channel enabled

Note: The software is allowed to write this bit only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bit 22 **VREFEN**: V_{REFINT} enable

This bit is set and cleared by software to enable/disable the V_{REFINT} buffer.

0: V_{REFINT} channel disabled

1: V_{REFINT} channel enabled

Note: The software is allowed to write this bit only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 21:18 **PRESC[3:0]**: ADC prescaler

These bits are set and cleared by software to select the frequency of the ADC clock. The clock is common to all 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

Others: Reserved, must not be used

Note: The software is allowed to write this bit only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 17:0 Reserved, must be kept at reset value.

29.8 ADC register map

Table 246. ADC 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	**ADC_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LDORDY	Res.	Res.	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.	Res.	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x08	**ADC_CR**	ADCAL	Res.	DEEPPWD	ADVREGEN	CALINDEX[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADCALIN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
	Reset value	0		0	0	0	0	0	0								0											0	0	0	0	0	0
0x0C	**ADC_CFGR1**	Res.	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	Res.	JDISCEN	DISCNUM [2:0]			DISCEN	Res.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL[4:0]					Res.	RES [1:0]		DMN GT [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

Table 246. ADC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x10	**ADC_CFGR2**	LSHIFT[3:0]				LSTRIG	Res.	OSR[9:0]										SMPTRIG	SWTRIG	BULB	Res.	Res.	ROVSM	TROVS	OVSS[3:0]				Res.	Res.	Res.	JOVSE	ROVSE
	Reset value	0	0	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	**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.	SMP19 [2:0]			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	0	0	0
0x1C	**ADC_PCSEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCSEL19	PCSEL18	PCSEL17	PCSEL16	PCSEL15	PCSEL14	PCSEL13	PCSEL12	PCSEL11	PCSEL10	PCSEL9	PCSEL8	PCSEL7	PCSEL6	PCSEL5	PCSEL4	PCSEL3	PCSEL2	PCSEL1	PCSEL0
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20-0x28	Reserved	Res.																															
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**	RDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44-0x48	Reserved	Res.																															
0x4C	**ADC_JSQR**	JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:0]				Res.	JSQ1[4:0]						JEXTEN[1:0]		JEXTSEL[4:0]					JL[1:0]	
	Reset value	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50-0x5C	Reserved	Res.																															
0x60	**ADC_OFR1**	OFFSET1_CH[4:0]					SSAT	USAT	POSOFF	OFFSET[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
0x64	**ADC_OFR2**	OFFSET2_CH[4:0]					SSAT	USAT	POSOFF	OFFSET[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

Table 246. ADC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x68	**ADC_OFR3**			OFFSET3_CH[4:0]			SSAT	USAT	POSOFF	OFFSET[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
0x6C	**ADC_OFR4**			OFFSET4_CH[4:0]			SSAT	USAT	POSOFF	OFFSET[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
0x70	**ADC_GCOMPR**	GCOMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GCOEFF[13:0]													
	Reset value	0																		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74-0x7C	Reserved	Res.																															
0x80	**ADC_JDR1**	JDATA1[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
0x84	**ADC_JDR2**	JDATA2[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	**ADC_JDR3**	JDATA3[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	**ADC_JDR4**	JDATA4[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
0x90-0x9C	Reserved	Res.																															
0xA0	**ADC_AWD2CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD2CH[19:0]																			
	Reset value													0	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.	AWD3CH[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA8	**ADC_LTR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR1[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
0xAC	**ADC_HTR1**	AWDFILT1[2:0]			Res.	Res.	Res.	Res.	HTR1[24:0]																								
	Reset value	0	0	0					1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0xB0	**ADC_LTR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR2[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
0xB4	**ADC_HTR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HTR2[24:0]																								
	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
0xB8	**ADC_LTR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR3[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
0xBC	**ADC_HTR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HTR3[24:0]																								
	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
0xC0	**ADC_DIFSEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIFSEL[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 246. ADC 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
0xC4	**ADC_CALFACT**	Res.	Res.	Res.	Res.	Res.	Res.	CAPTURE_COEF	LATCH_COEF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VALIDITY	I_APB_DATA[7:0]								I_APB_ADDR[7:0]								
	Reset value							0	0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC8	**ADC_ CALFACT2**	CALFACT[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 247. ADC register map and reset values (master and slave ADC common registers) offset = 0x300

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
0x00-0x04	Reserved	Res.																															
0x08	**ADC12_CCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBATEN	VSENSESEL	VREFEN	PRESC[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0	0	0	0	0	0	0																		

Refer to *Section 2.3 on page 115* for the register boundary addresses.

30 Analog-to-digital converter (ADC4)

30.1 Introduction

The 12-bit ADC is a successive approximation analog-to-digital converter. It has up to 25 multiplexed channels enabling it to measure signals from 19 external and 6 internal sources. A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned 16-bit data register.

The analog watchdog feature enables the application to detect if the input voltage goes outside the user-defined higher or lower thresholds.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

A built-in hardware oversampler allows improving analog performances while off-loading the related computational burden from the CPU.

30.2 ADC main features

- High performance
  - 12-bit, 10-bit, 8-bit or 6-bit configurable resolution
  - ADC conversion time: 0.4 µs for 12-bit resolution (2.5 Msps), faster conversion times can be obtained by lowering resolution.
  - Self-calibration
  - Programmable sampling time
  - Data alignment with built-in data coherency
  - DMA support
- Low-power
  - The application can reduce the bus clock frequency for low-power operation while still keeping optimum ADC performance. For example, 0.4 µs conversion time is kept, whatever the bus clock frequency
  - Wait mode: prevents ADC overrun in applications with low bus clock frequency
  - Auto off mode: ADC is automatically powered off except during the active conversion phase. This dramatically reduces the power consumption of the ADC.
- Autonomous mode
  - Conversion and DMA transfers supported in Stop mode
  - Wakeup from Stop on ADC interrupts
  - Enter and exit from Deep-power-down mode managed automatically
- Analog input channels
  - 19 external analog inputs
  - 1 channel for the internal temperature sensor (V_{SENSE})
  - 1 channel for the internal reference voltage (V_{REFINT})
  - 1 channel for the internal digital core voltage (V_{CORE})
  - 1 channel for monitoring the external VBAT power supply pin

  - Connection to DAC internal channels
- Start-of-conversion can be initiated:
  - By software
  - By hardware triggers with configurable polarity (timer events or GPIO input events)
- Conversion modes
  - Can convert a single channel or can scan a sequence of channels.
  - Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode
- Interrupt generation at the end of sampling, end of conversion, end of sequence conversion, and in case of analog watchdog or overrun events
- Analog watchdog
- Oversampler
  - 16-bit data register
  - Oversampling ratio adjustable from 2 to 256x
  - Programmable data shift up to 8 bits
- ADC input range: $V_{SSA} \leq V_{IN} \leq V_{REF+}$

30.3 ADC implementation

Table 248. ADC features(1)

ADC modes/features	ADC1	ADC4
Resolution	14 bits	12 bits
Maximum sampling speed for 14-bit resolution	2.5 Msps	2.5 Msps
Hardware offset calibration	X	X
Hardware linearity calibration	X	-
Single-ended inputs	X	X
Differential inputs	X	-
Injected channel conversion	X	-
Oversampling	up to x1024	up to x256
Data register	32 bits	16 bits
DMA support	X	X
parallel data output to MDF	X	-
Dual mode	-	-
Autonomous mode	-	X
Offset compensation	X	-
Gain compensation	X	-

Table 248. ADC features(1) (continued)

ADC modes/features	ADC1	ADC4
Number of analog watchdogs	3	3
Wakeup from Stop mode	-	X(2)

1. Note: 'X' = supported, '-'= not supported.
2. Wake up supported from Stop 0, Stop 1 and Stop 2 modes.

Table 249. Memory location of the temperature sensor calibration values

Name	Description	Memory address
TS_CAL1	Temperature sensor 14-bit raw data acquired by ADC1 at 30 °C (± 5 °C), $V_{DDA} = V_{REF+}$ = 3.0 V (±10 mV)	0x0BFA 0710 - 0x0BFA 0711
TS_CAL2	Temperature sensor 14-bit raw data acquired by ADC1 at 130 °C (± 5 °C), $V_{DDA} = V_{REF+}$ = 3.0 V (±10 mV)	0x0BFA 0742 - 0x0BFA 0743

Table 250. Memory location of the internal reference voltage sensor calibration value

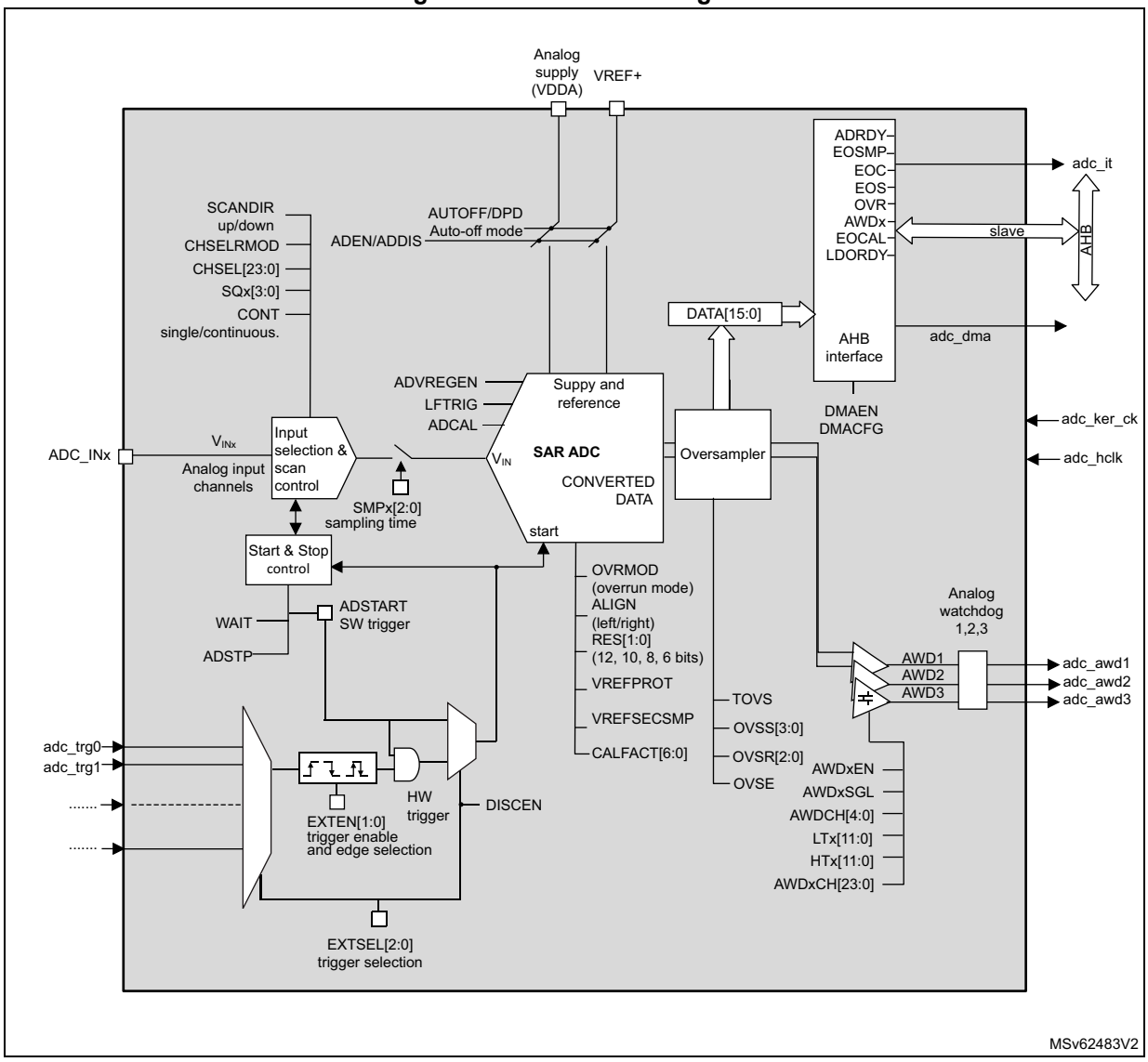
Name	Description	Memory address
VREFINT_CAL	14-bit raw data acquired by ADC1 at 30 °C (± 5 °C), $V_{DDA} = V_{REF+}$ = 3.0 V (±10 mV)	0x0BFA 07A5 - 0x0BFA 07A6

30.4 ADC functional description

30.4.1 ADC block diagram

Figure 187 shows the ADC block diagram and *Table 251* gives the ADC pin description.

Figure 187. ADC block diagram



30.4.2 ADC pins and internal signals

Table 251. ADC input/output pins

Pin name	Signal type	Description
VDDA	Input, analog power supply	Analog power supply and positive reference voltage for the ADC, $V_{DDA} \geq V_{DD}$
VSSA	Input, analog supply ground	Ground for analog power supply, equal to V_{SS}.
VREF+	Input, reference positive	The higher/positive reference voltage for the ADC.
ADC_INx	Analog input signals	19 external analog input channels.

Table 252. ADC internal input/output signals

Internal signal name	Signal type	Description
V_{IN}[x]	Analog inputs	Analog input channels connected either to internal channels or to ADC_INx external channels.
adc_trgx	Inputs	ADC conversion triggers.
adc_awdx	Output	Internal analog watchdog output signal connected to on-chip timers (x = Analog watchdog number = 1,2,3).
adc_it	Output	ADC interrupt.
adc_hclk	Input	AHB clock.
adc_ker_ck	Input	ADC kernel clock input from the RCC block.
adc_dma	Output	ADC DMA request

Table 253. ADC interconnection

Signal name	Source/destination
ADC4 V_{IN}[13]	V_{SENSE} (internal temperature sensor output voltage)
ADC4 V_{IN}[0]	V_{REFINT} (buffered voltage from internal reference voltage)
ADC4 V_{IN}[14]	$V_{BAT/4}$ (VBAT pin input voltage divided by 4)
ADC4 V_{IN}[12]	V_{CORE} (internal logic supply voltage).
ADC4 V_{IN}[21]	dac1_out1
	dac1_out2
adc_trg0	tim1_trgo2
adc_trg1	tim1_oc4
adc_trg2	tim2_trgo
adc_trg3	tim15_trgo
adc_trg4	tim6_trgo
adc_trg5	lptim1_ch1
adc_trg6	lptim3_ch2
adc_trg7	exti15

30.4.3 ADC voltage regulator (ADVREGEN)

The ADC has a specific internal voltage regulator which must be enabled and stable before using the ADC.

The ADC internal voltage regulator can be enabled by setting ADVREGEN bit to 1 in the ADC_CR register. The software must wait for the ADC voltage regulator startup time ($t_{ADCVREG\_SETUP}$) before launching a calibration or enabling the ADC. The LDO status can be verified by checking the LDORDY bit in ADC_ISR register.

After ADC operations are complete, the ADC can be disabled (ADEN = 0). It is then possible to save additional power by disabling the ADC voltage regulator (refer to *Section : ADC voltage regulator disable sequence*).

Note: When the internal voltage regulator is disabled, the internal analog calibration factor is reset, and a new calibration must be performed.

ADC voltage regulator enable sequence

To enable the ADC voltage regulator, follow the sequence below:

1. Clear the LDORDY bit in ADC_ISR register by programming this bit to 1.
2. Set the ADVREGEN bit to 1 in ADC_CR register.
3. Wait until LDORDY = 1 in the ADC_ISR register (LDORDY is set after the ADC voltage regulator startup time). This can be handled by interrupt if the interrupt is enabled by setting the LDORDYIE bit in the ADC_IER register.

ADC voltage regulator disable sequence

To disable the ADC voltage regulator, follow the sequence below:

1. Make sure that the ADC is disabled (ADEN = 0).
2. Clear ADVREGEN bit in ADC_CR register.
3. Clear the LDORDY bit in ADC_ISR register by programming this bit to 1(optional),

30.4.4 Calibration (ADCAL)

The ADC has a calibration feature. During the procedure, the ADC calculates a calibration factor which is internally applied to the ADC until the next ADC power-off. The application must not use the ADC during calibration and must wait until it is complete.

The calibration must be performed before starting analog-to-digital conversion. It removes the offset error which may vary from chip to chip due to process variation, supply voltage and temperature.

The calibration is initiated by software by setting bit ADCAL to 1. It can be initiated only when all the following conditions are met:

- the ADC voltage regulator is enabled (ADVREGEN = 1 and LDORDY = 1),
- the ADC is disabled (ADEN = 0), and
- the Auto-off mode is disabled (AUTOFF = 0).

ADCAL bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon the calibration completes. After this, the calibration factor can be read from the ADC_DR register (from bits 6 to 0).

The internal analog calibration is kept if the ADC is disabled (ADEN = 0). When the ADC operating conditions change (V_{DDA} changes are the main contributor to ADC offset variations and temperature change to a lesser extend), it is recommended to re-run a calibration cycle. It is recommended to recalibrate when V_{REF+} voltage changed more than 10%.

The calibration factor is lost in the following cases:

- The power supply is removed from the ADC (for example when the product enters Standby or V_{BAT} mode).
- The ADC peripheral is reset.

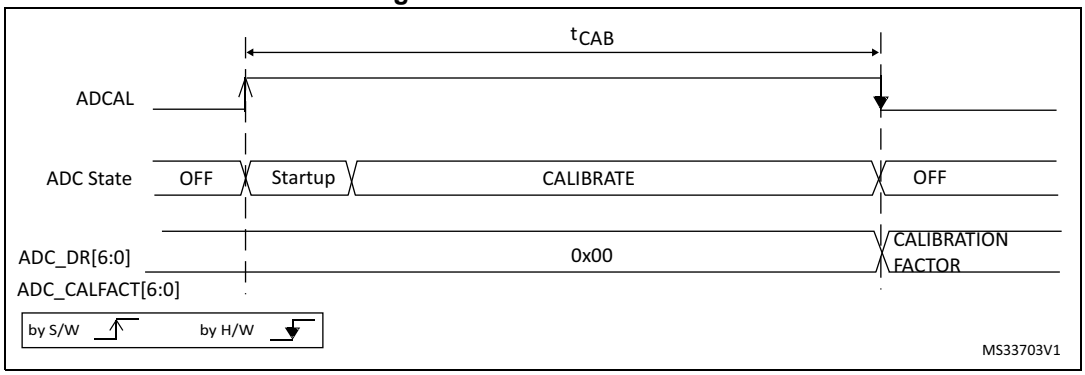
The calibration factor is lost each time power is removed from the ADC (for example when the product enters Standby or V_{BAT} mode). Still, it is possible to save and restore the calibration factor by software to save time when re-starting the ADC (as long as temperature and voltage are stable during the ADC power-down).

The calibration factor can be written if the ADC is enabled but not converting (ADEN = 1 and ADSTART = 0). Then, at the next start of conversion, the calibration factor is automatically injected into the analog ADC. This loading is transparent and does not add any cycle latency to the start of the conversion.

Software calibration procedure

1. Ensure that ADEN = 0, ADVREGEN = 1, AUTOFF = 0, DPD = 0, and DMAEN = 0.
2. Set ADCAL = 1.
3. Wait until ADCAL = 0 (or until EOCAL = 1). This can be handled by interrupt if the interrupt is enabled by setting the EOCALIE bit in the ADC_IER register
4. The calibration factor can be read from bits 6:0 of ADC_DR or ADC_CALFACT registers.

Figure 188. ADC calibration

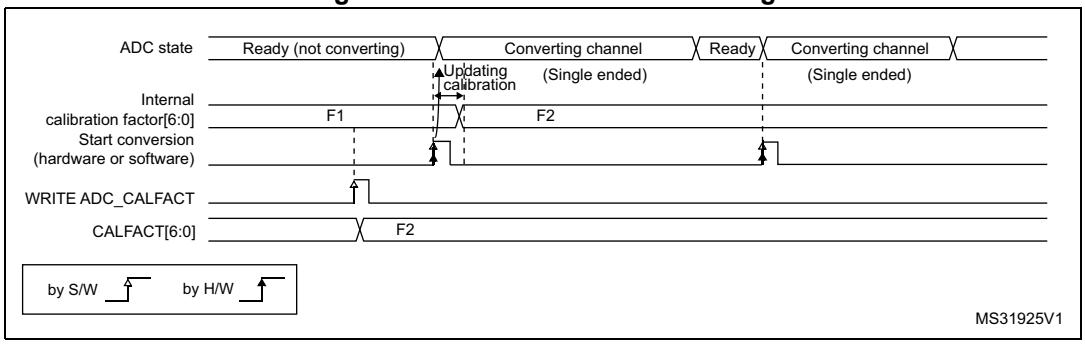


1. Refer to the device datasheet for the value of t_{CAB}.

Calibration factor forcing software procedure

1. Ensure that ADEN = 1 and ADSTART = 0 (ADC started with no conversion ongoing).
2. Write ADC_CALFACT with the saved calibration factor.
3. The calibration factor is used as soon as a new conversion is launched.

Figure 189. Calibration factor forcing



30.4.5 ADC on-off control (ADEN, ADDIS, ADRDY)

At power-up, the ADC is disabled and put in power-down mode (ADEN = 0).

As shown in *Figure 190*, the ADC needs a stabilization time of t_{STAB} before it starts converting accurately.

Two control bits are used to enable or disable the ADC:

- Set ADEN = 1 to enable the ADC. The ADRDY flag is set as soon as the ADC is ready for operation.
- Set ADDIS = 1 to disable the ADC and put the ADC in Power-down. The ADEN and ADDIS bits are then automatically cleared by hardware as soon as the ADC is fully disabled.

Conversion can then start either by setting ADSTART to 1 (refer to *Section 30.4.16: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN) on page 1099*) or when an external trigger event occurs if triggers are enabled.

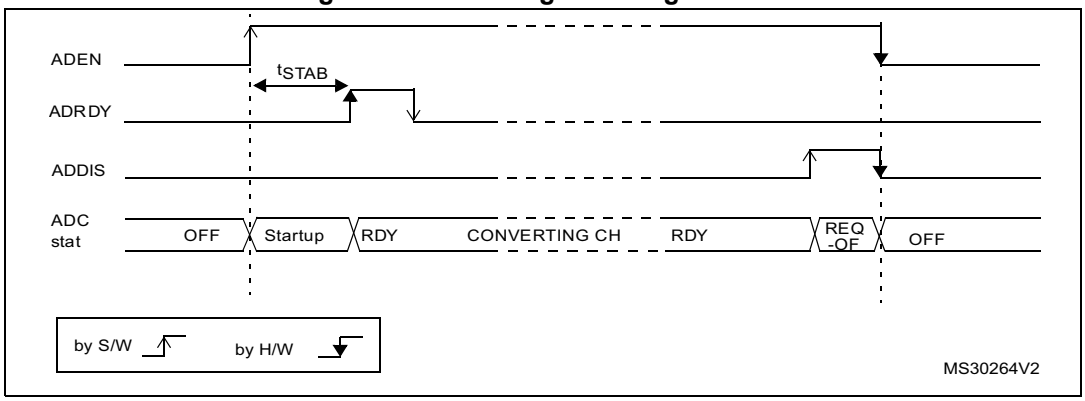
Follow the procedure below to enable the ADC:

1. Clear the ADRDY bit in ADC_ISR register by programming this bit to 1.
2. Set ADEN = 1 in the ADC_CR register.
3. Wait until ADRDY = 1 in the ADC_ISR register (ADRDY is set after the ADC startup time). This can be handled by interrupt if the interrupt is enabled by setting the ADRDYIE bit in the ADC_IER register.

Follow the procedure below to disable the ADC:

1. Check that ADSTART = 0 in the ADC_CR register to ensure that no conversion is ongoing. If the Software trigger mode was used, stop the Software trigger mode by writing 1 to the ADSTP bit of the ADC_CR register and waiting until this bit is read at 0.
2. Set ADDIS = 1 in the ADC_CR register.
3. If required by the application, wait until ADEN = 0 in the ADC_CR register, indicating that the ADC is fully disabled (ADDIS is automatically reset once ADEN = 0).
4. Clear the ADRDY bit in ADC_ISR register by programming this bit to 1 (optional).

Figure 190. Enabling/disabling the ADC

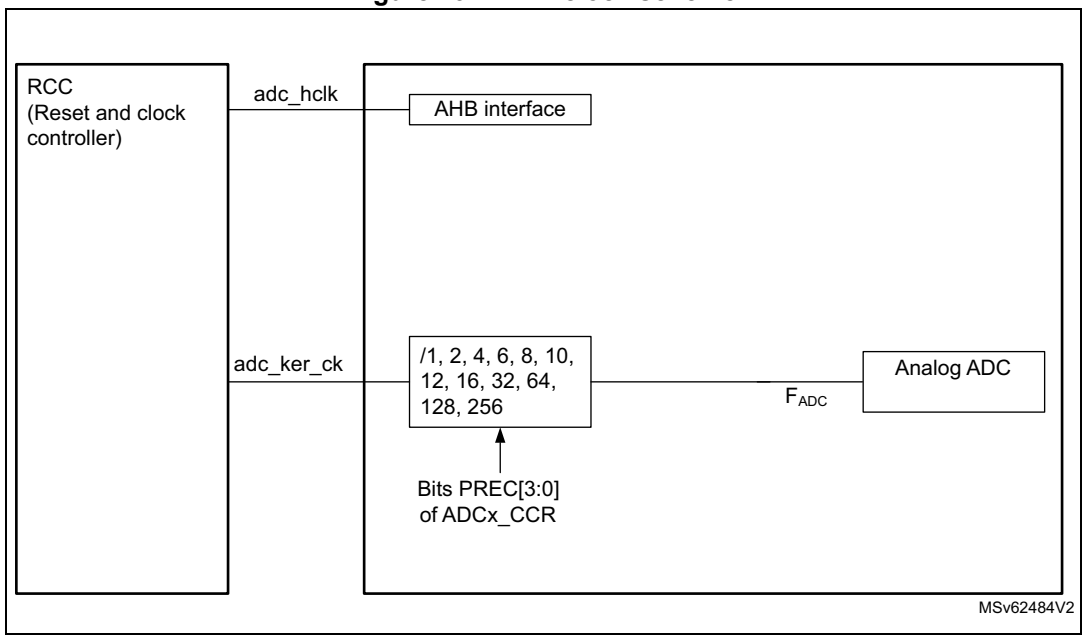


Note: In Auto-off mode (AUTOFF = 1) the power-on/off phases are performed automatically, by hardware and the ADRDY flag is not set.

30.4.6 ADC clock (PRESC[3:0])

The ADC has a dual clock-domain architecture, so that the ADC can be fed with a clock (ADC asynchronous clock) independent from the bus clock.

Figure 191. ADC clock scheme



1. Refer to *Section Reset and clock control (RCC)* for how the bus clock and ADC asynchronous clock are enabled.

The adc_ker_ck input clock can be selected between different clock sources (see *Figure 191: ADC clock scheme*). This selection is done in the RCC (refer to th RCC section for more information):

a) The ADC clock can be provided by an internal or external clock source, which is independent and asynchronous with the bus clock.

b) The ADC clock can be derived from the bus clock.

Option a) has the advantage of reaching the maximum ADC clock frequency whatever the clock scheme selected. The ADC clock can eventually be divided by a programmable ratio of 1, 2, 4, 6, 8, 10, 12, 16, 32, 64, 128 or 256, configured through PRESC[3:0] bits in the ADCx_CCR register.

Option b) has the advantage of bypassing the clock domain resynchronizations. This can be useful when the ADC is triggered by a timer and if the application requires that the ADC is precisely triggered without any uncertainty (otherwise, an uncertainty of the trigger instant is added by the resynchronizations between the two clock domains).

Table 254. Latency between trigger and start of conversion(1)

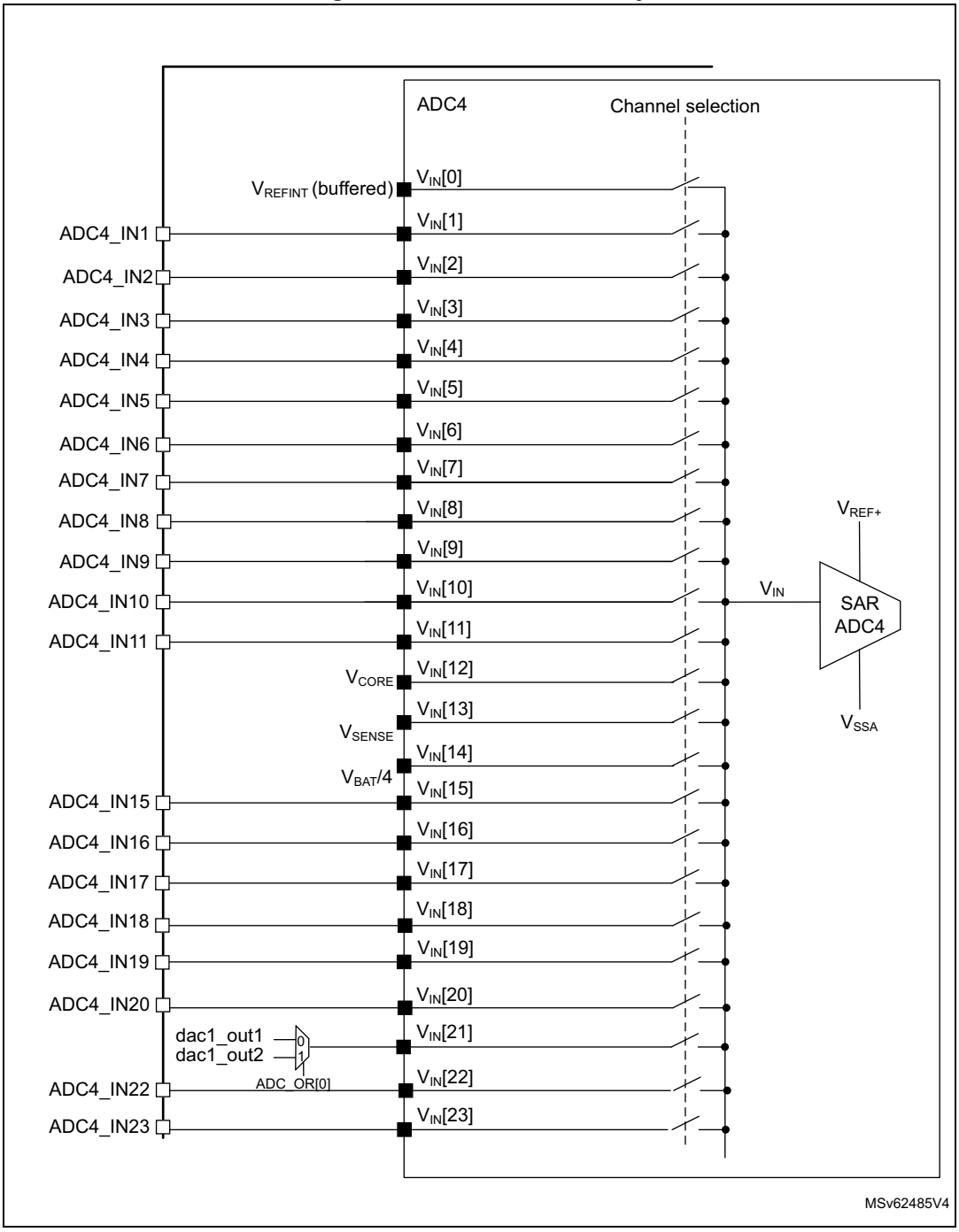
ADC clock source	Latency between the trigger event and the start of conversion
Clock different from bus clock	Latency is not deterministic (jitter)
Bus clock divided by 2	Latency is deterministic (no jitter) and equal to 4 ADC clock cycles
Bus clock divided by 4	Latency is deterministic (no jitter) and equal to 3.75 ADC clock cycles
Bus clock divided by 1	Latency is deterministic (no jitter) and equal to 4 ADC clock cycles

1. Refer to the device datasheet for the maximum F_{ADC} frequency.

30.4.7 ADC connectivity

ADC inputs are connected to the external channels as well as internal sources as described in *Figure 192*.

Figure 192. ADC4 connectivity



30.4.8 Configuring the ADC

The software can write to the ADCAL and ADEN bits in the ADC_CR and ADC_PWRR register if the ADC is disabled (ADEN must be 0).

The software must only write to the ADSTART and ADDIS bits in the ADC_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN = 1 and ADDIS = 0).

For all the other control bits in the ADC_IER, ADC_CFGRi, ADC_SMPR, ADC_CHSELR and ADC_CCR registers, refer to the description of the corresponding control bit in *Section 30.7: ADC registers*. If the ADC operates in Software trigger mode, set the ADSTP bit in ADC_CR register, then wait until ADSTP bit become 0 before reconfiguring the above registers.

ADC_AWDTRi registers can be modified when a conversion is ongoing.

The software must only write to the ADSTP bit in the ADC_CR register if the ADC is enabled (and possibly converting) and there is no pending request to disable the ADC (ADSTART = 1 and ADDIS = 0).

Note: There is no hardware protection preventing software from making write operations forbidden by the above rules. If such a forbidden write access occurs, the ADC may enter an undefined state. To recover correct operation in this case, the ADC must be disabled (clear ADEN = 0 and all the bits in the ADC_CR register).

30.4.9 Channel selection (CHSEL, SCANDIR, CHSELRMOD)

There are up to 25 multiplexed channels:

- 19 analog inputs from GPIO pins (ADC_INx)
- 6 internal analog inputs: temperature Sensor, internal reference voltage, V_{CORE}, V_{BAT} channel, DAC internal channels

It is possible to convert a single channel or a sequence of channels.

The sequence of the channels to be converted can be programmed in the ADC_CHSELR channel selection register: each analog input channel has a dedicated selection bit (CHSELx).

The ADC scan sequencer can be used in two different modes:

- Sequencer not fully configurable:

  The order in which the channels are scanned is defined by the channel number (CHSELRMOD bit must be cleared in ADC_CFGR1 register):
  - Sequence length configured through CHSELx bits in ADC_CHSELR register
  - Sequence direction: the channels are scanned in a forward direction (from the lowest to the highest channel number) or backward direction (from the highest to the lowest channel number) depending on the value of SCANDIR bit (SCANDIR = 0: forward scan, SCANDIR = 1: backward scan)

  - Any channel can belong to in these sequences
- Fully-configurable sequencer

  The CHSELRMOD bit is set in ADC_CFGR1 register.
  - Sequencer length is up to eight channels
  - The order in which the channels are scanned is independent from the channel number. Any order can be configured through SQ1[3:0] to SQ8[3:0] bits in ADC_CHSELR register.
  - Only 15 channels can be selected in this sequence (refer to *Section 30.7.10: ADC channel selection register [alternate] (ADC_CHSELR)*.
  - If the sequencer detects SQx[3:0] = 0b1111, the following SQx[3:0] registers are ignored.
  - If no 0b1111 is programmed in SQx[3:0], the sequencer scans full eight channels.

The software is allowed to program the CHSEL, SCANDIR and CHSELRMOD bit only when ADSTART bit is cleared in ADC_CR register. This ensures that no conversion is ongoing. If the ADC operated in Software trigger mode, set ADSTP bit then wait until ADSTP bit become 0 before reconfiguring these registers. This sequence must be respected even if ADSTART bit is cleared to 0 after the conversion,

Temperature sensor, DAC output, V_{REFINT}, V_{BAT}, and V_{CORE} internal channels

The temperature sensor, the internal DAC channels, the internal reference voltage (V_{REFINT}), V_{BAT}, and V_{CORE} are connected to ADC internal channels. Refer to *Table ADC interconnection* in *Section 30.4.2: ADC pins and internal signals* for details.

30.4.10 Programmable sampling time (SMPx[2:0])

Before starting a conversion, the ADC needs to establish a direct connection between the voltage source to be measured and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the sample and hold capacitor to the input voltage level.

Having a programmable sampling time allows the conversion speed to be trimmed according to the input resistance of the input voltage source.

The ADC samples the input voltage for a number of ADC clock cycles that can be modified using the SMP1[2:0] and SMP2[2:0] bits in the ADC_SMPR register.

Each channel can choose one out of two sampling times configured in SMP1[2:0] and SMP2[2:0] bitfields, through SMPSELx bits in ADC_SMPR register.

The total conversion time is calculated as follows:

t_{CONV} = Sampling time + 12.5 x ADC clock cycles

Example:

With ADC_CLK = 16 MHz and a sampling time of 1.5 ADC clock cycles:

t_{CONV} = 1.5 + 12.5 = 14 ADC clock cycles = 0.875 µs

The ADC indicates the end of the sampling phase by setting the EOSMP flag.

30.4.11 Single conversion mode (CONT = 0)

In Single conversion mode, the ADC performs a single sequence of conversions, converting all the channels once. This mode is selected when CONT is cleared in the ADC_CFGR1 register. Conversion is started by either:

- Setting the ADSTART bit in the ADC_CR register
- Hardware trigger event

Inside the sequence, after each conversion is complete:

- The converted data are stored in the 16-bit ADC_DR register
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then the ADC stops until a new external trigger event occurs or the ADSTART bit is set again.

Note: To convert a single channel, program a sequence with a length of 1.

30.4.12 Continuous conversion mode (CONT = 1)

In continuous conversion mode, when a software or hardware trigger event occurs, the ADC performs a sequence of conversions, converting all the channels once and then automatically re-starts and continuously performs the same sequence of conversions. This mode is selected when CONT is set to 1 in the ADC_CFGR1 register. Conversion is started by either:

- Setting the ADSTART bit in the ADC_CR register
- Hardware trigger event

Inside the sequence, after each conversion is complete:

- The converted data are stored in the 16-bit ADC_DR register
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

Note: To convert a single channel, program a sequence with a length of 1.

It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN = 1 *and CONT* = 1*.*

30.4.13 Starting conversions (ADSTART)

Software starts ADC conversions by setting ADSTART to 1.

When ADSTART is set, the conversion:

- Starts immediately if EXTEN = 00 (software trigger)
- At the next active edge of the selected hardware trigger if EXTEN ≠ 00

The ADSTART bit is also used to indicate whether an ADC operation is currently ongoing. It is possible to re-configure the ADC while ADSTART remains at 0, indicating that the ADC is idle.

The ADSTART bit is cleared by hardware:

- In single mode with software trigger (CONT = 0, EXTEN = 00)
  - At any end of conversion sequence (EOS = 1)
- In discontinuous mode with software trigger (CONT = 0, DISCEN = 1, EXTEN = 00)
  - At end of conversion (EOC = 1)
- In all cases (CONT = x, EXTEN = XX)
  - After execution of the ADSTP procedure invoked by software (see *Section 30.4.15: Stopping an ongoing conversion (ADSTP) on page 1099*).

When the ADC operates in Autonomous mode (DPD bit transition from 1 to 0, see *Section : Autonomous mode (AUTOFF, DPD)*), the ADSTART bit can be set only when the ADC is powered on. (both LDORDY = 1 and ADRDY = 1). In Continuous mode (CONT = 1), the ADSTART bit is not cleared by hardware when the EOS flag is set because the sequence is automatically relaunched.

Note: When hardware trigger is selected in single mode (CONT = 0 and EXTEN = 01), ADSTART is not cleared by hardware when the EOS flag is set. This avoids the need for software having to set the ADSTART bit again and ensures the next trigger event is not missed.

It is necessary to set ADSTP to 1 and wait until ADSTP is cleared before reconfiguring or disabling the ADC, even if ADSTART bit is cleared to after the software triggered ADC conversion mode.

30.4.14 Timings

The elapsed time between the start of a conversion and the end of conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

$t_{CONV} = t_{SMPL} + t_{SAR} = [1.5\ _{|min} + 12.5\ _{|12bit}] \times t_{ADC\_CLK}$

$t_{CONV} = t_{SMPL} + t_{SAR} = 42.9\ ns\ _{|min} + 357.1\ ns\ _{|12bit} = 0.400\ \mu s\ _{|min}$ (for $f_{ADC\_CLK} = 35$ MHz)

Figure 193. Analog to digital conversion time

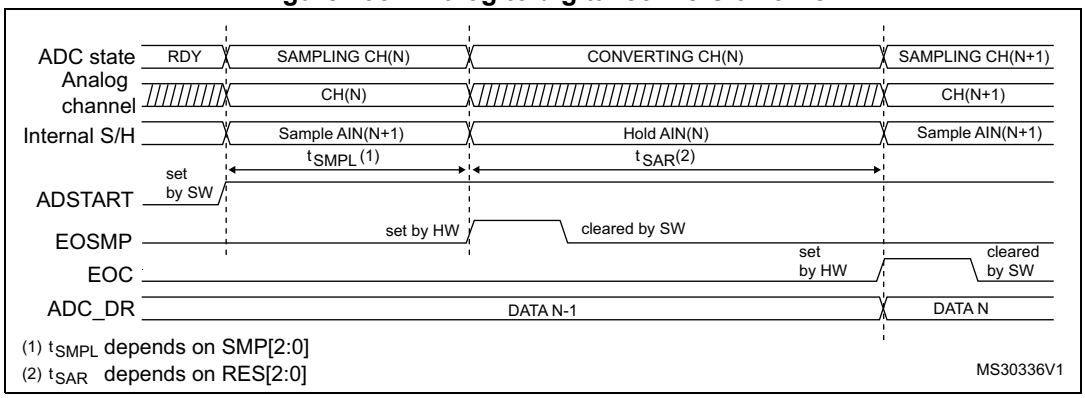
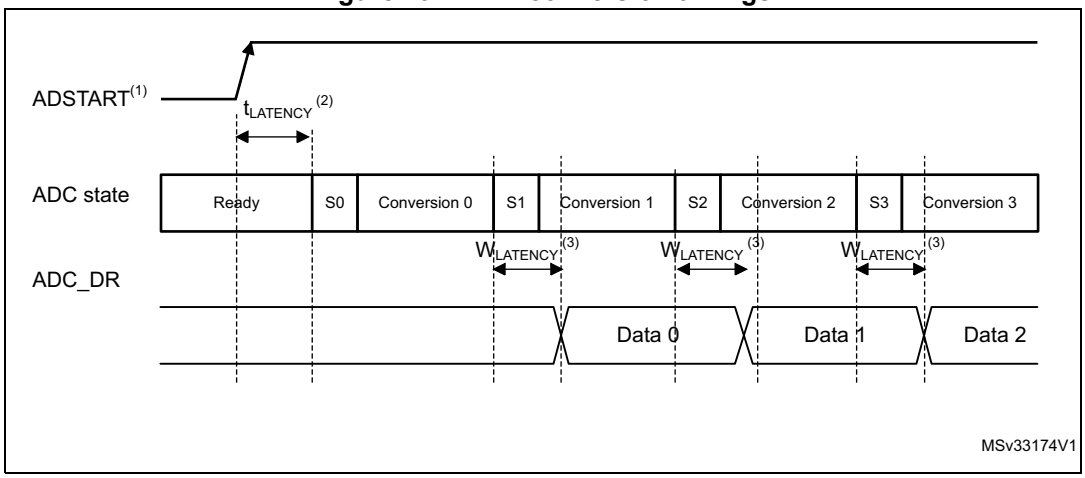


Figure 194. ADC conversion timings



1. EXTEN = 00 or EXTEN ≠ 00.
2. Trigger latency (refer to datasheet for more details).
3. ADC_DR register write latency (refer to datasheet for more details).

30.4.15 Stopping an ongoing conversion (ADSTP)

The software can decide to stop any ongoing conversions by setting ADSTP to 1 in the ADC_CR register.

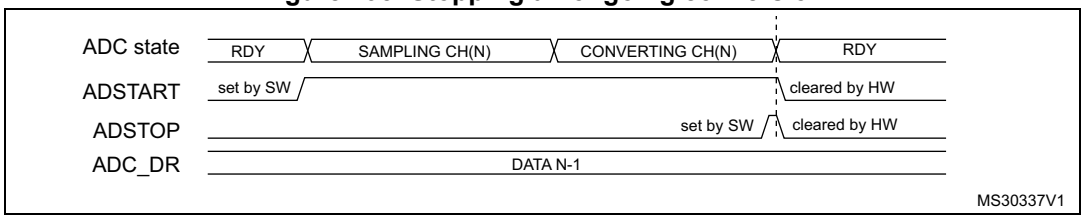
This resets the ADC operation and the ADC is idle, ready for a new operation.

When the ADSTP bit is set by software, any ongoing conversion is aborted and the result is discarded (ADC_DR register is not updated with the current conversion).

The scan sequence is also aborted and reset (meaning that restarting the ADC would re-start a new sequence).

Once this procedure is complete, the ADSTP and ADSTART bits are both cleared by hardware and the software must wait until ADSTART is cleared to 0 before starting new conversions.

Figure 195. Stopping an ongoing conversion



30.4.16 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN)

A conversion or a sequence of conversion can be triggered either by software or by an external event (for example timer capture). If the EXTEN[1:0] control bits are not equal to “0b00”, then external events are able to trigger a conversion with the selected polarity. The trigger selection is effective once software has set bit ADSTART to 1.

Any hardware triggers which occur while a conversion is ongoing are ignored.

If bit ADSTART is cleared, any hardware triggers which occur are ignored.

Table 255 provides the correspondence between the EXTEN[1:0] values and the trigger polarity.

Table 255. Configuring the trigger polarity

Source	EXTEN[1:0]
Trigger detection disabled	00
Detection on rising edge	01
Detection on falling edge	10
Detection on both rising and falling edges	11

Note: The polarity of the external trigger can be changed only when the ADC is not converting (ADSTART = 0).

The EXTSEL[2:0] control bits are used to select which of 8 possible events can trigger conversions.

Refer to Table ADC interconnection in *Section 30.4.2: ADC pins and internal signals* for the list of all the external triggers that can be used for regular conversion.

The software source trigger events can be generated by setting the ADSTART bit in the ADC_CR register.

Note: The trigger selection can be changed only when the ADC is not converting (ADSTART = 0).

30.4.17 Discontinuous mode (DISCEN)

This mode is enabled by setting the DISCEN bit in the ADC_CFGR1 register.

In this mode (DISCEN = 1), a hardware or software trigger event is required to start each conversion defined in the sequence. On the contrary, if DISCEN is cleared, a single hardware or software trigger event successively starts all the conversions defined in the sequence.

Example:

- DISCEN = 1, channels to be converted are channels 0, 3, 7 and 10
  - 1st trigger: channel 0 is converted and an EOC event is generated
  - 2nd trigger: channel 3 is converted and an EOC event is generated
  - 3rd trigger: channel 7 is converted and an EOC event is generated
  - 4th trigger: channel 10 is converted and both EOC and EOS events are generated.
  - 5th trigger: channel 0 is converted an EOC event is generated
  - 6th trigger: channel 3 is converted and an EOC event is generated
  - ...
- DISCEN = 0, channels to be converted are channels 0, 3, 7 and 10
  - 1st trigger: the complete sequence is converted: channel 0, then 3, 7 and 10. Each conversion generates an EOC event and the last one also generates an EOS event.
  - Any subsequent trigger events restarts the complete sequence.

Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN = 1 and CONT = 1.

30.4.18 Programmable resolution (RES) - fast conversion mode

It is possible to obtain faster conversion times (t_{SAR}) by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the RES[1:0] bits in the ADC_CFGR1 register. Lower resolution allows faster conversion times for applications where high data precision is not required.

Note: The RES[1:0] bit must only be changed when the ADEN bit is reset.

The result of the conversion is always 12 bits wide and any unused LSB bits are read as zeros.

Lower resolution reduces the conversion time needed for the successive approximation steps as shown in *Table 256*.

Table 256. t_{SAR} timings depending on resolution

RES[1:0] bits	t_{SAR} (ADC clock cycles)	t_{SAR} (ns) at f_{ADC} = 35 MHz	$t_{SMPL\ (min)}$ (ADC clock cycles)	t_{CONV} (ADC clock cycles) (with min. $t_{SMPL)}$	t_{CONV} (ns) at f_{ADC} = 35 MHz
12	12.5	357	1.5	14	400
10	10.5	300	1.5	12	343
8	8.5	243	1.5	10	286
6	6.5	186	1.5	8	229

30.4.19 End of conversion, end of sampling phase (EOC, EOSMP flags)

The ADC indicates each end of conversion (EOC) event.

The ADC sets the EOC flag in the ADC_ISR register as soon as a new conversion data result is available in the ADC_DR register. An interrupt can be generated if the EOCIE bit is set in the ADC_IER register. The EOC flag is cleared by software either by writing 1 to it, or by reading the ADC_DR register.

The ADC also indicates the end of sampling phase by setting the EOSMP flag in the ADC_ISR register. The EOSMP flag is cleared by software by writing1 to it. An interrupt can be generated if the EOSMPIE bit is set in the ADC_IER register.

The aim of this interrupt is to allow the processing to be synchronized with the conversions. Typically, an analog multiplexer can be accessed in hidden time during the conversion phase, so that the multiplexer is positioned when the next sampling starts.

Note: As there is only a very short time left between the end of the sampling and the end of the conversion, it is recommenced to use polling or a WFE instruction rather than an interrupt and a WFI instruction.

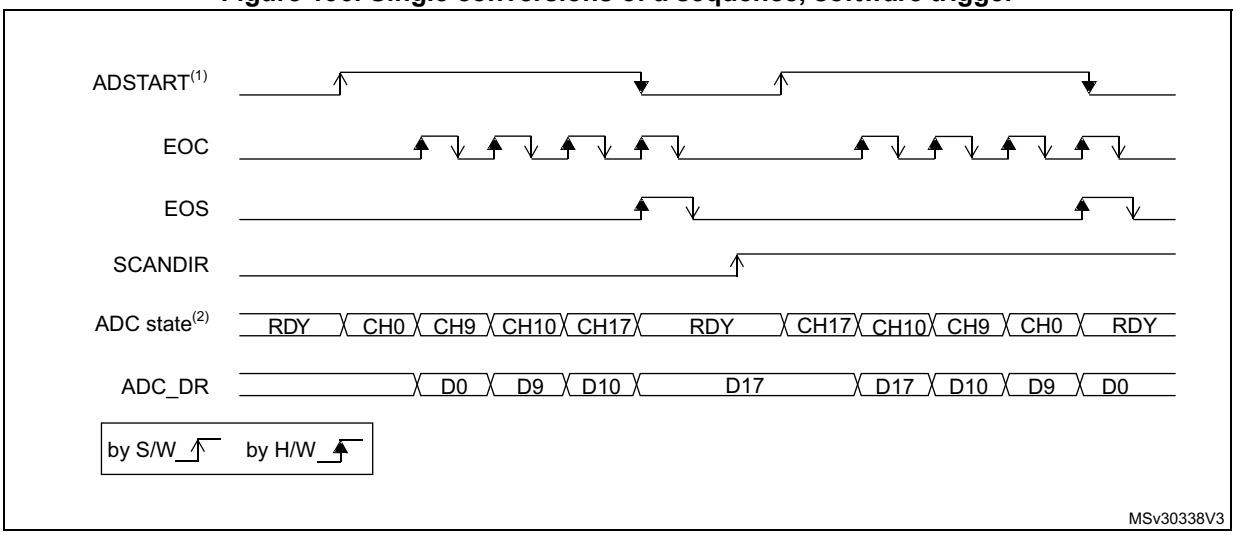
30.4.20 End of conversion sequence (EOS flag)

The ADC notifies the application of each end of sequence (EOS) event.

The ADC sets the EOS flag in the ADC_ISR register as soon as the last data result of a conversion sequence is available in the ADC_DR register. An interrupt can be generated if the EOSIE bit is set in the ADC_IER register. The EOS flag is cleared by software by writing 1 to it.

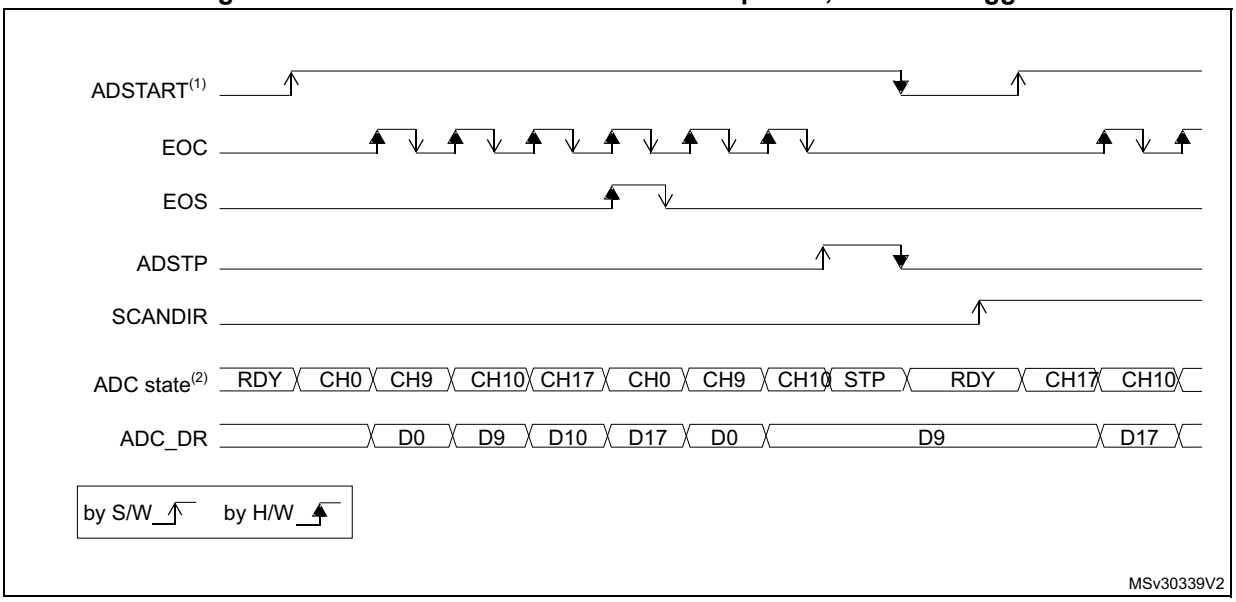
30.4.21 Example timing diagrams (single/continuous modes hardware/software triggers)

Figure 196. Single conversions of a sequence, software trigger



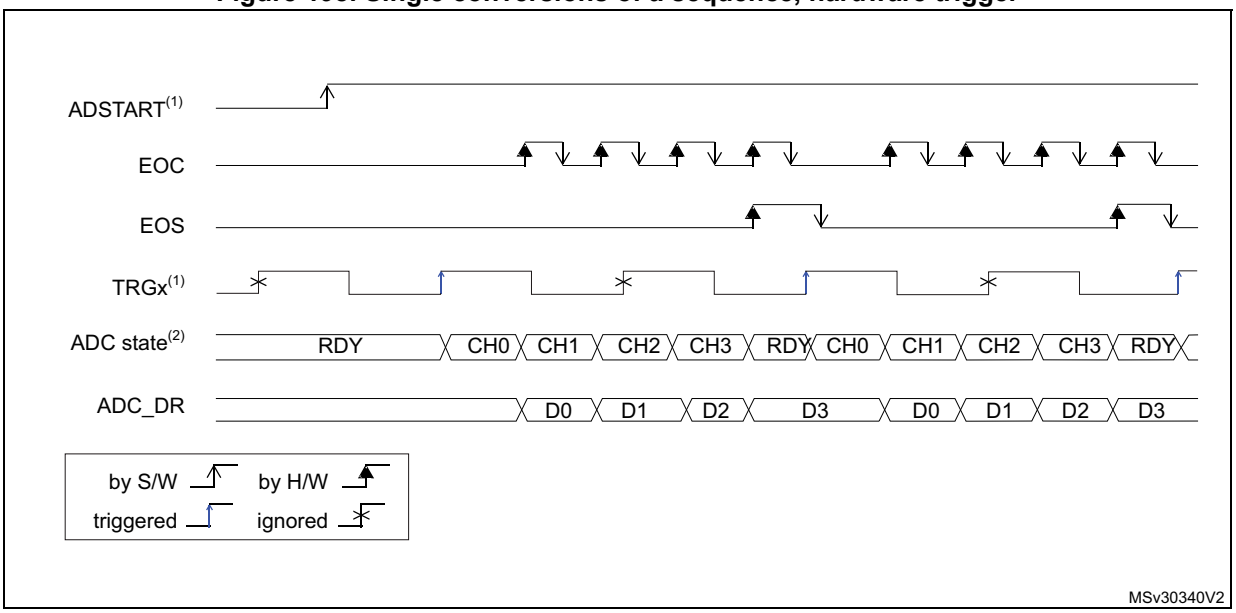
1. EXTEN = 00, CONT = 0.
2. CHSEL = 0x20601, WAIT = 0, AUTOFF = 0.

Figure 197. Continuous conversion of a sequence, software trigger



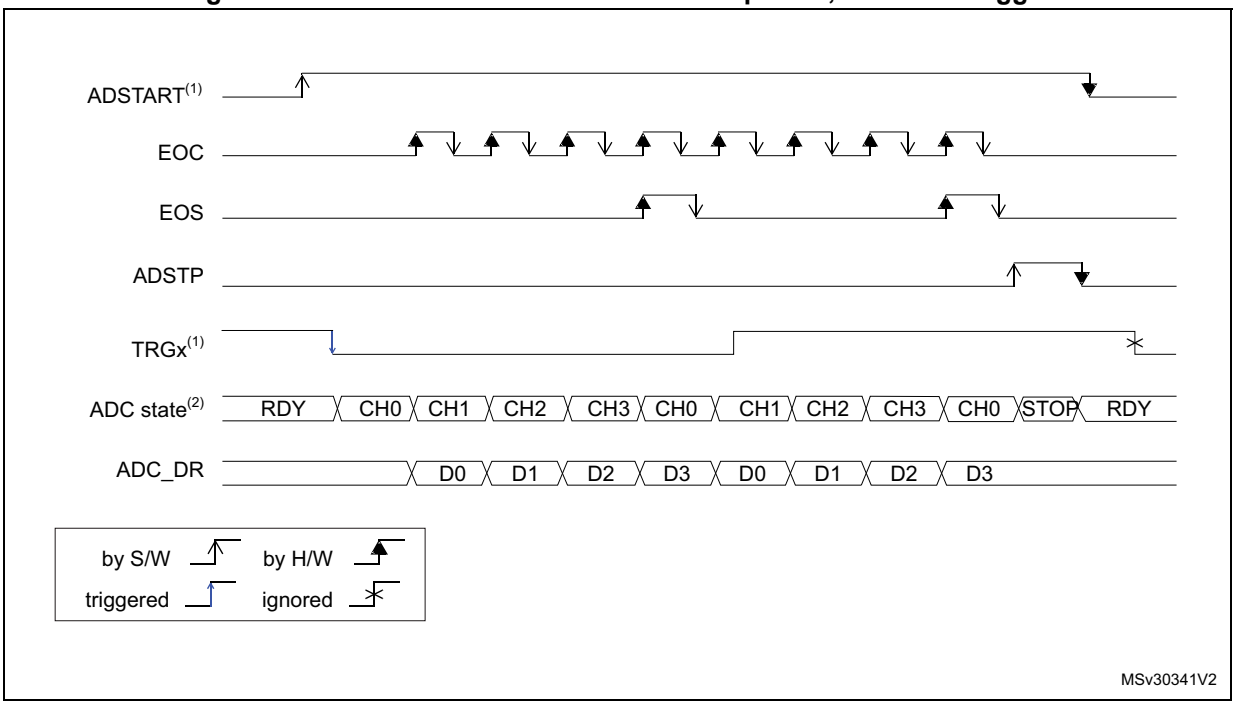
1. EXTEN = 00, CONT = 1.
2. CHSEL = 0x20601, WAIT = 0, AUTOFF = 0.

Figure 198. Single conversions of a sequence, hardware trigger



1. EXTSEL = TRGx (over-frequency), EXTEN = 01 (rising edge), CONT = 0.
2. CHSEL = 0xF, SCANDIR = 0, WAIT = 0, AUTOFF = 0.

Figure 199. Continuous conversions of a sequence, hardware trigger



1. EXTSEL = TRGx, EXTEN = 10 (falling edge), CONT = 1.
2. CHSEL = 0xF, SCANDIR = 0, WAIT = 0, AUTOFF = 0.

30.4.22 Low frequency trigger mode

If the application has to support a time longer than the maximum t_{IDLE} value (between one trigger to another for single conversion mode or between the ADC enable and the first ADC conversion), then the ADC internal state needs to be rearmed. This mechanism can be enabled by setting LFTRIG bit to 1 in ADC_CFGR2 register. By setting this bit, any trigger (software or hardware) sends a rearm command to ADC. The conversion is started after a two ADC clock cycle delay compared to LFTRIG set to 0.

It is not necessary to use this mode when AUTOFF bit is set to 1. For Wait mode, only the first trigger generates an internal rearm command.

30.4.23 Data management

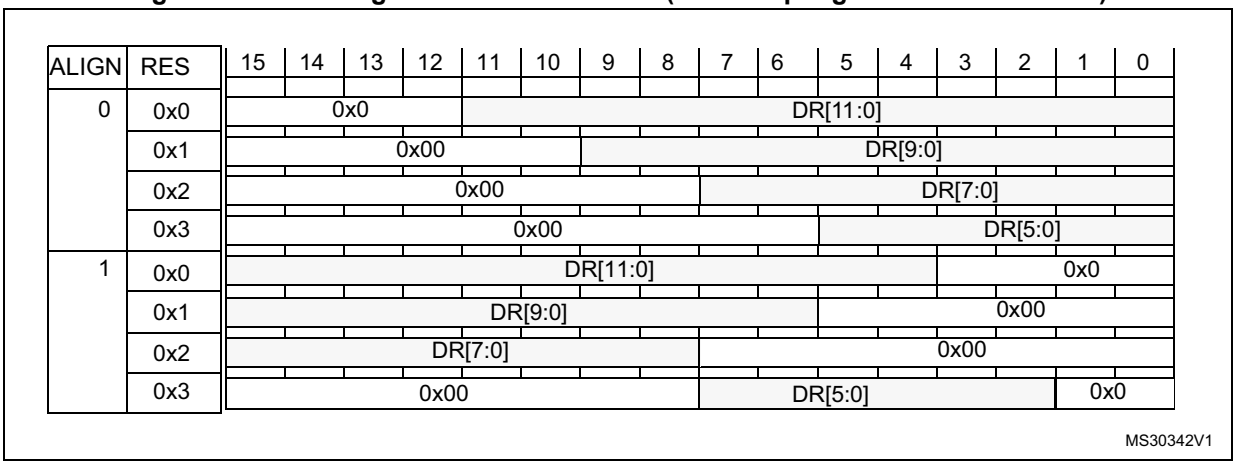
Data register and data alignment (ADC_DR, ALIGN)

At the end of each conversion (when an EOC event occurs), the result of the converted data is stored in the ADC_DR data register which is 16-bit wide.

The format of the ADC_DR depends on the configured data alignment and resolution.

The ALIGN bit in the ADC_CFGR1 register selects the alignment of the data stored after conversion. Data can be right-aligned (ALIGN = 0) or left-aligned (ALIGN = 1) as shown in *Figure 200*.

Figure 200. Data alignment and resolution (oversampling disabled: OVSE = 0)



ADC overrun (OVR, OVRMOD)

The overrun flag (OVR) indicates a data overrun event, when the converted data was not read in time by the CPU or the DMA, before the data from a new conversion is available.

The OVR flag is set in the ADC_ISR register if the EOC flag is still at 1 at the time when a new conversion completes. An interrupt can be generated if the OVRIE bit is set in the ADC_IER register.

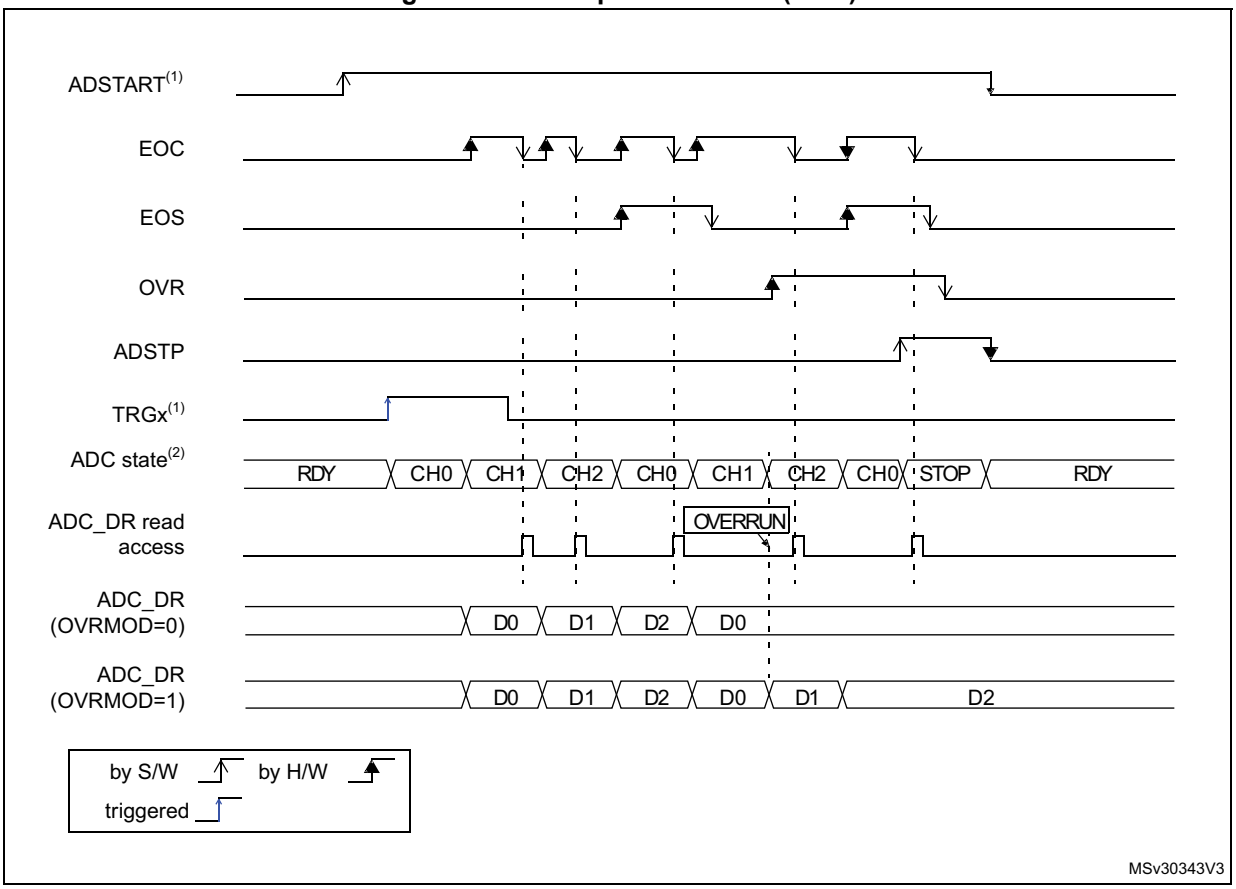
When an overrun condition occurs, the ADC keeps operating and can continue to convert unless the software decides to stop and reset the sequence by setting the ADSTP bit in the ADC_CR register.

The OVR flag is cleared by software by writing 1 to it.

It is possible to configure if the data is preserved or overwritten when an overrun event occurs by programming the OVRMOD bit in the ADC_CFGR1 register:

- OVRMOD = 0
  - An overrun event preserves the data register from being overwritten: the old data is maintained and the new conversion is discarded. If OVR remains at 1, further conversions can be performed but the resulting data is discarded.
- OVRMOD = 1
  - The data register is overwritten with the last conversion result and the previous unread data is lost. If OVR remains at 1, further conversions can be performed and the ADC_DR register always contains the data from the latest conversion.

Figure 201. Example of overrun (OVR)



Managing a sequence of data converted without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by software. In this case the software must use the EOC flag and its associated interrupt to handle each data result. Each time a conversion is complete, the EOC bit is set in the ADC_ISR register and the ADC_DR register can be read. The OVRMOD bit in the ADC_CFGR1 register must be configured to 0 to manage overrun events as an error.

Managing converted data without using the DMA without overrun

It may be useful to let the ADC convert one or more channels without reading the data after each conversion. In this case, the OVRMOD bit must be configured at 1 and the OVR flag must be ignored by the software. When OVRMOD is et to 1, an overrun event does not prevent the ADC from continuing to convert and the ADC_DR register always contains the latest conversion data.

Managing converted data using the DMA

Since all converted channel values are stored in a single data register, it is efficient to use DMA when converting more than one channel. This avoids losing the conversion data results stored in the ADC_DR register.

When DMA mode is enabled (DMAEN bit set to 1 in the ADC_CFGR1 register), a DMA request is generated after the conversion of each channel. This allows the transfer of the

converted data from the ADC_DR register to the destination location selected by the software.

Note: The DMAEN bit in the ADC_CFGR1 register must be set after the ADC calibration phase.

Despite this, if an overrun occurs (OVR = 1) because the DMA could not serve the DMA transfer request in time, the ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA. Which means that all the data transferred to the RAM can be considered as valid.

Depending on the configuration of OVRMOD bit, the data is either preserved or overwritten (refer to *Section : ADC overrun (OVR, OVRMOD) on page 1105*).

The DMA transfer requests are blocked until the software clears the OVR bit.

Two different DMA modes are proposed depending on the application use and are configured with bit DMACFG in the ADC_CFGR1 register:

- DMA one shot mode (DMACFG = 0).

  This mode must be selected when the DMA is programmed to transfer a fixed number of data words.
- DMA circular mode (DMACFG = 1)

  This mode must be selected when programming the DMA in circular mode or double buffer mode.

DMA one shot mode (DMACFG = 0)

In this mode, the ADC generates a DMA transfer request each time a new conversion data word is available and stops generating DMA requests once the DMA has reached the last DMA transfer (when a transfer complete interrupt occurs - refer to DMA section), even if a conversion has been started again.

When the DMA transfer is complete (all the transfers configured in the DMA controller have been done):

- The content of the ADC data register is frozen.
- Any ongoing conversion is aborted and its partial result discarded
- No new DMA request is issued to the DMA controller. This avoids generating an overrun error if there are still conversions which are started.
- The scan sequence is stopped and reset
- The DMA is stopped

DMA circular mode (DMACFG = 1)

In this mode, the ADC generates a DMA transfer request each time a new conversion data word is available in the data register, even if the DMA has reached the last DMA transfer. This allows the DMA configuration in circular mode in order to handle a continuous analog input data stream.

30.4.24 Low-power features

Wait conversion mode (WAIT)

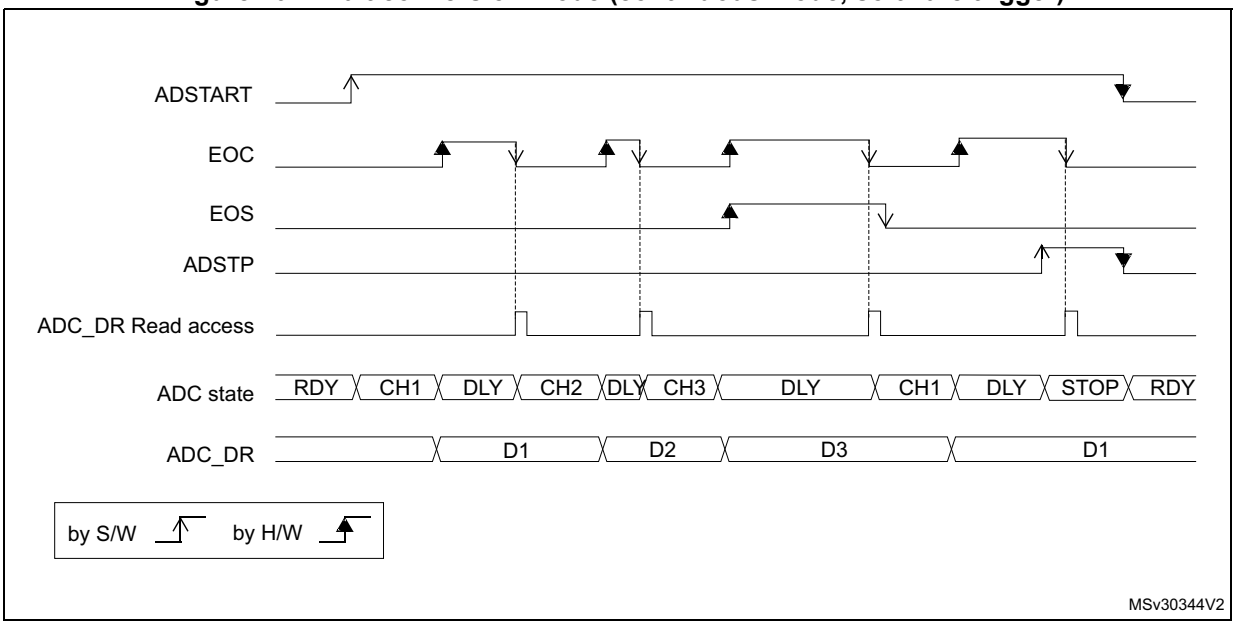
Wait conversion mode can be used to simplify the software as well as optimizing the performance of applications clocked at low frequency where there might be a risk of ADC overrun occurring.

When the WAIT bit is set to 1 in the ADC_CFGR1 register, a new conversion can start only if the previous data has been treated, once the ADC_DR register has been read or if the EOC bit has been cleared.

This is a way to automatically adapt the speed of the ADC to the speed of the system that reads the data.

Note: Any hardware triggers which occur while a conversion is ongoing or during the wait time preceding the read access are ignored.

Figure 202. Wait conversion mode (continuous mode, software trigger)



1. EXTEN = 00, CONT = 1.
2. CHSEL = 0x3, SCANDIR = 0, WAIT = 1, AUTOFF = 0.

ADC power-saving modes

The ADC embeds two power-saving modes, the Auto-off and the Autonomous modes.

Auto-off mode (AUTOFF)

The Auto-off mode is enabled by setting the AUTOFF bit to 1 in the ADC_PWRR register.

Below the Auto-off mode operating sequence:

1. When AUTOFF is set to 1, the ADC is always powered off when no conversion is ongoing.
2. It then automatically wakes up when a conversion is triggered by software or by hardware, and a startup time is inserted between the trigger event and the ADC sampling time.
3. The ADC is then automatically disabled once the conversion or sequence of conversions is complete.
4. When consecutive hardware or software triggers occur, the ADC is automatically enabled and the conversion is processed.

Refer to *Figure 203* for a description of Auto-off mode state diagram.

The Auto-off mode dramatically reduces power consumption in applications requiring a limited number of conversions or conversion requests far between enough (for example with a low frequency hardware trigger) to justify the extra power and time used for switching the ADC on and off.

Auto-off mode can be combined with Wait mode (WAIT = 1) for applications clocked at low frequency. This combination can achieve significant power saving if the ADC is automatically powered off during the wait phase and restarted as soon as the ADC_DR register is read by the application (see *Figure 204: ADC behavior with WAIT = 0 and AUTOFF = 1* and *Figure 205: ADC behavior with WAIT = 1 and AUTOFF = 1*).

The Auto-off mode is compatible with the low-power background autonomous mode (LPBAM).

Note: Refer to the Section Reset and clock control (RCC) for the description of how to manage the dedicated internal oscillators. The ADC interface can automatically switch on/off these internal oscillators to save power.

Figure 203. Auto-off mode state diagram

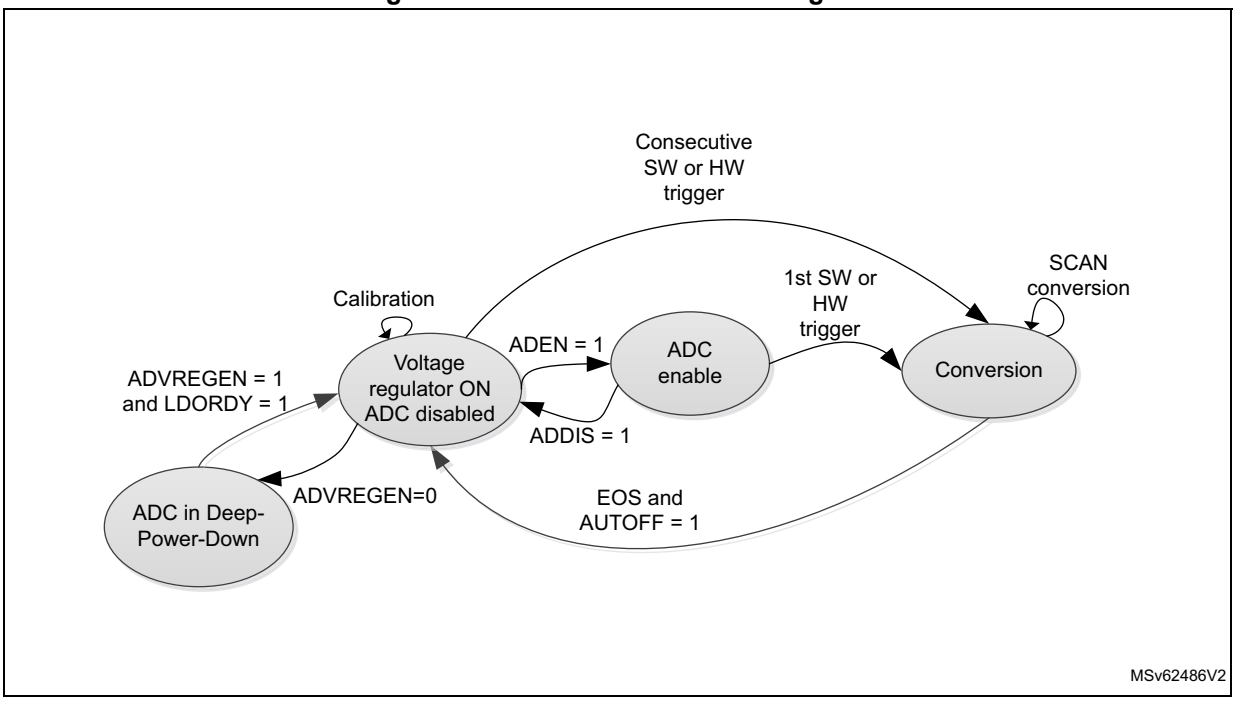
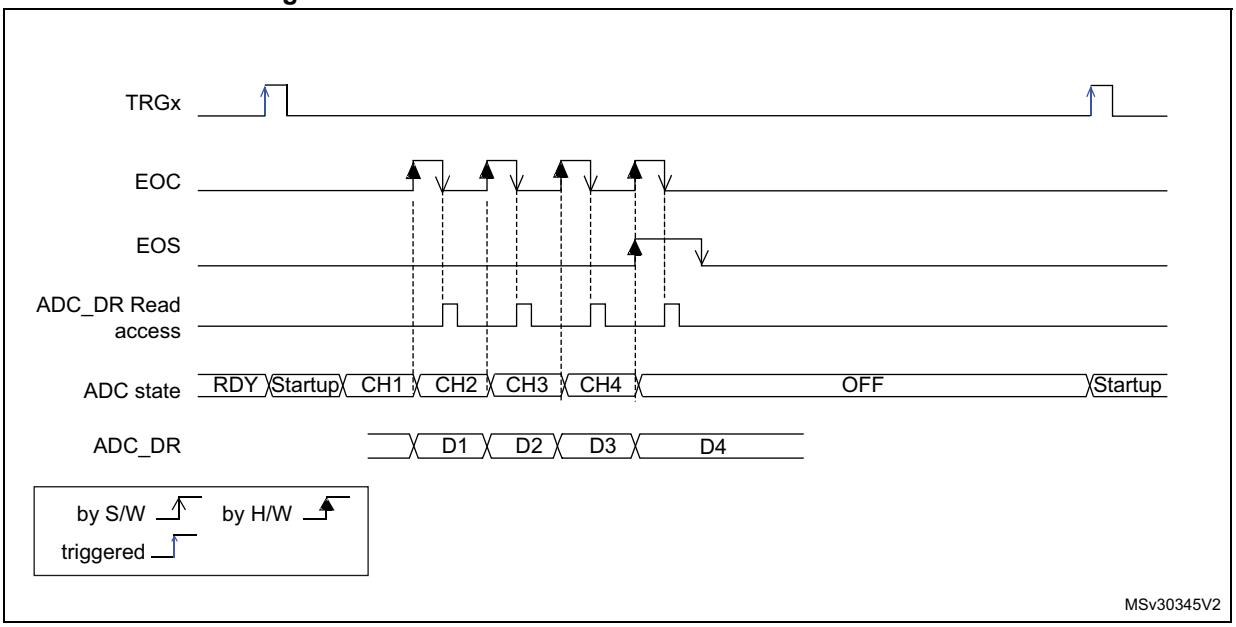
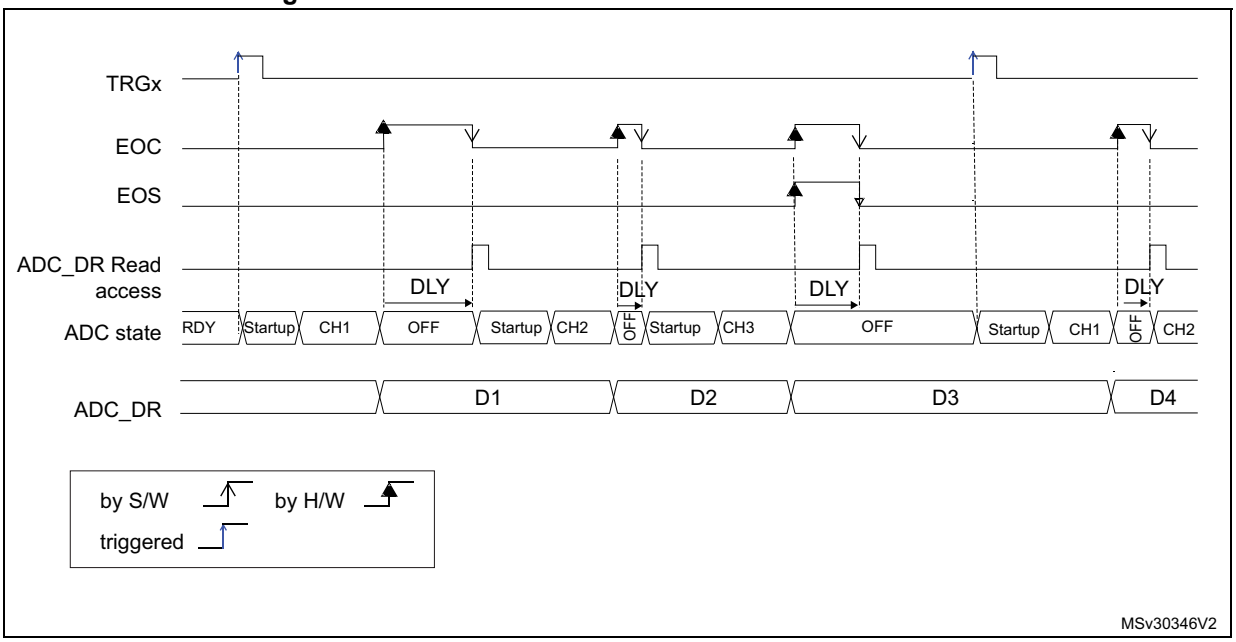


Figure 204. ADC behavior with WAIT = 0 and AUTOFF = 1



1. EXTSEL = TRGx, EXTEN = 01 (rising edge), CONT = x, ADSTART = 1, CHSEL = 0xF, SCANDIR = 0, WAIT = 0, AUTOFF = 1.

Figure 205. ADC behavior with WAIT = 1 and AUTOFF = 1



1. EXTSEL = TRGx, EXTEN = 01 (rising edge), CONT = x, ADSTART = 1, CHSEL = 0xF, SCANDIR = 0, WAIT = 1, AUTOFF = 1.

Autonomous mode (AUTOFF, DPD)

The Autonomous mode is enabled by setting both AUTOFF and DPD bits to 1 in ADC_PWRR register. In addition, the Autonomous mode must be enabled in the RCC.

Below the Autonomous mode operating sequence:

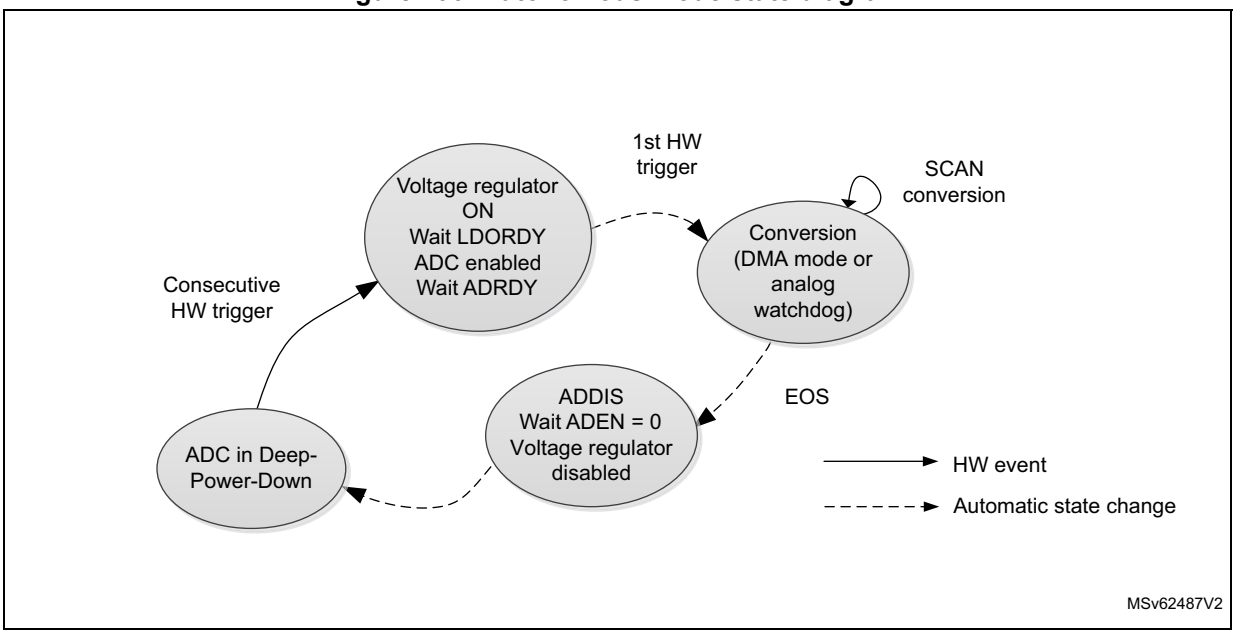
1. When AUTOFF and DPD are both set to 1, the ADC is powered off when no conversion is ongoing.
2. Upon hardware trigger reception, the ADC requests the adc_ker_ck and adc_hclk clocks to the RCC, the ADC voltage regulator is enabled, the calibration factor is loaded, the ADC is enabled and the conversion starts.
3. Once the ADC conversion is complete, the ADC can either generate an AWDx interrupt or a DMA request, depending on peripheral configuration:
   - When DMA mode is enabled, the ADC generates a DMA request to transfer data to memory or to another peripherals.
   - When an analog watchdog is enabled, ADC data do not need to be transferred. The analog watchdog compares the data to the threshold value and generates an AWDx interrupt to wake up the device if the data is under or over the programmed threshold.
4. When the ADC conversion/sequence or conversion is complete, the ADC and the ADC voltage regulator are automatically disabled as well as V_{REFINT} buffer and the temperature sensor, and further clock requests are deasserted. This allows the minimization of current consumption.
5. When consecutive hardware triggers occur, the ADC is automatically enabled and the conversion is processed.

Refer to *Figure 206* for a description of Autonomous mode state diagram.

The Autonomous mode enables the ADC peripheral to operate when the device is in Stop mode. However it can also be used in Run or Sleep mode.

It is compatible with the low-power background autonomous mode (LPBAM).

Figure 206. Autonomous mode state diagram



30.4.25 Analog window watchdog

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

Description of analog watchdog 1

AWD1 analog watchdog is enabled by setting the AWD1EN bit in the ADC_CFGR1 register. It is used to monitor that either one selected channel or all enabled channels (see *Table 258: Analog watchdog 1 channel selection*) remain within a configured voltage range (window) as shown in *Figure 207*.

The AWD1 analog watchdog status bit is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold. These thresholds are programmed in HT1[11:0] and LT1[11:0] bits of ADC_AWD1TR register. An interrupt can be enabled by setting the AWD1IE bit in the ADC_IER register.

The AWD1 flag is cleared by software by programing it to 1.

When converting data with a resolution of less than 12-bit (according to bits DRES[1:0]), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data (left aligned).

Table 257 describes how the comparison is performed for all the possible resolutions.

Table 257. Analog watchdog comparison

Resolution bits RES[1:0]	Analog Watchdog comparison between:		Comments
	Raw converted data, left aligned(1)	Thresholds	
00: 12-bit	DATA[11:0]	LTx[11:0] and HTx[11:0]	-
01: 10-bit	DATA[11:2],00	LTx[11:0] and HTx[11:0]	The user must configure LTx[1:0] and HTx[1:0] to “00”
10: 8-bit	DATA[11:4],0000	LTx[11:0] and HTx[11:0]	The user must configure LTx[3:0] and HTx[3:0] to “0000”
11: 6-bit	DATA[11:6],000000	LTx[11:0] and HTx[11:0]	The user must configure LTx[5:0] and HTx[5:0] to “000000”

1. The watchdog comparison is performed on the raw converted data before any alignment calculation.

Table 258 shows how to configure the AWD1SGL and AWD1EN bits in the ADC_CFGR1 register to enable the analog watchdog on one or more channels.

Figure 207. Analog watchdog guarded area

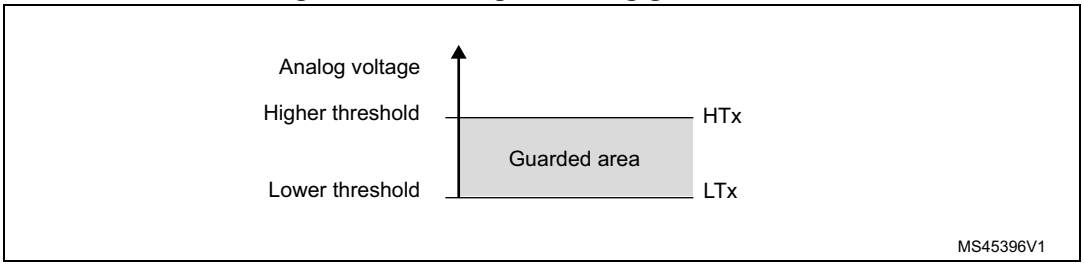


Table 258. Analog watchdog 1 channel selection

Channels guarded by the analog watchdog	AWD1SGL bit	AWD1EN bit
None	x	0
All channels	0	1
Single(1) channel	1	1

1. Selected by the AWD1CH[4:0] bits

Description of analog watchdog 2 and 3

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the AWDxCHy in ADC_AWDxCR (x = 2, 3).

The corresponding watchdog is enabled when any AWDxCHy bit (x = 2,3) is set in ADC_AWDxCR register.

When converting data with a resolution of less than 12 bits (configured through DRES[1:0] bits), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data (left aligned).

Table 257 describes how the comparison is performed for all the possible resolutions.

The AWD2/3 analog watchdog status bit is set if the analog voltage converted by the ADC is below a low threshold or above a high threshold. These thresholds are programmed in

HTx[11:0] and LTx[11:0] of ADC_AWDxTR registers (x = 2 or 3). An interrupt can be enabled by setting the AWDxIE bit in the ADC_IER register.

The AWD2 and AWD3 flags are cleared by software by programming them to 1.

ADC_AWDx_OUT signal output generation

Each analog watchdog is associated to an internal hardware signal, ADC_AWDx_OUT (x being the watchdog number) that is directly connected to the ETR input (external trigger) of some on-chip timers (refer to the timers section for details on how to select the ADC_AWDx_OUT signal as ETR).

ADC_AWDx_OUT is activated when the associated analog watchdog is enabled:

- ADC_AWDx_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADC_AWDx_OUT is reset after the end of the next guarded conversion which is inside the programmed thresholds. It remains at 1 if the next guarded conversions are still outside the programmed thresholds.
- ADC_AWDx_OUT is also reset when disabling the ADC (when setting ADDIS to 1). Note that stopping conversions (ADSTP set to 1), might clear the ADC_AWDx_OUT state.
- ADC_AWDx_OUT state does not change when the ADC converts the none-guarded channel (see *Figure 210*)

AWDx flag is set by hardware and reset by software: AWDx flag has no influence on the generation of ADC_AWDx_OUT (as an example, ADC_AWDx_OUT can toggle while AWDx flag remains at 1 if the software has not cleared the flag).

The ADC_AWDx_OUT signal is generated by the ADC_CLK domain. This signal can be generated even the bus clock is stopped.

The AWD comparison is performed at the end of each ADC conversion. The ADC_AWDx_OUT rising edge and falling edge occurs two ADC_CLK clock cycles after the comparison.

As ADC_AWDx_OUT is generated by the ADC_CLK domain and AWD flag is generated by the bus clock domain, the rising edges of these signals are not synchronized.

Figure 208. ADC_AWDx_OUT signal generation

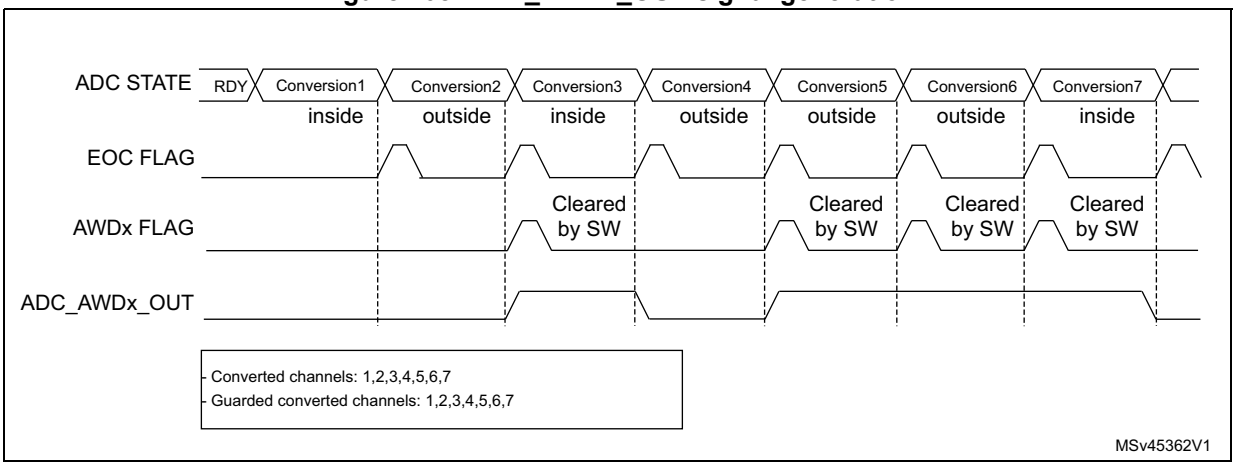


Figure 209. ADC_AWDx_OUT signal generation (AWDx flag not cleared by software)

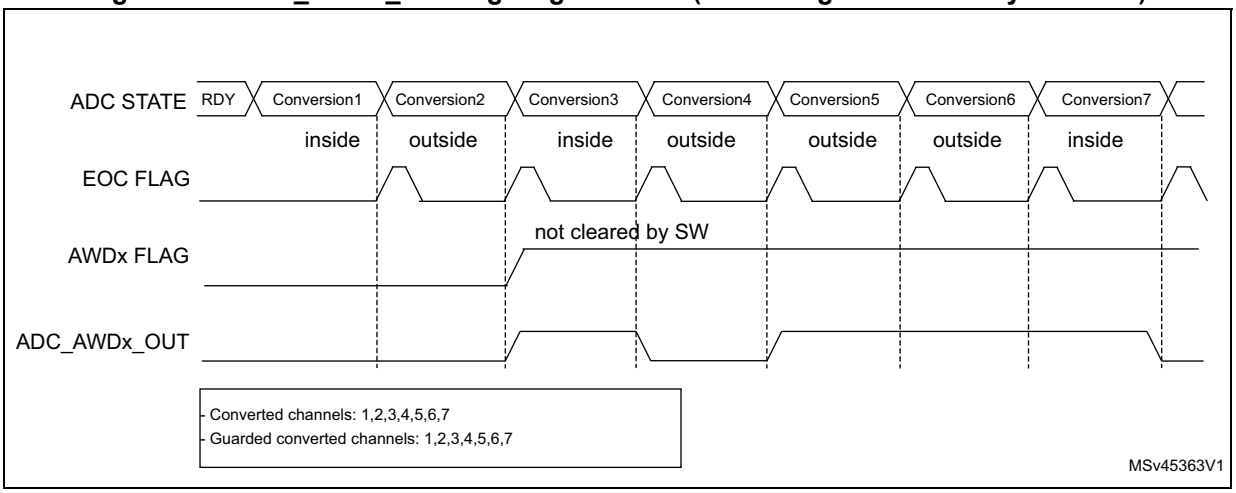
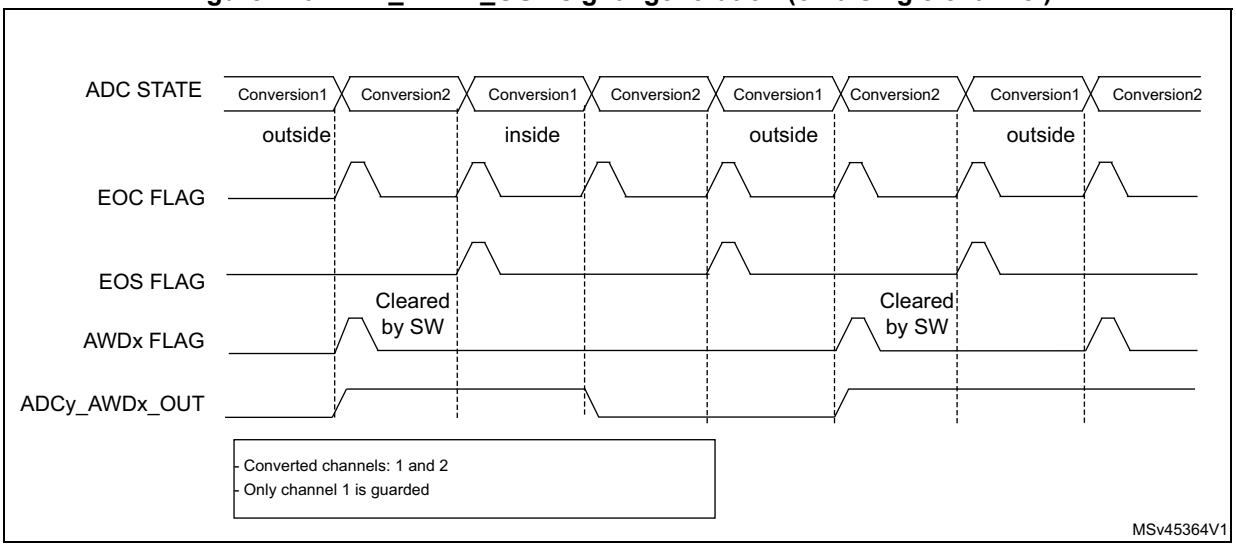


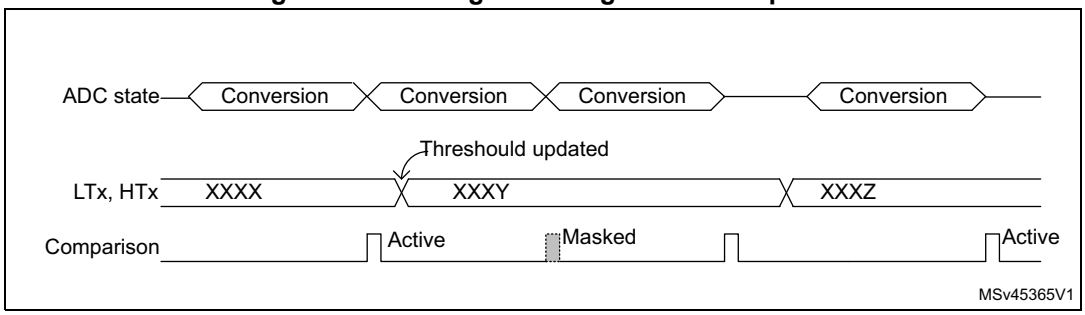
Figure 210. ADC_AWDx_OUT signal generation (on a single channel)



Analog watchdog threshold control

LTx[11:0] and HTx[11:0] can be changed during an analog-to-digital conversion (that is between the start of the conversion and the end of conversion of the ADC internal state). If HTx and LTx bits are programmed during the ADC guarded channel conversion, the watchdog function is masked for this conversion. This mask is cleared when starting a new conversion, and the resulting new AWD threshold is applied starting the next ADC conversion result. AWD comparison is performed at each end of conversion. If the current ADC data are out of the new threshold interval, this does not generated any interrupt or an ADC_AWDx_OUT signal. The Interrupt and the ADC_AWDx_OUT generation only occurs at the end of the ADC conversion that started after the threshold update. If ADC_AWDx_OUT is already asserted, programming the new threshold does not deassert the ADC_AWDx_OUT signal.

Figure 211. Analog watchdog threshold update



30.4.26 Oversampler

The oversampling unit performs data preprocessing to offload the CPU. It can handle multiple conversions and average them into a single data with increased data width, up to 16-bit.

It provides a result with the following form, where N and M can be adjusted:

$$\text{Result} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{Conversion}(t_n)$$

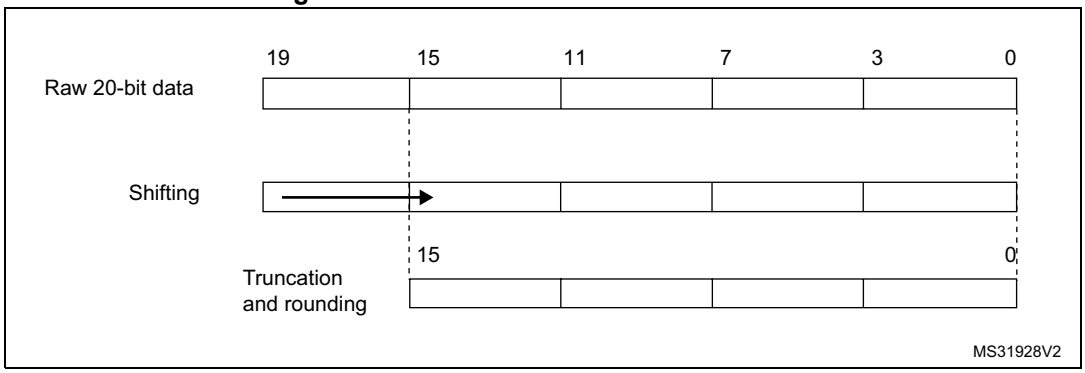
It allows the following functions to be performed by hardware: averaging, data rate reduction, SNR improvement, basic filtering.

The oversampling ratio N is defined using the OVFS[2:0] bits in the ADC_CFGR2 register. It can range from 2x to 256x. The division coefficient M consists of a right bit shift up to 8 bits. It is configured through the OVSS[3:0] bits in the ADC_CFGR2 register.

The summation unit can yield a result up to 20 bits (256 x 12-bit), which is first shifted right. The lower bits of the result are then truncated, keeping only the 16 least significant bits rounded to the nearest value using the least significant bits left apart by the shifting, before being finally transferred into the ADC_DR data register.

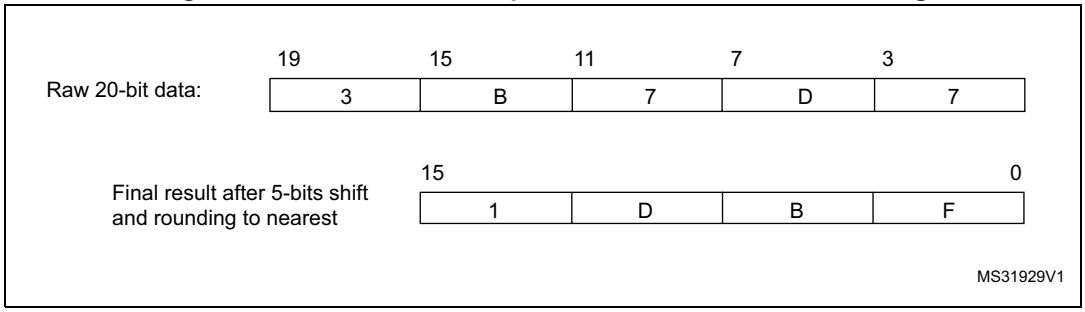
Note: If the intermediate result after the shifting exceeds 16 bits, the upper bits of the result are simply truncated.

Figure 212. 20-bit to 16-bit result truncation



The *Figure 213* gives a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

Figure 213. Numerical example with 5-bits shift and rounding



The *Table 259* below gives the data format for the various N and M combination, for a raw conversion data equal to 0xFFF.

Table 259. Maximum output results vs N and M. Grayed values indicates truncation

Oversampling ratio	Max Raw data	No-shift OVSS = 0000	1-bit shift OVSS = 0001	2-bit shift OVSS = 0010	3-bit shift OVSS = 0011	4-bit shift OVSS = 0100	5-bit shift OVSS = 0101	6-bit shift OVSS = 0110	7-bit shift OVSS = 0111	8-bit shift OVSS = 1000
2x	0x1FFE	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040	0x0020
4x	0x3FFC	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040
8x	0x7FF8	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080
16x	0xFFF0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100
32x	0x1FFE0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200
64x	0x3FFC0	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400
128x	0x7FF80	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800
256x	0xFFF00	0xFF00	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF

The conversion timings in oversampled mode do not change compared to standard conversion mode: the sample time is maintained equal during the whole oversampling sequence. New data are provided every N conversion, with an equivalent delay equal to N x t_{CONV} = N x (t_{SMPL} + t_{SAR}). The flags features are raised as following:

- the end of the sampling phase (EOSMP) is set after each sampling phase
- the end of conversion (EOC) occurs once every N conversions, when the oversampled result is available
- the end of sequence (EOCSEQ) occurs once the sequence of oversampled data is completed (i.e. after N x sequence length conversions total)

ADC operating modes supported when oversampling

In oversampling mode, most of the ADC operating modes are available:

- Single or continuous mode conversions, forward or backward scanned sequences and up to 8 channels programmed sequence
- ADC conversions start either by software or with triggers
- ADC stop during a conversion (abort)
- Data read via CPU or DMA with overrun detection
- Low-power modes (WAIT, AUTOFF)
- Programmable resolution: in this case, the reduced conversion values (as per RES[1:0] bits in ADC_CFGR1 register) are accumulated, truncated, rounded and shifted in the same way as 12-bit conversions are

Note: The alignment mode is not available when working with oversampled data. The ALIGN bit in ADC_CFGR1 is ignored and the data are always provided right-aligned.

Analog watchdog

The analog watchdog functionality is available (AWDxSGL, AWD1EN and AWDxCH bits), with the following differences:

- the RES[1:0] bits are ignored, comparison is always done on using the full 12-bits values HTx[11:0] and LTx[11:0]
- the comparison is performed on the most significant 12 bits of the 16 bits oversampled results ADC_DR[15:4]

Note: Care must be taken when using high shifting values. This reduces the comparison range. For instance, if the oversampled result is shifted by 4 bits thus yielding a 12-bit data right-aligned, the effective analog watchdog comparison can only be performed on 8 bits. The comparison is done between ADC_DR[11:4] and HTx[7:0] / LTx[[7:0], and HTx[11:8] / LTx[11:8] must be kept reset.

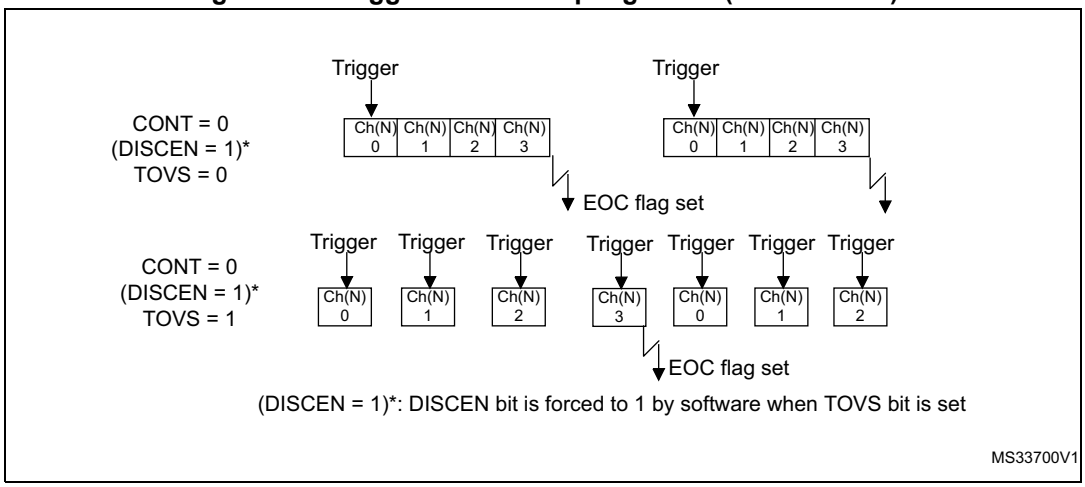
Triggered mode

The averager can also be used for basic filtering purposes. Although not a very efficient filter (slow roll-off and limited stop band attenuation), it can be used as a notch filter to reject constant parasitic frequencies (typically coming from the mains or from a switched mode power supply). For this purpose, a specific discontinuous mode can be enabled with TOVS bit in ADC_CFGR2, to be able to have an oversampling frequency defined by a user and independent from the conversion time itself.

Figure 214 below shows how conversions are started in response to triggers in discontinuous mode.

If the TOVS bit is set, the content of the DISCEN bit is ignored and considered as 1.

Figure 214. Triggered oversampling mode (TOVS bit = 1)



30.4.27 Temperature sensor and internal reference voltage

The temperature sensor can be used to measure the junction temperature (T_J) of the device. The temperature sensor is internally connected an ADC internal input channel which is used to convert the sensor's output voltage to a digital value. The sampling time for the temperature sensor analog pin must be greater than the minimum T_{S_temp} value specified in the datasheet. When not in use, the sensor can be put in Power-down mode.

The internal voltage reference (V_{REFINT}) provides a stable (bandgap) voltage output for the ADC and the comparators.

Refer to Table *ADC interconnection* in *Section 30.4.2: ADC pins and internal signals* for details on the ADC internal input channel to which the above voltages are connected.

Figure 215 shows the block diagram of connections between the temperature sensor, the internal voltage reference and the ADC.

The VSENSESEL bit must be set to enable the conversion of V_{SENSE} while VREFEN bit must be set to enable the conversion of V_{REFINT}.

When the ADC operates in Autonomous mode, these signals are controlled automatically to reduce power consumption (VSENSESEL and VREFEN must be set to measure the voltage in Autonomous mode).

The temperature sensor output voltage linearly changes with the 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 STMicroelectronics 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.

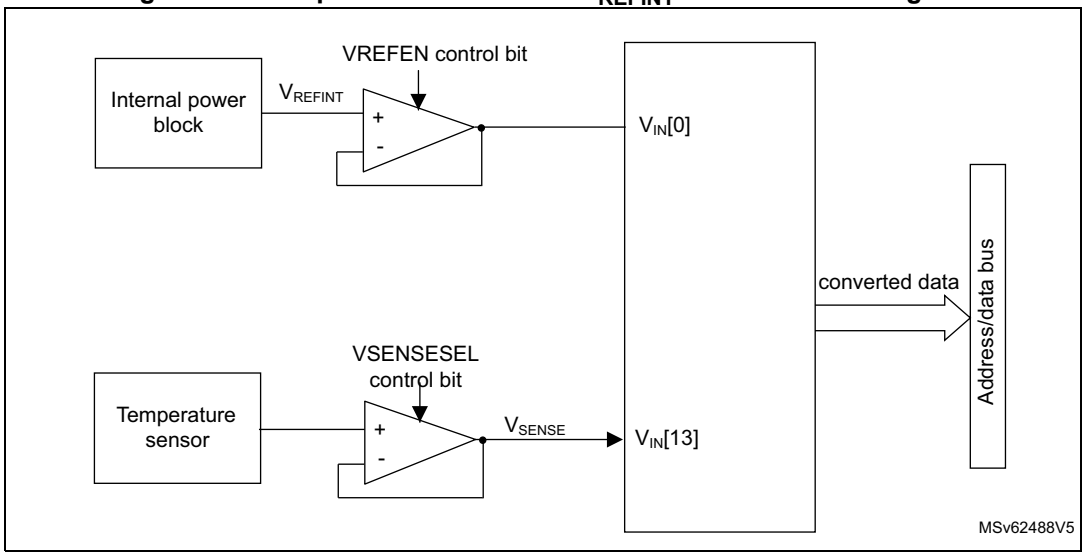
During the manufacturing process, the calibration data of the internal voltage reference are stored in the system memory area. The user application can then read them and use them

to improve the accuracy of the internal reference. Refer to the electrical characteristics section of the device datasheet for additional information.

Main features

- Linearity: ±2 °C max., precision depending on calibration

Figure 215. Temperature sensor and V_{REFINT} channel block diagram



Reading the temperature

1. Select the input channel connected to V_{SENSE} (refer to Table *ADC interconnection* in *Section 30.4.2: ADC pins and internal signals*).
2. Select an appropriate sampling time specified in the device datasheet (T_{S_temp}).
3. Set the VSENSESEL bit in the ADC_CCR register to wake up the temperature sensor from Power-down mode and wait for its stabilization time (t_{START}).
4. Start the ADC conversion by setting the ADSTART bit in the ADC_CR register (or by external trigger)
5. Read the resulting V_{SENSE} data in the ADC_DR register
6. Calculate the temperature using the following formula

$$\text{Temperature (in °C)} = \frac{\text{TS\_CAL2\_TEMP} - \text{TS\_CAL1\_TEMP}}{\text{TS\_CAL2} - \text{TS\_CAL1}} \times (\text{TS\_DATA} - \text{TS\_CAL1}) + \text{TS\_CAL1\_TEMP}$$

Where:

– TS_CAL2 is the temperature sensor calibration value acquired at TS_CAL2_TEMP.

– TS_CAL1 is the temperature sensor calibration value acquired at TS_CAL1_TEMP.

– TS_DATA is the actual temperature sensor output value converted by the ADC.

Refer to *Section 30.3: ADC implementation* for more information on TS_CAL1 and TS_CAL2 calibration points.

Note: The sensor has a startup time after waking up from Power-down mode before it can output V_{SENSE} at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADEN and VSENSESEL bits must be set at the same time.

Calculating the actual V_{REF+} voltage using the internal reference voltage

The V_{DDA} power supply voltage applied to the microcontroller may be subject to variation or not precisely known. The embedded internal voltage reference (VREFINT) and its calibration data acquired by the ADC during the manufacturing process at V_{REF+} = 3.0 V can be used to evaluate the actual V_{REF+} voltage level, if VREF+ pin is connected to a variable V_{DDA} power supply.

The following formula gives the actual V_{REF+} voltage supplying the device:

$$V_{REF+} = 3.0\ V \times VREFINT\_CAL / VREFINT\_DATA$$

Where:

- VREFINT_CAL is the VREFINT calibration value
- VREFINT_DATA is the actual VREFINT output value converted by ADC

Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between the voltage reference V_{REF+} and the voltage applied on the converted channel. For most application use cases, it is necessary to convert this ratio into a voltage independent from V_{REF+}. For applications where V_{REF+} is known and ADC converted values are right-aligned, the following formula can be used to calculate this absolute value:

$$V_{CHANNELx} = \frac{V_{REF+}}{FULL\_SCALE} \times ADC\_DATA_x$$

For applications where V_{REF+} value is not known, the internal voltage reference and V_{REF+} can be replaced by the expression provided in *Section : Calculating the actual V_{REF+} voltage using the internal reference voltage*, resulting in the following formula:

$$V_{CHANNELx} = \frac{3.0\ V \times VREFINT\_CAL \times ADC\_DATA_x}{VREFINT\_DATA \times FULL\_SCALE}$$

Where:

- VREFINT_CAL is the VREFINT calibration value (refer to *Section 30.3: ADC implementation* for the value of VREFINT_CAL).
- ADC_DATA_x is the value measured by the ADC on channelx (right-aligned).
- VREFINT_DATA is the actual VREFINT output value converted by the ADC.
- FULL_SCALE is the maximum digital value of the ADC output. For example with 12-bit resolution, it is 2^{12} - 1 = 4095 or with 8-bit resolution, 2^8 - 1 = 255.

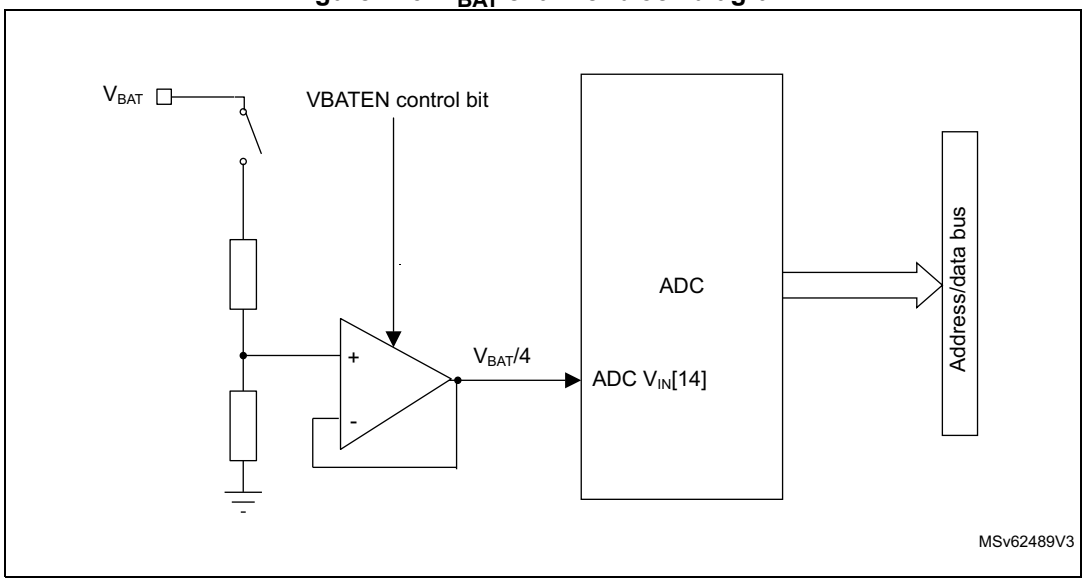
Note: If ADC measurements are done using an output format other than 12 bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.

30.4.28 Battery voltage monitoring

The VBATEN bit in the ADC_CCR register allows the application to measure the battery voltage on the VBAT pin. 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 4.

This bridge is automatically enabled when VBATEN is set, to connect V_{BAT}/4 to the corresponding ADC input channel (refer to Table ADC interconnection in *Section 30.4.2: ADC pins and internal signals*). As a consequence, the converted digital value is half the V_{BAT} voltage. To prevent any unwanted consumption on the battery, it is recommended to enable the bridge divider only during ADC conversion.

Figure 216. V_{BAT} channel block diagram



30.4.29 Concurrent operation with another ADC

When ADC4 is used simultaneously with another ADC (let us call it ADCx, x being different from 4), ADCx operation might generate noise on V_{REF+} voltage. Since V_{REF+} is also ADC4 reference voltage, this might cause conversion errors. To prevent this issue from happening, set VREFPROTEN bit in ADC_PWRR register. As soon as ADCx sampling phase starts, ADC4 is put on hold during one ADC4 clock cycle, thus resulting in ADC4 conversion time to be longer of one clock cycle.

In addition, ADCx might have two sampling phases during ADC4 conversion. This is due to the injected conversion function of ADCx. By setting VREFSECSMP to 1 in ADC_PWRR, ADC4 operation can be held twice during the conversion phase. When VREFSECSMP bit is set, ADC4 conversion time is longer of two clock cycles.

VREFSECSMP and VREFPROTEN bits must be set simultaneously.

ADCx and ADC4 must use the same clock source to be able to use the concurrent operation feature.

It is recommended to use this function when ADC4 works at a frequency greater or equal than ADCx.

30.5 ADC low-power modes

Table 260. Effect of low-power modes on the ADC

Mode	Description
Sleep	No effect, DMA requests are functional. ADC interrupts cause the device to exit Sleep mode.
Stop	The content of the ADC register is kept. ADC can be functional. DMA request are functional, and the interrupt cause the device to exit Stop mode.
Standby	The ADC peripheral is powered down and must be reinitialized after exiting Standby mode.

30.6 ADC interrupts

An interrupt can be generated by any of the following events:

- End of calibration (EOCAL flag)
- ADC power-up, when the ADC is ready (ADRDY flag)
- End of any conversion (EOC flag)
- End of a sequence of conversions (EOS flag)
- When an analog watchdog detection occurs (AWD1, AWD2, AWD3 flags)
- When the end of sampling phase occurs (EOSMP flag)
- when a data overrun occurs (OVR flag)
- LDO ready, when LDO output is stabilized (LDORDY flag)

Separate interrupt enable bits are available for flexibility.

Table 261. ADC wakeup and interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop mode(1)	Exit from Standby mode
ADC	LDO ready	LDORDY	LDORDYIE	Program LDORDY to 1	Yes	Yes	No
	End of calibration	EOCAL	EOCALIE	Program EOCAL to 1		Yes	
	ADC ready	ADRDY	ADRDYIE	Program ADRDY to 1		Yes	
	End of conversion	EOC	EOCIE	Program EOC to 1 and red ADC_DR		Yes	
	End of sequence of conversions	EOS	EOSIE	Program EOS to 1		Yes	
	Analog watchdog 1 status bit is set	AWD1	AWD1IE	Program AWD1 to 1		Yes	
	Analog watchdog 2 status bit is set	AWD2	AWD2IE	Program AWD2 to 1		Yes	
	Analog watchdog 3 status bit is set	AWD3	AWD3IE	Program AWD2 to 1		Yes	
	End of sampling phase	EOSMP	EOSMPIE	Program EOSMP to 1		No	
	Overrun	OVR	OVRIE	Program OVR to 1		Yes	

1. The ADC can wake up the device from Stop mode only if the peripheral instance supports the Wakeup from Stop mode feature. Refer to *Section 30.3: ADC implementation* for the list of supported Stop modes.

30.7 ADC registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

30.7.1 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.	LDO RDY	EOCAL	Res.	AWD3	AWD2	AWD1	Res.	Res.	OVR	EOS	EOC	EOSMP	ADRDY
			rc_w1	rc_w1		rc_w1	rc_w1	rc_w1			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LDORDY**: LDO ready
This bit is set by hardware. It indicates that the ADC internal LDO output is ready.
It is cleared by software by writing 1 to it.
0: ADC voltage regulator disabled
1: ADC voltage regulator enabled and stabilized

Bit 11 **EOCAL**: End of calibration flag
This bit is set by hardware when calibration is complete. It is cleared by software writing 1 to it.
0: Calibration is not complete
1: Calibration is complete

Bit 10 Reserved, must be kept at reset value.

Bit 9 **AWD3**: Analog watchdog 3 flag
This bit is set by hardware when the converted voltage crosses the values programmed in ADC_AWD3TR and ADC_AWD3TR registers. It is cleared by software by writing 1 to it.
0: No analog watchdog event occurred (or the flag event was already acknowledged and cleared by software)
1: Analog watchdog event occurred

Bit 8 **AWD2**: Analog watchdog 2 flag
This bit is set by hardware when the converted voltage crosses the values programmed in ADC_AWD2TR and ADC_AWD2TR registers. It is cleared by software writing 1 to it.
0: No analog watchdog event occurred (or the flag event was already acknowledged and cleared by software)
1: Analog watchdog event occurred

Bit 7 **AWD1**: Analog watchdog 1 flag
This bit is set by hardware when the converted voltage crosses the values programmed in ADC_TR1 and ADC_HR1 registers. It is cleared by software by writing 1 to it.
0: No analog watchdog event occurred (or the flag event was already acknowledged and cleared by software)
1: Analog watchdog event occurred

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **OVR**: ADC overrun

This bit is set by hardware when an overrun occurs, meaning that a new conversion has complete while the EOC flag was already set. It is cleared by software writing 1 to it.

0: No overrun occurred (or the flag event was already acknowledged and cleared by software)

1: Overrun has occurred

Bit 3 **EOS**: End of sequence flag

This bit is set by hardware at the end of the conversion of a sequence of channels selected by the CHSEL bits. It is cleared by software writing 1 to it.

0: Conversion sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Conversion sequence complete

Bit 2 **EOC**: End of conversion flag

This bit is set by hardware at the end of each conversion of a channel when a new data result is available in the ADC_DR register. It is cleared by software writing 1 to it or by reading the ADC_DR register.

0: Channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Channel conversion complete

Bit 1 **EOSMP**: End of sampling flag

This bit is set by hardware during the conversion, at the end of the sampling phase.It is cleared by software by writing 1 to it.

0: Not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)

1: End of sampling phase reached

Bit 0 **ADRDY**: ADC ready

This bit is set by hardware after the ADC has been enabled (ADEN = 1) and when the ADC reaches a state where it is ready to accept conversion requests.

It is cleared by software writing 1 to it.

0: ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)

1: ADC is ready to start conversion

30.7.2 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.	LDORDYIE	EOCALIE	Res.	AWD3IE	AWD2IE	AWD1IE	Res.	Res.	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
			rw	rw		rw	rw	rw			rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LDORDYIE**: LDO ready interrupt enable

This bit is set and cleared by software. It is used to enable/disable the LDORDY interrupt.

0: LDO ready interrupt disabled

1: LDO ready interrupt enabled. An interrupt is generated when the LDO output is ready.

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensure that no conversion is ongoing).

Bit 11 **EOCALIE**: End of calibration interrupt enable

This bit is set and cleared by software to enable/disable the end of calibration interrupt.

0: End of calibration interrupt disabled

1: End of calibration interrupt enabled

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog interrupt.

0: Analog watchdog interrupt disabled

1: Analog watchdog interrupt enabled

Note: The Software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 8 **AWD2IE**: Analog watchdog 2 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog interrupt.

0: Analog watchdog interrupt disabled

1: Analog watchdog interrupt enabled

Note: The Software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 7 **AWD1IE**: Analog watchdog 1 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog interrupt.

0: Analog watchdog interrupt disabled

1: Analog watchdog interrupt enabled

Note: The Software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **OVRIE**: Overrun interrupt enable

This bit is set and cleared by software to enable/disable the overrun interrupt.

0: Overrun interrupt disabled

1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 3 **EOSIE**: End of conversion sequence interrupt enable

This bit is set and cleared by software to enable/disable the end of sequence of conversions interrupt.

0: EOS interrupt disabled

1: EOS interrupt enabled. An interrupt is generated when the EOS bit is set.

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 2 **EOCIE**: End of conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of conversion interrupt.

0: EOC interrupt disabled

1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 1 **EOSMPIE**: End of sampling flag interrupt enable

This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt.

0: EOSMP interrupt disabled.

1: EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 0 **ADRDYIE**: ADC ready interrupt enable

This bit is set and cleared by software to enable/disable the ADC Ready interrupt.

0: ADRDY interrupt disabled.

1: ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

30.7.3 ADC control register (ADC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	Res.	Res.	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs			rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADSTP	Res.	ADSTART	ADDIS	ADEN
											rs		rs	rs	rs

Bit 31 **ADCAL**: ADC calibration

This bit is set by software to start the calibration of the ADC.

It is cleared by hardware after calibration is complete.

0: Calibration complete

1: Write 1 to calibrate the ADC. Read at 1 means that a calibration is in progress.

Note: The software is allowed to set ADCAL only when the ADC is disabled (ADCAL = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0, AUTOFF = 0, and ADEN = 0).

The software is allowed to update the calibration factor by writing ADC_CALFACT only when ADEN is set to 1 and ADSTART is cleared to 0 by writing ADSTP to 1 (ADC enabled and no conversion is ongoing).

Bits 30:29 Reserved, must be kept at reset value.

Bit 28 **ADVREGEN**: ADC voltage regulator enable

This bit is set by software, to enable the ADC internal voltage regulator. The voltage regulator output is available after $t_{ADCVREG\_SETUP}$.

It is cleared by software to disable the voltage regulator. It can be cleared only if ADEN is et to 0.

0: ADC voltage regulator disabled

1: ADC voltage regulator enabled

Note: The software is allowed to program this bit field only when the ADC is disabled (ADCAL = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 27:5 Reserved, must be kept at reset value.

Bit 4 **ADSTP**: ADC stop conversion command

This bit is set by software to stop and discard an ongoing conversion (ADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC is ready to accept a new start conversion command.

0: No ADC stop conversion command ongoing

1: Write 1 to stop the ADC. Read 1 means that an ADSTP command is in progress.

Note: To clear the A/D converter state, ADSTP must be set to 1 even if ADSTART is cleared to 0 after the software trigger A/D conversion. It is recommended to set ADSTP to 1 whenever the configuration needs to be modified.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **ADSTART**: ADC start conversion command

This bit is set by software to start ADC conversion. Depending on the EXTEN [1:0] configuration bits, a conversion either starts immediately (software trigger configuration) or once a hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

– In single conversion mode (CONT = 0, DISCEN = 0), when software trigger is selected (EXTEN = 00): at the assertion of the end of Conversion Sequence (EOS) flag.

– In discontinuous conversion mode(CONT=0, DISCEN = 1), when the software trigger is selected (EXTEN = 00): at the assertion of the end of Conversion (EOC) flag.

– In all other cases: after the execution of the ADSTP command, at the same time as the ADSTP bit is cleared by hardware.

0: No ADC conversion is ongoing.

1: Write 1 to start the ADC. Read 1 means that the ADC is operating and may be converting.

Note: The software is allowed to set ADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable the ADC).

Bit 1 **ADDIS**: ADC disable command

This bit is set by software to disable the ADC (ADDIS command) and put it into power-down state (OFF state).

It is cleared by hardware once the ADC is effectively disabled (ADEN is also cleared by hardware at this time).

0: No ADDIS command ongoing

1: Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress.

Note: Setting ADDIS to 1 is only effective when ADEN = 1 and ADSTART = 0 (which ensures that no conversion is ongoing)

Bit 0 **ADEN**: ADC enable command

This bit is set by software to enable the ADC. The ADC is effectively ready to operate once the ADRDY flag has been set.

It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.

0: ADC is disabled (OFF state)

1: Write 1 to enable the ADC.

Note: The software is allowed to set ADEN only when all bits of ADC_CR registers are 0 (ADCAL = 0, ADSTP = 0, ADSTART = 0, ADDIS = 0 and ADEN = 0)

30.7.4 ADC configuration register 1 (ADC_CFGR1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AWD1CH[4:0]					Res.	Res.	AWD1EN	AWD1SGL	CHSELRMOD	Res.	Res.	Res.	Res.	DISCEN
	rw	rw	rw	rw	rw			rw	rw	rw					rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	WAIT	CONT	OVRMOD	EXTEN[1:0]		Res.	EXTSEL[2:0]			ALIGN	SCANDIR	RES[1:0]		DMACFG	DMAEN
	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:26 **AWD1CH[4:0]**: Analog watchdog channel selection

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

00000: ADC analog input Channel 0 monitored by AWD

00001: ADC analog input Channel 1 monitored by AWD

.....

10001: ADC analog input Channel 17 monitored by AWD

10110: ADC analog input Channel 22 monitored by AWD

10111: ADC analog input Channel 23 monitored by AWD

Others: Reserved

Note: The channel selected by the AWDCH[4:0] bits must be also set into the CHSELR register. The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bits 25:24 Reserved, must be kept at reset value.

Bit 23 **AWD1EN**: Analog watchdog enable

This bit is set and cleared by software.

0: Analog watchdog **1** disabled

1: Analog watchdog **1** enabled

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 22 **AWD1SGL**: Enable the watchdog on a single channel or on all channels

This bit is set and cleared by software to enable the analog watchdog on the channel identified by the AWDCH[4:0] bits or on all the channels

0: Analog watchdog **1** enabled on all channels

1: Analog watchdog **1** enabled on a single channel

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 21 **CHSELRMOD**: Mode selection of the ADC_CHSELR register

This bit is set and cleared by software to control the ADC_CHSELR feature:

0: Each bit of the ADC_CHSELR register enables an input

1: ADC_CHSELR register is able to sequence up to 8 channels

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bits 20:17 Reserved, must be kept at reset value.

Bit 16 **DISCEN**: Discontinuous mode

This bit is set and cleared by software to enable/disable discontinuous mode.

0: Discontinuous mode disabled

1: Discontinuous mode enabled

Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN = 1 and CONT = 1.

The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **WAIT**: Wait conversion mode

This bit is set and cleared by software to enable/disable wait conversion mode.

0: Wait conversion mode off

1: Wait conversion mode on

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 13 **CONT**: Single / continuous conversion mode

This bit is set and cleared by software. If it is set, conversion takes place continuously until it is cleared.

0: Single conversion mode

1: Continuous conversion mode

Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN = 1 and CONT = 1.

The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 12 **OVRMOD**: Overrun management mode

This bit is set and cleared by software and configure the way data overruns are managed.

0: ADC_DR register is preserved with the old data when an overrun is detected.

1: ADC_DR register is overwritten with the last conversion result when an overrun is detected.

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection

These bits are set and cleared by software to select the external trigger polarity and enable the trigger.

00: Hardware trigger detection disabled (conversions can be started by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 9 Reserved, must be kept at reset value.

Bits 8:6 **EXTSEL[2:0]**: External trigger selection

These bits select the external event used to trigger the start of conversion (refer to table *ADC interconnection* in *Section 30.4.2: ADC pins and internal signals* for details):

000: adc_trg0
001: adc_trg1
010: adc_trg2
011: adc_trg3
100: adc_trg4
101: adc_trg5
110: adc_trg6
111: adc_trg7

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 5 **ALIGN**: Data alignment

This bit is set and cleared by software to select right or left alignment. Refer to *Figure 200: Data alignment and resolution (oversampling disabled: OVSE = 0) on page 1105*

0: Right alignment
1: Left alignment

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 4 **SCANDIR**: Scan sequence direction

This bit is set and cleared by software to select the direction in which the channels is scanned in the sequence. It is effective only if CHSELRMOD bit is cleared to 0.

0: Upward scan (from CHSEL0 to CHSEL23)
1: Backward scan (from CHSEL23 to CHSEL0)

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bits 3:2 **RES[1:0]**: Data resolution

These bits are written by software to select the resolution of the conversion.

00: 12 bits
01: 10 bits
10: 8 bits
11: 6 bits

Note: The software is allowed to write these bits only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

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 converted data using the DMA on page 1106*

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bit 0 **DMAEN**: Direct memory access enable

This bit is set and cleared by software to enable the generation of DMA requests. This allows the automatic management of the converted data by the DMA controller. For more details, refer to *Section : Managing converted data using the DMA on page 1106*.

0: DMA disabled

1: DMA enabled

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

30.7.5 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.	LFTRIG	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.	TOVS	OVSS[3:0]				OVSR[2:0]			Res.	OVSE
						rw	rw	rw	rw	rw	rw	rw	rw		rw

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

Bit 29 **LFTRIG**: Low frequency trigger mode enable

This bit must be set by software.

0: Reserved

1: Low frequency trigger mode enabled.

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 (this ensures that no conversion is ongoing).

Bits 28:10 Reserved, must be kept at reset value.

Bit 9 **TOVS**: Triggered *Oversampling*

This bit is set and cleared by software.

0: All oversampled conversions for a channel are done consecutively after a trigger

1: Each oversampled conversion for a channel needs a trigger

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bits 8:5 **OVSS[3:0]**: *Oversampling shift*

This bit is set and cleared by software.

0000: No shift

0001: Shift 1-bit

0010: Shift 2-bits

0011: Shift 3-bits

0100: Shift 4-bits

0101: Shift 5-bits

0110: Shift 6-bits

0111: Shift 7-bits

1000: Shift 8-bits

Others: Reserved

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1(which ensures that no conversion is ongoing).

Bits 4:2 **OVSR[2:0]**: *Oversampling ratio*

This bit filed defines the number of oversampling ratio.

000: 2x

001: 4x

010: 8x

011: 16x

100: 32x

101: 64x

110: 128x

111: 256x

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bit 1 Reserved, must be kept at reset value.

Bit 0 **OVSE**: *Oversampler Enable*

This bit is set and cleared by software.

0: Oversampler disabled

1: Oversampler enabled

Note: Software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

30.7.6 ADC sampling time register (ADC_SMPR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SMPSEL23	SMPSEL22	SMPSEL21	SMPSEL20	SMPSEL19	SMPSEL18	SMPSEL17	SMPSEL16	SMPSEL15	SMPSEL14	SMPSEL13	SMPSEL12	SMPSEL11	SMPSEL10	SMPSEL9	SMPSEL8
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
SMPSEL7	SMPSEL6	SMPSEL5	SMPSEL4	SMPSEL3	SMPSEL2	SMPSEL1	SMPSEL0	Res.	SMP2[2:0]			Res.	SMP1[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:8 **SMPSELx:** Channel-x sampling time selection (x = 23 to 0)

These bits are written by software to define which sampling time is used.

0: Sampling time of CHANNELx use the setting of SMP1[2:0] register.

1: Sampling time of CHANNELx use the setting of SMP2[2:0] register.

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **SMP2[2:0]:** Sampling time selection 2

These bits are written by software to select the sampling time that applies to all channels.

000: 1.5 ADC clock cycles

001: 3.5 ADC clock cycles

010: 7.5 ADC clock cycles

011: 12.5 ADC clock cycles

100: 19.5 ADC clock cycles

101: 39.5 ADC clock cycles

110: 79.5 ADC clock cycles

111: 814.5 ADC clock cycles

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SMP1[2:0]:** *Sampling time selection 1*

These bits are written by software to select the sampling time that applies to all channels.

000: 1.5 ADC clock cycles

001: 3.5 ADC clock cycles

010: 7.5 ADC clock cycles

011: 12.5 ADC clock cycles

100: 19.5 ADC clock cycles

101: 39.5 ADC clock cycles

110: 79.5 ADC clock cycles

111: 814.5 ADC clock cycles

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

30.7.7 ADC watchdog threshold register (ADC_AWD1TR)

Address offset: 0x20

Reset value: 0x0FFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HT1[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LT1[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 27:16 **HT1[11:0]**: Analog watchdog 1 higher threshold
These bits are written by software to define the higher threshold for the analog watchdog.
Refer to *Section 30.4.25: Analog window watchdog on page 1112*.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **LT1[11:0]**: Analog watchdog *1* lower threshold
These bits are written by software to define the lower threshold for the analog watchdog.
Refer to *Section 30.4.25: Analog window watchdog on page 1112*.

30.7.8 ADC watchdog threshold register (ADC_AWD2TR)

Address offset: 0x24

Reset value: 0x0FFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HT2[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LT2[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 27:16 **HT2[11:0]**: Analog watchdog 2 higher threshold
These bits are written by software to define the higher threshold for the analog watchdog.
Refer to *Section 30.4.25: Analog window watchdog on page 1112*.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **LT2[11:0]**: Analog watchdog 2 lower threshold
These bits are written by software to define the lower threshold for the analog watchdog.
Refer to *Section 30.4.25: Analog window watchdog on page 1112*.

30.7.9 ADC channel selection register [alternate] (ADC_CHSELR)

Address offset: 0x28

Reset value: 0x0000 0000

The same register can be used in two different modes:

– Each ADC_CHSELR bit enables an input (CHSELRMOD = 0 in ADC_CFGR1). Refer to the current section.

– ADC_CHSELR is able to sequence up to 8 channels (CHSELRMOD = 1 in ADC_CFGR1). Refer to next section.

CHSELRMOD = 0 in ADC_CFGR1

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

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

Bits 23:0 **CHSELx**: Channel x selection (x = 23 to 0)

These bits are written by software and define which channels are part of the sequence of channels to be converted.

0: Input Channel-x is not selected for conversion

1: Input Channel-x is selected for conversion

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

30.7.10 ADC channel selection register [alternate] (ADC_CHSELR)

Address offset: 0x28

Reset value: 0x0000 0000

The same register can be used in two different modes:

– Each ADC_CHSELR bit enables an input (CHSELRMOD = 0 in ADC_CFGR1). Refer to the current previous section.

– ADC_CHSELR is able to sequence up to 8 channels (CHSELRMOD = 1 in ADC_CFGR1). Refer to this section.

CHSELRMOD = 1 in ADC_CFGR1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SQ8[3:0]				SQ7[3:0]				SQ6[3:0]				SQ5[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ4[3:0]				SQ3[3:0]				SQ2[3:0]				SQ1[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **SQ8[3:0]**: 8th conversion of the sequence

These bits are programmed by software with the channel number assigned to the 8th conversion of the sequence. 0b1111 indicates the end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

0000: CH0

0001: CH1

...

1010: CH10

1011: CH11

1100: CH12

1101: CH13

1110: CH14

1111: No channel selected (End of sequence)

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bits 27:24 **SQ7[3:0]**: 7th conversion of the sequence

These bits are programmed by software with the channel number assigned to the 7th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bits 23:20 **SQ6[3:0]**: 6th conversion of the sequence

These bits are programmed by software with the channel number assigned to the 6th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bits 19:16 **SQ5[3:0]**: 5th conversion of the sequence

These bits are programmed by software with the channel number assigned to the 5th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bits 15:12 **SQ4[3:0]**: 4th conversion of the sequence

These bits are programmed by software with the channel number assigned to the 4th conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1(which ensures that no conversion is ongoing).

Bits 11:8 **SQ3[3:0]**: 3rd conversion of the sequence

These bits are programmed by software with the channel number assigned to the 3rd conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bits 7:4 **SQ2[3:0]**: 2nd conversion of the sequence

These bits are programmed by software with the channel number assigned to the 2nd conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bits 3:0 **SQ1[3:0]**: 1st conversion of the sequence

These bits are programmed by software with the channel number assigned to the 1st conversion of the sequence. 0b1111 indicates end of the sequence.

When 0b1111 (end of sequence) is programmed to the lower sequence channels, these bits are ignored.

Refer to SQ8[3:0] for a definition of channel selection.

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

30.7.11 ADC watchdog threshold register (ADC_AWD3TR)

Address offset: 0x2C

Reset value: 0x0FFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HT3[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LT3[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 27:16 **HT3[11:0]**: *Analog watchdog 3 higher threshold*

These bits are written by software to define the higher threshold for the analog watchdog.

Refer to *Section 30.4.25: Analog window watchdog on page 1112*.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **LT3[11:0]**: Analog watchdog *3*lower threshold

These bits are written by software to define the lower threshold for the analog watchdog.

Refer to *Section 30.4.25: Analog window watchdog on page 1112*.

30.7.12 ADC data register (ADC_DR)

Address offset: 0x40

Reset value: 0x0000 0000

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

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DATA[15:0]**: Converted data

These bits are read-only. They contain the conversion result from the last converted channel. The data are left- or right-aligned as shown in *Figure 200: Data alignment and resolution (oversampling disabled: OVSE = 0) on page 1105*.

Just after a calibration is complete, DATA[6:0] contains the calibration factor.

30.7.13 ADC power register (ADC_PWRR)

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.	Res.	Res.	Res.	Res.	Res.	Res.	VREFSECSMP	VREFPROT	DPD	AUTOFF
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **VREFSECSMP**: V_{REF+} second sample bit

This bit is set and cleared by software. It is used to enable/disable the second V_{REF+} protection when multiple ADCs are working simultaneously and a clock divider of 1 is used.

0: V_{REF+} second sample disabled

1: V_{REF+} second sample enabled

Note: The software is allowed to write this bit only when ADEN bit is cleared to 0 (this ensures that no conversion is ongoing).

Bit 2 **VREFPROT**: V_{REF+} protection bit

This bit is set and cleared by software. It is used to enable/disable V_{REF+} protection when multiple ADCs are working simultaneously and a clock divider is used.

0: V_{REF+} protection disabled

1: V_{REF+} protection enabled

Note: The software is allowed to write this bit only when ADEN bit is cleared to 0 (this ensures that no conversion is ongoing).

Bit 1 **DPD**: Deep-power-down mode bit

This bit is set and cleared by software. It is used to enable/disable Deep-power-down mode in Autonomous mode when the ADC is not used.

0: Deep-power-down mode disabled

1: Deep-power-down mode enabled

Note: The software is allowed to write this bit only when ADEN bit is cleared to 0 (this ensures that no conversion is ongoing).

Setting DPD in Auto-off mode automatically disables the LDO.

Bit 0 **AUTOFF**: Auto-off mode bit

This bit is set and cleared by software. it is used to enable/disable the Auto-off mode.

0: Auto-off mode disabled

1: Auto-off mode enabled

Note: The software is allowed to write this bit only when ADEN bit is cleared to 0 (this ensures that no conversion is ongoing).

30.7.14 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.	AWD2 CH23	AWD2 CH22	AWD2 CH21	AWD2 CH20	AWD2 CH19	AWD2 CH18	AWD2 CH17	AWD2 CH16
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD2 CH15	AWD2 CH14	AWD2 CH13	AWD2 CH12	AWD2 CH11	AWD2 CH10	AWD2 CH9	AWD2 CH8	AWD2 CH7	AWD2 CH6	AWD2 CH5	AWD2 CH4	AWD2 CH3	AWD2 CH2	AWD2 CH1	AWD2 CH0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:0 **AWD2CHx:** Analog watchdog channel selection (x = 23 to 0)

These bits are set and cleared by software. They enable and select the input channels to be guarded by analog watchdog 2 (AWD2).

0: ADC analog channel-x is not monitored by AWD2

1: ADC analog channel-x is monitored by AWD2

Note: The channels selected through ADC_AWD2CR must be also configured into the ADC_CHSELR registers. Refer to SQ8[3:0] for a definition of channel selection. The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

30.7.15 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.	AWD3 CH23	AWD3 CH22	AWD3 CH21	AWD3 CH20	AWD3 CH19	AWD3 CH18	AWD3 CH17	AWD3 CH16
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD3 CH15	AWD3 CH14	AWD3 CH13	AWD3 CH12	AWD3 CH11	AWD3 CH10	AWD3 CH9	AWD3 CH8	AWD3 CH7	AWD3 CH6	AWD3 CH5	AWD3 CH4	AWD3 CH3	AWD3 CH2	AWD3 CH1	AWD3 CH0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:0 **AWD3CHx:** Analog watchdog channel selection (x = 23 to 0)

These bits are set and cleared by software. They enable and select the input channels to be guarded by analog watchdog 3 (AWD3).

0: ADC analog channel-x is not monitored by AWD3

1: ADC analog channel-x is monitored by AWD3

Note: The channels selected through ADC_AWD3CR must be also configured into the ADC_CHSELR registers. Refer to SQ8[3:0] for a definition of channel selection. The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

30.7.16 ADC Calibration factor (ADC_CALFACT)

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.	CALFACT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **CALFACT[6:0]**: Calibration factor

These bits are written by hardware or by software.

– Once a calibration is complete, they are updated by hardware with the calibration factors.

– Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it is then applied once a new calibration is launched.

– Just after a calibration is complete, DATA[6:0] contains the calibration factor.

Note: Software can write these bits only when ADEN = 1 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

30.7.17 ADC option register (ADC_OR)

Address offset: 0xD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	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.	CHN21 SEL
															rw

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

Bit 0 **CHN21SEL**: Channel 21 selection bit

This bit is set and cleared by software. It is used to select the internal source connected to ADC input channel 21:

0: dac1_out1 selected

1: dac1_out2 selected

Note: The software is allowed to write this bit only when ADSTART bit is cleared to 0 by writing ADSTP to 1 = 0 (which ensures that no conversion is ongoing).

30.7.18 ADC common configuration register (ADC_CCR)

Address offset: 0x308

Reset value: 0x0000 0000

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

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **VBATEN**: V_{BAT} enable

This bit is set and cleared by software to enable/disable the V_{BAT} channel.

0: V_{BAT} channel disabled

1: V_{BAT} channel enabled

Note: The software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing)

Bit 23 **VSENSESEL**: Temperature sensor selection

This bit is set and cleared by software to enable/disable the temperature sensor.

0: Temperature sensor disabled

1: Temperature sensor enabled

Note: Software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bit 22 **VREFEN**: V_{REFINT} enable

This bit is set and cleared by software to enable/disable the V_{REFINT} buffer.

0: V_{REFINT} disabled

1: V_{REFINT} enabled

Note: Software is allowed to write this bit only when ADSTART is cleared to 0 by writing ADSTP to 1 (which ensures that no conversion is ongoing).

Bits 21:18 **PRESC[3:0]:** ADC prescaler

Set and cleared by software to select the frequency of the clock to the ADC.

0000: input ADC clock not divided
0001: input ADC clock divided by 2
0010: input ADC clock divided by 4
0011: input ADC clock divided by 6
0100: input ADC clock divided by 8
0101: input ADC clock divided by 10
0110: input ADC clock divided by 12
0111: input ADC clock divided by 16
1000: input ADC clock divided by 32
1001: input ADC clock divided by 64
1010: input ADC clock divided by 128
1011: input ADC clock divided by 256
Other: Reserved

Note: Software is allowed to write these bits only when the ADC is disabled (ADCAL = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 17:0 Reserved, must be kept at reset value.

30.8 ADC register map

The following table summarizes the ADC registers.

Table 262. ADC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**ADC_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LDORDY	EOCAL	Res.	AWD3	AWD2	AWD1	Res.	Res.	OVR	EOS	EOC	EOSMP	ADRDY
	Reset value																				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.	LDORDYIE	EOCALIE	Res.	AWD3IE	AWD2IE	AWD1IE	Res.	Res.	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
	Reset value																				0	0		0	0	0			0	0	0	0	0
0x08	**ADC_CR**	ADCAL	Res.	Res.	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADSTP	Res.	ADSTART	ADDIS	ADEN
	Reset value	0			0																								0		0	0	0
0x0C	**ADC_CFGR1**	Res.	AWDCH[4:0]					Res.	Res.	AWD1EN	AWD1SGL	CHSELRMOD	Res.	Res.	Res.	Res.	DISCEN	Res.	WAIT	CONT	OVRMOD	EXTEN[1:0]		Res.	EXTSEL[2:0]			ALIGN	SCANDIR	RES[1:0]		DMACFG	DMAEN
	Reset value		0	0	0	0	0			0	0	0					0		0	0	0	0	0		0	0	0	0	0	0	0	0	0
0x10	**ADC_CFGR2**	Res.	Res.	LFTRIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOVS	OVSS[3:0]				OVSR [2:0]			Res.	OVSE
	Reset value			0																				0	0	0	0	0	0	0	0		0
0x14	**ADC_SMPR**	SMPSEL23.	SMPSEL22	SMPSEL21.	SMPSEL20.	SMPSEL19.	SMPSEL18	SMPSEL17	SMPSEL16	SMPSEL15	SMPSEL14	SMPSEL13	SMPSEL12	SMPSEL11	SMPSEL10	SMPSEL9	SMPSEL8	SMPSEL7	SMPSEL6	SMPSEL5	SMPSEL4	SMPSEL3	SMPSEL2	SMPSEL1	SMPSEL0	Res.	SMP2 [2:0]			Res.	SMP1 [2:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0		0	0	0
0x18-0x1C	Reserved	Reserved																															
0x20	**ADC_AWD1TR**	Res.	Res.	Res.	Res.	HT1[11:0]												Res.	Res.	Res.	Res.	LT1[11:0]											
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x24	**ADC_AWD2TR**	Res.	Res.	Res.	Res.	HT2[11:0]												Res.	Res.	Res.	Res.	LT2[11:0]											
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x28	**ADC_CHSELR** (CHSELRMOD = 0)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHSEL23	CHSEL22	CHSEL21	CHSEL20	CHSEL19	CHSEL18	CHSEL17	CHSEL16	CHSEL15	CHSEL14	CHSEL13	CHSEL12	CHSEL11	CHSEL10	CHSEL9	CHSEL8	CHSEL7	CHSEL6	CHSEL5	CHSEL4	CHSEL3	CHSEL2	CHSEL1	CHSEL0
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**ADC_CHSELR** (CHSELRMOD = 1)	SQ8[3:0]				SQ7[3:0]				SQ6[3:0]				SQ5[3:0]				SQ4[3:0]				SQ3[3:0]				SQ2[3:0]				SQ1[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**ADC_AWD3TR**	Res.	Res.	Res.	Res.	HT3[11:0]												Res.	Res.	Res.	Res.	LT3[11:0]											
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1					0	0	0	0	0	0	0	0	0	0	0	0
0x30 - 0x3C	Reserved	Reserved																															
0x40	**ADC_DR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 262. ADC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x44	**ADC_PWRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VREFSECSMP	VREFPROT	DPD	AUTOFF
	Reset value																													0	0	0	0
0x48 - 0xBF	Reserved	Reserved																															
0xA0	**ADC_AWD2CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD2CH23	AWD2CH22	AWD2CH21	AWD2CH20	AWD2CH19	AWD2CH18	AWD2CH17	AWD2CH16	AWD2CH15	AWD2CH14	AWD2CH13	AWD2CH12	AWD2CH11	AWD2CH10	AWD2CH9	AWD2CH8	AWD2CH7	AWD2CH6	AWD2CH5	AWD2CH4	AWD2CH3	AWD2CH2	AWD2CH1	AWD2CH0
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA4	**ADC_AWD3CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD3CH23	AWD3CH22	AWD3CH21	AWD3CH20	AWD3CH19	AWD3CH18	AWD3CH17	AWD3CH16	AWD3CH15	AWD3CH14	AWD3CH13	AWD3CH12	AWD3CH11	AWD3CH10	AWD3CH9	AWD3CH8	AWD3CH7	AWD3CH6	AWD3CH5	AWD3CH4	AWD3CH3	AWD3CH2	AWD3CH1	AWD3CH0
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA4 - 0xC0	Reserved	Reserved																															
0xC4	**ADC_CALFACT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT[6:0]						
	Reset value																										0	0	0	0	0	0	0
0xB8 - 0xCF	Reserved	Reserved																															
0xD0	**ADC_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.	Res.	CHN21SEL
	Reset value																																0
0x308	**ADC_CCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBATEN	VSENSESEL	VREFEN	PRESC3	PRESC2	PRESC1	PRESC0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0	0	0	0	0	0	0																		

Refer to *Section 2.3 on page 115* for the register boundary addresses.

31 Digital-to-analog converter (DAC)

31.1 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 obtain a high drive output current. An individual calibration can be applied on each DAC output channel. The DAC output channels support a low power mode, the Sample and hold mode.

31.2 DAC main features

The DAC main features are the following (see *Figure 217: 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
- Double data DMA capability to reduce the bus activity
- External triggers for conversion
- DAC output channel buffered/unbuffered modes
- Buffer offset calibration
- Each DAC output can be disconnected from the DACx_OUTy output pin
- DAC output connection to on-chip peripherals
- Sample and hold mode for low power operation in Stop mode
- Autonomous mode to reduce the power consumption for the system
- Input voltage reference from V_{REF+} pin or internal VREFBUF reference

Figure 217 shows the block diagram of a DAC channel and *Table 264* gives the pin description.

31.3 DAC implementation

Table 263. DAC features

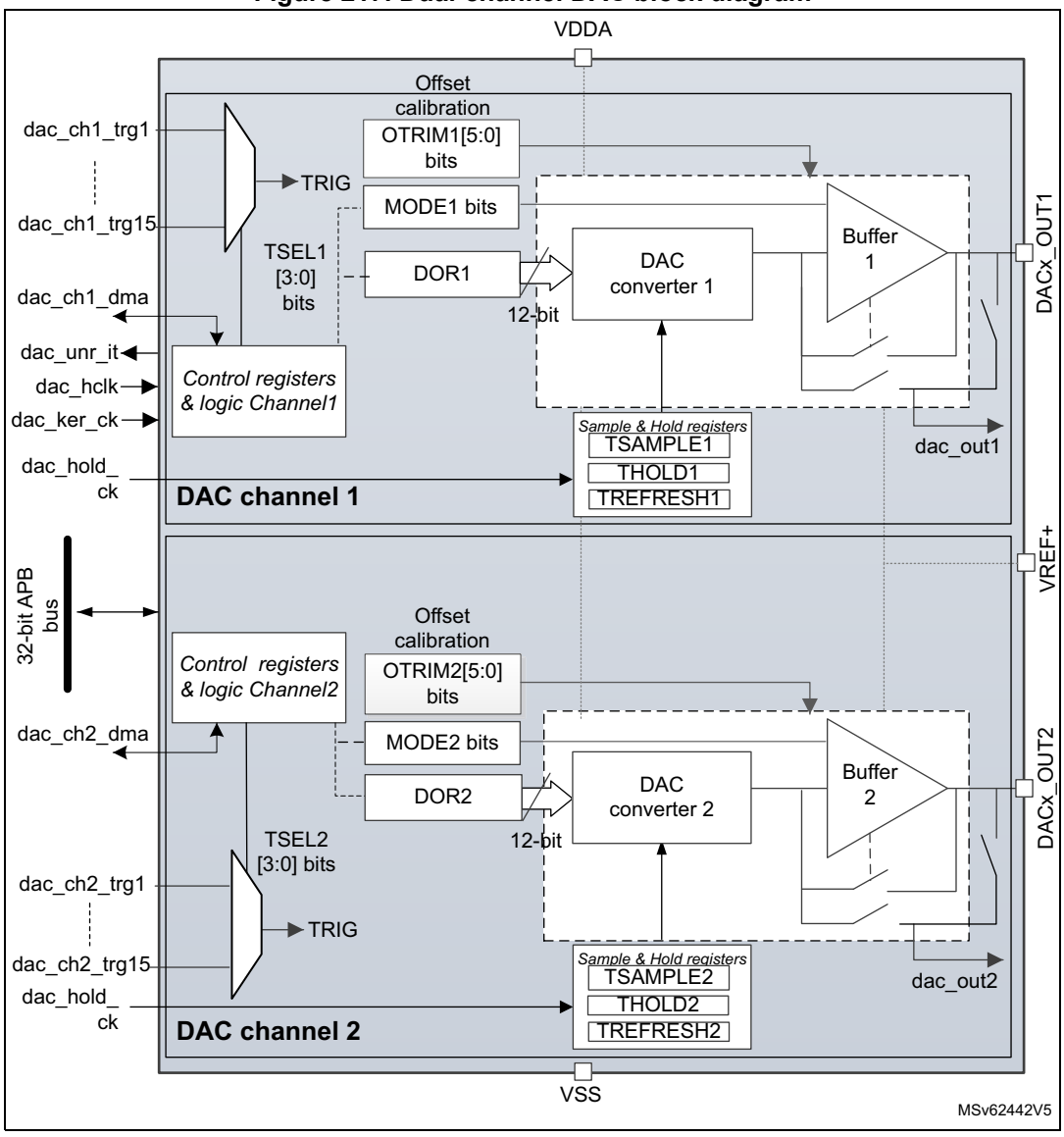
DAC features	DAC1
Dual channel	X
Output buffer	X
I/O connection	DAC1_OUT1 on PA4, DAC1_OUT2 on PA5
Maximum sampling time	1 MSPS
Autonomous mode(1)	X
VREF+ pin	X

1. The Autonomous mode is supported in Stop 0, Stop 1 and Stop 2 modes.

31.4 DAC functional description

31.4.1 DAC block diagram

Figure 217. Dual-channel DAC block diagram



1. MODEx bits in the DAC_MCR control the output mode and the switching between the Normal mode in buffer/unbuffered configuration and the Sample and hold mode.
2. Refer to *Section 31.3: DAC implementation* for channel2 availability.

31.4.2 DAC pins and internal signals

The DAC includes:

- Up to two output channels
- The DACx_OUTy can be disconnected from the output pin and used as an ordinary GPIO
- The dac_outx can use an internal pin connection to on-chip peripherals such as comparator, operational amplifier and ADC (if available).
- DAC output channel buffered or non buffered
- Sample and hold block and registers operational in Stop mode, using the LSI/LSE clock source (dac_hold_ck) for static conversion.

The DAC includes up to two separate output channels. Each output channel can be connected to on-chip peripherals such as comparator, operational amplifier and ADC (if available). In this case, the DAC output channel can be disconnected from the DACx_OUTy output pin and the corresponding GPIO can be used for another purpose.

The DAC output can be buffered or not. The Sample and hold block and its associated registers can run in Stop mode using the LSI or LSE clock source (dac_hold_ck).

Table 264. DAC input/output pins

Pin name	Signal type	Remarks
VREF+	Input, analog reference positive	The higher/positive reference voltage for the DAC, $V_{REF+} \leq V_{DDAmax}$ (refer to datasheet)
VDDA	Input, analog supply	Analog power supply
VSSA	Input, analog supply ground	Ground for analog power supply
DACx_OUTy	Analog output signal	DACx channely analog output

Table 265. DAC internal input/output signals

Internal signal name	Signal type	Description
dac_ch1_dma	Bidirectional	DAC channel1 DMA request/acknowledge
dac_ch2_dma	Bidirectional	DAC channel2 DMA request/acknowledge
dac_ch1_trgx (x = 1 to 15)	Inputs	DAC channel1 trigger inputs
dac_ch2_trgx (x = 1 to 15)	Inputs	DAC channel2 trigger inputs
dac_unr_it	Output	DAC underrun interrupt
dac_hclk	Input	DAC peripheral clock
dac_ker_ck	Input	DAC kernel clock
dac_hold_ck	Input	DAC low-power clock used in Sample and hold mode
dac_out1	Analog output	DAC channel1 output for on-chip peripherals
dac_out2	Analog output	DAC channel2 output for on-chip peripherals

Table 266. DAC interconnection

Signal name	Source	Source type
dac_hold_ck	ck_lsi or ck_lse	LSI or LSE clock selected in the RCC
dac_chx_trg1 (x = 1, 2)	tim1_trgo	Internal signal from on-chip timers
dac_chx_trg2 (x = 1, 2)	tim2_trgo	Internal signal from on-chip timers
dac_chx_trg3 (x = 1, 2)	tim4_trgo	Internal signal from on-chip timers
dac_chx_trg4 (x = 1, 2)	tim5_trgo	Internal signal from on-chip timers
dac_chx_trg5 (x = 1, 2)	tim6_trgo	Internal signal from on-chip timers
dac_chx_trg6 (x = 1, 2)	tim7_trgo	Internal signal from on-chip timers
dac_chx_trg7 (x = 1, 2)	tim8_trgo	Internal signal from on-chip timers
dac_chx_trg8 (x = 1, 2)	tim15_trgo	Internal signal from on-chip timers
dac_chx_trg11 (x = 1, 2)	lptim1_ch1	Internal signal from on-chip timers
dac_chx_trg12 (x = 1, 2)	lptim3_ch1	Internal signal from on-chip timers
dac_chx_trg13 (x = 1, 2)	exti9	External pin

31.4.3 DAC clocks

Two clock sources can be used to update the DAC:

- dac_hclk: DAC peripheral clock (AHB clock)
- dac_ker_ck: DAC kernel clock: this clock can be used to synchronize DAC and ADC.
- dac_hold_ck: low-power clock used in Sample and hold mode

The DAC clock is selected in the RCC.

31.4.4 DAC channel enable

Each DAC channel can be powered on by setting its corresponding ENx bit in the DAC_CR register. The DAC channel is then enabled after a t_{WAKEUP} startup time.

DACxRDY bit is set in the DAC_SR register when the DAC interface is ready to accept data. Writing new data or asserting the trigger is not allowed when ENx bit is set while DACxRDY signal is reset.

Note: The ENx bit enables the analog DAC channelx only. The DAC channelx digital interface is enabled even if the ENx bit is reset.

31.4.5 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:

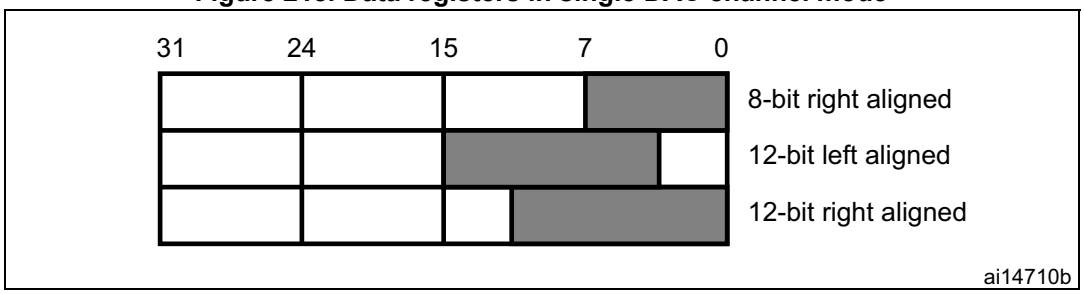
  – 8-bit right alignment: the software has to load data into the DAC_DHR8Rx[7:0] bits (stored into the DHRx[11:4] bits)

  – 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 218. Data registers in single DAC channel mode



- Dual DAC channels (when available)

  There are three possibilities:

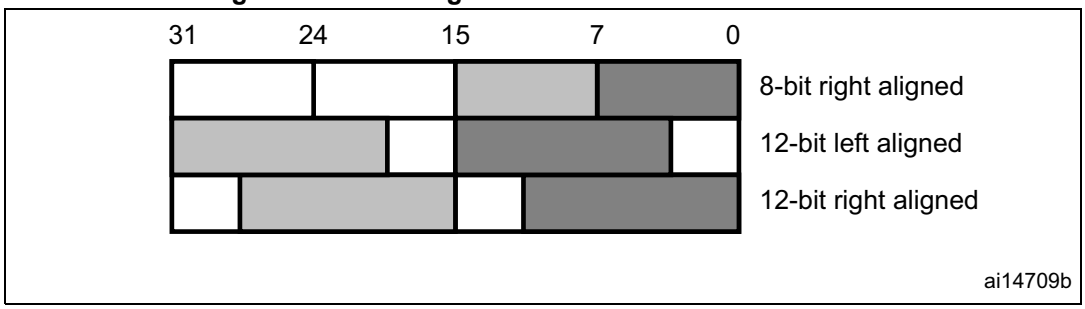
  – 8-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR8RD [7:0] bits (stored into the DHR1[11:4] bits) and data for DAC channel2 to be loaded into the DAC_DHR8RD [15:8] bits (stored into the DHR2[11:4] bits)

  – 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 219. Data registers in dual DAC channel mode



Signed/unsigned data

DAC input data are unsigned: 0x000 corresponds to the minimum value and 0xFFF to the maximum value for 12-bit mode.

The DAC can also handle signed input data in 2's complement format. This is done by setting SINFORMATx bit in the DAC_MCR register.

When SINFORMATx bit is set, the MSB bit of the data written to DHRx registers is inverted when it is copied to the DAC_DORx register, and the DAC interface can accept signed data (Q1.15, Q1.11 or Q1.7 format). DAC_DHR12Lx register can be used to store 16-bit signed data in the data holding registers. The 12 MSBs of 16-bit data are used for the DAC output data and the MSB bit is inverted. The four LSBs are simply ignored.

Table 267. Data format (case of 12-bit data)

SINFORMATx bit	DATA written to DHRx register	DATA transfered to DORx register
0	0x000	0x000
0	0xFFF	0xFFF
1	0x7FF	0xFFF
1	0x000	0x800
1	0xFFF	0x7FF
1	0x800	0x000

31.4.6 DAC conversion

The DAC_DORx cannot be written directly and any data transfer to the DAC channelx must be performed by loading the DAC_DHRx register (write operation to DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx, DAC_DHR8RD, DAC_DHR12RD or DAC_DHR12LD).

Data stored in the DAC_DHRx register are automatically transferred to the DAC_DORx register after one dac_hclk clock cycle, if no hardware trigger is selected (TENx bit in DAC_CR register is reset). However, when a hardware trigger is selected (TENx bit in DAC_CR register is set) and a trigger occurs, the transfer is performed three dac_hclk clock cycles after the trigger signal.

When DAC_DORx is loaded with the DAC_DHRx contents, the analog output voltage becomes available after a time $t_{SETTLING}$ that depends on the power supply voltage and the analog output load.

To synchronize DAC and ADC, the same clock source can be used for both peripherals. This is done by selecting the dac_ker_ck clock instead of the dac_hclk clock (AHB clock) in the RCC.

HFSEL bits of DAC_MCR must be set when dac_hclk or dac_ker_ck clock speed is faster than 80 MHz. It adds an extra delay to the transfer from DAC_DHRx register to DAC_DORx register.

Refer to Table *HFSEL description* below for the limitation of the DAC_DORx update rate depending on HFSEL bits and dac_hclk clock frequency.

If the data is updated or a software/hardware trigger event occurs during the non-allowed period, the peripheral behavior is unpredictable.

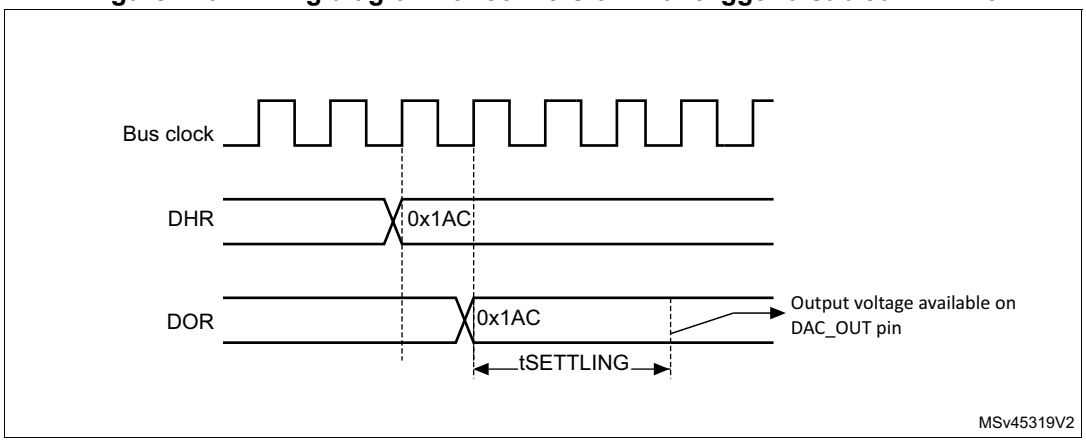
The above timing is only related to the limitation of the DAC interface. Refer also to the $t_{SETTLING}$ parameter value in the product datasheet.

Table 268. HFSEL description

HFSEL [1:0]	AHB frequency	Latency using AHB clock (dac_hclk)	Latency using dac_ker_ck clock	Function
00	≤ 80 MHz	3	4	DAC_DOR update rate up to 3 AHB clock cycles or 4 dac_ker_ck cycles.
01	>80 MHz(1)	5	5	DAC_DOR update rate up to 5 AHB clock or dac_ker_ck cycles.
10	>160 MHz	7	6	DAC_DOR update rate up to 7 AHB clock cycles or 6 dac_ker_ck cycles.
11	Reserved	-	-	-

1. Refer to the device datasheet for the value of the maximum AHB frequency.

Figure 220. Timing diagram for conversion with trigger disabled TEN = 0



31.4.7 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:

$$\text{DACoutput} = V_{REF} \times \frac{DOR}{4096}$$

31.4.8 DAC trigger selection

If the TENx control bit is set, the conversion can then be triggered by an external event (timer counter, external interrupt line). The TSELx[3:0] control bits determine which out of 16 possible events triggers the conversion as shown in TSELx[3:0] bits of the DAC_CR register. These events can be either the software trigger or hardware triggers. Refer to the interconnection table in *Section 31.4.2: DAC pins and internal signals*.

Each time a DAC interface detects a rising edge on the selected trigger source (refer to the table below), the last data stored into the DAC_DHRx register are transferred into the DAC_DORx register. The DAC_DORx register is updated three dac_hclk cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC_DORx register has been loaded with the DAC_DHRx register contents.

Note: TSELx[3:0] bit cannot be changed when the ENx bit is set.

When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DORx register takes only one dac_hclk clock cycle.

31.4.9 DMA requests

Each DAC channel has a DMA capability. Two DMA channels are used to service DAC channel DMA requests.

When an external trigger (but not a software trigger) occurs while the DMAENx bit is set, the value of the DAC_DHRx register is transferred into the DAC_DORx register when the transfer is complete, and a DMA request is generated.

In dual mode, if both DMAENx bits are set, two DMA requests are generated. If only one DMA request is needed, only the corresponding DMAENx bit must be set. In this way, the application can manage both DAC channels in dual mode by using one DMA request and a unique DMA channel.

As DAC_DHRx to DAC_DORx data transfer occurred before the DMA request, the very first data has to be written to the DAC_DHRx before the first trigger event occurs.

DMA underrun

The DAC DMA request is not queued so that if a second external trigger arrives before the acknowledgment for the first external trigger is received (first request), then no new request is issued and the DMA channelx underrun flag DMAUDRx in the DAC_SR register is set, reporting the error condition. The DAC channelx continues to convert old data.

The software must clear the DMAUDRx flag by writing 1, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channelx to restart the transfer correctly. The software must modify the DAC trigger conversion frequency or lighten the DMA

workload to avoid a new DMA underrun. Finally, the DAC conversion could be resumed by enabling both DMA data transfer and conversion trigger.

For each DAC channelx, an interrupt is also generated if its corresponding DMAUDRIEx bit in the DAC_CR register is enabled.

DMA Double data mode

When the DMA controller is used in Normal mode, only 12-bit (or 8-bit) data are transferred by a DMA request. As the AHB width is 32 bits, two 12-bit data may be transferred simultaneously. To use this mode, set the DMADOUBLEx bit of DAC_MCR register.

A DAC DMA request is generated every two external triggers (except for software triggers) when the DMAENx bit is set:

1. When the first trigger is detected, the value of the DAC_DHRx and DAC_DHRBx registers are transferred into the DAC_DORx and DAC_DORBx registers. The actual DAC data is loaded into the DAC_DORx register. A DMA request is then generated. The DMA writes the new data to the DAC_DHRx and DAC_DHRBx data registers.
2. When the next trigger is detected, the actual DAC data is loaded into the DAC_DHRBx register. This second trigger does not generate any DMA request. The DORSTATx bit indicates which DOR data is actually loaded into the analog DAC input.

DMA underrun function is also supported in DMA Double data mode.

In DMA Double mode, DMA requests can only handle one DAC channel. To use two channel outputs in DMA Double mode, each DMA channel has to be configured separately.

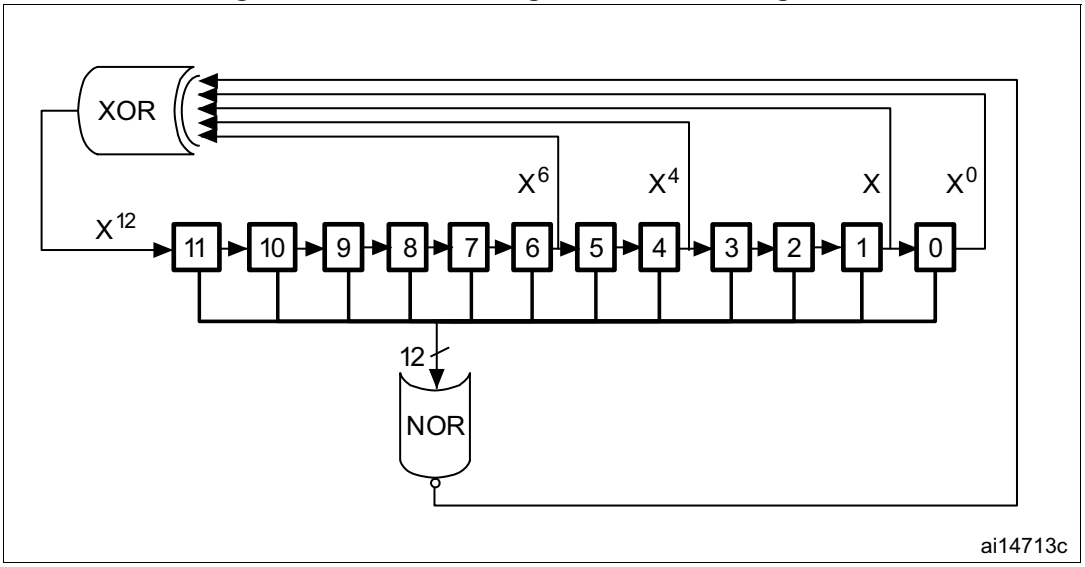
The following conditions must be met to change from Double data to single data mode or vice versa:

- The DAC must be disabled.
- DMAEN bit must be cleared (ENx = 0 and DMAEN = 0).

31.4.10 Noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVEx[1:0] to 01. The preloaded value in LFSR is 0xAAA. This register is updated three dac_hclk clock cycles after each trigger event, following a specific calculation algorithm.

Figure 221. DAC LFSR register calculation algorithm

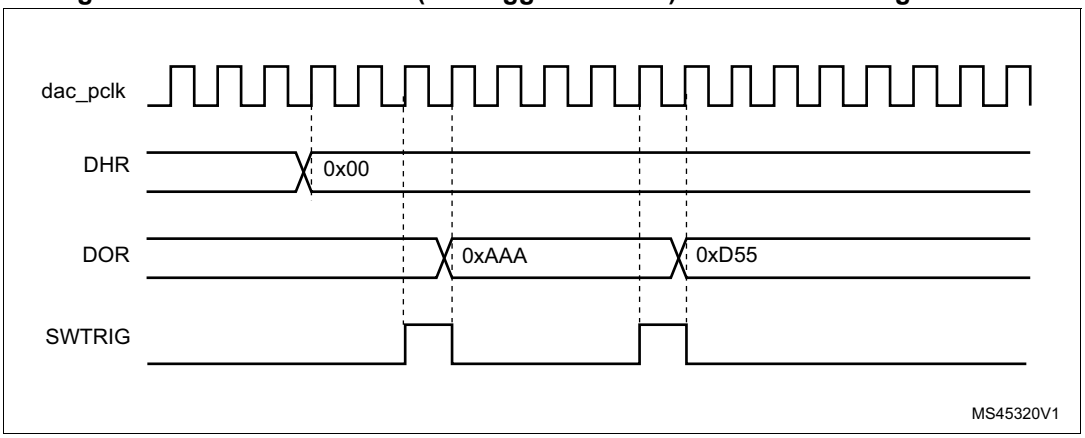


The LFSR value, that may be masked partially or totally by means of the MAMPx[3:0] bits in the DAC_CR register, is added up to the DAC_DHRx contents without overflow and this value is then transferred into the DAC_DORx register.

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 222. 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.

31.4.11 Triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVEx[1:0] to 10”. The amplitude is configured through the MAMPx[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three dac_hclk clock cycles after each trigger event. The value of this counter is then added to the DAC_DHRx register without overflow and the sum is transferred into the DAC_DORx register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the MAMPx[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the WAVEx[1:0] bits.

Figure 223. DAC triangle wave generation

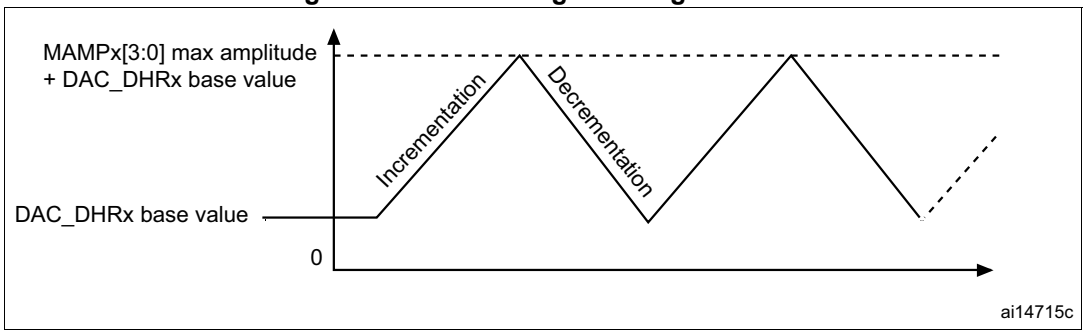
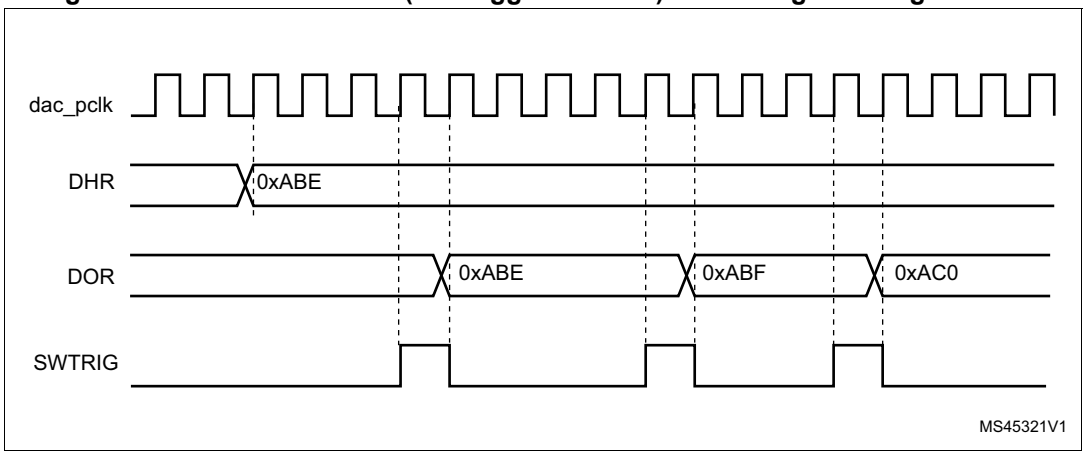


Figure 224. DAC conversion (SW trigger enabled) with triangle wave generation



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.

31.4.12 DAC channel modes

Each DAC channel can be configured in Normal mode or Sample and hold mode. The output buffer can be enabled to obtain a high drive capability. Before enabling output buffer, the voltage offset needs to be calibrated. This calibration is performed at the factory (loaded after reset) and can be adjusted by software during application operation.

Normal mode

In Normal mode, there are four combinations, by changing the buffer state and by changing the DACx_OUTy pin interconnections.

To enable the output buffer, the MODEx[2:0] bits in DAC_MCR register must be:

- 000: DAC is connected to the external pin
- 001: DAC is connected to external pin and to on-chip peripherals

To disable the output buffer, the MODEx[2:0] bits in DAC_MCR register must be:

- 010: DAC is connected to the external pin
- 011: DAC is connected to on-chip peripherals

Sample and hold mode

In Sample and hold mode, the DAC core converts data on a triggered conversion, and then holds the converted voltage on a capacitor. When not converting, the DAC cores and buffer are completely turned off between samples and the DAC output is tri-stated, therefore reducing the overall power consumption. A stabilization period, which value depends on the buffer state, is required before each new conversion.

In this mode, the DAC core and all corresponding logic and registers are driven by the LSI or LSE low-speed clock (dac_hold_ck) in addition to the dac_hclk clock, allowing the DAC channels to be used in deep low power modes such as Stop mode.

The LSI or LSE low-speed clock (dac_hold_ck) must not be stopped when the Sample and hold mode is enabled.

The sample/hold mode operations can be divided into 3 phases:

1. Sample phase: the sample/hold element is charged to the desired voltage. The charging time depends on capacitor value (internal or external, selected by the user). The sampling time is configured with the TSAMPLEx[9:0] bits in DAC_SHSRx register. During the write of the TSAMPLEx[9:0] bits, the BWSTx bit in DAC_SR register is set to 1 to synchronize between both clocks domains (AHB and low speed clock) and allowing the software to change the value of sample phase during the DAC channel operation
2. Hold phase: the DAC output channel is tri-stated, the DAC core and the buffer are turned off, to reduce the current consumption. The hold time is configured with the THOLDx[9:0] bits in DAC_SHHR register
3. Refresh phase: the refresh time is configured with the TREFRESHx[7:0] bits in DAC_SHRR register

The timings for the three phases above are in units of LSI/LSE clock periods. As an example, to configure a sample time of 350 µs, a hold time of 2 ms and a refresh time of 100 µs assuming LSI/LSE ~32 KHz is selected:

12 cycles are required for sample phase: TSAMPLEx[9:0] = 11,

62 cycles are required for hold phase: THOLDx[9:0] = 62,

and 4 cycles are required for refresh period: TREFRESHx[7:0] = 4.

In this example, the power consumption is reduced by almost a factor of 15 versus Normal modes.

The formulas to compute the right sample and refresh timings are described in the table below, the Hold time depends on the leakage current.

Table 269. Sample and refresh timings

Buffer State	t_{SAMP} (1)(2)	$t_{REFRESH}$ (2)(3)
Enable	7 µs + (10*R_{BON}*C_{SH})	7 µs + (R_{BON}*C_{SH})*ln(2*N_{LSB})
Disable	3 µs + (10*R_{BOFF}*C_{SH})	3 µs + (R_{BOFF}*C_{SH})*ln(2*N_{LSB})

1. In the above formula the settling to the desired code value with ½ LSB or accuracy requires 10 constant time for 12 bits resolution. For 8 bits resolution, the settling time is 7 constant time.
2. C_{SH} is the capacitor in Sample and hold mode.
3. The tolerated voltage drop during the hold phase “Vd” is represented by the number of LSBs after the capacitor discharging with the output leakage current. The settling back to the desired value with ½ LSB error accuracy requires ln(2*Nlsb) constant time of the DAC.

Example of the sample and refresh time calculation with output buffer on

The values used in the example below are provided as indication only. Refer to the product datasheet for product data.

C_{SH} = 100 nF

V_{REF+} = 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)

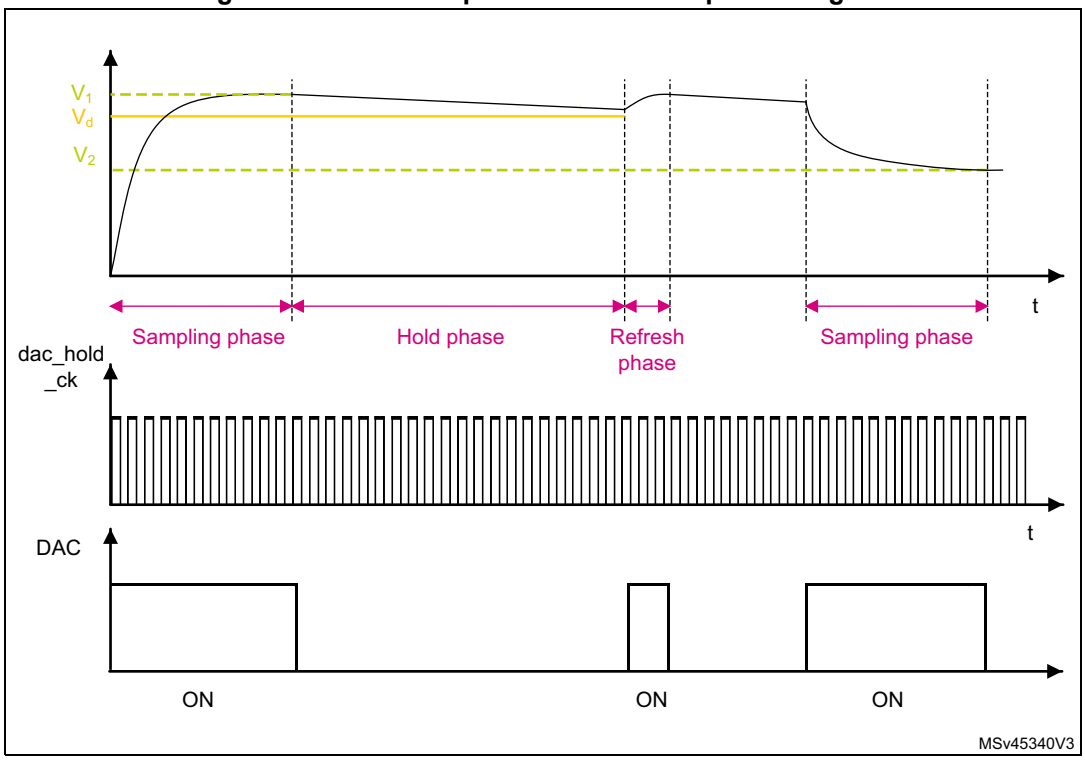
Hold phase:

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 225. 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 270. 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 270. 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)

31.4.13 DAC channel buffer calibration

The transfer function for an N-bit digital-to-analog converter (DAC) is:

$$V_{out} = ((D/2^N) \times G \times V_{ref}) + V_{OS}$$

Where V_{OUT} is the analog output, D is the digital input, G is the gain, V_{ref} is the nominal full-scale voltage, and V_{os} is the offset voltage. For an ideal DAC channel, G = 1 and V_{os} = 0.

Due to output buffer characteristics, the voltage offset may differ from part-to-part and introduce an absolute offset error on the analog output. To compensate the V_{os}, a calibration is required by a trimming technique.

The calibration is only valid when the DAC channelx is operating with buffer enabled (MODEx[2:0] = 000b or 001b or 100b or 101b). if applied in other modes when the buffer is off, it has no effect. During the calibration:

- The buffer output is disconnected from the pin internal/external connections and put in tristate mode (HiZ).
- The buffer acts as a comparator to sense the middle-code value 0x800 and compare it to VREF+/2 signal through an internal bridge, then toggle its output signal to 0 or 1 depending on the comparison result (CAL_FLAGx bit).

Two calibration techniques are provided:

- Factory trimming (default setting)

  The DAC buffer offset is factory trimmed. The default value of OTRIMx[4:0] bits in DAC_CCR register is the factory trimming value and it is loaded once DAC digital interface is reset.
- User trimming

  The user trimming can be done when the operating conditions differs from nominal factory trimming conditions and in particular when V_{DDA} voltage, temperature, VREF+ values change and can be done at any point during application by software.

Note: Refer to the datasheet for more details of the Nominal factory trimming conditions

In addition, when V_{DD} is removed (example the device enters in STANDBY or VBAT modes) the calibration is required.

The steps to perform a user trimming calibration are as below:

1. If the DAC channel is active, write 0 to ENx bit in DAC_CR to disable the channel.
2. Select a mode where the buffer is enabled, by writing to DAC_MCR register, MODEx[2:0] = 000b or 001b or 100b or 101b.
3. Start the DAC channelx calibration, by setting the CENx bit in DAC_CR register to 1.
4. Apply a trimming algorithm:
   a) Write a code into OTRIMx[4:0] bits, starting by 00000b.
   b) Wait for t_{TRIM} delay.
   c) Check if CAL_FLAGx bit in DAC_SR is set to 1.
   d) If CAL_FLAGx is set to 1, the OTRIMx[4:0] trimming code is found and can be used during *device* operation to compensate the output value, else increment OTRIMx[4:0] and repeat sub-steps from (a) to (d) again.

The software algorithm may use either a successive approximation or dichotomy techniques to compute and set the content of OTRIMx[4:0] bits in a faster way.

The commutation/toggle of CAL_FLAGx bit indicates that the offset is correctly compensated and the corresponding trim code must be kept in the OTRIMx[4:0] bits in DAC_CCR register.

Note: A t_{TRIM} delay must be respected between the write to the OTRIMx[4:0] bits and the read of the CAL_FLAGx bit in DAC_SR register in order to get a correct value.This parameter is specified into datasheet electrical characteristics section.

If V_{DDA}, VREF+ and temperature conditions do not change during device operation while it enters more often in standby and VBAT mode, the software may store the OTRIMx[4:0] bits found in the first user calibration in the flash or in back-up registers. then to load/write them directly when the device power is back again thus avoiding to wait for a new calibration time.

When CENx bit is set, it is not allowed to set ENx bit.

31.4.14 Dual DAC channel conversion modes (if dual channels are available)

To efficiently use the bus bandwidth in applications that require the two DAC channels at the same time, three dual registers are implemented: DHR8RD, DHR12RD and DHR12LD. A unique register access is then required to drive both DAC channels at the same time. For the wave generation, no accesses to DHRxxxD registers are required. As a result, two output channels can be used either independently or simultaneously.

15 conversion modes are possible using the two DAC channels and these dual registers. All the conversion modes can nevertheless be obtained using separate DHRx registers if needed.

All modes are described in the paragraphs below.

Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DHR1 register is transferred into DAC_DOR1 (three dac_hclk clock cycles later).

When a DAC channel2 trigger arrives, the DHR2 register is transferred into DAC_DOR2 (three dac_hclk clock cycles later).

Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the LFSR2 counter is updated.

Independent trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR masks values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the LFSR2 counter is updated.

Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 triangle counter is then updated.

Independent trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bits.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 triangle counter is then updated.

Simultaneous software start

To configure the DAC in this conversion mode, the following sequence is required:

- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

In this configuration, one dac_hclk clock cycle later, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively.

Simultaneous trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively (after three dac_hclk clock cycles).

Simultaneous trigger with single LFSR generation

1. To configure the DAC in this conversion mode, the following sequence is required:
2. Set the two DAC channel trigger enable bits TEN1 and TEN2.
3. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
4. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
5. Load the dual DAC channel data to the desired DHR register (DHR12RD, DHR12LD or DHR8RD).

When a trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The LFSR1 counter is then updated. At the same time, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The LFSR2 counter is then updated.

Simultaneous trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR mask values using the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The LFSR1 counter is then updated.
At the same time, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The LFSR2 counter is then updated.

Simultaneous trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value using the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

At the same time, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 triangle counter is then updated.

Simultaneous trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three AHB clock cycles later). Then the DAC channel1 triangle counter is updated.

At the same time, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the DAC channel2 triangle counter is updated.

31.4.15 DAC Autonomous mode

The Autonomous mode can be used to update the DAC output voltage in Stop mode. This allows DMA transfers to be performed when the device operates in Run, Sleep or Stop mode. The Autonomous mode is supported AUTOMODE only when the DAC is in Sample and hold mode. It is enabled by setting the bit in the DAC_AUTOCR register. DMA requests must also be enabled (DMAEN = 1).

When the AUTOMODE bit is set, each hardware trigger signal generates an AHB clock request to the RCC. Once the peripheral receives the AHB clock, the content of the DHRx register is loaded into DAC_DOR register and a DMA request is generated. When the DMA transaction is complete, the DAC deasserts the AHB clock request and waits for a new trigger event.

When the Sample and hold mode is selected, the dac_ker_ck low-power clock must be enabled by the RCC.

DMA transfers in Autonomous mode are performed following the sequence described below:

1. The DAC outputs the voltage on a GPIO or on an on-chip peripheral (dac_ker_ck and dac_hclk are not required).
2. The DAC receives a hardware trigger from another peripheral.
3. The DAC asserts the clock request.
4. The DAC loads DHRx register into DAC_DOR register.
5. The DAC generates a DMA request.
6. The DMA transfers the new data from memory to DHRx registers.
7. The DAC deasserts the clock request.

To initialize the DAC to operate autonomously in Stop mode, follow the sequence below:

1. Enable the DAC clock Autonomous mode in the RCC.
2. Select the DAC kernel clock in the RCC.
3. Enable dac_hold_ck clock for the DAC in the RCC.
4. Select a Sample and hold mode through MODEx bit.
5. Configure the Sample and hold mode by setting TSAMPLEx[9:0], THOLDx[9:0], TREFRESHx[7:0] bits.
6. Load the target value into DHRx register.
7. Set TENx bit to output the DAC value.
8. Configure the trigger setting through TSEL[3:0] bits.
9. Set DMAENx bit and configure the DMA interface.
10. Set the AUTOMODE bit.

Timing uncertainty of DHRx to DAC_DOR transfer

After each trigger event, the data contained in DHRx register is transferred to the DAC_DOR register. Since this transfer is based on AHB clock, the transfer timing depends on the clock availability.

When the AHB clock is stopped, a minimum time is required to restart it. This time depends on the clock source.

31.5 DAC in low-power modes

Table 271. Effect of low-power modes on DAC

Mode	Description
Sleep	No effect, DAC used with DMA.
Stop(1)	The DAC remains functional and can perform DMA transfers in Sample and hold mode.
Standby	The DAC peripheral is powered down and must be reinitialized after exiting Standby or Shutdown mode.
Shutdown	

1. Refer to *Section 31.3: DAC implementation* for information on the Stop modes supported by the DAC peripheral.

31.6 DAC interrupts

Table 272. 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

31.7 DAC registers

Refer to *Section 1 on page 104* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

31.7.1 DAC control register (DAC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CEN2	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[3]	TSEL2[2]	TSEL2[1]	TSEL2[0]	TEN2	EN2
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[3]	TSEL1[2]	TSEL1[1]	TSEL1[0]	TEN1	EN1
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **CEN2**: DAC channel2 calibration enable

This bit is set and cleared by software to enable/disable DAC channel2 calibration, it can be written only if EN2 bit is set to 0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel2 in Normal operating mode

1: DAC channel2 in calibration mode

Note: This bit is available only on dual-channel DACs. Refer to Section 31.3: 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 31.3: 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 31.3: 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 31.3: 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 31.3: 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 31.4.2: DAC pins and internal signals* for details on trigger configuration and mapping.

Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled).
These bits are available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bit 17 **TEN2**: DAC channel2 trigger enable

This bit is set and cleared by software to enable/disable DAC channel2 trigger

0: DAC channel2 trigger disabled and data written into the DAC_DHR2 register are transferred one dac_hclk clock cycle later to the DAC_DOR2 register
1: DAC channel2 trigger enabled and data from the DAC_DHR2 register are transferred three dac_hclk clock cycles later to the DAC_DOR2 register

Note: When software trigger is selected, the transfer from the DAC_DHR2 register to the DAC_DOR2 register takes only one dac_hclk clock cycle.
These bits are available only on dual-channel DACs. Refer to Section 31.3: 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 31.3: DAC implementation.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CEN1**: DAC channel1 calibration enable

This bit is set and cleared by software to enable/disable DAC channel1 calibration, it can be written only if bit EN1 = 0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel1 in Normal operating mode

1: DAC channel1 in calibration mode

Bit 13 **DMAUDRIE1**: DAC channel1 DMA Underrun Interrupt enable

This bit is set and cleared by software.

0: DAC channel1 DMA Underrun Interrupt disabled

1: DAC channel1 DMA Underrun Interrupt enabled

Bit 12 **DMAEN1**: DAC channel1 DMA enable

This bit is set and cleared by software.

0: DAC channel1 DMA mode disabled

1: DAC channel1 DMA mode enabled

Bits 11:8 **MAMP1[3:0]**: DAC channel1 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1

0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3

0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7

0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15

0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31

0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63

0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127

0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255

1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511

1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023

1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047

≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bits 7:6 **WAVE1[1:0]**: DAC channel1 noise/triangle wave generation enable

These bits are set and cleared by software.

00: wave generation disabled

01: Noise wave generation enabled

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 31.4.2: DAC pins and internal signals* for details on trigger configuration and mapping.
Note: Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bit 1 **TEN1**: DAC channel1 trigger enable
This bit is set and cleared by software to enable/disable DAC channel1 trigger.
0: DAC channel1 trigger disabled and data written into the DAC_DHR1 register are transferred one dac_hclk clock cycle later to the DAC_DOR1 register
1: DAC channel1 trigger enabled and data from the DAC_DHR1 register are transferred three dac_hclk clock cycles later to the DAC_DOR1 register
Note: When software trigger is selected, the transfer from the DAC_DHR1 *register to the DAC_DOR1 register takes only one dac_hclk clock cycle.*

Bit 0 **EN1**: DAC channel1 enable
This bit is set and cleared by software to enable/disable DAC channel1.
0: DAC channel1 disabled
1: DAC channel1 enabled

31.7.2 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 dac_hclk clock cycle later) once the DAC_DHR2 register value has been loaded into the DAC_DOR2 register.

This bit is available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bit 0 **SWTRIG1**: DAC channel1 software trigger

This bit is set by software to trigger the DAC in software trigger mode.

0: No trigger

1: Trigger

Note: This bit is cleared by hardware (one dac_hclk clock cycle later) once the DAC_DHR1 register value has been loaded into the DAC_DOR1 register.

31.7.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC1DHRB[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 27:16 **DACC1DHRB[11:0]**: DAC channel1 12-bit right-aligned data B

These bits are written by software. They specify 12-bit data for DAC channel1 when the DAC operates in Double data mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data

These bits are written by software. They specify 12-bit data for DAC channel1.

31.7.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC1DHRB[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC1DHRB[11:0]**: DAC channel1 12-bit left-aligned data B
These bits are written by software. They specify 12-bit data for DAC channel1 when the DAC operates in Double data mode.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software.
They specify 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

31.7.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHRB[7:0]								DACC1DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DACC1DHRB[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1 when the DAC operates in Double data mode.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1.

31.7.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2)

This register is available only on dual-channel DACs. Refer to *Section 31.3: DAC implementation*.

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DHRB[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 27:16 **DACC2DHRB[11:0]**: DAC channel2 12-bit right-aligned data

These bits are written by software. They specify 12-bit data for DAC channel2 when the DAC operates in DMA Double data mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data

These bits are written by software. They specify 12-bit data for DAC channel2.

31.7.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2)

This register is available only on dual-channel DACs. Refer to *Section 31.3: DAC implementation*.

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHRB[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC2DHRB[11:0]**: DAC channel2 12-bit left-aligned data B

These bits are written by software. They specify 12-bit data for DAC channel2 when the DAC operates in Double data mode.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data

These bits are written by software which specify 12-bit data for DAC channel2.

Bits 3:0 Reserved, must be kept at reset value.

31.7.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2)

This register is available only on dual-channel DACs. Refer to *Section 31.3: DAC implementation*.

Address offset: 0x1C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHRB[7:0]								DACC2DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DACC2DHRB[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel2 when the DAC operates in Double data mode.

Bits 7:0 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

31.7.9 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.

31.7.10 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.

31.7.11 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.

31.7.12 DAC channel1 data output register (DAC_DOR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC1DORB[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 27:16 **DACC1DORB[11:0]**: DAC channel1 data output
These bits are read-only. They contain data output for DAC channel1 B.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DOR[11:0]**: DAC channel1 data output
These bits are read-only, they contain data output for DAC channel1.

31.7.13 DAC channel2 data output register (DAC_DOR2)

This register is available only on dual-channel DACs. Refer to *Section 31.3: DAC implementation*.

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DORB[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC2DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 27:16 **DACC2DORB[11:0]**: DAC channel2 data output
These bits are read-only. They contain data output for DAC channel2 B.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DOR[11:0]**: DAC channel2 data output
These bits are read-only, they contain data output for DAC channel2.

31.7.14 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	DMAUDR2	DORSTAT2	DAC2RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1	r	r											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BWST1	CAL_FLAG1	DMAUDR1	DORSTAT1	DAC1RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1	r	r											

Bit 31 **BWST2**: DAC channel2 busy writing sample time flag

This bit is systematically set just after Sample and hold mode enable. It is set each time the software writes the register DAC_SHSR2, It is cleared by hardware when the write operation of DAC_SHSR2 is complete. (It takes about 3 LSI/LSE periods of synchronization).

0:There is no write operation of DAC_SHSR2 ongoing: DAC_SHSR2 can be written

1:There is a write operation of DAC_SHSR2 ongoing: DAC_SHSR2 cannot be written

Note: This bit is available only on dual-channel DACs. Refer to Section 31.3: 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 31.3: 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 31.3: DAC implementation.

Bit 28 **DORSTAT2**: DAC channel2 output register status bit

This bit is set and cleared by hardware. It is applicable only when the DAC operates in Double data mode.

0: DOR[11:0] is used actual DAC output

1: DORB[11:0] is used actual DAC output

Note: This bit is available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bit 27 **DAC2RDY**: DAC channel2 ready status bit

This bit is set and cleared by hardware.

0: DAC channel2 is not yet ready to accept the trigger nor output data

1: DAC channel2 is ready to accept the trigger or output data

Note: This bit is available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bits 26:16 Reserved, must be kept at reset value.

Bit 15 **BWST1**: DAC channel1 busy writing sample time flag

This bit is systematically set just after Sample and hold mode enable and is set each time the software writes the register DAC_SHSR1, It is cleared by hardware when the write operation of DAC_SHSR1 is complete. (It takes about 3 LSI/LSE periods of synchronization).

0:There is no write operation of DAC_SHSR1 ongoing: DAC_SHSR1 can be written

1:There is a write operation of DAC_SHSR1 ongoing: DAC_SHSR1 cannot be written

Bit 14 **CAL_FLAG1**: DAC channel1 calibration offset status

This bit is set and cleared by hardware

0: calibration trimming value is lower than the offset correction value

1: calibration trimming value is equal or greater than the offset correction value

Bit 13 **DMAUDR1**: DAC channel1 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel1

1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)

Bit 12 **DORSTAT1**: DAC channel1 output register status bit

This bit is set and cleared by hardware. It is applicable only when the DAC operates in Double data mode.

0: DOR[11:0] is used actual DAC output

1: DORB[11:0] is used actual DAC output

Bit 11 **DAC1RDY**: DAC channel1 ready status bit

This bit is set and cleared by hardware.

0: DAC channel1 is not yet ready to accept the trigger nor output data

1: DAC channel1 is ready to accept the trigger or output data

Bits 10:0 Reserved, must be kept at reset value.

31.7.15 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 31.3: DAC implementation*.

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **OTRIM1[4:0]**: DAC channel1 offset trimming value

31.7.16 DAC mode control register (DAC_MCR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SINFORMAT2	DMADOUBLE2	Res.	Res.	Res.	Res.	Res.	MODE2[2:0]		
						rw	rw						rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HFSEL1	HFSEL0	Res.	Res.	Res.	Res.	SINFORMAT1	DMADOUBLE1	Res.	Res.	Res.	Res.	Res.	MODE1[2:0]		
rw	rw					rw	rw						rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **SINFORMAT2**: Enable signed format for DAC channel2

This bit is set and cleared by software.

0: Input data is in unsigned format

1: Input data is in signed format (2's complement). The MSB bit represents the sign.

Note: This bit is available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

Bit 24 **DMADOUBLE2**: DAC channel2 DMA double data mode

This bit is set and cleared by software.

0: DMA Normal mode selected

1: DMA Double data mode selected

Note: This bit is available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

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

Bits 18:16 **MODE2[2:0]:** DAC channel2 mode

These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN2 = 0 and bit CEN2 = 0 in the DAC_CR register). If EN2 = 1 or CEN2 = 1 the write operation is ignored.

They can be set and cleared by software to select the DAC channel2 mode:

– DAC channel2 in Normal mode

000: DAC channel2 is connected to external pin with Buffer enabled

001: DAC channel2 is connected to external pin and to on chip peripherals with buffer enabled

010: DAC channel2 is connected to external pin with buffer disabled

011: DAC channel2 is connected to on chip peripherals with Buffer disabled

– DAC channel2 in Sample and hold mode

100: DAC channel2 is connected to external pin with Buffer enabled

101: DAC channel2 is connected to external pin and to on chip peripherals with Buffer enabled

110: DAC channel2 is connected to external pin and to on chip peripherals with Buffer disabled

111: DAC channel2 is connected to on chip peripherals with Buffer disabled

Note: This register can be modified only when EN2 = 0.

Refer to Section 31.3: DAC implementation for the availability of DAC channel2.

Bits 15:14 **HFSEL[1:0]**: High frequency interface mode selection
00: High frequency interface mode disabled
01: High frequency interface mode compatible to AHB>80 MHz enabled
10: High frequency interface mode compatible to AHB>160 MHz enabled
11: Reserved

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **SINFORMAT1**: Enable signed format for DAC channel1
This bit is set and cleared by software.
0: Input data is in unsigned format
1: Input data is in signed format (2's complement). The MSB bit represents the sign.

Bit 8 **DMADOUBLE1**: DAC channel1 DMA double data mode
This bit is set and cleared by software.
0: DMA Normal mode selected
1: DMA Double data mode selected

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

Bits 2:0 **MODE1[2:0]:** DAC channel1 mode
These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN1 = 0 and bit CEN1 = 0 in the DAC_CR register). If EN1 = 1 or CEN1 = 1 the write operation is ignored.
They can be set and cleared by software to select the DAC channel1 mode:
– DAC channel1 in Normal mode
000: DAC channel1 is connected to external pin with Buffer enabled
001: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled
010: DAC channel1 is connected to external pin with Buffer disabled
011: DAC channel1 is connected to on chip peripherals with Buffer disabled
– DAC channel1 in sample & hold mode
100: DAC channel1 is connected to external pin with Buffer enabled
101: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled
110: DAC channel1 is connected to external pin and to on chip peripherals with Buffer disabled
111: DAC channel1 is connected to on chip peripherals with Buffer disabled
Note: This register can be modified only when EN1 = 0.

31.7.17 DAC channel1 sample and hold sample time register (DAC_SHSR1)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TSAMPLE1[9:0]:** DAC channel1 sample time (only valid in Sample and hold mode)

These bits can be written when the DAC channel1 is disabled or also during normal operation. in the latter case, the write can be done only when BWST1 of DAC_SR register is low, If BWST1 = 1, the write operation is ignored.

*Note: It represents the number of LSI/LSE clocks to perform a sample phase. Sampling time = (*TSAMPLE1[9:0] + 1) x *LSI/LSE clock period.*

31.7.18 DAC channel2 sample and hold sample time register (DAC_SHSR2)

This register is available only on dual-channel DACs. Refer to *Section 31.3: DAC implementation*.

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE2[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TSAMPLE2[9:0]:** DAC channel2 sample time (only valid in Sample and hold mode)
These bits can be written when the DAC channel2 is disabled or also during normal operation. in the latter case, the write can be done only when BWST2 of DAC_SR register is low, if BWST2 = 1, the write operation is ignored.

*Note: It represents the number of LSI/LSE clocks to perform a sample phase. Sampling time = (*TSAMPLE1[9:0] + 1) x *LSI/LSE clock period.*

31.7.19 DAC sample and hold time register (DAC_SHHR)

Address offset: 0x48

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	THOLD2[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **THOLD2[9:0]:** DAC channel2 hold time (only valid in Sample and hold mode).
Hold time = (THOLD[9:0]) x LSI/LSE clock period

Note: This register can be modified only when EN2 = 0.
These bits are available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

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

Bits 9:0 **THOLD1[9:0]:** DAC channel1 hold time (only valid in Sample and hold mode)
Hold time = (THOLD[9:0]) x LSI/LSE clock period

Note: This register can be modified only when EN1 = 0.

Note: These bits can be written only when the DAC channel is disabled and in Normal operating mode (when bit ENx = 0 and bit CENx = 0 in the DAC_CR register). If ENx = 1 or CENx = 1 the write operation is ignored.

31.7.20 DAC sample and hold refresh time register (DAC_SHRR)

Address offset: 0x4C

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 23:16 **TREFRESH2[7:0]:** DAC channel2 refresh time (only valid in Sample and hold mode)
Refresh time = (TREFRESH[7:0]) x LSI/LSE clock period

Note: This register can be modified only when EN2 = 0.
These bits are available only on dual-channel DACs. Refer to Section 31.3: DAC implementation.

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

Bits 7:0 **TREFRESH1[7:0]:** DAC channel1 refresh time (only valid in Sample and hold mode)
Refresh time = (TREFRESH[7:0]) x LSI/LSE clock period

Note: This register can be modified only when EN1 = 0.

Note: These bits can be written only when the DAC channel is disabled and in Normal operating mode (when bit ENx = 0 and bit CENx = 0 in the DAC_CR register). If ENx = 1 or CENx = 1 the write operation is ignored.

31.7.21 DAC Autonomous mode control register (DAC_AUTOCR)

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

Bit 22 **AUTOMODE:** DAC Autonomous mode
This bit is set and cleared by software.
0: DAC Autonomous mode disabled
1: DAC Autonomous mode enabled

Bits 21:0 Reserved, must be kept at reset value.

31.7.22 DAC register map

Table 273 summarizes the DAC registers.

Table 273. DAC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DAC_CR	Res.	CEN2	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[2:0]		TSEL23	TSEL22	TSEL21	TSEL20	TEN2	EN2	Res.	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL13	TSEL12	TSEL11	TSEL10	TEN1	EN1
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	DAC_SWTRGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	DACC1DHRB[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x0C	DAC_DHR12L1	DACC1DHRB[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				
0x10	DAC_DHR8R1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHRB[7:0]								DACC1DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	DAC_DHR12R2	Res.	Res.	Res.	Res.	DACC2DHRB[11:0]												Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x18	DAC_DHR12L2	DACC2DHRB[11:0]												Res.	Res.	Res.	Res.	DACC2DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				
0x1C	DAC_DHR8R2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHRB[7:0]								DACC2DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	DAC_DHR12RD	Res.	Res.	Res.	Res.	DACC2DHR[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x24	DAC_DHR12LD	DACC2DHR[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				
0x28	DAC_DHR8RD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[7:0]								DACC1DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	DAC_DOR1	Res.	Res.	Res.	Res.	DACC1DORB[11:0]												Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x30	DAC_DOR2	Res.	Res.	Res.	Res.	DACC2DORB[11:0]												Res.	Res.	Res.	Res.	DACC2DOR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x34	DAC_SR	BWST2	CAL_FLAG2	DMAUDR2	DORSTAT2	DAC2RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BWST1	CAL_FLAG1	DMAUDR1	DORSTAT1	DAC1RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0												0	0	0	0	0											

Table 273. 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.	SINFORMAT2	DMADOUBLE2	Res.	Res.	Res.	Res.	Res.	MODE2 [2:0]			HFSEL1	HFSEL0	Res.	Res.	Res.	Res.	SINFORMAT1	DMADOUBLE1	Res.	Res.	Res.	Res.	Res.	MODE1 [2:0]		
	Reset value							0	0						0	0	0	0	0					0	0						0	0	0
0x40	**DAC_ SHSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x44	**DAC_ SHSR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE2[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x48	**DAC_ SHHR**	Res.	Res.	Res.	Res.	Res.	Res.	THOLD2[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
	Reset value							0	0	0	0	0	0	0	0	0	1							0	0	0	0	0	0	0	0	0	1
0x4C	**DAC_ SHRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH2[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
	Reset value									0	0	0	0	0	0	0	1									0	0	0	0	0	0	0	1
0x54	**DAC_ AUTOCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AUTOMODE	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																						

Refer to *Section 2.3 on page 115* for the register boundary addresses.

32 Voltage reference buffer (VREFBUF)

32.1 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.

32.2 VREFBUF implementation

The table below describes the VREFBUF voltages typical values:

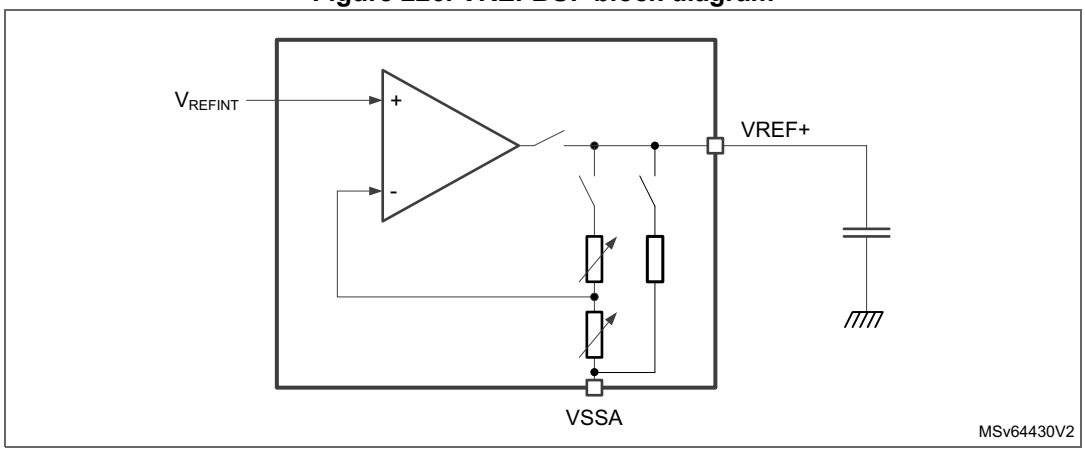
Table 274. VREFBUF typical values

Symbol	Value
VREFBUF0	1.5 V
VREFBUF1	1.8 V
VREFBUF2	2.048 V
VREFBUF3	2.5 V

Note: Refer to the product datasheet for more details.

32.3 VREFBUF functional description

Figure 226. VREFBUF block diagram



The internal voltage reference buffer is an operational amplifier, with programmable gain. The amplifier input is connected to the internal voltage reference V_{REFINT}. The VREFBUF supports four voltages[(a)], which are configured with VRS bits in the VREFBUF_CSR register:

- VRS = 000: VREFBUF0 voltage selected.
- VRS = 001: VREFBUF1 voltage selected.
- VRS = 010: VREFBUF2 voltage selected.
- VRS = 011: VREFBUF3 voltage selected.

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 275. VREF buffer modes

ENVR	HIZ	VREF buffer configuration
0	0	VREFBUF buffer off mode: – V_{REF+} pin pulled-down to V_{SSA}
0	1	External voltage reference mode (default value): – VREFBUF buffer off – V_{REF+} pin input mode
1	0	Internal voltage reference mode: – VREFBUF buffer on – V_{REF+} pin connected to VREFBUF buffer output
1	1	Hold mode: – VREF is enable without output buffer, VREF+ pin voltage is hold with the external capacitor – VRR detection disabled and VRR bit keeps last state

After enabling the VREFBUF by setting ENVR bit and clearing HIZ bit in the VREFBUF_CSR register, the user must wait until VRR bit is set, meaning that the voltage reference output has reached its expected value.

32.4 VREFBUF trimming

The VREFBUF output voltage is factory-calibrated by ST. At reset, and each time the VRS setting is changed, the calibration data is automatically loaded to the TRIM register.

Optionally user can trim the output voltage by changing the TRIM register bits directly. In this case, the VRS setting has no more effect on the TRIM register until the device is reset.

a. The minimum V_{DDA} voltage depends on VRS setting, refer to the product datasheet.

32.5 VREFBUF registers

32.5.1 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.	VRS[2:0]			VRR	Res.	HIZ	ENVR
									rw	rw	rw	r		rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **VRS[2:0]**: Voltage reference scale

These bits select the value generated by the voltage reference buffer.

VRS = 000: VREFBUF0 voltage selected.
VRS = 001: VREFBUF1 voltage selected.
VRS = 010: VREFBUF2 voltage selected.
VRS = 011: VREFBUF3 voltage selected.
Others: Reserved

Note: Refer to the product datasheet for each VREFBUFx voltage setting value.
The software can program this bitfield only when the VREFBUF is disabled (ENVR=0).

Bit 3 **VRR:** Voltage reference buffer ready

0: the voltage reference buffer output is not ready.
1: the voltage reference buffer output reached the requested level.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **HIZ:** High impedance mode

This bit controls the analog switch to connect or not the V_{REF+} pin.
0: V_{REF+} pin is internally connected to the voltage reference buffer output.
1: V_{REF+} pin is high impedance.
Refer to *Table 275: 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.

32.5.2 VREFBUF calibration control register (VREFBUF_CCR)

Address offset: 0x04

Reset value: 0x0000 00XX

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

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

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **TRIM[5:0]**: Trimming code

The TRIM code is a 6-bit unsigned data (minimum 000000, maximum 111111) that is set and updated according the mechanism described below.

Reset:

TRIM[5:0] is automatically initialized with the VRS = 0 trimming value stored in the Flash memory during the production test.

VRS change:

TRIM[5:0] is automatically initialized with the trimming value (corresponding to VRS setting) stored in the Flash memory during the production test.

Write in TRIM[5:0]:

User can modify the TRIM[5:0] with an arbitrary value. This is permanently disabling the control of the trimming value with VRS (until the device is reset).

Note: If the user application performs the trimming, the trimming code must start from 000000 to 111111 in ascending order.

32.5.3 VREFBUF register map

The following table gives the VREFBUF register map and the reset values.

Table 276. 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.	VRS[2:0]			VRR	Res.	HIZ	ENVR
	Reset value																										0	0	0	0		1	0
0x04	**VREFBUF_CCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]					
	Reset value																											x	x	x	x	x	x

Refer to *Section 2.3 on page 115* for the register boundary addresses.

33 Comparator (COMP)

33.1 Introduction

The device embeds two ultra-low-power comparators COMP1 and COMP2.

These comparators can be used for a variety of functions including:

- Wake-up from low-power mode triggered by an analog signal
- Analog signal conditioning
- Cycle-by-cycle current control loop when combined with a PWM output from a timer

33.2 COMP main features

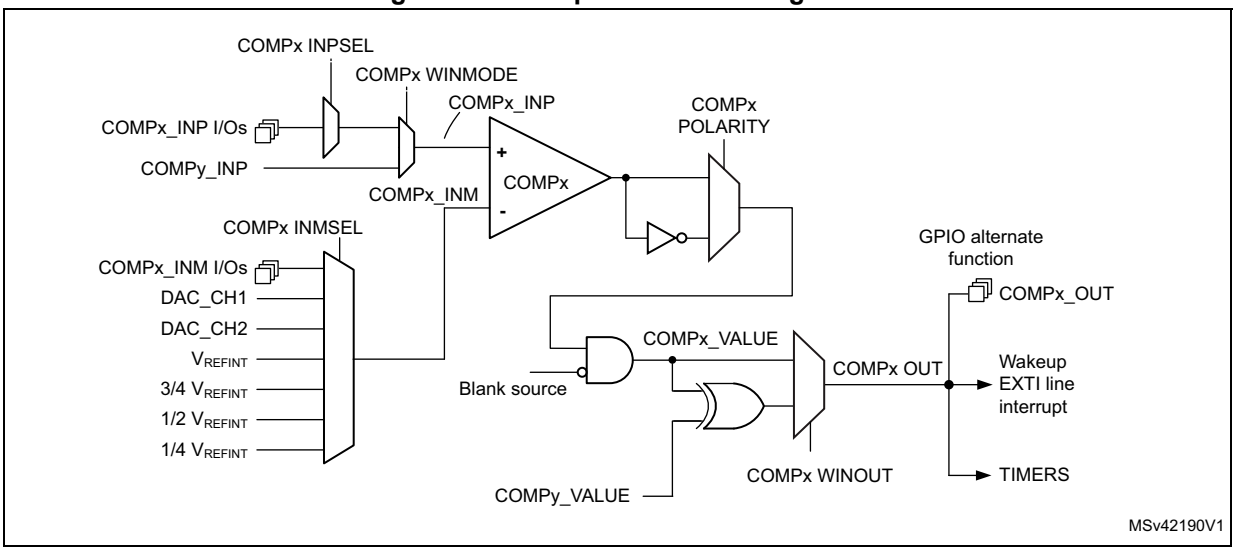
- Each comparator has configurable plus and minus inputs used for flexible voltage selection:
  - Multiplexed I/O pins
  - DAC channel1 and channel2
  - Internal reference voltage and three submultiple values (1/4, 1/2, 3/4) provided by a scaler (buffered voltage divider)
- Programmable hysteresis
- Programmable speed/consumption
- Outputs that can be redirected to an I/O or to timer inputs for triggering break events for fast PWM shutdowns
- Comparator outputs with blanking source
- Comparators that can be combined as a window comparator
- Interrupt generation capability for each comparator with wake-up from Sleep and Stop modes (through the EXTI controller)

33.3 COMP functional description

33.3.1 COMP block diagram

The block diagram of the comparators is shown in the figure below.

Figure 227. Comparator block diagrams



33.3.2 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

The comparator output can be simultaneously redirected internally and externally.

Table 277. COMP1 non-inverting input assignment

COMP1_INP	COMP1_INPSEL[1:0]
COMP1_INP1	00
COMP1_INP2	01
COMP1_INP3	10
Reserved	11

Table 278. COMP1 inverting input assignment

COMP1_INM	COMP1_INMSEL[3:0]
¼ V_{REFINT}	0000
½ V_{REFINT}	0001
¾ V_{REFINT}	0010
V_{REFINT}	0011
DAC Channel1	0100
DAC Channel2	0101
COMP1_INM1	0110
COMP1_INM2	0111
Reserved	≥ 1000

Table 279. COMP2 non-inverting input assignment

COMP2_INP	COMP2_INPSEL[1:0]
COMP2_INP1	00
COMP2_INP2	01
Reserved	10
Reserved	11

Table 280. COMP2 inverting input assignment

COMP2_INM	COMP2_INMSEL[3:0]
¼ V_{REFINT}	0000
½ V_{REFINT}	0001
¾ V_{REFINT}	0010
V_{REFINT}	0011
DAC Channel1	0100
DAC Channel2	0101
COMP2_INM1	0110
COMP2_INM2	0111
Reserved	≥ 1000

Table 281. COMP1 output-blanking PWM assignment

PWM output	COMP1_BLANKSEL[4:0]
None (no blanking)	00000
tim1_oc5	xxxx1
tim2_oc3	xxx1x

Table 281. COMP1 output-blanking PWM assignment (continued)

PWM output	COMP1_BLANKSEL[4:0]
tim3_oc3	xx1xx
Reserved	Others

Table 282. COMP2 output-blanking PWM assignment

PWM output	COMP2_BLANKSEL[4:0]
None (no blanking)	00000
tim3_oc4	xxxx1
tim8_oc5	xxx1x
tim15_oc1	xx1xx
Reserved	Others

33.3.3 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, the comparator programming must not 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 COMPxLOCK bit can be set to 1. This causes the whole COMPx_CSR register to become read-only, including the COMPxLOCK bit.

The write protection can only be reset by a MCU reset.

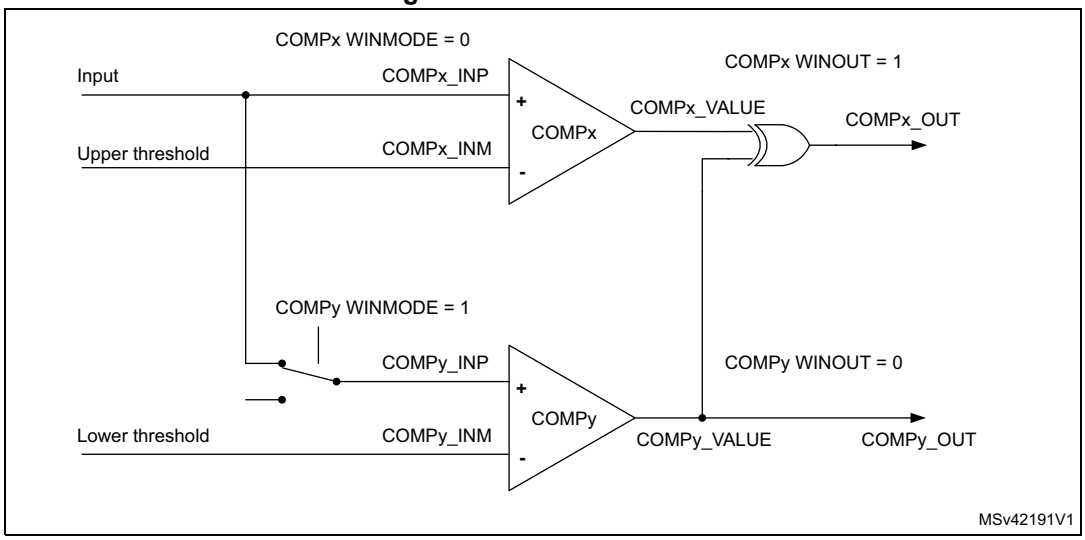
33.3.4 Window comparator

The purpose of window comparator is to monitor the analog voltage if it is within the voltage range defined by lower and upper threshold.

The two embedded comparators can be used to create a window comparator. The monitored analog voltage is connected to the non-inverting (plus) inputs of comparators connected together. 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.

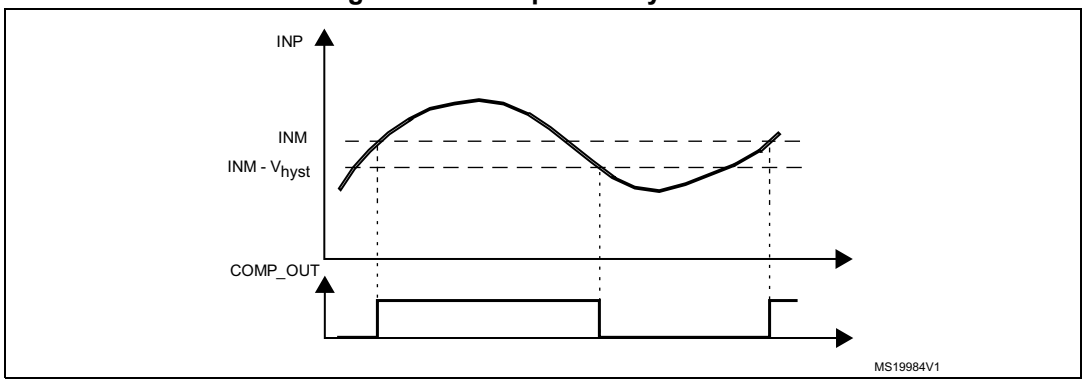
Figure 228. Window mode



33.3.5 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 a low-power mode) to be able to force the hysteresis value using external components.

Figure 229. Comparator hysteresis

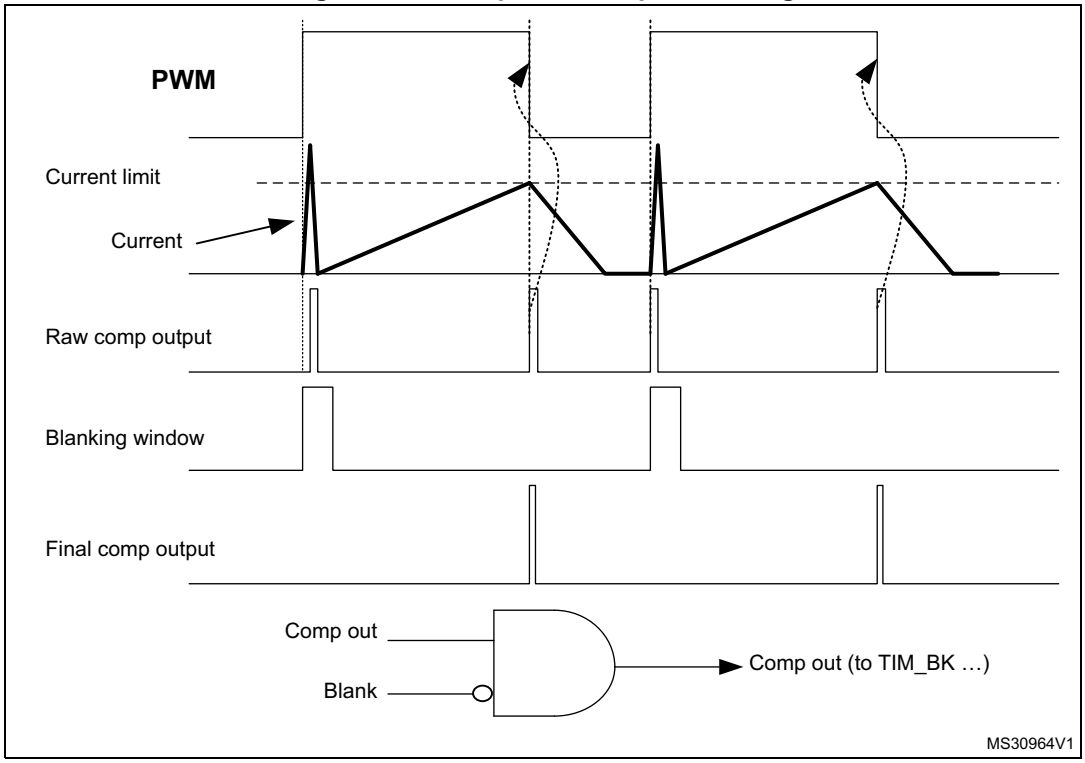


33.3.6 Comparator output-blanking function

The blanking function prevents 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). This blanking function consists of a selection of a blanking window that is a timer output compare signal. The selection is done by software (refer to the comparator register description for possible blanking signals).

The complementary of the blanking signal is ANDed with the comparator output to provide the wanted comparator output (see the example in the figure below).

Figure 230. Comparator output blanking



33.3.7 COMP power and speed modes

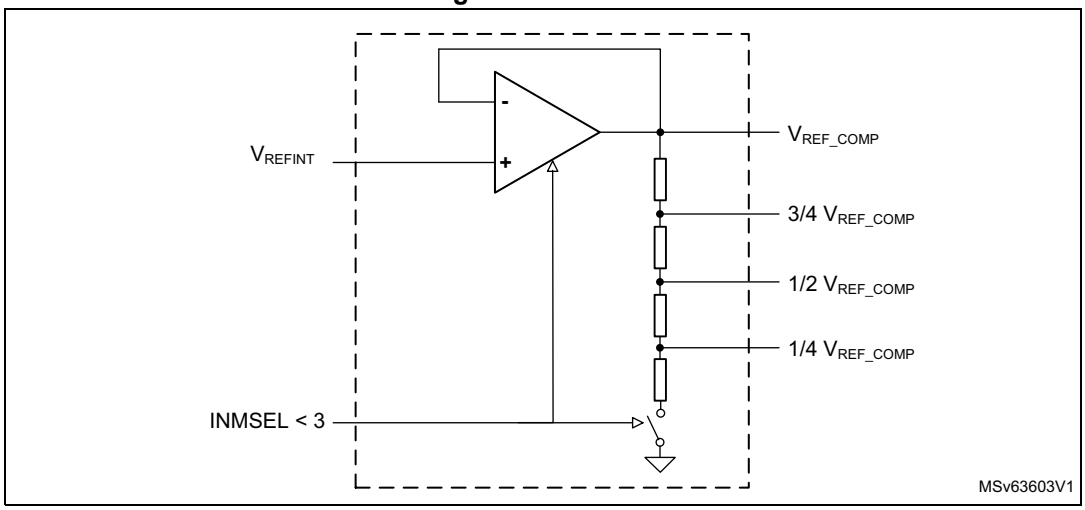
COMP1 and COMP2 power consumption versus propagation delay can be adjusted to have the optimum trade-off for a given application.

33.3.8 Scaler function

The scaler block provides the different voltage reference levels to the comparator inputs. This block is based on an amplifier driving a resistor bridge. The amplifier input is connected to the internal voltage reference.The amplifier and the resistor bridge are enabled by setting the INMSEL value in the COMP_CFGRx registers, to connect the corresponding inverting input to the scaler output.

When the resistor divided voltage is not used, the resistor bridge and the amplifier are disabled to reduce the consumption. When the resistor bridge is disconnected, the 1/4 VREF_COMP, 1/2 VREF_COMP and 3/4 VREF_COMP levels are equal to VREF_COMP.

Figure 231. Scaler



33.4 COMP low-power modes

Table 283. Comparator behavior in the low-power modes

Mode	Description
Sleep	No effect on the comparators. Comparator interrupts cause the device to exit Sleep mode.
Stop	No effect on the comparators. Comparator interrupts cause the device to exit Stop mode.
Standby	The COMP registers are powered down and must be reinitialized after exiting Standby mode.

33.5 COMP interrupts

The comparator outputs are internally connected to the extended interrupts and events controller (EXTI). Each comparator has its own EXTI line and can generate either interrupts or events. The same mechanism is used to exit the low-power modes.

Refer to *Section 21: Extended interrupts and event controller (EXTI)*for more details.

To enable the COMPx interrupt, 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 the COMPx.

Table 284. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Exit Sleep mode	Exit Stop modes	Exit Standby mode
COMP1 output	VALUE in COMP1_CSR	through EXTI	Yes	Yes	No
COMP2 output	VALUE in COMP2_CSR	through EXTI	Yes	Yes	No

33.6 COMP registers

33.6.1 COMP1 control and status register (COMP1_CSR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	BLANKSEL[4:0]					PWRMODE[1:0]		HYST[1:0]	
rw	r						rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POLARITY	WINOUT	Res.	Res.	WINMODE	Res.	INPSEL[1:0]		INMSEL[3:0]				Res.	Res.	Res.	EN
rw	rw			rw		rw	rw	rw	rw	rw	rw				rw

Bit 31 **LOCK:** COMP1_CSR register lock

This bit is set by software and cleared by reset. It locks the whole content of COMP1_CSR.

0: COMP1_CSR read/write bits can be written by software.

1: COMP1_CSR bits can be read but not written by software.

Bit 30 **VALUE:** COMP1 output status

This bit is read-only. It reflects the level of the COMP1 output after the polarity selector and blanking (see *Figure 230*).

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

Bits 24:20 **BLANKSEL[4:0]:** COMP1 blanking source selector

This field is controlled by software (if not locked) and selects the PWM signal for comparator output-blanking (see *Table 281* for the assignment).

Bits 19:18 **PWRMODE[1:0]:** COMP1 power mode selector

This bitfield is controlled by software (if not locked). It selects the power consumption and, as a consequence, the speed of the COMP1.

00: High speed

01, 10: Medium speed and power

11: Ultra-low-power

Bits 17:16 **HYST[1:0]:** COMP1 hysteresis selector
This bitfield is controlled by software (if not locked). It selects the COMP1 hysteresis.
00: None
01: Low hysteresis
10: Medium hysteresis
11: High hysteresis

Bit 15 **POLARITY:** COMP1 polarity selector
This bit is controlled by software (if not locked). It selects the COMP1 output polarity.
0: Non-inverted
1: Inverted

Bit 14 **WINOUT:** COMP1 output selector
This bit is controlled by software (if not locked). It selects the COMP1 output.
0: COMP1_VALUE
1: COMP1_VALUE XOR COMP2_VALUE (required for window mode, see *Figure 228*)

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WINMODE**: COMP1 non-inverting input selector for window mode
This bit is controlled by software (if not locked). It selects the signal for COMP1_INP input of the COMP1.
0: Signal selected with INPSEL[1:0]
1: COMP2_INP signal of COMP2 (required for window mode, see *Figure 228*)

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **INPSEL[1:0]**: COMP1 signal selector for non-inverting input
This field is controlled by software (if not locked). It selects the signal for the non-inverting input COMP1_INP (see *Table 277* for the assignment).

Bits 7:4 **INMSEL[3:0]**: COMP1 signal selector for inverting input INM
This field is controlled by software (if not locked). It selects the signal for the inverting input COMP1_INM (see *Table 278* for the assignment).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **EN:** COMP1 enable
This bit is controlled by software (if not locked). It enables COMP1.
0: COMP1 disabled
1: COMP1 enabled

33.6.2 COMP2 control and status register (COMP2_CSR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	BLANKSEL[4:0]					PWRMODE[1:0]		HYST[1:0]	
rw	r						rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POLARITY	WINOUT	Res.	Res.	WINMODE	Res.	INPSEL[1:0]		INMSEL[3:0]				Res.	Res.	Res.	EN
rw	rw			rw		rw	rw	rw	rw	rw	rw				rw

Bit 31 **LOCK:** COMP2_CSR register lock

This bit is set by software and cleared by reset. It locks the whole content of COMP2_CSR.

0: COMP2_CSR read/write bits can be written by software.

1: COMP2_CSR bits can be read but not written by software.

Bit 30 **VALUE:** COMP2 output status

This bit is read-only. It reflects the level of the COMP2 output after the polarity selector and blanking (see *Figure 230*).

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

Bits 24:20 **BLANKSEL[4:0]:** COMP2 blanking source selector

This field is controlled by software (if not locked) and selects the PWM signal for comparator output-blanking (see *Table 282* for the assignment).

Bits 19:18 **PWRMODE[1:0]:** COMP2 power mode selector

This bitfield is controlled by software (if not locked). It selects the power consumption and, as a consequence, the speed of the COMP2.

00: High speed

01, 10: Medium speed and power

11: Ultra-low-power

Bits 17:16 **HYST[1:0]:** COMP2 hysteresis selector

This bitfield is controlled by software (if not locked). It selects the COMP2 hysteresis.

00: None

01: Low hysteresis

10: Medium hysteresis

11: High hysteresis

Bit 15 **POLARITY:** COMP2 polarity selector

This bit is controlled by software (if not locked). It selects the COMP2 output polarity.

0: Non-inverted

1: Inverted

Bit 14 **WINOUT:** COMP2 output selector
This bit is controlled by software (if not locked). It selects the COMP2 output.
0: COMP2_VALUE
1: COMP1_VALUE XOR COMP2_VALUE (required for window mode, see *Figure 228*)

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WINMODE**: COMP2 non-inverting input selector for window mode
This bit is controlled by software (if not locked). It selects the signal for COMP2_INP input of the COMP2.
0: Signal selected with INPSEL[1:0]
1: COMP1_INP signal of COMP1 (required for window mode, see *Figure 228*)

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **INPSEL[1:0]**: COMP2 signal selector for non-inverting input
This field is controlled by software (if not locked). It selects the signal for the non-inverting input COMP2_INP (see *Table 279* for the assignment).

Bits 7:4 **INMSEL[3:0]**: COMP2 signal selector for inverting input INM
This field is controlled by software (if not locked). It selects the signal for the inverting input COMP2_INM (see *Table 280* for the assignment).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **EN:** COMP2 enable
This bit is controlled by software (if not locked). It enables COMP2.
0: COMP2 disabled
1: COMP2 enabled

33.6.3 COMP register map

Table 285. COMP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	COMP1_CSR	LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	BLANKSEL[4:0]					PWRMODE[1:0]		HYST[1:0]		POLARITY	WINOUT	Res.	Res.	WINMODE	Res.	INPSEL[1:0]		INPSEL[3:0].				Res.	Res.	Res.	EN
	Reset value	0	0						0	0	0	0	0	0	0	0	0	0	0			0		0	0	0	0	0	0				0
0x04	COMP2_CSR	LOCK	VALUE	Res.	Res.	Res.	Res.	Res.	BLANKSEL[4:0]					PWRMODE[1:0]		HYST[1:0]		POLARITY	WINOUT	Res.	Res.	WINMODE	Res.	INPSEL[1:0]		INPSEL[3:0].				Res.	Res.	Res.	EN
	Reset value	0	0						0	0	0	0	0	0	0	0	0	0	0			0		0	0	0	0	0	0				0

Refer to *Section 2.3* for the register boundary addresses.

34 Operational amplifier (OPAMP)

34.1 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.

34.2 OPAMP main features

- Rail-to-rail input and output voltage range
- Low input bias current
- Low input offset voltage
- Low-power mode
- High-speed mode to achieve a better slew rate
- Fast wakeup time
- Gain bandwidth of 1.6 MHz

34.3 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 three functional modes: the low-power mode, the high-speed mode and the normal mode. In functional mode the inputs and output of the OPAMP are connected as described in the *Section 34.3.3: Signal routing*.

34.3.1 OPAMP reset and clocks

The operational amplifier clock is necessary for accessing the registers. When the application does not need read or write access to those registers, the clock can be switched off using the peripheral clock enable register (see OPAMPEN bit in *RCC APB3 peripheral clock enable register (RCC_APB3ENR)*).

The OPAEN bit in OPAMPx_CSR enables and disables the OPAMP operation. The OPAMP registers configurations can be changed when the OPAEN bit is set in OPAMPx_CSR. However it can create spurious effects (noise, glitch, overshoot or saturation). If the configuration is changed, the application firmware must take care of these spurious effects (such as ignore the ADC result on the OPAMP output).

When the OPAMP output is no more needed, the OPAMP can be disabled to save power. All the configurations previously set (including the calibration) are maintained while the OPAMP is disabled.

34.3.2 Initial configuration

The OPAMP default configuration is a functional mode where the three I/Os are connected to external pins. In the default mode, the OPAMP 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 34.3.5: Calibration*). The default configuration uses the normal mode, that provides the standard performance. The OPALPM bit in OPAMPx_CSR can be set in order to switch the OPAMP to low-power mode and reduced performance. Normal, low-power and high-speed modes characteristics are defined in the “electrical characteristics” section of the datasheet. Before utilization, the OPA_RANGE bit in OPAMP1_CSR must be set to 1.

As soon as the OPAEN bit in OPAMPx_CSR is set, the OPAMP is functional. The two input pins and the output pin are connected as defined in *Section 34.3.3: 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.

34.3.3 Signal routing

The routing for the OPAMP pins is determined by the OPAMPx_CSR register.

The connections of the two operational amplifiers (OPAMP1 and OPAMP2) are described in the table below

Table 286. Operational amplifier possible connections

Signal	Pin	Internal	comment
OPAMP1_VINM	PA1 or dedicated pin(1)	OPAMP1_OUT or PGA	Controlled by 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 the ADC.
OPAMP2_VINM	PA7 or dedicated pin(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 ADC4_IN18	The pin is connected when the OPAMP is enabled. The ADC input is controlled by the ADC.

1. The dedicated pin is only available on BGA132 and BGA169 packages. This configuration provides the lowest input bias current (see datasheet).

34.3.4 OPAMP modes

The OPAMP inputs and outputs are all accessible on terminals. The amplifiers can be used in the following configuration environments:

- Standalone mode (external gain setting mode)
- Follower configuration mode
- PGA modes

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.

The impedance of the signal must be maintained below a level that avoids the input leakage to create significant artifacts (due to a resistive drop in the source). Refer to the “electrical characteristics” section in the datasheet for further details.

Standalone mode (external gain setting mode)

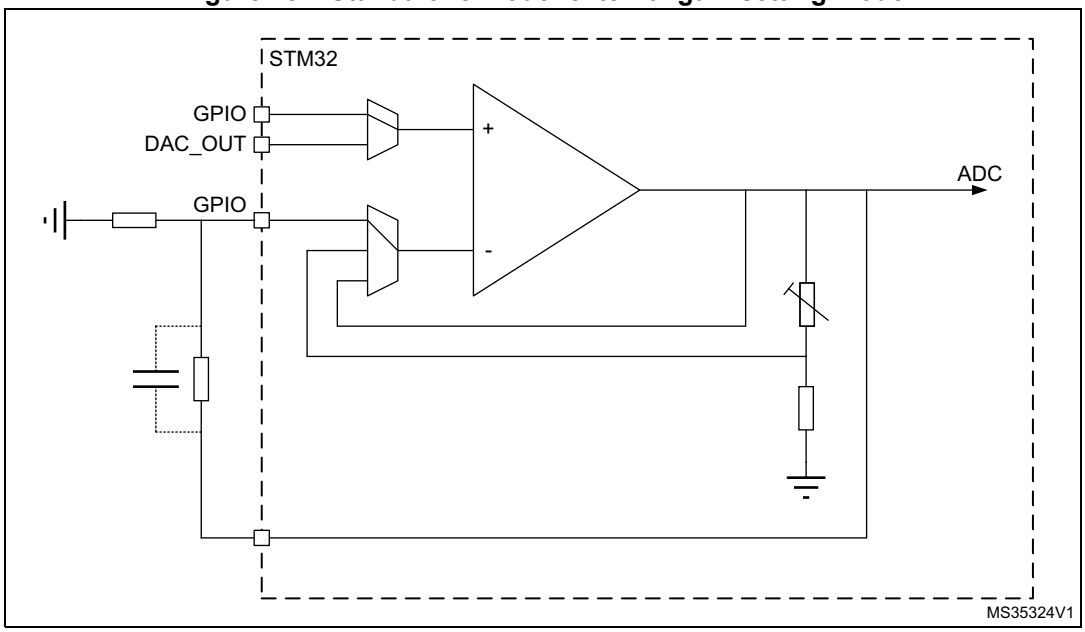
The procedure to use the OPAMP in standalone mode is detailed below:

1. Keep default value of OPAMPx_CSR and the default state of GPIOx_MODER.
2. As soon as OPAEN is set in OPAMPx_CSR, 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 (standard performance). The OPAMP behavior can be changed with the following bits in OPAMPx_CSR:

- If OPALPM is set to 1, the OPAMP switches in low-power mode in order to save power.
- If OPAHSM is set to 1, the OPAMP switches in high-speed mode in order to have high slew rate.
- If USERTRIM is set to 1, the input offset values can be trimmed.

Figure 232. Standalone mode: external gain setting mode



Follower configuration mode

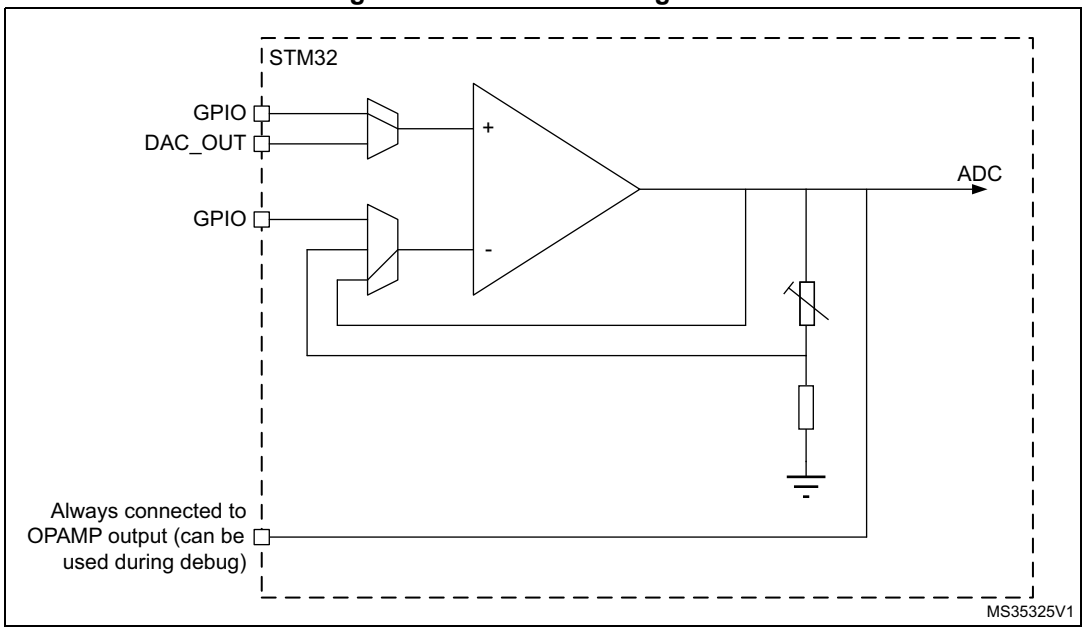
The procedure to use the OPAMP in follower mode is detailed below (all bits in OPAMPx_CSR):

1. Set OPAMODE[1:0] = 11 (internal follower).
2. Clear VP_SEL to 0 (GPIO connected to OPAMPx_VINP, named VINP in this document).
3. As soon as OPAEN is set to 1, the signal on the VINP pin is copied to the OPAMP_VOUT pin.

Note: The pin corresponding to OPAMP_VINM is free for another usage.

The signal on the OPAMP output is also seen as an ADC input. As a consequence, the OPAMP configured in follower mode can be used to perform impedance adaptation on input signals before feeding them to the ADC input, assuming the input signal frequency is compatible with the operational amplifier gain bandwidth specification.

Figure 233. Follower configuration



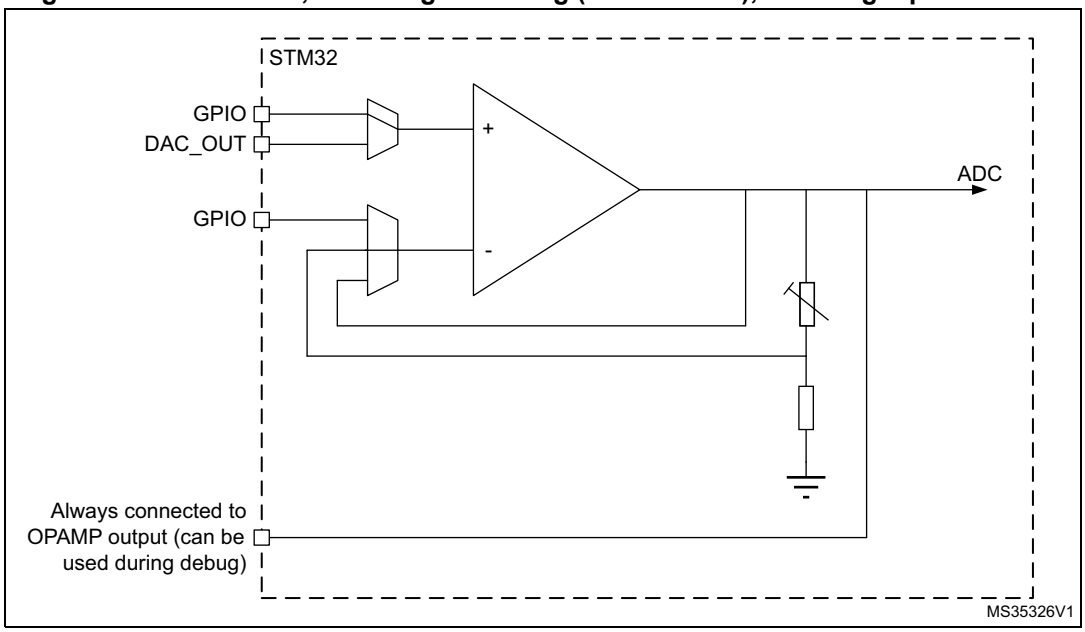
Programmable gain amplifier mode

The procedure to use the OPAMP to amplify the amplitude of an input signal is presented hereafter (all bits in OPAMPx_CSR):

1. Set OPAMODE[1:0] to 10 (internal PGA enabled).
2. Set PGA_GAIN[1:0] to the selected PGA gain (2, 4, 8 or 16) in OPAMPx_CSR.
3. Set VM_SEL[1:0] to 1x in OPAMPx_CSR (inverting input not externally connected).
4. Clear VP_SEL to 0 in OPAMPx_CSR (GPIO connected to VINP).
5. As soon as OPAEN is set in OPAMPx_CSR, the signal on the OPAMP_VINP pin is amplified by the selected gain and visible on the OPAMP_VOUT pin.

Note: To avoid saturation, the input voltage must stay below V_{DDA} divided by the selected gain.

Figure 234. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input not used



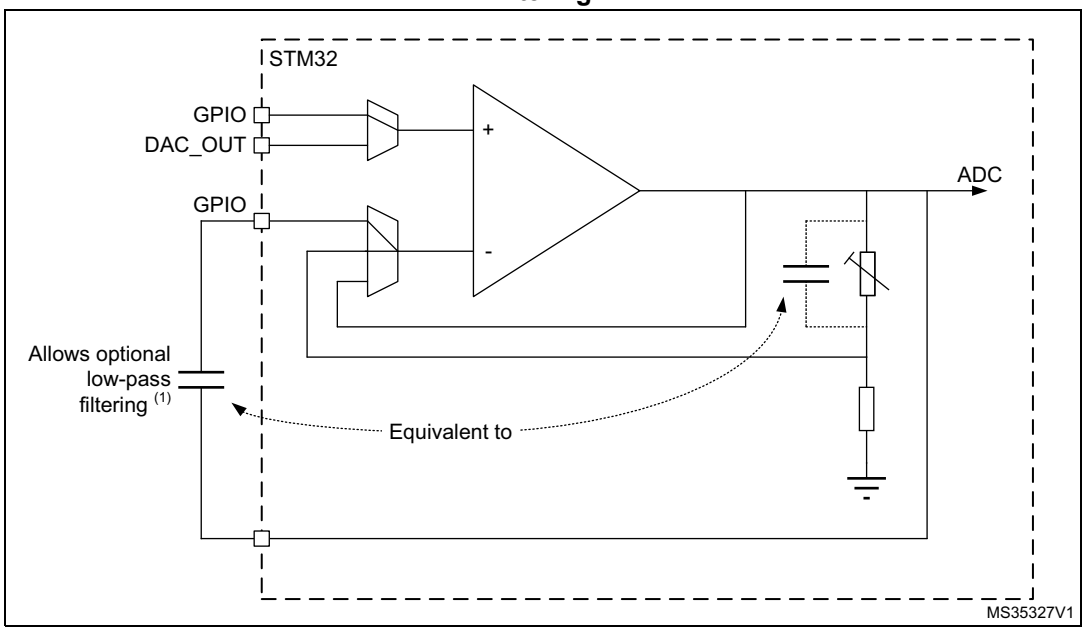
Programmable gain amplifier mode with external filtering

The procedure to use the OPAMP to amplify the amplitude of an input signal, with an external Set, is detailed below (all bits in OPAMPx_CSR):

1. Configure OPAMODE[1:0] to 10 (internal PGA enabled).
2. Set PGA_GAIN[1:0] to the selected PGA gain (2, 4, 8 or 16).
3. Clear VM_SEL[1:0] to 00 or 01(GPIO connected to VINM).
4. Clear VP_SEL to 0 (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 235. PGA mode, internal gain setting (x2/x4/x8/x16), inverting input used for filtering



1. The gain depends on the cut-off frequency.

34.3.5 Calibration

At startup, the trimming values are initialized with the preset ‘factory’ trimming value.

Each OPAMP offset can be trimmed by the user. Specific registers allow different trimming values for normal and low-power modes.

The calibration purpose is to cancel as much as possible the OPAMP inputs offset voltage. The calibration circuitry allows the inputs offset voltage to be reduced to less than +/- 1.5 mV within stable voltage and temperature conditions.

For each OPAMP and each mode, two trimming values can be trimmed: one for N differential pair and one for P differential pair.

The registers used to trim the offsets for each operational amplifiers are:

- OPAMPx_OTR for normal mode
- OPAMPx_LPOTR for low-power mode

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, with the USERTRIM bit in OPAMPx_CSR. This bit is reset at startup and the ‘factory’ value are applied by default to the OPAMP trimming registers.

The trimming values can be changed in calibration or in functional mode.

The offset trimming registers are typically configured after the initialization of the calibration operation (CALON set to 1 in OPAMPx_CSR). When CALON = 1, the OPAMP inputs are disconnected from the functional environment.

Setting CALSEL to 1 in OPAMPx_CSR initializes the offset calibration for the P differential pair (low-voltage reference used).

Clearing CALSEL to 0 initializes the offset calibration for the N differential pair (high voltage reference used).

When CALON = 1, the CALOUT bit in OPAMPx_CSR reflects 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).

When the calibration operation is done, OPAHSM must be cleared to 0 in OPAMPx_CSR.

Table 287. Operating modes and calibration

Mode	Control bits				Output	
	OPAEN	OPALPM	CALON	CALSEL	V_{OUT}	CALOUT
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. Set OPAEN to 1 in OPAMPx_CSR to enable the OPAMP and set OPA_RANGE = 1 in OPAMP1_CSR.
2. Clear OPAHSM to 0 in OPAMPx_CSR.
3. Set CALON and USERTRIM to 1 in OPAMPx_CSR.
4. Choose a calibration mode (refer to *Table 287*). Steps 4 to 5 must be repeated four times. For the first iteration, select normal mode, offset cal high (N differential pair), with OPALPM = 0 and CALSEL = 0 in OPAMPx_CSR.
5. Increment TRIMOFFSETN[4:0] in OPAMPx_OTR starting from 0, until CALOUT changes to 1 in OPAMPx_CSR.

   Between the write to OPAMP_OTR 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 OPAMP_OTR.
6. Repeat steps 4 to 5 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, the corresponding calibration can be skipped.

All OPAMPs can be calibrated at the same time.

Note: During the whole calibration phase:

- the external connection of the OPAMP output must not pull up or down currents higher than 500 µA.

- OPAMODE[1:0] must be set up to 00 or 01 (PGA disable) or 11 (internal follower).

34.4 OPAMP low-power modes

Table 288. Effect of CPU low-power modes on the OPAMP

Mode	Description
Sleep	No effect.
Stop	No effect, OPAMP registers content is kept.
Standby	The OPAMP registers are powered down and must be re-initialized after exiting Standby mode.

34.5 OPAMP registers

These registers are only accessible by word.(byte and half-word not supported)

34.5.1 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	OPAHSM	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
CALOUT	USERTRIM	CALSEL	CALON	Res.	VP_SEL	VM_SEL[1:0]		Res.	Res.	PGA_GAIN[1:0]		OPAMODE[1:0]		OPALPM	OPAEN
r	rw	rw	rw		rw	rw	rw			rw	rw	rw	w	rw	rw

Bit 31 **OPA_RANGE:** OPAMP range setting
This bit must be set before enabling the OPAMP and this bit affects all OPAMP instances.
0: reserved
1: OPAMP range set

Bit 30 **OPAHSM:** OPAMP high-speed mode
This bit is effective for both normal and low-power modes.
0: Normal mode (standard slew rate)
1: Increased consumption to improve the slew rate

Bits 29:16 Reserved, must be kept at reset value.

Bit 15 **CALOUT:** OPAMP calibration output
During the calibration mode, the offset is trimmed when this signal toggles.

Bit 14 **USERTRIM:** ‘factory’ or ‘user’ offset trimmed values selection
This bit is active for normal and low-power modes.
0: ‘factory’ trim code used
1: ‘user’ trim code used

Bit 13 **CALSEL:** Calibration selection
0: NMOS calibration (200 mV applied on OPAMP inputs)
1: PMOS calibration (V_{DDA} - 200 mV applied on OPAMP inputs)

Bit 12 **CALON:** Calibration mode enable
0: Normal mode
1: Calibration mode (all switches opened by hardware)

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[1:0]:** 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

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **PGA_GAIN[1:0]:** OPAMP 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[1:0]:** OPAMP PGA mode
00 and 01: internal PGA disabled
10: internal PGA enabled, gain programmed in PGA_GAIN
11: internal follower

Bit 1 **OPALPM:** OPAMP low-power mode
The OPAMP must be disabled to change this configuration.
0: normal mode
1: low-power mode

Bit 0 **OPAEN:** OPAMP enable
0: OPAMP disabled
1: OPAMP enabled

34.5.2 OPAMP1 offset trimming register in normal mode (OPAMP1_OTR)

Address offset: 0x04

Reset value: 0x0000 XXXX

XXXX are 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

34.5.3 OPAMP1 offset trimming register in low-power mode (OPAMP1_LPOTR)

Address offset: 0x08

Reset value: 0x0000 XXXX

XXXX are 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

34.5.4 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.	OPAHSM	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
CALOUT	USERTRIM	CALSEL	CALON	Res.	VP_SEL	VM_SEL[1:0]		Res.	Res.	PGA_GAIN[1:0]		OPAMODE[1:0]		OPALPM	OPAEN
r	rw	rw	rw		rw	rw	rw			rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **OPAHSM**: OPAMP high-speed mode
This bit is effective for both normal and high-speed modes.
0: Normal mode (standard slew rate)
1: Increased consumption to improve the slew rate

Bits 29:16 Reserved, must be kept at reset value.

Bit 15 **CALOUT**: OPAMP calibration output
During calibration mode, the offset is trimmed when this signal toggles.

Bit 14 **USERTRIM**: ‘factory’ or ‘user’ offset trimmed values selection
This bit is active for normal and low-power modes.
0: ‘factory’ trim code used
1: ‘user’ trim code used

Bit 13 **CALSEL**: Calibration selection
0: NMOS calibration (200 mV applied on OPAMP inputs)
1: PMOS calibration (V_{DDA} - 200 mV applied on OPAMP inputs)

Bit 12 **CALON**: Calibration mode enable
0: Normal mode
1: Calibration mode (all switches opened by hardware)

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[1:0]:** 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

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **PGA_GAIN[1:0]**: OPAMP 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[1:0]**: OPAMP PGA mode
00 and 01: internal PGA disabled
10: internal PGA enabled, gain programmed in PGA_GAIN
11: internal follower

Bit 1 **OPALPM**: OPAMP low-power mode
The OPAMP must be disabled to change this configuration.
0: normal mode
1: low-power mode

Bit 0 **OPAEN**: OPAMP enable
0: OPAMP disabled
1: OPAMP enabled

34.5.5 OPAMP2 offset trimming register in normal mode (OPAMP2_OTR)

Address offset: 0x14

Reset value: 0x0000 XXXX

XXXX are 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

34.5.6 OPAMP2 offset trimming register in low-power mode (OPAMP2_LPOTR)

Address offset: 0x18

Reset value: 0x0000 XXXX

XXXX are 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

34.5.7 OPAMP register map

Table 289. OPAMP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	OPAMP1_CSR	OPA_RANGE	OPAHSM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALOUT	USERTRIM	CALSEL	CALON	Res.	VP_SEL	VM_SEL[1:0]		Res.	Res.	PGA_GAIN[1:0]		OPAMODE[1:0]		OPALPM	OPAEN
	Reset value	0	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																				X	X	X	X	X				X	X	X	X	X
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																				X	X	X	X	X				X	X	X	X	X
0x10	OPAMP2_CSR	Res.	OPAHSM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALOUT	USERTRIM	CALSEL	CALON	Res.	VP_SEL	VM_SEL[1:0]		Res.	Res.	PGA_GAIN[1:0]		OPAMODE[1:0]		OPALPM	OPAEN
	Reset value		0															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																				X	X	X	X	X				X	X	X	X	X
0x18	OPAMP2_ 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																				X	X	X	X	X				X	X	X	X	X

Refer to *Section 2.3* for the register boundary addresses.

35 Multi-function digital filter (MDF)

35.1 Introduction

The multi-function digital filter (MDF) is a high-performance module dedicated to the connection of external sigma-delta (ΣΔ) modulators. It is mainly targeted for the following applications:

- audio capture signals
- motor control
- metering

The MDF features 6 digital serial interfaces (SITFx) and digital filters (DFLTx) with flexible digital processing options to offer up to 24-bit final resolution.

The DFLTx of the MDF also include the filters of the audio digital filter (ADF).

The MDF can receive, via its serial interfaces, streams coming from various digital sensors.

The MDF supports the following standards allowing the connection of various ΣΔ modulator sensors:

- SPI interface
- Manchester coded 1-wire interface
- PDM interface

A flexible bitstream matrix (BSMX) allows the connection of any incoming bitstream to any filter.

The MDF converts an input data stream into clean decimated digital data words. This conversion is done thanks to low-pass digital filters and decimation blocks. In addition it is possible to insert a high-pass filter or a DC offset correction block.

The conversion speed and resolution are adjustable according to configurable parameters for digital processing: filter type, filter order, decimation ratio, integrator length. The maximum output data resolution is up to 24 bits. There are two conversion modes: single conversion and continuous modes. The data can be automatically stored in a system RAM buffer through DMA, thus reducing the software overhead.

A flexible trigger interface can be used to control the conversion start. This timing control can trigger simultaneous conversions or insert a programmable delay between conversions.

The MDF features an out-off limit detectors (OLD) function. There is one OLD for each digital filter chain. Independent programmable thresholds are available for each OLD, making it very suitable for over-current detection.

A short circuit detector (SCD) is also available for every selected bitstream. The SCD is able to detect a short-circuit condition with a very short latency. Independent programmable thresholds are offered in order to define the short circuit condition.

The digital processing is performed using only the kernel clock. The MDF requests the bus interface clock (AHB clock) only when data must be transfered or when a specific event requests the attention of the system processor.

35.2 MDF main features

- AHB interface
- 6 serial digital inputs:
  - configurable SPI interface to connect various digital sensors
  - configurable Manchester coded interface support
  - compatible with PDM interface to support digital microphones
- 2 common clocks input/output for ΣΔ modulators
- Flexible matrix (BSMX) for connection between filters and digital inputs
- 2 inputs to connect the internal ADCs
- 6 flexible digital filter paths, including:
  - A configurable CIC filter:
    - Can be split into 2 CIC filters: high-resolution filter and out-off limit detector
    - Can be configured in $Sinc^4$ filter
    - Can be configured in $Sinc^5$ filter
    - Adjustable decimation ratio
  - A reshape filter to improve the out-off band rejection and in-band ripple
  - A high-pass filter to cancel the DC offset
  - An offset error cancellation
  - Gain control
  - Saturation blocks
  - An out-off limit detector
- Short-circuit detector
- Clock absence detector
- 16- or 24-bit signed output data resolution
- Continuous or single conversion
- Possibility to delay independently each bitstream
- Various trigger possibilities
- Break generation on out-of limit or short-circuit detector events
- Autonomous functionality in Stop modes
- DMA can be used to read the conversion data
- Interrupts services

35.3 MDF implementation

The devices embed one MDF instance and one ADF instance, both being digital filters with common features

Table 290. ADF/MDF features (1)

MDF modes/features	ADF1	MDF1
Number of filters (DFLTx) and serial interfaces (SITFx)	1	6
MDF_CKIy/ADF_CKI0 connected to pins	-	X
Sound activity detection (SAD)	X	-

Table 290. ADF/MDF features (1)

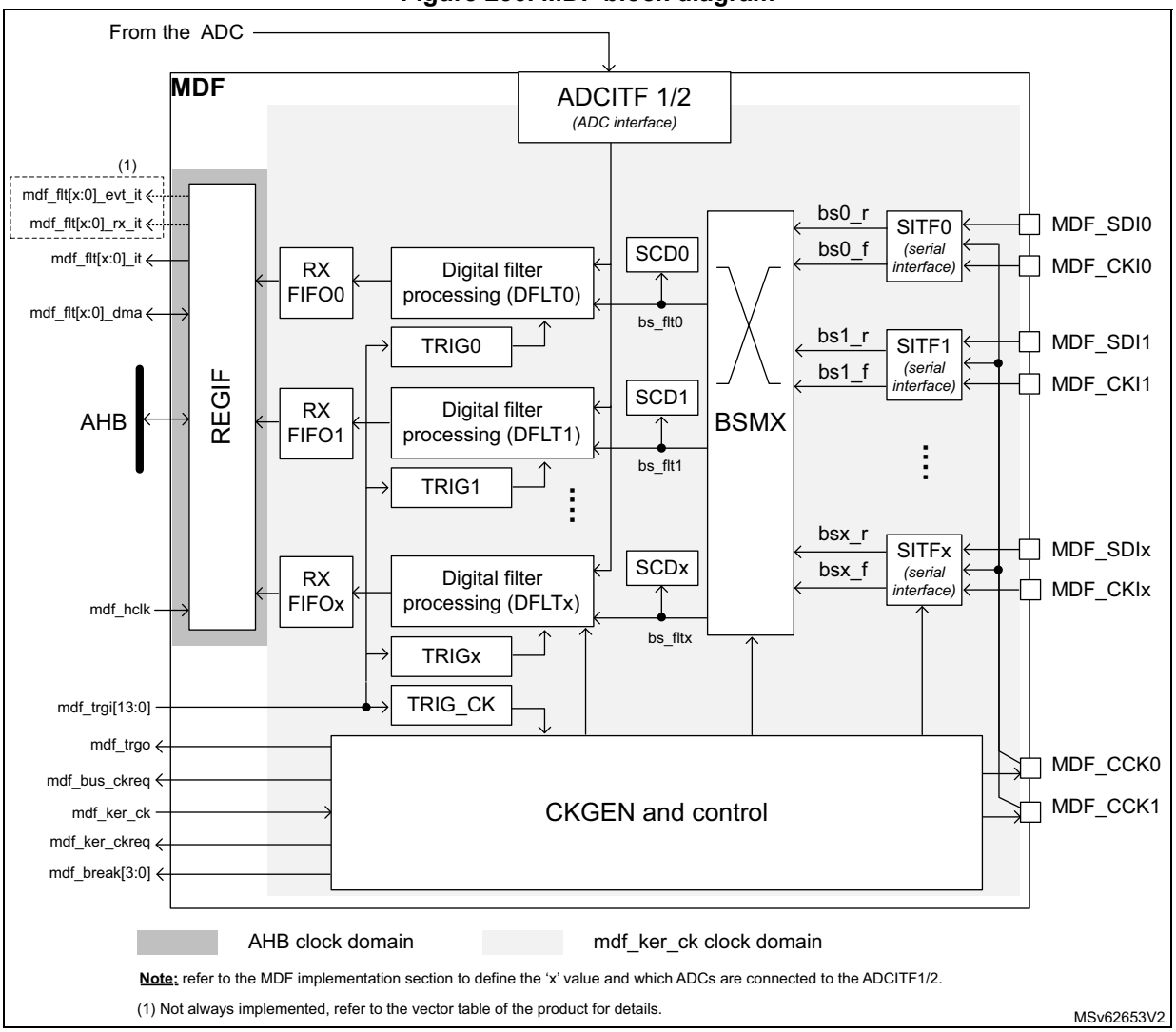
MDF modes/features	ADF1	MDF1
RXFIFO depth (number of 24-bit words)	4	4
ADC connected to ADCITF1	-	ADC1
ADC connected to ADCITF2	-	-
Motor dedicated features (SCD, OLD, OEC, INT, snapshot, break)	-	X
Main path with CIC4, CIC5	X	X
Main path with CIC1,2, 3 or FastSinc	-	X
RSFLT, HPF, SAT, SCALE, DLY, Discard functions	X	X
Autonomous in Stop modes	X(2)	X(3)

1. 'X' = supported, '-' = not supported.
2. Only Stop 0, Stop 1 and Stop 2 modes.
3. Only Stop 0 and Stop 1 modes.

35.4 MDF functional description

35.4.1 MDF block diagram

Figure 236. MDF block diagram



35.4.2 MDF pins and internal signals

Table 291. MDF external pins

Name	Signal type	Comment
MDF_CKIy (y = 0..5)	Input	Dedicated clock signals from external sensors
MDF_SDIy (y = 0..5)	Input	Data signal from external sensors
MDF_CCKy (y = 0, 1)	Input/output	Clock outputs for external sensor, or common clock input from external sensors

Table 292. MDF internal signals

Name	Signal type	Comment
mdf_trgi[13:0]	Input	Trigger inputs in order to control the acquisition (see the next table for details)
mdf_trgo	Output	Trigger output for synchronizing several MDFs
mdf_break[3:0]	Output	Break signals event generation from over-current detector or short-circuit detector (see the next table for details)
mdf_flt[5:0]_dma	Input/output	DMA request/acknowledge signals for each filter processing chain
mdf_flt[5:0]_it	Output	Global interrupt signals, for each MDF filter
mdf_flt[5:0]_rx_it	Output	Receive interrupt signals, for each MDF filter. Not always connected. See the vector table for details.
mdf_flt[5:0]_evt_it	Output	Event interrupt signals, for each MDF filter. Not always connected. See the vector table for details.
mdf_bus_ckreq	Output	Bus interface clock request output
mdf_ker_ckreq	Output	Kernel clock request output
mdf_ker_ck	Input	Kernel clock input
mdf_hclk	Input	AHB bus interface clock input
mdf_adcitf1_dat[15:0]	Input	ADCITF1 data input
mdf_adcitf2_dat[15:0]	Input	ADCITF2 data input

The table below shows the way the trigger inputs of the MDF are connected.

Table 293. MDF trigger connections

Trigger name	Trigger source
mdf_trgi0	tim1_trgo
mdf_trgi1	tim1_trgo2
mdf_trgi2	tim8_trgo
mdf_trgi3	tim8_trgo2
mdf_trgi4	tim3_trgo
mdf_trgi5	tim4_trgo
mdf_trgi6	tim16_oc1
mdf_trgi7	tim6_trgo
mdf_trgi8	tim7_trgo
mdf_trgi9	adf1_sad_det (sound activity detection signal from ADF1)
mdf_trgi10	exti11
mdf_trgi11	exti15
mdf_trgi12	lptim1_ch1
mdf_trgi13	adf1_trgo signal from ADF1

The table below shows the way the break outputs of the MDF are connected.

Table 294. MDF break connections

Trigger name	Trigger source
mdf_break0	tim1_brk_cmp7
mdf_break1	tim1_brk2_cmp7
mdf_break2	tim8_brk_cmp7
mdf_break3	tim8_brk2_cmp7

The table below shows the way the ADC data are connected to the MDF.

Table 295. MDF ADC data connections

ADC data bus name	ADC source
mdf_adcitf1_dat[15:0]	adc1_dat
mdf_adcitf2_dat[15:0]	-

35.4.3 Serial input interfaces (SITF)

The SITFx input interfaces allow the connection of the external sensors to the digital filters, via the bitstream matrix (BSMX). The SITFx serial interface can be configured in the following modes:

- LF_MASTER SPI mode (low-frequency)
- normal SPI mode
- Manchester mode

The amount of SITFx instances is equal to the amount of filters.

The data from each serial interface can be routed to any filter in order to perform:

- the PDM to PCM conversion
- the out-off limit detection
- the short detection

The serial interfaces are enabled by setting the corresponding SITFEN bit to 1. Once the interface is enabled, it receives serial data from the external ΣΔ modulator.

Note: Before enabling the serial interface, the user must insure that the mdf_proc_ck is already enabled (see Section 35.4.5: Clock generator (CKGEN) for details).

The SITFx are controlled via the *MDF serial interface control register x (MDF_SITFxCR)*.

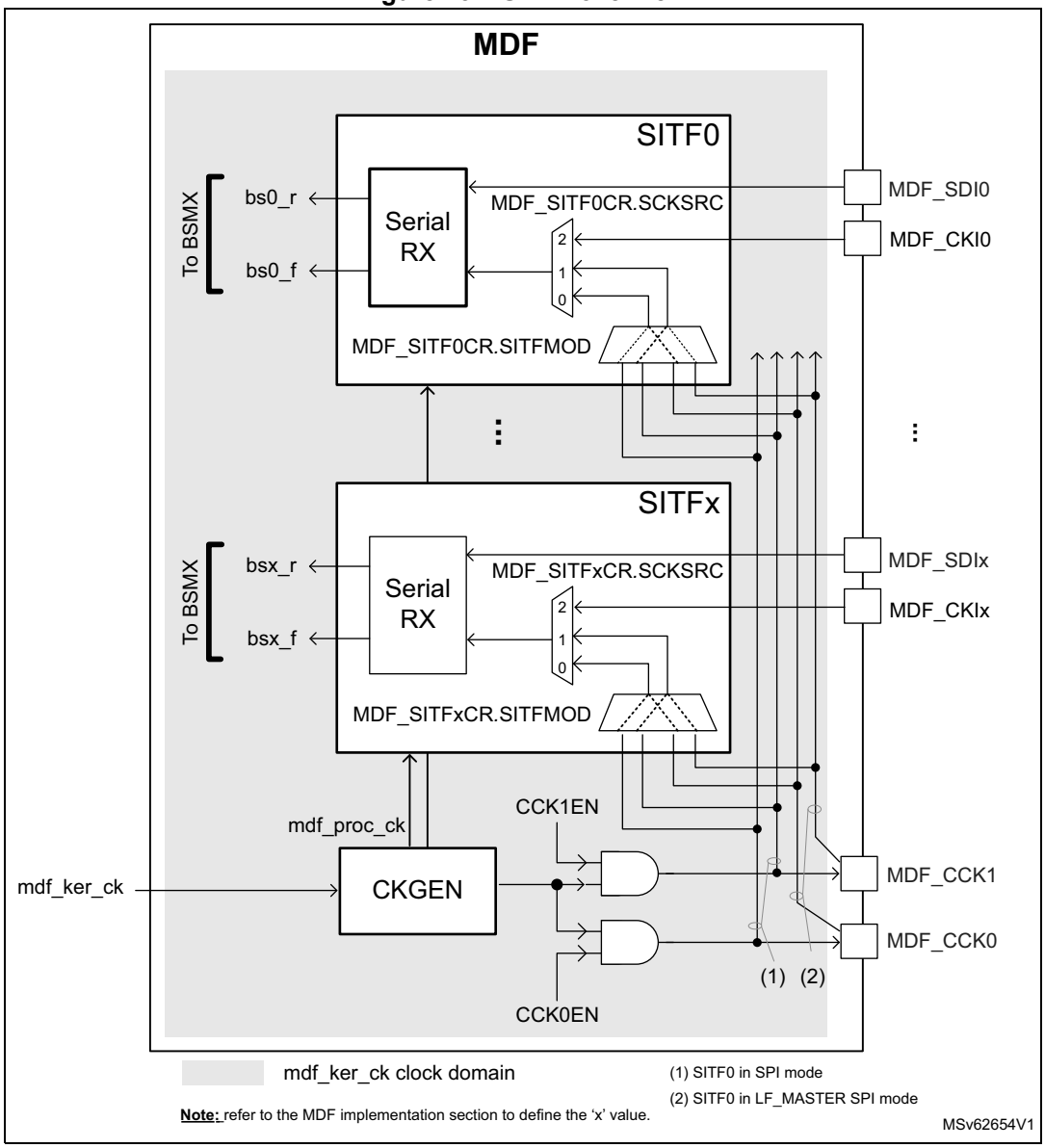
As shown in *Figure 237*, for each SITF, there is a large choice of clocking possibilities:

- If the serial interface is programmed in SPI mode, the selected clock source is a copy of the clock present on MDF_CCK0 or MDF_CCK1 or MDF_CKIx pin (see 2 in *Figure 237*).
- If the serial interface is programmed in LF_MASTER SPI mode, the selected clock source must be the clock directly provided by the CCKDIV to the MDF_CCK0 or MDF_CCK1 pin (see 1 in *Figure 237*). In this case MDF_CKIx must not be selected.

See *Table 296* for additional information.

Note: Using the common clock (MDF_CCK0 or MDF CCK1) can be helpful to share the same clock between several SITFx.

Figure 237. SITFx overview



LF_MASTER and normal SPI modes

The LF_MASTER SPI mode is a special mode allowing the usage of a mdf_proc_ck clock frequency, only two times higher than the sensor clock. This mode is dedicated to low-power use-cases, using low-speed sensors.

In LF_MASTER SPI mode, the MDF must provide the bitstream clock to the external sensors via MDF_CCK0 and MDF_CCK1 pins, and receives the bitstream data via the serial data input MDF_SDIx.

For each SITFx, the application must select the same clock than the one provided to the external sensor (MDF_CCK0 or MDF_CCK1), in order to guarantee optimal timing performances. This selection is done via SCKSRC[1:0].

Warning: The MDF_CKIx pin cannot be used in LF_MASTER SPI mode.

The normal SPI interface is a more flexible interface than the LF_MASTER SPI, but the mdf_proc_ck frequency must be at least four times higher than the sensor clock.

The application can select MDF_CCK0, MDF_CCK1 or MDF_CKIx clock for the capture of the data received via the MDF_SDIx pin.

The MDF can generate a clock to the sensors via MDF_CCK0 or MDF_CCK1 if needed.

For all SPI modes, all SITFs can share the same clock input (MDF_CCK0 or MDF_CCK1), in order to optimize the amount of requested I/Os.

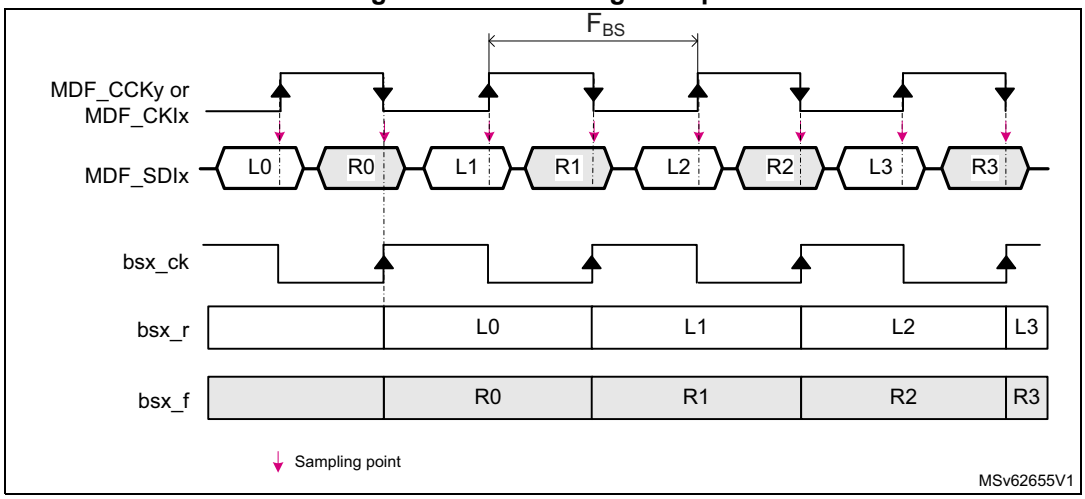
For all SPI modes, the serial data is captured using the rising and the falling edge of the selected clock. The SITFx always provides the following bitstreams:

- bitstream received using the bitstream clock falling edge (bsx_f)
- bitstream received using the bitstream clock rising edge (bsx_r)

According to the sensors connected, one of the two bitstreams may not be available.

The application can select the wanted stream via the BSMX matrix.

Figure 238. SPI timing example



To properly synchronize/receive the data stream, the frequency of the mdf_proc_ck clock must be adjusted according to the constraints listed in *Table 297*.

Clock absence detection

A no-clock-transition period may be detected when the serial interface works in normal SPI mode. This feature can be used to detect a clock failure in the SPI link.

The application can program a timeout value via the STH[4:0] field of the corresponding SITFx. If the MDF does not detect clock transitions for a duration of STH[4:0] x $T_{mdf\_proc\_ck}$, then the CKABF flag is set.

An interrupt can be generated if CKABIE is set to 1. The STH[4:0] field is in the *MDF serial interface control register x (MDF_SITFxCR)*.

When the serial interface is enabled, the CKABF flag remains to 1 until a first clock transition is detected.

To avoid spurious clock absence detection, the following sequence must be respected:

1. Configure the serial interface in normal SPI mode and enable it.
2. Clear the CKABF flag by writing CKABF bit to 1.

   If no clock transition is detected on the serial interface, the hardware immediately sets the CKABF flag to 1.
3. Read the CKABF flag:
   - If CKABF = 1, go back to step 2.
   - If CKABF i= 0, a clock has been detected. The CKABIE bit can be set to 1 if the application wants an interrupt on detection of a clock absence.

Note: The clock absence detection feature is not available in the LF_MASTER SPI mode.

Manchester mode

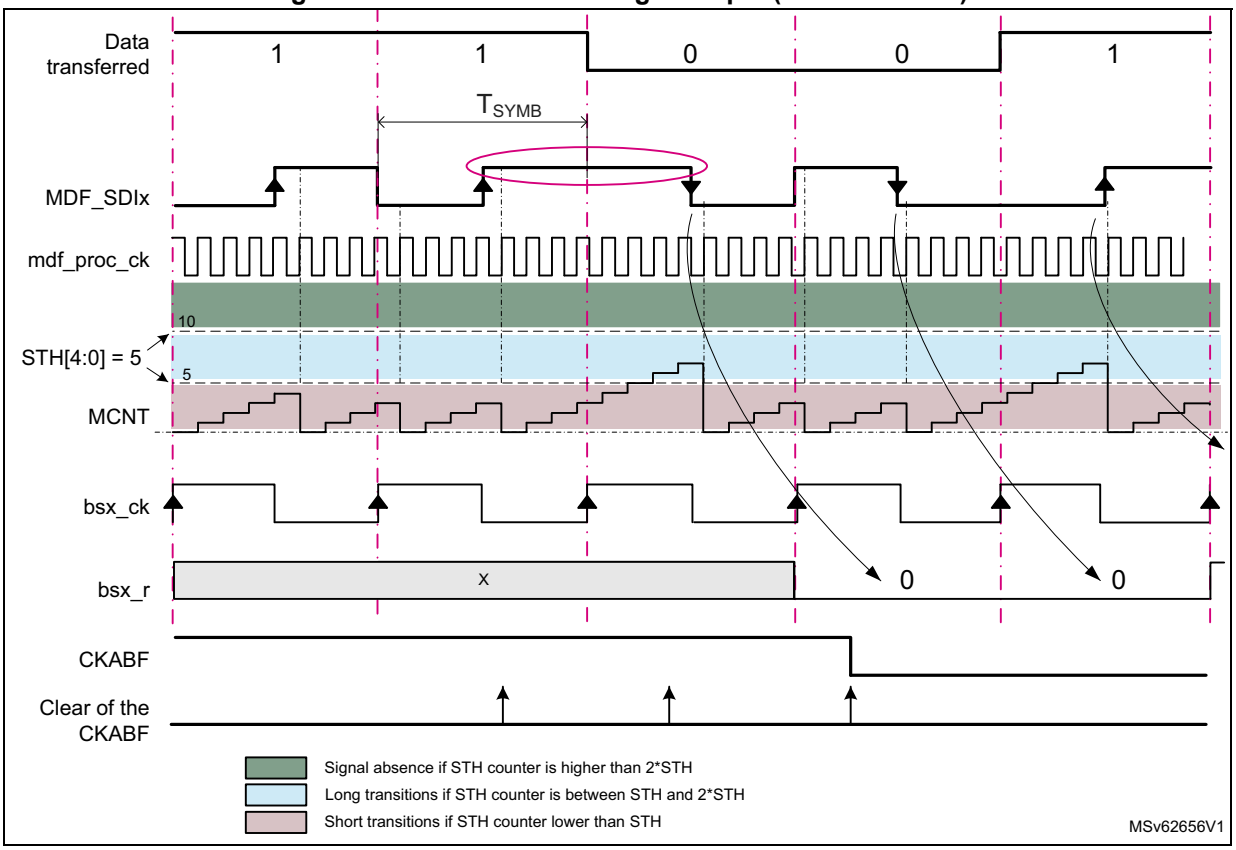
In Manchester coded format, the MDF receives data stream from the external sensor via the MDF_SDIx pin only.

The MDF_CKIx pins are not needed in this mode.

Decoded data and clock signals are recovered from serial stream after Manchester decoding. They are available on bsx_r. There are two possible settings of Manchester codings:

- signal rising edge decoded as 0 and signal falling edge decoded as 1
- signal rising edge decoded as 1 and signal falling edge decoded as 0

Figure 239. Manchester timing example (SITFMOD = 11)



To decode the incoming Manchester stream, the user must program the STH[4:0] field in the *MDF serial interface control register x (MDF_SITFxCR)*. The STH[4:0] field is used by the SITFx to estimate the Manchester symbol length and to detect a clock absence. An internal counter (MCNT) is restarted every time a transition is detected in the MDF_SDIx input. It is used to detect short transitions, long transitions or clock absence. A long transition indicates that the data value changed. *Figure 239* shows a case where the OVR is around height and STH[4:0] = 5.

The estimated Manchester symbol rate (T_{SYMB}) must respect the following formula:

$$(STH + 1)\times T_{mdf\_proc\_ck} < T_{SYMB} < (2 \times STH \times T_{mdf\_proc\_ck})$$

It is recommended to compute STH as follows:

$$STH[4:0] = round\left(\frac{(2 \times OVR) - 1}{3}\right)$$

where OVR represents the ratio between the mdf_proc_ck frequency and the expected Manchester symbol frequency. OVR must be higher than five, and the mdf_proc_ck clock must be adjusted according to the constraints listed in *Table 297*.

The clock absence flag CKABF is set to 1 when no transition is detected during more than 2 x STH[4:0] x $T_{mdf\_proc\_ck}$, or when the SITFx is not yet synchronized to the incoming Manchester stream. In addition, an interrupt can be generated if the bit CKABIE is set to 1.

When the serial interface is enabled, the MDF must first be synchronized to the incoming Manchester stream. The synchronization ends when a data transition from 0 to 1 or from 1 to 0 (pink circle in the *Figure 239*) is detected.

The end of the synchronization phase can be checked by following the software sequence:

1. Clear the CKABF flag in the *MDF DFLTx interrupt status register x (MDF_DFLTxISR)* by writing CKABF bit to 1. If the serial interface is not yet synchronized, the hardware immediately sets the CKABF flag to 1.
2. Read the CKABF flag.
   - If CKABF = 1, go back to step 1.
   - If CKABF = 0, the Manchester interface is synchronized and provides valid data.

Programming example

In the following example, the MDF kernel clock frequency ($F_{mdf\_ker\_ck}$) is 100 MHz and the received Manchester stream is at about 6 MHz (F_{SYMB}).

1. Provide a valid mdf_proc_ck to the SITFx.

   The mdf_proc_ck frequency must be at least six times higher than the Manchester symbol frequency (means at least 36 MHz).

   PROCDIV is programmed to 1 to perform a division by two of the kernel clock. In that case, $F_{mdf\_proc\_ck}$ = 50 MHz (8.33 times higher than the Manchester symbol frequency).
2. Compute STH.

   OVR is given by: OVR = $F_{mdf\_proc\_ck}$ / F_{SYMB} = 50 MHz / 6 MHz = 8.33.

   Then STH[4:0] = $\text{round}\left(\frac{(2 \times 8.33) - 1}{3}\right) = 5$

   The minimum allowed frequency for the Manchester stream is then:

   1 / (2 × STH × $T_{mdf\_proc\_ck}$) = 1 / (10 x 20 ns) = 5 MHz

   The maximum allowed frequency for the Manchester stream is then:

   1 / ((STH+1) × $T_{mdf\_proc\_ck}$) = 1 / (6 x 20 ns) = 8.33 MHz

35.4.4 ADC slave interface (ADCITF)

The ADCs are not always connected to the MDF. Refer to *Section 35.3: MDF implementation* to check the situation for this product.

The MDF allows the connection of up to two ADCs to the filter path. For each filter, the DATSRC[1:0] field in the *MDF digital filter configuration register x (MDF_DFLTxCICR)* allows the application to select either data from the ADCs.

Warning: The MDF does not support receiving interleaved data from one of the ADCITF input.

35.4.5 Clock generator (CKGEN)

The RCC (reset and clock controller) provides the following clocks to the MDF:

- AHB clock (mdf_hclk) used for the register interface
- kernel clock (mdf_ker_ck) mainly used by all other parts of the circuit via the CKGEN

Those clocks are not supposed to be phase locked, so all signals crossing those clock domains are re-synchronized.

The clock generator (CKGEN) is responsible of the generation of the processing clock, and the clock provided to the MDF_CCK0 and MDF_CCK1 pins. All those clocks are generated from the mdf_ker_ck.

The processing clock (mdf_proc_ck) is used to run all the signals processing and to re-sample the incoming serial or parallel stream.

Note: The reshape filter (RSFLT) needs up to 24 cycles of mdf_proc_ck clock to process a sample.

To adapt the kernel clock frequency provided by the RCC, the following dividers are available:

- PROCDIV[6:0] used to adapt the kernel clock frequency to the constraints of the parallel and serial interfaces, and to the processing blocks
- CCKDIV[3:0] used to adapt the frequency of the MDF_CCK0 and MDF_CCK1 clocks

PROCDIV[6:0] and CCKDIV[3:0] must be programmed when no clock is provided to the dividers (CKGDEN = 0).

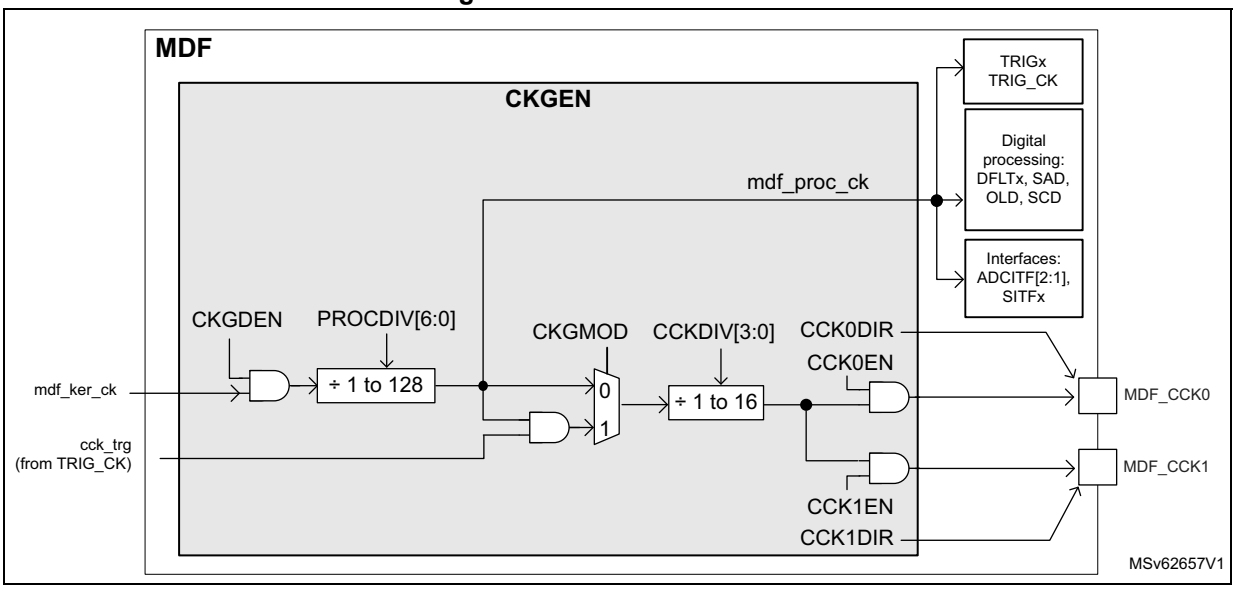
The mdf_proc_ck generation is controlled by CKGDEN.

In addition, the CKGMOD bit allows the application to define the way to trigger the CCKDIV divider:

- When CKGMOD = 0, the CCKDIV divider is started as soon as CKGDEN is set to 1.
- When CKGMOD = 1, the CCKDIV divider is started when CKGDEN is set to 1 and the programmed trigger condition occurred.

All bits and fields controlling the CKGEN are in the *MDF clock generator control register (MDF_CKGCR)*.

Figure 240. CKGEN overview



The trigger logic for CKGEN is handled by the TRG_CK block. As shown in *Figure 249*, the CCKDIV divider can be triggered on the rising or falling edge of one of the 16 trigger sources. When the proper trigger condition occurs, the cck_trg signal goes to high, allowing the CCKDIV divider to start. The TRG_CK logic is reset when CKGDEN is set to 0.

This feature can be helpful to synchronize the MDF_CCKy (y = 0,1) clock of several MDF instances, or to synchronize the clock generation to a timer event.

The application can control the activation of the MDF_CCK0 or MDF_CCK1 pin CCK0EN/CCK1EN and CCK0DIR/CCK1DIR bits:

- CCKyEN is used to enable the CCKDIV, and thus generates a clock for the external sensors.
- CCKyDIR is used to control the direction of the ADF_CCKy pin (input or output)

Table 296. Control of the common clock generation(1)

CCKyEN	CCKyDIR	Description
0	0	The MDF_CCKy pin is in input. An external clock can be connected to the MDF_CCKy pin and used by the SITFx in order to decode the serial stream
0	1	The MDF_CCKy pin is in output. No clock is generated, thus the MDF_CCKy pin is driven low.
1	1	The MDF_CCKy pin is in output. A clock is generated on the MDF_CCKy pin. The SITFx can use this pin as clock source in order to decode the serial stream

1. The configuration with CCKyEN = 1 and CCKyDIR = 0 must be avoided (no interest).

When CCKyDIR = 1, as soon as the CCKyEN bit is set to 1, a clock is generated to the corresponding output without any spurs.

Note: The mdf_proc_ck must be enabled (by CKGDEN = 1) before enabling other blocks (such as SITFx or DFLTx).

CKGEN activation sequence example

1. Set CKGDEN to 0.
2. Wait for CKGACTIVE = 0. If CKGDEN was previously enabled, this phase can take two periods of mdf_hclk and two periods of mdf_proc_ck.
3. Program PROCDIV[6:0], CKGMOD, CCKDIV[3:0], TRGSRC[3:0] and TRGSENS.
4. Set CKGDEN to 1.
5. Set CCKxDIR to 1 (optional).
6. Set CCKxEN to 1 (optional).

When needed, at any moment, the CCK0EN or CCK1EN value can be changed without disabling the clock generator.

Clock frequency constraints

The table below shows the frequency constraints to receive and process properly the samples.

Table 297. Clock constraints with respect to the incoming stream(1)

SITFx mode	MDF clock constraints	
	With RSFLT disabled	With RSFLT enabled
LF_MASTER SPI	F_{MDF_CCKy} max frequency limited to 5 MHz	
	$F_{mdf\_proc\_ck} > 2 * F_{MDF\_CCKy}$ and $F_{mdf\_hclk} \geq F_{mdf\_proc\_ck}$	$F_{mdf\_proc\_ck} > 24 * F_{MDF\_CCKy} / (MCICD+1)$ and $F_{mdf\_proc\_ck} > 2 * F_{MDF\_CCKy}$ and $F_{mdf\_hclk} \geq F_{mdf\_proc\_ck}$
MASTER SPI SLAVE SPI	F_{MDF_CKx} max frequency limited to 25 MHz	
	$F_{mdf\_proc\_ck} > 4 * F_{MDF\_CKKy}$ and F_{mdf_hclk} higher or equal to $F_{mdf\_proc\_ck}$	$F_{mdf\_proc\_ck} > 24 * F_{MDF\_CKKy} / (MCICD+1)$ and $F_{mdf\_proc\_ck}$ higher than 4 x F_{MDF_CKKy} and F_{mdf_hclk} higher or equal to $F_{mdf\_proc\_ck}$
Manchester	F_{SYMB} max frequency limited to 20 MHz	
	$F_{mdf\_proc\_ck}$ higher than 6 * F_{SYMB} and $F_{mdf\_hclk} \geq F_{mdf\_proc\_ck}$	$F_{mdf\_proc\_ck}$ higher than 24 * $F_{MDF\_CKKy} / (MCICD+1)$ and $F_{mdf\_proc\_ck} > 6 * F_{SYMB}$ and $F_{mdf\_hclk} \geq F_{mdf\_proc\_ck}$

1. F_{MDF_CKKy} represents the frequency of the clock received via MDF_CKIx and MDF_CCKy, or generated via MDF_CCKy.F_{SYMB} represents the frequency of the received symbol rate for Manchester mode.

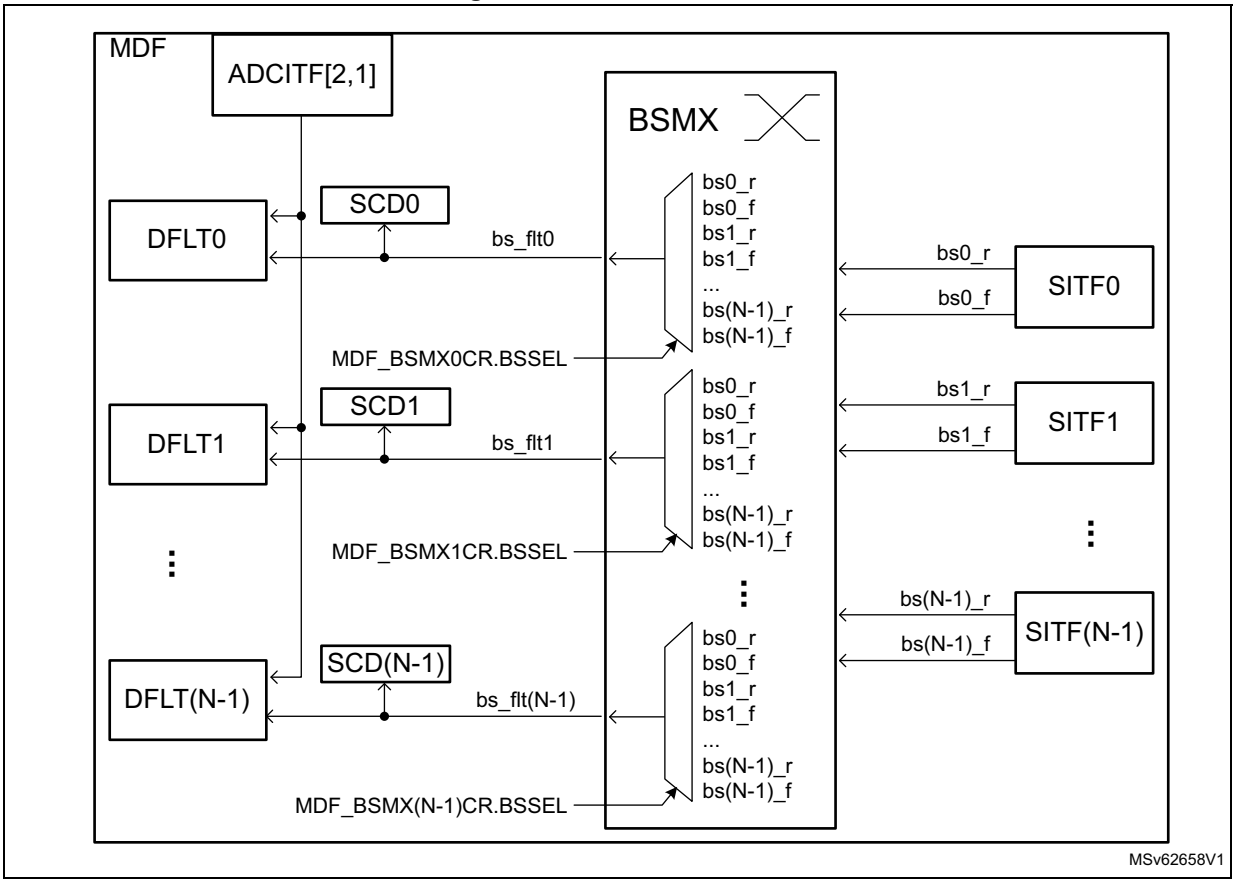
35.4.6 Bitstream matrix (BSMX)

The BSMX receives the bitstreams from all serial interfaces (SITFx) and provides the selected input to the digital filters (DFLTx) and to the short-circuit detectors (SCDx). For each filter path, any of the bitstream input can be selected.

As shown in the *Figure 237*, each SITFx block provides two bitstreams (bsx_r and bsx_f) to the BSMX.

The application to select the wanted stream via the *MDF bitstream matrix control register x (MDF_BSMXxCR)*. This selection is intended to be static.

Figure 241. BSMX overview



BSMX programming sequence example

The BSSEL[4:0] field cannot be changed if the corresponding SCDx, OLDx or DFLTx is enabled. The following steps are needed to change the value of BSMX, for filter path x:

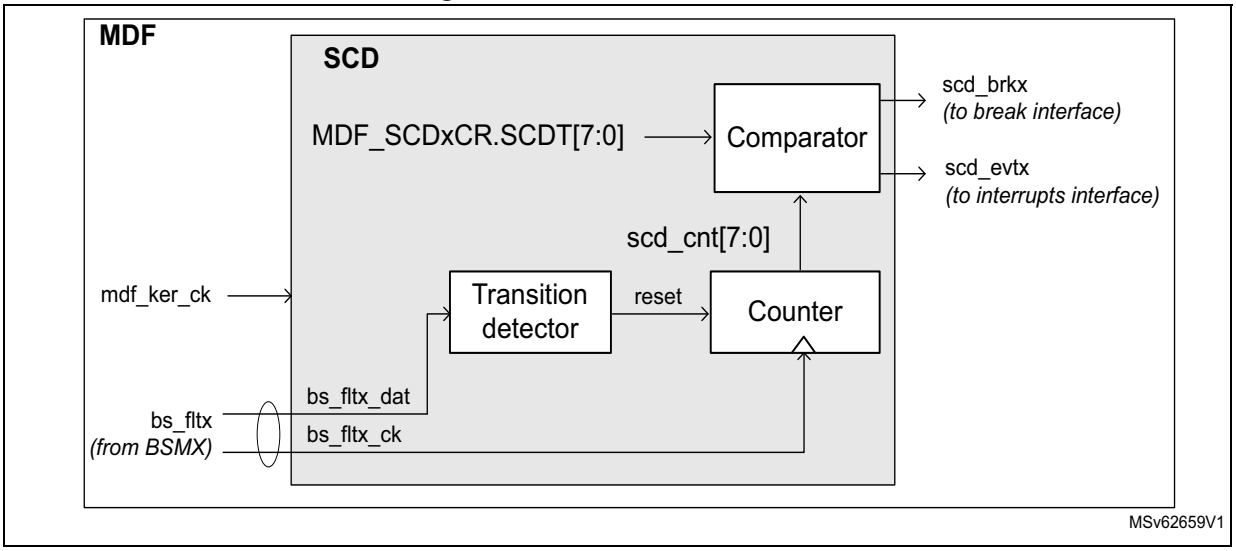
1. Set SCDEN of SCDx to 0.
2. Set DFLTEN of DFLTx to 0.
3. Set OLDEN of OLDx to 0.
4. Wait for BSMXACTIVE = 0.
5. Program BSSEL[4:0] of filter path x.
6. Set SCDEN of SCDx to 1 (optional).
7. Set DFLTEN of DFLTx to 1 (optional).
8. Set OLDEN of OLDx block to 1 (optional).

35.4.7 Short-circuit detectors (SCD)

The SCDx detects, with a very fast response time, if an analog signal reached saturated values (out-of-full scale ranges) and remained on this value for a given time.

This behavior can detect short-circuit or open-circuit errors (such as over current or over voltage). An interrupt event or/and a break signal can be generated.

Figure 242. SCD functional view

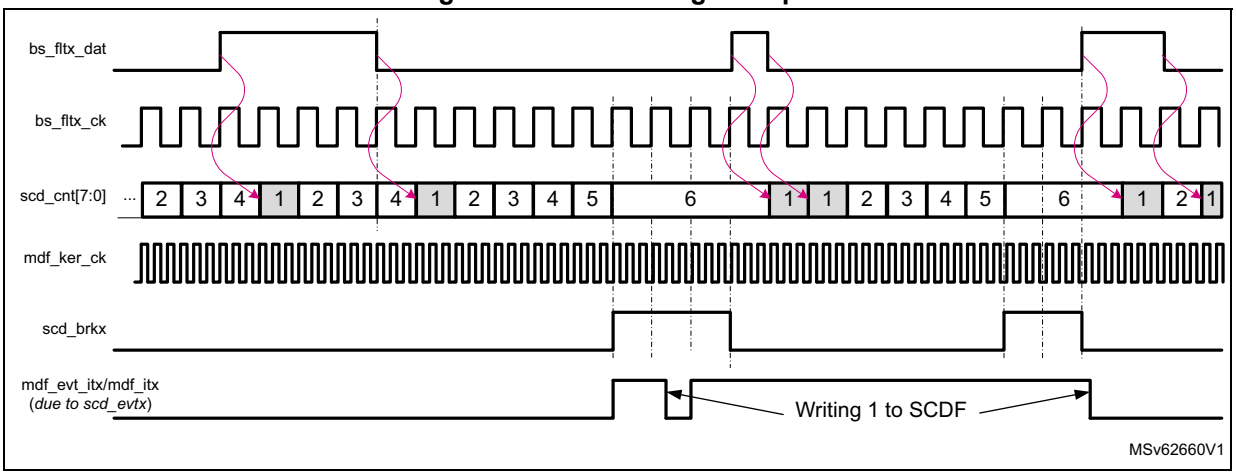


The SCDx is counting consecutive zeros or consecutive ones received from the serial interface (SITFx). A counter is restarted if there is a change in the data stream received. If this counter reaches a short-circuit threshold value (SCDT[7:0] in the *MDF short circuit detector control register x (MDF_SCDxCR)*), then a short-circuit event is invoked. Each BSMX output has its own short-circuit detector. Any BSMX output can be continuously monitored by setting the corresponding SCDEN bit to 1 in MDF_SCDxCR. Each SCD has its own threshold settings (SCDT) and its own status bit (SCDF).

The figure below shows an example where SCDT[7:0] is set to six. The break signal remains high as long as the short circuit condition is present.

No overrun event is generated when the interrupt event is pending while a new short-circuit condition is detected.

Figure 243. SCD timing example



The SCDx event generated by the SCDx block can be assigned to break output signals mdf_break[3:0]. The break signal assignment to a given short-circuit detector event is done

by BKSCD[3:0] in the *MDF short circuit detector control register x (MDF_SCDxCR)*. The break outputs are shared with the over-current function.

Note: SCDs cannot be used to monitor the ADC data interface (ADCITF).

SCD activation sequence example

1. Enable and configure CKGEN to generate the mdf_proc_ck.
2. Set SCDEN to 0.
3. Wait for SCDACTIVE = 0. If SCDEN was previously enabled, this phase can take two periods of mdf_hclk, and two periods of mdf_proc_ck.
4. Program BKSCD[3:0] and SCDT[7:0].
5. Set SCDEN to 1.

Note: BKSCD[3:0] and SCDT[7:0] must not be changed when SCDACTIVE = 1.

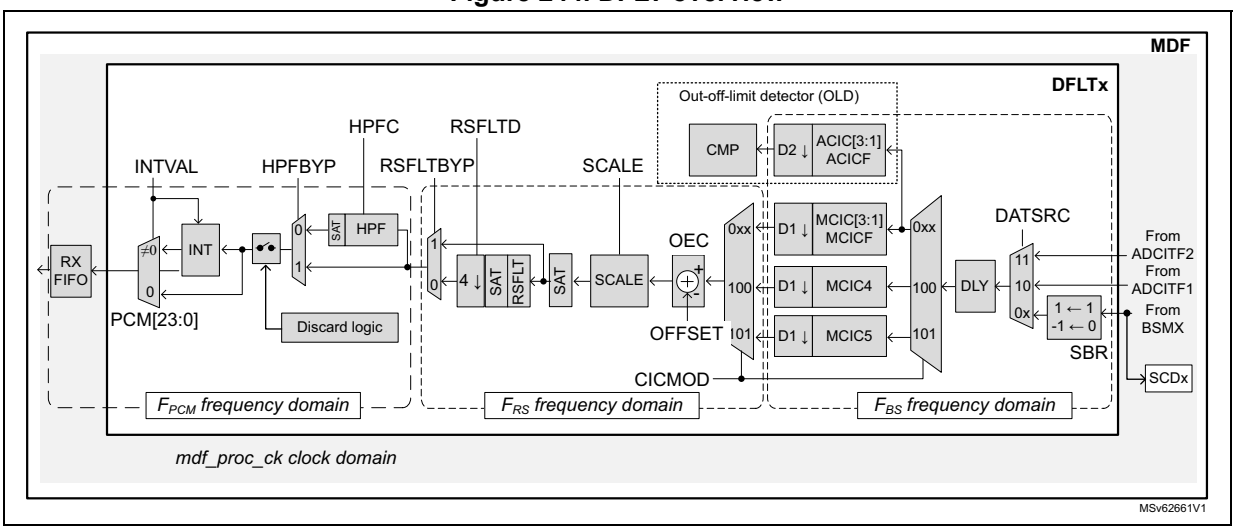
35.4.8 Digital filter processing (DFLT)

The digital filter processing includes the following sub-blocks:

- symbol remap (SBR)
- source selector
- clock skipper delay (DLY)
- CIC decimation filter that can be configured:
  - in single $Sinc^4$ or $Sinc^5$ order CIC (MCIC)
  - in two CIC filters:
    - a main filter (MCIC), high resolution
    - an auxiliary filter (ACIC), low-latency, for the out-off-limit detector (OLD)
    - both MCIC and ACIC can be configured as $Sinc^1$, $Sinc^2$, $Sinc^3$ or FastSinc
- gain control (SCALE)
- signal saturation (SAT)
- reshape filter (RSFLT)
- high-pass filter (HPF)
- integrator (INT)
- receive RXFIFO

The figure below shows the filter-path configuration according to CICMOD[2:0]. Several configuration bits are available to configure the digital filter to the application needs.

Figure 244. DFLT overview



Symbol remap and source selection

The symbol remap (SBR) converts the bitstream selected by the BSMX into data usable by the filter path. More especially:

- The high levels are converted into a 16-bit signed number + 1.
- The low levels are converted into a 16-bit signed number - 1.

The signal source of the digital filter can be selected via DATSRC[1:0] between the two following:

- data coming from the BSMX
- data coming from one of the ADC interfaces (ADCITF2 or 1)

Programmable micro-delay control (DLY)

The digital filter has a delay line that allows the timing adjustment of each stream with the resolution of the bitstream clock.

This feature is particularly helpful in the case of microphone beam forming applications where delays smaller than the final sampling rate, must be applied to the incoming stream.This feature is helpful when the MDF is synchronized with other ADF or MDF blocks for a beam forming application for example.

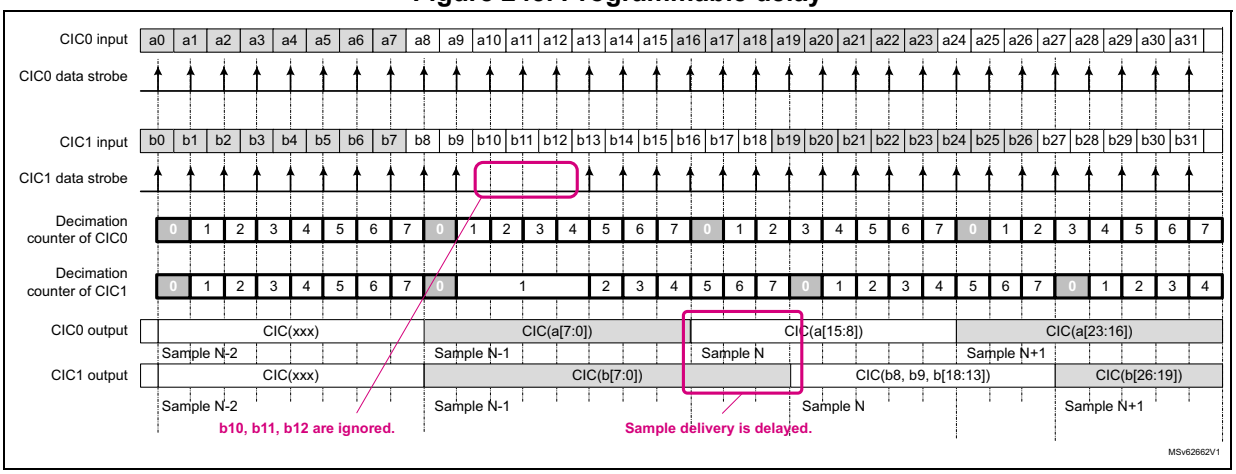
The delay is performed by discarding a given number of samples from the selected input stream, before samples enter into the CIC filter. This data discarding is performed by skipping a given number of data strobe, preventing the CIC filter to take into account those data.

When the wanted amount of data strobe has been skipped, the next incoming samples are strobed normally.

The figure below shows an example of how to apply dynamically small delay to an incoming stream. For simplification, the CIC filter performs a decimation by height in this example.

CIC1 represents the CIC included in the ADF and CIC0 represents a filter from another ADF or MDF instance.

Figure 245. Programmable delay



The CIC of filter 1 (CIC1) receives a command in order to skip three incoming samples. So the input samples named b10, b11 and b12 are not processed by CIC1. As a consequence, the output sample N+1 generated by CIC0 is build from input samples a[23:16] while the sample N+1 of CIC1 is build from input samples b[26:19].

Finally, the non-skipped data stream looks delayed by three bitstream periods.

Note: When the input data strobes are skipped, the decimation counter remains frozen. As a consequence, the samples delivered by the CIC1 are a bit delayed.

Warning: It is not recommended to apply a delay bigger than the programmed decimation ratio (CIC + RSFLT decimation), especially if the MDF is programmed in interleaved acquisition mode. There is a strong risk of data misalignment inside the FIFOs.

The following steps are needed to program the amount of bitstream clock periods to be skipped:

1. Wait for SKPBF equal to 0.
2. Write SKPDLY[6:0] to the wanted number of bitstream clock periods to be skipped. The SKPBF flag goes immediately to 1, indicating that the delay value entered into SKPDLY[6:0] is under process.
   - If the corresponding DFLTx is not yet enabled (DFLTEN = 0), then the DLY logic waits for DFLTEN = 1. When the application sets DFLTEN to 1, the DLY logic starts to skip the amount of wanted data strobes.
   - If the corresponding digital filter is already enabled (DFLTEN = 1), then the DLY logic immediately starts to skip the amount of wanted data strobes.

   When the MDF skipped the amount of wanted data strobes, then SKPBF goes back to 0.
3. If the application needs to skip more data strobes, then the operation must be restarted from step 1.

The effect of the delay performed with this mechanism is cumulative as long as the MDF is enabled. In a given filter, if the application performs a D1 delay, followed by a D2 delay, then all other active filters are delayed by D1 + D2.

Data coming from ADCs can also benefit of this feature.

In interleaved acquisition mode, it is up to the application to insure that the delay applied on the different microphones is in line with the depth of the RXFIFO buffers. If the relative delay between each activated filter is less than the decimation ratio, then it costs one FIFO location.

If the interleaved acquisition mode is not used, then the delay value has no impact on the RXFIFO buffering.

Note: If SKPDLY[6:0] is written when SKPBF = 1, the write operation is ignored.

Cascaded-integrator-comb (CIC) filter

The CIC digital filters are an efficient implementation of low-pass filters, often used for decimation and interpolation. The CIC frequency response is equal to a $Sinc^N$ function, this is why they are often called Sinc filters.

The $Sinc^N$ digital filter embedded into the MDF is configurable according to the application targeted.

- For audio applications, such as speech capture from digital microphones, the application can select a high-resolution low-pass decimation filter by setting CICMOD to 100 or 101.
- If the targeted application is motor control or any other sensor capture, then the application can configure the CIC in order to offer a main filter path for acquisition (MCIC) and an auxiliary filter for fast-event generation (ACIC). This auxiliary filter can be used for example, to detect over-current conditions. This mode is selected by setting CICMOD[2:0] = 0xx.

When CICMOD[2:0] = 0xx, the following CIC filters are available:

- main filter (MCIC)
- auxiliary CIC filter (ACIC)

Both of them are configurable in FastSinc, $Sinc^1$ to $Sinc^3$.

When CICMOD[2:0] = 100, the CIC is configured into a single Sinc^4 and when CICMOD[2:0] = 101, the CIC is configured into a single Sinc^5 filter.

The filters have the following transfer function (impulse response in z domain):

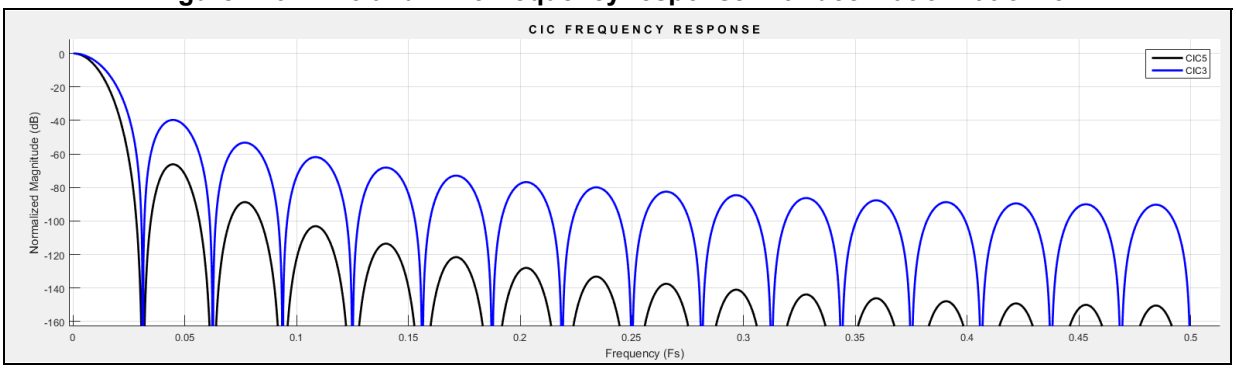
- Sinc^N filter type: $$H(z) = \left(\frac{1 - z^{-D}}{1 - z^{-1}}\right)^N$$

- FastSinc filter type: $$H(z) = \left(\frac{1 - z^{-D}}{1 - z^{-1}}\right)^2 \cdot (1 + z^{-(2 \cdot D)})$$

where N can be 1, 2, 3, 4 or 5, and D is the decimation ratio.

D is equal to MCICD+1 or ACICD+1.

Figure 246. CIC3 and CIC5 frequency response with decimation ratio = 32



CIC output data size

The size of samples delivered by the CIC (DS_{CIC}) depends on the following parameters:

- CIC order (N)
- CIC decimation ratio (D)
- data size of the input stream (DSIN)

The CIC order and decimation ratio must be programmed in order to insure that the data size does not exceed the 26-bit CIC capability.

The following formula gives the output data size (DS_{CIC}) according to the parameters above.

$$DS_{CIC} = \left(\frac{N \times Ln(D)}{Ln(2)}\right) + DS_{IN}$$

and the CIC gain is given by this formula:

$$G_{CIC} = (D)^N$$

The decimation ratio can be adjusted from 2 to 512 for the main CIC filter and from 1 to 32 for the auxiliary CIC filter.

The table below gives some data output size in bits for some decimation values, when the data source is a full-scale signal coming from the serial interface or from a 12-bit ADC.

Note: DS_{IN} = 1 bit for a serial bitstream but can be up to 16 bits when coming from the ADCITF.

Table 298. Data size according to CIC order and CIC decimation values

Decimation	Data size (bits) when DS_{IN} = 1 bit (data from SITFx)						Data size (bits) when DS_{IN} = 12 bits (data from ADCITF)					
	Sinc^1	Sinc^2	FastSinc	Sinc^3	Sinc^4	Sinc^5	Sinc^1	Sinc^2	FastSinc	Sinc^3	Sinc^4	Sinc^5
4	3	5	6	7	9	11	14	16	17	18	20	22
8	4	7	8	10	13	16	15	18	19	21	24	-
12	5	9	9	12	16	19	16	20	21	23	-	-
16	5	9	10	13	17	21	16	20	21	24	-	-
24	6	11	12	15	20	24	17	22	23	-	-	-
32	6	11	12	16	21	26	17	22	23	-	-	-
64	7	13	14	19	25	-	18	24	25	-	-	-
128	8	15	16	22	-	-	19	26	-	-	-	-
256	9	17	18	25	-	-	20	-	-	-	-	-
512	10	19	20	-	-	-	21	-	-	-	-	-

The LSB part of the data provided by the CIC is not necessarily significant: it depends on the sensor performances and the ability of the CIC to reject the out-off-band noise.

The sample size at CIC output can be adjusted thanks to the SCALE block.

The table below shows the maximum allowed decimation ratio for the CIC filter, depending on the input data size. Bigger decimation ratio causes a wrap-around of the signal at CIC output, for strong input signals.

Note: The MDF cannot detect or prevent a CIC wrap-around.

Table 299. Maximum decimation ratio versus order and input data size

Filter order	Max. decimation ratio when DS_{IN} = 1 bit (SITFx)	Max. decimation ratio when DS_{IN} = 12 bits (ADCITF)	Max. decimation ratio when DS_{IN} = 16 bits (ADCITF)
Sinc^1	512	512	512
Sinc^2	512	128	32
FastSinc	512	90	22
Sinc^3	322	25	10
Sinc^4	76	11	5
Sinc^5	32	6	4

Scaling (SCALE) and saturation (SAT)

The SCALE block allows the application to adjust the amplitude of the signal provided by the CIC, by steps of 3 dB (± 0.5 dB).

The signal amplitude can be decreased by up to 8 bits (- 48.2 dB) and can be increased by up to 12 bits (+ 72.2 dB).

The gain is adjusted by the SCALE[5:0] field in the *MDF digital filter configuration register x (MDF_DFLTxCICR)*.

SCALE[5:0] can be changed even if the corresponding DLFTx is enabled. During the gain transition, the signal provided by the filter is disturbed.

Due to internal resynchronization, there is a delay of some cycles of mdf_proc_ck clock between the moment where the application writes the new gain, and the moment where the gain is effectively applied to the samples. If the application attempts to write a new gain value while the previous one is not yet applied, this new gain value is ignored. Reading back SCALE[5:0] informs the application on the current gain value.

The table below shows the possible gain values.

Table 300. Possible gain values

SCALE[5:0]	Gain (dB)	SCALE[5:0]	Gain (dB)	SCALE[5:0]	Gain (dB)	SCALE[5:0]	Gain (dB)
0x20	- 48.2	0x2B	- 14.5	0x06	+ 18.1	0x11	+ 51.7
0x21	- 44.6	0x2C	- 12.0	0x07	+ 21.6	0x12	+ 54.2
0x22	- 42.1	0x2D	- 8.5	0x08	+ 24.1	0x13	+ 57.7
0x23	- 38.6	0x2E	- 6.0	0x09	+ 27.6	0x14	+ 60.2
0x24	- 36.1	0x2F	- 2.5	0x0A	+ 30.1	0x15	+ 63.7
0x25	- 32.6	0x00	0.0	0x0B	+ 33.6	0x16	+ 66.2
0x26	- 30.1	0x01	+ 3.5	0x0C	+ 36.1	0x17	+ 69.7
0x27	- 26.6	0x02	+ 6.0	0x0D	+ 39.6	0x18	+ 72.2
0x28	- 24.1	0x03	+ 9.5	0x0E	+ 42.1	-	-
0x29	- 20.6	0x04	+ 12.0	0x0F	+ 45.7	-	-
0x2A	- 18.1	0x05	+ 15.6	0x10	+ 48.2	-	-

The SAT blocks avoid having a wrap-around of the binary code when the code exceeds its maximal or minimal value.

The MDF performs saturation operations at the following levels:

- after the SCALE block (performed by the SAT block). The signal is saturated at 24 bits.
- inside the RSFLT, to insure a good filter behavior
- at the output of the HPF, to insure that the output signal does not exceed 24 bits

The SATF bit informs the application that a saturation occurred either after the SCALE, inside the RSFLT or after the HPF. In addition, an interrupt can be generated if SATIE is set to 1. As soon as a saturation is detected, the SATF flag is set to 1. It is up to the application to clear this flag in order to be able to detect a new saturation.

Those bits are in the *MDF DFLTx interrupt enable register x (MDF_DFLTxIER)* and *MDF DFLTx interrupt status register x (MDF_DFLTxISR)*.

Gain adjustment policy

To get the best MDF performances, it is important to properly adjust the gain value via SCALE[5:0].

A usual way to adjust the gain is to select the SCALE[5:0] value that gives a final signal amplitude as close as possible to the 24-bit full scale, for the maximum input signal.

A way to select the optimal gain is detailed below:

1. Check that, for the expected input signal, the data size into the CIC filter does not exceed 26 bits. This can be checked using this formula:

$$\frac{LN(SIN_{pp} \cdot D^N)}{LN(2)} < 26$$

where N represents the CIC order, D the decimation ratio and SIN_{pp} the maximum peak-to-peak amplitude of the input signal.

SIN_{pp}can take:

– a maximum peak-to-peak amplitude of 2 (± 1), for samples coming from SITF

– a maximum peak-to-peak amplitude of 4095 (+ 2047, - 2048) for samples coming from a 12-bit ADC

Example: a $Sinc^4$ can be used with a decimation ratio of 96, if the maximum input signal does not exceed ± 0.35. Indeed:

$$\frac{LN(0.7 \cdot 96^4)}{LN(2)} \sim 25.82 \text{ bits} < 26 \text{ bits}$$

2. Adjust the SCALE value.

To select the most appropriate SCALE value, the user must check if the RSFLT is used or not. If the RSFLT is used, the data size at SCALE output must not exceed 22 bits, otherwise the data size can be up to 24 bits.

The SCALE value in dB is selected using this formula:

$$SCALE_{dB} < 20 \cdot \log 10\left(\frac{2^{NB}}{SIN_{pp} \cdot D^N}\right)$$

where NB is equal to 22 if RSFLT is enabled, or 24 if RSFLT is bypassed. $SCALE_{dB}$ represents the gain value selected by SCALE[5:0].

Example: for a $Sinc^4$ with a decimation ratio of 96 and a SIN_{pp} of 0.7:

– If the RSFLT is bypassed:

$$20 \cdot \log 10\left(\frac{2^{24}}{0.7 \cdot 96^4}\right) \sim -11 \text{ dB}$$

$SCALE_{dB}$ value must be lower than - 11 dB. The closest lower value is - 12dB (SCALE[5:0] = 0x2C).

– If the RSFLT is enabled:

$$20 \cdot \log 10\left(\frac{2^{22}}{0.7 \cdot 96^4}\right) \sim -23 \text{ dB}$$

$SCALE_{dB}$ value must be lower than - 23 dB, the closest lower value is - 24.1 dB (SCALE[5:0] = 0x28).

If SCALE[5:0] is set to a higher value, then a saturation may occur. An event flag informs the user if a saturation occurred.

The table below proposes gain values for different filter configurations, when the data comes from the SITFx, according to the CIC order, and the CIC decimation ratio. This table is not exhaustive, and considers a full-scale input signal (see *Section 35.7.5: Total MDF gain* for details).

Table 301. Recommended maximum gain values versus CIC decimation ratios

CIC decimation ratio	Gain settings (dB) for configuration SITF + CICx + RSFLT (+ HPF)					Gain settings (dB) for configuration SITF + CICx (+ HPF)				
	CIC5	CIC4	CIC3	CIC2	CIC1	CIC5	CIC4	CIC3	CIC2	CIC1
8	33.6	51.7	69.7	72.2	72.2	45.7	63.7	72.2	72.2	72.2
12	18.1	39.6	60.2			30.1	51.7			
16	3.5	27.6	51.7			15.6	39.6	63.7		
20	- 6.0	21.6	48.2			6.0	33.6	60.2		
24	- 12.0	15.6	42.1	69.7		0	27.6	54.2		
28	- 20.6	9.5	36.1	66.2		- 8.5	21.6	48.2		
32	-26.6	3.5	33.6	63.7		- 14.5	15.6	45.7		
48	-	- 8.5	24.1	57.7		-	3.5	31.6	69.7	
64	-	- 20.6	15.6	51.7		-	-8.5	27.6	63.7	
128	-	-	- 2.5	39.6		-	-	9.5	51.7	
256	-	-	- 20.6	27.6		-	-	-8.5	39.6	

Reshaping filter (RSFLT)

In addition to the CIC, the MDF offers a reshaping IIR filter mainly dedicated to the audio application but also usable in other applications.

When the RSFLT is used, the sample size at its input must not exceed 22 bits.

The samples at the RSFLT output can be decimated by four or not according to the RSFLTD bit in the *MDF reshape filter configuration register x (MDF_DFLTxRSFR)*.

The RSFLT can be bypassed by setting RSFBYP to 1 in MDF_DFLTxRSFR.

The table below shows which sampling rate must be provided to the RSFLT in order to process the most common audio streams.

The RSFLT cutoff frequency (F_C) depends on the sample rates at its input (F_{RS}), and is given by the following formula:

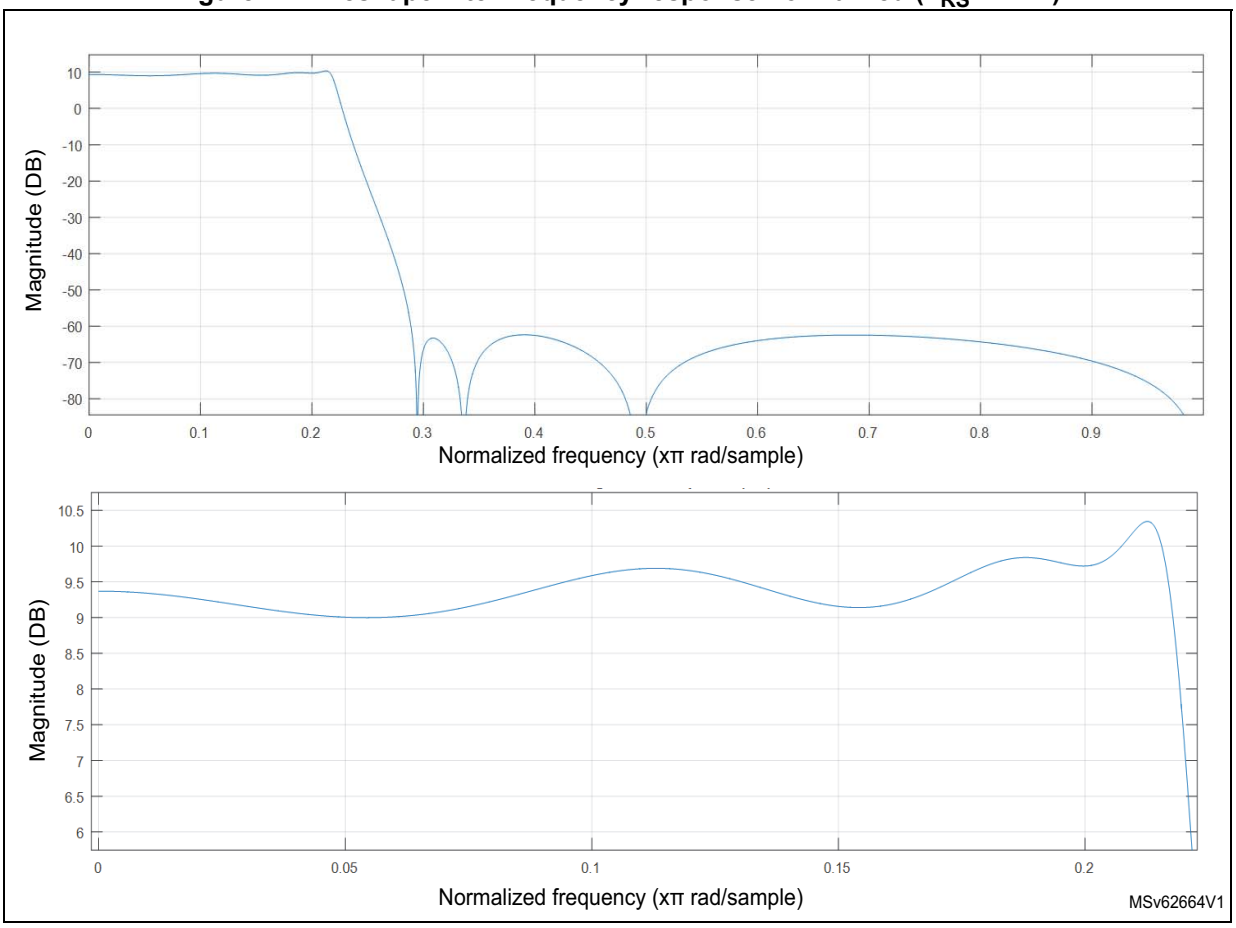
$$F_C = 0.111 \times F_{RS}$$

Table 302. Most common microphone settings

Sample rate (kHz) at RSFLT (F_{RS})	Pass band (kHz)	D2	PCM sampling rate (kHz)
32	3.55	4	8
64	7.1	4	16
128	14.2	4	32
192	21.3	4	48

The figure below shows the frequency response of the reshape filter.

Figure 247. Reshape filter frequency response normalized (F_{RS} / 2 = 1)



The RSFLT gain is about 9.3 dB, so the output data size is a little bit lower than 24 bits for a 22-bit wide input signal.

The RSFLT takes 24 clock cycles of mdf_proc_ck clock to process one sample at F_{RS}. When the RSFLT is enabled, the application must insure that the mdf_proc_ck is at least 24 times faster F_{RS}.

The RSFLT generates an event (rfovr_evt) and sets the RFOVRF flag, if the RSFLT receives a new sample while the previous one is still under processing.

When RFOVRF is set, the samples provided by the RSFLT are invalid. The application must then stop the data acquisition and provide x a faster mdf_proc_ck clock to the RSFLT.

High-pass filter (HPF)

The high-pass filter suppresses the low-frequency content from the final output data stream in case of continuous conversion mode. The high-pass filter can be enabled or disabled via HPFBYP in the *MDF reshape filter configuration register x (MDF_DFLTxRSFR)*.

The HPF is useful when there is parasitic low-frequency noise (or DC signal) in the input data source and it must be removed from the final data.

The HPF is a first order IIR filter and the cut-off frequency can be selected via HPFC[1:0] in the *MDF reshape filter configuration register x (MDF_DFLTxRSFR)*, among the following values:

- 0.000625 x F_{PCM}
- 0.00125 x F_{PCM}
- 0.00250 x F_{PCM}
- 0.00950 x F_{PCM}

Table 303. HPF 3 dB cut-off frequencies examples

HPFC	3 dB cut-off frequency for common F_{PCM} frequencies (Hz)		
	F_{PCM} = 8 kHz	F_{PCM} = 16 kHz	F_{PCM} = 48 kHz
0	5	10	30
1	10	20	60
2	20	40	120
3	76	152	456

The HPF output is saturated at 24 bits. The SATF flag is set if a sample is saturated.

Offset error compensation (OEC)

Each digital filter has its own OEC. The offset correction is performed by subtracting to the signal provided by the CIC, the OFFSET[25:0] in the *MDF offset error compensation control register x (MDF_OECxCR)*.

Due to the internal resynchronization, there is a delay of some cycles of mdf_proc_ck clock between the moment where the application writes the new offset, and the moment where the new offset value is effectively applied to the samples. If the application attempts to write a new offset value while the previous one is not yet applied, this new offset value is ignored. Reading back OFFSET[25:0] informs the application on the current offset value.

Integrator (INT)

The INT performs additional decimation and a resolution increase of data coming from the digital filter. The INT simply performs the sum of data from a digital filter for a given number of data samples from a filter.

The INT is enabled by setting INTVAL[6:0] to a value different from 0.

The amount of integrated values can be defined by INTVAL[6:0] in the range of 2 to 128.

In order to control the data width at the integrator output, the resulting data can be divided by 1, 4, 32 or 128. This feature is controlled by INTDIV[1:0].

35.4.9 Out-of-limit detector (OLD)

The OLD triggers an event when a signal reaches or crosses given maximum and minimum threshold values. The generated event can drive an interrupt or break signals (mdf_break[3:0]) when conditions are met.

The OLD can be used only if the CIC filter is configured in motor and sensing mode (CICMOD[2:0] = 0xx).

The OLD takes the input signal selected by the main filter, process it using a $Sinc^N$ or FastSinc filter (ACIC), and compares the resulting signal to programmed thresholds.

The OLD is enabled via the OLDEN bit in MDF_OLD0CR. Once enabled, the input data are continuously monitored. There is no need to have the DFLTx enabled for using the OLD function.

The MDF offers a high- and low-threshold register that are compared with given data values.

The application can generate an event if the signal is inside or outside the boundary defined by those two thresholds. This behavior is controlled via THINB in the MDF_OLD0CR.

If the application only wants to generate an event when the input signal is higher than OLDTHH, then THINB and OLDTHL must both be cleared to 0.

If the application only wants to generate an event when the input signal is lower than OLDTHH, then THINB must be set to 1 and OLDTHL must be cleared to 0.

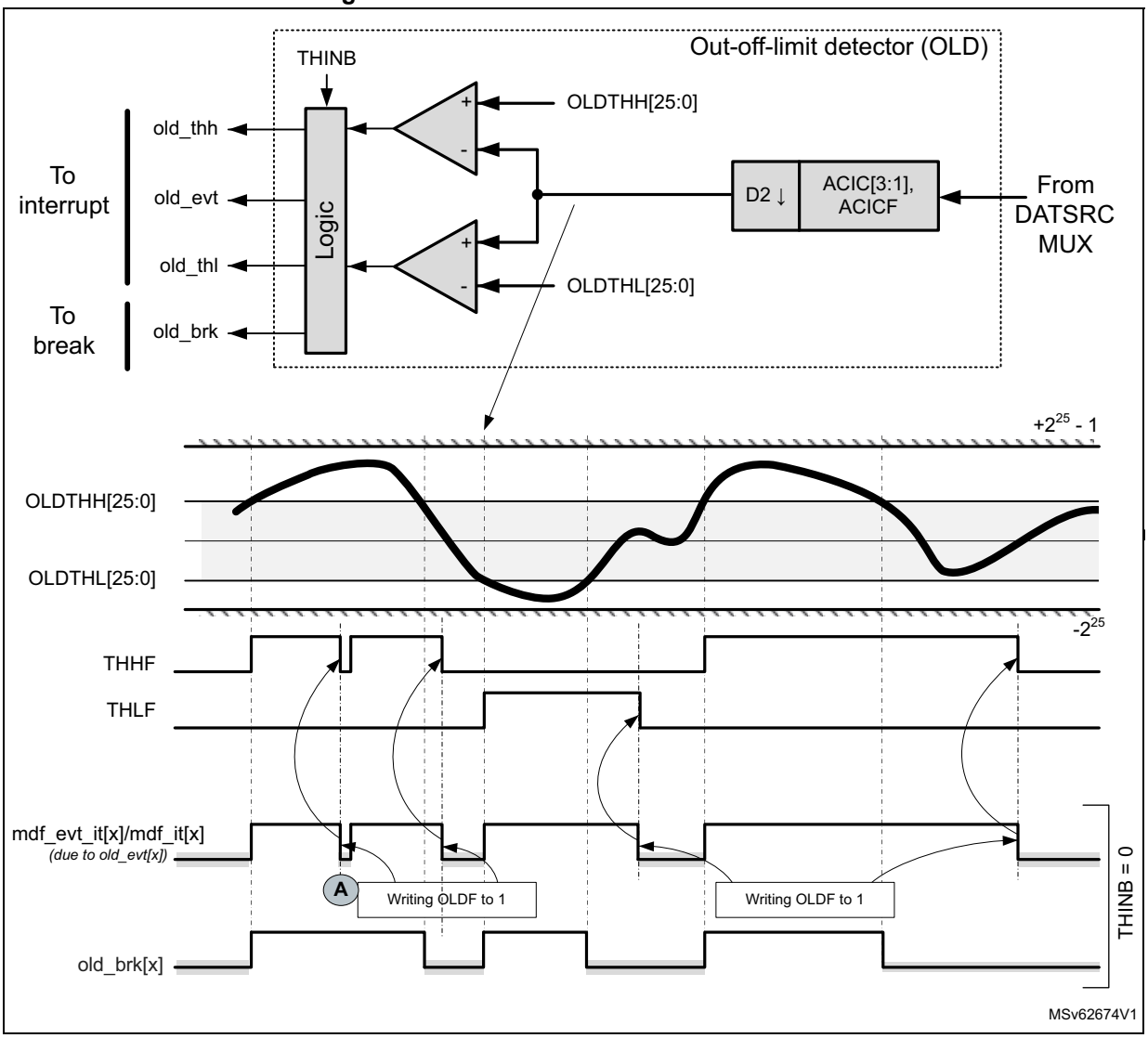
Note: It is not recommended to set a OLDTHL to a value bigger than OLDTHH.

The response time of the OLD depends on several parameters, the most important are listed hereafter:

- sampling rate frequency used by the external sensor
- ACIC decimation ratio
- ACIC order

The OLD can be used for over-current detection but also as current limiter if the PWM signal is generated by a timer receiving a old_brk signal. Generally, to get a fast response time for over-current detection, it is recommended to use the ACIC with lowest order as possible and the minimum decimation ratio. FastSinc is also a good choice for over-current detection. The application must perform a trade-off between the filter resolution and the response time.

Figure 248. Out-of-limit detector thresholds



Status flags are available in order to inform the application that an OLD event is detected. Latched events are cleared by writing 1 into the OLDF bit.

As shown in *Figure 248*, when THINB = 0, the interrupt signal remains active as long as the signal is outside the gray area. At position A, the application clears the interrupt, but the interrupt is re-asserted because the signal is still outside the gray area. When the application clears OLDF, THHF and THLF are cleared as well.

An OLD event can be assigned to one or several break output signals (mdf_break[3:0]). The break signal assignment to a given OLD event is done by BKOLD[3:0] in the *MDF out-of limit detector control register x (MDF_OLDxCR)*.

Note: The generation of break signals is independent from the interrupts generation.

OLD activation sequence example

1. Enable and configure CKGEN.
2. Set OLDEN to 0.
3. Wait for OLDACTIVE = 0. If OLDEN was previously enabled, this phase can take two periods of mdf_hclk and two periods of mdf_proc_ck.
4. Program BKOLD[3:0], ACICN[1:0], ACICD[4:0], THINB,OLDTHL[25:0] and OLDTHH[25:0].
5. Set OLDEN to 1.

35.4.10 Digital filter acquisition modes

The MDF offers the following modes to perform a data capture:

- Asynchronous continuous acquisition mode
- Asynchronous single-shot acquisition mode
- Synchronous continuous acquisition mode
- Synchronous single-shot acquisition mode
- Window continuous acquisition mode
- Synchronous snapshot acquisition mode

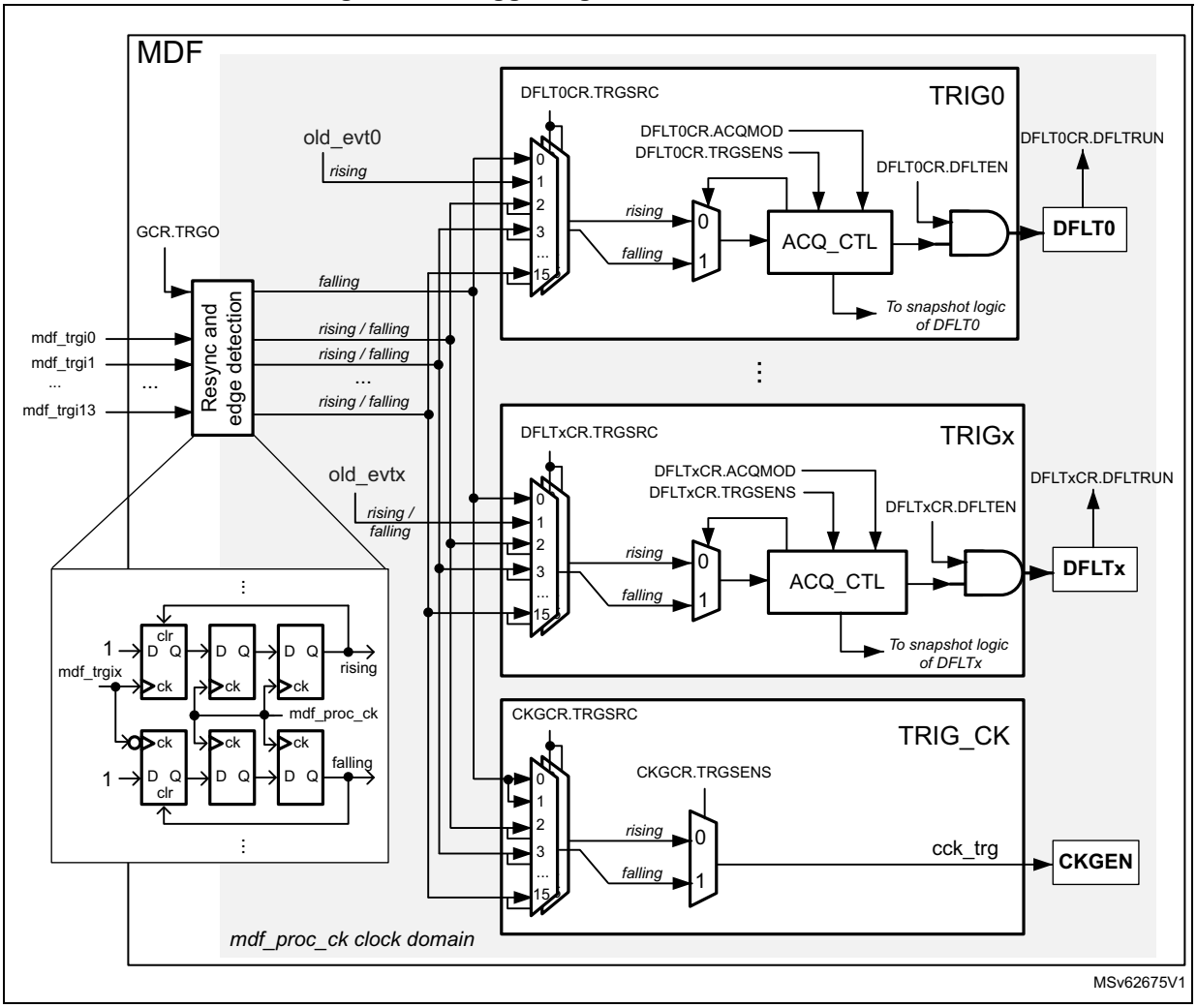
For each filter, one of these modes can be selected independently.

Note: To perform a data capture, the filters, the interfaces providing the data (SITFx or ADCITF) and the CKGEN must be enabled. If needed, MDF_CCK0 or MDF_CCK1 must be enabled as well.

The filter can be stopped immediately when DFLTEN is set to 0. This resets the filters and flushes the RXFIFO of the corresponding filter path. The DFLTACTIVE flag also goes back to 0 when the RXFIFO and the filters are reset.

The figure below shows a simplified view of the trigger logic available for each filter and for the clock generator.

Figure 249. Trigger logic for DFLT and CKGEN



A block common to all TRIG blocks performs the rising and falling edges detection and the resynchronization of the input triggers to the mdf_ker_ck clock domain. This implementation allows the application to use triggers with pulse width smaller than the mdf_ker_ck period.

In synchronous modes, the TRIG block offers the possibility to select one of the following trigger sources:

- mdf_trgi[13:0] signals (refer to *Table 293* for the triggers connections)
- TRGO bit in the *MDF global control register (MDF_GCR)*
- OLD event of each filter path

The edge sensitivity can also be selected, except for TRGO and OLD events.

Asynchronous continuous acquisition mode

This mode allows the application to start a continuous acquisition on one or several filters by simply writing their DFLTEN bits to 1.

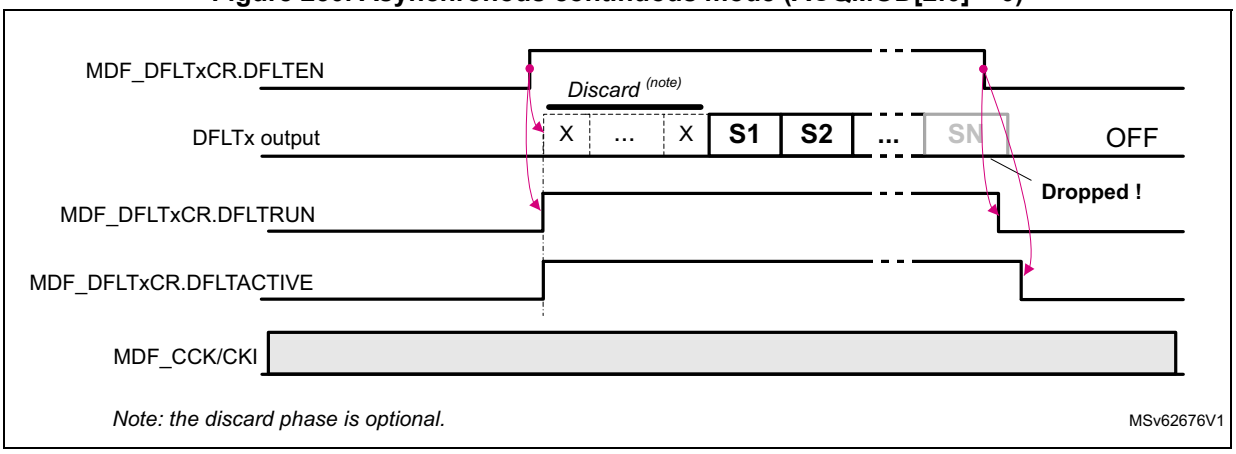
The Asynchronous continuous acquisition mode is selected when ACQMOD[2:0] = 0.

The sequence below shows the most important programming steps (assuming that DFLTEN bits of the filters are set to 0):

1. Configure and enable the clock generator (CKGEN) so that the mdf_proc_ck frequency is compatible with the targeted application (see examples in *Table 307*).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the MDF_CCK0 and MDF_CCK1 clocks.
3. Program the filter configuration and set the ACQMOD[2:0] to 0.
4. Set to 1 the SITFEN bit of the requested data interfaces.
5. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
6. When DFLTEN is set to 1 for the filters to enable, the acquisition sequence starts immediately.

The figure below shows a simplified example of the samples generated by the DFLTx.

Figure 250. Asynchronous continuous mode (ACQMOD[2:0] = 0)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Asynchronous single-shot acquisition mode

This mode allows the application to start the acquisition of one sample on one or several filters by simply writing their DFLTEN bits to 1.

The Asynchronous single-shot acquisition mode is selected when ACQMOD[2:0] = 001.

The sequence below shows the most important programming steps (assuming that DFLTEN bits of the filters are cleared to 0):

1. Configure and enable the clock generator (CKGEN), so that the mdf_proc_ck frequency is compatible with the targeted application (see examples in *Table 303*).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the MDF_CCK0 and MDF_CCK1 clocks.

3. Program the filter configuration and set the ACQMOD[2:0] to 001.
4. Set to 1 the SITFEN bit of the requested data interfaces.
5. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
6. When DFLTEN is set to 1 for the filters to enable, each selected filter provides one data to the RXFIFO and stops the acquisition.

To trigger a new acquisition, for each filter, the application must:

1. Check that the previous acquisition is completed, by waiting that DFLTRUN = 0.
2. Set again DFLTEN to 1.

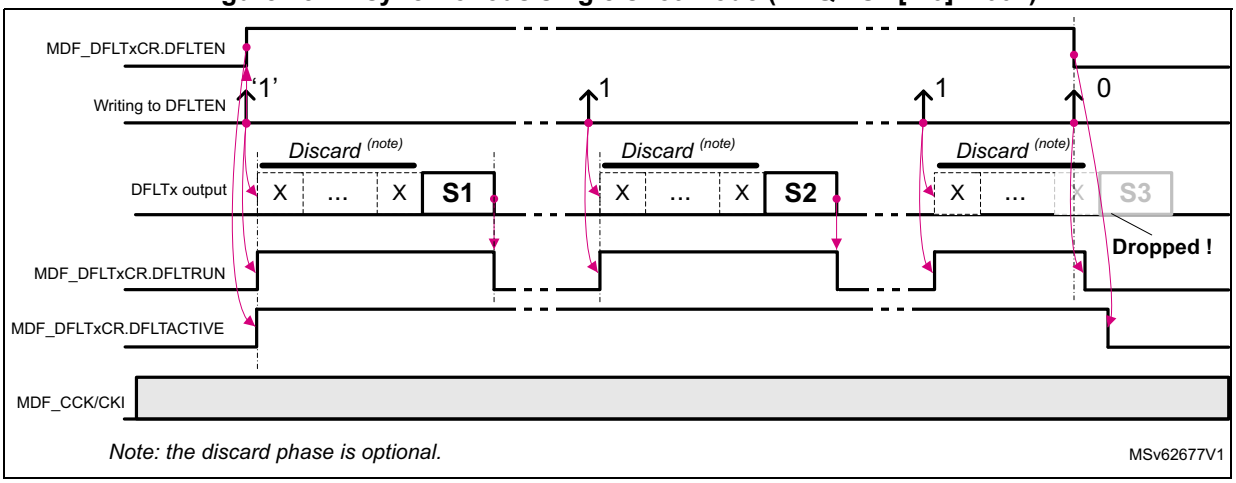
This sequence can be repeated every time a new data must be converted.

As shown in the *Figure 251*, every time DFLTEN is set to 1, an acquisition sequence is triggered. The first samples provided by the filter can be discarded if needed. At the end of each conversion, the decimation counters and filter taps are reset, and the filter is ready to start a new conversion.

If the DFLTEN is set to 0 while an acquisition is ongoing, the ongoing conversion is stopped (in the example, S3 is lost). This situation can be avoided with the following steps:

1. Wait for DFLTRUN = 0.
2. Read the sample from the RXFIFO.
3. Set DFLTEN to 0.

Figure 251. Asynchronous single-shot mode (ACQMOD[2:0] = 001)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Synchronous continuous acquisition mode

This mode allows the application to start a continuous acquisition on one or several filters by using the following trigger sources:

- one of the mdf_trg[13:0] signals
- OLD event of the corresponding filter
- TRGO bit

The Synchronous continuous acquisition mode is selected when ACQMOD[2:0] = 010.

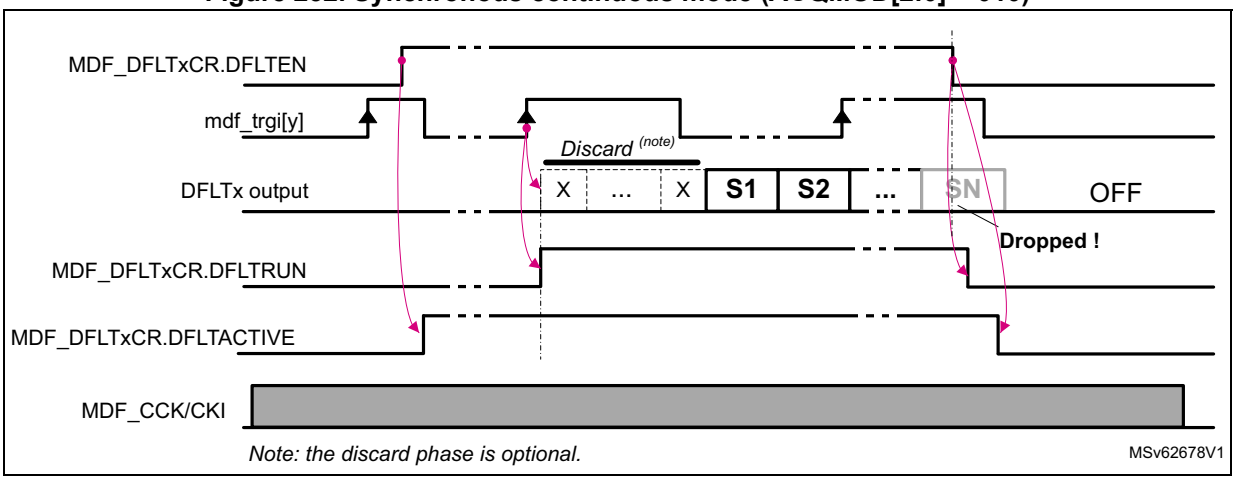
The sequence below shows the most important programming steps (assuming that DFLTEN bits of the filters are set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of mdf_proc_ck clock is compatible with the targeted application (see examples in *Table 303*).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the MDF_CCK0 and MDF_CCK1 clocks.
3. Program the filter configuration, and set the ACQMOD[2:0] to 010.
4. Set to 1 the SITFEN bit of the requested data interfaces.
5. Select the proper trigger source and sensitivity for each filter.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
7. Set DFLTEN to 1 for the filters to enable.
8. When the trigger condition is met, the filters start the acquisition.

The TRGSENS bit allows the selection of the trigger edge (rising or falling). The trigger is ignored if an acquisition is ongoing or if DFLTEN is set to 0.

The figure below shows a simplified example where the trigger logic is sensitive to a rising edge trigger (TRGSENS = 0). The first rising edge of the trigger signal is ignored because DFLTEN = 0. The next rising edge is taken into account and starts the acquisition. All other rising edges are ignored. The trigger logic is re-initialized when DFLTRUN goes back to 0.

Figure 252. Synchronous continuous mode (ACQMOD[2:0] = 010)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Synchronous single-shot acquisition mode

This mode allows the application to start a single acquisition on one or several filters by using the following trigger sources:

- one of the mdf_trg[13:0] signals
- OLD event of the corresponding filter
- TRGO bit

The Synchronous single-shot acquisition mode is selected when ACQMOD[2:0] = 011.

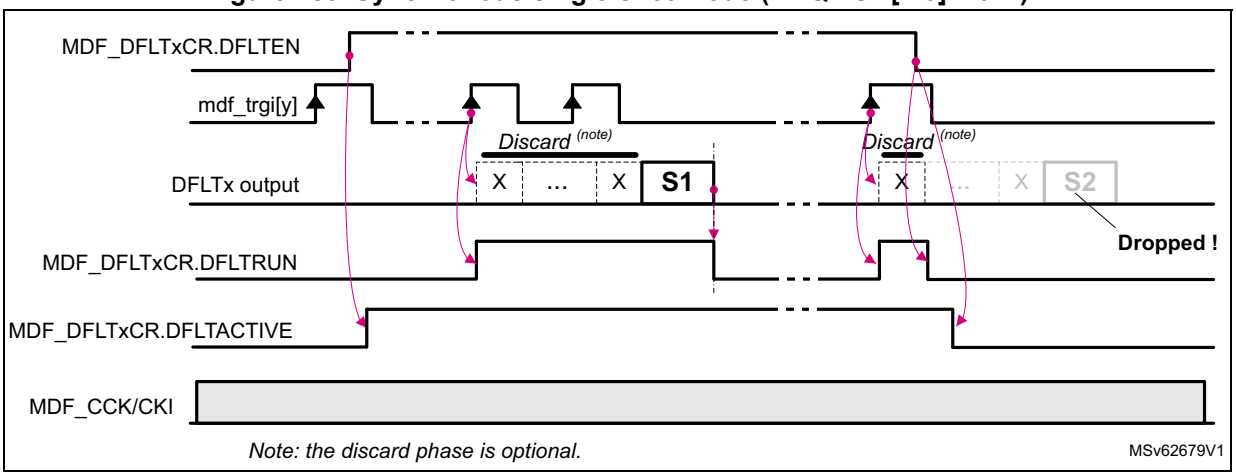
The sequence below shows the most important programming steps (assuming that DFLTEN bits of the filters are set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of mdf_proc_ck clock is compatible with the targeted application (see examples in *Table 303*).
2. Enable the CKGEN and, if needed, enable the MDF_CCK0 and MDF_CCK1 clocks.
3. Program the wanted filter configuration and set the ACQMOD[2:0] to 011.
4. Set to 1 the SITFEN bit of the requested data interfaces.
5. Select the proper trigger source and sensitivity for each filter.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence has been properly terminated.
7. Set DFLTEN to 1 for the filters to enable.
8. Then when the trigger condition is met, the filters start the acquisition and provide one data to the RXFIFO, then the filters are ready to accept a new trigger.

TRGSENS allows the selection of the trigger edge (rising or falling). The trigger is ignored if an acquisition is ongoing or if DFLTEN is set to 0.

The figure below shows a simplified example where the trigger logic is sensitive to a rising edge trigger (TRGSENS = 0). Every time a trigger rising edge is detected with DFLTEN = 1, an acquisition sequence is triggered. The first samples provided by the filter can be discarded if needed. At the end of each conversion, the decimation counters and filter taps are reset. DFLTRUN is set to 0 and the filter is ready to start a new conversion.

Figure 253. Synchronous single-shot mode (ACQMOD[2:0] = 011)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA

transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Figure 253 shows a case where DFLTEN is cleared to 0 while an acquisition is ongoing: thus the sample S2 is lost. This situation can be avoided with the following steps:

1. Wait for DFLTRUN = 0.
2. Read the sample from the RXFIFO.
3. Clear DFLTEN to 0.

Note: The ongoing DMA transfer is properly terminated.

Window continuous acquisition mode

This mode allows the application to start or stop a continuous acquisition on one or several filters controlled by consecutive edges of one of the following trigger sources:

- one of the mdf_trg[13:0] signals
- TRGO bit

The Window continuous acquisition mode is selected when ACQMOD[2:0] = 100.

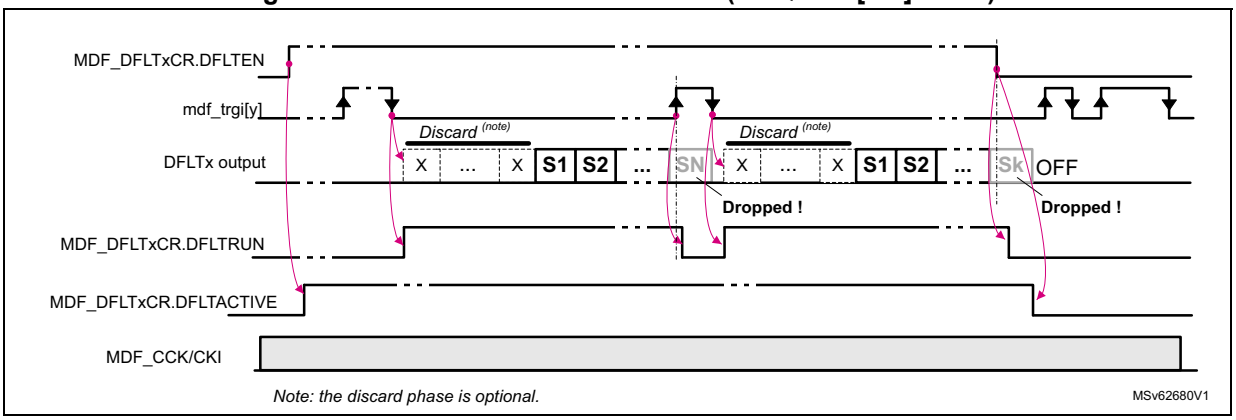
The sequence below shows the most important programming steps (assuming that DFLTEN bits of the filters are set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of mdf_proc_ck clock is compatible with the targeted application (see examples in *Table 303*).
2. Enable the CKGEN and, if needed, enable the MDF_CCK0 and MDF_CCK1 clocks.
3. Program the wanted filter settings and set the ACQMOD[2:0] to 100.
4. Set to 1 the SITFEN bit of the requested data interfaces.
5. Select the proper trigger source and sensitivity for each filter.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence has been properly terminated.
7. Set DFLTEN to 1 for the filters to enable.
8. If TRGSENS = 0, the acquisition starts on trigger rising edge and stops on trigger falling edge. If TRGSENS = 1, the acquisition starts on trigger falling edge and stops on trigger rising edge.

Note: The acquisition may restart if the trigger condition becomes again active.

Figure 254 shows a simplified example of window continuous acquisition mode, with TRGSENS = 1. Once DFLTEN is set to 1, the MDF waits for a falling edge on the selected trigger input. When the trigger condition is met, DFLTRUN goes to 1 and the acquisition starts. The acquisition stops if the MDF detects a rising edge on the selected trigger input. If DFLTEN is still set to 1, the MDF waits again for a falling edge on the selected trigger input.

Figure 254. Window continuous mode (ACQMOD[2:0] = 100)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Synchronous snapshot acquisition mode

In the acquisition modes described on previous sections, the application must wait for filters settlings time or the end of conversion before getting a new valid sample. For applications very critical in latency, the last valid sample can be get immediately, with an information on the age of this sample by using the snapshot acquisition mode.

In snapshot mode, the DFLTx is continuously acquiring the samples, but the processed samples are not stored into the RXFIFO. When a trigger occurs, the last valid sample, the current decimation value of MCIC and the sample counter of the INT are stored in MDF_SNPS0DR.

The possible trigger sources are the following:

- one of the mdf_trg[13:0] signals
- OLD event of the corresponding filter
- TRGO bit

The Synchronous snapshot acquisition mode is selected when ACQMOD[2:0] = 101.

The sequence below shows the most important programming steps (assuming that DFLTEN bits of the filters are set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of mdf_proc_ck clock is compatible with the targeted application (see examples in *Table 303*).
2. Enable the CKGEN and, if needed, enable the MDF_CCK0 and MDF_CCK1 clocks.
3. Program the wanted filter settings and set the ACQMOD[2:0] to 101. FTH is usually cleared to 0 to receive an interrupt as soon as the RXFIFO is not empty.
4. Set the SITFEN bit of the requested data interfaces to 1.
5. Select the proper trigger source and sensitivity for each filter.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence has been properly terminated,
7. Set DFLTEN to 1 for the filters to enable.

When the trigger condition is met,

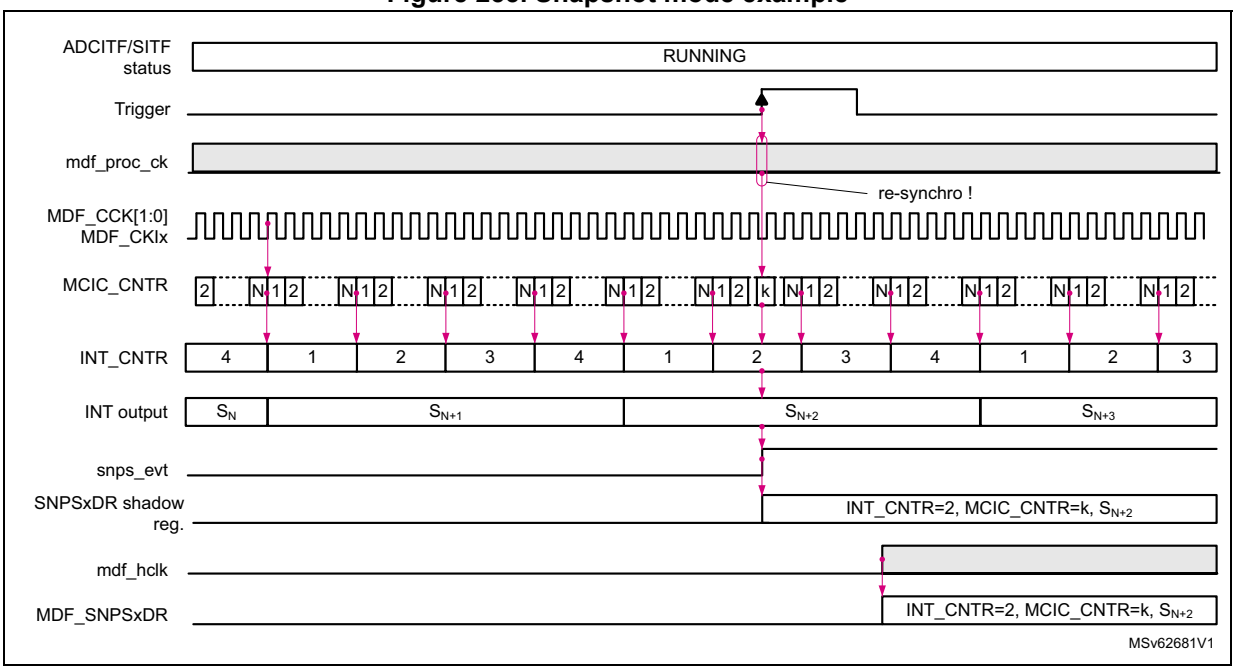
1. The MDF stores the last valid data provided by the corresponding filter, the value of the MCIC decimation counter, and the counter value of INT if selected by SNPSFMT into a shadow register.
2. The snps_evt is activated, indicating that a data is ready. If SSDRIE is set to 1, the MDF requests the AHB clock.
3. When the AHB clock is available the shadow register is stored in *MDF snapshot data register x (MDF_SNPSxDR)* and an interrupt is generated.
4. In the interrupt sub-routine, the application must do the following:
   a) Read the data located in MDF_SNPSxDR.
   b) Clear the SSDRF flag by writing it to 1.

As shown in the *Figure 255*, in snapshot mode, when the trigger event occurs, it is resynchronized with the processing clock (mdf_proc_ck), the decimation counter of the MCIC (MCIC_CNTR), the integrator counter (INT_CNTR) and the last valid sample (S_{N+2}), are stored into a shadow register, and the snps_evt event is generated. When the AHB clock is available, an interrupt is generated and the shadow register is copied in *MDF snapshot data register x (MDF_SNPSxDR)*. The application can read immediately the sample and the timing information.

The SSOVRF overrun flag is also available, allowing the application to check if an overrun condition occurs. A snapshot overrun condition is detected when the SSDRF flag is set to 1 while a new trigger event occurs.

In that case, the new trigger event is not taken into account. The application must clear the SSOVRF flag.

Figure 255. Snapshot mode example



Note: When an activated filter did not receive a data strobe, the filter is frozen. If the interface providing the stream is not enabled (SITF or ADCITF), the filter is frozen. Once a DFLT x is

activated, CIC, HPF, IIR and INT are reinitialized. For each filter a status bit indicates if the filter is currently running or not.

Starting several filters synchronously

To start the acquisition of several filters synchronously, the following sequence must be performed (assuming that DFLTEN is set to 0):

1. Enable the CKGEN and, if needed, enable the MDF_CCK0 and MDF_CCK1 clocks.
2. Set the SITFEN bit of the requested data interfaces to 1.
3. For each filter, set the acquisition mode to synchronous (ACQMOD[2:0] = 01x).
4. For each filter, set TRGSRC[3:0] to 0 (TRGO is selected).
5. For each filter, set TRGSENS to 0 (rising edge).
6. For each filter, set DFLTEN to 1.
7. Read TRGO bit until it is read to 0.
8. Set TRGO to 1. Then the acquisition sequence for all selected filters starts immediately.

To trigger a new acquisition (in case of single-shot), the application must do the following:

1. Check that the previous acquisition is completed, by waiting DFLTRUN = 0.
2. Read TRGO until it is read to 0.
3. Set again TRGO to 1.

Discarded samples

The MDF offers the possibility to program the amount of samples to be discarded after each restart:

- to avoid capturing samples affected by the impulse response of the filter
- to delay the acquisition of filters by a specific amount of samples

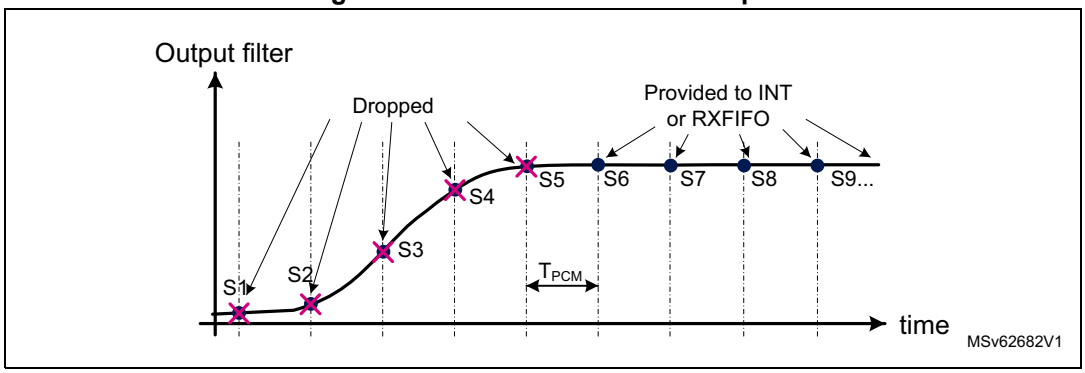
The discard function is controlled via NBDIS[7:0] as follows:

- When NBDIS[7:0] = 0, the discard function is disabled.
- When NBDIS[7:0] ≠ 0, the discard function is activated in one of the following condition:
  - when the DFLTEN bit goes to 1
  - every time an acquisition is started in (A)synchronous single-shot modes

Refer to *Figure 250* to *Figure 254*, *Figure 257* and *Figure 258*.

In the example shown in the figure below, the discard function is used to drop the first five samples provided by the digital filter (S1 to S5). The first sample transfered to the RXFIFO (or INT block if enabled) is S6.

Figure 256. Discard function example



Warning: All filters working in interleaved DMA mode must have the same NBDIS[7:0] value.

Data interface activation

The data interfaces are enabled by setting the corresponding SITFEN or ADCITFEN bit to 1. Once the digital filter is enabled, it receives the serial data from the external ΣΔ modulator or parallel internal data sources (ADC).

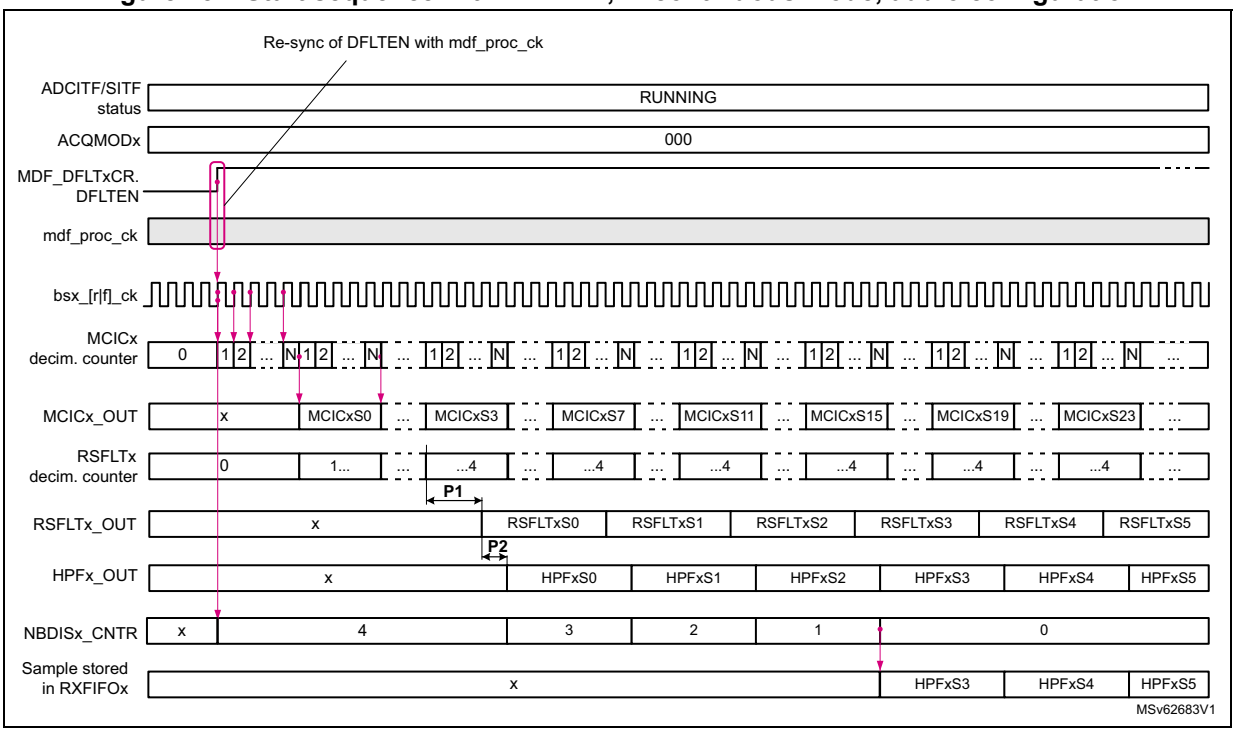
35.4.11 Start-up sequence examples

The figure below details an acquisition sequence start of a digital filter triggered by DFLTEN (ACQMOD[2:0] = 0), with NBDIS[7:0] = 3 (three samples to discard before acquisition).

The DFLTx is configured for audio application: MCIC, RSFLT and HPF activated.The data interface (SITFx or ADCITF) is assumed to be already activated.

Note: NBDIS[7:0] is set on purpose to a low value to simplify the drawing.

Figure 257. Start sequence with DFLTEN, in continuous mode, audio configuration



The DFLTEN bit is re-sampled into the MDF processing clock domain. When DFLTEN is detected high, the filter chain is enabled and the decimation counter of the MCIC filter is incremented at the rate of the bitstream clock.

When the MCIC decimation counter reaches its programmed value N, a sample is available for the RSFLT.

The RSFLT processes all the samples provided by the MCIC and delivers a sample to the HPF every time it processes four samples (decimation by 4) The RSFLT needs up to 24 cycles of mdf_proc_ck clock before delivering a sample (P1).

The HPF processes all the samples provided by the RSFLT, but the NBDIS function prevents the data writing in the RXFIFO as long as NBDIS_CNTR does not reach 0. This counter is decremented every time the HPF delivers a sample.

When NBDIS_CNTR reaches 0, the samples provided by the HPF are stored into the RXFIFO.

The example shown in *Figure 258* is based in a motor-control filter configuration:

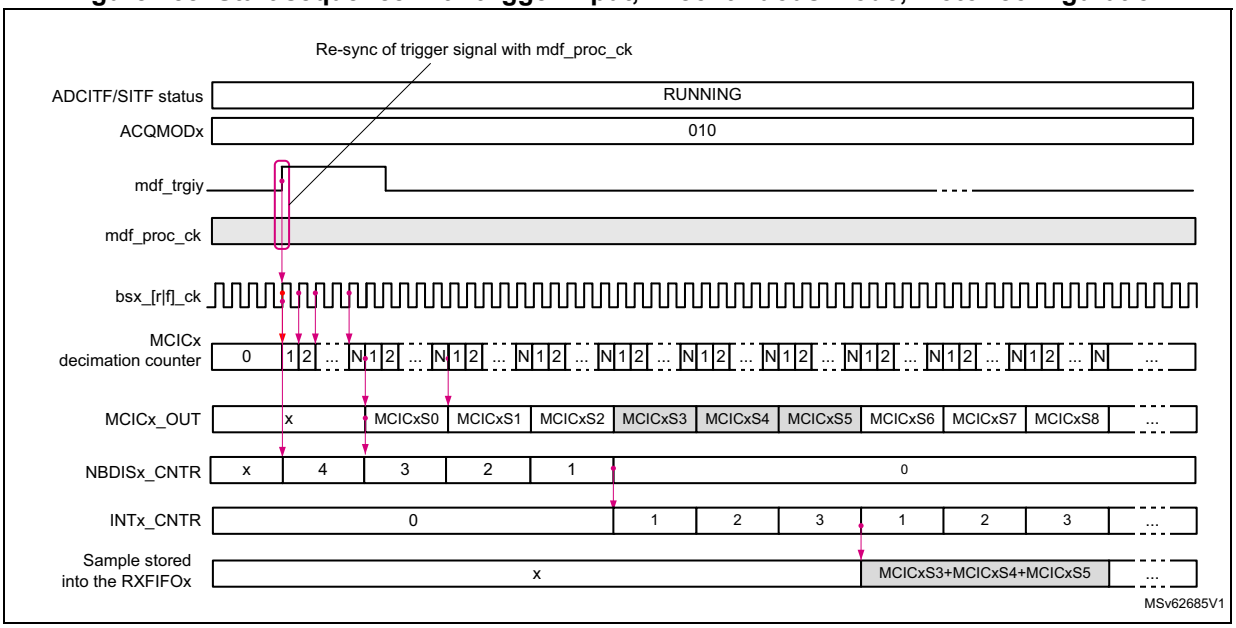
- MCIC and INT are enabled.
- SFLT and HPF are disabled.
- ACQMOD[2:0] is set to 1 and uses a mdf_trgiy input as trigger.
- NBDIS[7:0] is set to 3.
- INTVAL[6:0] is set to 2.

The mdf_trgiy input signal is re-sampled into the MDF processing clock domain to avoid any metastability issues. When the trigger condition is met (rising edge), the decimation counter of the MCIC filter is incremented at the rate of the bitstream clock.

When the MCIC decimation counter reaches its programmed value N, a sample is available. The NBDIS function prevents the samples writing in the INT as long as NBDIS_CNTR does not reach 0. This counter is decremented every time the MCIC delivers a sample.

When NBDIS_CNTR reaches 0, the new samples are provided to the INT, that performs the integration of three consecutive samples and stores them into the RXFIFO.

Figure 258. Start sequence with trigger input, in continuous mode, motor configuration

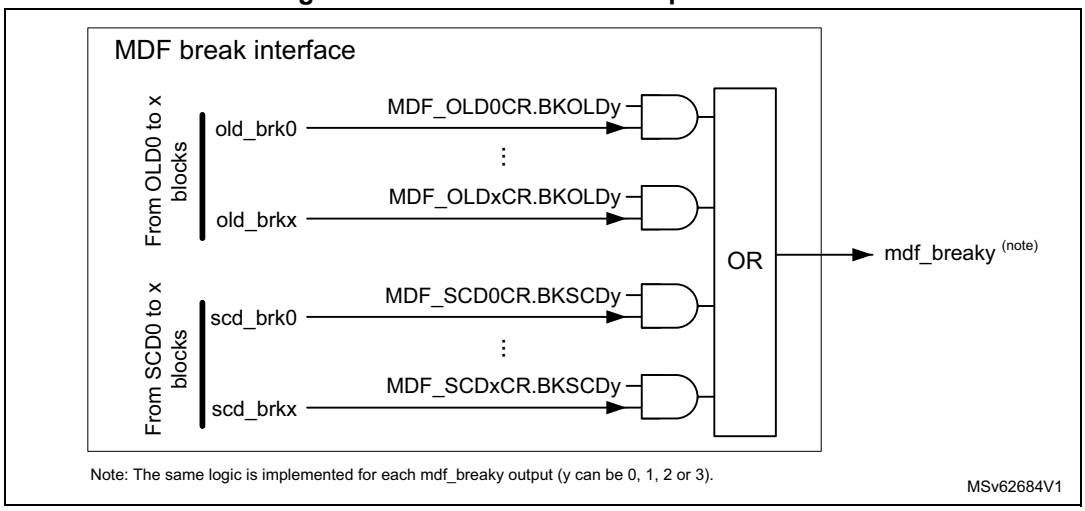


35.4.12 Break interface

The break interface merges the break events coming from the OLDx and SDCx blocks into four break signals connected to various peripherals of the product (see *Table 293* for details).

As shown in the figure below, several blocks can share the same break line. A same block can also drive several break lines. The break interface is controlled via *MDF out-of limit detector control register x (MDF_OLDxCR)* and *MDF short circuit detector control register x (MDF_SCDxCR)*.

Figure 259. Break interface simplified view



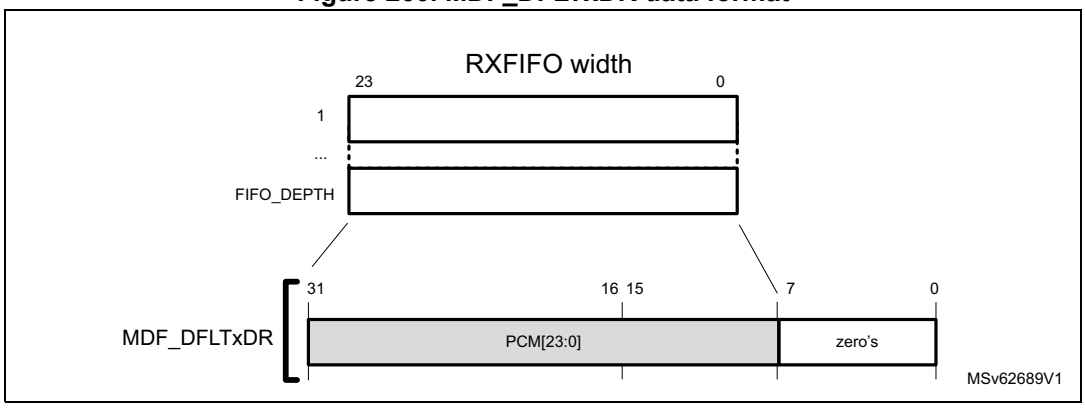
35.4.13 Data transfer to memory

Data format

The samples processed by DFLTx are stored into a RXFIFO. The application can read the samples stored into the FIFOs via the *MDF digital filter data register x (MDF_DFLTxDR)*. The samples inside this register are signed and left aligned. The bit 31 always represents the sign.

The MDF provides 24-bit left-aligned data. Performing a 16-bit access to MDF_DFLTxDR allows the application to get the 16 most significant bits. Performing a 32-bit access to MDF_DFLTxDR allows the application to get a 24-bit data size.

Figure 260. MDF_DFLTxDR data format



Data re-synchronization

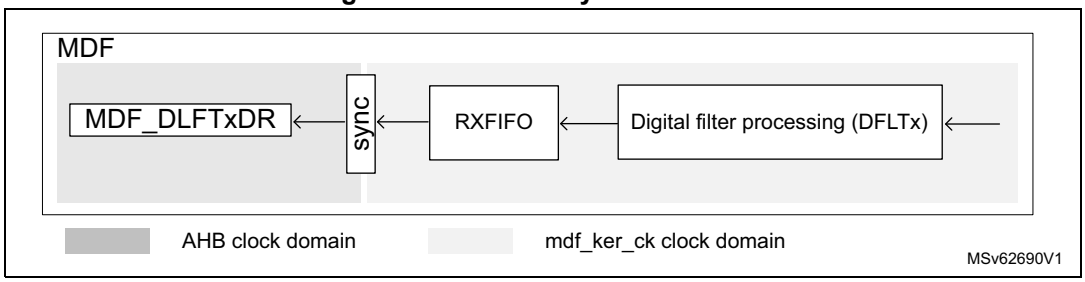
The samples stored into the RXFIFOs can be transferred into the memory by using either DMA requests or interrupt signaling.

Note: The RXFIFOs are located into the mdf_ker_ck clock domain, while MDF_DFLTxDR *are located into the mdf_hclk (AHB) clock domain.*

When the AHB clock is available, if MDF_DFLTxDR is empty and if a sample is available into the RXFIFO, this sample is transferred into MDF_DFLTxDR.

The sample transfer from the RXFIFO to MDF_DFLTxDR takes two periods of the AHB clock (mdf_hclk) and two periods of the mdf_ker_ck clock. The MDF inserts automatically wait-states if the application performs a read operation of MDF_DFLTxDR while the transfer of the new sample from the RXFIFO to MDF_DFLTxDR is not yet completed.

Figure 261. Data re-synchronization



The MDF can also combine two transfer types: the independent-transfer and the interleaved-transfer modes.

Independent-transfer mode

In this mode, each RXFIFO has its own DMA channel and its own FTHF flag event.

Both single and burst DMA transfers are supported in this mode, but the application must care about the following points:

- The RXFIFO must contain at least the same amount of samples than the burst size.
- The burst mode efficiency may be reduced due to the data re-synchronization explained in the previous section.

In addition, the application can select the RXFIFO threshold (FTH bit) to trigger the data transfer: a data transfer can be triggered as soon as the RXFIFO is not empty, or when the RXFIFO is half-full (containing depth / 2 samples).

For the DMA transfer, as soon as one of the RXFIFO reaches the threshold level, the corresponding DMA request is asserted in order to ask for data transfer. Successive DMA requests are performed as long as the corresponding RXFIFO is not empty.

The DMA mode of the RXFIFOx is enabled via the corresponding DMAEN bit in the *MDF digital filter control register x (MDF_DFLTxCR)*.

For the interrupt signaling, the following cases must be considered:

- If FTH = 0, as soon as a data is available into the *MDF digital filter data register x (MDF_DFLTxDR)*, the corresponding FTHF is set, allowing the generation of an interrupt. FTHF is released as soon as MDF_DFLTxDR is read.
- If FTH = 1, as soon as one of the RXFIFO reaches the threshold level and a data is available into MDF_DFLTxDR, the corresponding FTHF is set, allowing the generation of an interrupt. The FTHF flag is released as soon as one data is read. FTHF is set again if the threshold condition is met again. In this mode, every time an interrupt occurs, the application is supposed to read FIFO_SIZE / 2 data.

The independent-transfer mode must be used when the sample rates provided by each filter paths are not perfectly synchronous, or when the streams are independent. This situation may occur for example:

- when SITF is in SLAVE mode and each external sensor provides its own sampling clock
- when the decimation ratios of the DFLTx are not the same
- when all the streams are not starting at the same time

Interleaved-transfer mode

This mode optimizes the DMA request resources by sharing a single DMA request with several RXFIFOs.

Only single DMA transfers are supported in this mode.

In interleaved-transfer mode, FTH cannot be used and only the FTHF of RXFIFO0 is available for the interrupt signaling.

For the DMA transfer, when all the RXFIFOs working in interleaved-transfer mode are not empty, some DMA requests are generated in order to read sequentially one sample of each RXFIFOs, via MDF_DFLT0DR.

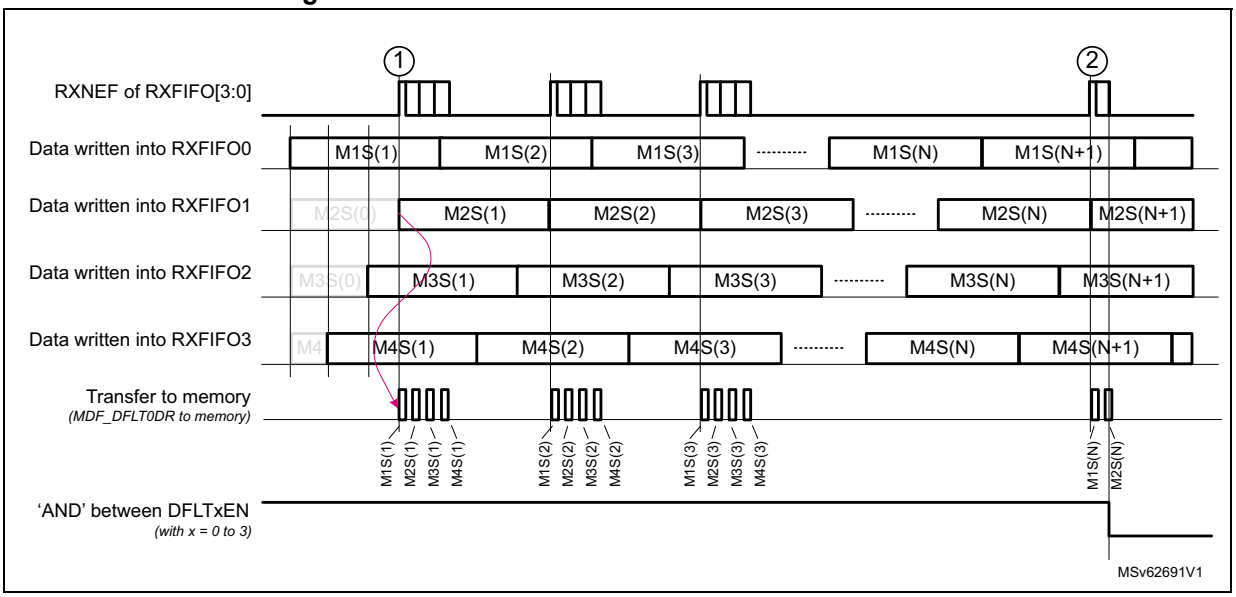
For the RXFIFOs working in interleaved-transfer mode, the DMA channel is enabled by setting to 1 DMAEN in MDF_DFLT0DR and by defining the amount of RXFIFOs working in interleaved mode via ILVNB[3:0]. DMAEN for the other RXFIFOs working in interleaved-transfer mode (RXFIFO 1 to ILVNB[3:0]) are not taken into account.

Figure 262 shows four RXFIFOs working in interleaved-transfer mode. Each RXFIFO has a delay due to the programming of the DLY block.

When all the RXFIFOs have at least one sample available (in this example the last FIFO receiving a sample is RXFIFO1), the MDF requests the data transfer to the memory (see 1 in *Figure 262*). In this figure, the DMA is assumed to be used for the data transfer to memory.

The acquisition can be simply stopped by setting DFTLEN to 0 for one of the filter configured in interleaved-transfer mode. In the example, when a DFLTEN is set to 0 (see 2 in *Figure 262*), the transfer to memory is immediately stopped. Samples M2S(N), M3S(N) and M4S(N) may be lost.

Figure 262. Data transfer in interleaved-transfer mode



In interrupt mode, when all the RXFIFOs working in interleaved-transfer mode have a data ready, the FTHF flag of RXFIFO0 is asserted to allow the interrupt generation. The application is then supposed to read sequentially one sample of each RXFIFOs, via MDF_DFLT0DR. The FTHF event is released when the first data is read.

For the RXFIFOs working in interleaved-transfer mode, the interrupt mode is enabled by setting FTHIE to 1 in MDF_DFLT0DR and by defining the amount of RXFIFOs working in interleaved-transfer mode via ILVNB[3:0]. FTHIE of the other RXFIFOs working in interleaved-transfer mode (RXFIFO1 to ILVNB[3:0]) are not taken into account.

The interleaved-transfer mode can only be applied starting at RXFIFO0 up to RXFIFOk. The number of RXFIFOs working in interleaved-transfer mode are defined by ILVNB[3:0].

When ILVNB[3:0] = 0, all RXFIFOs work in independent-transfer mode.

The interleaved-transfer mode can be used only for digital filters delivering samples perfectly synchronous each other. This is typically the case of sound capture with an array of digital microphones sharing the same bitstream clock.

Caution: To make the MDF working properly in interleaved-transfer mode, the following rules must be respected:

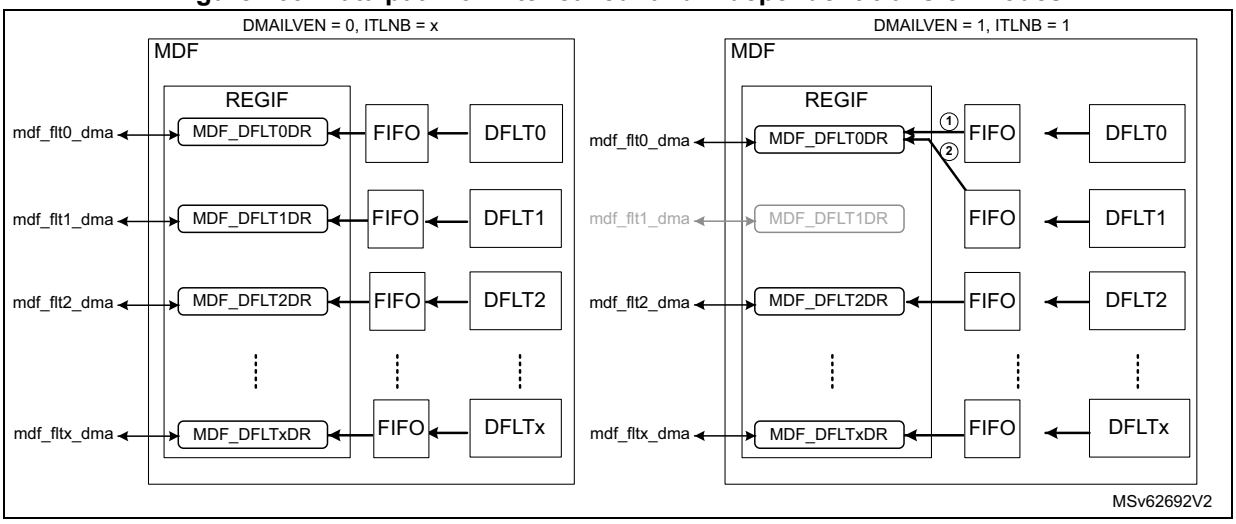
- All filters working in interleaved-transfer mode must be all synchronized together, meaning that all filters must select the same ACQMOD[2:0] (must be equal to 010 or 011), the same trigger source and the same trigger sensitivity.
- All filters working in interleaved-transfer mode must have the same configuration.
- All sensors providing the data for the filters working in interleaved-transfer mode must use the same bitstream clock frequency.
- The maximum delay difference applied with SPKDLY[6:0] for the filters working in interleaved-transfer mode must remain strictly lower than the RXFIFO depth. Normally this delay does not need to exceed one T_{PCM} period.

Note: Both independent- and interleaved-transfer modes can work in parallel: for example, the first RXFIFOs (RXFIFO0 to RXFIFOk) may work in interleaved-transfer mode when the others (RXFIFO(k+1) to RXFIFO(N)) works in independent-transfer mode.

If the AHB clock is not present when a data transfer must be performed, the MDF first requests the AHB clock (refer to Section 35.4.5: Clock generator (CKGEN) for details).

The figure below shows the data path for a configuration in independent-transfer mode (left-hand figure) and a configuration mixing both independent- and interleaved-transfer modes (right-hand figure).

Figure 263. Data path for interleaved- and independent-transfer modes



In the right-hand figure, the DFLT0 and DFLT1 filters are configured in interleaved-transfer mode. For those filters, FTH is no longer taken into account by the hardware. The samples provided by DFLT0 and DFLT1 are read from MDF_DFLT0DR. When both RXFIFOs are no longer empty, the MDF generates two DMA requests to read the sample from DFLT0 and the sample from DFLT1. If both RXFIFOs are not empty again, the same sequence is triggered.

Note: When one of the filter working in interleaved-transfer mode is disabled, the data transfer to memory of all filters in interleaved-transfer mode is stopped immediately as well.

RXFIFO overrun

A RXFIFO overrun condition is detected when the RXFIFO is full and a new sample from the DFLTx must be written.

In this case, DOVRF is set and the new sample is dropped. When the RXFIFO has at least one location available, the new incoming sample is written into the RXFIFO.

Figure 264 shows an example based on a RXFIFO depth of four words and FTH set to 1, so that FTHF goes to 1 when the RXFIFO is half-full.

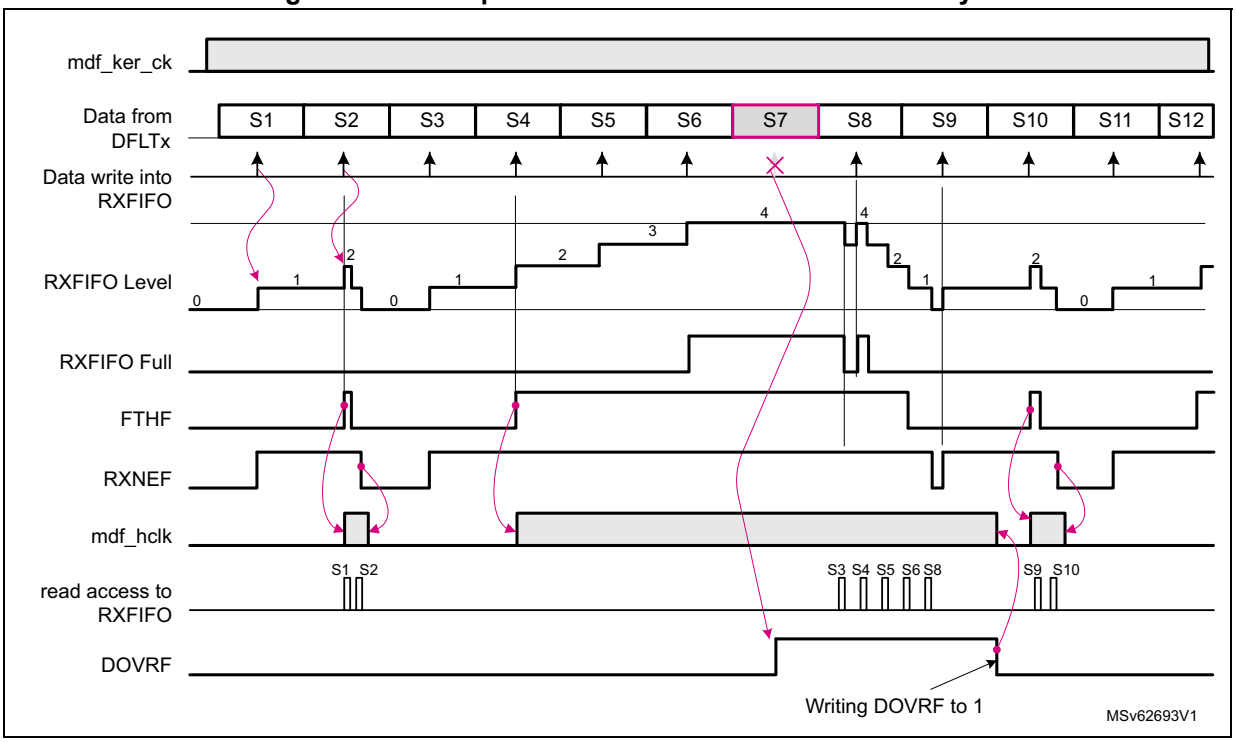
The S7 sample is lost due to an overrun: the RXFIFO is full while S7 must be written into the RXFIFO. The S7 write operation is not performed. DOVRF is set to 1 at the moment where the write operation was expected. The overflow event remains to 1 as long as it is not cleared by the application.

In this example, DOVRIE is set to 1 to have an interrupt if an overrun condition is detected.

After the S7 sample, the application manages to read data from the RXFIFO and the MDF can write the S8 sample and consecutive. Later, the application clears DOVR, allowing the detection of a new overrun situation.

In the mdf_hclk line, the gray boxes indicate that the MDF requested the AHB clock. The figure below shows the AHB clock available only when the MDF requests it. In real applications, the AHB clock may also be present if the MDF does not request it.

Figure 264. Example of overflow and transfer to memory



Note: If the MDF works in interleaved-transfer mode, the application must check the overrun status of all RXFIFOs that works in interleaved-transfer mode.

35.4.14 Autonomous mode

The MDF can work even if the AHB bus clock is not available (Stop modes). The MDF uses the AHB clock only for the register interface. All the processing part is clocked with the kernel clock.

In Stop mode, the MDF receives a kernel clock if the following conditions are met:

- The MDF autonomous mode is enabled in the RCC.
- The selected kernel clock source is taken from an oscillator available in Stop mode.

In Stop mode, the MDF receives the AHB clock if the following conditions are met:

- The MDF autonomous mode is enabled in the RCC.
- The MDF requests the AHB clock, in the following situations:
  - when the MDF must transfer data into the memory via the DMA

    The data are directly transferred to the SRAM thanks to the DMA while the product remains in Stop mode. The AHB clock request is maintained until the DMA transfer is completed.
  - when the MDF must generate an interrupt

    An interrupt generally wakes up the device from Stop mode, as an action from the application is needed. Once the AHB clock is available, the interrupt is generated.The AHB clock request is maintained as long as an enabled interrupt flag is active. More precisely, when an interrupt must be generated, the MDF requests the AHB clock.

35.4.15 Register protection

The MDF embeds some hardware protection to prevent invalid situations. The table below shows the list of write-protected and unprotected fields.

Table 304. Register protection summary

Registers	Unprotected fields	Write-protected fields	Write-protection condition
MDF global control register (MDF_GCR)	TRGO	ILVNB[3:0]	DFLTACTIVE0 = 1
MDF clock generator control register (MDF_CKGCR)	CKGDEN CCK0EN CCK1EN	PROCDIV[6:0], CCKDIV[3:0], CKGMOD, TRGSRC[3:0], TRGSENS, CCK[1:0]DIR	CKGACTIVE = 1
MDF serial interface control register x (MDF_SITFxCR)	SITFEN	STH[4:0], SITFMOD[1:0], SCKSRC[1:0]	SITFACTIVEx = 1
MDF bitstream matrix control register x (MDF_BSMXxCR)	-	BSSEL[4:0]	DFLTACTIVEx = 1 or SCDACTIVEx = 1 or OLDACTIVEx = 1
MDF digital filter control register x (MDF_DFLTxCR)	DFLTEN	NBDIS[7:0], TRGSRC[3:0], TRGSENS, FTH, DMAEN, SNPSFMT, ACQMOD[2:0]	DFLTACTIVEx = 1
MDF digital filter configuration register x (MDF_DFLTxCICR)	SCALE[5:0]	MCICD[8:0], CICMOD[2:0], DATSRC[1:0]	
MDF reshape filter configuration register x (MDF_DFLTxRSFR)	-	All fields	
MDF integrator configuration register x (MDF_DFLTxINTR)	-	All fields	

Table 304. Register protection summary (continued)

Registers	Unprotected fields	Write-protected fields	Write-protection condition
MDF out-of limit detector control register x (MDF_OLDxCR)	OLDEN	ACICD[4:0], ACICN[1:0],THINB BKOLD[3:0]	OLDACTIVEx = 1
MDF OLDx low threshold register x (MDF_OLDxTHLR)	-	All fields	
MDF OLDx high threshold register x (MDF_OLDxTHHR)	-	All fields	
MDF delay control register x (MDF_DLYxCR)	-	SKPDLY[6:0]	SKPBF = 1
MDF short circuit detector control register x (MDF_SCDxCR)	SCDEN	BKSCD[3:0], SCDT[7:0]	SCDACTIVEx = 1
MDF DFLTx interrupt enable register x (MDF_DFLTxIER)	All fields	-	-
MDF DFLTx interrupt status register x (MDF_DFLTxISR)	All fields	-	-
MDF offset error compensation control register x (MDF_OECxCR)	OFFSET[25:0]	-	-

All the MDF processing is performed in the mdf_proc_ck clock domain. For that reason, enabling or disabling a MDF sub-block may take some time due to the re-synchronization between the AHB clock domain and the mdf_proc_ck clock domain. XXXACTIVE flags are available to allow the application to check that the synchronization between the two clock domains is completed.

To change a write-protected field, the application must follow this sequence:

1. Set the enable bit of the sub-block to 0.
2. Wait for corresponding flag XXXACTIVE = 0.
3. Modify the wanted fields.
4. Set the enable bit of the sub-block to 1.

Refer to the description of each sub-block for more details.

35.5 MDF low-power modes

Table 305. Effect of low-power modes on MDF

Mode	Description
Sleep	No effect. MDF interrupts cause the device to exit Sleep mode.
Stop(1)	The MDF registers content is kept. If the MDF is clocked by an internal oscillator available in Stop mode, the MDF remains active. The interrupts cause the device to exit Stop mode.
Standby	The MDF is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 35.3: MDF implementation* for details about Stop modes supported by the MDF.

35.6 MDF interrupts

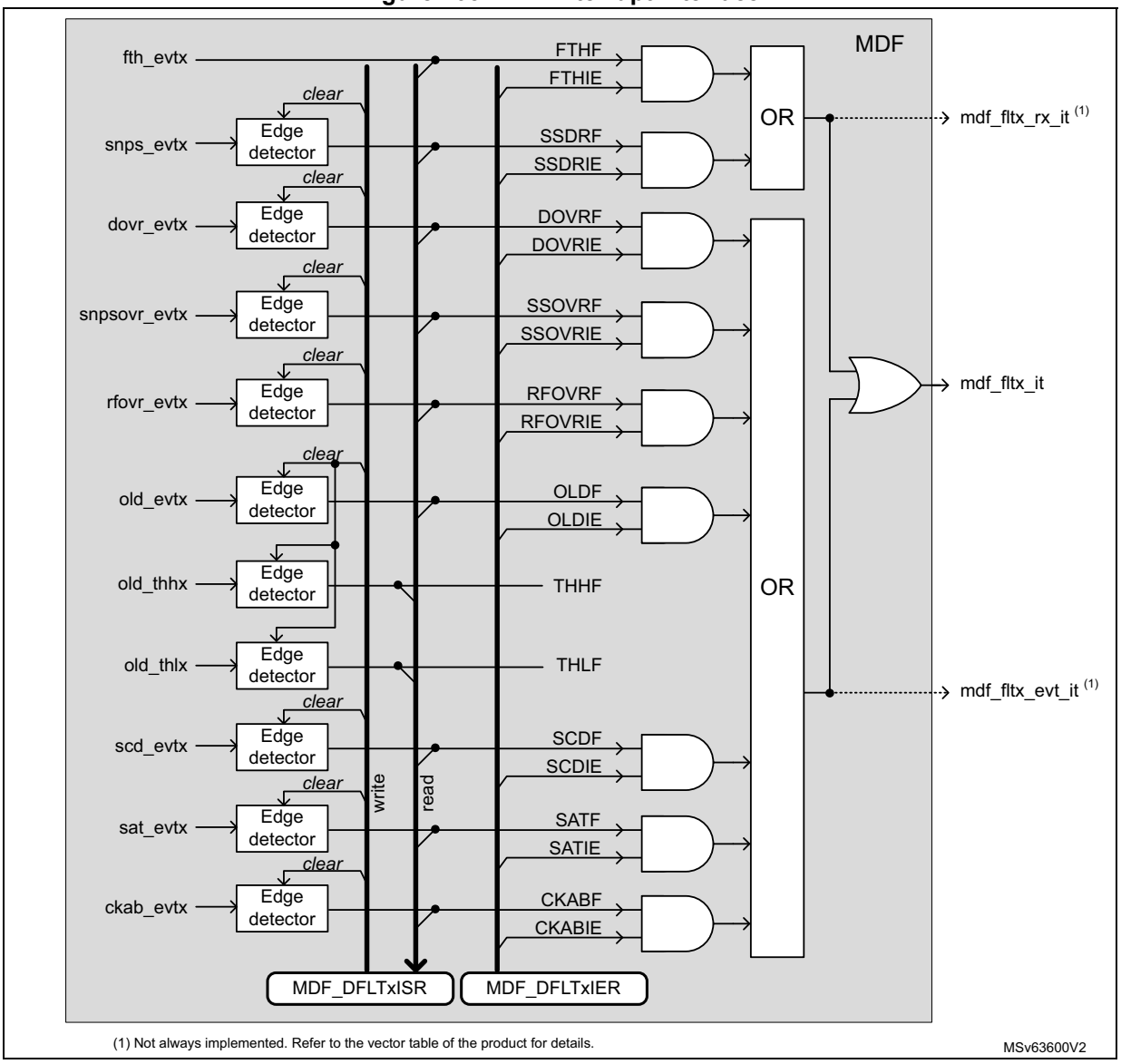
To increase the CPU performance, the MDF offers the following interrupt lines per digital filter:

- receive interrupt MDF_FLTx_RX (mdf_fltx_rx_it)
- event interrupt MDF_FLTx_EVT (mdf_fltx_evt_it)
- a combination of both interrupts MDF_FLTx (mdf_fltx_it)

Note: Interrupts are not always connected to the device (see Section 35.3: MDF implementation for more details).
The status flags are available even if the corresponding interrupt enable flag is not enabled.

The interrupt interface is controlled via the *MDF DFLTx interrupt enable register x (MDF_DFLTxIER)* and the *MDF DFLTx interrupt status register x (MDF_DFLTxISR)*.

Figure 265. MDF interrupt interface



The table below shows which interrupt line is affected by which event, and how to clear and activate each interrupt/event.

Table 306. MDF interrupt requests

Interrupt vectors		Interrupt event	Event flag	Event/interrupt clearing method	Exit Sleep mode	Exit Stop modes[1]	Exit Standby mode
MDF_FLTx[2]	MDF_FLTx_RX[3]	RXFIFO threshold reached	FTHF	Read MDF_DFLTxDR until RXFIFO level is lower than the threshold.	Yes	Yes	No
		Snapshot data ready	SSDRF	Write SSDRF to 1.			
	MDF_FLTx_EVT[4]	Snapshot data overrun	SSOVRF	Write SSOVRF to 1.			
		RXFIFO overrun	DOVRF	Write DOVRF to 1.			
		RSFLT overrun	RFOVRF	Write RFOVRF to 1.			
		Short-circuit detector	SCDF	Write SCDF to 1.			
		Saturation detection	SATF	Write SATF to 1.			
		Channel clock absence detection	CKABF	Write CKABF to 1.			
		Out-of-limit detector	OLDF	Write OLDF to 1.			
-	-		THHF	Write OLDF to 1.	-	-	-
-	-		THLF	Write OLDF to 1.	-	-	-

1. Refer to *Section 35.3: MDF implementation* for details.
2. MDF_FLTx vector corresponds to the assertion of mdf_fltx_it signal
3. MDF_FLTx_RX vector corresponds to the assertion of mdf_fltx_rx signal.
4. MDF_FLTx_EVT vector corresponds to the assertion of mdf_fltx_evt signal.

35.7 MDF application informations

35.7.1 MDF configuration examples for audio capture

Table 307 gives some examples of the MDF settings for the digital microphones, focusing on 16 and 48 kHz output data rate. In these examples, the following is expected:

- The INT block is bypassed (INTVAL = 0).
- The offset error compensation is disabled (OFFSET = 0).

Configurations #1 and #2 are for very low-power use-cases and have a reduced signal-to-noise ratio. The user must also insure that the selected digital microphone can work properly at 512 kHz. These configurations can be used for sound detection. The RSFLT is not used to reduce as much as possible the frequency of the kernel clock (mdf_ker_ck).

Configurations #3, #4, #9, #10, #11 give signal-to-noise ratios around 115 dB, with an ideal microphone model, with a sinus signal of 997 Hz. Using the RSFLT allows a good control on the in-band ripple and a good image rejection.

Configurations #7, #8, #10 give signal-to-noise ratio around 120 dB, with an ideal microphone model, using a sinus signal of 997 Hz.

Table 307. Examples of MDF settings for microphone capture

Configuration	mdf_ker_ck (MHz)	PROCDIV + 1	CCKDIV + 1	CIC order (1)	MCICD + 1	SCALE	RSFLTBYP	RSFLTD	HPFBYP	-	mdf_proc_ck (MHz)	Total dec. ratio	F_{RS} (kHz)	F_{MDF_CCKx} (MHz)	F_{PCM} (kHz)
#1	1.024	1	2	4	64	0x2D (- 8.5 dB)	1	x	x	=>	1.024	64	-	0.512	8
#2		1	2	5	32	0x2B (- 14.5 dB)						32	-		16
#3		1	2	5	16	0x01 (+ 3.5 dB)	0	0				64	32		8
#4	2.048	1	2	5	16	0x01 (+ 3.5 dB)	0	0			2.048	64	64	1.024	16
#5	3.072	1	6	5	8	0x0B (+ 33.6 dB)	0	0			3.072	32	32	0.512	8
#6		2	2	5	12	0x06 (+ 18.1 dB)						48	64	0.768	16
#7		1	2	5	24	0x2C (- 12 dB)						96		1.536	
#8	4.096	1	2	5	32	0x27 (- 26.6 dB)	0	0			4.096	128	64	2.048	16
#9	6.144	3	2	5	16	0x02 (+ 6.0 dB)	0	0			2.048	64	64	1.024	16
#10		2	2	5	24	0x2C (- 12 dB)					3.072	96		1.536	
#11		1	2	5	16	0x01 (+ 3.5 dB)					6.144	64	192	3.072	48
#12	7.680	1	2	5	20	0x2E (- 6.0 dB)	0	0			7.680	80	192	3.840	48

1. CICMOD = 100 for CIC order equal to 4. CICMOD = 101 for CIC order equal to 5.

35.7.2 Programming examples

This example describes how to capture sound from four microphones, assuming that each microphone pair shares the same data line. The MDF_SD0 and MDF_SD2 data lines are used. The microphone clock is provided by the MDF via MDF_CCK0 pin.

Table 308. Programming sequence

Operations	Comments
Adjust the proper kernel clock frequency via the RCC block	Assuming that the RCC is programmed to provide a kernel clock (mdf_ker_ck) of 12.288 MHz
Select the proper MDF kernel clock source via the RCC block	Refer to the RCC of the product.
Enable the MDF clocks via the RCC block	Refer to the RCC of the product.
Reset the MDF via the RCC block	Refer to the RCC of the product.
AFMUX programming	Program the AFMUX to select MDF_SD0, MDF_SD2 and MDF_CCK0 function.

Table 308. Programming sequence (continued)

Operations	Comments
Enable MDF processing clock: MDF_CKGCR = 0x0103 0023	PROCDIV = 1 (division by 2): mdf_proc_ck frequency is 6.144 MHz. CCKDIV = 3 (division by 4): MDF_CCK0 clock frequency is 1.536 MHz. The MDF_CCK0 pad is set in output and generates a clock so that the microphones can exit from low-power mode.
Serial interfaces configuration: MDF_SITF0CR = 0x0000 1F01 MDF_SITF2CR = 0x0000 1F01	SCKSRC = 0 to select MDF_CCK0 as serial clock. SIFTMOD = 0 to select LF_MASTER mode. Clock absence feature is not working in this mode. The serial interfaces are enabled.
Bitstream matrix configuration: MDF_BSMX0CR = 0x0000 0000 MDF_BSMX1CR = 0x0000 0001 MDF_BSMX2CR = 0x0000 0004 MDF_BSMX3CR = 0x0000 0005	DFLT0 filter takes the bitstream of SITF0, sampled on rising edge. DFLT1 filter takes the bitstream of SITF0, sampled on falling edge. DFLT2 filter takes the bitstream of SITF2, sampled on rising edge. DFLT3 filter takes the bitstream of SITF2, sampled on falling edge.
Filters configuration (CIC): MDF_DFLT0CICR = 0x02C0 1750 MDF_DFLT1CICR = 0x02C0 1750 MDF_DFLT2CICR = 0x02C0 1750 MDF_DFLT3CICR = 0x02C0 1750	SCALE = 0x2C (- 12 dB) to avoid any saturation MCICD = 0x17 (decimation by 24) CICMOD = 5 to select a $Sinc^5$ DATSCR = 0 to select data coming from BSMX
Filters configuration (RSFLT and HPF): MDF_DFLT0RSFR = 0x0000 0100 MDF_DFLT1RSFR = 0x0000 0100 MDF_DFLT2RSFR = 0x0000 0100 MDF_DFLT3RSFR = 0x0000 0100	HPFC = 1: cut-off frequency of 16 kHz * 0.00125 = 20 Hz HPFBYP = 0: HPF not bypassed RSFLTD = 0: RSFLT decimates by 4 RSFLTBYP = 0: RSFLT is not bypassed
Filters configuration (INT): MDF_DFLT0INTR = 0x0000 0000 MDF_DFLT1INTR = 0x0000 0000 MDF_DFLT2INTR = 0x0000 0000 MDF_DFLT3INTR = 0x0000 0000	INTVAL = 0: INT filter not used Other parameter is not significant.
Micro delay adjust: MDF_DLY0CR = 0x0000 0005 MDF_DLY0CR = 0x0000 0012 MDF_DLY0CR = 0x0000 0023 MDF_DLY0CR = 0x0000 0000	Initial micro-delay for each microphone, values just given as example
Offset error correction: MDF_OEC0CR = 0x0000 0000 MDF_OEC1CR = 0x0000 0000 MDF_OEC2CR = 0x0000 0000 MDF_OEC3CR = 0x0000 0000	No correction. DC offset is removed by HPF.

Table 308. Programming sequence (continued)

Operations	Comments
Short circuit detection: MDF_SCD0CR = 0x0000 0000 MDF_SCD1CR = 0x0000 0000 MDF_SCD2CR = 0x0000 0000 MDF_SCD3CR = 0x0000 0000	SCD function for each filter is disabled.
Enable interrupt events: MDF_DFLT0IER = 0x0000 0202 MDF_DFLT1IER = 0x0000 0202 MDF_DFLT2IER = 0x0000 0202 MDF_DFLT3IER = 0x0000 0202	Enable the interrupt event the application wants to handle. In this example, the SATIE and DOVRIE bits are set to 1 to have an interrupt if a saturation or an data overflow occurs.
Digital filter control: MDF_DFLT0CR = 0x0000 0027 MDF_DFLT1CR = 0x0000 0027 MDF_DFLT2CR = 0x0000 0027 MDF_DFLT3CR = 0x0000 0027 Wait for DFLTACTIVE = 0 for the filters	NBDIS = 0: no samples discarded TRGSRC = 0: TRGO selected as trigger source TRGSENS = 0: Trigger on rising edge ACQMOD = 2: Synchronous continuous acquisition mode DMAEN = 1: DMA interface enabled DFLTEN = 1: digital filter enabled
Clear status flags: MDF_DFLT0ISR = 0x0000 0FFF MDF_DFLT1ISR = 0x0000 0FFF MDF_DFLT2ISR = 0x0000 0FFF MDF_DFLT3ISR = 0x0000 0FFF	Clear all the status flags before running the filters.
Program the DMA Enable the DMA	The DMA must be programmed in order to read the data inside MDF_DFLT0DR every time a DMA request is generated. Note that when the MDF is in interleaved acquisition mode, data of filters 0, 1, 2, and 3 are read via MDF_DFLT0DR register.
Start acquisition: MDF_GCR = 0x0000 0031	ILVNB = 3: interleaved mode with DFLT0,1,2, and 3 TRGO = 1 to trigger the acquisition of all filters waiting for TRGO rising edge event

35.7.3 Connection examples

Figure 266 shows simple connection examples of the MDF to external sensors:

- Picture on the left: four digital microphones connected to the MDF

  In this connection, DMIC1 is intended to be used alone. This is why it uses a dedicated clock (MDF_CCK0). DMIC1 is also connected to a dedicated data line in case the application intends to power-down DMIC2/3/4, when only DMIC1 is used. If the application keeps DMIC2/3/4 powered, DMIC2 and DMIC1 can share the same data line, reducing the connection to four I/Os.

  If the power consumption is not an issue, when a single microphone is used, all the microphones can be connected to the same common clock (MDF_CCK0) and each

microphone pair can share the same data line. In this case, only three I/Os are required.

In this example, the data transfer to memory can be performed either using the interleaved- or the independent-transfer mode.

- Picture in the center: sensor providing several data lines with a common clock

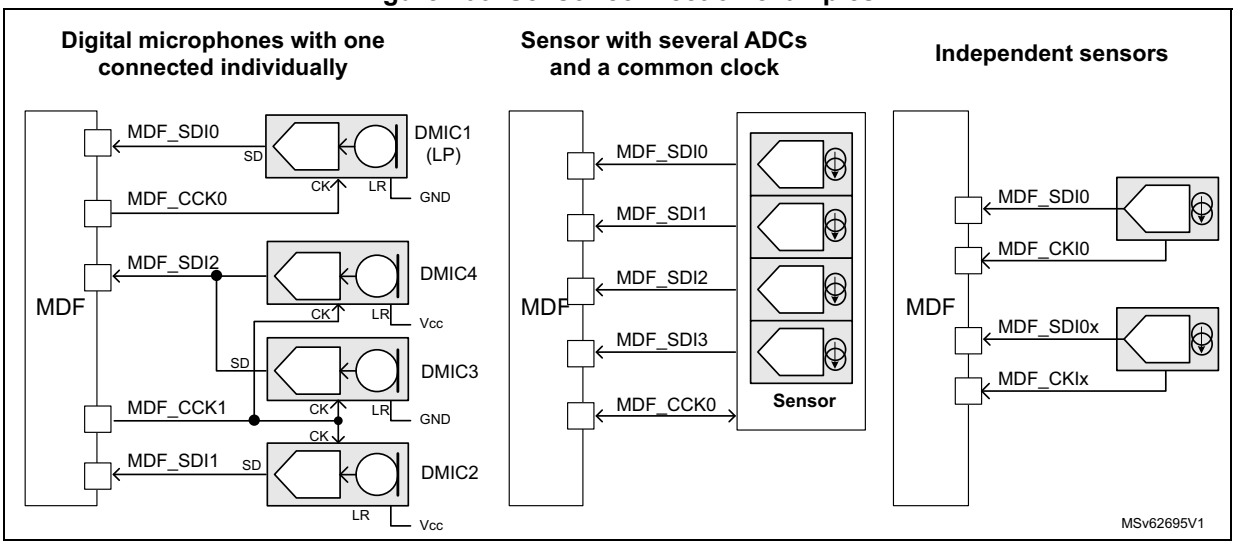
  Each data line can represent different parameters (such as current or voltage). The common clock can be provided either by the sensor or by the MDF. The data line can be shared or not by two sensors if the sensor allows it. In the figure below, the sensor does not allow the sharing of the data lines.

  In this example, the data transfer to memory can be performed either using the interleaved- or independent-transfer mode.

- Picture on the right: two independent sensors connected to the MDF

  Each of them has its dedicated clock and data lines. In this case, the data transfer to memory must use the independent-transfer mode.

Figure 266. Sensor connection examples



35.7.4 Global frequency response

Figure 267 shows the global frequency response for a 16 kHz audio signal with a digital microphone working at 1.024 MHz. The filter configuration is the following:

- CIC order 4 or 5, with a decimation ratio of 16
- RSFLT enabled, with a decimation ratio of 4
- HPF enable with a cut-off frequency of 40 Hz

The figure below shows the theoretical frequency response using a CIC4 and a CIC5.

Figure 267. Global frequency response

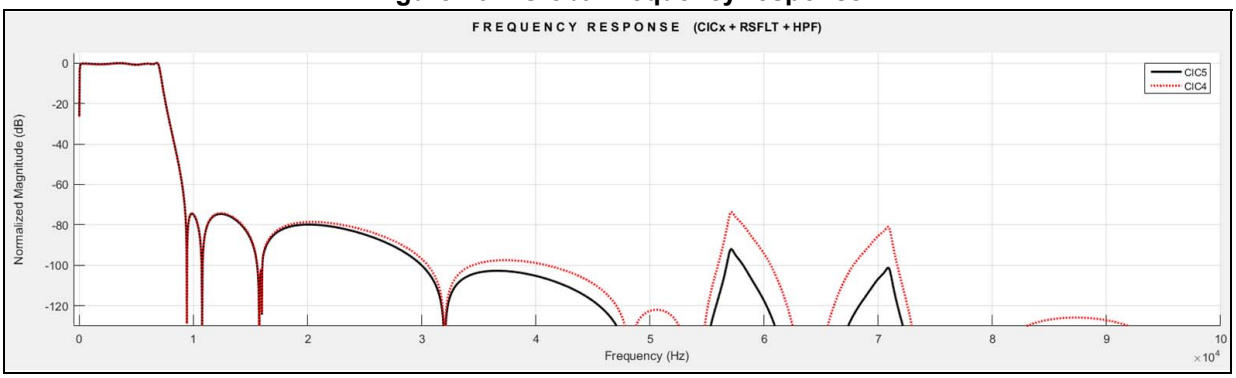


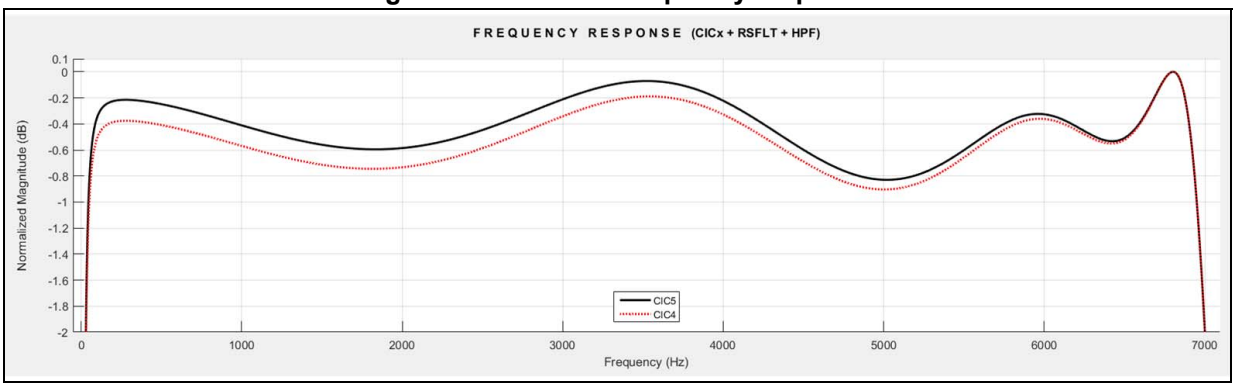
Figure 268 shows the in-band ripple for a 16 kHz audio signal with a digital microphone working at 1.024 MHz. The filter configuration is the following:

- CIC order 4 or 5, with a decimation ratio of 16
- RSFLT enabled, with a decimation ratio of 4
- HPF enable with a cut-off frequency of 20 Hz

The resulting in-band ripple is ± 0.41 dB for CIC5, and ± 0.45 for CIC4.

The -3 dB cut-off frequency is 7061 Hz.

Figure 268. Detailed frequency response



35.7.5 Total MDF gain

This section details how to compute the signal level provided by the MDF according to the filter settings. The formula does not take into account the filters transfer function.

A signal level may be expressed in dBFS (decibel full scale). A 0 dBFS level is assigned to the maximum possible digital level. For example, a signal that reaches 50 % of the maximum level, has a - 6 dBFS level (6 dB below full scale).

For example, for the MDF offering a final data width of 24 bits, a signal having an amplitude of 2 * 10^6 LSB has a level of:

$$20 \times \log_{10}\left(\frac{2 \times 10^6}{2^{(24-1)}}\right) = -12.45 \text{ dBFS}$$

In addition, the data size of a signal having an amplitude (Amp) expressed in LSB is given by:

$$DS = \left(\frac{\ln(Amp)}{\ln(2)} + 1\right) \text{ bits}$$

One bit need to be added for negative values.

So a signal having an amplitude of 2 * 10^6 LSB, has a data size of 21.9 bits.

CIC gain

The CIC gain (G_{CIC} and GdB_{CIC}) can be deduced from the following formula giving data size in bits (DS_{CIC}).

$$DS_{CIC} = (N \times \log_2(D1)) + DS_{IN}$$

And the bit growth is:

$$BG_{CIC} = (N \times \log_2(D1))$$

where N represents the CIC order (selected by CICMOD[2:0]), and D1 is the decimation ratio (given by MCICD).

DS_{IN} represents the data size (in bits) of the input signal.

Warning: DS_{CIC} is very important for CIC filters. In order to work fine, DS_{CIC} must not exceed 26 bits.

$$G_{CIC} = 2^{(BG_{CIC})} = (D1)^N$$

which gives, in decibels:

$$GdB_{CIC} = 20 \times \log_{10}((D1)^N)$$

Note: The same formulas are valid for the ACIC.

Data size at SCALE output

The data size at SCALE output (including the CIC gain) is a key information as the RSFLT starts to have some saturations if the peak-to-peak signal amplitude at SCALE output is higher than 22 bits.

If the RSFLT is bypassed, then a peak-to-peak signal amplitude of 24 bits is accepted. The resulting data size is given by:

$$DS_{SCALE} = N \times \log_2(D1) + \log_2\left(10^{\frac{GDB_{SCALE}}{20}}\right) + DS_{IN}$$

The data size at SCALE output (DS_{SCALE}) is expressed in bits and GdB_{SCALE} represents the gain selected by SCALE[5:0], in dB.

RSFLT gain

The RSFLT gain in the useful bandwidth is typically 9.5 dB, but due to ripple a margin of about ± 0.5 dB must be considered. Typically, the RSFLT increases the bit size by BG_{RSFLT}:

$$BG_{RSFLT} = 10^{\frac{9.5\ dB}{20}} = 2.98 = 1.6\ bits$$

INT gain

The INT block can also introduce a gain if the rescaling value is different from the integration value.

$$G_{INT} = \frac{IVAL}{IDIV}$$

and:

$$GdB_{INT} = 20 \times \log_{10}\left(\frac{IVAL}{IDIV}\right)$$

The bit growth of the INT is then given by the following formula:

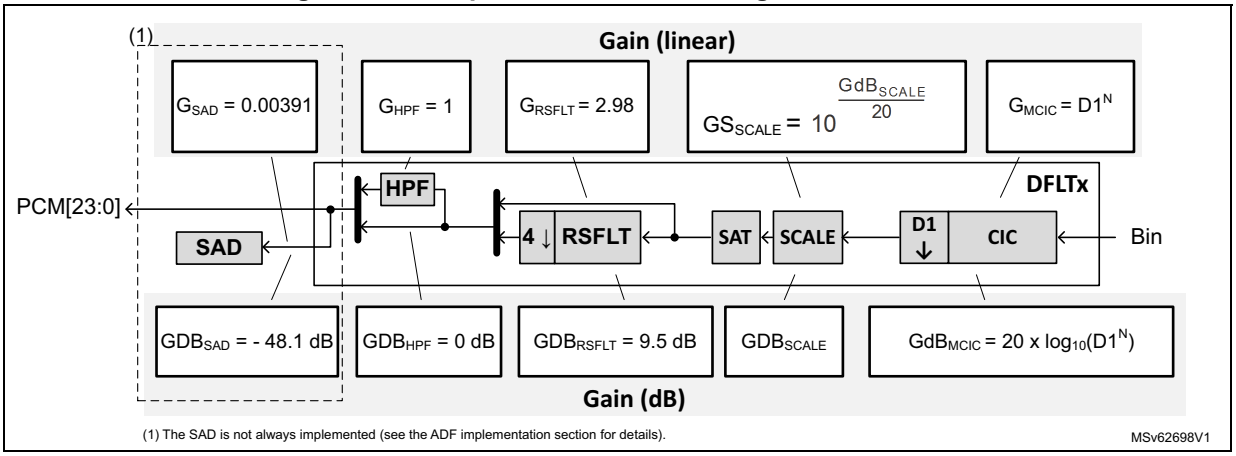
$$BG_{INT} = \log_2\left(\frac{IVAL}{IDIV}\right)$$

IVAL represents the integration value selected by INTVAL[6:0], and IDIV represents the integrator output division selected by INTDIV[1:0].

Note: The HPF filter has a gain of 0 dB.

The figure below shows a simplified view of the filter path, and gives for each significant component the expression of the bit growth and the gain.

Figure 269. Simplified DFLT view with gain information



The table below summarizes the final data size for different filter configurations.

Table 309. Output signal levels

Filter configurations	Final signal level (dBFS)	Final signal size (bits)
CIC + RSFLT (+ HPF)	$SdB_{OUT} = 20 \times \log_{10}\left(\frac{2^{DS_{OUT}}}{2^{24}}\right)$	$DS_{OUT} = DS_{SCALE} + 1.6$ bits DS_{SCALE} must be lower than 22 bits.
CIC + RSFLT (+ HPF) + INT		$DS_{OUT} = DS_{SCALE} + BG_{INT} + 1.6$ bits DS_{SCALE} must be lower than 22 bits.
CIC (+ HPF) + INT		$DS_{OUT} = DS_{SCALE} + BG_{INT}$ DS_{SCALE} must be lower than 24 bits.

Example using the main filter chain

If the MDF filter is programmed as follows:

- The input signal is coming from a serial interface (DS_{IN} = 1 bit).
- CIC order = 5 (N), with a decimation value of 24 (D1).
- SCALE[5:0] is set to -12 dB.
- RSFLT is enabled and the decimation by four is enabled.
- HPF is enabled.
- INT is disabled.

Check first the data size at CIC output:

$$DS_{CIC} = (5 \times \log_2(24)) + 1 \text{ bit} = 23.92 \text{ bits}$$

The size is lower than 26 bits, so the CIC works in good conditions.

The data size at CIC output is very close to 24 bits, so the SCALE must be adjusted in order to provide a signal 22 bits max to the RSFLT. An attenuation of 12 dB is needed.

Then the signal level provided to the RSFLT is:

$$DS_{SCALE} = DS_{CIC} + \log_2\left(10^{\frac{-12}{20}}\right) = 23.92 - 1.99 = 21.93 \text{ bits}$$

At the end, the final signal amplitude is:

$$DS_{OUT} = DS_{SCALE} + 1.6 \text{ bits} = (21.93 + 1.6) = 23.52 \text{ bits}$$

or

$$SDB_{OUT} = 20 \times \log_{10}\left(\frac{2^{23.52}}{2^{24}}\right) = -2.84 \text{ dBFS}$$

The RSFLT ripple a margin of about ± 0.41 dB (in this configuration) must be considered.

Example using the OLD filter chain

In the following example, the application wants to trigger an OLD event when the voltage coming from a shunt resistor reaches ± 200 mV.

Hypothesis:

- The ΣΔ sensor is specified to support a full-scale signal of ± 350 mV.
- The ACIC decimation ratio (D2) is fixed to 32.
- The ACIC order (N) is fixed to 3.

The ΣΔ sensor provides a full-scale digital signal (amplitude of one bit) for a signal higher or equal to ± 350 mV. If the input signal is equal 200 mV, then the digital signal amplitude provided by the sensor is 200 / 350 = 0.571.

The gain of the ACIC filter is:

$G_{CIC} = (D2)^N = 32^3 = 32768$

A signal having an amplitude of 200 mV at sensor input has then an amplitude of about 32768 * 0.571 = 18725 LSB at ACIC output.

OLDTHH must be set to 18725 and OLDTHL must be set to -18725.

35.8 MDF registers

All the MDF registers must be accessed either in word (32-bit) or half-word (16-bit) formats.

35.8.1 MDF global control register (MDF_GCR)

Address offset: 0x000

Reset value: 0x0000 0000

This register is used for controls common to all digital filters.

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

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

Bits 7:4 **ILVNB[3:0]**: Interleaved number

This field is set and reset by software. it enables or disables the interleaved-transfer mode and defines how many digital filters work in this mode.

This field cannot be changed when DFLTEN = 1 in MDF_DFLT0CR.

0000: Interleaved-transfer mode disabled

0001: Data from DFLT0 and DFLT1 are interleaved.

0010: Data from DFLT0, DFLT1 and DFLT2 are interleaved.

...

1111: Data from DFLT0 to DFLT15 are interleaved.

Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **TRGO**: Trigger output control

This bit is set by software and reset by hardware. It is used to start the acquisition of several filters synchronously. It is also able to synchronize several MDF together by controlling the mdf_trgo signal.

0: Write 0 has no effect. Read 0 means that the trigger can be set again to 1.

1: Write 1 generates a positive pulse on mdf_trgo signal and triggers the acquisition on the enabled filters having ACQMOD[2:0] = 0x1 and selecting TRGO as trigger. Read 1 means that the trigger pulse is still active.

35.8.2 MDF clock generator control register (MDF_CKGCR)

Address offset: 0x004

Reset value: 0x0000 0000

This register is used to control the clock generator. The mdf_proc_ck clock must be enabled before enabling other MDF parts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CKGACTIVE	PROCDIV[6:0]							Res.	Res.	Res.	Res.	CCKDIV[3:0]			
r	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRGSRC[3:0]				Res.	Res.	Res.	TRGSENS	Res.	CCK1DIR	CCK0DIR	CKGMOD	Res.	CCK1EN	CCK0EN	CKGDEN
rw	rw	rw	rw				rw		rw	rw	rw		rw	rw	rw

Bit 31 **CKGACTIVE**: Clock generator active flag

This bit is set and cleared by hardware. This flag must be used by the application to check if the clock generator is effectively enabled (active) or not. The protected fields of this function can only be updated when CKGACTIVE = 0 (refer to *Section 35.4.15: Register protection* for details). The delay between a transition on CKGDEN and a transition on CKGACTIVE is two periods of AHB clock and two periods of mdf_proc_ck.

0: The clock generator is not active and can be configured if needed.

1: The clock generator is active and protected fields cannot be configured.

Bits 30:24 **PROCDIV[6:0]**: Divider to control the serial interface clock

This field is set and reset by software. It is used to adjust the frequency of the clock provided to the SITF.

$$F_{mdf\_itf\_ck} = \frac{F_{mdf\_ker\_ck}}{(PROCDIV + 1)}$$

This field must not be changed if one of the filters is enabled (DFTEN = 1).

0: mdf_ker_ck provided to the SITF

1: mdf_ker_ck/2 provided to the SITF

2: mdf_ker_ck/3 provided to the SITF

...

127: mdf_ker_ck/128 provided to the SITF

Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details).

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **CCKDIV[3:0]**: Divider to control the MDF_CCK clock

This field is set and reset by software. It is used to adjust the frequency of the MDF_CCK clock. The input clock of this divider is the clock provided to the SITF. More globally, the frequency of the MDF_CCK is given by the following formula:

$$F_{MDF\_CCK} = \frac{F_{mdf\_ker\_ck}}{(PROCDIV + 1) \times (CCKDIV + 1)}$$

This field must not be changed if one of the filters is enabled (DFTEN = 1).

0000: The MDF_CCK clock is mdf_proc_ck.

0001: The MDF_CCK clock is mdf_proc_ck / 2.

0010: The MDF_CCK clock is mdf_proc_ck / 3.

...

1111: The MDF_CCK clock is mdf_proc_ck / 16.

Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details).

Bits 15:12 **TRGSRC[3:0]**: Digital filter trigger signal selection

This field is set and cleared by software.It is used to select which external signals trigger for the corresponding filter. This field is not significant if CKGMOD = 0.

000x: TRGO selected

0010: mdf_trg[0] selected

0011: mdf_trg[1] selected

...

1111: mdf_trg[13] selected

Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details.)

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **TRGSENS**: CKGEN trigger sensitivity selection

This bit is set and cleared by software. It is used to select the trigger sensitivity of the trigger signals. This bit is not significant if the CKGMOD = 0.

0: A rising edge event triggers the activation of CKGEN dividers.

1: A falling edge even triggers the activation of CKGEN dividers.

Note: When the trigger source is TRGO, TRGSENS value is not taken into account. When TRGO is selected, the sensitivity is forced to falling edge. This bit can be write-protected (refer to Section 35.4.15: Register protection for details).

Bit 7 Reserved, must be kept at reset value.

Bit 6 **CCK1DIR**: MDF_CCK1 direction

This bit is set and reset by software.It is used to control the direction of the MDF_CCK1 pin.

0: MDF_CCK1 pin direction is in input.

1: MDF_CCK1 pin direction is in output.

Note: This bit can be write-protected (refer to Section 35.4.15: Register protection for details).

Bit 5 **CCK0DIR**: MDF_CCK0 direction

This bit is set and reset by software.It is used to control the direction of the MDF_CCK0.

0: MDF_CCK0 pin direction is in input.

1: MDF_CCK0 pin direction is in output.

Note: This bit can be write-protected (refer to Section 35.4.15: Register protection for details).

Bit 4 **CKGMOD**: Clock generator mode

This bit is set and reset by software. It is used to define the way the clock generator is enabled. This bit must not be changed if one of the filters is enabled (DFTEN = 1).

0: The kernel clock is provided to the dividers as soon as CKGDEN is set to 1.

1: The kernel clock is provided to the dividers when CKGDEN is set to 1 and the trigger condition met.

Note: This bit can be write-protected (refer to Section 35.4.15: Register protection for details).

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CCK1EN**: MDF_CCK1 clock enable

This bit is set and reset by software. It is used to control the generation of the bitstream clock on the MDF_CCK1 pin.

0: Bitstream clock not generated

1: Bitstream clock generated on the MDF_CCK1 pad

Bit 1 **CCK0EN**: MDF_CCK0 clock enable

This bit is set and reset by software.It is used to control the generation of the bitstream clock on the MDF_CCK0 pin.

0: Bitstream clock not generated

1: Bitstream clock generated on the MDF_CCK0 pad

Bit 0 **CKGDEN**: CKGEN dividers enable

This bit is set and reset by software. It is used to enable/disable the clock dividers of the CKGEN: PROCDIV and CCKDIV.

0: CKGEN dividers disabled

1: CKGEN dividers enabled

35.8.3 MDF serial interface control register x (MDF_SITFxCR)

Address offset: 0x080 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 1F00

This register is used to control the serial interfaces (SITFx).

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

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

Bit 31 **SITFACTIVE**: Serial interface active flag

This bit is set and cleared by hardware. It is used by the application to check if the serial interface is effectively enabled (active) or not. The protected fields of this function can only be updated when the SITFACTIVE is set to 0 (refer to *Section 35.4.15: Register protection* for details).

The delay between a transition on SITFEN and a transition on SITFACTIVE is two periods of AHB clock and two periods of mdf_proc_ck.

0: The serial interface is not active and can be configured if needed.

1: The serial interface is active and protected fields cannot be configured.

Bits 30:13 Reserved, must be kept at reset value.

Bits 12:8 **STH[4:0]**: Manchester symbol threshold/SPI threshold

This field is set and cleared by software. It is used for Manchester mode to define the expected symbol threshold levels (refer to *Manchester mode* for details on computation). In addition this field is used to define the timeout value for the clock absence detection in Normal SPI mode. STH[4:0] values lower than four are invalid.

Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details).

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **SITFMOD[1:0]**: Serial interface type

This field is set and cleared by software. it is used to define the serial interface type.

00: LF_MASTER SPI mode

01: Normal SPI mode

10: Manchester mode: rising edge = logic 0, falling edge = logic 1

11: Manchester mode: rising edge = logic 1, falling edge = logic 0

Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details).

Bit 3 Reserved, must be kept at reset value.

Bits 2:1 **SCKSRC[1:0]**: Serial clock source

This field is set and cleared by software. it is used to select the clock source of the serial interface.

00: Serial clock source is MDF_CCK0.

01: Serial clock source is MDF_CCK1.

1x: Serial clock source is MDF_CKIx (not allowed in LF_MASTER SPI mode).

Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details).

Bit 0 **SITFEN**: Serial interface enable

This bit is et and cleared by software. It is used to enable/disable the serial interface.

0: Serial interface disabled

1: Serial interface enabled

35.8.4 MDF bitstream matrix control register x (MDF_BSMXxCR)

Address offset: 0x084 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register is used to select the bitstream to be provided to the corresponding digital filter and to the SCD.

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

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

Bit 31 **BSMXACTIVE**: BSMX active flag

This bit is set and cleared by hardware. It is used by the application to check if the BSMX is effectively enabled (active) or not. BSSEL[4:0] can only be updated when the BSMXACTIVE is set to 0. Th is BSMXACTIVE flag is a logical OR between OLDACTIVE, DFLTACTIVE, and SCDACTIVE flags. Both of them must be set to 0 in order update BSSEL[4:0] field.

0: BSMX is not active and can be configured if needed.

1: BSMX is active and protected fields cannot be configured.

Bits 30:5 Reserved, must be kept at reset value.

Bits 4:0 **BSSEL[4:0]**: Bitstream Selection

This field is set and cleared by software. It is used to select the bitstream to be processed for DFLTx and SCDx. The size of this field depends on the number of DFLTx instantiated. If this field selects a not instantiated input, the MDF selects the valid stream bsx_f having the higher index number.

00000: bs0_r provided to DFLTx and SCDx

00001: bs0_f provided to DFLTx and SCDx

00010: bs1_r provided to DFLTx and SCDx (if instantiated)

00011: bs1_f provided to DFLTx and SCDx (if instantiated)

...

11110: bs15_r provided to DFLTx and SCDx (if instantiated)

11111: bs15_f provided to DFLTx and SCDx (if instantiated)

Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details).

35.8.5 MDF digital filter control register x (MDF_DFLTxCR)

Address offset: 0x088 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DFLTACTIVE	DFLTRUN	Res.	Res.	NBDIS[7:0]								Res.	Res.	Res.	SNPSFMT
r	r			rw	rw	rw	rw	rw	rw	rw	rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRGSRC[3:0]				Res.	Res.	Res.	TRGSENS	Res.	ACQMOD[2:0]			Res.	FTH	DMAEN	DFLTEN
rw	rw	rw	rw				rw		rw	rw	rw		rw	rw	w

Bit 31 **DFLTACTIVE**: Digital filter active flag

This bit is set and cleared by hardware. It indicates if the digital filter is active: can be running or waiting for events.

0: Digital filter not active (can be re-enabled again, via DFLTEN, if needed)

1: Digital filter active

Bit 30 **DFLTRUN**: Digital filter run status flag

This bit is set and cleared by hardware. It indicates if the digital filter is running or not.

0: Digital filter not running and ready to accept a new trigger event

1: Digital filter running

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

Bits 27:20 **NBDIS[7:0]**: Number of samples to be discarded

This field is set and cleared by software. it is used to define the number of samples to be discarded every time the DFLTx is re-started.

0: No sample discarded

1: 1 sample discarded

2: 2 samples discarded

...

255: 255 samples discarded

Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details).

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

Bit 16 **SNPSFMT**: Snapshot data format

This bit is set and cleared by software. It is used to select the data format for the snapshot mode.

0: Integrator counter (INT_CNT) not inserted into MDF_SNPSxDR, leaving a data resolution of 23 bits

1: Integrator counter (INT_CNT) inserted at position [15:9] of MDF_SNPSxDR, leaving a data resolution of 16 bits.

Note: This bit can be write-protected (refer to Section 35.4.15: Register protection for details).

Bits 15:12 **TRGSRC[3:0]**: Digital filter trigger signal selection
This field is set and cleared by software. It is used to select which external signals trigger the corresponding filter.
0000: TRGO selected
0001: OLDx event selected
0010: mdf_trg[0] selected
...
1111: mdf_trg[13] selected
Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details).

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **TRGSENS**: Digital filter trigger sensitivity selection
this bit is set and cleared by software. It is used to select the trigger sensitivity of the external signals
0: A rising edge event triggers the acquisition.
1: A falling edge even triggers the acquisition.
Note: When the trigger source is TRGO or OLDx event, TRGSENS value is not taken into account. When TRGO is selected, the sensitivity is forced to falling edge, when OLDx event is selected, the sensitivity is forced to rising edge.
This bit can be write-protected (refer to Section 35.4.15: Register protection for details).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ACQMOD[2:0]**: Digital filter trigger mode
This field is set and cleared by software. It is used to select the filter trigger mode.
000: Asynchronous continuous acquisition mode
001: Asynchronous single-shot acquisition mode
010: Synchronous continuous acquisition mode
011: Synchronous, single-shot acquisition mode
100: Window continuous acquisition mode
101: Synchronous snapshot mode
Others: same a 000
Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details).

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FTH**: RXFIFO Threshold selection

This bit is set and cleared by software. It is used to select the RXFIFO threshold. This bit is not significant for RXFIFOs working in interleaved-transfer mode (see *Interleaved-transfer mode* for details).

0: RXFIFO threshold event generated when the RXFIFO is not empty

1: RXFIFO threshold event generated when the RXFIFO is half-full

Note: This bit can be write-protected (refer to Section 35.4.15: Register protection for details).

Bit 1 **DMAEN**: DMA requests enable

This bit is set and cleared by software. It is used to control the generation of DMA request to transfer the processed samples into the memory.

0: DMA interface for the corresponding digital filter disabled

1: DMA interface for the corresponding digital filter enabled

Note: This bit can be write-protected (refer to Section 35.4.15: Register protection for details).

Bit 0 **DFLTEN**: Digital filter enable

This bit is set and cleared by software. It is used to control the start of acquisition of the corresponding digital filter path. This bit behavior depends on ACQMOD[2:0]and external events. The serial or parallel interface delivering the samples must be enabled as well.

0: Acquisition immediately stopped

1: Acquisition immediately started if ACQMOD[2:0] = 00x or 101, or acquisition started when the proper trigger event occurs if ACQMOD[2:0] = 01x or 100.

35.8.6 MDF digital filter configuration register x (MDF_DFLTxCICR)

Address offset: 0x08C + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register is used to control the main CIC filter.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SCALE[5:0]						Res.	Res.	Res.	MCICD8
						rw	rw	rw	rw	rw	rw				rw

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

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:20 **SCALE[5:0]**: Scaling factor selection

This field is set and cleared by software. It is used to select the gain to be applied at CIC output (refer to *Table 300* for details). If the application attempts to write a new gain value while the previous one is not yet applied, this new gain value is ignored. Reading back this field informs the application on the current gain value.

000000: 0 dB
000001: + 3.5 dB
000010: + 6 dB or shift left by 1 bit
...
011000: + 72 dB or shift left by 12 bits
100000: - 48.2 dB or shift right by 8 bits (default value)
100001: - 44.6 dB
100010: - 42.1 dB or shift right by 7 bits
100011: - 38.6 dB
...
101110: - 6 dB or shift right by 1 bit
101111: - 2.5 dB
Others: Reserved

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

Bits 16:8 **MCICD[8:0]**: CIC decimation ratio selection

This field is set and cleared by software. It is used to select the CIC decimation ratio. A decimation ratio smaller than two is not allowed. The decimation ratio is given by (CICDEC+1).

0: Decimation ratio is 2.
1: Decimation ratio is 2.
2: Decimation ratio is 3..
3: Decimation ratio is 4
...
511: Decimation ratio is 512.

Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **CICMOD[2:0]**: Select the CIC mode

This field is set and cleared by software.It is used to select the configuration and the order of the MCIC. When CICMOD[2:0] = 0xx, the CIC is split into two filters: the main CIC (MCIC) and the auxiliary CIC (ACIC, used for the out-off limit detector).

000: CIC split in two filters and MCIC configured in FastSinc filter
001: CIC split in two filters and MCIC configured in $Sinc^1$ filter
010: CIC split in two filters and MCIC configured in $Sinc^2$ filter
011: CIC split in two filters and MCIC configured in $Sinc^3$ filter
100: CIC configured in single $Sinc^4$ filter
others: CIC configured in single $Sinc^5$ filter

Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details).

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **DATSRC[1:0]**: Source data for the digital filter

This field is set and cleared by software.

0x: Stream coming from the BSMX selected

10: Stream coming from the ADCITF1 selected

11: Stream coming from the ADCITF2 selected

Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details).

35.8.7 MDF reshape filter configuration register x (MDF_DFLTxRSFR)

Address offset: 0x090 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register is used to control the reshape and HPF filters.

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

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:8 **HPFC[1:0]**: High-pass filter cut-off frequency

This field is set and cleared by software. It is used to select the cut-off frequency of the high-pass filter. F_{PCM} represents the sampling frequency at HPF input.

00: Cut-off frequency = 0.000625 x F_{PCM}

01: Cut-off frequency = 0.00125 x F_{PCM}

10: Cut-off frequency = 0.00250 x F_{PCM}

11: Cut-off frequency = 0.00950 x F_{PCM}

Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details).

Bit 7 **HPFBYP**: High-pass filter bypass

This bit is set and cleared by software. It is used to bypass the high-pass filter.

0: HPF not bypassed (default value)

1: HPF bypassed

Note: This bit can be write-protected (refer to Section 35.4.15: Register protection for details).

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **RSFLTD**: Reshaper filter decimation ratio

This bit is set and cleared by software. It is used to select the decimation ratio for the reshape filter.

0: Decimation ratio is 4 (default value).

1: Decimation ratio is 1.

Note: This bit can be write-protected (refer to Section 35.4.15: Register protection for details).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **RSFLTBYP**: Reshaper filter bypass

This bit is set and cleared by software. It is used to bypass the reshape filter and its decimation block.

0: Reshape filter not bypassed (default value)

1: Reshape filter bypassed

Note: This bit can be write-protected (refer to Section 35.4.15: Register protection for details).

35.8.8 MDF integrator configuration register x (MDF_DFLTxINTR)

Address offset: 0x094 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register is used to the integrator (INT) settings.

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

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:4 **INTVAL[6:0]**: Integration value selection

This field is set and cleared by software. It is used to select the integration value.

0: The integration value is 1, meaning bypass mode (default after reset).

1: The integration value is 2.

2: The integration value is 3.

...

127: The integration value is 128.

Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details).

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **INTDIV[1:0]**: Integrator output division

This field is set and cleared by software. It is used to rescale the signal at the integrator output in order keep the data width lower than 24 bits.

00: The integrator data outputs are divided by 128 (default value).

01: The integrator data outputs are divided by 32.

10: The integrator data outputs are divided by 4.

11: The integrator data outputs are not divided.

Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details).

35.8.9 MDF out-of limit detector control register x (MDF_OLDxCR)

Address offset: 0x098 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register is used to configure the OLDx.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OLDACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACICD[4:0]					Res.
r										rw	rw	rw	rw	rw	

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

Bit 31 **OLDACTIVE**: OLDx active flag

This bit is set and cleared by hardware. It is used to check if the OLDx is effectively enabled (active) or not. The protected fields and registers of this function can only be updated when the OLDACTIVE is set to 0 (refer to *Section 35.4.15: Register protection* for details).
The delay between a transition on OLDEN and a transition on OLDACTIVE is two periods of AHB clock and two periods of mdf_proc_ck.
0: OLDx not active and can be configured if needed
1: OLDx active and protected fields cannot be configured

Bits 30:22 Reserved, must be kept at reset value.

Bits 21:17 **ACICD[4:0]**: OLDx CIC decimation ratio selection

This filed is set and cleared by software. It is used to select the decimation ratio of the ACIC. This field is only taken into account by the MDF when CICMOD[2:0] = 0xx. The decimation ratio is given by (ACICD+1).
0: Decimation ratio is 1.
1: Decimation ratio is 2.
2: Decimation ratio is 3.
3: Decimation ratio is 4.
...
31: Decimation ratio is 32.

Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details).

Bits 16:14 Reserved, must be kept at reset value.

Bits 13:12 **ACICN[1:0]**: OLDx CIC order selection

This field is set and cleared by software. It is used to select the ACIC type and order. This field is only taken into account by the MDF when CICMOD[2:0] = 0xx.
00: FastSinc filter type
01: $Sinc^1$ filter type
10: $Sinc^2$ filter type
11: $Sinc^3$ filter type

Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details).

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:4 **BKOLD[3:0]**: Break signal assignment for out-of limit detector
This field is set and cleared by software.
BKOLD[i] = 0: Break signal (mdf_break[i]) not assigned to threshold event
BKOLD[i] = 1: Break signal (mdf_break[i]) assigned to threshold event
Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details).

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **THINB**: Threshold In band
This bit is set and cleared by software.
0: The OLDx generates an event if the signal is lower than OLDTHL or higher than OLDTHH (default value).
1: The OLDx generates an event if the signal is lower than OLDTHH and higher than OLDTHL
Note: This bit can be write-protected (refer to Section 35.4.15: Register protection for details).

Bit 0 **OLDEN**: OLDx enable
This bit is set and cleared by software.
0: OLDx disabled (default value)
1: OLDx enabled, including the ACIC filter working in continuous mode

35.8.10 MDF OLDx low threshold register x (MDF_OLDxTHLR)

Address offset: 0x09C + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register is used for the adjustment of the out-off-limit low threshold.

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

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **OLDTHL[25:0]**: OLD low threshold value
This field is set and cleared by software. OLDTHL represents a 26-bit signed value. The real threshold compared to the signal provided by the filter is OLDTHL.
Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details).

35.8.11 MDF OLDx high threshold register x (MDF_OLDxTHHR)

Address offset: 0x0A0 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register is used for the adjustment of the OLDx high threshold.

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

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **OLDTHH[25:0]**: OLDx high threshold value

This field is set and cleared by software. OLDTHH represents a 26-bit signed value. The real threshold compared to the signal provided by the filter is OLDTHH.

Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details).

35.8.12 MDF delay control register x (MDF_DLYxCR)

Address offset: 0x0A4 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register is used for the adjustment stream delays.

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

Bit 31 **SKPBF**: Skip busy flag

This bit is set and cleared by hardware. It is used to control if the delay sequence is completed.

0: MDF ready to accept a new value into SKPDLY[6:0]

1: Last valid SKPDLY[6:0] still under processing

Bits 30:7 Reserved, must be kept at reset value.

Bits 6:0 **SKPDLY[6:0]**: Delay to apply to a bitstream

This field is set and cleared by software. It defines the number of input samples that are skipped. Skipping is applied immediately after writing to this field, if SKPBF = 0 and the corresponding DFLTEN = 1. If SKPBF = 1, the value written into the register is ignored by the delay state machine.

0: No input sample skipped

1: 1 input sample skipped

...

127: 127 input samples skipped

35.8.13 MDF short circuit detector control register x (MDF_SCDxCR)

Address offset: 0x0A8 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register is used for the adjustment stream delays.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCDACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCDT[7:4]			
r												rw	rw	rw	rw

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

Bit 31 **SCDACTIVE**: SCDx active flag

This bit is set and cleared by hardware. It is used to check if the SCDx is effectively enabled (active) or not. The protected fields of this function can only be updated when the SCDACTIVE is set to 0 (refer to *Section 35.4.15: Register protection* for details).

The delay between a transition on SCDEN and a transition on SCDACTIVE is two periods of AHB clock and two periods of mdf_proc_ck.

0: SCDx not active and can be configured if needed

1: SCDx active and protected fields cannot be configured

Bits 30:20 Reserved, must be kept at reset value.

Bits 19:12 **SCDT[7:0]**: SCDx threshold

This field is set and cleared by software. These bits are written by software to define the threshold counter for SCDx. If this value is reached, a short-circuit detector event occurs on a given input stream.

0: 2 consecutive 1's or 0's generate an event.

1: 2 consecutive 1's or 0's generate an event.

2: 3 consecutive 1's or 0's generate an event.

...

255: 256 consecutive 1's or 0's generate an event.

Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details).

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:4 **BKSCD[3:0]**: Break signal assignment for short circuit detector

This filed is set and cleared by software.

BKSCD[i] = 0: Break signal (mdf_break[i]) not assigned to this SCD event

BKSCD[i] = 1: Break signal (mdf_break[i]) assigned to this SCD event

Note: This field can be write-protected (refer to Section 35.4.15: Register protection for details).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **SCDEN**: SCDx enable

This bit is set and cleared by software.

0: SCDx disabled

1: SSCDx enabled,

35.8.14 MDF DFLT0 interrupt enable register 0 (MDF_DFLT0IER)

Address offset: 0x0AC

Reset value: 0x0000 0000

This register is used for allowing or not the events to generate an interrupt.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RFOVRIE	CKABIE	SATIE	SCDIE	SSOVRIE	Res.	Res.	OLDIE	Res.	SSDRIE	DOVRIE	FTHIE
				rw	rw	rw	rw	rw			rw		rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **RFOVRIE**: Reshape filter overrun interrupt enable
This bit is set and cleared by software.
0: Reshape filter overrun interrupt disabled
1: Reshape filter overrun interrupt enabled

Bit 10 **CKABIE**: Clock absence detection interrupt enable
This bit is set and cleared by software.
0: Clock absence interrupt disabled
1: Clock absence interrupt enabled

Bit 9 **SATIE**: Saturation detection interrupt enable
This bit is set and cleared by software.
0: Saturation interrupt disabled
1: Saturation interrupt enabled

Bit 8 **SCDIE**: SCD0 interrupt enable
This bit is set and cleared by software.
0: SCD0 interrupt disabled
1: SCD0 interrupt enabled

Bit 7 **SSOVRIE**: Snapshot overrun interrupt enable
This bit is set and cleared by software.
0: Snapshot overrun interrupt disabled
1: Snapshot overrun interrupt enabled

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **OLDIE**: OLD0 interrupt enable
This bit is set and cleared by software.
0: OLD0 event interrupt disabled
1: OLD0 event interrupt enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SSDRIE**: Snapshot data ready interrupt enable
This bit is set and cleared by software.
0: Snapshot data ready interrupt disabled
1: Snapshot data ready interrupt enabled

Bit 1 **DOVRIE**: Data overflow interrupt enable
This bit is set and cleared by software.
0: Data overflow interrupt disabled
1: Data overflow interrupt enabled

Bit 0 **FTHIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: RXFIFO threshold interrupt disabled
1: RXFIFO threshold interrupt enabled

35.8.15 MDF DFLTx interrupt enable register x (MDF_DFLTxIER)

Address offset: 0x12C + 0x80 * (x - 1), (x = 1 to 5)

Reset value: 0x0000 0000

This register is used for allowing or not, the events to generate an interrupt.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RFOVRIE	CKABIE	SATIE	SCDIE	SSOVRIE	Res.	Res.	OLDIE	Res.	SSDRIE	DOVRIE	FTHIE
				rw	rw	rw	rw	rw			rw		rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **RFOVRIE**: Reshape filter overrun interrupt enable
This bit is set and cleared by software.
0: Reshape filter overrun interrupt disabled
1: Reshape filter overrun interrupt enabled

Bit 10 **CKABIE**: Clock absence detection interrupt enable
This bit is set and cleared by software.
0: Clock absence interrupt disabled
1: Clock absence interrupt enabled

Bit 9 **SATIE**: Saturation detection interrupt enable
This bit is set and cleared by software.
0: Saturation interrupt disabled
1: Saturation interrupt enabled

Bit 8 **SCDIE**: SCDx interrupt enable
This bit is set and cleared by software.
0: SCDx interrupt disabled
1: SCDx interrupt enabled

Bit 7 **SSOVRIE**: Snapshot overrun interrupt enable
This bit is set and cleared by software.
0: Snapshot overrun interrupt disabled
1: Snapshot overrun interrupt enabled

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **OLDIE**: OLDx interrupt enable
This bit is set and cleared by software.
0: OLDx event interrupt disabled
1: OLDx event interrupt enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SSDRIE**: Snapshot data ready interrupt enable
This bit is set and cleared by software.
0: Snapshot data ready interrupt disabled
1: Snapshot data ready interrupt enabled

Bit 1 **DOVRIE**: Data overflow interrupt enable
This bit is set and cleared by software.
0: Data overflow interrupt disabled
1: Data overflow interrupt enabled

Bit 0 **FTHIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: RXFIFO threshold interrupt disabled
1: RXFIFO threshold interrupt enabled

35.8.16 MDF DFLT0 interrupt status register 0 (MDF_DFLT0ISR)

Address offset: 0x0B0

Reset value: 0x0000 0000

This register contains the status flags for each digital filter path.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RFOVRF	CKABF	SATF	SCDF	SSOVRF	THHF	THLF	OLDF	RXNEF	SSDRF	DOVRF	FTHF
				rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	r	r	rc_w1	r	rc_w1	rc_w1	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **RFOVRF**: Reshape filter overrun detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no reshape filter overrun is detected. Write 0 has no effect.
1: Read 1 means that reshape filter overrun is detected. Write 1 clears this flag.

Bit 10 **CKABF**: Clock absence detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no clock absence is detected. Write 0 has no effect.
1: Read 1 means that a clock absence is detected. Write 1 clears this flag.

Bit 9 **SATF**: Saturation detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no saturation is detected. Write 0 has no effect.
1: Read 1 means that a saturation is detected. Write 1 clears this flag.

Bit 8 **SCDF**: Short-circuit detector flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no SCD0 event is detected. Write 0 has no effect.
1: Read 1 means that a SCD0 event is detected. Write 1 clears this flag.

Bit 7 **SSOVRF**: Snapshot overrun flag

This bit is set by hardware and cleared by software by writing this bit to 1.

0: Read 0 means that no snapshot overrun event is detected. Write 0 has no effect.

1: Read 1 means that a snapshot overrun event is detected. Write 1 clears this flag.

Bit 6 **THHF**: High-threshold status flag

This bit is set by hardware and cleared by software by writing this bit to 1. It indicates the status of the high-threshold comparator when the last OLD0 event occurred. This bit gives additional information on the conditions triggering the last OLD0 event. It can be cleared by writing OLDF flag to 1.

0: The signal was lower than OLDTHH when the last OLD0 event occurred.

1: The signal was higher than OLDTHH when the last OLD0 event occurred.

Bit 5 **THLF**: Low-threshold status flag

This bit is set by hardware and cleared by software by writing this bit to 1. It indicates the status of the low-threshold comparator when the last OLD0 event occurred. This bit gives additional information on the conditions triggering the last OLD0 event. It can be cleared by writing OLDF flag to 1.

0: The signal was higher than OLDTHL when the last OLD0 event occurred.

1: The signal was lower than OLDTHL when the last OLD0 event occurred.

Bit 4 **OLDF**: OLD0 flag

This bit is set by hardware and cleared by software by writing this bit to 1.

0: Read 0 means that no OLD0 event is detected. Write 0 has no effect.

1: Read 1 means that an OLD0 event is detected Write 1 clears THHF, THLF and OLDF.

Bit 3 **RXNEF**: RXFIFO not-empty flag

this bit is set and cleared by hardware according to the RXFIFO level.

0: RXFIFO empty

1: RXFIFO not empty

Bit 2 **SSDRF**: Snapshot data ready flag

This bit is set by hardware and cleared by software by writing this bit to 1.

0: Read 0 means that no data is available. Write 0 has no effect.

1: Read 1 means that a new data is available. Write 1 clears this flag.

Bit 1 **DOVRF**: Data overflow flag

This bit is set by hardware and cleared by software by writing this bit to 1.

0: Read 0 means that no overflow is detected. Write 0 has no effect.

1: Read 1 means that an overflow is detected; Write 1 clears this flag.

Bit 0 **FTHF**: RXFIFO threshold flag

This bit is set by hardware and cleared by hardware when the RXFIFO level is lower than the threshold.

0: RXFIFO threshold not reached

1: RXFIFO threshold reached

35.8.17 MDF DFLTx interrupt status register x (MDF_DFLTxISR)

Address offset: 0x130 + 0x80 * (x - 1), (x = 1 to 5)

Reset value: 0x0000 0000

This register contains the status flags for each digital filter path.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RFOVRF	CKABF	SATF	SCDF	SSOVRF	THHF	THLF	OLDF	RXNEF	SSDRF	DOVRF	FTHF
				rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	r	r	rc_w1	r	rc_w1	rc_w1	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **RFOVRF**: Reshape filter overrun detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no reshape filter overrun is detected. Write 0 has no effect.
1: Read 1 means that reshape filter overrun is detected. Write 1 clears this flag.

Bit 10 **CKABF**: Clock absence detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no clock absence is detected. Write 0 has no effect.
1: Read 1 means that a clock absence is detected. Write 1 clears this flag.

Bit 9 **SATF**: Saturation detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no saturation is detected. Write 0 has no effect.
1: Read 1 means that a saturation is detected. Write 1 clears this flag.

Bit 8 **SCDF**: Short-circuit detector flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no SCD event is detected. Write 0 has no effect.
1: Read 1 means that a SCD event is detected. Write 1 clears this flag.

Bit 7 **SSOVRF**: Snapshot overrun flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no snapshot overrun event is detected. Write 0 has no effect.
1: Read 1 means that a snapshot overrun event is detected. Write 1 clears this flag.

Bit 6 **THHF**: High-threshold status flag
This bit is set by hardware and cleared by software by writing this bit to 1. It indicates the status of the high-threshold comparator when the last OLDx event occurred. This bit gives additional information on the conditions triggering the last OLDx event. It can be cleared by writing OLDF flag to 1.
0: The signal was lower than OLDTHH when the last OLDx event occurred.
1: The signal was higher than OLDTHH when the last OLDx event occurred.

Bit 5 **THLF**: Low-threshold status flag

This bit is set by hardware and cleared by software by writing this bit to 1. It indicates the status of the low-threshold comparator when the last OLDx event occurred. This bit gives additional information on the conditions triggering the last OLDx event. It can be cleared by writing OLDF flag to 1.

0: The signal was higher than OLDTHL when the last OLDx event occurred.

1: The signal was lower than OLDTHL when the last OLDx event occurred.

Bit 4 **OLDF**: OLDx flag

This bit is set by hardware and cleared by software by writing this bit to 1.

0: Read 0 means that no OLDx event is detected. Write 0 has no effect.

1: Read 1 means that an OLDx event is detected Write 1 clears THHF, THLF and OLDF.

Bit 3 **RXNEF**: RXFIFO not-empty flag

this bit is set and cleared by hardware according to the RXFIFO level.

0: RXFIFO empty

1: RXFIFO not empty

Bit 2 **SSDRF**: Snapshot data ready flag

This bit is set by hardware and cleared by software by writing this bit to 1.

0: Read 0 means that no data is available. Write 0 has no effect.

1: Read 1 means that a new data is available. Write 1 clears this flag.

Bit 1 **DOVRF**: Data overflow flag

This bit is set by hardware and cleared by software by writing this bit to 1.

0: Read 0 means that no overflow is detected. Write 0 has no effect.

1: Read 1 means that an overflow is detected; Write 1 clears this flag.

Bit 0 **FTHF**: RXFIFO threshold flag

This bit is set by hardware and cleared by hardware when the RXFIFO level is lower than the threshold.

0: RXFIFO threshold not reached

1: RXFIFO threshold reached

35.8.18 MDF offset error compensation control register x (MDF_OECxCR)

Address offset: 0x0B4 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register contains the offset compensation value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[25:16]									
						rw	rw	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[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **OFFSET[25:0]**: Offset error compensation
This field is set and cleared by software.
If the application attempts to write a new offset value while the previous one is not yet applied, this new offset value is ignored. Reading back this] field informs the application on the current offset value.
This field represents the value to be subtracted to the signal before going to the SCALE.

35.8.19 MDF snapshot data register x (MDF_SNPSxDR)

Address offset: 0x0EC + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register is used to read the data processed by each digital filter in snapshot mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SDR[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
EXTSDR[6:0]							MCICDC[8:0]								
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **SDR[15:0]**: Contains the 16 MSB of the last valid data processed by the digital filter.

Bits 15:9 **EXTSDR[6:0]**: Extended data size
If SNPSFMT = 0, this field contains the bits 7 to 1 of the last valid data processed by the digital filter.
If SNPSFMT = 1, this field contains the INT accumulator counter value when the last trigger event occurs (INT_CNT).

Bits 8:0 **MCICDC[8:0]**: Contains the MCIC decimation counter value when the last trigger event occurs (MCIC_CNT)

35.8.20 MDF digital filter data register x (MDF_DFLTxDR)

Address offset: 0x0F0 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register is used to read the data processed by each digital filter.

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

Bits 31:8 **DR[23:0]**: Data processed by digital filter

Bits 7:0 Reserved, must be kept at reset value.

35.8.21 MDF register map

Table 310. MDF register map and reset values

Offset	Register name and 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
0x000	MDF_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.	ILVNB[3:0]				Res.	Res.	Res.	TRGO
	Reset value																									0	0	0	0				0
0x004	MDF_CKGCR	CKGACTIVE	PROCDIV[6:0]							Res.	Res.	Res.	Res.	CCKDIV[3:0]				TRGSRC[3:0]				Res.	Res.	Res.	TRGSENS	Res.	CCK1DIR	CCK0DIR	CKGMOD	Res.	CCK1EN	CCK0EN	CKGDEN
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0				0		0	0	0		0	0	0
0x008 - 0x07C	Reserved	Reserved																															
0x080 + 0x80 * x (x = 0 to 5)	MDF_SITFxCR	SITFACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STH[4:0]				Res.	Res.	SITFMOD[1:0]		Res.	SCKSRC[1:0]		SITFEN
	Reset value	0																			1	1	1	1	1			0	0		0	0	0
0x084 + 0x80 * x (x = 0 to 5)	MDF_ BSMXxCR	BSMXACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BSSEL[4:0]				
	Reset value	0																											0	0	0	0	0
0x088 + 0x80 * x (x = 0 to 5)	MDF_DFLTxCR	DFLTACTIVE	DFLTRUN	Res.	Res.	NBDIS[7:0]								Res.	Res.	Res.	SNPSFMT	TRGSRC[3:0]				Res.	Res.	Res.	TRGSENS	Res.	ACQMOD[2:0]			Res.	FTH	DMAEN	DFLTEN
	Reset value	0	0			0	0	0	0	0	0	0	0				0	0	0	0	0				0		0	0	0		0	0	0
0x08C + 0x80 * x (x = 0 to 5)	MDF_ DFLTxCICR	Res.	Res.	Res.	Res.	Res.	Res.	SCALE[5:0]						Res.	Res.	Res.	MCICD[8:0]									Res.	CICMOD[2:0]			Res.	Res.	DATSRC[1:0]	
	Reset value							0	0	0	0	0	0				0	0	0	0	0	0	0	0	0		0	0	0			0	0
0x090 + 0x80 * x (x = 0 to 5)	MDF_ DFLTxRSFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HPFC[1:0]		HPFBYP	Res.	Res.	RSFLTD	Res.	Res.	Res.	RSFLTBYP
	Reset value																							0	0	0			0				0

Table 310. MDF register map and reset values (continued)

Offset	Register name and 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
0x094 + 0x80 * x (x = 0 to 5)	MDF_ DFLTxINTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTVAL[6:0]							Res.	Res.	INTDIV[1:0]	
	Reset value																						0	0	0	0	0	0	0			0	0
0x098 + 0x80 * x (x = 0 to 5)	MDF_OLDxCR	OLDACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACICD[4:0]					Res.	Res.	Res.	ACICN[1:0]		Res.	Res.	Res.	Res.	BKOLD[3:0]				Res.	Res.	THINB	OLDEN
	Reset value	0										0	0	0	0	0				0	0					0	0	0	0			0	0
0x09C + 0x80 * x (x = 0 to 5)	MDF_ OLDxTHLR	Res.	Res.	Res.	Res.	Res.	Res.	OLDTHL[25:0]																									
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A0 + 0x80 * x (x = 0 to 5)	MDF_ OLDxTHHR	Res.	Res.	Res.	Res.	Res.	Res.	OLDTHH[25:0]																									
	Reset value							0	0	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 + 0x80 * x (x = 0 to 5)	MDF_DLYxCR	SKPBF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SKPDLY[6:0]						
	Reset value	0																									0	0	0	0	0	0	0
0x0A8 + 0x80 * x (x = 0 to 5)	MDF_SCDxCR	SCDACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCDT[7:0]								Res.	Res.	Res.	Res.	BKSCD[3:0]				Res.	Res.	Res.	SCDEN
	Reset value	0												0	0	0	0	0	0	0	0					0	0	0	0				0
0x0AC	MDF_ DFLT0IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFOVRIE	CKABIE	SATIE	SCDIE	SSOVRIE	Res.	Res.	OLDIE	Res.	SSDRIE	DOVRIE	FTHIE
	Reset value																					0	0	0	0	0			0		0	0	0
0x0B0	MDF_ DFLT0ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFOVRF	CKABF	SATF	SCDF	SSOVRF	THHF	THLF	OLDF	RXNEF	SSDRF	DOVRF	FTHF
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x0B4 + 0x80 * x (x = 0 to 5)	MDF_OECxCR	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[25:0]																									
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B8-0x0E8	Reserved	Reserved																															
0x0EC + 0x80 * x (x = 0 to 5)	MDF_ SNPSxDR	SDR[15:0]																EXTSDR[6:0]								MCICDC[8:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 310. MDF register map and reset values (continued)

Offset	Register name and 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
0x0F0 + 0x80 * x (x = 0 to 5)	MDF_DFLTxDR	DR[23:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x12C + 0x80*(x-1) (x=1 to 5)	MDF_ DFLTxIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFOVRIE	CKABIE	SATIE	SCDIE	SSOVRIE	Res.	Res.	OLDIE	Res.	SSDRIE	DOVRIE	FTHIE
	Reset value																					0	0	0	0	0			0		0	0	0
0x130 + 0x80*(x-1) (x=1 to 5)	MDF_ DFLTxISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFOVRF	CKABF	SATF	SCDF	SSOVRF	THHF	THLF	OLDF	RXNEF	SSDRF	DOVRF	FTHF
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

36 Audio digital filter (ADF)

36.1 Introduction

The audio digital filter (ADF) is a high-performance module dedicated to the connection of external sigma-delta (ΣΔ) modulators. It is mainly targeted for the following applications:

- audio capture signals
- metering

The ADF features one digital serial interface (SITF0) and one digital filter (DFLT0) with flexible digital processing options in order to offer up to 24-bit final resolution.

The ADF serial interface supports several standards allowing the connection of various ΣΔ modulator sensors:

- SPI interface
- Manchester coded 1-wire interface
- PDM interface

The ADF converts an input data stream into clean decimated digital data words. This conversion is done thanks to low-pass digital filters and decimation blocks. In addition, it is possible to insert a high-pass filter.

The conversion speed and resolution are adjustable according to configurable parameters for digital processing: filter bypass, filter order, decimation ratio. The maximum output data resolution is up to 24 bits. There are two conversion modes: single conversion and continuous modes. The data can be automatically stored in a system RAM buffer through DMA, thus reducing the software overhead.

A sound activity detector (SAD) is available for the detection of sounds or voice signals. The SAD is connected at the output of the DFLT0. Several parameters can be programmed in order to adjust properly the SAD to the sound environment. The SAD strongly reduces the power consumption by preventing the storage of samples into the system memory, as long as the observed signal does not match the programmed criteria.

The digital processing is performed using only the kernel clock. The ADF requests the bus interface clock (AHB clock) only when data must be transfered or when a specific event requests the attention of the system processor.

36.2 ADF main features

- AHB Interface
- 1 serial digital input:
  - configurable SPI interface to connect various digital sensors
  - configurable Manchester coded interface support
  - compatible with PDM interface to support digital microphones
- 2 common clocks input/output for ΣΔ modulators
- 1 flexible digital filter path including:
  - A MCIC filter configurable in $Sinc^4$ or $Sinc^5$ filter with an adjustable decimation ratio
  - A reshape filter to improve the out-off band rejection and in-band ripple
  - A high-pass filter to cancel the DC offset
  - Gain control
  - Saturation blocks
- Clock absence detector
- Sound activity detector
- 24-bit signed output data resolution
- Continuous or single conversion
- Possibility to delay the selected bitstream
- One trigger input
- Autonomous functionality in Stop modes
- DMA can be used to read the conversion data
- Interrupts services

36.3 ADF implementation

The devices embed one MDF instance and one ADF instance, both being digital filters with common features

Table 311. ADF/MDF features (1)

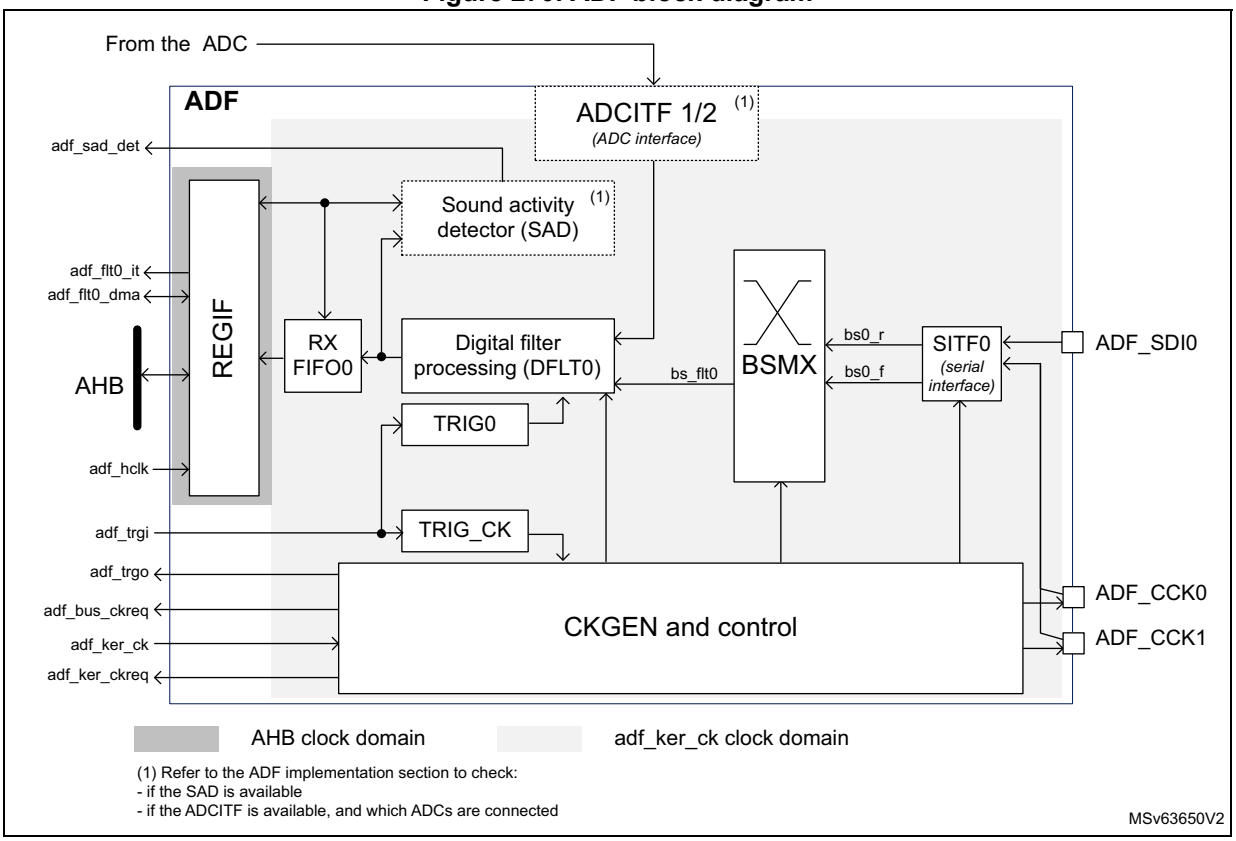
Mode or feature	ADF1	MDF1
Number of filters (DFLTx) and serial interfaces (SITFx)	1	6
MDF_CKIy/ADF_CKI0 connected to pins	-	X
Sound activity detection (SAD)	X	-
RXFIFO depth (number of 24-bit words)	4	4
ADC connected to ADCITF1	-	ADC1
ADC connected to ADCITF2	-	-
Motor dedicated features (SCD, OLD, OEC, INT, snapshot, break)	-	X
Main path with CIC4, CIC5	X	X
Main path with CIC1,2, 3 or FastSinc	-	X
RSFLT, HPF, SAT, SCALE, DLY, Discard functions	X	X
Autonomous in Stop modes	X(2)	X(3)

1. 'X' = supported, '-' = not supported.
2. Only Stop 0, Stop 1 and Stop 2 modes.
3. Only Stop 0 and Stop 1 modes.

36.4 ADF functional description

36.4.1 ADF block diagram

Figure 270. ADF block diagram



36.4.2 ADF pins and internal signals

Table 312. ADF external pins

Name	Signal type	Remarks
ADF_SDI0	Input	Data signal from external sensors.
ADF_CCKy (y = 0,1)	Input/output	Clock outputs for external sensor, or common clock input from external sensors

Table 313. ADF internal signals

Name	Signal type	Remarks
adf_trgi	Input	Trigger inputs to control the acquisition (see the next table for details)
adf_trgo	Output	Trigger output for synchronizing with other MDF instances
adf_flt0_dma	Input/output	DMA request/acknowledge signals for the ADF processing chain.
adf_flt0_it	Output	Global interrupt signals

Table 313. ADF internal signals (continued)

Name	Signal type	Remarks
adf_bus_ckreq	Output	Bus interface clock request output
adf_ker_ckreq	Output	Kernel clock request output
adf_ker_ck	Input	Kernel clock input
adf_hclk	Input	AHB bus interface clock input
adf_sad_det	Output	SAD sound detection: 1 means that detecting sound
adf_adcitf1_dat[15:0]	Input	ADCITF1 data input
adf_adcitf2_dat[15:0]	Input	ADCITF2 data input

Table 314. ADF trigger connections

Trigger name	Direction	Trigger source/destination
adf_trgi	Input	From exti15
adf_trgo	Output	To mdf_trgi13

36.4.3 Serial input interface (SITF)

The SITF0 input interface allows the connection of the external sensor to the digital filter via the bitstream matrix (BSMX). The SITF0 can be configured in the following modes:

- LF_MASTER SPI mode (low-frequency)
- normal SPI mode
- Manchester mode

The data from the serial interface is routed to the filter in order to perform the PDM to PCM conversion and the sound activity detection.

The serial interface is enabled by setting the SITFEN bit to 1. Once the interface is enabled, it receives serial data from the external ΣΔ modulator.

Note: Before enabling the *serial interface, the user must insure that the adf_proc_ck is already enabled (see Section 36.4.5: Clock generator (CKGEN) for details).*

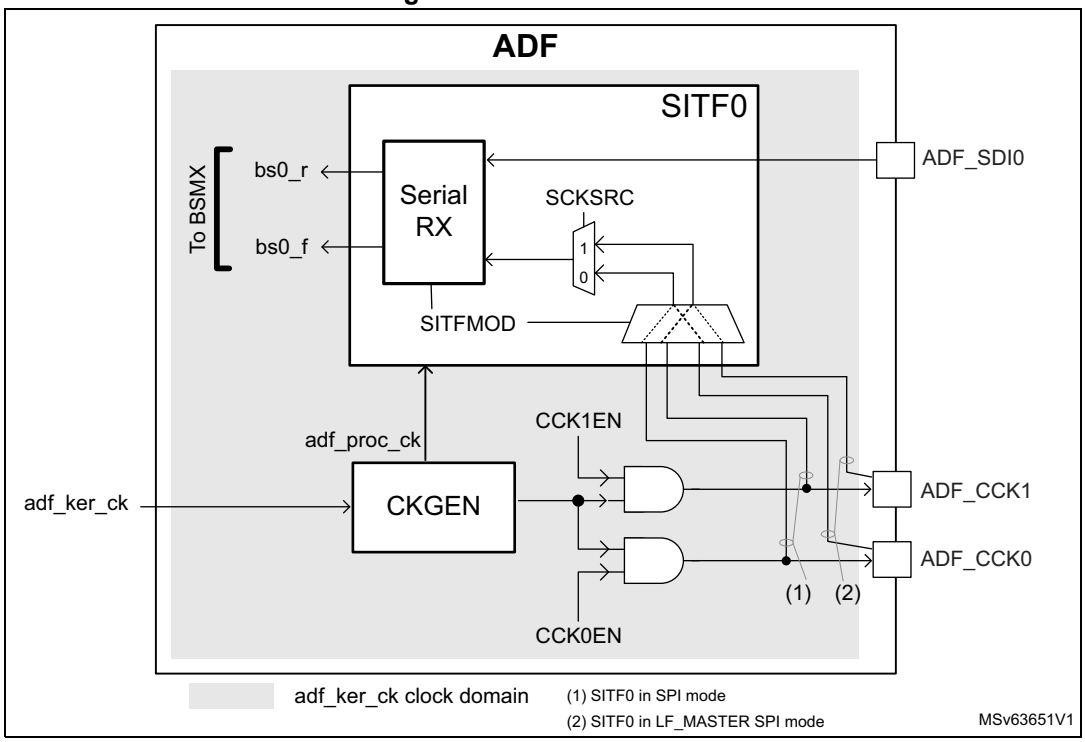
The SITF0 is controlled via the *ADF serial interface control register 0 (ADF_SITF0CR)*.

As shown in the *Figure 271*, ADF_CCK0 or ADF_CCK1 can be selected as clock source, in order to sample the incoming bitstream:

- If the serial interface is programmed in SPI mode, the selected clock source is a copy of the clock present on the ADF_CCK0 or ADF_CCK1 pin.
- If the serial interface is programmed in LF_MASTER SPI mode, the selected clock source is the clock directly provided by the CCKDIV to the ADF_CCK0 or ADF_CCK1 pin.

See *Table 315* for additional information.

Figure 271. SITF overview



LF_MASTER and normal SPI modes

The LF_MASTER SPI mode is a special mode allowing the use of an adf_proc_ck clock frequency, only two times bigger than the sensor clock. This mode is dedicated to low-power use-cases, using low-speed sensors.

In LF_MASTER SPI mode, the ADF must provide the bitstream clock to the external sensors via ADF_CCK0 and ADF_CCK1 pins. The ADF receives the bitstream data via the serial data input ADF_SDI0.

For the SITF0, the application must select the same clock than the one provided to the external sensor (ADF_CCK0 or ADF_CCK1), in order to guarantee optimal timing performances. This selection is done via SCKSRC[1:0].

The normal SPI interface is a more flexible interface than the LF_MASTER SPI, but the adf_proc_ck frequency must be at least four times higher than the sensor clock.

The application can select ADF_CCK0 or ADF_CCK1 clock for the capture of the data received via the ADF_SDI0 pin.

The ADF can generate a clock to the sensors via ADF_CCK0 or ADF_CCK1 if needed.

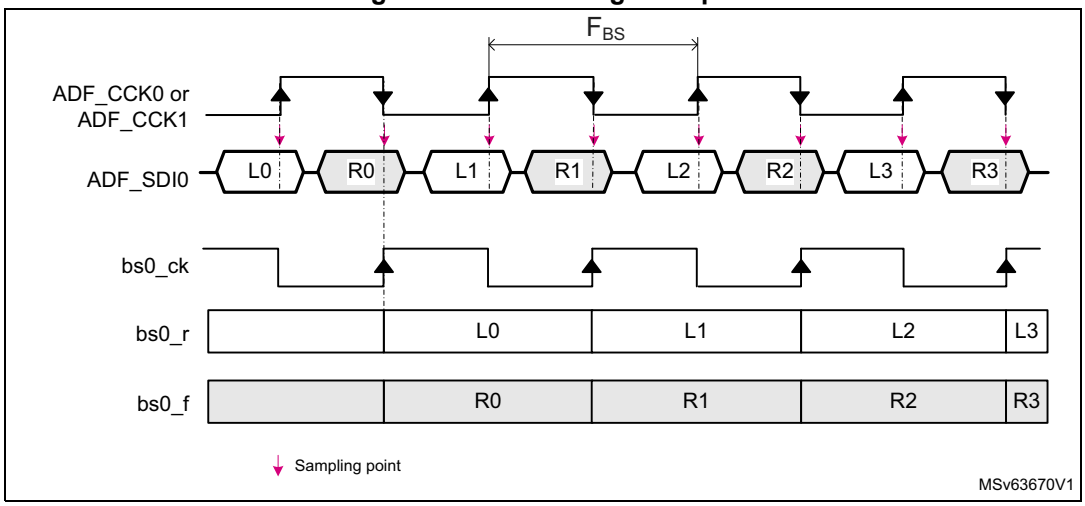
For all SPI modes, the serial data is captured using the rising and the falling edge of the selected clock. The SITF0 always provides the following bitstreams:

- bitstream received using the bitstream clock falling edge (bs0_f)
- bitstream received using the bitstream clock rising edge (bs0_r)

According to the sensors connected, one of the two bitstreams may not be available.

The application can select the wanted stream via the BSMX matrix.

Figure 272. SPI timing example



To properly synchronize/receive the data stream, the adf_proc_ck frequency must be adjusted according to the constraints listed in *Table 316*.

Clock absence detection

A no-clock-transition period may be detected when the serial interface works in normal SPI mode. This feature can be used to detect a clock failure in the SPI link.

The application can program a timeout value via the STH[4:0] field of the SITF0. If the ADF does not detect clock transitions for a duration of STH[4:0] x $T_{adf\_proc\_ck}$, then the CKABF flag is set.

An interrupt can be generated if CKABIE is set to 1. The STH[4:0] field is in the *ADF serial interface control register 0 (ADF_SITF0CR)*.

When the serial interface is enabled, the CKABF flag remains to 1 until a first clock transition is detected.

To avoid spurious clock absence detection, the following sequence must be respected:

1. Configure the serial interface in normal SPI mode and enable it.
2. Clear the CKABF flag by writing CKABF bit to 1.

   If no clock transition is detected on the serial interface, the hardware immediately sets the CKABF flag to 1.
3. Read the CKABF flag:
   - If CKABF = 1, go back to step 2.
   - If CKABF = 0, a clock has been detected. The CKABIE bit can be set to 1 if the application wants an interrupt on detection of a clock absence.

Note: The clock absence detection feature is not available in the LF_MASTER SPI mode.

Manchester mode

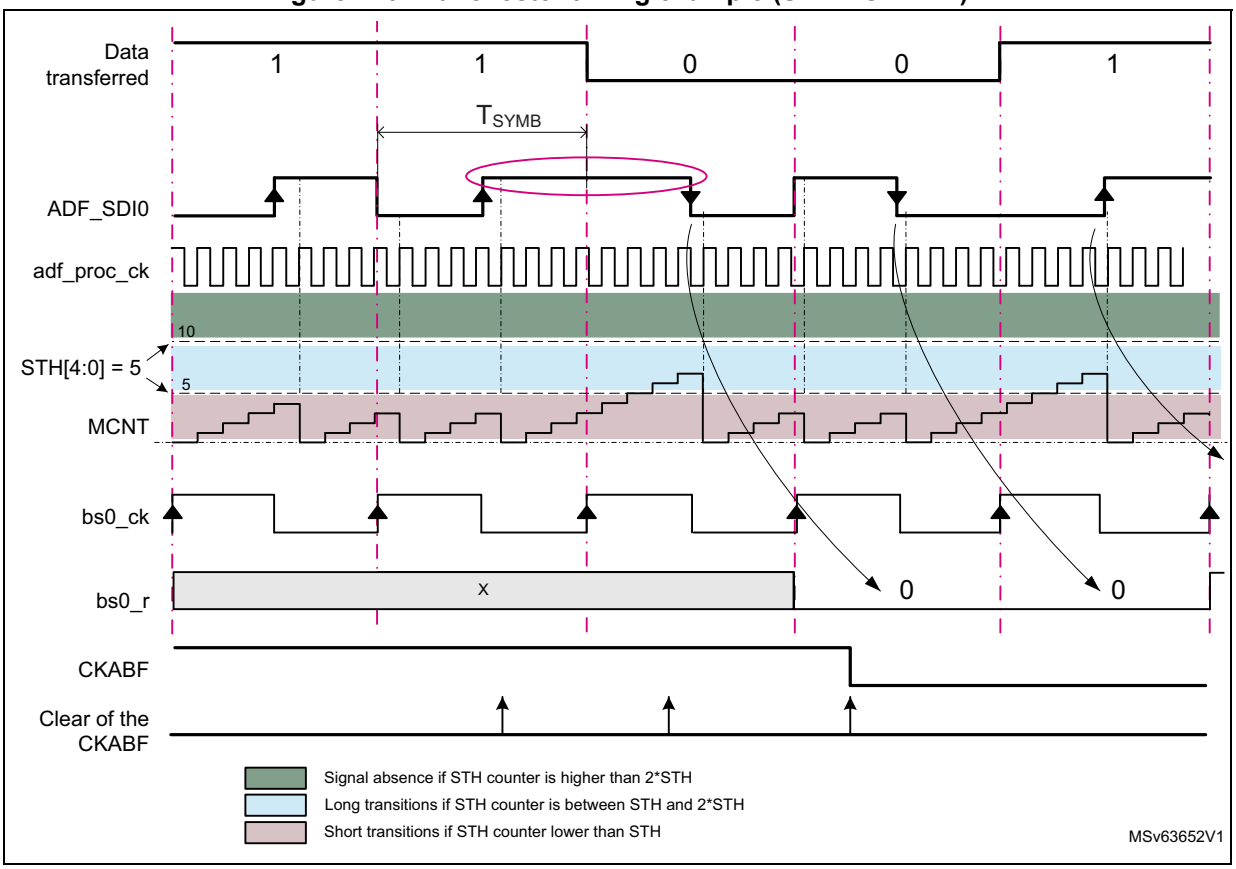
In Manchester coded format, the ADF receives data stream from the external sensor via the ADF_SDI0 pin only.

The ADF_CCK0 and ADF_CCK1 pins are not needed in this mode.

Decoded data and clock signals are recovered from serial stream after Manchester decoding. They are available on bs0_r. There are two possible settings of Manchester codings:

- signal rising edge decoded as 0 and signal falling edge decoded as 1
- signal rising edge decoded as 1 and signal falling edge decoded as 0

Figure 273. Manchester timing example (SITFMOD = 11)



To decode the incoming Manchester stream, the user must program STH[4:0] in the *ADF serial interface control register 0 (ADF_SITF0CR)*. The STH[4:0] field is used by the SITF0 to estimate the Manchester symbol length and to detect a clock absence. An internal counter (MCNT) is restarted every time a transition is detected in the ADF_SDI0 input. It is used to detect short transitions, long transitions or clock absence. A long transition indicates that the data value changed. *Figure 273* shows a case where the OVR is around height and STH[4:0] = 5.

The estimated Manchester symbol rate (T_{SYMB}) must respect the following formula:

$$(STH + 1) \times T_{adf\_proc\_ck} < T_{SYMB} < (2 \times STH \times T_{adf\_proc\_ck})$$

It is recommended to compute STH as follows:

$$STH[4:0] = \text{round}\left(\frac{(2 \times OVR) - 1}{3}\right)$$

where OVR represents the ratio between the adf_proc_ck frequency and the expected Manchester symbol frequency. OVR must be higher than five, and the adf_proc_ck clock must be adjusted according to the constraints listed in *Table 316*.

The clock absence flag CKABF is set to 1 when no transition is detected during more than 2 x STH[4:0] x $T_{adf\_proc\_ck}$, or when the SITF0 is not yet synchronized to the incoming Manchester stream. In addition, an interrupt can be generated if the bit CKABIE is set to 1.

When the serial interface is enabled, the ADF must first be synchronized to the incoming Manchester stream. The synchronization ends when a data transition from 0 to 1 or from 1 to 0 (pink circle in the *Figure 273*) is detected.

The end of the synchronization phase can be checked by following the software sequence:

1. Clear the CKABF flag in the *ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR)* by writing CKABF bit to 1. If the serial interface is not yet synchronized the hardware immediately set the CKABF flag to 1.
2. Read the CKABF flag:
   – If CKABF= 1, go back to step 1.
   – If CKABF = 0, the Manchester interface is synchronized and provides valid data.

Programming example

In the following example, the ADF kernel clock frequency ($F_{adf\_ker\_ck}$) is 100 MHz and the received Manchester stream is at about 6 MHz (F_{SYMB}):

1. Provide a valid adf_proc_ck to the SITF0.

   The adf_proc_ck frequency must be at least six times higher than the Manchester symbol frequency (means at least 36 MHz).

   PROCDIV is programmed to 1 to perform a division by two of the kernel clock. In that case, $F_{adf\_proc\_ck}$ = 50 MHz (8.33 times higher than the Manchester symbol frequency).
2. Compute STH.

   OVR is given by: OVR = $F_{adf\_proc\_ck}$ / F_{SYMB} = 50 MHz / 6 MHz = 8.33.

   Then STH[4:0] = $\text{round}\left(\frac{(2 \times 8.33) - 1}{3}\right) = 5$

   The minimum allowed frequency for the Manchester stream is then:

   1 / (2 × STH × $T_{adf\_proc\_ck}$) = 1 / (10 x 20 ns) = 5 MHz

   The maximum allowed frequency for the Manchester stream is then:

   1 / ((STH+1) × $T_{adf\_proc\_ck}$) = 1 / (6 x 20 ns) = 8.33 MHz

36.4.4 ADC slave interface (ADCITF)

The ADCs are not always connected to the ADF. Refer to *Section 36.3: ADF implementation* to check the situation for this product.

The ADF allows the connection of up to two ADCs to the filter path. For the filter, the DATSRC[1:0] field in the *ADF digital filer configuration register 0 (ADF_DFLT0CICR)* allows the application to select data from the ADCs.

Warning: The ADF does not support receiving interleaved data from one of the ADCITF input.

36.4.5 Clock generator (CKGEN)

The RCC (reset and clock controller) provides the following clocks to the ADF:

- AHB clock (adf_hclk) used for the register interface
- kernel clock (adf_ker_ck) mainly used by all other parts of the circuit via the CKGEN

Those clocks are not supposed to be phase locked, so all signals crossing those clock domains are re-synchronized.

The clock generator (CKGEN) is responsible of the generation of the processing clock, and the clock provided to the ADF_CCK0 and ADF_CCK1 pins. All those clocks are generated from the adf_ker_ck.

The processing clock (adf_proc_ck) is used to run all the signal processing and to re-sample the incoming serial or parallel stream.

To adapt the kernel clock frequency provided by the RCC, the following dividers are available:

- PROCDIV[6:0] used to adapt the kernel clock frequency to the constraints of the parallel and serial interfaces, and to the processing blocks
- CCKDIV[3:0] used to adapt the frequency of the ADF_CCK0 and ADF_CCK1 clocks

PROCDIV[6:0] and CCKDIV[3:0] must be programmed when no clock is provided to the dividers (CKGDEN = 0).

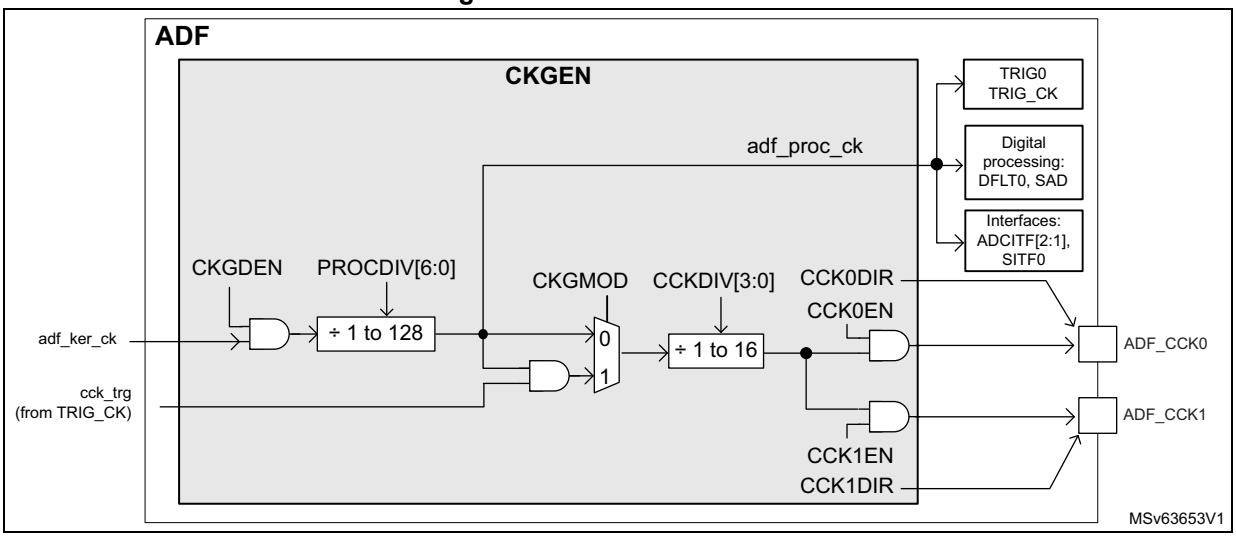
The adf_proc_ck generation is controlled by CKGDEN.

In addition, the CKGMOD bit allows the application to define the way to trigger the CCKDIV divider:

- When CKGMOD = 0, the CCKDIV divider is started as soon as CKGDEN is set to 1.
- When CKGMOD = 1, the CCKDIV divider is started when CKGDEN is set to 1 and the programmed trigger condition occurred.

All the bits and fields controlling the CKGEN are in the *ADF clock generator control register (ADF_CKGCR)*.

Figure 274. CKGEN overview



The trigger logic for CKGEN is handled by the block TRG_CK. As shown in *Figure 280*, the CCKDIV divider can be triggered on the rising or falling edge of an external trigger source. When the proper trigger condition occurs, the cck_trg signal goes to high, allowing the CCKDIV divider to start. The TRG_CK logic is reset when CKGDEN is set to 0.

This feature can be helpful to synchronize the ADF_CCKy (y = 0,1) clock of several ADF instances, or to synchronize the clock generation to a timer event.

The application can control the activation of the ADF_CCK0 or ADF_CCK1 pin thanks to CCK0EN/CCK1EN and CCK0DIR/CCK1DIR bits:

- CCKyEN is used to enable the CCKDIV, and thus generates a clock for the external sensors.
- CCKyDIR is used to control the direction of the ADF_CCKy pin (input or output)

Table 315. Control of the common clock generation(1)

CCKyEN	CCKyDIR	Description
0	0	The ADF_CCKy pin is in input. An external clock can be connected to the ADF_CCKy pin and used by the SITF0 in order to decode the serial stream
0	1	The ADF_CCKy pin is in output. No clock is generated. The ADF_CCKy pin is driven low.
1	1	The ADF_CCKy pin is in output. A clock is generated on the ADF_CCKy pin. The SITF0 can use this pin as clock source in order to decode the serial stream

1. The configuration with CCKyEN = 1 and CCyDIR = 0 is not shown must be avoided (no interest).

Note: The adf_proc_ck must be enabled (by CKGDEN = 1) before enabling other blocks (such as SITF0 or DFLT0).

CKGEN activation sequence example

- Set CKGDEN to 0.
- Wait for CKGACTIVE = 0. If CKGDEN was previously enabled, this phase can take two periods of adf_hclk, and two periods of adf_proc_ck.
- Program PROCDIV[6:0], CKGMOD, CCKDIV[3:0], TRGSRC[3:0], TRGSENS, CCK1EN and CCK0EN.
- Set CKGDEN to 1.

When needed, at any moment, CCK[1:0]EN field value can be changed without disabling the clock generator.

Clock frequency constraints

The table below shows the frequency constraints to receive and process properly the samples.

Note: The reshape filter (RSFLT) needs up to 24 cycles of adf_proc_ck clock to process one sample.

Table 316. Clock constraints with respect to the incoming stream(1)

SITF0 mode	ADF clock constraints	
	With RSFLT disabled	With RSFLT enabled
LF_MASTER SPI	F_{ADF_CCKy} max frequency limited to 5 MHz	
	$F_{adf\_proc\_ck} > 2 \times F_{ADF\_CCKy}$ and $F_{adf\_hclk} \geq F_{adf\_proc\_ck}$	$F_{adf\_proc\_ck} > 24 \times F_{ADF\_CCKy} / (MCICD+1)$ and $F_{adf\_proc\_ck} > 2 \times F_{ADF\_CCKy}$ and $F_{adf\_hclk} \geq F_{adf\_proc\_ck}$
MASTER SPI SLAVE SPI	F_{ADF_CKx} max frequency limited to 25 MHz	
	$F_{adf\_proc\_ck} > 4 \times F_{ADF\_CCKy}$ and $F_{adf\_hclk} \geq F_{adf\_proc\_ck}$	$F_{adf\_proc\_ck} > 24 \times F_{ADF\_CCKy} / (MCICD+1)$ and $F_{adf\_proc\_ck} > 4 \times F_{ADF\_CCKy}$ and $F_{adf\_hclk} \geq F_{adf\_proc\_ck}$
Manchester	F_{SYMB} max frequency limited to 20 MHz	
	$F_{adf\_proc\_ck} > 6 \times F_{SYMB}$ and $F_{adf\_hclk} \geq F_{adf\_proc\_ck}$	$F_{adf\_proc\_ck} > 24 \times F_{ADF\_CCKy} / (MCICD+1)$ and $F_{adf\_proc\_ck} > 6 \times F_{SYMB}$ and $F_{adf\_hclk} \geq F_{adf\_proc\_ck}$

1. F_{ADF_CCKy} represents the frequency of clock received via ADF_CCKy, or generated via ADF_CCKy. F_{SYMB} represents the frequency of the received symbol rate for Manchester mode.

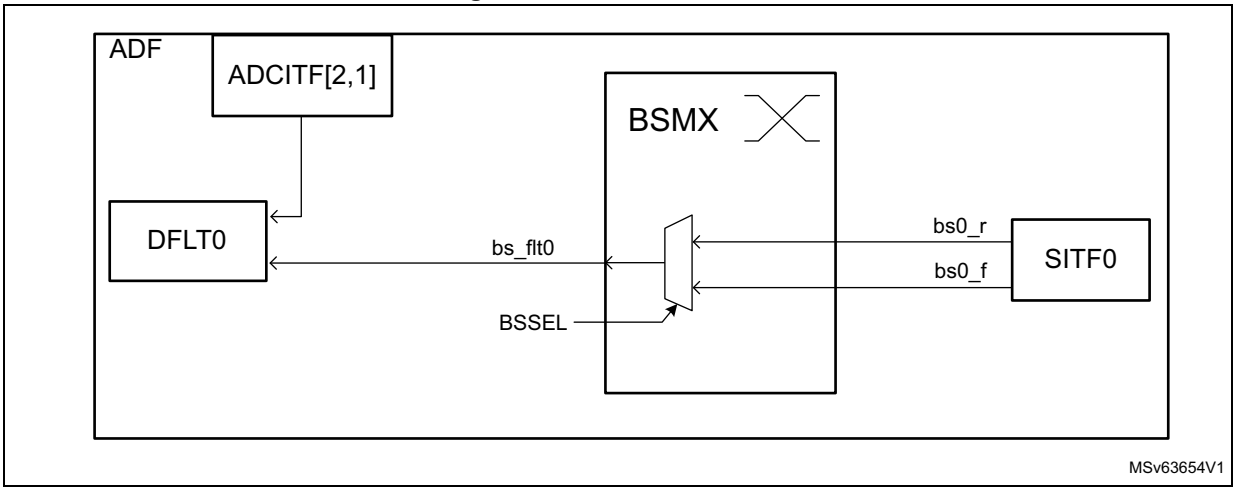
36.4.6 Bitstream matrix (BSMX)

The BSMX receives the bitstreams from the serial interface SITF0 and provides the selected stream to the digital filter DFLT0.

As shown in the *Figure 271*, the SITF0 provides two bitstreams (bs0_r and bs0_f) to the BSMX.

The application to select the wanted stream via the *ADF bitstream matrix control register 0 (ADF_BSMX0CR)*. This selection is intended to be static.

Figure 275. BSMX overview



BSMX programming sequence example

The BSSEL[4:0] field cannot be changed if the DFLT0 is enabled. The following steps are needed to change the value of BSMX:

- Set DFLTEN of DFLT0 to 0.
- Wait for BSMXACTIVE = 0.
- Program BSSEL[4:0].
- Set DFLTEN of DFLT0 to 1.

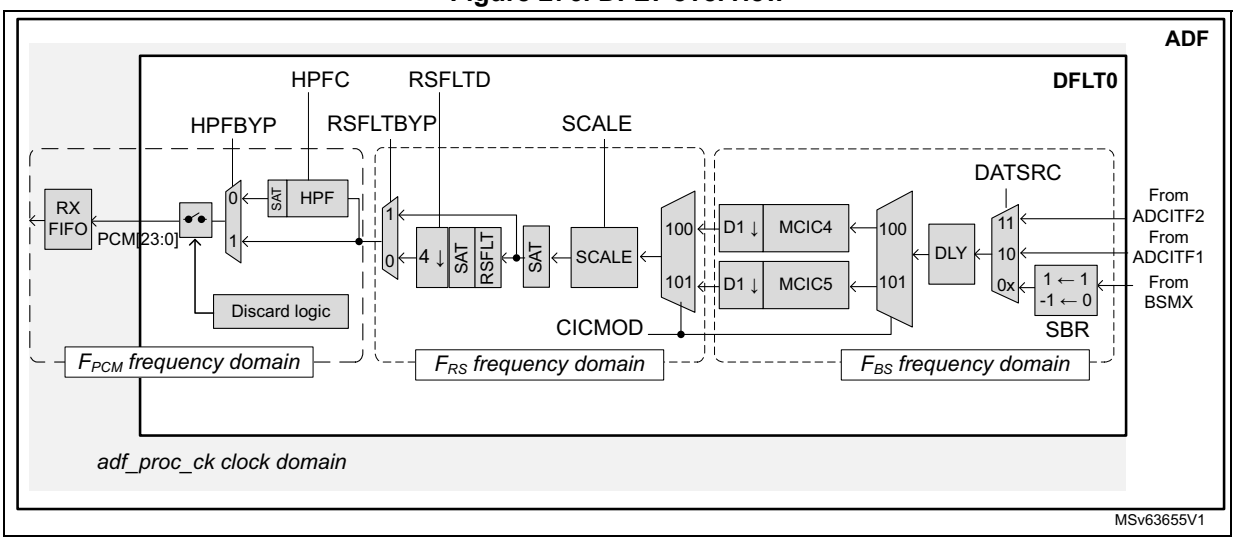
36.4.7 Digital filter processing (DFLT)

The digital filter processing includes the following sub-blocks:

- symbol remap (SBR)
- clock skipper delay (DLY)
- MCIC decimation filter that can be configured in $Sinc^4$ or $Sinc^5$
- gain control (SCALE)
- signal saturation (SAT)
- reshape filter (RSFLT)
- high-pass filter (HPF)
- receive RXFIFO

The figure below shows the filter path configuration according to CICMOD[2:0]. Several configuration bits are available to configure the digital filter to the application needs.

Figure 276. DFLT overview



Symbol remap and source selection

The symbol remap (SBR) converts the bitstream selected by the BSMX into data usable by the filter path. More especially:

- The high levels are converted into a 16-bit signed number + 1.
- The low levels are converted into a 16-bit signed number - 1.

The signal source of the digital filter can be selected via DATSRC[1:0] between the two following:

- data coming from the BSMX
- data coming from one of the ADC interfaces (ADCITF2 or 1)

Programmable micro-delay control (DLY)

The digital filter has a delay line that allows the timing adjustment of each stream with the resolution of the bitstream clock.

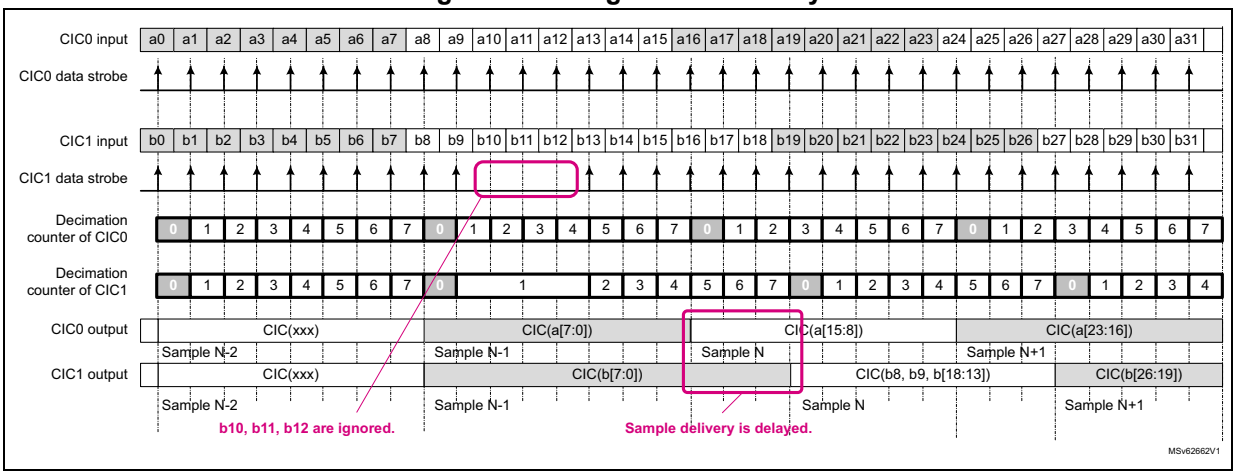
This feature is particularly helpful in the case of microphone beam forming applications where delays smaller than the final sampling rate must be applied to the incoming stream. This feature can be used when the ADF is synchronized with another MDF instance (if present in the product) for a beam forming application for example.

The delay is performed by discarding a given number of samples from the selected input stream, before samples enter into the CIC filter. This data discarding is done by skipping a given number of data strobe, preventing the CIC filter to take into account those data.

When the wanted amount of data strobe has been skipped, the next incoming samples are strobed normally.

The figure below shows an example on how to apply dynamically small delay to an incoming stream. For simplification, the CIC filter performs a decimation by height in this example. CIC1 represents the CIC included in the ADF and CIC0 represents a filter from another MDF instance (if present in the product).

Figure 277. Programmable delay



The CIC of the ADF (CIC1) receives a command in order to skip three incoming samples. So the input samples named b10, b11 and b12 are not processed by CIC1. As a consequence, the output sample N+1 generated by CIC0 is built from input samples a[23:16] while the sample N+1 of CIC1 is built from input samples b[26:19].

Finally, the non-skipped data stream looks delayed by three bitstream periods.

Note: When the input data strobes are skipped, the decimation counter remains frozen. As a consequence, the samples delivered by the CIC1 are a bit delayed.

The following steps are needed to program the amount of bitstream clock periods to be skipped:

1. Wait for SKPBF equal to 0.
2. Write SKPDLY[6:0] to the wanted number of bitstream clock periods to be skipped. The SKPBF flag goes immediately to 1, indicating that the delay value entered into SKPDLY[6:0] is under process.
   - If the DFLT0 is not yet enabled (DFLTEN = 0), then the DLY logic waits for DFLTEN = 1. When the application sets DFLTEN to 1, the DLY logic starts to skip the amount of wanted data strobes.
   - If the DFLT0 is already enabled (DFLTEN = 1), then the DLY logic immediately starts to skip the amount of wanted data strobes.

   When the ADF skipped the amount of wanted data strobes, then SKPBF goes back to 0.
3. If the application needs to skip more data strobes, then the operation must be restarted from step 1.

The effect of the delay performed with this mechanism is cumulative as long as the ADF is enabled. If the application performs a D1 delay followed by a D2 delay, then all other active filters are delayed by D1 + D2.

Note: If SKPDLY[6:0] is written when SKPBF = 1, the write operation is ignored.

Cascaded-integrator-comb (CIC) filter

The CIC digital filters are an efficient implementation of low-pass filters, often used for decimation and interpolation. The CIC frequency response is equal to a $Sinc^N$ function, this is why they are often called Sinc filters.

The Sinc^N digital filter embedded into the ADF can be configurable in Sinc^4 or Sinc^5, according to CICMOD:

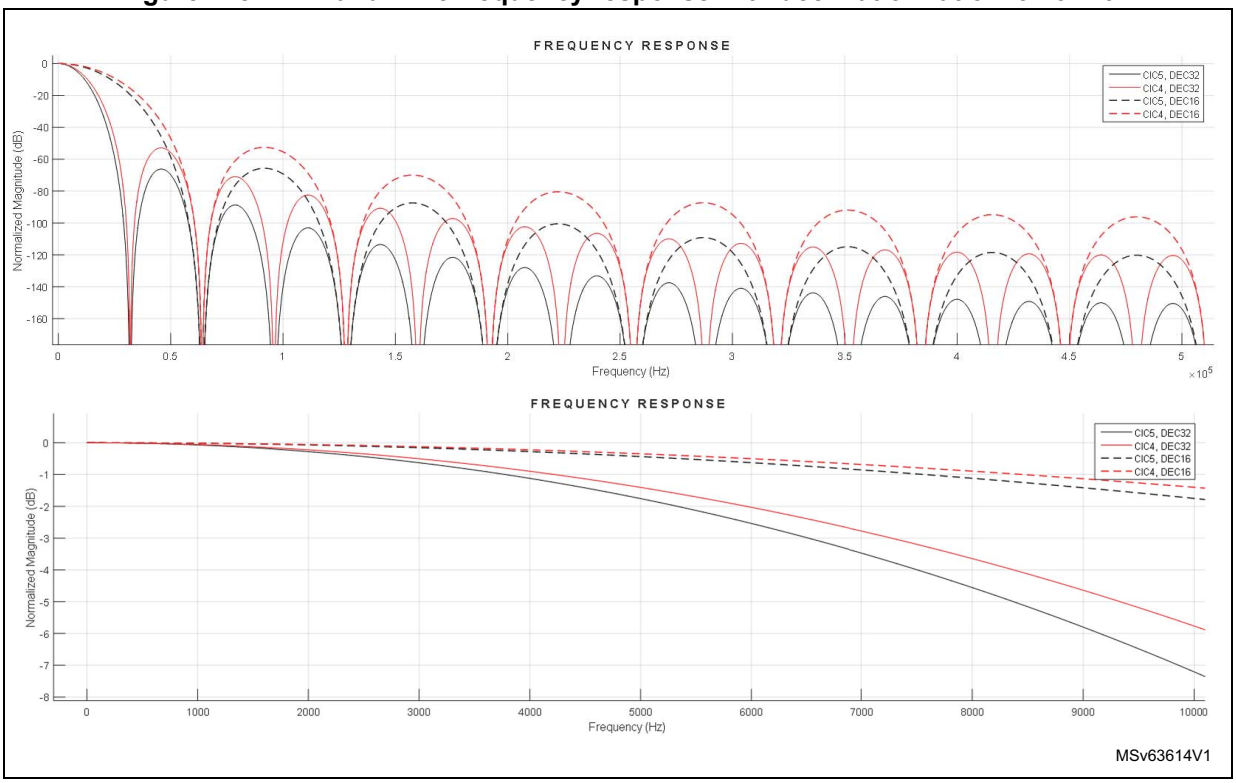
- If CICMOD[2:0] = 4, Sinc^4 is selected.
- If CICMOD[2:0] = 5, Sinc^5 is selected.

The filters have the following transfer function:

$$H(z) = \left(\frac{1 - z^{-D}}{1 - z^{-1}}\right)^N$$

where N can be 4 or 5, and D is the decimation ratio. D is equal to MCICD+1.

Figure 278. CIC4 and CIC5 frequency response with decimation ratio = 32 or 16



CIC output data size

The size of samples delivered by the CIC (DS_{CIC}), depends on the following parameters:

- CIC order (N)
- CIC decimation ratio (D)
- data size of the input stream (DSIN)

The CIC order and decimation ratio must be programmed in order to insure that the data size does not exceed the 26-bit CIC capability.

The following formula gives the output data size (DS_{CIC}) according to the parameters above.

$$DS_{CIC} = \left(\frac{N \times Ln(D)}{Ln(2)}\right) + DS_{IN}$$

and the CIC gain is given by this formula:

$$G_{CIC} = (D)^N$$

The decimation ratio can be adjusted from 2 to 512 for the CIC filter.

The table below gives some data output size in bits for some decimation values, when the data source is a full-scale signal coming from the serial interface or from a 12-bit ADC.

Note: DS_{IN} = 1 bit for a serial bitstream, but can be up to 16 bits when coming from the ADCITF.

Table 317. Data size according to CIC order and CIC decimation values

Decimation	Data size (bits) when DS_{IN} = 1 bit (data from SITF0)		Data size (bits) when DS_{IN} = 12 bits (data from ADCITF)	
	$Sinc^4$	**$Sinc^5$**	**$Sinc^4$**	**$Sinc^5$**
4	9	11	20	22
8	13	16	24	-
10	15	18	26	-
12	16	19	-	-
16	17	21	-	-
20	19	23	-	-
24	20	24	-	-
32	21	26	-	-
48	24	-	-	-
64	25	-	-	-
76	26	-	-	-

Note: For a full-scale input signal, the decimation ratio must not exceed 76 for a $Sinc^4$ and 32 for a $Sinc^5$.

The LSB parts of the data provided by the CIC is not necessarily significant: it depends on the sensor performances and the ability of the CIC to reject the out-off band noise.

The sample size at CIC output can be adjusted thanks to the SCALE block.

Scaling (SCALE) and saturation (SAT)

The SCALE block allows the application to adjust the amplitude of the signal provided by the CIC, by steps of 3 dB (± 0.5 dB).

The signal amplitude can be decreased by up to 8 bits (- 48.2 dB), and can be increased by up to 12 bits (+ 72.2 dB).

The gain is adjusted by the SCALE[5:0] field in the *ADF digital filer configuration register 0 (ADF_DFLT0CICR)*.

SCALE[5:0] can be changed even if the DFLT0 is enabled. During the gain transition, the signal provided by the filter is disturbed.

Due to internal resynchronization, there is a delay of some cycles of adf_proc_ck clock between the moment where the application writes the new gain, and the moment where the gain is effectively applied to the samples. If the application attempts to write a new gain value while the previous one is not yet applied, this new gain value is ignored. Reading back SCALE[5:0] informs the application on the current gain value.

The table below shows the possible gain values.

Table 318. Possible gain values

SCALE[5:0]	Gain (dB)	SCALE[5:0]	Gain (dB)	SCALE[5:0]	Gain (dB)	SCALE[5:0]	Gain (dB)
0x20	- 48.2	0x2B	- 14.5	0x06	+ 18.1	0x11	+ 51.7
0x21	- 44.6	0x2C	- 12.0	0x07	+ 21.6	0x12	+ 54.2
0x22	- 42.1	0x2D	- 8.5	0x08	+ 24.1	0x13	+ 57.7
0x23	- 38.6	0x2E	- 6.0	0x09	+ 27.6	0x14	+ 60.2
0x24	- 36.1	0x2F	- 2.5	0x0A	+ 30.1	0x15	+ 63.7
0x25	- 32.6	0x00	0.0	0x0B	+ 33.6	0x16	+ 66.2
0x26	- 30.1	0x01	+ 3.5	0x0C	+ 36.1	0x17	+ 69.7
0x27	- 26.6	0x02	+ 6.0	0x0D	+ 39.6	0x18	+ 72.2
0x28	- 24.1	0x03	+ 9.5	0x0E	+ 42.1	-	-
0x29	- 20.6	0x04	+ 12.0	0x0F	+ 45.7	-	-
0x2A	- 18.1	0x05	+ 15.6	0x10	+ 48.2	-	-

The SAT blocks avoid having a wrap-around of the binary code when the code exceeds its maximal or minimal value.

The ADF performs saturation operations at the following levels:

- after the SCALE block (performed by the SAT block): The signal is saturated at 24 bits.
- inside the RSFLT, to insure a good filter behavior
- at the output of the HPF, to insure that the output signal does not exceed 24 bits

The SATF bit informs the application that a saturation occurred either after the SCALE, inside the RSFLT or after the HPF. In addition, an interrupt can be generated if SATIE is set to 1. As soon as a saturation is detected, the SATF flag is set to 1. It is up to the application to clear this flag in order to be able to detect a new saturation.

Those bits are in the *ADF DFLT0 interrupt enable register (ADF_DFLT0IER)* and *ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR)*.

Gain adjustment policy

To get the best ADF performances, it is important to properly adjust the gain value via SCALE[5:0].

A usual way to adjust the gain is to select the SCALE[5:0] value that gives a final signal amplitude as close as possible to the 24-bit full-scale, for the maximum input signal.

A way to select the optimal gain is detailed below:

1. Check that, for the expected input signal, the data size into the CIC filter does not exceed 26 bits. This can be checked using this formula:

$$\frac{LN(SIN_{pp} \cdot D^N)}{LN(2)} < 26$$

where N represents the CIC order, D the decimation ratio and SIN_{pp} the maximum peak-to-peak amplitude of the input signal.

SIN_{pp}can take:

– a maximum peak-to-peak amplitude of 2 (± 1), for samples coming from SITF0

– A maximum peak-to-peak amplitude of 4095 (+ 2047, - 2048), for samples coming from a 12-bit ADC

Example: a $Sinc^4$ can be used with a decimation ratio of 96, if the maximum input signal does not exceed ± 0.35. Indeed:

$$\frac{LN(0.7 \cdot 96^4)}{LN(2)} \sim 25.82 \text{ bits} < 26 \text{ bits}$$

2. Adjust the SCALE value.

   To select the most appropriate SCALE value, the user must check if the RSFLT is used or not. If the RSFLT is used, the data size at SCALE output must not exceed 22 bits, otherwise the data size can be up to 24 bits.

   The SCALE value in dB must be selected using this formula:

$$SCALE_{dB} < 20 \cdot \log 10\left(\frac{2^{NB}}{SIN_{pp} \cdot D^{N}}\right)$$

   where NB is equal to 22 if RSFLT is enabled, or 24 if RSFLT is bypassed. $SCALE_{dB}$ represents the gain value selected by SCALE[5:0].

   Example: For a $Sinc^4$ with a decimation ratio of 96 and a SIN_{pp} of 0.7.

   – If the RSFLT is bypassed:

$$20 \cdot \log 10\left(\frac{2^{24}}{0.7 \cdot 96^{4}}\right) \sim -11 \text{ dB}$$

     $SCALE_{dB}$ value must be lower than - 11 dB, the closest lower value is - 12dB (SCALE[5:0] = 0x2C).

   – If the RSFLT is enabled:

$$20 \cdot \log 10\left(\frac{2^{22}}{0.7 \cdot 96^{4}}\right) \sim -23 \text{ dB}$$

     $SCALE_{dB}$ value must be lower than - 23 dB, the closest lower value is - 24.1 dB (SCALE[5:0] = 0x28).

   If SCALE[5:0] is set to a higher value, then a saturation may occur. An event flag informs the user if a saturation occurred.

The table below proposes gain values for different filter configurations, when the data comes from the SITF0, according to the MCIC order, and the MCIC decimation ratio. This table is not exhaustive and considers a full-scale input signal (see *Section 36.7.5: Total ADF gain* for details).

Table 319. Recommended maximum gain values versus CIC decimation ratios

CIC decimation ratio	Gain settings (dB) for configuration SITF + CICx + RSFLT (+ HPF)		Gain settings (dB) for configuration SITF + CICx (+ HPF)	
	CIC5	CIC4	CIC5	CIC4
8	33.6	51.7	45.7	63.7
12	18.1	39.6	30.1	51.7
16	3.5	27.6	15.6	39.6
20	- 6.0	21.6	6.0	33.6
24	- 12.0	15.6	0	27.6
28	- 20.6	9.5	- 8.5	21.6

Table 319. Recommended maximum gain values versus CIC decimation ratios (continued)

CIC decimation ratio	Gain settings (dB) for configuration SITF + CICx + RSFLT (+ HPF)		Gain settings (dB) for configuration SITF + CICx (+ HPF)	
	CIC5	CIC4	CIC5	CIC4
32	- 26.6	3.5	- 4.5	15.6
48	-	- 8.5	-	3.5
64	-	- 20.6	-	- 8.5

Reshaping filter (RSFLT)

In addition to the CIC, the ADF offers a reshaping IIR filter mainly dedicated to the audio application, but also usable in other applications.

When the RSFLT is used, the sample size at its input must not exceed 22 bits.

The samples at the RSFLT output can be decimated by four or not according to the RSFLTD bit in the *ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)*.

The RSFLT can be bypassed by setting RSFBYP to 1 in the *ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)*.

The table below shows which sampling rate must be provided to the RSFLT in order to process the most common audio streams.

The RSFLT cutoff frequency (F_C) depends on the sample rates at its input (F_{RS}), and is given by the following formula:

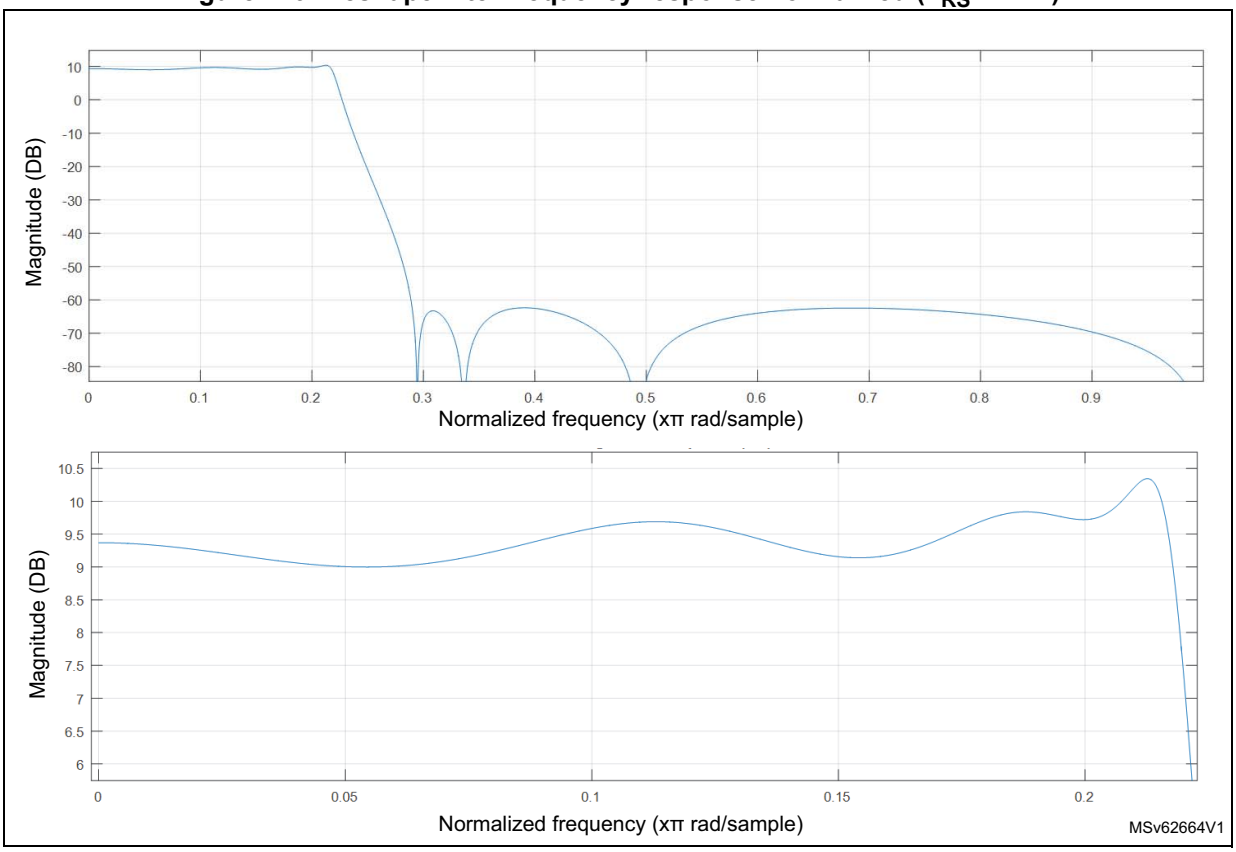
$$F_C = 0.111 \times F_{RS}$$

Table 320. Most common microphone settings

Sample rate (kHz) at RSFLT (F_{RS})	Pass band (kHz)	D2	PCM sampling rate (kHz)
32	3.55	4	8
64	7.1	4	16
128	14.2	4	32
192	21.3	4	48

The figure below shows the frequency response of the reshape filter.

Figure 279. Reshape filter frequency response normalized (F_{RS} / 2 = 1)



The RSFLT gain is close to 9.3 dB, so the output data size is a little bit lower than 24 bits for a 22-bit wide input signal.

The RSFLT takes 24 clock cycles of adf_proc_ck clock to process one sample at F_{RS}. When the RSFLT is enabled, the application must insure that the adf_proc_ck is at least 24 times faster F_{RS}.

The RSFLT generates an event (rfovr_evt) and sets the RFOVRF flag, if the RSFLT receives a new samples while the previous one is still under processing.

When RFOVRF is set, the samples provided by the RSFLT are invalid. The application must then stop the data acquisition and provides a faster adf_proc_ck clock to the RSFLT.

High-pass filter (HPF)

The high-pass filter suppresses the low-frequency content from the final output data stream in case of continuous conversion mode. The high-pass filter can be enabled or disabled via HPFBYP in the *ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)*.

The HPF is useful when there is parasitic low-frequency noise (or DC signal) in the input data source that must be removed from the final data.

The HPF is a first order IIR filter and the cut-off frequency can be selected via HPFC[1:0] in the *ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)*, among the following values:

- 0.000625 x F_{PCM}
- 0.00125 x F_{PCM}
- 0.00250 x F_{PCM}
- 0.00950 x F_{PCM}

Table 321. HPF 3 dB cut-off frequencies examples

HPFC	3 dB cut-off frequency for common F_{PCM} frequencies (Hz)		
	F_{PCM} = 8 kHz	F_{PCM} = 16 kHz	F_{PCM} = 48 kHz
0	5	10	30
1	10	20	60
2	20	40	120
3	76	152	456

The HPF output is saturated at 24 bits. The SATF flag is set if a sample is saturated.

36.4.8 Digital filter acquisition modes

The ADF offers the following modes to perform a data capture:

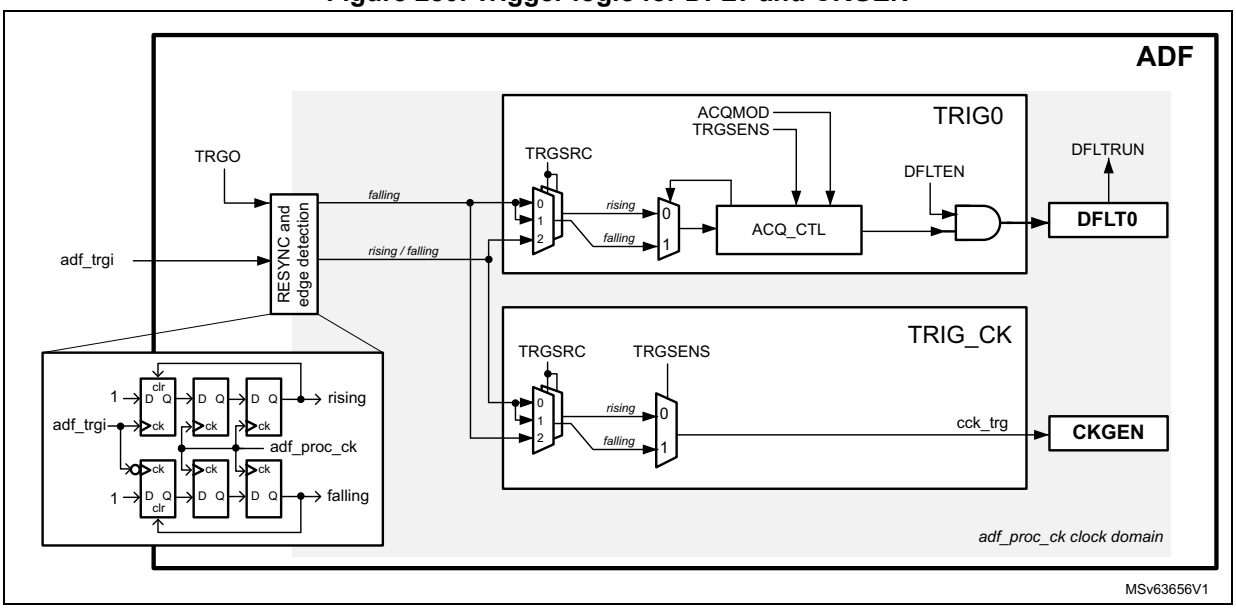
- Asynchronous continuous acquisition mode
- Asynchronous single-shot acquisition mode
- Synchronous continuous acquisition mode
- Synchronous single-shot acquisition mode
- Window continuous acquisition mode

Note: To perform a data capture, the filter, the interface providing the data (SITF0 or ADCITF) and the CKGEN must be enabled. If needed, the ADF_CCK0 or ADF_CCK1 must be enabled as well.

The filter can be stopped immediately when DFLTEN is set to 0. This action resets the filter and flushes the RXFIFO. The DFLTACTIVE flag also goes back to 0 when the RXFIFO and the filter is reset.

The figure below shows a simplified view of the trigger logic available for each filter and for the clock generator.

Figure 280. Trigger logic for DFLT and CKGEN



A block common to all TRIG blocks performs the rising and falling edges detection and the re-synchronization of the input trigger to the adf_ker_ck clock domain. This implementation allows the application to use triggers with pulse width smaller than the adf_ker_ck period.

In synchronous modes, the TRIG block offers the possibility to select adf_trgi or TRGO bit as trigger sources. The TRGO bit is in the *ADF global control register (ADF_GCR)*.

The edge sensitivity can also be selected.

Asynchronous continuous acquisition mode

This mode allows the application to start a continuous acquisition by simply writing the DFLTEN bit to 1.

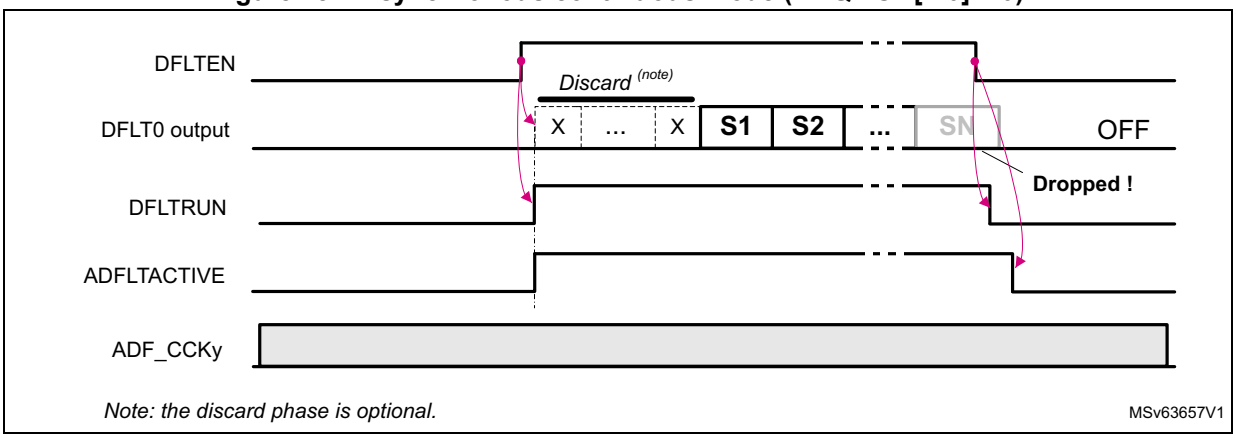
The Asynchronous continuous acquisition mode is selected when ACQMOD[2:0] = 0.

The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN) so that the adf_proc_ck frequency is compatible with the targeted application (see examples in *Table 327*).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter configuration and set the ACQMOD[2:0] to 0.
4. Set to 1 the SITFEN bit of the serial data interface.
5. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
6. When DFLTEN is set to 1, the acquisition sequence starts immediately.

The figure below shows a simplified example of the samples generated by the DFLT0.

Figure 281. Asynchronous continuous mode (ACQMOD[2:0] = 0)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Asynchronous single-shot acquisition mode

This mode allows the application to start the acquisition of one sample by simply writing the DFLTEN bit to 1.

The Asynchronous single-shot acquisition mode is selected when ACQMOD[2:0] = 001.

The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN), so that the adf_proc_ck frequency is compatible with the targeted application (see examples in *Table 327*).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter configuration, and set the ACQMOD[2:0] to 001.
4. Set to 1 the SITFEN bit.
5. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
6. When DFLTEN is set to 1, the filter provides one data to the RXFIFO and stops the acquisition.

To trigger a new acquisition, the application must:

1. Check that the previous acquisition is completed, by waiting that DFLTRUN = 0.
2. Set again DFLTEN to 1.

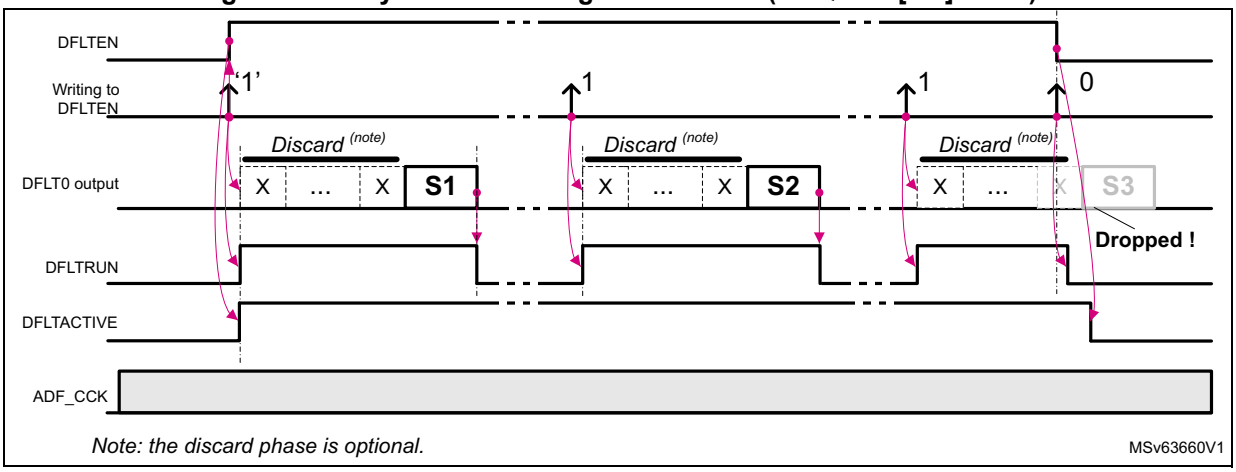
This sequence can be repeated every time a new data must be converted.

As shown in the *Figure 282*, every time DFLTEN is set to 1, an acquisition sequence is triggered. The first samples provided by the filter can be discarded if needed. At the end of each conversion, the decimation counters and filter taps are reset, and the filter is ready to start a new conversion.

If DFLTEN is set to 0 while an acquisition is ongoing, the ongoing conversion is stopped (in the example, S3 is lost). This situation can be avoided with the following steps:

1. Wait for DFLTRUN = 0.
2. Read the sample from the RXFIFO.
3. Set DFLTEN to 0.

Figure 282. Asynchronous single-shot mode (ACQMOD[2:0] = 001)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Synchronous continuous acquisition mode

This mode allows the application to start a continuous acquisition by using one of the following trigger sources:

- adf_trgi signal
- TRGO bit

The Synchronous continuous acquisition mode is selected when ACQMOD[2:0] = 010.

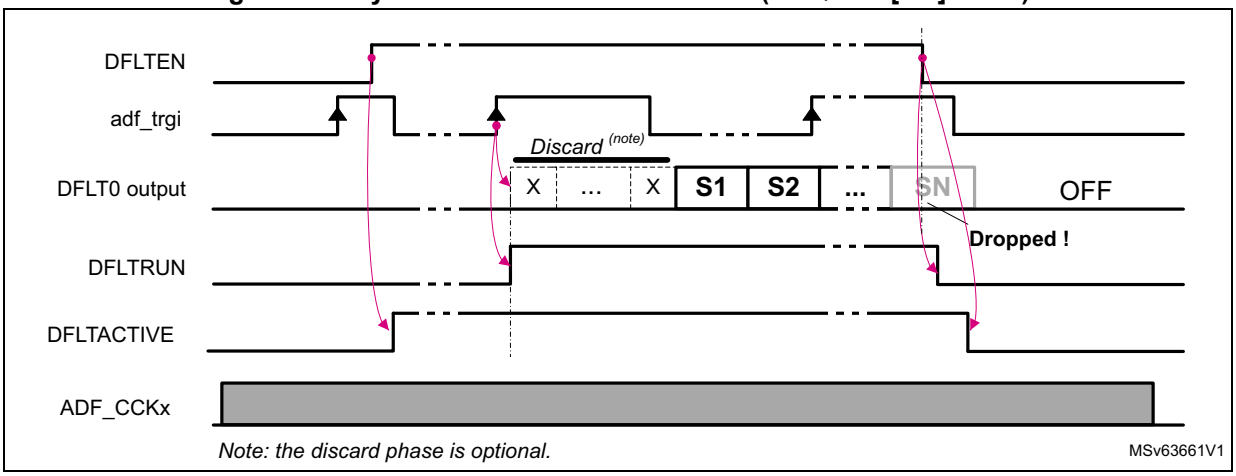
The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of adf_proc_ck clock is compatible with the targeted application (see examples in *Table 327*).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter configuration and set the ACQMOD[2:0] to 010.
4. Set to 1 the bit SITFEN.
5. Select the proper trigger source and sensitivity.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
7. Set DFLTEN to 1.
8. When the trigger condition is met, the filter starts the acquisition.

The TRGSENS bit allows the selection of the trigger edge (rising or falling). The trigger is ignored if an acquisition is ongoing or if DFLTEN is set to 0.

The figure below shows a simplified example where the trigger logic is sensitive to a rising edge trigger (TRGSENS = 0). The first rising edge of the trigger signal is ignored because DFLTEN = 0. Then the next rising edge is taken into account and starts the acquisition. All other rising edges are ignored. The trigger logic is re-initialized when DFLTRUN goes back to 0.

Figure 283. Synchronous continuous mode (ACQMOD[2:0] = 010)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Synchronous single-shot acquisition mode

This mode allows the application to start a single acquisition by using one of the following trigger sources:

- adf_trgi signal
- TRGO bit

The Synchronous single-shot acquisition mode is selected when ACQMOD[2:0] = 011.

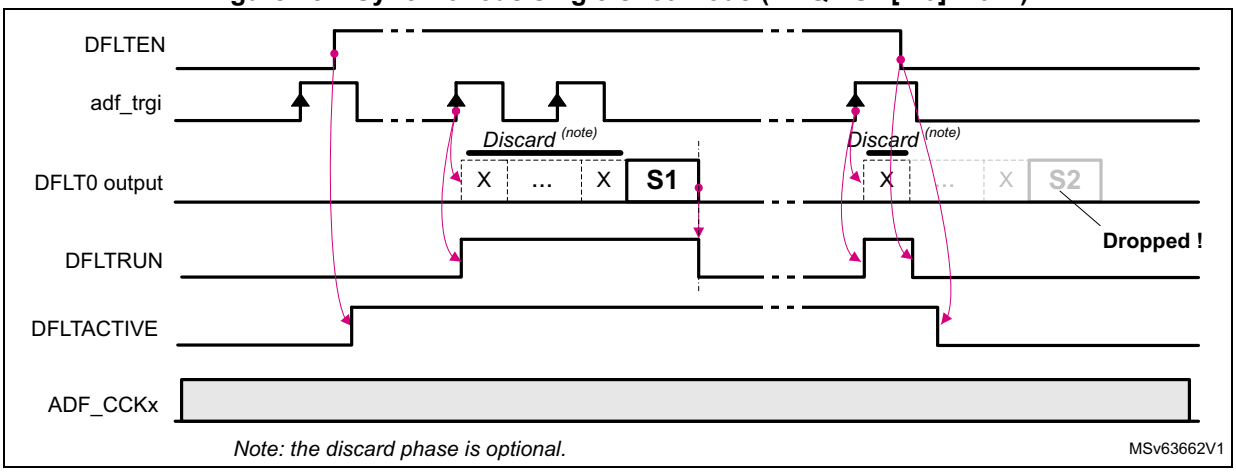
The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of adf_proc_ck clock is compatible with the targeted application (see examples in *Table 327*).
2. Enable the CKGEN and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter configuration, and set the ACQMOD[2:0] to 011.
4. Set to 1 the SITFEN bit.
5. Select the proper trigger source and sensitivity.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
7. Set DFLTEN to 1.
8. When the trigger condition is met, the filter starts the acquisition and provides one data to the RXFIFO, then the filter is ready to accept a new trigger.

TRGSENS allows the selection of the trigger edge (rising or falling). The trigger is ignored if an acquisition is ongoing, or if DFLTEN is set to 0.

The figure below shows a simplified example where the trigger logic is sensitive to a rising edge trigger (TRGSENS = 0). Every-time a trigger rising edge is detected with DFLTEN = 1, an acquisition sequence is triggered. The first samples provided by the filter can be discarded if needed. At the end of each conversion, the decimation counters and filter taps are reset. DFLTRUN is set to 0 and the filter is ready to start a new conversion.

Figure 284. Synchronous single-shot mode (ACQMOD[2:0] = 011)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Figure 284 shows a case where the DFLTEN is set to 0 while an acquisition is ongoing (the sample S2 is lost). This situation can be avoided with the following steps:

1. Wait for DFLTRUN = 0.
2. Read the sample from the RXFIFO.
3. Clear DFLTEN to 0.

Window continuous acquisition mode

This mode allows the application to start or stop a continuous acquisition controlled by consecutive edges of one of the following trigger sources:

- adf_trgi signal
- TRGO bit

The Window continuous acquisition mode is selected when ACQMOD[2:0] = 100.

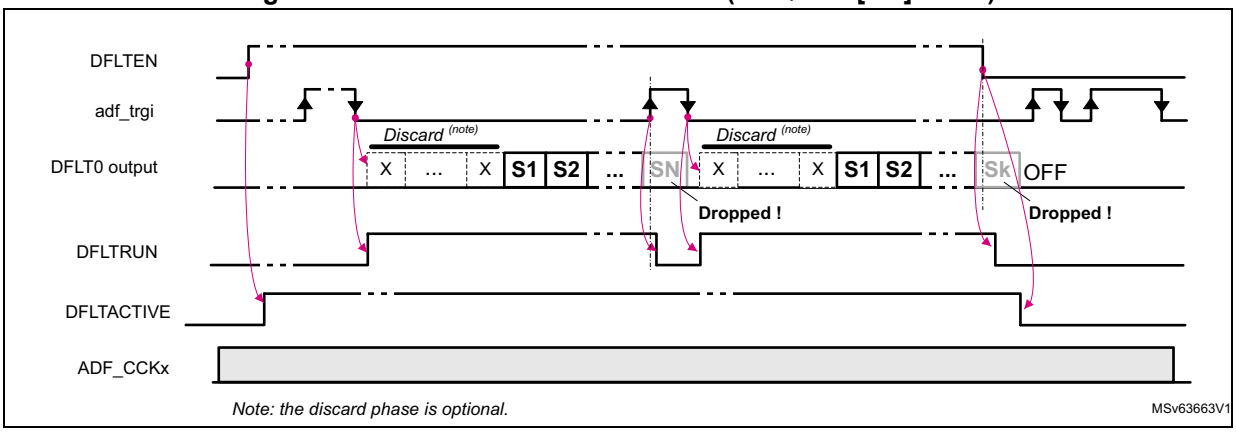
The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of adf_proc_ck clock is compatible with the targeted application (see examples in *Table 327*).
2. Enable the CKGEN and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter settings and set the ACQMOD[2:0] to 100.
4. Set to 1 the SITFEN bit.
5. Select the proper trigger source and sensitivity.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
7. Set DFLTEN to 1.
8. If TRGSENS = 0, the acquisition starts on trigger rising edge and stops on trigger falling edge. If TRGSENS = 1, the acquisition starts on trigger falling edge and stops on trigger rising edge.

Note: The acquisition may restart if the trigger condition becomes again active.

Figure 285 shows a simplified example of window continuous acquisition mode, with TRGSENS = 1. Once DFLTEN is set to 1, the ADF waits for a falling edge on the selected trigger input. When the trigger condition is met, DFLTRUN goes to 1 and the acquisition starts. The acquisition stops if the ADF detects a rising edge on the selected trigger input. If DFLTEN is still set to 1, the ADF waits again for a falling edge on the selected trigger input.

Figure 285. Window continuous mode (ACQMOD[2:0] = 100)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Starting several filters synchronously

If the ADF is used with MDF instances (if present in the product), it is possible to start simultaneously the acquisition of all the filters. This synchronization capability depends on the way the triggers are connected in the product. Generally, an ADF is able to trigger MDF instances, if its adf_trgo signal is connected as trigger input to those blocks (see *Section 36.4.2: ADF pins and internal signals* to check trigger capabilities).

In the following programing example, one ADF has its adf_trgo signal connected to some MDFs. To start the acquisition of several filters synchronously, the following sequence must be performed (assuming that DFLTEN bits of the filters are set to 0):

On MDFs receiving the adf_trgo trigger:

1. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
2. Set to 1 the SITFEN bit of the requested data interfaces.
3. For each filter, set the acquisition mode to synchronous (ACQMOD[2:0] = 01x).
4. For each filter, set TRGSRC[3:0] in order to select the adf_trgo trigger input.
5. For each filter, set TRGSENS to 0 (rising edge).
6. For each filter, set DFLTEN to 1.

On the ADF generating the adf_trgo trigger:

1. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK[1:0] clocks.
2. Set to 1 the SITFEN bit of the requested data interfaces.
3. Set the acquisition mode to synchronous (ACQMOD[2:0] = 01x).
4. Set TRGSRC[3:0] to 0 (TRGO selected).
5. Set TRGSENS to 0 (rising edge).
6. Set DFLTEN to 1.
7. Read TRGO bit until it is read to 0.
8. Set TRGO to 1. Then the acquisition sequence for all selected filters starts immediately.

To trigger a new acquisition (in case of single-shot) the application must do the following:

- Check that the previous acquisition is completed, by waiting DFLTRUN = 0.
- Read TRGO until it is read to 0.
- Set again the bit TRGO to 1.

Discarded samples

The ADF offers the possibility to program the amount of samples to be discarded after each restart:

- to avoid capturing samples affected by the impulse response of the filter
- to delay the acquisition of filters by a specific amount of samples

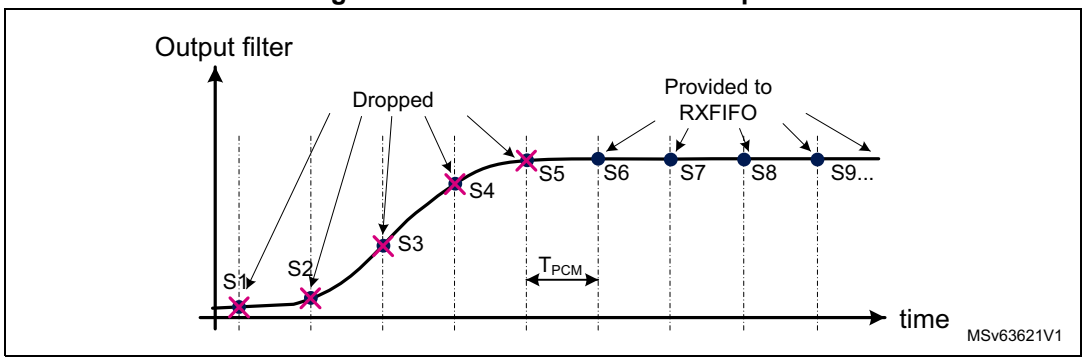
The discard function is controlled via NBDIS[7:0] as follows:

- When NBDIS[7:0] = 0, the discard function is disabled.
- When NBDIS[7:0] ≠ 0, the discard function is activated in one of the following condition:
  - when the DFLTEN bit goes to 1
  - every time an acquisition is started in (A)synchronous single-shot modes

Refer to *Figure 281* to *Figure 285*, and *Figure 287*.

In the example shown in the figure below, the discard function is used to drop the first five samples provided by the digital filter (S1 to S5). The first sample transfered to the RXFIFO is S6.

Figure 286. Discard function example



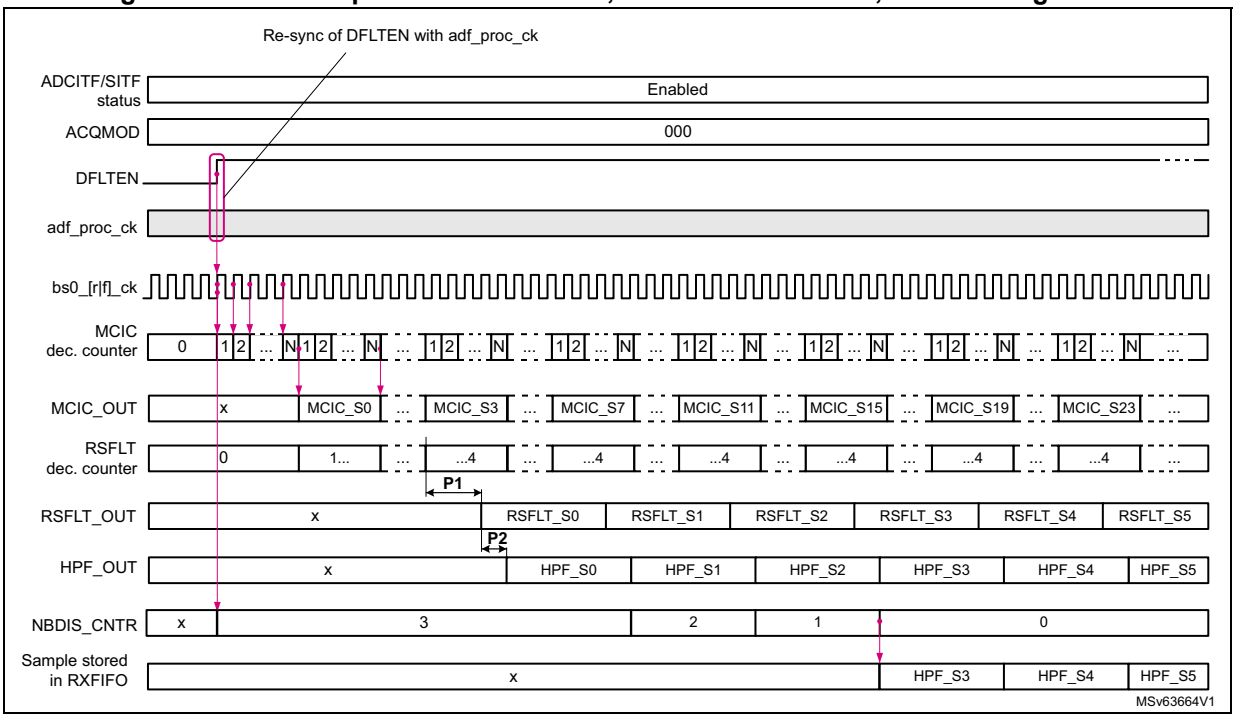
36.4.9 Start-up sequence examples

The figure below details a start of acquisition sequence of a digital filter triggered by DFLTEN (ACQMOD[2:0] = 0), with NBDIS[7:0] = 3 (three samples to discard before acquisition).

The DFLT0 is configured for audio application: MCIC, RSFLT and HPF activated.The data interface (SITF0 or ADCITF) is assumed to be already activated.

Note: NBDIS[7:0] is set on purpose to a small value to simplify the drawing.

Figure 287. Start sequence with DFLTEN, in continuous mode, audio configuration



The DFLTEN bit is re-sampled into the ADF processing clock domain. When DFLTEN is detected high, the filter chain is enabled, and the decimation counter of the MCIC filter is incremented at the rate of the bitstream clock.

When the MCIC decimation counter reached its programmed value N, a sample is available for the RSFLT.

The RSFLT processes all the samples provided by the MCIC, and delivers a sample to the HPF every-time it processes four samples (decimation by 4). The RSFLT needs up to 24 cycles of adf_proc_ck clock before delivering a sample (P1).

The HPF processes all the samples provided by the RSFLT, but the NBDIS function prevents the data writing in the RXFIFO as long as NBDIS_CNTR does not reach 0.

When NBDIS_CNTR reaches 0, the samples provided by the HPF are stored into the RXFIFO.

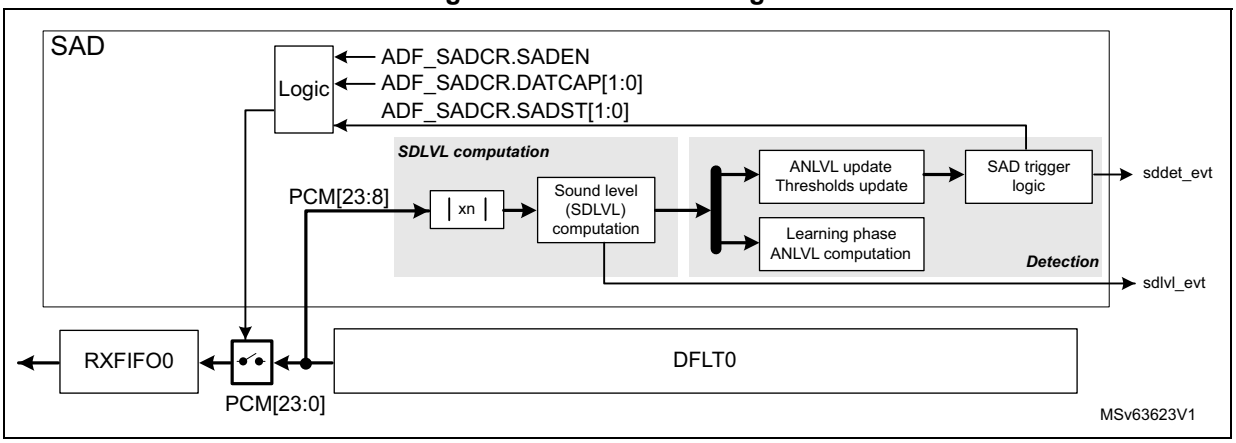
36.4.10 Sound activity detection (SAD)

The SAD is based on the computation of the ambient noise level (ANLVL) and of the short-term sound level (SDLVL). The SAD offers the following ways to detect a sound:

- when the SDLVL reaches a threshold referenced to the ambient noise level
- when the SDLVL reaches a fixed threshold
- when the ANLVL reaches a fixed threshold

As shown in the figure below, the SAD takes the 16 MSB samples from the DFLT0 output.

Figure 288. SAD block diagram



The SAD is highly configurable, and the application can adjust several parameters:

- SAD detection behavior (SADMOD)
- number of samples used to compute the sound level (FRSIZE)
- number of frames used to compute the ambient noise level during the learning phase (LFRNB)
- slope of the ambient noise estimator (ANSLP)
- minimum expected ambient noise level (ANMIN)
- threshold level (SNTHR)
- threshold hysteresis (HYSTEN)
- hangover window in order to filter spurious transitions between DETECT and MONITOR states (HGOVR)
- data capture mode (DATCAP)

SAD detection behavior

The SAD can use the following ways to detect a sound, selected by SADMOD[1:0]:

- When SADMOD[1:0] = 0, the SAD works like a voice-activity detection. In this mode, the SAD estimates the ambient noise level according to the computed sound level values. The threshold of the trigger is elaborated from the estimated ambient noise. Finally the current sound level is compared to this threshold. In a first approximation, the SAD triggers if the peak-to-average value of the input signal reaches a level defined by SNTHR[3:0].
- When SADMOD[1:0] = 01, the SAD compares the current sound level (SDLVL) to a fixed trigger value defined by the application via SNTHR[3:0] and ANMIN[12:0]. This mode allows a fast SAD reaction as the amount of samples used to compute the sound level can be configured via FRSIZE[2:0].
- When SADMOD[1:0] = 1x, the SAD compares the estimated ambient noise level (ANLVL) to a fixed trigger value defined by the application via SNTHR[3:0] and ANMIN[12:0]. This mode avoids unwanted triggers, due to peak levels, but the SAD reacts more slowly to an input signal variation. It is nevertheless possible to adjust the reaction time via FRSIZE[2:0] and ANSLP[2:0].

SAD states

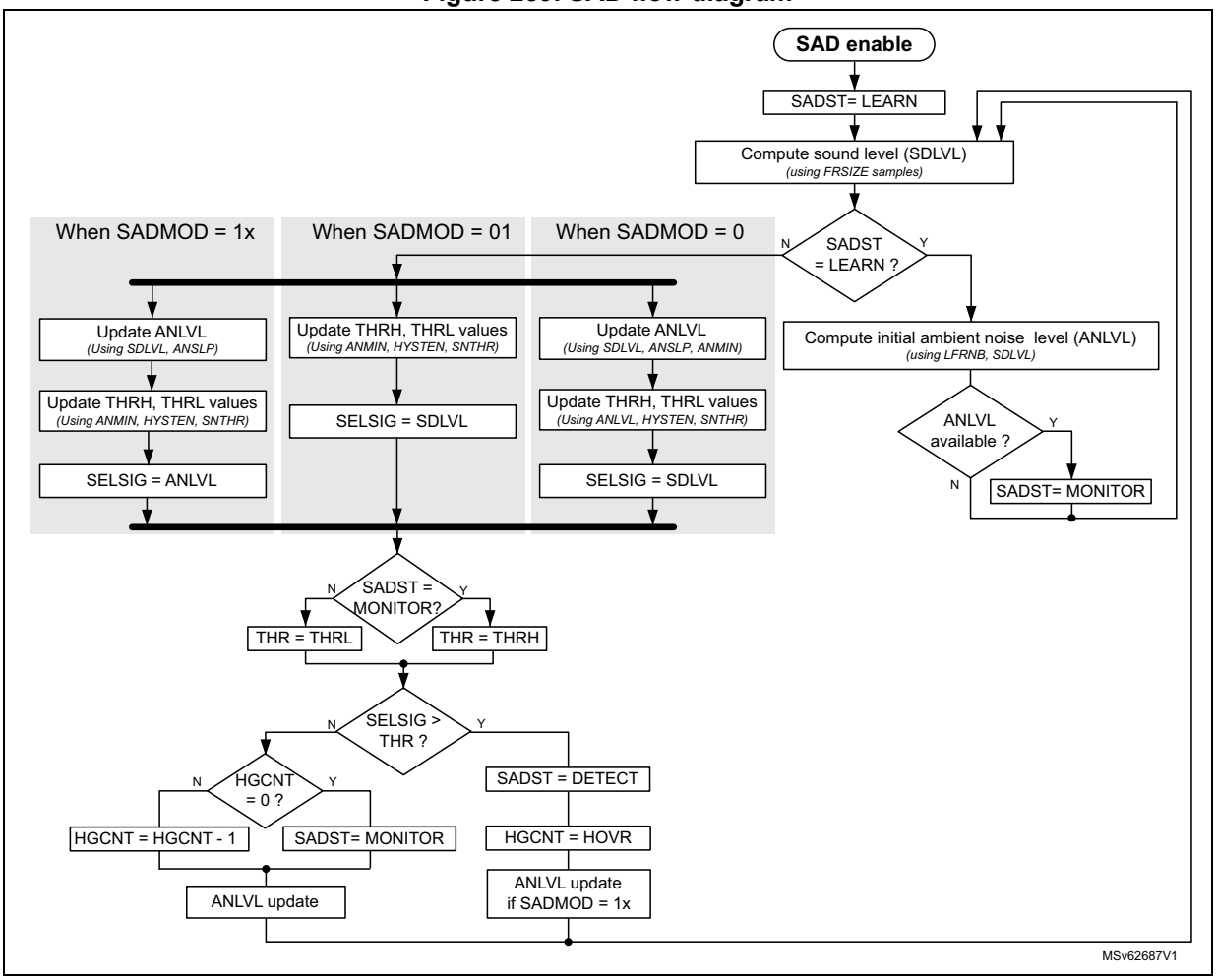
As shown in *Figure 289*, the SAD works as follows:

1. When enabled (SADEN = 1), the SAD is first in LEARN state to perform a first estimation of the ambient noise level.
2. The SAD continuously computes the short-term sound level (SDLVL) using the samples provided by the DFLT0. The amount of samples used to compute the sound

level is given by FRSIZE[2:0]. The samples processed by the DFLT0 can be transferred into the memory or not depending on DATCAP[1:0] value.

3. The initial ambient noise level (ANLVL) is computed using the consecutive sound level values. The application can define how much sound level values are used to perform the computation of this initial ambient noise estimation (LFRNB).
4. When the initial ambient noise level (ANLVL) is computed, the SAD switches to the MONITOR state.
5. Every time a new short-term sound level value is available, the SAD updates the ambient noise level and the thresholds according to the selected detection mode.
6. If the SAD triggers, then the following happens:
   – The SAD switches to DETECT state.
   – The sddet_evt event is asserted.
   – The adf_sad_det signal is set to high.
7. The hangover function insures that the DETECT state is maintained even if the sound level goes below the threshold level for a time given by HGOVR.

Figure 289. SAD flow diagram



Sound level computation (SDLVL)

Once enabled, the SAD computes continuously the sound level value. The sound level represents the average of the absolute value of an amount of PCM samples given by FRSIZE[2:0].

$$SDLVL = \frac{1}{N_{FRSIZE}} \times \sum_{n=1}^{N_{FRSIZE}} |PCM(n)|$$

where N_{FRSIZE} is the amount of PCM samples given by FRSIZE[2:0].

Ambient noise estimation (ANLVL)

The ambient noise level (ANLVL) is computed when SADMOD[1:0] is 00 or 10.

The ambient noise level is computed differently according to the state of the SAD as detailed below:

- ANLVL computation during the LEARN state

  Every time the SAD is enabled, a learning phase is initiated in order to estimate a first value of the ambient noise level. During this phase, the SAD cannot trigger.

  During the LEARN phase, the ambient noise level is computed as follows:

$$ANLVL = \frac{1}{N_{LFRNB}} \times \sum_{n=1}^{N_{LFRNB}} |SDLVL(n)|$$

  where N_{LFRNB} is the amount of frames given by LFRNB[2:0] field.

- ANLVL computation during the MONITOR or DETECT state

  When the learning phase is completed, the SAD updates the ambient noise level in the following way:

  a) The SAD computes the new possible values for the ambient noise level:

     ANLVL_UP = ANLVL * (1+2^(ANSLP-12))

     ANLVL_DN = ANLVL * (1-2^(ANSLP-10))

  b) The ANLVL takes the ANLVL_DN value if the current sound level is lower than ANLVL_DN, otherwise ANLVL takes the value of ANLVL_UP.

     The ANLVL is not updated if the current sound level is higher than the threshold level, except if SADMOD[1:0] = 10.

  c) When SADMOD[1:0] = 0, if the new ANLVL value is lower than ANMIN[12:0], ANLVL is replaced by ANMIN.

  The slope of the noise estimator can be adjusted to optimize the detection of the wanted signal. This slope is adjusted via ANSLP[2:0] in the *ADF SAD configuration register (ADF_SADCFGR)*.

The table below shows the allowed values according to the frame size and the sampling rate of the data observed by the SAD. The recommended values when the SADMOD[1:0] = 0 are the ones into the gray shaded cells.

Table 322. ANSLP values versus FRSIZE and sampling rates

FRSIZE	ANSLP values for Fs = 8 kHz			ANSLP values for Fs = 16 kHz		
	Slow(1)	typical(2)	Fast(3)	Slow(1)	typical(2)	Fast(3)
0 (8 samples)	0	1	2	-	0	1
1 (16 samples)	1	2	3	0	1	2
2 (32 samples)	2	3	4	1	2	3
3 (64 samples)	3	4	5	2	3	4
4 (128 samples)	4	5	6	3	4	5
5 (256 samples)	5	6	7	4	5	6
6 (512 samples)	6	7	-	5	6	7

1. The slow slope is equal to - 8.5 dB/s for the negative slope and + 2.1 dB/s for the positive slope.
2. The typical slope is equal to - 17.1 dB/s for the negative slope and + 4.2 dB/s for the positive slope.
3. The fast slope is equal to - 34.2 dB/s for the negative slope and + 8.5 dB/s for the positive slope.

The slopes can also be computed using the following formulas:

$$SLC_{UP} = 20 \times \frac{F_s}{F_{SIZE}} \times \log 10(1 + 2^{(ANSLP - 12)})$$

$$SLC_{DN} = 20 \times \frac{F_s}{F_{SIZE}} \times \log 10(1 - 2^{(ANSLP - 10)})$$

where F_S is the sampling rate of the stream observed by the SAD and F_{SIZE} is the frame size defined by FRSIZE[2:0].

Threshold computation

The way the threshold value is computed depends on SADMOD[1:0]:

- If SADMOD[1:0] = 0, THRH is obtained by multiplying the current ANLVL value with the gain defined in SNTHR[3:0].

  $$THR_H = ANLVL \times 10^{\frac{GdB_{SNTHR}}{20}}$$

  This threshold value is then compared to the current sound level (SDLVL).
- If SADMOD[1:0] = 01, THRH is obtained by multiplying the current ANMIN[12:0] with the gain defined by SNTHR[3:0].

  $$THR_H = ANMIN \times 10^{\frac{GdB_{SNTHR}}{20}}$$

  This threshold value is then compared to the current sound level (SDLVL).
- If SADMOD[1:0] = 1x, THRH is obtained by multiplying the current ANMIN[12:0] by 4.

$$THR_H = ANMIN \times 4$$

This threshold value is then compared to:

$$ANLVL \times 10^{\frac{GdB_{SNTHR}}{20}}$$

The hysteresis mode can be enabled to reduce the spurious transitions between MONITOR and DETECT states. In hysteresis mode (HYSTEN = 1), the following threshold values are used:

- THRH when the SAD is in MONITOR state.
- THRL when the SAD is in DETECT state.

The table below shows the thresholds values according to SNTHR.

Table 323. Threshold values according SNTHR(1)

SNTHR[3:0]	THR_H		THR_L		Comments
0	LVL + 3.5 dB	LVL x 1.5	LVL + 3.5 dB	LVL x 1.5	No hysteresis
1	LVL + 6.0 dB	LVL x 2	LVL + 3.5 dB	LVL x 1.5	Hysteresis of 2.5 dB
2	LVL + 9.5 dB	LVL x 3	LVL + 6.0 dB	LVL x 2	Hysteresis of 3.5 dB
3	LVL + 12.0 dB	LVL x 4	LVL + 9.5 dB	LVL x 3	Hysteresis of 2.5 dB
4	LVL + 15.6 dB	LVL x 6	LVL + 12.0 dB	LVL x 4	Hysteresis of 3.5 dB
5	LVL + 18.1 dB	LVL x 8	LVL + 15.6 dB	LVL x 6	Hysteresis of 2.5 dB
6	LVL + 21.6 dB	LVL x 12	LVL + 18.1 dB	LVL x 8	Hysteresis of 3.5 dB
7	LVL + 24.1 dB	LVL x 16	LVL + 21.6 dB	LVL x 12	Hysteresis of 2.5 dB
8	LVL + 27.6 dB	LVL x 24	LVL + 24.1dB	LVL x 16	Hysteresis of 3.5 dB
9	LVL + 30.1 dB	LVL x 32	LVL + 27.6dB	LVL x 24	Hysteresis of 2.5 dB

1. LVL must be replaced by ANLVL when SADMOD[1:0] = 0 and by ANMIN for other SADMOD[1:0] values.

When the hysteresis function is disabled, the SAD always use THRH.

Note: The hysteresis mode must not be used when SADMOD[1:0] = 1x.

Trigger logic

The signal compared to this threshold depends also on SADMOD[1:0].

The trigger condition is reached when the selected signal (SELSIG) is bigger than the threshold level.

If the trigger condition is met, the following happens:

- The SAD switches to DETECT state.
- The SAD refreshes the hangover counter with HGOVR.
- The sddet_evt event is asserted if the SAD transits from MONITOR to DETECT.
- The adf_sad_det signal is set to high.

The SAD remains in DETECT state as long as the trigger condition is met or the hangover down-counter is different from 0.

The sddet_evt event indicates when the SAD enters and/or exits the DETECT state. This event is used to generate an interrupt when a sound is detected or when a sound is no longer detected:

- When DETCFG = 0, the application receives an event only when the SAD enters the DETECT state.
- When DETCFG = 1, the application receives an event when the SAD enters or exits the DETECT state.

The adf_sad_det signal remains high as long as the SAD is in DETECT state.

The SAD also provides a flag indicating that a new sound level value is available (SDLVLF). The last computed sound level (SDLVL[14:0]) is available in the *ADF SAD sound level register (ADF_SADSDLVR)*, and the last computed ambient noise level (ANLVL[14:0]), in the *ADF SAD ambient noise level register (ADF_SADANLVR)*.

Note: The SAD can work even when the AHB clock is not present. In that case, the SAD does not update SDLVL[14:0] and ANLVL[14:0].

To get the latest valid SDLVL[14:0] and ANLVL[14:0] values, the application must read the ADF_SADSDLVR, and ADF_SADANLVR registers, when the SDLVLF flag goes high. This can be done in the following ways:

- by polling the SDLVLF flag:
  a) Clear the SDLVLF flag by writing SDLVLF to 1.
  b) Wait for SDLVLF = 1, by reading ADF_DFLTxISR.
  c) Read ADF_SADSDLVR and ADF_SADANLVR.
  d) Clear SDLVLF by writing it to 1.
  e) Go to step 2 if other values must to be read.
- by generating an interrupt:
  a) Read ADF_DFLTxISR.
  b) If SDLVLF = 1, read ADF_SADSDLVR and ADF_SADANLVR, and clear SDLVLF by writing it to 1.
  c) Handle other status flags and exit from ISR.

Sample transfer to memory

The SAD offers the following options to control the samples transfer from DFLT0 to the system memory:

- If DATCAP[1:0] = 1x, the samples are transferred into the system memory as soon as DFLT0 and SAD are enabled. The transfer does not depend on the SAD state.
- If DATCAP[1:0] = 01, the samples are transferred into the system memory when the SAD detects a sound (when the SAD is in DETECT state), assuming that DFLT0 and SAD are enabled.
- If DATCAP[1:0] = 0, the samples are not transferred into the memory. This mode can be used if the application only wants to observe but does not need samples for other processing.

Note: DATCAP[1:0] is taken into account only when the SADEN = 1. For example, if the SAD configuration is DATCAP[1:0] = 0, SADEN = DFLTEN = 1, and if the application sets now SADEN to 0, the samples provided by the DFLT0 are transferred to the RXFIFO.

Programming recommendations

To make the SAD function working properly, the ADF must be programmed as follows:

1. Provide the proper kernel clock (adf_ker_ck) to the ADF.
2. Configure the CKGEN and enable it.
3. Configure the SITF and enable it (note that microphones have a settling time of several milliseconds).
4. Configure the DFLT0. A typical setting is the following:
   – CIC5 with a decimation ratio of 12, 16 or 24
   – RSFLT with a decimation ratio of 4
   – HPF with HPFC = 2 or 3

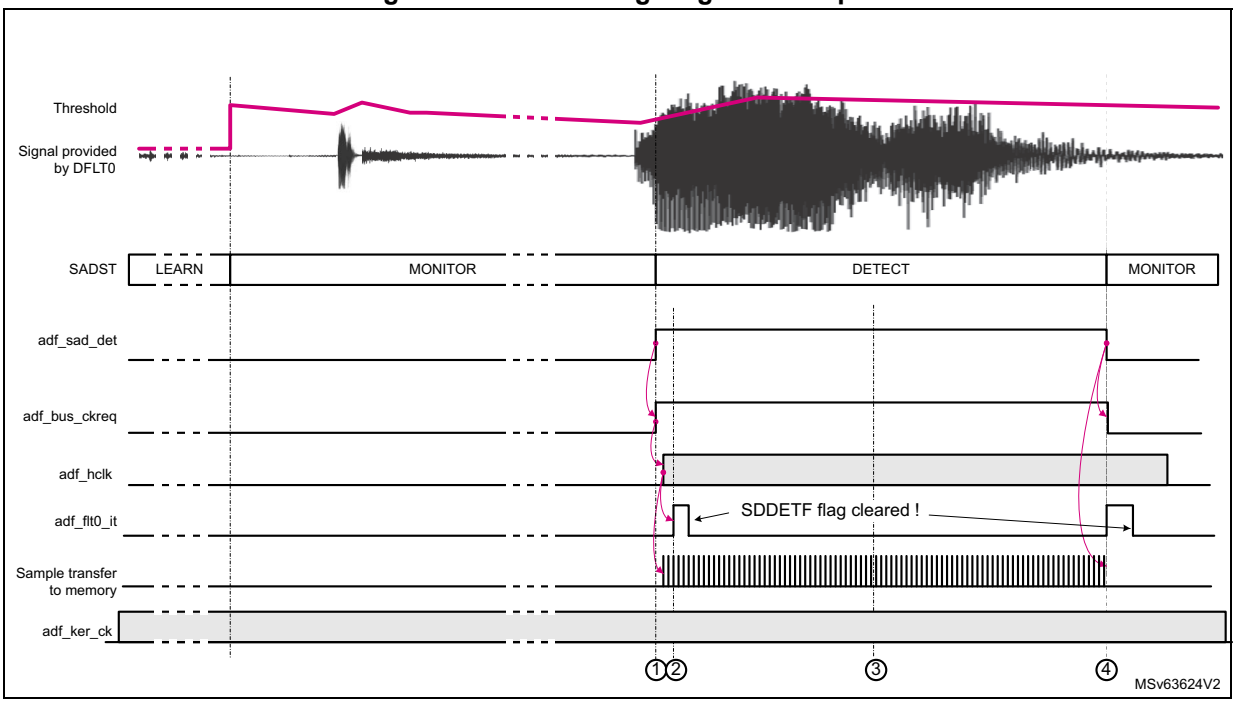
   For a very-low power implementation, the RSFLT can be bypassed.
5. Set SADEN to 0.
6. Wait for SADACTIVE = 0. If SADEN was previously enabled, this phase can take two periods of adf_hclk, and two periods of adf_proc_ck.
7. Configure the SAD as follows:
   – Set DATCAP[1:0] to 0, if the application does not want to store the samples into the system memory.
   – Set DATCAP[1:0] to 01, if the application wants to store the samples into the system memory only when the SAD detects a sound.
   – Set DATCAP[1:0] to 11, if the application wants to store the samples into the system memory continuously.
8. Configure the DMA (optional).
9. Enable the SAD.
10. Enable the DFLT0.

Figure 290 shows a simplified timing diagram when the SAD works with DATCAP[1:0] = 01.

Thanks to the kernel clock (adf_ker_ck), the SAD continuously monitors the audio signal provided by the DFLT0. The threshold is also continuously updated according to the ambient noise level estimation.

- When the SAD detects a sound higher than the programmed threshold (1), the ADF requests the bus clock (adf_bus_ckreq asserted).
- When the bus clock is available (see 2 in *Figure 290*) then:
  – The data transfer to the memory is triggered.
  – The event interrupt (adf_evt_it) can be generated.
- In this example, the event interrupt (adf_evt_it) is used to wake up the application. The interrupt line is released by clearing SDDETF by writing 1 to it.
- As long as the SAD remains in DETECT state, the application waits to get enough samples and calls, for example the keyword recognition algorithm (see 3 in *Figure 290*).
- In the case shown in the figure below, the SAD state (SADST) goes back to MONITOR before the keyword is recognized. If DETCFG is set to 1, an event signals when the SAD goes back to MONITOR state. The SAD stops the transfer of samples into the memory and the application can clean up the receive buffer for the next detection (see 4 in *Figure 290*).

Figure 290. SAD timing diagram example



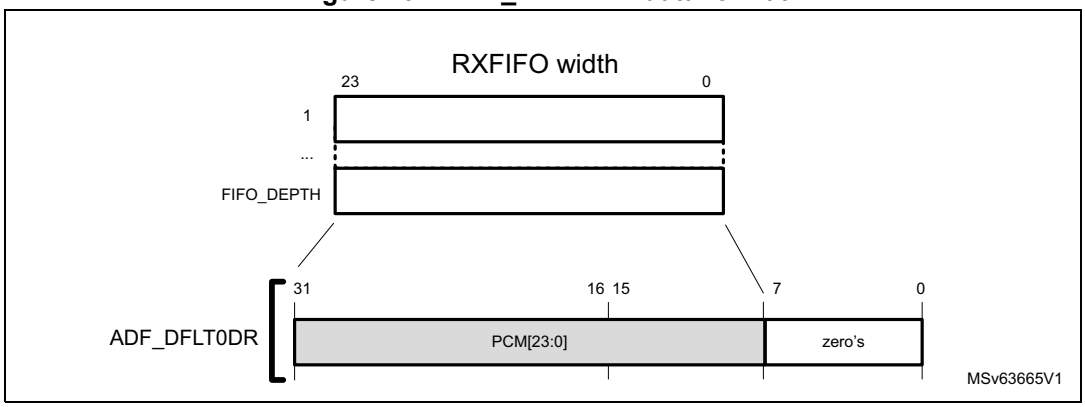
36.4.11 Data transfer to memory

Data format

The samples processed by DFLT0 are stored into a RXFIFO. The application can read the samples stored into these FIFOs via the *ADF digital filter data register 0 (ADF_DFLT0DR)*. The samples inside this register are signed and left aligned. The bit 31 always represents the sign.

The ADF provides 24-bit left-aligned data. Performing a 16-bit access to ADF_DFLT0DR allows the application to get the 16 most significant bits. Performing a 32-bit access to ADF_DFLT0DR allows the application to get a 24-bit data size.

Figure 291. ADF_DFLTxDR data format



Data re-synchronization

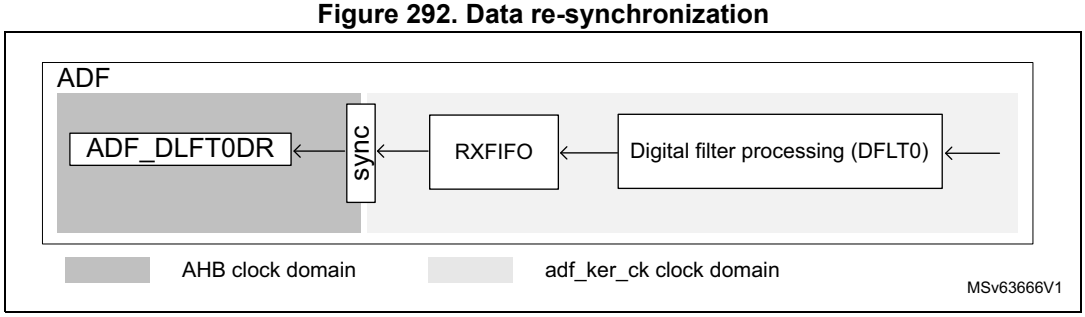
The samples stored into the RXFIFO can be transferred into the memory by using either DMA requests or interrupt signaling.

Note: The RXFIFO is located into the adf_ker_ck clock domain, while ADF_DFLT0DR *is located into the adf_hclk (AHB) clock domain.*

When the AHB clock is available, if ADF_DFLT0DR is empty and if a sample is available into the RXFIFO, this sample is transferred into ADF_DFLT0DR.

The sample transfer from the RXFIFO to ADF_DFLT0DR takes two periods of the AHB clock (adf_hclk) and two periods of the adf_ker_ck clock. The ADF inserts automatically wait-states if the application performs a read operation of ADF_DFLT0DR while the transfer of the new sample from the RXFIFO to ADF_DFLT0DR is not yet completed.

Figure 292. Data re-synchronization



Data transfer

The content of the RXFIFO can be transfered to the memory either by using a DMA channel or interrupt services.

Both single and burst, DMA transfers are supported by the ADF, but the application has to care about the following points:

- The RXFIFO must contain at least the same amount of samples than the burst size.
- The burst mode efficiency may be reduced due to the data re-synchronization explained in the previous section.

Note: The burst mode is not available in all products (see the DMA section to check if the product supports it).

In addition, the application can select the RXFIFO threshold (FTH bit) in order to trigger the data transfer: a data transfer can be triggered as soon as the RXFIFO is not empty, or when the RXFIFO is half-full (containing depth/2 samples).

For the DMA transfer, as soon as one of the RXFIFO reaches the threshold level, the DMA request is asserted in order to ask for data transfer. Successive DMA requests are performed as long as the RXFIFO is not empty.

The DMA mode of the RXFIFO is enabled via the DMAEN bit in ADF_DFLT0DR.

For the interrupt signaling, the following cases must be considered:

- If FTH = 0, as soon as a data is available in ADF_DFLT0DR, the FTHF is set, allowing the generation of an interrupt. FTHF is released as soon as ADF_DFLT0DR is read.
- If FTH = 1, as soon as the RXFIFO reaches the threshold level and a data is available in ADF_DFLT0DR, the FTHF is set, allowing the generation of an interrupt. FTHF is released as soon as one data is read. FTHF is set again if the threshold condition is

met again. In this mode, every time an interrupt occurs, the application is supposed to read FIFO_SIZE/2 data.

RXFIFO Overrun

A RXFIFO overrun condition is detected when the RXFIFO is full, and a new sample from the DFLT0 must be written.

In this case, DOVRF is set and the new sample is dropped. When the RXFIFO has at least one location available, the new incoming sample is written into the RXFIFO.

Figure 293 shows an example based on a RXFIFO depth of four words and FTH set to 1, so that FTHF goes to 1 when the RXFIFO is half-full.

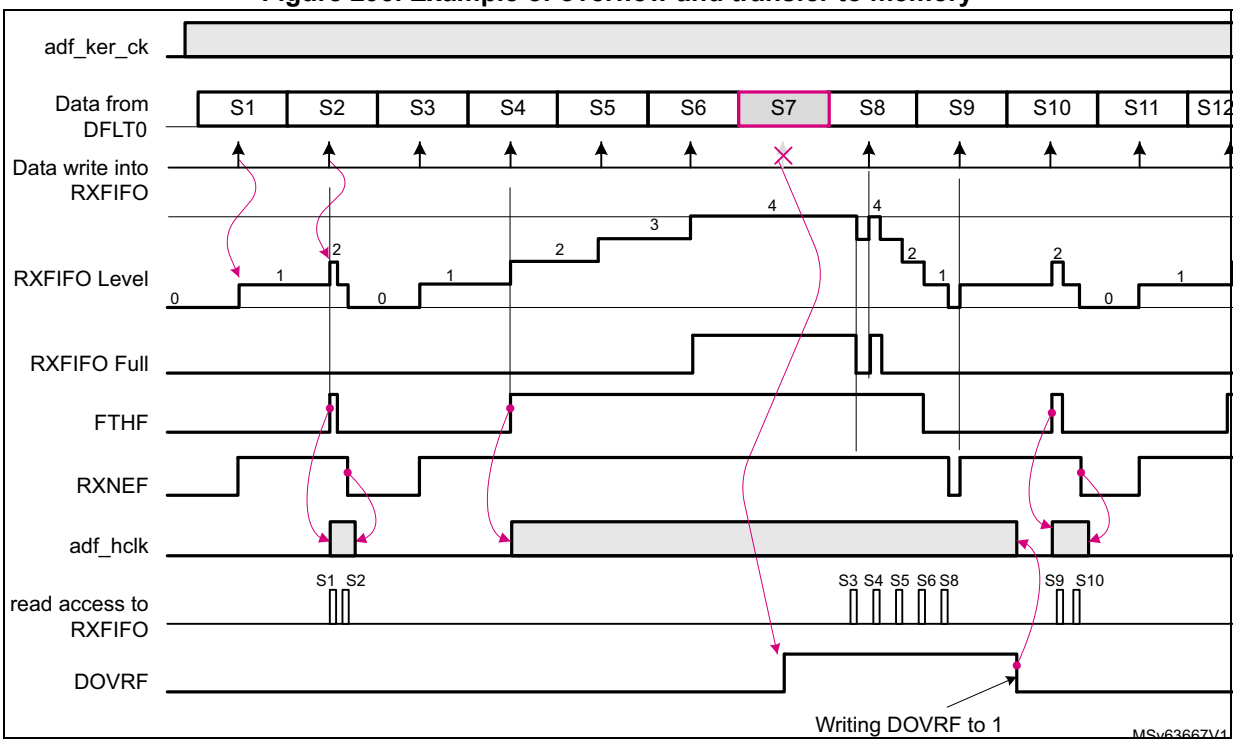
The S7 sample is lost due to an overrun: the RXFIFO is full while S7 must be written into the RXFIFO. The S7 write operation is not performed. DOVRF is set to 1 at the moment where the write operation was expected. The overflow event remains to 1 as long as it is not cleared by the application.

In this example, DOVRIE is set to 1 to have an interrupt if an overrun condition is detected.

After the S7 sample, the application manages to read data from the RXFIFO, and the ADF can write the S8 sample and consecutive. Later, the application clears DOVR, allowing the detection of a new overrun situation.

In the adf_hclk line, the gray boxes indicate that the ADF requested the AHB clock. The figure below shows the AHB clock available only when the ADF requests it. In real applications, the AHB clock may also be present if the ADF does not request it.

Figure 293. Example of overflow and transfer to memory



36.4.12 Autonomous mode

The ADF can work even if the AHB bus clock is not available (Stop modes). The ADF uses the AHB clock only for the register interface. All the processing part is clocked with the kernel clock.

In Stop mode, the ADF receives a kernel clock if the following conditions are met:

- The ADF autonomous mode is enabled in the RCC.
- The selected kernel clock source is taken from an oscillator available in Stop mode.

In Stop mode, the ADF receives the AHB clock if the following conditions are met:

- The ADF autonomous mode is enabled in the RCC.
- The ADF requests the AHB clock in the following situations:
  - when the ADF must transfer data into memory via the DMA

    The data is directly transferred to the SRAM thanks to the DMA while the product remains in Stop mode. The AHB clock request is maintained until the DMA transfer is completed.
  - when the ADF needs to generate an interrupt

    An interrupt generally wakes up the device from Stop mode, as an action from the application is needed. Once the AHB clock is available, the interrupt is generated.The AHB clock request is maintained as long as an enabled interrupt flag is still active.

36.4.13 Register protection

The ADF embeds some hardware protection to prevent invalid situations. The table below shows the list of write-protected and unprotected fields.

Table 324. Register protection summary

Registers	Unprotected fields	Write-protected fields	Write-protection condition
ADF global control register (ADF_GCR)	TRGO	-	DFLTACTIVE0 = 1
ADF clock generator control register (ADF_CKGCR)	CKGDEN CCK0EN CCK1EN	PROCDIV[6:0], CCKDIV[3:0], CKGMOD, TRGSRC[3:0], TRGSENS, CCK[1:0]DIR	CKGACTIVE = 1
ADF serial interface control register 0 (ADF_SITF0CR)	SITFEN	STH[4:0], SITFMOD[1:0], SCKSRC[1:0]	SITFACTIVEx = 1
ADF bitstream matrix control register 0 (ADF_BSMX0CR)	-	BSSEL[4:0]	DFLTACTIVEx = 1
ADF digital filter control register 0 (ADF_DFLT0CR)	DFLTEN	NBDIS[7:0], TRGSRC[3:0], TRGSENS, FTH, DMAEN, SNPSFMT, ACQMOD[2:0]	DFLTACTIVEx = 1
ADF digital filer configuration register 0 (ADF_DFLT0CICR)	SCALE[5:0]	MCICD[8:0], CICMOD[2:0], DATSRC[1:0]	
ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)	-	All fields	
ADF delay control register 0 (ADF_DLY0CR)	-	SKPDLY[6:0]	SKPBF = 1

Table 324. Register protection summary (continued)

Registers	Unprotected fields	Write-protected fields	Write-protection condition
ADF DFLT0 interrupt enable register (ADF_DFLT0IER)	All fields	-	-
ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR)	All fields	-	-
ADF SAD control register (ADF_SADCR)	SADEN	FRSIZE[2:0], DETCFG, SADMOD[1:0], HYSTEN, DATCAP[1:0]	SADACTIVE = 1
ADF SAD configuration register (ADF_SADCFGR)	-	ANMIN[12:0], SNTHR[3:0], HGOVR[3:0], LFRNB[2:0], ANSLP[2:0]	SADACTIVE = 1

All the ADF processing is performed in the adf_proc_ck clock domain. For that reason, enabling or disabling an ADF sub-block may take some time due to the re-synchronization between the AHB clock domain and the adf_proc_ck clock domain. XXXACTIVE flags are available to allow the application to check that the synchronization between the two clock domains is completed.

To change a write-protected field, the application must follow this sequence:

1. Set the enable bit of the sub-block to 0.
2. Wait for corresponding flag XXXACTIVE = 0.
3. Modify the wanted fields.
4. Set the enable bit of the sub-block to 1.

Refer to the description of each sub-block for details.

36.5 ADF low-power modes

Table 325. Effect of low-power modes on ADF

Mode	Description
Sleep	No effect. ADF interrupts cause the device to exit the Sleep mode.
Stop(1)	The ADF registers content is kept. If the autonomous mode is enabled in the RCC and the ADF is clocked by an internal oscillator available in Stop mode, the ADF remains active. The DMA requests are functional and the interrupts in these modes cause the device to exit Stop mode.
Standby	The ADF is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 36.3: ADF implementation* for details about Stop modes supported by the ADF.

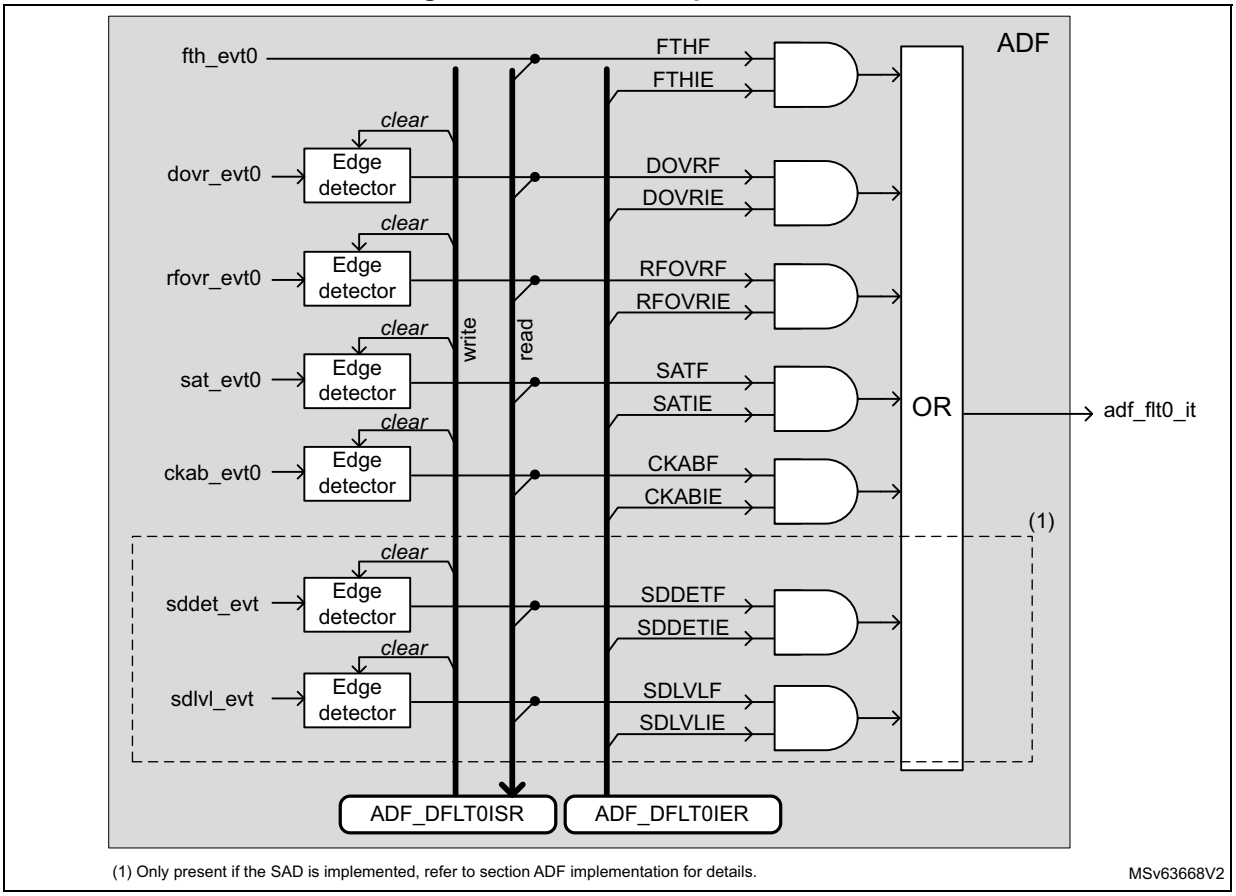
36.6 ADF interrupts

To increase the CPU performance, the ADF offers an interrupt line (adf_flt0_it), sensitive to several events.

Note: The status flags are available even if the corresponding interrupt enable flag is not enabled.

The interrupt interface is controlled via the *ADF DFLT0 interrupt enable register (ADF_DFLT0IER)* and the *ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR)*.

Figure 294. ADF interrupt interface



The table below shows which interrupt line is affected by which event, and how to clear and activate each interrupt/event.

Table 326. ADF interrupt requests

Interrupt vector	Interrupt event	Event flag	Event/interrupt clearing method	Exit Sleep mode	Exit Stop modes(1)	Exit Standby mode
ADF_FLT0(2)	RXFIFO threshold reached	FTHF	Read ADF_DFLT0DR until RXFIFO level is lower than the threshold.	Yes	Yes	No
	RXFIFO overrun	DOVRF	Write DOVRF to 1.			
	RSFLT overrun	RFOVRF	Write RFOVRF to 1.			
	Saturation detection	SATF	Write SATF to 1.			
	Channel clock absence detection	CKABF	Write CKABF to 1.			
	SAD: sound detected	SDDETF	Write SDDETF to 1.			
	SAD: sound level value available	SDLVLF	Write SDLVLF to 1.			

1. Refer to *Section 36.3: ADF implementation* for details.
2. ADF_FLT0 vector corresponds to the assertion of adf_flt0_it signal.

36.7 ADF application informations

36.7.1 ADF configuration examples for audio capture

Table 327 gives some examples of the ADF settings for the digital microphones, focusing on 16 and 48 kHz output data rate.

Configurations #1 and #2 are for very low-power use-cases and have a reduced signal-to-noise ratio. The user must also insure that the selected digital microphone can work properly at 512 kHz. These configurations can be used for sound detection. The RSFLT is not used to reduce as much as possible the frequency of the frequency of the kernel clock (adf_ker_ck).

Configurations #3, #4, #9, #10, #11 give signal-to-noise ratios around 115 dB, with an ideal microphone model, with a sinus signal of 997 Hz. Using the RSFLT allows a good control on the in-band ripple, and a good image rejection.

Configurations #7, #8, #10 give signal-to-noise ratio around 120 dB, with an ideal microphone model, using a sinus signal of 997 Hz.

Table 327. Examples of ADF settings for microphone capture

Configuration	adf_ker_ck (MHz)	PROCDIV + 1	CCKDIV + 1	CIC order (1)	MCICD + 1	SCALE	RSFLTBYP	RSFLTD	HPFBYP	-	adf_proc_ck (MHz)	Total dec. ratio	F_{RS} (kHz)	F_{ADF_CCKx} (MHz)	F_{PCM} (kHz)
#1	1.024	1	2	4	64	0x2D (- 8.5 dB)	1	x	x	=>	1.024	64	-	0.512	8
#2		1	2	5	32	0x2B (- 14.5 dB)						32	-		16
#3		1	2	5	16	0x01 (+ 3.5 dB)	0	0				64	32		8
#4	2.048	1	2	5	16	0x01 (+ 3.5 dB)	0	0			2.048	64	64	1.024	16
#5	3.072	1	6	5	8	0x0B (+ 33.6 dB)	0	0			3.072	32	32	0.512	8
#6		2	2	5	12	0x06 (+ 18.1 dB)						48	64	0.768	16
#7		1	2	5	24	0x2C (- 12 dB)						96		1.536	
#8	4.096	1	2	5	32	0x27 (- 26.6 dB)	0	0			4.096	128	64	2.048	16
#9	6.144	3	2	5	16	0x02 (+ 6.0 dB)	0	0			2.048	64	64	1.024	16
#10		2	2	5	24	0x2C (- 12 dB)					3.072	96		1.536	
#11		1	2	5	16	0x01 (+ 3.5 dB)					6.144	64	192	3.072	48
#12	7.680	1	2	5	20	0x2E (- 6.0 dB)	0	0			7.680	80	192	3.840	48

1. CICMOD = 100 for CIC order equal to 4. CICMOD = 101 for CIC order equal to 5.

36.7.2 Programming examples

Example 1

This example describes the programing of ADF for the capture of a signal coming from a digital microphone, using only the CIC4, with a decimation of 48, assuming that the kernel clock is 1.536 MHz. Typically, this configuration can be used to detect sound using the SAD.

Table 328. Programming sequence (CIC4)

Operations	Comments
Adjust the proper kernel clock frequency via the RCC	Assuming that the RCC is programmed to provide a kernel clock (adf_ker_ck) of 1.536 MHz coming from a RC oscillator.
Select the proper ADF kernel clock source via the RCC	Refer to the RCC of the product.
Enable the ADF clocks via the RCC	Refer to the RCC of the product.
Reset the ADF via the RCC	Refer to the RCC of the product.
AFMUX programming	Program the AFMUX to select ADF_SD0 and ADF_CCK0 functions.
Enable ADF processing clock: ADF_CKGCR = 0x0001 0023	PROCDIV = 0 (bypass): adf_proc_ck frequency is 1.536 MHz. CCKDIV = 1 (division by 2): ADF_CCK0 clock frequency is 768 kHz. The ADF_CCK0 pin is set in output and generates a clock so that the microphone can exit from low-power mode.
Serial interfaces configuration: ADF_SITF0CR = 0x0000 1F01	SCKSRC = 0 to select ADF_CCK0 as serial clock. SIFTMOD = 0 to select LF_MASTER SPI mode. Clock absence feature is not working in this mode. The serial interface is enabled.
Bitstream matrix configuration: ADF_BSMX0CR = 0x0000 0000	DFLT0 filter takes the bitstream of SITF0, sampled on rising edge
Filters configuration (CIC): ADF_DFLT0CICR = 0x0040 2F40	SCALE = 0x04 (12 dB): RSFLT is not enabled. Note that the gain is 8.5 dB, higher than recommended in order to improve signal accuracy for the SAD. Saturation is not an issue in this case, as only the detection of a signal much lower than the full scale is needed. MCICD = 0x2F (decimation by 48) CICMOD = 4 to select a $Sinc^4$ DATSCR = 0 to select data coming from BSMX
Filters configuration (RSFLT and HPF): ADF_DFLT0RSFR = 0x0000 0301	HPFC = 3: cut-off frequency of 16kHz * 0.0095 = 152 Hz HPFBYP = 0: HPF not bypassed RSFLTBYP = 1: RSFLT bypassed
Micro delay adjust: ADF_DLY0CR = 0x0000 0000	Not used in this example
Enable interrupt events: ADF_DFLT0IER = 0x0000 1000	Enable the interrupt events the application wants to handle. In this example, SDDETIE is set to 1 to have an interrupt if a sound is detected.

Table 328. Programming sequence (CIC4) (continued)

Operations	Comments
SAD setting: ADF_SADCR = 0x0000 1400	SADMOD = 1: Trigger based on sound level values FRSIZE = 4: Sound level computed on 128 samples HYSTEN = 0: No hysteresis DETCFG = 0: interrupt generated when the SAD enters in DETECT state DATCAP = 0: samples not transferred to memory SADEN = 0: SAD not yet enabled
SAD setting: ADF_SADCFGR = 0x0025 0001	ANMIN = 0x25: Sound threshold of 37 * 10^SNTHR/20 = 74 LSB (1) HGOVR = 0: four frames (4 x 128 = 512 samples) LFRNB = 0: Learning phase set to two frames. Not useful in this mode. ANSLP = 0: Not used in this mode SNTHR = 1: Threshold set to 6 dB (see ANMIN)
Clear status flags: ADF_DFLT0ISR = 0x0000 0FFF	Clear all the status flags before running the ADF
Enable the SAD: ADF_SADCR = 0x0000 1401 Wait for SADACTIVE flag = 1	Enable first the SAD, in order to be sure that data transfer to memory are blocked by the SAD.
Enable the filter: ADF_DFLT0CR = 0x0000 0001	NBDIS = 0: no samples discarded ACQMOD = 0: Asynchronous continuous acquisition mode DMAEN = 0: DMA interface not used DFLTEN = 1: Filter enabled

1. The SNTHR and ANMIN values are computed using the same approach than in *Threshold programming with SADMOD = 01* (detection of a sound higher than 63 dBSPL).

Example 2

This example describes the programing of ADF for the capture of a signal coming from a digital microphone, using the CIC5, and the RSFLT, with a total decimation of 64.

Table 329. Programming sequence (CIC5)

Operations	Comments
Adjust the proper kernel clock frequency via the RCC	Assuming that the RCC is programmed to provide a kernel clock (adf_ker_ck) of 6.144 MHz
Select the proper ADF kernel clock source via the RCC	Refer to the RCC of the product.
Enable the ADF clocks via the RCC	Refer to the RCC of the product.
Reset the ADF via the RCC	Refer to the RCC of the product.
AFMUX programming	Program the AFMUX to select ADF_SD0 and ADF_CCK0 functions.
Enable ADF processing clock: ADF_CKGCR = 0x0201 0023	PROCDIV = 2 (division by 3): adf_proc_ck frequency is 6.144 MHz. CCKDIV = 1 (division by 2): ADF_CCK0 clock frequency is 1.024 MHz. The ADF_CCK0 pin is set in output and generates a clock so that the microphone can exit from low-power mode.

Table 329. Programming sequence (CIC5) (continued)

Operations	Comments
Serial interfaces configuration: ADF_SITF0CR = 0x0000 1F01	SCKSRC = 0 to select ADF_CCK0 as serial clock. SIFTMOD = 0 to select LF_MASTER SIPI mode. Clock absence feature is not working in this mode. The serial interface is enabled.
Bitstream matrix configuration: ADF_BSMX0CR = 0x0000 0000	DFLT0 filter takes the bitstream of SITF0, sampled on rising edge
Digital filter control: ADF_DFLT0CR = 0x0000 0006	NBDIS = 0: no samples discarded ACQMOD = 0: Asynchronous continuous acquisition mode DMAEN = 1: DMA interface not used DFLTEN = 0: Filter disabled FTH = 1: RXFIFO half-full
Filters configuration (CIC): ADF_DFLT0CICR = 0x0010 0F50	SCALE = 0x01(+3.5 dB): Output limited to 22 bits due to RSFLT. MCICD = 0xF (decimation by 16) CICMOD = 5 to select a $Sinc^5$ DATSCR = 0 to select data coming from BSMX
Filters configuration (RSFLT and HPF): ADF_DFLT0RSFR = 0x0000 0200	HPFC = 2: cut-off frequency of 16kHz * 0.0025 = 40 Hz HPFBYP = 0: HPF not bypassed RSFLTBYP = 0: RSFLT not bypassed RSFLTD = 0: Decimation by 4
Micro delay adjust: ADF_DLY0CR = 0x0000 0000	Not used in this example
Enable interrupt events: ADF_DFLT0IER = x	Enable the interrupt events the application wants to handle.
Clear status flags: ADF_DFLT0ISR = 0x0000 0FFF	Clear all the status flags before running the ADF
Program the DMA Enable the DMA	The DMA must be programmed in order to read the data inside the ADF_DFLT0DR register every time a DMA request is generated.
Start acquisition: ADF_DFLT0CR = 0x0000 0007	The ADF starts to filter data.

36.7.3 Connection examples

Figure 295 shows simple connection examples of the ADF to external sensors.

- Picture on the left: two digital microphones connected to the ADF

  In this connection, the amount of connections is optimized; DMIC1 and DMIC2 are sharing the same data line, and the same clock line. BSMX allows the application to connect the digital filter either to DMIC1 or to DMIC2. In this configuration when one of the microphone is used, the other is activated as well, as they share the same clock.
- Picture in the center: two digital microphones connected to the ADF

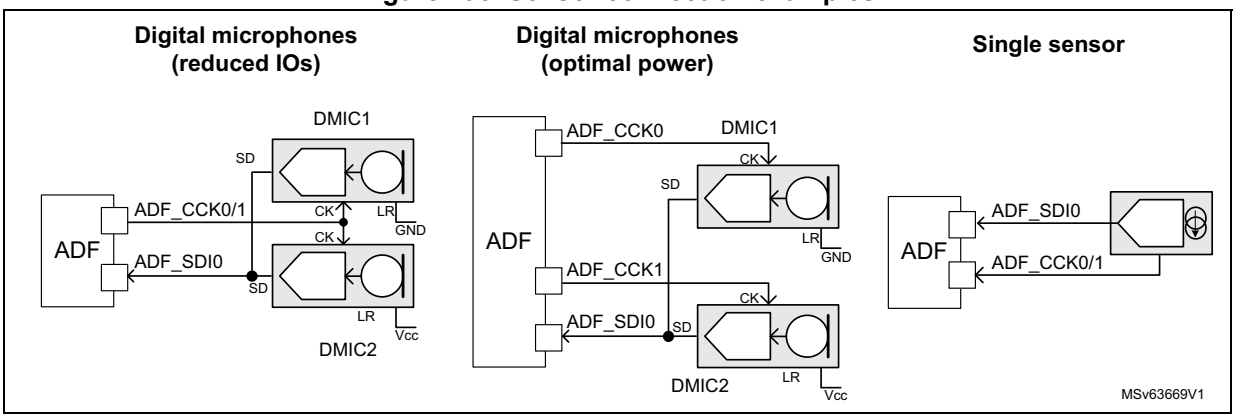
  In this connection, DMIC1 and DMIC2 are sharing the same data line, but have a dedicated clock line. BSMX allows the application to connect the filter either to DMIC1

or to DMIC2. When the application wants to use a microphone, it is possible to keep the other in low-power mode by forcing its clock line to 0 (CCKyEN = 0, CCKyDIR = 1).

- Picture on the right: single sensor connected to the ADF

  It is also possible to configure the CCK0 and CCK1 pins to input in order to connect sensors providing the clock.

Figure 295. Sensor connection examples



36.7.4 Global frequency response

Figure 296 shows the global frequency response for a 16 kHz audio signal with a digital microphone working at 1.024 MHz. The filter configuration is the following:

- CIC order 4 or 5, with a decimation ratio of 16
- RSFLT enabled, with a decimation ratio of 4
- HPF enabled with a cut-off frequency of 40 Hz

The figure below shows the theoretical frequency response using a CIC4 and a CIC5.

Figure 296. Global frequency response

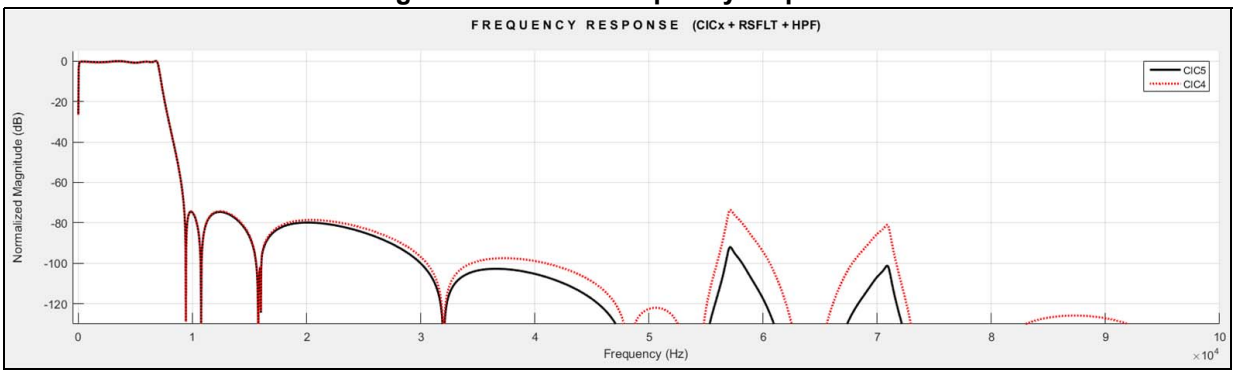


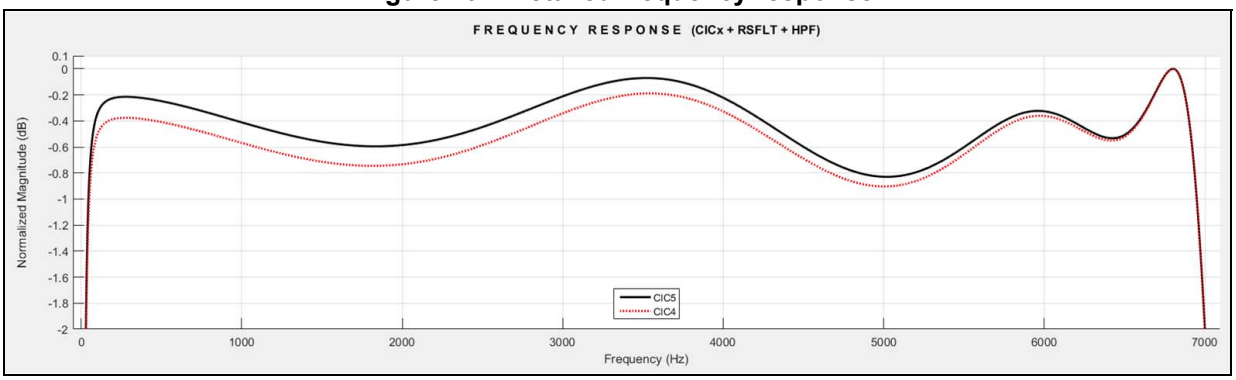
Figure 297 shows the in-band ripple for a 16 kHz audio signal with a digital microphone working at 1.024 MHz. The filter configuration is the following:

- CIC order 4 or 5, with a decimation ratio of 16
- RSFLT enabled, with a decimation ratio of 4
- HPF enabled with a cut-off frequency of 20 Hz

The resulting in-band ripple is ± 0.41 dB for CIC5, and ± 0.45 dB for CIC4.

The - 3 dB cut-off frequency is 7061 Hz.

Figure 297. Detailed frequency response



36.7.5 Total ADF gain

This section details how to compute the signal level provided by the ADF according to the filter settings.

A signal level may be expressed in dBFS (decibel full scale). A 0 dBFS level is assigned to the maximum possible digital level. For example, a signal that reaches 50 % of the maximum level, has a − 6 dBFS level (6 dB below full scale).

For example, for the ADF offering a final data width of 24 bits, a signal having an amplitude of 2 * 10^6 LSB has a level of:

$$20 \times \log_{10}\left(\frac{2 \times 10^6}{2^{(24-1)}}\right) = -12.45 \text{ dBFS}$$

In addition, the data size of a signal having an amplitude (Amp) expressed in LSB is given by:

$$DS = \left(\frac{\ln(Amp)}{\ln(2)} + 1\right) \text{ bits}$$

One bit need to be added for negative values.

So a signal having an amplitude of 2 * 10^6 LSB, has a data size of 21.9 bits.

CIC gain

The CIC gain (G_{CIC} and GdB_{CIC}) can be deduced from the following formula giving data size in bits (DS_{CIC}).

$$DS_{CIC} = (N \times \log_2(D1)) + DSin$$

where N represents the CIC order (selected by CICMOD[2:0]), and D1 is the decimation ratio (given by MCICD[8:0]).

DSin represents the data size (in bits) of the signal at CIC input.

Warning: DS_{CIC} is very important for CIC filter. In order to work fine, DS_{CIC} must not exceed 26 bits.

The CIC gain G_{CIC} is given by:

$$G_{CIC} = (D1)^N$$

which gives in decibels:

$$GdB_{CIC} = 20 \times \log_{10}((D1)^N)$$

Data size at SCALE output

The data size at SCALE output (including the CIC gain), is a key information as the RSFLT starts to have some saturations, if the peak-to-peak signal amplitude at SCALE output is higher than 22 bits.

If the RSFLT is bypassed, then a peak-to-peak signal amplitude of 24 bits is accepted.

The signal amplitude at SCALE output is:

$$Asout_{SCALE} = D1^N \times 10^{\frac{GdB_{SCALE}}{20}} \times Asin_{DFLT}$$

GdB_{SCALE} represents the gain selected by SCALE[5:0], in dB.

$Asout_{SCALE}$ is the signal amplitude at SCALE output (in LSB), and $Asin_{DFLT}$ is the signal amplitude at CIC input (LSB).

$$DS_{SCALE} = \frac{\ln(Asout_{SCALE})}{\ln(2)} + 1$$

The data size at SCALE output (DS_{SCALE}) is expressed in bits.

RSFLT gain

The RSFLT gain in the useful bandwidth is typically 9.5 dB, but due to ripple a margin of about ± 0.41 dB must be considered.

$$G_{RSFLT} = 10^{\frac{9.5\ dB}{20}} = 2.98 \text{ typical}$$

Note: The HPF filter has a gain of 0 dB.

SAD gain

The SAD is using only the 16 MSB on the signal, as a consequence, from the SAD point of view, the truncation from 24 to 16 bits can be seen as an attenuation.

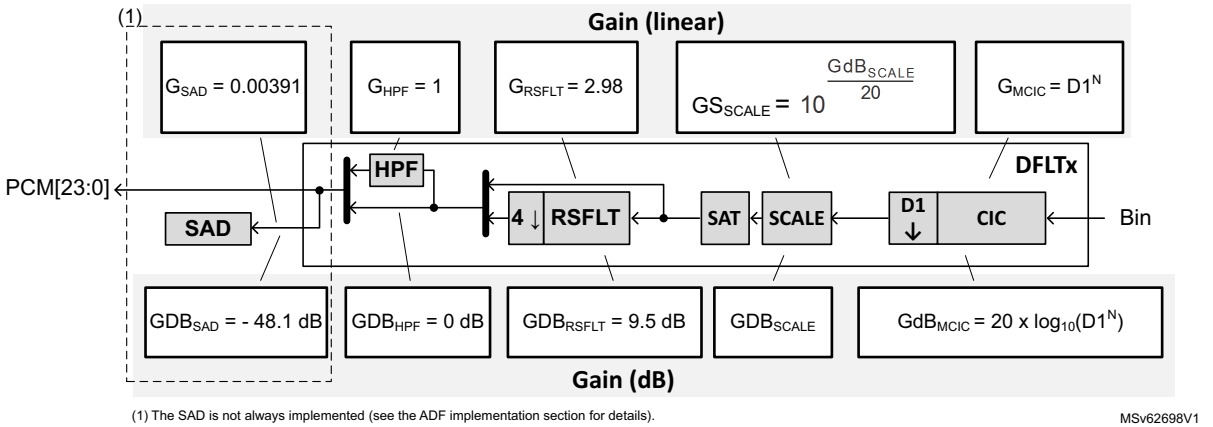
$$GdB_{SAD} = -48.1dB$$

and

$$G_{SAD} = 0.003906$$

The figure below shows a simplified view of the filter path and gives, for each significant component, the expression of the bit growth and the gain.

Figure 298. Simplified DFLT view with gain information



The table below summarizes of the final data size for different filter configurations.

Table 330. Output signal levels

Filter configurations	Final signal amplitude (LSB)
CIC + RSFLT + HPF + SAD	Samples provided to the RXFIFO: $Asout_{RXFIFO} = D1^N \times 10^{\frac{GdB_{SCALE}}{20}} \times 10^{\frac{9.5}{20}} \times Asin_{DFLT}$ Samples provided to the SAD: $Asout_{SAD} = D1^N \times 10^{\frac{GdB_{SCALE}}{20}} \times 10^{\frac{9.5}{20}} \times 0.003906 \times Asin_{DFLT}$ DS_{SCALE} must be lower than 22 bits

Table 330. Output signal levels (continued)

Filter configurations	Final signal amplitude (LSB)
CIC + RSFLT (+ HPF)	$ASout_{HPF} = D1^N \times 10^{\frac{GdB_{SCALE}}{20}} \times 10^{\frac{9.5}{20}} \times Asin_{DFLT}$ DS_{SCALE} must be lower than 22 bits
CIC (+ HPF)	$ASout_{HPF} = D1^N \times 10^{\frac{GdB_{SCALE}}{20}} \times Asin_{DFLT}$ DS_{SCALE} must be lower than 24 bits

Example using the main filter chain

If the ADF filter is programmed as follows:

- The input signal is coming from a serial interface ($Dsin_{RSFLT}$ = 1 bit).
- CIC order = 5 (N), with a decimation value of 24 (D1).
- SCALE[5:0] is set to - 12 dB.
- RSFLT enabled, and the decimation by four is enabled.
- HPF is enabled.

Check first the data size at CIC output:

$$DS_{CIC} = (5 \times \log_2(24)) + 1 \text{ bit} = 23.92 \text{ bits}$$

The size is lower than 26 bits, so the CIC works in good conditions.

The data size at CIC output is very close to 24 bits, so the SCALE must be adjusted in order to provide a 22-bit max signal to the RSFLT. An attenuation of 12 dB is needed.

Then the signal level provided to the RSFLT is:

$$Asout_{SCALE} = 24^5 \times 10^{\frac{-12}{20}} \times 1 = 2.10^6$$

$$DS_{SCALE} = \frac{\ln(2.10^6)}{\ln(2)} + 1 = 21.93 \text{ bits}$$

If a higher gain is used, the RSFLT may saturate the output signal for strong input signals.

At the end, the final signal amplitude is:

$$Asout_{HPF} = 24^5 \times 10^{\frac{-12}{20}} \times 10^{\frac{9.5}{20}} \times 1 = 5.9711 \times 10^6$$

$$Dsout_{HPF} = \left(\frac{\ln(5.9711 \times 10^6)}{\ln(2)} + 1\right) = 23.51 \text{ bits}$$

or:

$$SDB_{OUT} = 20 \times \log_{10}\left(\frac{2^{23.51}}{2^{24}}\right) = -2.84 \text{ dBFS}$$

36.7.6 How to compute SAD thresholds

The SAD does not compute the RMS value of the converted signal, but the average of the absolute values. As a consequence, the estimated level differs from the RMS value of the signal:

- For a sine signal having an RMS value of 1, the SAD computes a level of 0.9.
- For a white or pink noise signal having an RMS value of 1, the SAD computes a level of about 0.8.

Note: FRSIZE[2:0] has a big influence on the accuracy of the level estimation: big FRSIZE[2:0] values give better results.

Threshold programming with SADMOD = 01

Consider the case of a sound capture where the application wants to wake up the system when the captured sound is bigger than 63 dBSPL.

The sound capture can be performed with a digital microphone such as the MP45DT02.

The sensitivity of this microphone is typically -26 dBFS for an input signal of 94 dBSPL.

An acoustic signal at 63 dBSPL produces a digital signal of about:
- 26 dBFS - (94 - 63) = - 57 dBFS.

- SCALE value adjustment

  For this example, the filter configuration is the following:
  - CIC5 with a decimation by 16
  - RSFLT enabled with a decimation by 4
  - HPF enabled

  A SCALE value of 3.5 dB is recommended for this configuration. As DFTL0 provides samples only used for a sound detection (samples not provided to the application), a bigger gain value can be applied: it increases the SAD accuracy and a saturation does not affect the SAD behavior (for example, a SCALE value of 15.6 dB).
- Input signal amplitude

  The input signal is - 57 dBFS, corresponding to an amplitude

  Asin = $10^{(-57/20)}$ = 0.00141 LSB.
- Signal level at SAD input

  The total filter gain for the SAD is:

$$G_{SAD} = 16^5 \times 10^{\frac{15.6}{20}} \times 10^{\frac{9.5}{20}} \times \frac{1}{256} = 73.68 \times 10^3 \text{ or } 97.3 \text{ dB}$$

  The signal amplitude received by the SAD is:

$$Ain_{SAD} = 0.00141 \times G_{SAD} \sim 104 \text{ LSB}$$

The gain can be increased if the expected amplitude is too small. For the targeted application, 104 LSB is fine.

If the input signal is expected to be a sine, the sound level for a signal amplitude of 104 LSB is:

$$SDLVL = \frac{Ain_{SAD} \times \sqrt{2}}{2} \times 0.9 \sim 66 \text{ LSB}$$

where 0.9 is the correction factor to apply with respect to the RMS value.

- Program the trigger value

  ANMIN and SNTHR must be programmed to trigger the SAD when the input signal level reaches 66 LSB.

  For SADMOD[1:0] = 01, the threshold value is given by:

$$THRH = ANMIN \times 10^{\frac{GdB_{SNTHR}}{20}}$$

where GdB_{SNTHR} represents the decibel value selected by SNTHR[3:0].

When SNTHR[3:0] = 6 dB for example, this formula becomes:

$$THRH = 2 \times ANMIN$$

So ANMIN = THRH/2 = 66 /2 = 33 LSB.

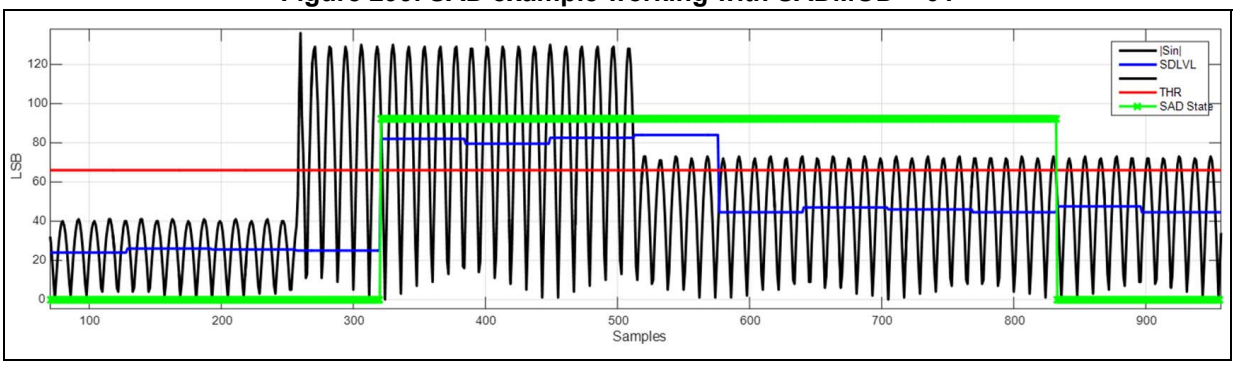
In *Figure 299*, the trigger value (THRH in red) is fixed to 66 LSB. The input signal is at - 65dBFS during 256 samples, then its value goes to - 55 dB for 256 samples, and finally it is reduced to - 60 dBFS.

The blue curve is showing the sound level estimation (SDLVL) versus time. Fluctuation on the estimated value can be observed due to windowing effect of FRSIZE samples.

The SAD DETECT state (when green signal is high) is maintained during four additional frames due to hangover function value.

In this example ANSLP = FRSIZE = 3 (64 samples), LFRNB = 0 (2 frames), HGOVR = 0 (4 frames), SNTHR = 1 (6 dB) and ANMIN = 33.

Figure 299. SAD example working with SADMOD = 01



Threshold programming with SADMOD = 1x

Consider the case of a sound capture where the application wants to wake up the system when the captured sound is bigger than 57 dBSPL.

The sound capture can be performed with a digital microphone such as the MP45DT02. The sensitivity of this microphone is typically - 26 dBFS for an input signal of 94 dBSPL.

An acoustic signal at 57 dBSPL produces a digital signal of about:
- 26 dBFS - (94-63) = - 63 dBFS.

- Adjust SCALE value

  For this example, the filter configuration is the following:
  - CIC4 with a decimation by 48
  - RSFLT bypassed
  - HPF enabled

  A SCALE value of 3.5 dB is recommended for this configuration. The samples provided by DFTL0 are only used for a sound detection, without providing the samples to the application, a bigger gain value can be provided: it increases the SAD accuracy and a saturation does not affect the SAD behavior (for example, a SCALE value of 24 dB).
- Input signal amplitude

  The input signal is - 63 dBFS, corresponding to an amplitude:

  Asin = $10^{(-63/20)}$ = 0.000708 LSB.
- Signal level at SAD input

  The total filter gain for the SAD is:

$$G_{SAD} = 48^4 \times 10^{\frac{24}{20}} \times 0.003906 = 328.6 \times 10^3 \text{ or } 110.3 \text{ dB}$$

  The signal amplitude received by the SAD is:

$$Ain_{SAD} = 0.000708 \times G_{SAD} \sim 232 \text{ LSB}$$

  The gain can be increased if the expected amplitude is too small.

  If the input signal is expected to be a sine, the sound level for a signal amplitude of 232 LSB is:

$$SDLVL = \frac{Ain_{SAD} \times \sqrt{2}}{2} \times 0.9 \sim 148 \text{ LSB}$$

  where 0.9 is the correction factor to apply with respect to the RMS value.

Note: ANLVL converges to average of SDLVL values, with a long constant time.
So SDLVL ~ ANLVL = 148 LSB for a constant input signal at 57 dBSPL.

- Programming trigger value

  For SADMOD = 1', the SAD compares the estimated ambient noise multiplied by the gain selected by SNTHR[3:0] to ANMIN[12:0] * 4.

  For simplification, SNTHR[3:0] is set to 1 (6 dB), meaning that ANLVL is multiplied by two.

  The SAD triggers if 2 * ANLVL > THRH.

  In this mode,

$$THRH = 4 \times ANMIN$$

  So the SAD triggers if:

$$ANLVL > 2 \times ANMIN$$

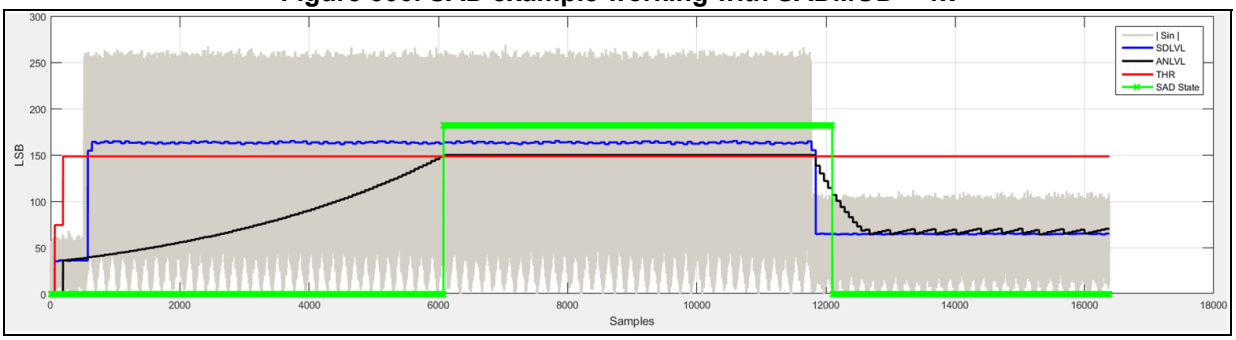
  So ANMIN = 148 / 2 = 74 LSB

  In *Figure 300*, the trigger value (THRH in red) is fixed to 148 LSB. The input signal is at - 75 dBFS during 512 samples, then its value goes to - 62 dB for 11000 samples, and finally it is reduced to - 70 dBFS.

  The blue curve shows the sound level estimation (SDLVL) versus time. The black curve shows the ambient noise estimation versus time, increasing or decreasing logarithmically. During the learning phase, it reaches the SDLVL value.

  In this example ANSLP = 6, FRSIZE = 3 (64 samples), LFRNB = 0 (2 frames), HGOVR = 0 (4 frames), SNTHR = 1 (6 dB) and ANMIN = 74.

Figure 300. SAD example working with SADMOD = 1x



36.8 ADF registers

All the ADF registers must be accessed either in word (32-bit) or half-word (16-bit) formats.

36.8.1 ADF global control register (ADF_GCR)

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.	TRGO
															rw

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

Bit 0 **TRGO**: Trigger output control

This bit is set by software and reset by hardware. It is used to start the acquisition of several filters synchronously. It is also used to synchronize several ADF together by controlling the adf_trgo signal.

0: Write 0 has no effect. Read 0 means that the trigger can be set again to 1.

1: Write 1 generates a positive pulse on the adf_trgo signal and triggers the acquisition on enabled filter having their ACQMOD[2:0] = 01x and selecting TRGO as trigger. Read 1 means that the trigger pulse is still active.

36.8.2 ADF clock generator control register (ADF_CKGCR)

Address offset: 0x004

Reset value: 0x0000 0000

This register is used to control the clock generator. The clock adf_proc_ck must be enabled before enabling other ADF parts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CKGACTIVE	PROCDIV[6:0]							Res.	Res.	Res.	Res.	CCKDIV[3:0]			
r	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRGSRC[3:0]				Res.	Res.	Res.	TRGSENS	Res.	CCK1DIR	CCK0DIR	CKGMOD	Res.	CCK1EN	CCK0EN	CKGDEN
rw	rw	rw	rw				rw		rw	rw	rw		rw	rw	rw

Bit 31 **CKGACTIVE**: Clock generator active flag

This bit is set and cleared by hardware. Ii is used by the application to check if the clock generator is effectively enabled (active) or not. The protected fields of this function can only be updated when CKGACTIVE = 0 (see *Section 36.4.13: Register protection* for details).
The delay between a transition on CKGDEN and a transition on CKGACTIVE is two periods of AHB clock and two 2 periods of adf_proc_ck.
0: The clock generator is not active and can be configured if needed.
1: The clock generator is active and protected fields cannot be configured.

Bits 30:24 **PROCDIV[6:0]**: Divider to control the serial interface clock

this field is set and reset by software. It is used to adjust the frequency of the clock provided to the SITF.

$$F_{adf\_itf\_ck} = \frac{F_{adf\_ker\_ck}}{(PROCDIV + 1)}$$

This field must not be changed if the filter is enabled (DFTEN = 1).
0: adf_ker_ck provided to the SITF
1: adf_ker_ck / 2 provided to the SITF
2: adf_ker_ck / 3 provided to the SITF
...
127: adf_ker_ck / 128 provided to the SITF

Note: This field can be write-protected (see Section 36.4.13: Register protection for details).

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **CCKDIV[3:0]**: Divider to control the ADF_CCK clock

This field is set and reset by software. It is used to adjust the frequency of the ADF_CCK clock. The input clock of this divider is the clock provided to the SITF. More globally, the frequency of the ADF_CCK is given by the following formula:

$$F_{ADF\_CCK} = \frac{F_{adf\_ker\_ck}}{(PROCDIV + 1) \times (CCKDIV + 1)}$$

This field must not be changed if the filter is enabled (DFTEN = 1).
0000: The ADF_CCK clock is adf_proc_ck.
0001: The ADF_CCK clock is adf_proc_ck / 2.
0010: The ADF_CCK clock is adf_proc_ck / 3.
...
1111: The ADF_CCK clock is adf_proc_ck / 16.

Note: This field can be write-protected (see Section 36.4.13: Register protection for details).

Bits 15:12 **TRGSRC[3:0]**: Digital filter trigger signal selection

This field is set and cleared by software. It is used to select which external signals trigger the corresponding filter. This field is not significant if the CKGMOD = 0.
000x: TRGO selected
0010: adf_trg1 selected
others: reserved

Note: This field can be write-protected (see Section 36.4.13: Register protection for details).

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **TRGSENS**: CKGEN trigger sensitivity selection

This bit is set and cleared by software. It is used to select the trigger sensitivity of the trigger signals. This bit is not significant if the CKGMOD = 0.

0: A rising edge event triggers the activation of CKGEN dividers.

1: A falling edge even triggers the activation of CKGEN dividers.

Note: When the trigger source is TRGO, the sensitivity is forced to falling edge, thus TRGSENS value is not taken into account. This bit can be write-protected (see Section 36.4.13: Register protection for details).

Bit 7 Reserved, must be kept at reset value.

Bit 6 **CCK1DIR**: ADF_CCK1 direction

This bit is set and reset by software. It is used to control the direction of the ADF_CCK1 pin.

0: The ADF_CCK1 pin direction is in input.

1: The ADF_CCK1 pin direction is in output.

Note: This bit can be write-protected (see Section 36.4.13: Register protection for details).

Bit 5 **CCK0DIR**: ADF_CCK0 direction

This bit is set and reset by software. It is used to control the direction of the ADF_CCK0 pin.

0: The ADF_CCK0 pin direction is in input.

1: The ADF_CCK0 pin direction is in output.

Note: This bit can be write-protected (see Section 36.4.13: Register protection for details).

Bit 4 **CKGMOD**: Clock generator mode

This bit is set and reset by software. It is used to define the way the clock generator is enabled. This bit must not be changed if the filter is enabled (DFTEN = 1).

0: The kernel clock is provided to the dividers as soon as CKGDEN is set to 1.

1: The kernel clock is provided to the dividers when CKGDEN is set to 1 and the trigger condition met.

Note: This bit can be write-protected (see Section 36.4.13: Register protection for details).

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CCK1EN**: ADF_CCK1 clock enable

This bit is set and reset by software. It is used to control the generation of the bitstream clock on the ADF_CCK1 pin.

0: Bitstream clock not generated

1: Bitstream clock generated on the ADF_CCK1 pin.

Bit 1 **CCK0EN**: ADF_CCK0 clock enable

This bit is set and reset by software. It is used to control the generation of the bitstream clock on the ADF_CCK0 pin.

0: Bitstream clock not generated

1: Bitstream clock generated on the ADF_CCK0 pin

Bit 0 **CKGDEN**: CKGEN dividers enable

This bit is set and reset by software. It is used to enable/disable the clock dividers of the CKGEN: PROCDIV and CCKDIV.

0: CKGEN dividers disabled

1: CKGEN dividers enabled

36.8.3 ADF serial interface control register 0 (ADF_SITF0CR)

Address offset: 0x080

Reset value: 0x0000 1F00

This register is used to control the serial interface SITF0.

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

Bit 31 **SITFACTIVE**: Serial interface active flag

This bit is set and cleared by hardware. It is used by the application to check if the serial interface is effectively enabled (active) or not. The protected fields of this function can only be updated when SITFACTIVE is set to 0 (see *Section 36.4.13: Register protection* for details). The delay between a transition on SITFEN and a transition on SITFACTIVE is two periods of AHB clock and two periods of adf_proc_ck.

0: The serial interface is not active, and can be configured if needed.

1: The serial interface is active and protected fields cannot be configured.

Bits 30:13 Reserved, must be kept at reset value.

Bits 12:8 **STH[4:0]**: Manchester symbol threshold/SPI threshold

This field is set and cleared by software. It is used for Manchester mode to define the expected symbol threshold levels (seer to *Manchester mode* for details on computation). In addition this field is used to define the timeout value for the clock absence detection in Normal SPI mode. STH[4:0] values lower than four are invalid.

Note: This field can be write-protected (see Section 36.4.13: Register protection for details).

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **SITFMOD[1:0]**: Serial interface type

This field is set and cleared by software. It is used to define the serial interface type.

00: LF_MASTER SPI mode

01: Normal SPI mode

10: Manchester mode: rising edge = logic 0, falling edge = logic 1

11: Manchester mode: rising edge = logic 1, falling edge = logic 0

Note: This field can be write-protected (see Section 36.4.13: Register protection for details).

Bit 3 Reserved, must be kept at reset value.

Bits 2:1 **SCKSRC[1:0]**: Serial clock source

This field is set and cleared by software. It is used to select the clock source of the serial interface.

00: Serial clock source is ADF_CCK0.

01: Serial clock source is ADF_CCK1.

others: reserved

Note: This field can be write-protected (see Section 36.4.13: Register protection for details).

Bit 0 **SITFEN**: Serial interface enable

This bit is set and cleared by software. It is used to enable/disable the serial interface.

0: Serial interface disabled

1: Serial interface enabled

36.8.4 ADF bitstream matrix control register 0 (ADF_BSMX0CR)

Address offset: 0x084

Reset value: 0x0000 0000

This register is used to select the bitstream to be provided to DFLT0.

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

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

Bit 31 **BSMXACTIVE**: BSMX active flag

This bit is set and cleared by hardware. It is used by the application to check if the BSMX is effectively enabled (active) or not. BSSEL[4:0] can only be updated when BSMXACTIVE is set to 0. This BSMXACTIVE flag cannot go to 0 if DFLT0 is enabled.

0: BSMX is not active and can be configured if needed.

1: BSMX is active and protected fields cannot be configured.

Bits 30:5 Reserved, must be kept at reset value.

Bits 4:0 **BSSEL[4:0]**: Bitstream selection

This field is set and cleared by software. It is used to select the bitstream to be processed for DFLT0.

00000: bs0_r provided to DFLT0

00001: bs0_f provided to DFLT0

others: reserved

Note: This field can be write-protected (see Section 36.4.13: Register protection for details).

36.8.5 ADF digital filter control register 0 (ADF_DFLT0CR)

Address offset: 0x088

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DFLTACTIVE	DFLTRUN	Res.	Res.	NBDIS[7:0]								Res.	Res.	Res.	Res.
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
TRGSRC[3:0]				Res.	Res.	Res.	TRGSENS	Res.	ACQMOD[2:0]			Res.	FTH	DMAEN	DFLTEN
rw	rw	rw	rw				rw		rw	rw	rw		rw	rw	w

Bit 31 **DFLTACTIVE**: DFLT0 active flag

This bit is set and cleared by hardware. It indicates if DFLT0 is active: can be running or waiting for events.

0: DFLT0 not active (can be re-enabled again, via DFLTEN bit, if needed)

1: DFLT0 active

Bit 30 **DFLTRUN**: DFLT0 run status flag

This bit is set and cleared by hardware. It indicates if DFLT0 is running or not.

0: DFLT0 not running and ready to accept a new trigger event

1: DFLT0running

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

Bits 27:20 **NBDIS[7:0]**: Number of samples to be discarded

This field is set and cleared by software. It is used to define the number of samples to be discarded every time DFLT0 is re-started.

0: No sample discarded

1: 1 sample discarded

2: 2 samples discarded

...

255: 255 samples discarded

Note: This field can be write-protected (see Section 36.4.13: Register protection for details).

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:12 **TRGSRC[3:0]**: DFLT0 trigger signal selection

This field is set and cleared by software. It is used to select which external signals trigger DFLT0.

0000: TRGO selected

0010: adf_trgi selected

others: Reserved

Note: This field can be write-protected (see Section 36.4.13: Register protection for details).

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **TRGSENS**: DFLT0 trigger sensitivity selection

This field is set and cleared by software. It is used to select the trigger sensitivity of the external signals

0: A rising edge event triggers the acquisition.

1: A falling edge even triggers the acquisition.

Note: When the trigger source is TRGO, TRGSENS value is not taken into account. When TRGO is selected, the sensitivity is forced to falling edge.
This bit can be write-protected (see Section 36.4.13: Register protection for details).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ACQMOD[2:0]**: DFLT0 trigger mode

This field is set and cleared by software. It is used to select the filter trigger mode.

000: Asynchronous continuous acquisition mode

001: Asynchronous single-shot acquisition mode

010: Synchronous continuous acquisition mode

011: Synchronous single-shot acquisition mode

100: Window continuous acquisition mode

others: same as 000

Note: This field can be write-protected (see Section 36.4.13: Register protection for details)..

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FTH**: RXFIFO threshold selection

This bit is set and cleared by software. It is used to select the RXFIFO threshold.

0: RXFIFO threshold event generated when the RXFIFO is not empty

1: RXFIFO threshold event generated when the RXFIFO is half-full

Note: This bit can be write-protected (see Section 36.4.13: Register protection for details).

Bit 1 **DMAEN**: DMA requests enable

This bit is set and cleared by software. It is used to control the generation of DMA request to transfer the processed samples into the memory.

0: DMA interface for the corresponding digital filter disabled

1: DMA interface for the corresponding digital filter enabled

Note: This bit can be write-protected (see Section 36.4.13: Register protection for details).

Bit 0 **DFLTEN**: DFLT0 enable

This bit is set and cleared by software. It is used to control the start of acquisition of the DFLT0 path. This bit behavior depends on ACQMOD[2:0] and external events. The serial or parallel interface delivering the samples must be enabled as well.

0: Acquisition immediately stopped

1: Acquisition immediately started if ACQMOD[2:0] = 00x or 101, or acquisition started when the proper trigger event occurs if ACQMOD[2:0] = 01x.

36.8.6 ADF digital filer configuration register 0 (ADF_DFLT0CICR)

Address offset: 0x08C

Reset value: 0x0000 0000

This register is used to control the main CIC filter.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SCALE[5:0]						Res.	Res.	Res.	MCICD8
						rw	rw	rw	rw	rw	rw				rw

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

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:20 **SCALE[5:0]**: Scaling factor selection

This field is set and cleared by software. It is used to select the gain to be applied at CIC output (see *Table 318* for details). If the application attempts to write a new gain value while the previous one is not yet applied, this new gain value is ignored. Reading back this field informs the application on the current gain value.

000000: 0 dB
000001: + 3.5 dB
000010: + 6 dB or shift left by 1 bit
...
011000: + 72 dB or shift left by 12 bits
100000: - 48.2 dB or shift right by 8 bits (default value)
100001: - 44.6 dB
100010: - 42.1 dB or shift right by 7 bits
100011: - 38.6 dB
...
101110: -6 dB or shift right by 1 bit
101111: -2.5 dB
others: Reserved

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

Bits 16:8 **MCICD[8:0]**: CIC decimation ratio selection

This field is set and cleared by software.It is used to select the CIC decimation ratio. A decimation ratio smaller than two is not allowed. The decimation ratio is given by (CICDEC+1).

0: Decimation ratio is 2.
1: Decimation ratio is 2.
2: Decimation ratio is 3.
3: Decimation ratio is 4.
...
511: Decimation ratio is 512.

Note: This field can be write-protected (see Section 36.4.13: Register protection for details).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **CICMOD[2:0]**: Select the CIC order
This field is set and cleared by software. It is used to select the order of the MCIC.
100: MCIC configured in single $Sinc^4$ filter
101: MCIC configured in single $Sinc^5$ filter
others: reserved
Note: This field can be write-protected (see Section 36.4.13: Register protection for details).

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **DATSRC[1:0]**: Source data for the digital filter
This field is set and cleared by software.
0x: Stream coming from the BSMX selected
10: Stream coming from the ADCITF1 selected
11: Stream coming from the ADCITF2 selected
Note: This field can be write-protected (see Section 36.4.13: Register protection for details).

36.8.7 ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)

Address offset: 0x090

Reset value: 0x0000 0000

This register is used to control the reshape and HPF filter.

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

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:8 **HPFC[1:0]**: High-pass filter cut-off frequency
This field is set and cleared by software. it is used to select the cut-off frequency of the high-pass filter. F_{PCM} represents the sampling frequency at HPF input.
00: Cut-off frequency = 0.000625 x FPCM
01: Cut-off frequency = 0.00125 x FPCM
10: Cut-off frequency = 0.00250 x FPCM
11: Cut-off frequency = 0.00950 x FPCM
Note: This field can be write-protected (see Section 36.4.13: Register protection for details).

Bit 7 **HPFBYP**: High-pass filter bypass
This bit is set and cleared by software. It is used to bypass the high-pass filter.
0: HPF not bypassed (default value)
1: HPF bypassed
Note: This bit can be write-protected (see Section 36.4.13: Register protection for details).

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **RSFLTD**: Reshaper filter decimation ratio

This bit is set and cleared by software. It is used to select the decimation ratio for the reshape filter

0: Decimation ratio is 4 (default value).

1: Decimation ratio is 1.

Note: This bit can be write-protected (see Section 36.4.13: Register protection for details).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **RSFLTBYP**: Reshaper filter bypass

This bit is set and cleared by software. It is used to bypass the reshape filter and its decimation block.

0: Reshape filter not bypassed (default value)

1: Reshape filter bypassed

Note: This bit can be write-protected (see Section 36.4.13: Register protection for details).

36.8.8 ADF delay control register 0 (ADF_DLY0CR)

Address offset: 0x0A4

Reset value: 0x0000 0000

This register is used for the adjustment stream delays.

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

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

Bit 31 **SKPBF**: Skip busy flag

This bit is set and cleared by hardware. It is used to control if the delay sequence is completed.

0: ADF ready to accept a new value into SKPDLY[6:0]

1: Last valid SKPDLY[6:0] still under precessing

Bits 30:7 Reserved, must be kept at reset value.

Bits 6:0 **SKPDLY[6:0]**: Delay to apply to a bitstream

This field is set and cleared by software. It defines the number of input samples that are skipped. Skipping is applied immediately after writing to this field, if SKPBF = 0 and DFLTEN = 1. If SKPBF = 1, the value written into the register is ignored by the delay state machine.

0: No input sample skipped

1: 1 input sample skipped

...

127: 127 input samples skipped

36.8.9 ADF DFLT0 interrupt enable register (ADF_DFLT0IER)

Address offset: 0x0AC

Reset value: 0x0000 0000

This register is used for allowing or not the events to generate an interrupt.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SDLVLIE	SDDETIE	RFOVRIE	CKABIE	SATIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOVRIE	FTHIE
		rw	rw	rw	rw	rw								rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **SDLVLIE**: SAD sound-level value ready enable
This bit is set and cleared by software.
0: Sound-level-ready interrupt disabled
1: Sound-level-ready interrupt enabled

Bit 12 **SDDETIE**: Sound activity detection interrupt enable
This bit is set and cleared by software.
0: Sound-trigger interrupt disabled
1: Sound-trigger interrupt enabled

Bit 11 **RFOVRIE**: Reshape filter overrun interrupt enable
This bit is set and cleared by software.
0: Reshape filter overrun interrupt disabled
1: Reshape filter overrun interrupt enabled

Bit 10 **CKABIE**: Clock absence detection interrupt enable
This bit is set and cleared by software.
0: Clock absence interrupt disabled
1: Clock absence interrupt enabled

Bit 9 **SATIE**: Saturation detection interrupt enable
This bit is set and cleared by software.
0: Saturation interrupt disabled
1: Saturation interrupt enabled

Bits 8:2 Reserved, must be kept at reset value.

Bit 1 **DOVRIE**: Data overflow interrupt enable
This bit is set and cleared by software.
0: Data overflow interrupt disabled
1: Data overflow interrupt enabled

Bit 0 **FTHIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: RXFIFO threshold interrupt disabled
1: RXFIFO threshold interrupt enabled

36.8.10 ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR)

Address offset: 0x0B0

Reset value: 0x0000 0000

This register contains the status flags for the digital filter path.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SDLVLF	SDDETF	RFOVRF	CKABF	SATF	Res.	Res.	Res.	Res.	Res.	RXNEF	Res.	DOVRF	FTHF
		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1						r		rc_w1	r

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **SDLVLF**: Sound level value ready flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that new sound level value is not ready. Write 0 has no effect.
1: Read 1 means that new sound level value is ready. Write 1 clears this flag.

Bit 12 **SDDETF**: Sound activity detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no sound activity is detected. Write 0 has no effect.
1: Read 1 means that sound activity is detected. Write 1 clears this flag.

Bit 11 **RFOVRF**: Reshape filter overrun detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no reshape filter overrun is detected. Write 0 has no effect.
1: Read 1 means that reshape filter overrun is detected. Write 1 clears this flag.

Bit 10 **CKABF**: Clock absence detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no clock absence is detected. Write 0 has no effect.
1: Read 1 means that a clock absence is detected. Write 1 clears this flag.

Bit 9 **SATF**: Saturation detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no saturation is detected. Write 0 has no effect.
1: Read 1 means that a saturation is detected. Write 1 clears this flag.

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **RXNEF**: RXFIFO not empty flag
This bit is set and cleared by hardware according to the RXFIFO level.
0: RXFIFO empty
1: RXFIFO not empty

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DOVRF**: Data overflow flag

This bit is set by hardware and cleared by software by writing this bit to 1.

0: Read 0 means that no overflow is detected. Write 0 has no effect.

1: Read 1 means that an overflow is detected; Write 1 clears this flag.

Bit 0 **FTHF**: RXFIFO threshold flag

This bit is set by hardware, and cleared by the hardware when the RXFIFO level is lower than the threshold.

0: RXFIFO threshold not reached

1: RXFIFO threshold reached

36.8.11 ADF SAD control register (ADF_SADCR)

Address offset: 0x0B8

Reset value: 0x0000 0000

This register is used for the configuration and the control of the sound activity detection.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SADMOD[1:0]		Res.	FRSIZE[2:0]			HYSTEN	Res.	SADST[1:0]		DETCFG	DATCAP[1:0]		SADEN
		rw	rw		rw	rw	rw	rw		r	r	rw	rw	rw	rw

Bit 31 **SADACTIVE**: SAD Active flag

This bit is set and cleared by hardware. It is used to check if the SAD is effectively enabled (active) or not. The protected fields and registers of this function can only be updated when the SADACTIVE is set to 0 (see *Section 36.4.13: Register protection* for details).

The delay between a transition on SADEN and a transition on SADACTIVE is two periods of AHB clock and two periods of adf_proc_ck.

0: SAD not active and can be configured if needed

1: SAD active and protected fields cannot be configured.

Bits 30:14 Reserved, must be kept at reset value.

Bits 13:12 **SADMOD[1:0]**: SAD working mode

This field is set and cleared by software. It is used to define the way the SAD works.

00: Threshold value computed according to the estimated ambient noise

The SAD triggers when the sound level (SDLVL) is bigger than the defined threshold. In this mode, the SAD works like a voice activity detector.

01: Threshold value equal to ANMIN[12:0], multiplied by the gain selected by SNTHR[3:0]

The SAD triggers when the sound level (SDLVL) is bigger than the defined threshold. In this mode, the SAD works like a sound detector.

1x: Threshold value given by 4 x ANMIN[12:0]

The SAD triggers when the estimated ambient noise (ANLVL), multiplied by the gain selected by SNTHR[3:0] is bigger than the defined threshold. In this mode, the SAD is working like an ambient noise estimator. Hysteresis function cannot be used in this mode.

Note: This field can be write-protected (see Section 36.4.13: Register protection for details).

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **FRSIZE[2:0]**: Frame size

This field is set and cleared by software. it is used to define the size of one frame and also to define how many samples are taken into account to compute the short-term signal level.

000: 8 PCM samples used to compute the short-term signal level

001: 16 PCM samples used to compute the short-term signal level

010: 32 PCM samples used to compute the short-term signal level

011: 64 PCM samples used to compute the short-term signal level

100: 128 PCM samples used to compute the short-term signal level

101: 256 PCM samples used to compute the short-term signal level

11x: 512 PCM samples used to compute the short-term signal level

Note: This field can be write-protected (see Section 36.4.13: Register protection for details).

Bit 7 **HYSTEN**: Hysteresis enable

This bit is set and cleared by software. It is used to enable/disable the hysteresis function (see *Table 318* for details). This bit must be kept to 0 when SADMOD[1:0] = 1x.

0: Hysteresis function disabled. THR_H is always used.

1: Hysteresis function enabled. THR_H is used for MONITOR to DETECT transition and THR_L is used for DETECT to MONITOR transition.

Note: This bit can be write-protected (see Section 36.4.13: Register protection for details).

Bit 6 Reserved, must be kept at reset value.

Bits 5:4 **SADST[1:0]**: SAD state

This field is set and cleared by hardware. It indicates the SAD state and is meaningful only when SADEN = 1. The SAD state can be:

- LEARN when the SAD is in learning phase or in SDLVL computation mode
- MONITOR when the SAD is in monitoring phase
- DETECT when the SAD detects a sound

00: SAD in LEARN state

01: SAD in MONITOR state

11: SAD in DETECT state

Bit 3 **DETCFG**: Sound trigger event configuration

This bit is set and cleared by software. It is used to define if the sddet_evt event is generated only when the SAD enters to MONITOR state or when the SAD enters or exits the DETECT state.

0: sddet_evt generated when SAD enters the MONITOR state

1: sddet_evt generated when SAD enters or exits the DETECT state

Note: This bit can be write-protected (see Section 36.4.13: Register protection for details).

Bits 2:1 **DATCAP[1:0]**: Data capture mode

This field is set and cleared by software. It is used to define in which conditions, the samples provided by DLFT0 are stored into the memory.

00: Samples from DFLT0 not transfered into the memory

01: Samples from DFLT0 transfered into the memory when SAD is in DETECT state

1x: Samples from DFLT0 transfered into memory when SAD and DFLT0 are enabled

Note: This field can be write-protected (see Section 36.4.13: Register protection for details).

Bit 0 **SADEN**: Sound activity detector enable

This bit is set and cleared by software. It is used to enable/disable the SAD.

0: SAD disabled and SAD state reset

1: SAD enabled

36.8.12 ADF SAD configuration register (ADF_SADCFGR)

Address offset: 0x0BC

Reset value: 0x0000 0000

This register is used for the configuration of the sound activity detection.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ANMIN[12:0]												
			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.	HGOVR[2:0]			Res.	LFRNB[2:0]			Res.	ANSLP[2:0]			SNTHR[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 28:16 **ANMIN[12:0]**: Minimum noise level

This field is set and cleared by software. It is used to define the minimum noise level and then the sensitivity. It represents a positive number.

Note: This field can be write-protected (see Section 36.4.13: Register protection for details).

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **HGOVR[2:0]**: Hangover time window

This field is set and cleared by software. Once the SAD state is DETECT, this parameter is used to define the amount of time the sound is allowed to remain below the threshold, before switching the SAD to MONITOR state (see FRSIZE field for the description of a frame).

000: SAD back to MONITOR state if sound is below threshold for 4 frames

001: SAD back to MONITOR state if sound is below threshold for 8 frames

010: SAD back to MONITOR state if sound is below threshold for 16 frames

011: SAD back to MONITOR state if sound is below threshold for 32 frames

100: SAD back to MONITOR state if sound is below threshold for 64 frames

101: SAD back to MONITOR state if sound is below threshold for 128 frames

110: SAD back to MONITOR state if sound is below threshold for 256 frames

111: SAD back to MONITOR state if sound is below threshold for 512 frames

Note: This field can be write-protected (see Section 36.4.13: Register protection for details).

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **LFRNB[2:0]**: Number of learning frames

This field is set and cleared by software. It is used to define the number of learning frames to perform the first estimate of the noise level.
000: 2 frames used to compute the initial noise level
001: 4 frames used to compute the initial noise level
010: 8 frames used to compute the initial noise level
011: 16 frames used to compute the initial noise level
1xx: 32 frames used to compute the initial noise level

Note: This field can be write-protected (see Section 36.4.13: Register protection for details).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ANSLP[2:0]**: Ambient noise slope control

This field is set and cleared by software. It is used to define the positive and negative slope of the noise estimator, in charge of updating the ANLVL (see *Ambient noise estimation (ANLVL)* for information about programming this field).

Note: This field can be write-protected (see Section 36.4.13: Register protection for details).

Bits 3:0 **SNTHR[3:0]**: Signal to noise threshold

This field is set and cleared by software. It is used to define THR_H (and THR_L if hysteresis is enabled). See *Table 318* for details.
0000: THR_H is 3.5 dB higher than ANLVL
0001: THR_H is 6.0 dB higher than ANLVL
0010: THR_H is 9.5 dB higher than ANLVL
0011: THR_H is 12 dB higher than ANLVL
0100: THR_H is 15.6 dB higher than ANLVL
0101: THR_H is 18 dB higher than ANLVL
0110: THR_H is 21.6 dB higher than ANLVL
0111: THR_H is 24.1 dB higher than ANLVL
1000: THR_H is 27.6 dB higher than ANLVL
1001: THR_H is 30.1dB higher than ANLVL
others: Reserved

Note: This field can be write-protected (see Section 36.4.13: Register protection for details).

36.8.13 ADF SAD sound level register (ADF_SADSDLVR)

Address offset: 0x0C0

Reset value: 0x0000 0000

This register contains the short term sound level computed by the SAD.

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

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

Bits 14:0 **SDLVL[14:0]**: Short term sound level

This field is set by hardware. It contains the latest sound level computed by the SAD. To refresh this value, SDLVLF must be cleared.

36.8.14 ADF SAD ambient noise level register (ADF_SADANLVR)

Address offset: 0x0C4

Reset value: 0x0000 0000

This register contains the ambient noise level computed by the SAD.

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

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

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

Bits 14:0 **ANLVL[14:0]**: Ambient noise level estimation

This field is set by hardware. It contains the latest ambient noise level computed by the SAD. To refresh this field, the SDLVLF flag must be cleared.

36.8.15 ADF digital filter data register 0 (ADF_DFLT0DR)

Address offset: 0x0F0

Reset value: 0x0000 0000

This register is used to read the data processed by the digital filter.

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

Bits 31:8 **DR[23:0]**: Data processed by DFT0

Bits 7:0 Reserved, must be kept at reset value.

36.8.16 ADF register map

Table 331. ADF 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	ADF_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.	Res.	Res.	Res.	Res.	Res.	TRGO
	Reset value																																0

Table 331. ADF register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	ADF_CKGCR	CKGACTIVE	PROCDIV[6:0]							Res.	Res.	Res.	Res.	CCKDIV[3:0]				TRGSRC[3:0]				Res.	Res.	Res.	TRGSENS	Res.	CCK1DIR	CCK0DIR	CKGMOD	Res.	CCK1EN	CCK0EN	CKGDEN
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0				0		0	0	0		0	0	0
0x008 - 0x07C	Reserved	Reserved																															
0x080	ADF_SITF0CR	SITFACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STH[4:0]					Res.	Res.	SITFMOD[1:0]		Res.	SCKSRC[1:0]		SITFEN
	Reset value	0																			1	1	1	1	1			0	0		0	0	0
0x084	ADF_ BSMX0CR	BSMXACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BSSEL[4:0]				
	Reset value	0																											0	0	0	0	0
0x088	ADF_DFLT0CR	DFLTACTIVE	DFLTRUN	Res.	Res.	NBDIS[7:0]								Res.	Res.	Res.	Res.	TRGSRC[3:0]				Res.	Res.	Res.	TRGSENS	Res.	ACQMOD[2:0]			Res.	FTH	DMAEN	DFLTEN
	Reset value	0	0			0	0	0	0	0	0	0	0					0	0	0	0				0		0	0	0		0	0	0
0x08C	ADF_ DFLT0CICR	Res.	Res.	Res.	Res.	Res.	Res.	SCALE[5:0]						Res.	Res.	Res.	MCICD[8:0]									Res.	CICMOD[2:0]			Res.	Res.	DATSRC[1:0]	
	Reset value							0	0	0	0	0	0				0	0	0	0	0	0	0	0	0		0	0	0			0	0
0x090	ADF_ DFLT0RSFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HPFC[1:0]	HPFBYP	Res.	Res.	Res.	RSFLTD	Res.	Res.	Res.	RSFLTBYP
	Reset value																							0	0	0			0				0
0x094 - 0x0A0	Reserved	Reserved																															
0x0A4	ADF_DLY0CR	SKPBF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SKPDLY[6:0]						
	Reset value	0																									0	0	0	0	0	0	0
0x0A8	Reserved	Reserved																															

Table 331. ADF 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
0x0AC	ADF_ DFLT0IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDLVLIE	SDDETIE	RFOVRIE	CKABIE	SATIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOVRIE	FTHIE
	Reset value																			0	0	0	0	0	0	0			0		0	0	0
0x0B0	ADF_ DFLT0ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDLVLF	SDDETF	RFOVRF	CKABF	SATF	Res.	Res.	Res.	Res.	Res.	RXNEF	Res.	DOVRF	FTHF
	Reset value																			0	0	0	0	0						0		0	0
0x0B4	Reserved	Reserved																															
0x0B8	ADF_SADCR	SADACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SADMOD[1:0]		Res.	FRSIZE[2:0]			HYSTEN	Res.	SADST[1:0]		DETCFG	DATCAP[1:0]		SADEN
	Reset value	0																		0	0		0	0	0	0		0	0	0	0	0	0
0x0BC	ADF_ SADCFGR	Res.	Res.	Res.	ANMIN[12:0]														Res.	HGOVR[2:0]		Res.	LFRNB[2:0]			Res.	ANSLP[2:0]			Res.	SNTHR[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
0x0C0	ADF_ SADSDLVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDLVL[14:0]															
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C4	ADF_ SADANLVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ANLVL[14:0]															
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C8 - 0x0EC	Reserved	Reserved																															
0x0F0	ADF_DFLT0DR	DR[23:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								

Refer to *Section 2.3* for the register boundary addresses.

37 Digital camera interface (DCMI)

37.1 Introduction

The digital camera is a synchronous parallel interface able to receive a high-speed data flow from an external 8-, 10-, 12- or 14-bit CMOS camera module. It supports different data formats: YCbCr4:2:2/RGB565 progressive video and compressed data (JPEG).

37.2 DCMI main features

- 8-, 10-, 12- or 14-bit parallel interface
- Embedded/external line and frame synchronization
- Continuous or snapshot mode
- Crop feature
- Supports the following data formats:
  - 8/10/12/14-bit progressive video: either monochrome or raw Bayer
  - YCbCr 4:2:2 progressive video
  - RGB 565 progressive video
  - Compressed data: JPEG

37.3 DCMI functional description

The digital camera interface is a synchronous parallel interface that can receive high-speed data flows. It consists of up to 14 data lines (DCMI_D[13:0]) and a pixel clock line (DCMI_PIXCLK). The pixel clock has a programmable polarity, so that data can be captured on either the rising or the falling edge of the pixel clock.

The data are packed into a 32-bit data register (DCMI_DR) and then transferred through a general-purpose DMA channel. The image buffer is managed by the DMA, not by the camera interface.

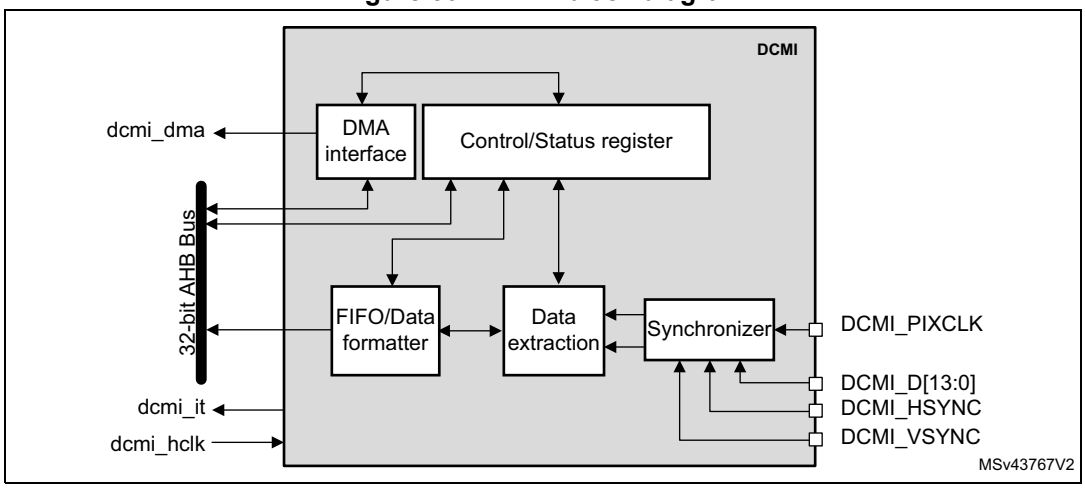
The data received from the camera can be organized in lines/frames (raw YUB/RGB/Bayer modes) or can be a sequence of JPEG images. To enable JPEG image reception, the JPEG bit (bit 3 of DCMI_CR register) must be set.

The data flow is synchronized either by hardware using the optional DCMI_HSYNC (horizontal synchronization) and DCMI_VSYNC (vertical synchronization) signals or by synchronization codes embedded in the data flow.

37.3.1 DCMI block diagram

Figure 301 shows the DCMI block diagram.

Figure 301. DCMI block diagram



37.3.2 DCMI pins and internal signals

The following table shows DCMI pins.

Table 332. DCMI input/output pins

Mode	Pin name	Signal type	Description
8 bits 10 bits 12 bits 14 bits	DCMI_D[7:0] DCMI_D[9:0] DCMI_D[11:0] DCMI_D[13:0]	Inputs	DCMI data
DCMI_PIXCLK		Input	Pixel clock
DCMI_HSYNC		Input	Horizontal synchronization / Data valid
DCMI_VSYNC		Input	Vertical synchronization

The following table shows DCMI internal signals.

Table 333. DCMI internal input/output signals

Internal signal name	Signal type	Description
dcmi_dma	Output	DCMI DMA request
dcmi_it	Output	DCMI interrupt request
dcmi_hclk	Input	DCMI interface clock

37.3.3 DCMI clocks

The digital camera interface uses two clock domains, DCMI_PIXCLK and HCLK. The signals generated with DCMI_PIXCLK are sampled on the rising edge of HCLK once they are stable. An enable signal is generated in the HCLK domain, to indicate that data coming from the camera are stable and can be sampled. The maximum DCMI_PIXCLK period must be higher than 2.5 HCLK periods.

37.3.4 DCMI DMA interface

The DMA interface is active when the CAPTURE bit of the DCMI_CR register is set. A DMA request is generated each time the camera interface receives a complete 32-bit data block in its register.

37.3.5 DCMI physical interface

The interface is composed of 11/13/15/17 inputs. Only the Slave mode is supported.

The camera interface can capture 8-bit, 10-bit, 12-bit or 14-bit data depending on the EDM[1:0] bits of the DCMI_CR register. If less than 14 bits are used, the unused input pins must be connected to ground.

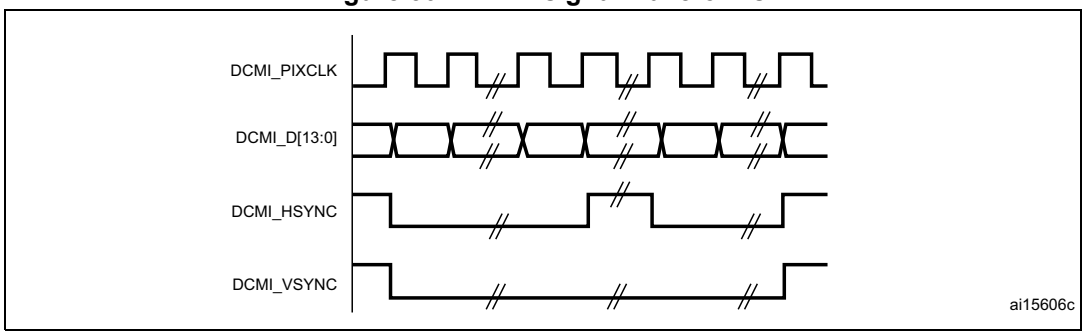
DCMI pins are shown in *Table 332*.

The data are synchronous with DCMI_PIXCLK and change on the rising/falling edge of the pixel clock depending on the polarity.

The DCMI_HSYNC signal indicates the start/end of a line.

The DCMI_VSYNC signal indicates the start/end of a frame

Figure 302. DCMI signal waveforms



1. The capture edge of DCMI_PIXCLK is the falling edge, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

8-bit data

When EDM[1:0] = 00 in DCMI_CR the interface captures 8 LSBs at its input (DCMI_D[7:0]) and stores them as 8-bit data. The DCMI_D[13:8] inputs are ignored. In this case, to capture a 32-bit word, the camera interface takes four pixel clock cycles.

The first captured data byte is placed in the LSB position in the 32-bit word and the 4th captured data byte is placed in the MSB position in the 32-bit word. The table below gives an example of the positioning of captured data bytes in two 32-bit words.

Table 334. Positioning of captured data bytes in 32-bit words (8-bit width)

Byte address	31:24	23:16	15:8	7:0
0	D_{n+3}[7:0]	D_{n+2}[7:0]	D_{n+1}[7:0]	D_n[7:0]
4	D_{n+7}[7:0]	D_{n+6}[7:0]	D_{n+5}[7:0]	D_{n+4}[7:0]

10-bit data

When EDM[1:0] = 01 in DCMI_CR, the camera interface captures 10-bit data at its input DCMI_D[9:0] and stores them as the 10 least significant bits of a 16-bit word. The remaining most significant bits of the DCMI_DR register (bits 11 to 15) are cleared to zero. So, in this case, a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in the table below.

Table 335. Positioning of captured data bytes in 32-bit words (10-bit width)

Byte address	31:26	25:16	15:10	9:0
0	0	D_{n+1}[9:0]	0	D_n[9:0]
4	0	D_{n+3}[9:0]	0	D_{n+2}[9:0]

12-bit data

When EDM[1:0] = 10 in DCMI_CR, the camera interface captures the 12-bit data at its input DCMI_D[11:0] and stores them as the 12 least significant bits of a 16-bit word. The remaining most significant bits are cleared to zero. So, in this case a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in the table below.

Table 336. Positioning of captured data bytes in 32-bit words (12-bit width)

Byte address	31:28	27:16	15:12	11:0
0	0	D_{n+1}[11:0]	0	D_n[11:0]
4	0	D_{n+3}[11:0]	0	D_{n+2}[11:0]

14-bit data

When EDM[1:0] = 11 in DCMI_CR, the camera interface captures the 14-bit data at its input DCMI_D[13:0] and stores them as the 14 least significant bits of a 16-bit word. The remaining most significant bits are cleared to zero. So, in this case a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in the table below.

Table 337. Positioning of captured data bytes in 32-bit words (14-bit width)

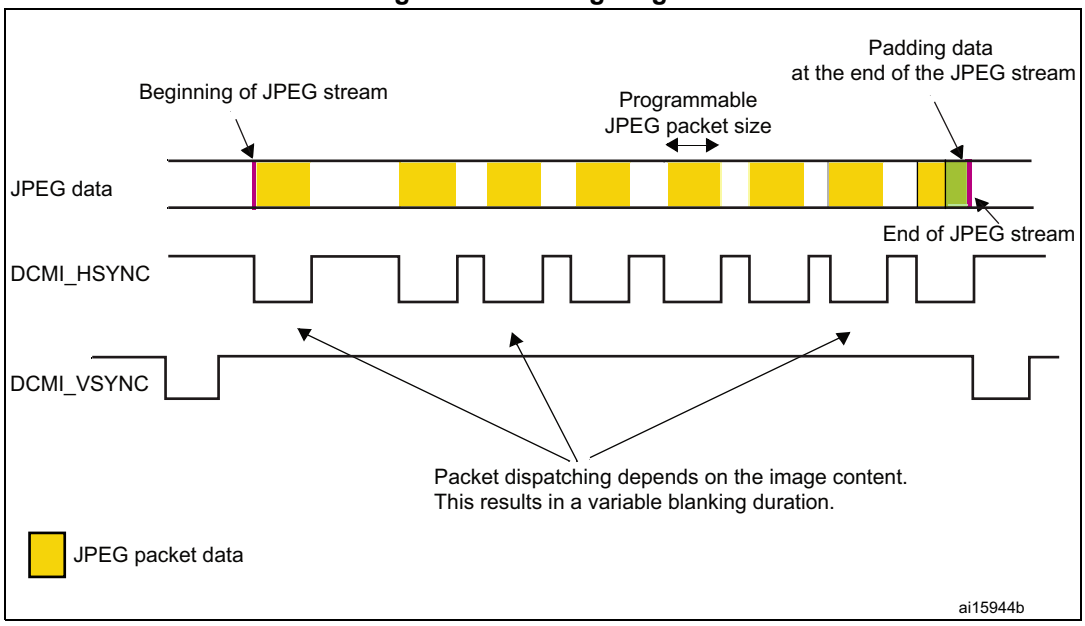
Byte address	31:30	29:16	15:14	13:0
0	0	D_{n+1}[13:0]	0	D_n[13:0]
4	0	D_{n+3}[13:0]	0	D_{n+2}[13:0]

37.3.6 DCMI synchronization

The digital camera interface supports embedded or hardware (DCMI_HSYNC and DCMI_VSYNC) synchronization. When embedded synchronization is used, it is up to the digital camera module to make sure that the 0x00 and 0xFF values are used ONLY for synchronization (not in data). Embedded synchronization codes are supported only for the 8-bit parallel data interface width (that is, in the DCMI_CR register, the EDM[1:0] bits must be cleared).

For compressed data, the DCMI supports only the hardware synchronization mode. In this case, DCMI_VSYNC is used as a start/end of the image, and DCMI_HSYNC is used as a Data Valid signal. *Figure 303* shows the corresponding timing diagram.

Figure 303. Timing diagram



Hardware synchronization mode

In hardware synchronization mode, the two synchronization signals (DCMI_HSYNC/DCMI_VSYNC) are used.

Depending on the camera module/mode, data may be transmitted during horizontal/vertical synchronization periods. The DCMI_HSYNC/DCMI_VSYNC signals act like blanking signals since all the data received during DCMI_HSYNC/DCMI_VSYNC active periods are ignored.

In order to correctly transfer images into the DMA/RAM buffer, data transfer is synchronized with the DCMI_VSYNC signal. When the hardware synchronization mode is selected, and

capture is enabled (CAPTURE bit set in DCMI_CR), data transfer is synchronized with the deactivation of the DCMI_VSYNC signal (next start of frame).

Transfer can then be continuous, with successive frames transferred by DMA to successive buffers or the same/circular buffer. To allow the DMA management of successive frames, a VSIF (Vertical synchronization interrupt flag) is activated at the end of each frame.

Embedded data synchronization mode

In this synchronization mode, the data flow is synchronized using 32-bit codes embedded in the data flow. These codes use the 0x00/0xFF values that are *not* used in data anymore. There are 4 types of codes, all with a 0xFF0000XY format. The embedded synchronization codes are supported only in 8-bit parallel data width capture (in the DCMI_CR register, the EDM[1:0] bits must be cleared). For other data widths, this mode generates unpredictable results and must not be used.

Note: Camera modules can have 8 such codes (in interleaved mode). For this reason, the interleaved mode is not supported by the camera interface (otherwise, every other half-frame would be discarded).

- Mode 2

  Four embedded codes signal the following events
  - Frame start (FS)
  - Frame end (FE)
  - Line start (LS)
  - Line end (LE)

  The XY values in the 0xFF0000XY format of the four codes are programmable (see *Section 37.5.7: DCMI embedded synchronization code register (DCMI_ESCR)*).

  A 0xFF value programmed as a “frame end” means that all the unused codes are interpreted as valid frame end codes.

  In this mode, once the camera interface has been enabled, the frame capture starts after the first occurrence of the frame end (FE) code followed by a frame start (FS) code.
- Mode 1

  An alternative coding is the camera mode 1. This mode is ITU656 compatible.

  The codes signal another set of events:
  - SAV (active line) - line start
  - EAV (active line) - line end
  - SAV (blanking) - end of line during interframe blanking period
  - EAV (blanking) - end of line during interframe blanking period

This mode can be supported by programming the following codes:

- FS $\leq$ 0xFF
- FE ≤ 0xFF
- LS ≤ SAV (active)
- LE ≤ EAV (active)

An embedded unmask code is also implemented for frame/line start and frame/line end codes. Using it, it is possible to compare only the selected unmasked bits with the programmed code. A bit can therefore be selected to compare in the embedded code and

detect a frame/line start or frame/line end. This means that there can be different codes for the frame/line start and frame/line end with the unmasked bit position remaining the same.

Example

FS = 0xA5

Unmask code for FS = 0x10

In this case the frame start code is embedded in the bit 4 of the frame start code.

37.3.7 DCMI capture modes

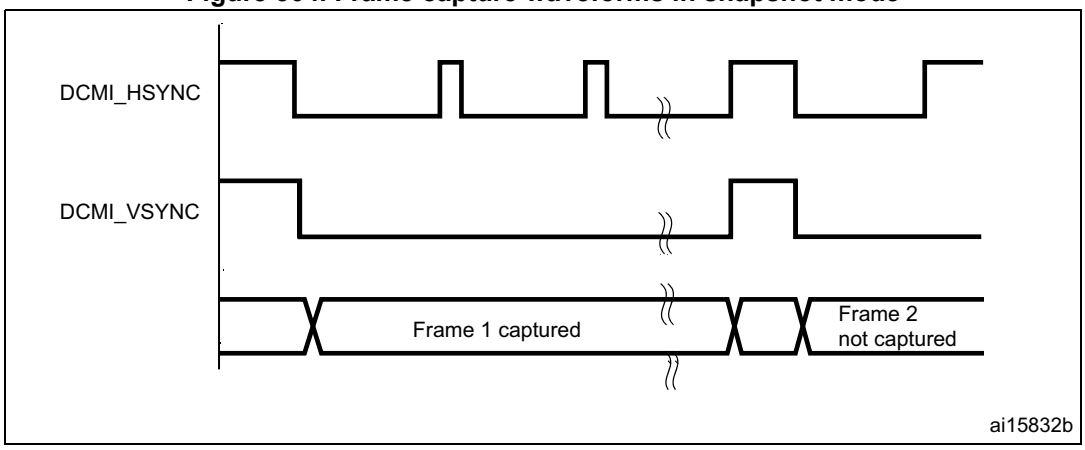
This interface supports two types of capture: snapshot (single frame) and continuous grab.

Snapshot mode (single frame)

In this mode, a single frame is captured (CM = 1 of the DCMI_CR register). After the CAPTURE bit is set in DCMI_CR, the interface waits for the detection of a start of frame before sampling the data. The camera interface is automatically disabled (CAPTURE bit cleared in DCMI_CR) after receiving the first complete frame. An interrupt is generated (IT_FRAME) if it is enabled.

In case of an overrun, the frame is lost and the CAPTURE bit is cleared.

Figure 304. Frame capture waveforms in snapshot mode

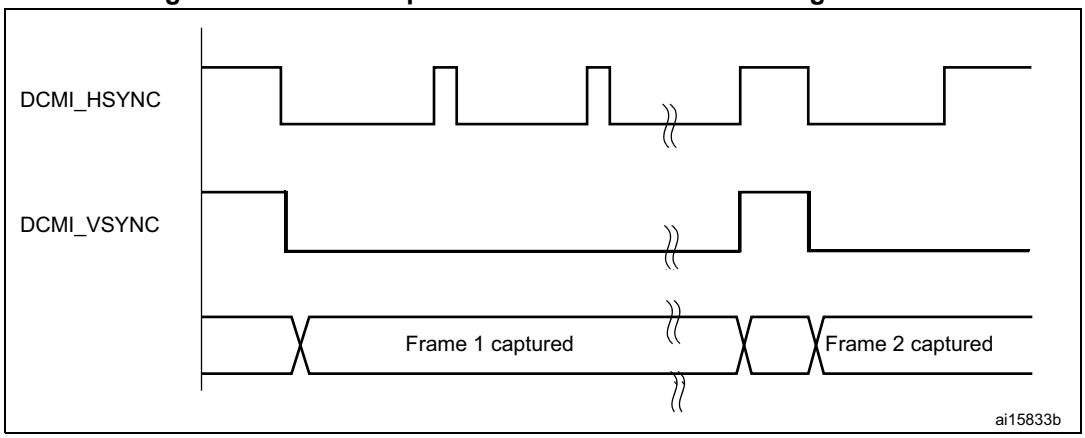


1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

Continuous grab mode

In this mode (CM bit = 0 in DCMI_CR), once the CAPTURE bit has been set in DCMI_CR, the grabbing process starts on the next DCMI_VSYNC or embedded frame start depending on the mode. The process continues until the CAPTURE bit is cleared in DCMI_CR. Once the CAPTURE bit has been cleared, the grabbing process continues until the end of the current frame.

Figure 305. Frame capture waveforms in continuous grab mode



1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

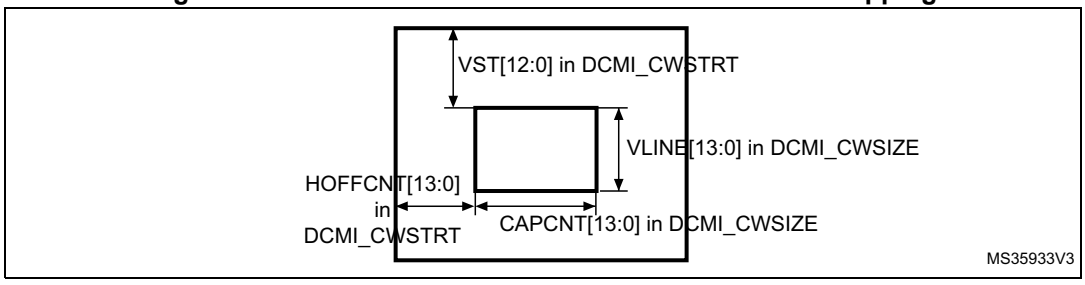
In continuous grab mode, the FCRC[1:0] bits in DCMI_CR can be configured to grab all pictures, every second picture or one out of four pictures to decrease the frame capture rate.

Note: In the hardware synchronization mode (ESS = 0 in DCMI_CR), the IT_VSYNC interrupt is generated (if enabled) even when CAPTURE = 0 in DCMI_CR so, to reduce the frame capture rate even further, the IT_VSYNC interrupt can be used to count the number of frames between 2 captures in conjunction with the Snapshot mode. This is not allowed by embedded data synchronization mode.

37.3.8 DCMI crop feature

With the crop feature, the camera interface can select a rectangular window from the received image. The start (upper left corner) coordinates and size (horizontal dimension in number of pixel clocks and vertical dimension in number of lines) are specified using two 32-bit registers (DCMI_CWSTRT and DCMI_CWSIZE). The size of the window is specified in number of pixel clocks (horizontal dimension) and in number of lines (vertical dimension).

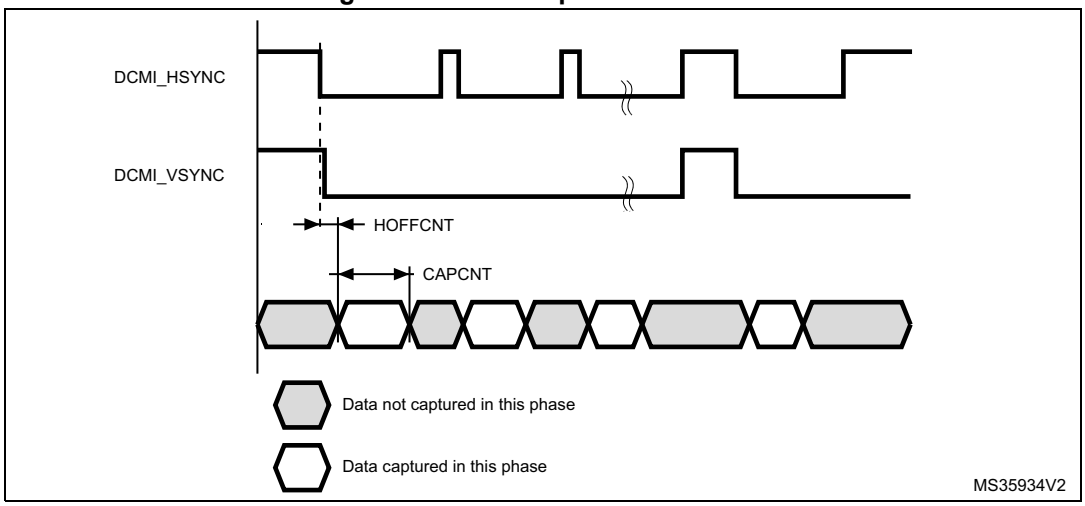
Figure 306. Coordinates and size of the window after cropping



These registers specify the coordinates of the starting point of the capture window as a line number (in the frame, starting from 0) and a number of pixel clocks (on the line, starting from 0), and the size of the window as a line number and a number of pixel clocks. The CAPCNT value can only be a multiple of 4 (two least significant bits are forced to 0) to allow the correct transfer of data through the DMA.

If the DCMI_VSYNC signal goes active before the number of lines is specified in the DCMI_CWSIZE register, then the capture stops and an IT_FRAME interrupt is generated when enabled.

Figure 307. Data capture waveforms



1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

37.3.9 DCMI JPEG format

To allow JPEG image reception, it is necessary to set the JPEG bit of the DCMI_CR register. JPEG images are not stored as lines and frames, so the DCMI_VSYNC signal is used to start the capture while DCMI_HSYNC serves as a data enable signal. The number of bytes in a line may not be a multiple of 4. This case must be carefully handled since a DMA request is generated each time a complete 32-bit word has been constructed from the captured data. When an end of frame is detected and the 32-bit word to be transferred has not been completely received, the remaining data are padded with zeros and a DMA request is generated.

The crop feature and embedded synchronization codes cannot be used in JPEG format.

37.3.10 DCMI FIFO

A 8-word FIFO is implemented to manage data rate transfers on the AHB. The DCMI features a simple FIFO controller with a read pointer incremented each time the camera interface reads from the AHB, and a write pointer incremented each time the camera interface writes to the FIFO. There is no overrun protection to prevent the data from being overwritten if the AHB interface does not sustain the data transfer rate.

In case of overrun or errors in the synchronization signals, the FIFO is reset and the DCMI interface waits for a new start of frame.

37.3.11 DCMI data format description

Data formats

Three types of data are supported:

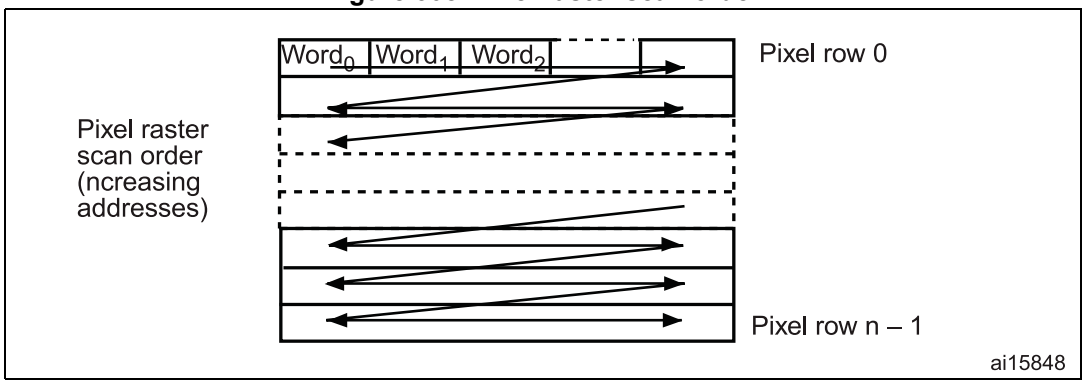
- 8/10/12/14-bit progressive video: either monochrome or raw Bayer format
- YCbCr 4:2:2 progressive video
- RGB565 progressive video. A pixel coded in 16 bits (5 bits for blue, 5 bits for red, 6 bits for green) takes two clock cycles to be transferred.

Compressed data: JPEG

For B&W (black and white), YCbCr or RGB data, the maximum input size is 2048 × 2048 pixels. No limit in JPEG compressed mode.

For monochrome, RGB and YCbCr, the frame buffer is stored in raster mode. 32-bit words are used. Only the little-endian format is supported.

Figure 308. Pixel raster scan order



Monochrome format

Characteristics:

- Raster format
- 8 bits per pixel

The table below shows how the data are stored.

Table 338. Data storage in monochrome progressive video format

Byte address	31:24	23:16	15:8	7:0
0	n + 3	n + 2	n + 1	n
4	n + 7	n + 6	n + 5	n + 4

RGB format

Characteristics:

- Raster format
- RGB
- Interleaved: one buffer: R, G and B interleaved (such as BRGBRGBRG)
- Optimized for display output

The RGB planar format is compatible with standard OS frame buffer display formats.

Only 16 BPP (bits per pixel): RGB565 (2 pixels per 32-bit word) is supported.

The 24 BPP (palletized format) and gray-scale formats are not supported. Pixels are stored in a raster scan order, that is from top to bottom for pixel rows, and from left to right within a pixel row. Pixel components are R (red), G (green) and B (blue). All components have the same spatial resolution (4:4:4 format). A frame is stored in a single part, with the components interleaved on a pixel basis.

The table below shows how the data are stored.

Table 339. Data storage in RGB progressive video format

Byte address	31:27	26:21	20:16	15:11	10:5	4:0
0	Red n + 1	Green n + 1	Blue n + 1	Red n	Green n	Blue n
4	Red n + 4	Green n + 3	Blue n + 3	Red n + 2	Green n + 2	Blue n + 2

YCbCr format

Characteristics:

- Raster format
- YCbCr 4:2:2
- Interleaved: one buffer: Y, Cb and Cr interleaved (such as CbYCrYCbYCr)

Pixel components are Y (luminance or “luma”), Cb and Cr (chrominance or “chroma” blue and red). Each component is encoded in 8 bits. Luma and chroma are stored together (interleaved) as shown in the table below.

Table 340. Data storage in YCbCr progressive video format

Byte address	31:24	23:16	15:8	7:0
0	Y n + 1	Cr n	Y n	Cb n
4	Y n + 3	Cr n + 2	Y n + 2	Cb n + 2

YCbCr format - Y only

Characteristics:

- Raster format
- YCbCr 4:2:2
- The buffer only contains Y information - monochrome image

Pixel components are Y (luminance or “luma”), Cb and Cr (chrominance or “chroma” blue and red). In this mode, the chroma information is dropped. Only the luma component of each pixel, encoded in 8 bits, is stored as shown in *Table 341*.

The result is a monochrome image having the same resolution as the original YCbCr data.

Table 341. Data storage in YCbCr progressive video format - Y extraction mode

Byte address	31:24	23:16	15:8	7:0
0	Y n + 3	Y n + 2	Y n + 1	Y n
4	Y n + 7	Y n + 6	Y n + 5	Y n + 4

Half resolution image extraction

This is a modification of the previous reception modes, being applicable to monochrome, RGB or Y extraction modes.

This mode is used to only store a half resolution image. It is selected through OELS and LSM control bits.

37.4 DCMI interrupts

Five interrupts are generated. All interrupts are maskable by software. The global interrupt (dcmi_it) is the OR of all the individual interrupts. The table below gives the list of all interrupts.

Table 342. DCMI interrupts

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exits Sleep mode	Exists Stop and Standby modes
dcmi_it	End of line	LINE_RIS	LINE_IE	Set LINE_ISC	Yes	No
	End of frame capture	FRAME_RIS	FRAME_IE	Set FRAME_ISC	Yes	No
	Overrun of data reception	OVR_RIS	OVR_IE	Set OVR_ISC	Yes	No
	Synchronization frame	VSYNC_RIS	VSYNC_IE	Set VSYNC_ISC	Yes	No
	Detection of an error in the embedded synchronization frame detection	ERR_RIS	ERR_IE	Set ERR_ISC	Yes	No

37.5 DCMI registers

Refer to *Section 1.2 on page 104* for list of abbreviations used in register descriptions. All DCMI registers must be accessed as 32-bit words, otherwise a bus error occurs.

37.5.1 DCMI control register (DCMI_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.	OELS	LSM	OEBS	BSM[1:0]	
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ENABLE	Res.	Res.	EDM[1:0]		FCRC[1:0]		VSPOL	HSPOL	PCKPOL	ESS	JPEG	CROP	CM	CAPTURE
	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 **OELS:** Odd/Even Line Select (Line Select Start)
This bit works in conjunction with the LSM field (LSM = 1).
0: Interface captures first line after the frame start, second one being dropped.
1: Interface captures second line from the frame start, first one being dropped.

Bit 19 **LSM:** Line Select mode
0: Interface captures all received lines.
1: Interface captures one line out of two.

Bit 18 **OEBS:** Odd/Even Byte Select (Byte Select Start)
This bit works in conjunction with BSM field (BSM $\neq$ 00).
0: Interface captures first data (byte or double byte) from the frame/line start, second one being dropped.
1: Interface captures second data (byte or double byte) from the frame/line start, first one being dropped.

Bits 17:16 **BSM[1:0]:** Byte Select mode
00: Interface captures all received data.
01: Interface captures every other byte from the received data.
10: Interface captures one byte out of four.
11: Interface captures two bytes out of four.
Note: This mode only works for EDM[1:0] = 00. For all other EDM values, this field must be programmed to the reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **ENABLE:** DCMI enable
0: DCMI disabled
1: DCMI enabled
Note: The DCMI configuration registers must be programmed correctly before enabling this bit.

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:10 **EDM[1:0]:** Extended data mode
00: Interface captures 8-bit data on every pixel clock.
01: Interface captures 10-bit data on every pixel clock.
10: Interface captures 12-bit data on every pixel clock.
11: Interface captures 14-bit data on every pixel clock.

Bits 9:8 **FCRC[1:0]:** Frame capture rate control
These bits define the frequency of frame capture. They are meaningful only in Continuous grab mode. They are ignored in snapshot mode.
00: All frames are captured.
01: Every alternate frame captured (50% bandwidth reduction)
10: One frame out of four captured (75% bandwidth reduction)
11: reserved

Bit 7 **VSPOL:** Vertical synchronization polarity
This bit indicates the level on the DCMI_VSYNC pin when the data are not valid on the parallel interface.
0: DCMI_VSYNC active low
1: DCMI_VSYNC active high

Bit 6 **HSPOL:** Horizontal synchronization polarity
This bit indicates the level on the DCMI_HSYNC pin when the data are not valid on the parallel interface.
0: DCMI_HSYNC active low
1: DCMI_HSYNC active high

Bit 5 **PCKPOL:** Pixel clock polarity
This bit configures the capture edge of the pixel clock.
0: Falling edge active
1: Rising edge active

Bit 4 **ESS:** Embedded synchronization select
0: Hardware synchronization data capture (frame/line start/stop) is synchronized with the DCMI_HSYNC/DCMI_VSYNC signals.
1: Embedded synchronization data capture is synchronized with synchronization codes embedded in the data flow.
Note: Valid only for 8-bit parallel data. HSPOL/VSPOL are ignored when the ESS bit is set.
This bit is disabled in JPEG mode.

Bit 3 **JPEG:** JPEG format
0: Uncompressed video format
1: This bit is used for JPEG data transfers. The DCMI_HSYNC signal is used as data enable. The crop and embedded synchronization features (ESS bit) cannot be used in this mode.

Bit 2 **CROP:** Crop feature
0: The full image is captured. In this case the total number of bytes in an image frame must be a multiple of four.
1: Only the data inside the window specified by the crop register is captured. If the size of the crop window exceeds the picture size, then only the picture size is captured.

Bit 1 **CM:** Capture mode
0: Continuous grab mode - The received data are transferred into the destination memory through the DMA. The buffer location and mode (linear or circular buffer) is controlled through the system DMA.
1: Snapshot mode (single frame) - Once activated, the interface waits for the start of frame and then transfers a single frame through the DMA. At the end of the frame, the CAPTURE bit is automatically reset.

Bit 0 **CAPTURE:** Capture enable

0: Capture disabled

1: Capture enabled

The camera interface waits for the first start of frame, then a DMA request is generated to transfer the received data into the destination memory.

In snapshot mode, the CAPTURE bit is automatically cleared at the end of the first frame received.

In continuous grab mode, if the software clears this bit while a capture is ongoing, the bit is effectively cleared after the frame end.

Note: The DMA controller and all DCMI configuration registers must be programmed correctly before enabling this bit.

37.5.2 DCMI status register (DCMI_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.	Res.	FNE	VSYNC	HSYNC
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **FNE**: FIFO not empty

This bit gives the status of the FIFO.

1: FIFO contains valid data.

0: FIFO empty

Bit 1 **VSYNC**: Vertical synchronization

This bit gives the state of the DCMI_VSYNC pin with the correct programmed polarity. When embedded synchronization codes are used, the meaning of this bit is the following:

0: active frame

1: synchronization between frames

In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMI_CR is set.

Bit 0 **HSYNC**: Horizontal synchronization

This bit gives the state of the DCMI_HSYNC pin with the correct programmed polarity. When embedded synchronization codes are used, the meaning of this bit is the following:

0: active line

1: synchronization between lines

In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMI_CR is set.

37.5.3 DCMI raw interrupt status register (DCMI_RIS)

DCMI_RIS gives the raw interrupt status and is accessible in read only. When read, this register returns the status of the corresponding interrupt before masking with the DCMI_IER register value.

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.	LINE_RIS	VSYNC_RIS	ERR_RIS	OVR_RIS	FRAME_RIS
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **LINE_RIS**: Line raw interrupt status

This bit gets set when the DCMI_HSYNC signal changes from the inactive state to the active state. It goes high even if the line is not valid.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit in DCMI_CR is set.

It is cleared by setting the LINE_ISC bit of the DCMI_ICR register.

Bit 3 **VSYNC_RIS**: DCMI_VSYNC raw interrupt status

This bit is set when the DCMI_VSYNC signal changes from the inactive state to the active state.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit is set in DCMI_CR.

It is cleared by setting the VSYNC_ISC bit of the DCMI_ICR register.

Bit 2 **ERR_RIS**: Synchronization error raw interrupt status

0: No synchronization error detected

1: Embedded synchronization characters are not received in the correct order.

This bit is valid only in the embedded synchronization mode. It is cleared by setting the ERR_ISC bit of the DCMI_ICR register.

Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_RIS**: Overrun raw interrupt status

0: No data buffer overrun occurred

1: A data buffer overrun occurred and the data FIFO is corrupted.

The bit is cleared by setting the OVR_ISC bit of the DCMI_ICR register.

Bit 0 **FRAME_RIS**: Capture complete raw interrupt status

0: No new capture

1: A frame has been captured.

This bit is set when a frame or window has been captured.

In case of a cropped window, this bit is set at the end of line of the last line in the crop. It is set even if the captured frame is empty (e.g. window cropped outside the frame).

The bit is cleared by setting the FRAME_ISC bit of the DCMI_ICR register.

37.5.4 DCMI interrupt enable register (DCMI_IER)

The DCMI_IER register is used to enable interrupts. When one of the DCMI_IER bits is set, the corresponding interrupt is enabled. This register is accessible in both read and write.

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.	LINE_IE	VSYNC_IE	ERR_IE	OVR_IE	FRAME_IE
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **LINE_IE**: Line interrupt enable
0: No interrupt generation when the line is received
1: An Interrupt is generated when a line has been completely received.

Bit 3 **VSYNC_IE**: DCMI_VSYNC interrupt enable
0: No interrupt generation
1: An interrupt is generated on each DCMI_VSYNC transition from the inactive to the active state.
The active state of the DCMI_VSYNC signal is defined by the VSPOL bit.

Bit 2 **ERR_IE**: Synchronization error interrupt enable
0: No interrupt generation
1: An interrupt is generated if the embedded synchronization codes are not received in the correct order.
Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_IE**: Overrun interrupt enable
0: No interrupt generation
1: An interrupt is generated if the DMA was not able to transfer the last data before new data (32-bit) are received.

Bit 0 **FRAME_IE**: Capture complete interrupt enable
0: No interrupt generation
1: An interrupt is generated at the end of each received frame/crop window (in crop mode).

37.5.5 DCMI masked interrupt status register (DCMI_MIS)

This DCMI_MIS register is a read-only register. When read, it returns the current masked status value (depending on the value in DCMI_IER) of the corresponding interrupt. A bit in this register is set if the corresponding enable bit in DCMI_IER is set and the corresponding bit in DCMI_RIS is set.

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.	LINE_MIS	VSYNC_MIS	ERR_MIS	OVR_MIS	FRAME_MIS
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **LINE_MIS**: Line masked interrupt status

This bit gives the status of the masked line interrupt.

0: No interrupt generation when the line is received

1: An Interrupt is generated when a line has been completely received and the LINE_IE bit is set in DCMI_IER.

Bit 3 **VSYNC_MIS**: VSYNC masked interrupt status

This bit gives the status of the masked VSYNC interrupt.

0: No interrupt is generated on DCMI_VSYNC transitions.

1: An interrupt is generated on each DCMI_VSYNC transition from the inactive to the active state and the VSYNC_IE bit is set in DCMI_IER.

The active state of the DCMI_VSYNC signal is defined by the VSPOL bit.

Bit 2 **ERR_MIS**: Synchronization error masked interrupt status

This bit gives the status of the masked synchronization error interrupt.

0: No interrupt is generated on a synchronization error.

1: An interrupt is generated if the embedded synchronization codes are not received in the correct order and the ERR_IE bit in DCMI_IER is set.

Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_MIS**: Overrun masked interrupt status

This bit gives the status of the masked overflow interrupt.

0: No interrupt is generated on overrun.

1: An interrupt is generated if the DMA was not able to transfer the last data before new data (32-bit) are received and the OVR_IE bit is set in DCMI_IER.

Bit 0 **FRAME_MIS**: Capture complete masked interrupt status

This bit gives the status of the masked capture complete interrupt

0: No interrupt is generated after a complete capture.

1: An interrupt is generated at the end of each received frame/crop window (in crop mode) and the FRAME_IE bit is set in DCMI_IER.

37.5.6 DCMI interrupt clear register (DCMI_ICR)

The DCMI_ICR register is write-only. Setting a bit of this register clears the corresponding flag in the DCMI_RIS and DCMI_MIS registers. Writing 0 has no effect.

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.	Res.	Res.	Res.	Res.	LINE_ISC	VSYNC_ISC	ERR_ISC	OVR_ISC	FRAME_ISC
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **LINE_ISC**: line interrupt status clear
Setting this bit clears the LINE_RIS flag in the DCMI_RIS register.

Bit 3 **VSYNC_ISC**: Vertical Synchronization interrupt status clear
Setting this bit clears the VSYNC_RIS flag in the DCMI_RIS register.

Bit 2 **ERR_ISC**: Synchronization error interrupt status clear
Setting this bit clears the ERR_RIS flag in the DCMI_RIS register.
Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_ISC**: Overrun interrupt status clear
Setting this bit clears the OVR_RIS flag in the DCMI_RIS register.

Bit 0 **FRAME_ISC**: Capture complete interrupt status clear
Setting this bit clears the FRAME_RIS flag in the DCMI_RIS register.

37.5.7 DCMI embedded synchronization code register (DCMI_ESCR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FEC[7:0]								LEC[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
LSC[7:0]								FSC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **FEC[7:0]**: Frame end delimiter code

This byte specifies the code of the frame end delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, FEC.

If FEC is programmed to 0xFF, all the unused codes (0xFF0000XY) are interpreted as frame end delimiters.

Bits 23:16 **LEC[7:0]**: Line end delimiter code

This byte specifies the code of the line end delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, LEC.

Bits 15:8 **LSC[7:0]**: Line start delimiter code

This byte specifies the code of the line start delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, LSC.

Bits 7:0 **FSC[7:0]**: Frame start delimiter code

This byte specifies the code of the frame start delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, FSC.

If FSC is programmed to 0xFF, no frame start delimiter is detected. But, the first occurrence of LSC after an FEC code is interpreted as a start of frame delimiter.

37.5.8 DCMI embedded synchronization unmask register (DCMI_ESUR)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FEU[7:0]								LEU[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
LSU[7:0]								FSU[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **FEU[7:0]**: Frame end delimiter unmask

This byte specifies the mask to be applied to the code of the frame end delimiter.

0: The corresponding bit in the FEC byte in DCMI_ESCR is masked while comparing the frame end delimiter with the received data.

1: The corresponding bit in the FEC byte in DCMI_ESCR is compared while comparing the frame end delimiter with the received data.

Bits 23:16 **LEU[7:0]**: Line end delimiter unmask

This byte specifies the mask to be applied to the code of the line end delimiter.

0: The corresponding bit in the LEC byte in DCMI_ESCR is masked while comparing the line end delimiter with the received data.

1: The corresponding bit in the LEC byte in DCMI_ESCR is compared while comparing the line end delimiter with the received data.

Bits 15:8 **LSU[7:0]**: Line start delimiter unmask

This byte specifies the mask to be applied to the code of the line start delimiter.

0: The corresponding bit in the LSC byte in DCMI_ESCR is masked while comparing the line start delimiter with the received data.

1: The corresponding bit in the LSC byte in DCMI_ESCR is compared while comparing the line start delimiter with the received data.

Bits 7:0 **FSU[7:0]**: Frame start delimiter unmask

This byte specifies the mask to be applied to the code of the frame start delimiter.
0: The corresponding bit in the FSC byte in DCMI_ESCR is masked while comparing the frame start delimiter with the received data.
1: The corresponding bit in the FSC byte in DCMI_ESCR is compared while comparing the frame start delimiter with the received data.

37.5.9 DCMI crop window start (DCMI_CWSTRT)

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.	VST[12:0]												
			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.	HOFFCNT[13:0]													
		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 28:16 **VST[12:0]**: Vertical start line count

The image capture starts with this line number. Previous line data are ignored.
0x0000: line 1
0x0001: line 2
0x0002: line 3
....

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **HOFFCNT[13:0]**: Horizontal offset count

This value gives the number of pixel clocks to count before starting a capture.

37.5.10 DCMI crop window size (DCMI_CWSIZE)

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.	VLINE[13:0]													
		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.	Res.	CAPCNT[13:0]													
		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 29:16 **VLINE[13:0]**: Vertical line count

This value gives the number of lines to be captured from the starting point.
0x0000: 1 line
0x0001: 2 lines
0x0002: 3 lines
....

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **CAPCNT[13:0]**: Capture count

This value gives the number of pixel clocks to be captured from the starting point on the same line. It value must corresponds to word-aligned data for different widths of parallel interfaces.
0x0000: 1 pixel
0x0001: 2 pixels
0x0002: 3 pixels
....

37.5.11 DCMI data register (DCMI_DR)

Address offset: 0x28

Reset value: 0x0000 0000

The digital camera Interface packages all the received data in 32-bit format before requesting a DMA transfer. A 8-word deep FIFO is available to leave enough time for DMA transfers and avoid DMA overrun conditions.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BYTE3[7:0]								BYTE2[7:0]							
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
BYTE1[7:0]								BYTE0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **BYTE3[7:0]**: Data byte 3

Bits 23:16 **BYTE2[7:0]**: Data byte 2

Bits 15:8 **BYTE1[7:0]**: Data byte 1

Bits 7:0 **BYTE0[7:0]**: Data byte 0

37.5.12 DCMI register map

Table 343. DCMI 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	DCMI_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OELS	LSM	OEBS	BSM[1:0]		Res.	ENABLE	Res.	Res.	EDM[1:0]		FCRC[1:0]		VSPOL	HSPOL	PCKPOL	ESS	JPEG	CROP	CM	CAPTURE
	Reset value												0	0	0	0	0		0			0	0	0	0	0	0	0	0	0	0	0	0

Table 343. DCMI 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
0x04	DCMI_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.	FNE	VSYNC	HSYNC
	Reset value																														0	0	0
0x08	DCMI_RIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_RIS	VSYNC_RIS	ERR_RIS	OVR_RIS	FRAME_RIS
	Reset value																												0	0	0	0	0
0x0C	DCMI_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.	LINE_IE	VSYNC_IE	ERR_IE	OVR_IE	FRAME_IE
	Reset value																												0	0	0	0	0
0x10	DCMI_MIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_MIS	VSYNC_MIS	ERR_MIS	OVR_MIS	FRAME_MIS
	Reset value																												0	0	0	0	0
0x14	DCMI_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.	LINE_ISC	VSYNC_ISC	ERR_ISC	OVR_ISC	FRAME_ISC
	Reset value																												0	0	0	0	0
0x18	DCMI_ESCR	FEC[7:0]								LEC[7:0]								LSC[7:0]								FSC[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
0x1C	DCMI_ESUR	FEU[7:0]								LEU[7:0]								LSU[7:0]								FSU[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
0x20	DCMI_CWSTRT	Res.	Res.	Res.	VST[12:0]													Res.	Res.	HOFFCNT[13:0]													
	Reset value				0	0	0	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	DCMI_CWSIZE	Res.	Res.	VLINE[13:0]														Res.	Res.	CAPCNT[13:0]													
	Reset value			0	0	0	0	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	DCMI_DR	BYTE3[7:0]								BYTE2[7:0]								BYTE1[7:0]								BYTE0[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

Refer to *Section 2.3* for the register boundary addresses.

38 Parallel synchronous slave interface (PSSI)

The PSSI peripheral and the DCMI (digital camera interface) use the same circuitry. As a result, these two peripherals cannot be used at the same time: when using the PSSI, the DCMI registers cannot be accessed, and vice-versa.

In addition, the PSSI and the DCMI share the same alternate functions and interrupt vector (see *Section 38.3.2: PSSI pins and internal signals*).

38.1 Introduction

The PSSI is a generic synchronous 8/16-bit parallel data input/output slave interface. It enables the transmitter to send a data valid signal that indicates when the data is valid, and the receiver to output a flow control signal that indicates when it is ready to sample the data.

38.2 PSSI main features

The PSSI peripheral main features are the following:

- Slave mode operation
- 8-bit or 16-bit parallel data input or output
- 8-word (32-byte) FIFO
- Data enable (PSSI_DE) alternate function input and Ready (PSSI_RDY) alternate function output

  When selected, these signals can either enable the transmitter to indicate when the data is valid, allow the receiver to indicate when it is ready to sample the data, or both.

38.3 PSSI functional description

The PSSI is a synchronous parallel slave interface that can send or receive high-speed data flows. It consists of up to 16 data lines (PSSI_D[15:0]) plus a clock line (PSSI_PDCK). The clock polarity can be configured so that data can be captured or transmitted on either the clock rising or falling edge.

Usually, a general-purpose DMA channel is used to pass 32-bit packed data via the data register (PSSI_DR).

The data flow can either be continuous or synchronized by hardware using the optional PSSI_DE (Data enable), and PSSI_RDY (Ready) signals.

Figure 309 shows the PSSI block diagram.

38.3.1 PSSI block diagram

Figure 309. PSSI block diagram

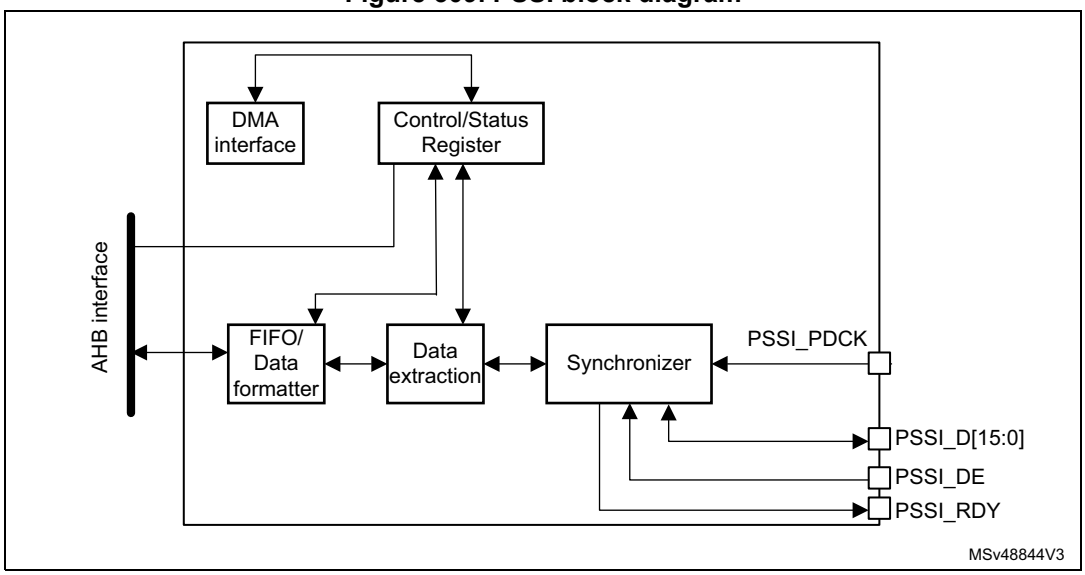
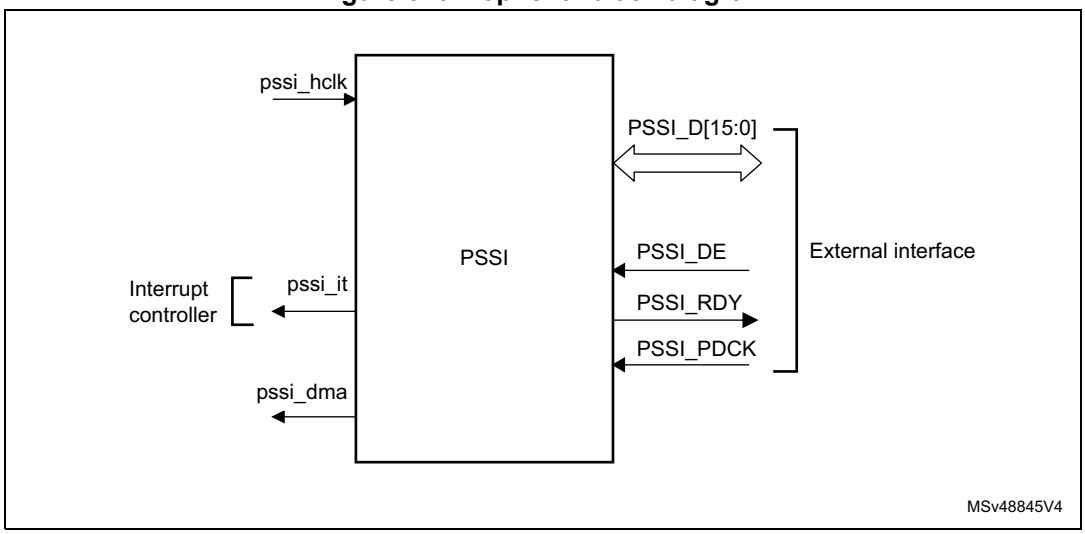


Figure 310. Top-level block diagram



38.3.2 PSSI pins and internal signals

The PSSI interface is composed of 19 pins, though nine signals are enough to transfer parallel data. *Table 344* shows the PSSI pins.

When the PSSI ENABLE bit (bit 14 of PSSI_CR) is set to 1, the alternate functions and the interrupt vector are associated with the PSSI. Otherwise, they are associated with the DCMI. The DCMI ENABLE bit (bit 15 of DCMI_CR) and the PSSI ENABLE bit (bit 14 of PSSI_CR) must not be set to 1 at the same time. As an example, if a GPIO is configured to use the alternate function PSSI_PDCK/DCMI_PIXCK, it is the PSSI_PDCK function which becomes active if PSSI_CR/ENABLE is set to 1.

Table 344. PSSI input/output pins

PSSI signal name	DCMI signal it is shared with	Signal type	Description
PSSI_PDCK	DCMI_PIXCK	Input	Parallel data clock input
PSSI_D[15:0]	DCMI_D[13:0]	Input/output	Data output when transmitting, data input when receiving
PSSI_DE	DCMI_HSYNC	Input	Data enable signal: data valid signal when receiving or flow control signal when transmitting
PSSI_RDY	DCMI_VSYNC	Output	Ready signal: flow control signal when receiving or data valid signal when transmitting

Table 345 shows the PSSI internal input/output signals.

Table 345. PSSI internal input/output signals

Internal signal name	Signal type	Description
pssi_it	Output	Interrupt
pssi_dma	Output	DMA request
pssi_hclk	Input	AHB clock

38.3.3 PSSI clock

The AHB clock frequency must be at least 2.5 times higher than the PSSI_PDCK frequency. At frequency ratios lower than 2.5, data might be corrupted or lost during transfers.

Data transfers are synchronous with PSSI_PDCK. The PSSI_PDCK polarity can be configured as follows, through CKPOL bit (bit 5 of PSSI_CR):

- When CKPOL = 0
  - Input pins are sampled on PSSI_PDCK falling edge
  - Output pins are driven on PSSI_PDCK rising edge
- When CKPOL = 1
  - Input pins are sampled on PSSI_PDCK rising edge
  - Output pins are driven on PSSI_PDCK falling edge

38.3.4 PSSI data management

Data direction

The direction of data transfers is configured through the OUTEN control bit (bit 31 of PSSI_CR):

- When OUTEN is cleared to 0 (default setting), the PSSI operates in receive mode and the data is input on the data pins.
- When OUTEN is set to 1, the peripheral operates in transmit mode and the data is output on the data pins.

OUTEN can be modified only when the ENABLE bit is cleared to 0.

Data register and DMA

Data are transferred from/to the FIFO using the PSSI_DR data register:

- In receive mode, data must be read from the FIFO by reading PSSI_DR.
- In transmit mode, data must be written to the FIFO by writing into PSSI_DR.

Word (32-bit) accesses to PSSI_DR and half-word (16-bit) accesses to PSSI_DR[15:0] are permitted in all modes. Byte (8-bit) accesses to PSSI_DR[7:0] are permitted only when the PSSI is configured to transfer 8 bits at a time (EDM=00 in the PSSI_CR register).

To reduce the load on the CPU, it is recommended to use the DMA to transfer data from/to the PSSI FIFO. When it is used, the DMA must be configured to transfer data via the PSSI_DR register. Using 32-bit transfers optimizes bandwidth and reduces the bus load. However, 8-bit and 16-bit transfers are also permitted.

To use the DMA, set the PSSI DMA enable bit (DMAEN in PSSI_CR) to 1 (default setting). When DMAEN is set to 1, a DMA transfer is initiated when the FIFO is ready for a 32-bit transfer (four valid bytes in receive mode or four empty bytes in transmit mode). As a result, in receive mode, no DMA transfers are initiated if there are three bytes or fewer in the FIFO, even if the DMA is configured to perform 8-bit transfers.

The RTT4B and RTT1B status bits (PSSI_SR) are useful when the CPU directly perform transfers to and from the FIFO. RTT4B set to 1 indicates that the FIFO is ready to transfer four bytes: at least four valid bytes in the FIFO in receive mode or at least four free bytes in transmit mode. RTT1B set to 1 indicates that the FIFO is ready to transfer one byte: at least one valid byte in the FIFO in receive mode or at least one free byte in transmit mode.

8-bit data

The PSSI parallel interface can transfer either 8-bit (using D[7:0]) or 16-bit data (using D[15:0]) depending on the EDM[1:0] control bits (bits 11:10 of PSSI_CR). If the 8-bit configuration is selected (EDM[1:0] set to 00), the unused D[15:0] pins can be used for GPIO or other functions.

When EDM[1:0] in PSSI_CR are programmed to 00, the interface transfers 8 bits using the D[7:0] pins. In this case, D[15:8] are not used and four PSSI_PDCK cycles are required to transfer a 32-bit word.

The least-significant byte (bits 7:0) correspond to the first byte transferred, and the most-significant byte (bits 31:28) corresponds to the forth byte transferred. *Table 346* illustrates the positioning of the data bytes in two 32-bit words.

Table 346. Positioning of captured data bytes in 32-bit words (8-bit width)

Byte address	31:24	23:16	15:8	7:0
0	D_{n+3}[7:0]	D_{n+2}[7:0]	D_{n+1}[7:0]	D_n[7:0]
4	D_{n+7}[7:0]	D_{n+6}[7:0]	D_{n+5}[7:0]	D_{n+4}[7:0]

16-bit data

When EDM[1:0] in PSSI_CR are programmed to 11, the interface transfers 16 bits using the D[15:0] pins. In this case, two PSSI_PDCK cycles are required to transfer a 32-bit word.

The least-significant half word (bits 15:0) correspond to the first half word transferred, and the most-significant half-word (bits 31:16) corresponds to the second half word transferred. *Table 347* illustrates the positioning of the data in two 32-bit words.

Table 347. Positioning of captured data bytes in 32-bit words (16-bit width)

Byte address	31:16	15:0
0	D_{n+1}[15:0]	D_n[15:0]
4	D_{n+3}[15:0]	D_{n+2}[15:0]

FIFO data buffer and error conditions

An eight-word FIFO helps improving performance and avoids overruns and underruns.

If the ready signal (PSSI_RDY) is disabled in receive mode, an overrun error is generated when a clock active edge occurs when the FIFO is full. In this case, the input data is lost.

If the data enable signal (PSSI_DE) is disabled in transmit mode, an underrun error is generated when a clock active edge occurs when the FIFO is empty. In this case, unpredictable data are output.

The OVR_RIS status bit indicates that either an overrun or an underrun occurred. An interrupt can be generated when these events occur.

38.3.5 PSSI optional control signals

Data Enable (PSSI_DE) alternate function input

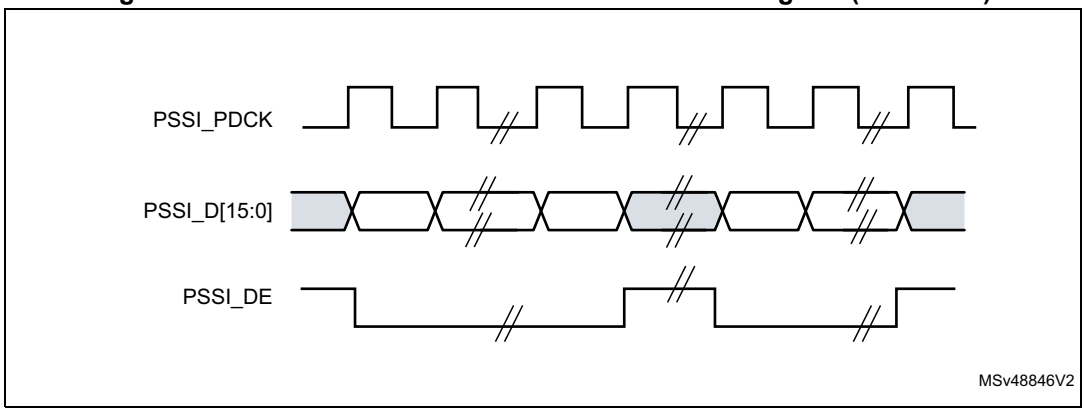
The data enable signal, PSSI_DE, is an optional signal. It is driven by the data source/transmitter in order to indicate that the data is valid to be transferred during the current cycle. When PSSI_DE is inactive, it means that the data must not be sampled by the receiver at the next clock edge.

This alternate function signal can be enabled using the DERDYCFG (bits 20:18 of PSSI_CR) control bits. PSSI_DE polarity is configured through DEPOL control bit (bit 6 of PSSI_CR). PSSI_DE is active low when DEPOL is cleared to 0, and high when DEPOL is et to 1.

The direction of the PSSI_DE signal is defined by the OUTEN value. It is the same as the data direction.

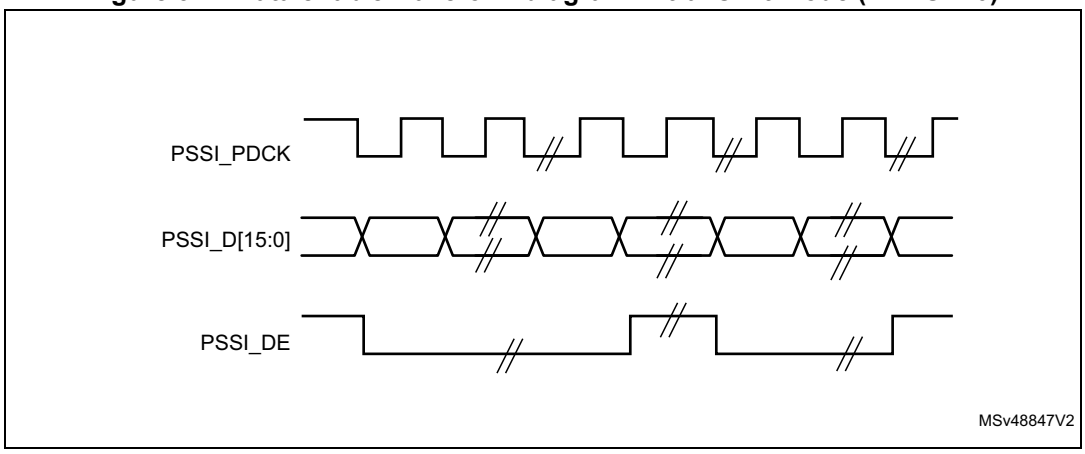
If the PSSI_DE alternate function input is enabled (through DERDYCFG) in receive mode (OUTEN cleared to 0), the PSSI samples PSSI_DE on the same PSSI_PDCK edge as the one used for sampling the data (D[15:0]). If PSSI_DE is active, the sampled data is saved in the FIFO. Otherwise, the sampled data is considered invalid and discarded. The transmitting device can use PSSI_DE as a data valid signal, driving it inactive when the data in the current cycle is not valid. This flow control function allows avoiding underrun errors.

Figure 311. Data enable in receive mode waveform diagram (CKPOL=0)



If the PSSI_DE alternate output function is enabled (through DERDYCFG) in transmit mode (OUTEN=1), the PSSI drives PSSI_DE on the same PSSI_PDCK edge that the one used to drive the data (D[15:0]). If a new 8 or 16-bit data (as programmed in the EDM[1:0] control bits in PSSI_CR) is available for transmission in the internal FIFO, this data is output on the data outputs (D[15:0]) and the PSSI_DE output becomes active on the current PSSI_PDCK edge. Otherwise (if the TX FIFO is empty), the D[15:0] outputs remains unchanged on the next clock edge and the PSSI_DE output becomes inactive.

Figure 312. Data enable waveform diagram in transmit mode (CKPOL=0)



Ready (PSSI_RDY) alternate function output

The ready signal, PSSI_RDY, is an optional signal. It is driven by the receiving device and indicates whether data is being accepted in the current cycle. When PSSI_RDY is inactive, it means that the data must not be sampled by the receiver at the next clock edge.

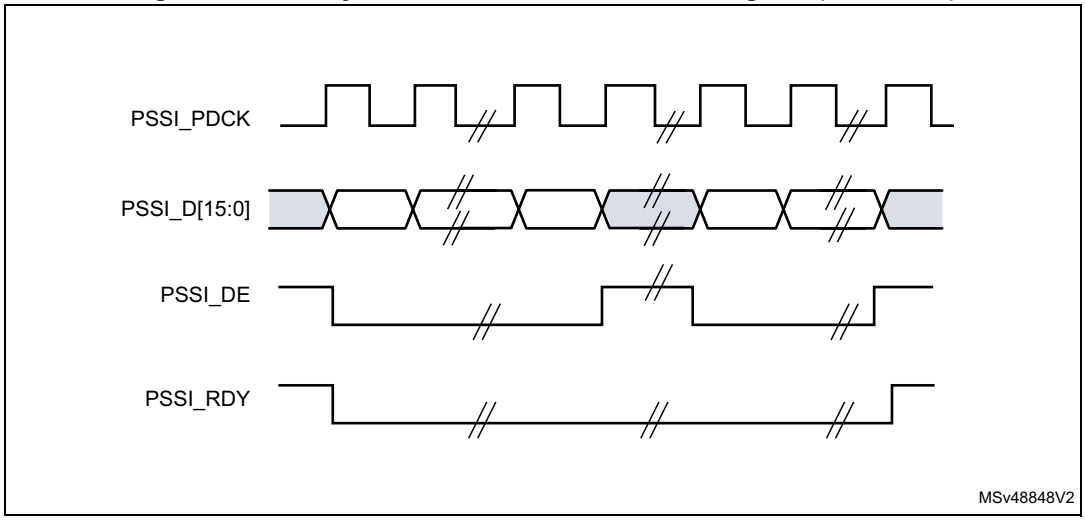
This alternate function signal can be enabled using the DERDYCFG control bits (bits 20:18 of PSSI_CR). PSSI_RDY polarity is configured through the RDYPOL control bit (bit 6 of PSSI_CR). PSSI_RDY is active low when RDYPOL is cleared to 0, and high when RDYPOL set to 1.

The direction of the PSSI_RDY signal is defined by the OUTEN (bit 31 of PSSI_CR). It is set in the opposite direction compared to the PSSI_DE and data signals.

If the PSSI_RDY alternate output function is enabled (through DERDYCFG) in receive mode (OUTEN=0), the PSSI drives PSSI_RDY one PSSI_PDCK half cycle after it samples

the data (D[15:0]). If the FIFO has enough free space to receive more data, the PSSI drives the PSSI_RDY signal active. Otherwise, if the FIFO is full and cannot accept more data, the PSSI drives the PSSI_RDY signal inactive. The transmitting device must repeat the current data in the next cycle when it detects that PSSI_RDY is inactive. This flow control function allows the PSSI to avoid overrun errors when the system (via the DMA) is unable to keep up with the data flow.

Figure 313. Ready in receive mode waveform diagram (CKPOL=0)



If the PSSI_RDY alternate input function is enabled (through DERDYCFG) in transmit mode (OUTEN=1), the PSSI samples the PSSI_RDY signal on the opposite PSSI_PDCK edge to the one at which D[15:0] are driven. If the PSSI_RDY signal is inactive, the PSSI keeps the same data (D[15:0]) and PSSI_DE signals that valid data are available during the next PSSI_PDCK clock cycle. Otherwise, if PSSI_RDY signal is sampled as active, the next data from the TX FIFO (if available) is output on the data outputs (D[15:0]). If no new data are available in the TX FIFO, the PSSI keeps the data output values and outputs the PSSI_DE signal as inactive (if enabled).

The receiving device uses the PSSI_RDY to control the data flow and avoid overrun errors when the system (via the DMA) is unable to keep up with the data flow.

Bidirectional PSSI_DE/PSSI_RDY signal

A single pin can be used for both data enable (PSSI_DE) and ready (PSSI_RDY) functions if DEPOL and RDYPOL are both set to 1 and DERDYCFG is set to 111 or 100 in the PSSI_CR register. In this case, the GPIO corresponding to selected alternate function (PSSI_DE when DERDYCFG=111 or PSSI_RDY when DERDYCFG=100) must be configured as open-drain. The other device must also be configured to drive the line as open-drain, and a weak pull-up must be applied to the line.

The signal thus becomes bidirectional. If either the sender drives the line low (to indicate that the data is not valid) or the receiver drives the line low (to indicate that it is not sampling the current data), then both devices know that the data is not being transferred in the current cycle.

Figure 314. Bidirectional PSSI_DE/PSSI_RDY waveform

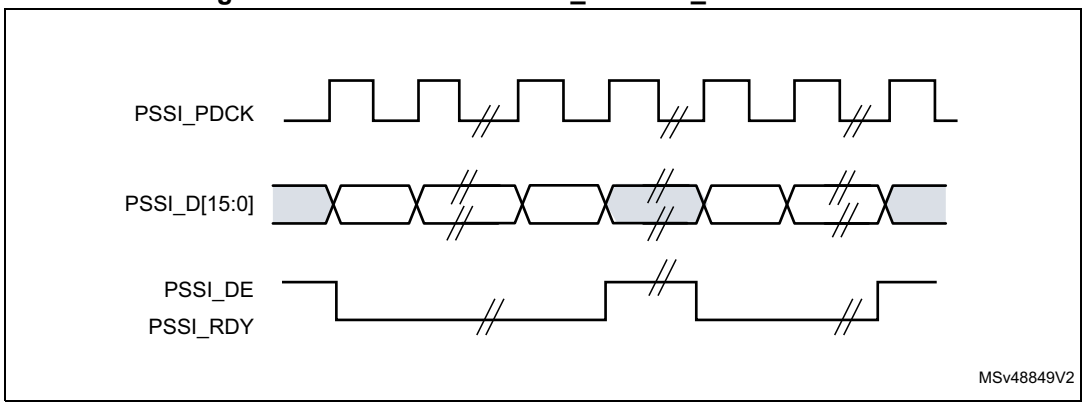
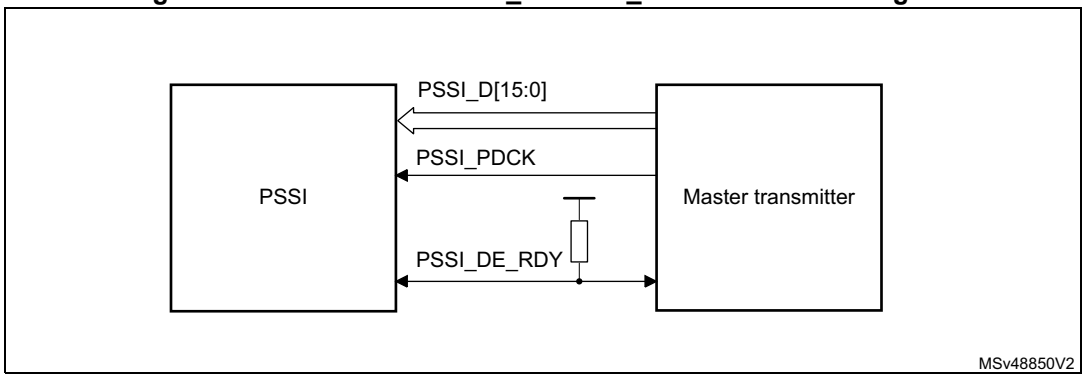


Figure 315. Bidirectional PSSI_DE/PSSI_RDY connection diagram



38.4 PSSI interrupts

The PSSI generates only one interrupt (IT_OVR). It is consequently equivalent to the global interrupt (pssi_it). Refer to *Table 348* for the list of interrupts.

The PSSI and the DCMI share the same interrupt vector. When the PSSI ENABLE bit (bit 14 of PSSI_CR) is set to 1, these interrupts are triggered by the PSSI. Otherwise, they are controlled by the DCMI.

The DCMI ENABLE bit (bit 14 of DCMI_CR) and PSSI ENABLE bit must not be set to 1 at the same time.

Table 348. PSSI interrupt requests

Interrupt acronym	Shared with DCMI	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from low-power mode
IT_OVR	IT_OVR	indicates overrun in receive mode or underrun in transmit mode	OVR_RIS	OVR_IE	OVR_ISC	NA

38.5 PSSI registers

An 8-bit write or a 16-bit write operation to any PSSI register besides PSSI_DR, results in a bus error. 32-bit read and write operations are permitted.

38.5.1 PSSI control register (PSSI_CR)

Address offset: 0x00

Reset value: 0x4000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OUTEN	DMAEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERDYCFG[2:0]			Res.	Res.
rw	rw										rw	rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ENABLE	Res.	Res.	EDM[1:0]		Res.	RDYPOL	Res.	DEPOL	CKPOL	Res.	Res.	Res.	Res.	Res.
	rw			rw	rw		rw		rw	rw					

Bit 31 **OUTEN:** Data direction selection bit

0: Receive mode: data is input synchronously with PSSI_PDCK

1: Transmit mode: data is output synchronously with PSSI_PDCK

Bit 30 **DMAEN:** DMA enable bit

0: DMA transfers are disabled. The user application can directly access the PSSI_DR register when DMA transfers are disabled.

1: DMA transfers are enabled (default configuration). A DMA channel in the general-purpose DMA controller must be configured to perform transfers from/to PSSI_DR.

Bits 29:21 Reserved, must be kept at reset value.

Bits 20:18 **DERDYCFG[2:0]:** Data enable and ready configuration

000: PSSI_DE and PSSI_RDY both disabled

001: Only PSSI_RDY enabled

010: Only PSSI_DE enabled

011: Both PSSI_RDY and PSSI_DE alternate functions enabled

100: Both PSSI_RDY and PSSI_DE features enabled - bidirectional on PSSI_RDY pin (see *Bidirectional PSSI_DE/PSSI_RDY signal on page 1414*)

101: Only PSSI_RDY function enabled, but mapped to PSSI_DE pin

110: Only PSSI_DE function enabled, but mapped to PSSI_RDY pin

111: Both PSSI_RDY and PSSI_DE features enabled - bidirectional on PSSI_DE pin (see *Bidirectional PSSI_DE/PSSI_RDY signal on page 1414*)

When the PSSI_RDY function is mapped to the PSSI_DE pin (settings 101 or 111), it is still the RDYPOL bit which determines its polarity. Similarly, when the PSSI_DE function is mapped to the PSSI_RDY pin (settings 110 or 111), it is still the DEPOL bit which determines its polarity.

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

Bit 14 **ENABLE:** PSSI enable

0: PSSI disabled

1: PSSI enabled

The contents of the FIFO are flushed when ENABLE is cleared to 0.

Note: When ENABLE=1, the content of PSSI_CR must not be changed, except for the ENABLE bit itself. All configuration bits can change as soon as ENABLE changes from 0 to 1.

The DMA controller and all PSSI configuration registers must be programmed correctly before setting the ENABLE bit to 1.

The ENABLE bit and the DCMI ENABLE bit (bit 15 of DCMI_CR) must not be set to 1 at the same time.

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:10 **EDM[1:0]:** Extended data mode

00: Interface captures 8-bit data on every parallel data clock

01: Reserved, must not be selected

10: Reserved, must not be selected

11: The interface captures 16-bit data on every parallel data clock

Bit 9 Reserved, must be kept at reset value.

Bit 8 **RDYPOL:** Ready (PSSI_RDY) polarity

This bit indicates the level on the PSSI_RDY pin when the data are not valid on the parallel interface.

0: PSSI_RDY active low (0 indicates that the receiver is ready to receive)

1: PSSI_RDY active high (1 indicates that the receiver is ready to receive)

Bit 7 Reserved, must be kept at reset value.

Bit 6 **DEPOL:** Data enable (PSSI_DE) polarity

This bit indicates the level on the PSSI_DE pin when the data are not valid on the parallel interface.

0: PSSI_DE active low (0 indicates that data is valid)

1: PSSI_DE active high (1 indicates that data is valid)

Bit 5 **CKPOL:** Parallel data clock polarity

This bit configures the capture edge of the parallel clock or the edge used for driving outputs, depending on OUTEN.

0: Falling edge active for inputs or rising edge active for outputs

1: Rising edge active for inputs or falling edge active for outputs.

Bits 4:0 Reserved, must be kept at reset value.

38.5.2 PSSI status register (PSSI_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.	RTT1B	RTT4B	Res.	Res.
												r	r		

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RTT1B:** FIFO is ready to transfer one byte

1: FIFO is ready for a one byte (32-bit) transfer. In receive mode, this means that at least one valid data byte is in the FIFO. In transmit mode, this means that there is at least one byte free in the FIFO.

0: FIFO is not ready for a 1-byte transfer

Bit 2 **RTT4B:** FIFO is ready to transfer four bytes

1: FIFO is ready for a four-byte (32-bit) transfer. In receive mode, this means that at least four valid data bytes are in the FIFO. In transmit mode, this means that there are at least four bytes free in the FIFO.

0: FIFO is not ready for a four-byte transfer

Bits 1:0 Reserved, must be kept at reset value.

38.5.3 PSSI raw interrupt status register (PSSI_RIS)

Address offset: 0x08

Reset value: 0x0000 0000

PSSI_RIS gives the raw interrupt status. This register is read-only. When read, it returns the status of the corresponding interrupt before masking with the PSSI_IER register value.

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

15	14	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.	OVR_RIS	Res.
														r	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OVR_RIS:** Data buffer overrun/underrun raw interrupt status
0: No overrun/underrun occurred
1: An overrun/underrun occurred: overrun in receive mode, underrun in transmit mode.
This bit is cleared by writing a 1 to the OVR_ISC bit in PSSI_ICR.

Bit 0 Reserved, must be kept at reset value.

38.5.4 PSSI interrupt enable register (PSSI_IER)

Address offset: 0x0C

Reset value: 0x0000 0000

The PSSI_IER register is used to enable interrupts. When one of the PSSI_IER bits is set, the corresponding interrupt is enabled. This register is accessible both in read and write modes.

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OVR_IE:** Data buffer overrun/underrun interrupt enable
0: No interrupt generation
1: An interrupt is generated if either an overrun or an underrun error occurred.

Bit 0 Reserved, must be kept at reset value.

38.5.5 PSSI masked interrupt status register (PSSI_MIS)

This PSSI_MIS register is read-only. When read, it returns the current masked status value of the corresponding interrupt (depending on the value in PSSI_IER). A bit in this register is set if the corresponding enable bit in PSSI_IER is set and the corresponding bit in PSSI_RIS is set.

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.	Res.	OVR_MIS	Res.
														r	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OVR_MIS:** Data buffer overrun/underrun masked interrupt status
This bit is set to 1 only when PSSI_IER/OVR_IE and PSSI_RIS/OVR_RIS are both set to 1.
0: No interrupt is generated when an overrun/underrun error occurs
1: An interrupt is generated if there is either an overrun or an underrun error and the OVR_IE bit is set in PSSI_IER.

Bit 0 Reserved, must be kept at reset value.

38.5.6 PSSI interrupt clear register (PSSI_ICR)

Address offset: 0x14

Reset value: 0x0000 0000

The PSSI_ICR register is write-only. Writing a 1 into a bit of this register clears the corresponding bit in the PSSI_RIS and PSSI_MIS registers. Writing a 0 has no effect. Reading this register always gives zeros.

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

15	14	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.	OVR_ISC	Res.
														w	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OVR_ISC:** Data buffer overrun/underrun interrupt status clear
Writing this bit to 1 clears the OVR_RIS bit in PSSI_RIS.

Bit 0 Reserved, must be kept at reset value.

38.5.7 PSSI data register (PSSI_DR)

Address offset: 0x28

Reset value: 0x0000 0000

In receive mode (OUTEN=0), the DMA controller must read the received data from this register. Write operations to PSSI_DR result in an error response. When more bytes than the number of valid bytes are read in the FIFO, the invalid bytes return zeros.

In transmit mode (OUTEN=1), the DMA controller must write the data to be transmitted into this register. Read operations to PSSI_DR result in an error response.

32-bit, 16-bit, and 8-bit accesses are all supported for PSSI_DR. For instance, 16-bit read/write operations remove/add two bytes from/to the FIFO. However, 8-bit accesses are permitted only when the PSSI is configured to transfer 8 data bits at a time (EDM=00 in PSSI_CR). 8-bit accesses to PSSI_DR when EDM is not set to 0 result in an error response.

All accesses must include byte 0: 8-bit accesses must be performed to bits 7 to 0 and 16-bit accesses from bits 15 to 0. Accesses that do not include byte 0 results in an error response.

Accessing PSSI_DR when ENABLE bit in PSSI_CR is set to 0 results in an error response.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BYTE3[7:0]								BYTE2[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
BYTE1[7:0]								BYTE0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **BYTE3[7:0]**: Data byte 3

Bits 23:16 **BYTE2[7:0]**: Data byte 2

Bits 15:8 **BYTE1[7:0]**: Data byte 1

Bits 7:0 **BYTE0[7:0]**: Data byte 0

38.5.8 PSSI register map

Table 349. PSSI 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	PSSI_CR	OUTEN	DMAEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERDYCFG			Res.	Res.	Res.	ENABLE[2:0]			EDM		Res.	RDYPOL	Res.	DEPOL	CKPOL	Res.	Res.	Res.	Res.	Res.
	Reset value	0	1										0	0	0				0			0	0		0		0	0					
0x04	PSSI_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.	RTT1B	RTT4B	Res.	Res.
	Reset value																													0	0		

Table 349. PSSI 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
0x08	PSSI_RIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_RIS	Res.
	Reset value																															0	
0x0C	PSSI_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.am	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_IE	Res.
	Reset value																															0	
0x10	PSSI_MIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_MIS	Res.
	Reset value																															0	
0x14	PSSI_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.	OVR_ISC	Res.
	Reset value																															0	
0x18-0x24	Reserved	Res.																															
0x28	PSSI_DR	BYTE3[7:0]								BYTE2[7:0]								BYTE1[7:0]								BYTE0[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

Refer to *Section 2.3 on page 115* for the register boundary addresses.

39 Touch sensing controller (TSC)

39.1 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.

39.2 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

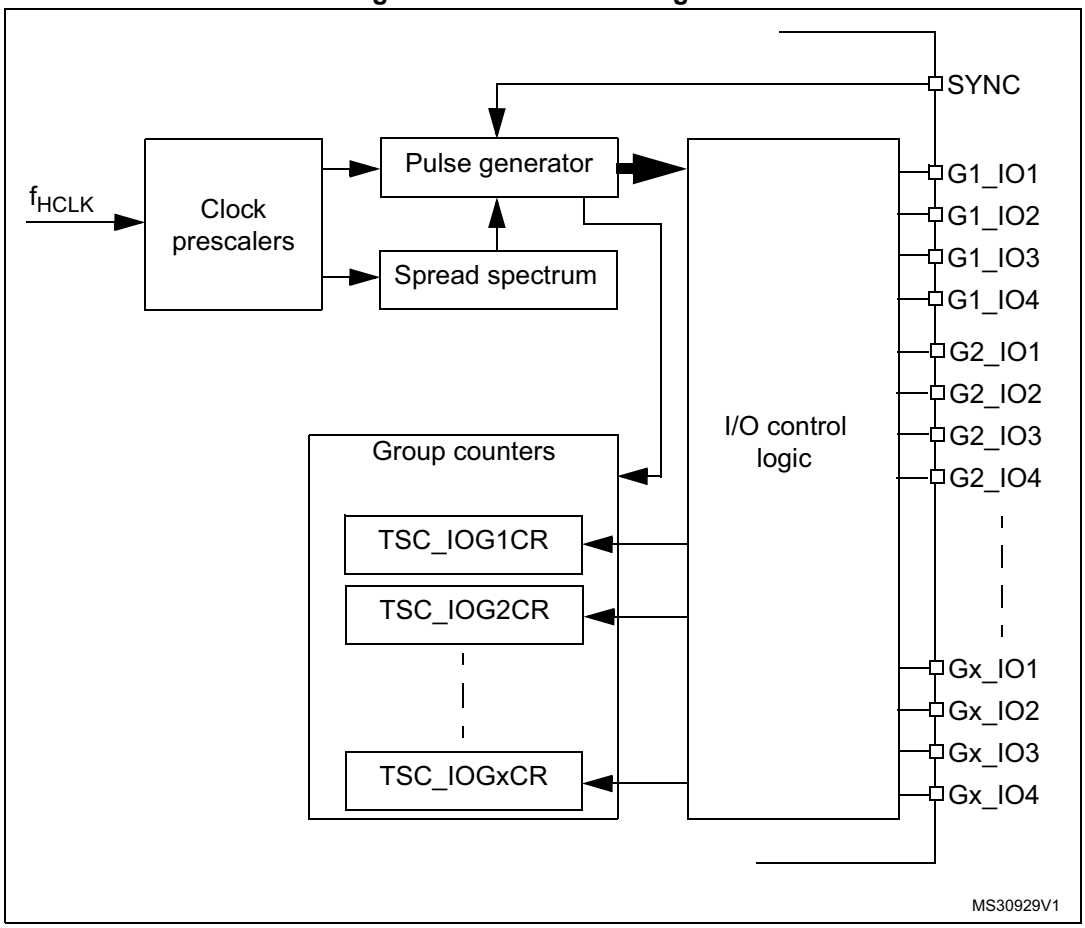
Note: The number of capacitive sensing channels is dependent on the size of the packages and subject to IO availability.

39.3 TSC functional description

39.3.1 TSC block diagram

The block diagram of the touch sensing controller is shown in *Figure 316*.

Figure 316. TSC block diagram



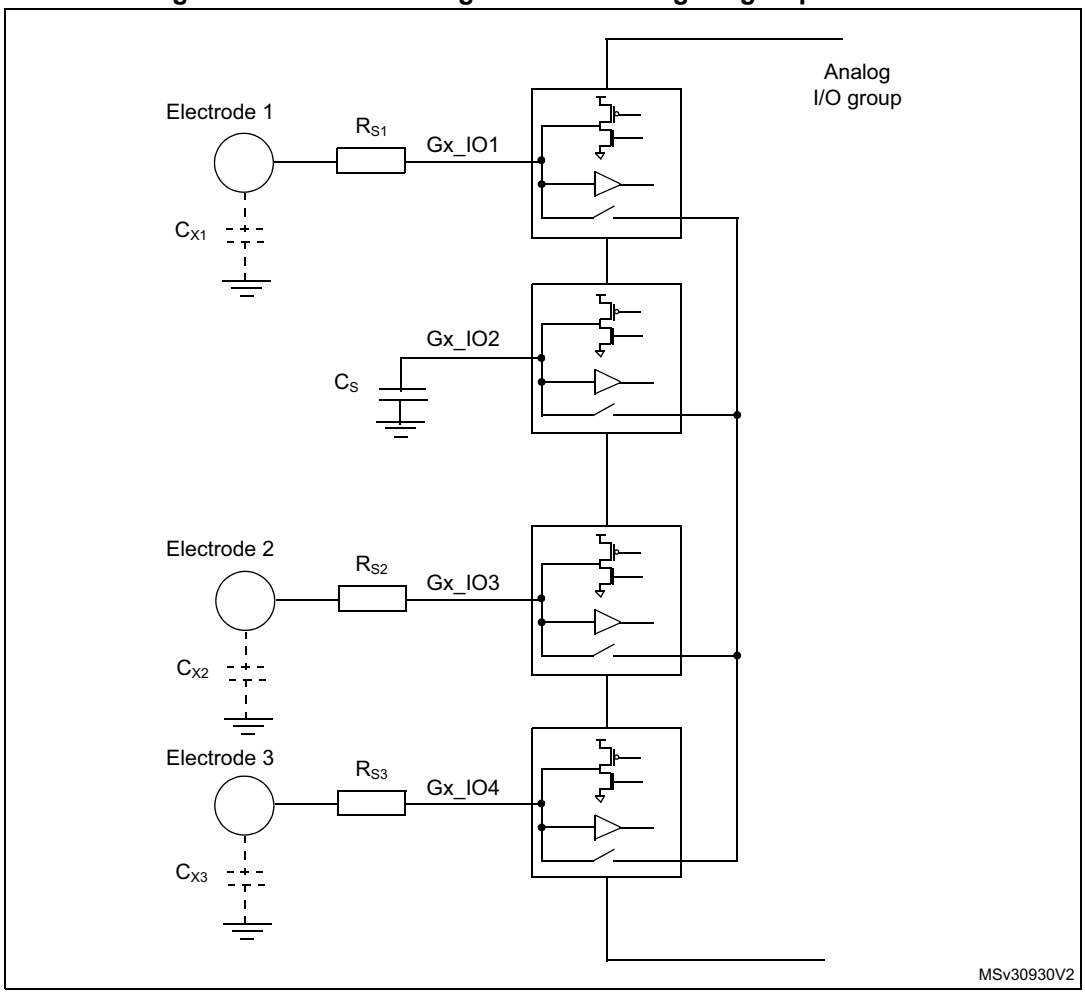
39.3.2 Surface charge transfer acquisition overview

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 317*). 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 317. 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 350 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.

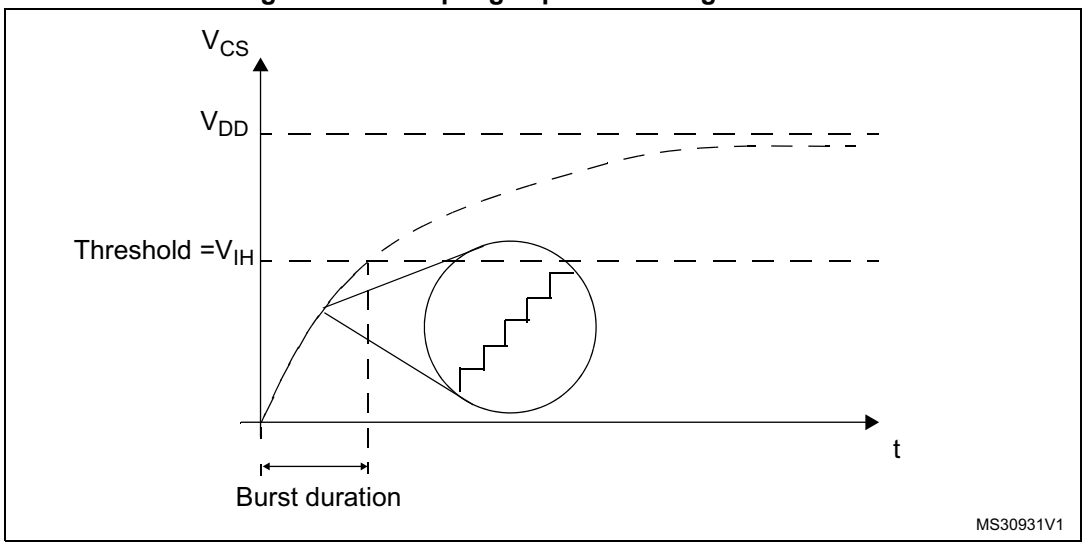
Table 350. Acquisition sequence summary

State	Gx_IO1 (channel)	Gx_IO2 (sampling)	Gx_IO3 (channel)	Gx_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

Note: Gx_IOy where x is the analog I/O group number and y the GPIO number within the selected group.

The voltage variation over the time on the sampling capacitor C_S is detailed below:

Figure 318. Sampling capacitor voltage variation



39.3.3 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:

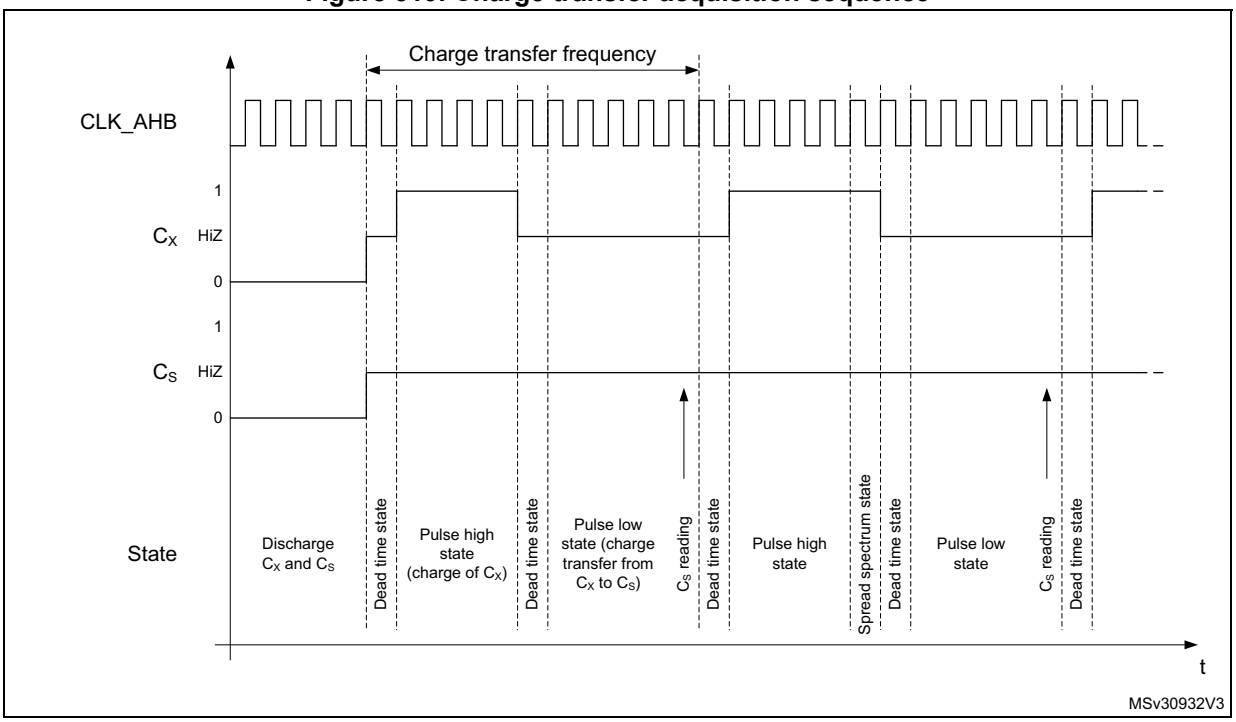
- The pulse generator clock (PGCLK) is defined using the PGPSC[2:0] bits of the TSC_CR register
- The spread spectrum clock (SSCLK) is defined using the SSPSC bit of the TSC_CR register

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 11: Reset and clock control (RCC)*.

39.3.4 Charge transfer acquisition sequence

An example of a charge transfer acquisition sequence is detailed in *Figure 319*.

Figure 319. 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 2 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:

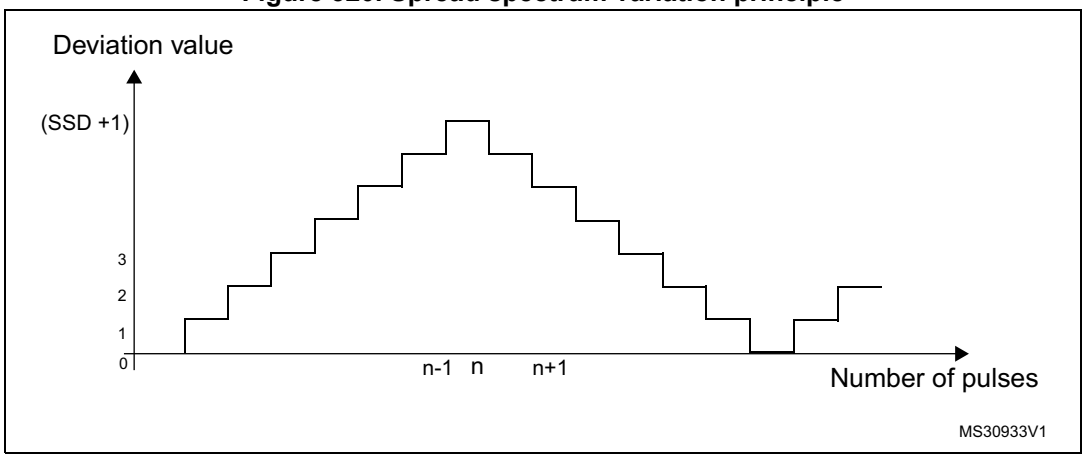
- *bits PGPSC are set to '000' and bits CTPL are set to '0000'*
- *bits PGPSC are set to '000' and bits CTPL are set to '0001'*
- *bits PGPSC are set to '001' and bits CTPL are set to '0000'*

39.3.5 Spread spectrum feature

The spread spectrum feature generates 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 320. Spread spectrum variation principle



The table below details the maximum frequency deviation with different HCLK settings:

Table 351. 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

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.

39.3.6 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.

39.3.7 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 352. 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.

39.3.8 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.

39.3.9 I/O hysteresis and analog switch control

In order to offer a higher flexibility, the touch sensing controller is able 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.

39.4 TSC low-power modes

Table 353. Effect of low-power modes on TSC

Mode	Description
Sleep	No effect. Peripheral interrupts cause the device to exit Sleep mode.
Stop	Peripheral registers content is kept.
Standby	Powered-down. The peripheral must be reinitialized after exiting Standby mode.

39.5 TSC interrupts

Table 354. 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

39.6 TSC registers

Refer to *Section 1.2 on page 104* of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

39.6.1 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 39.3.4: 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 39.3.4: 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.

39.6.2 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

39.6.3 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

39.6.4 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

39.6.5 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, x = 8 to 1, y = 4 to 1.

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).

39.6.6 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).

39.6.7 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.

39.6.8 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.

39.6.9 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

39.6.10 TSC I/O group x counter register (TSC_IOGxCR)

x represents the analog I/O group number.

Address offset: 0x30 + 0x04 * x, (x = 1 to 8)

Reset value: 0x0000 0000

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

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

39.6.11 TSC register map

Table 355. 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 355. 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 115* for the register boundary addresses.

40 True random number generator (RNG)

40.1 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).

40.2 RNG main features

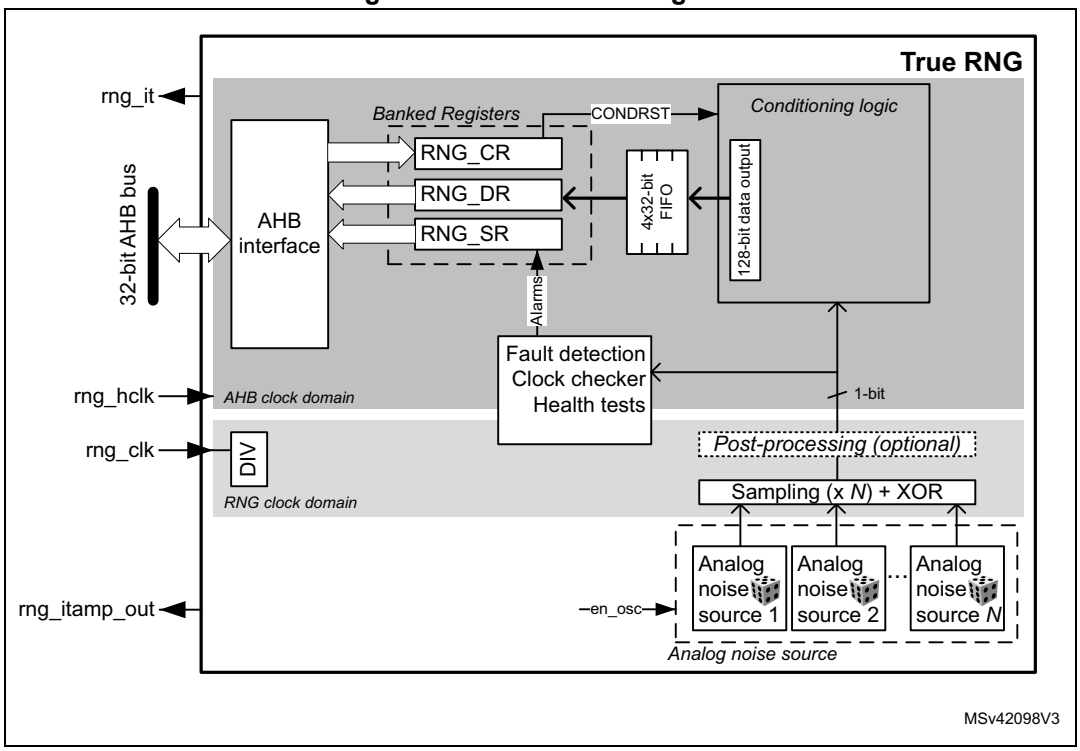
- The RNG delivers 32-bit true random numbers, produced by an analog entropy source conditioned by a NIST SP800-90B approved conditioning stage.
- It can be used as entropy source to construct a non-deterministic random bit generator (NDRBG).
- In the NIST configuration, it produces four 32-bit random samples every 412 AHB clock cycles if $f_{AHB} < f_{threshold}$ (256 RNG clock cycles otherwise).
- It embeds start-up and NIST SP800-90B approved continuous health tests (repetition count and adaptive proportion tests), associated with specific error management
- It can be disabled to reduce power consumption, or enabled with an automatic low power mode (default configuration).
- It has an AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (else an AHB bus error is generated, and the write accesses are ignored).

40.3 RNG functional description

40.3.1 RNG block diagram

Figure 321 shows the RNG block diagram.

Figure 321. RNG block diagram



40.3.2 RNG internal signals

Table 356 describes a list of useful-to-know internal signals available at the RNG level, not at the STM32 product level (on pads).

Table 356. RNG internal input/output signals

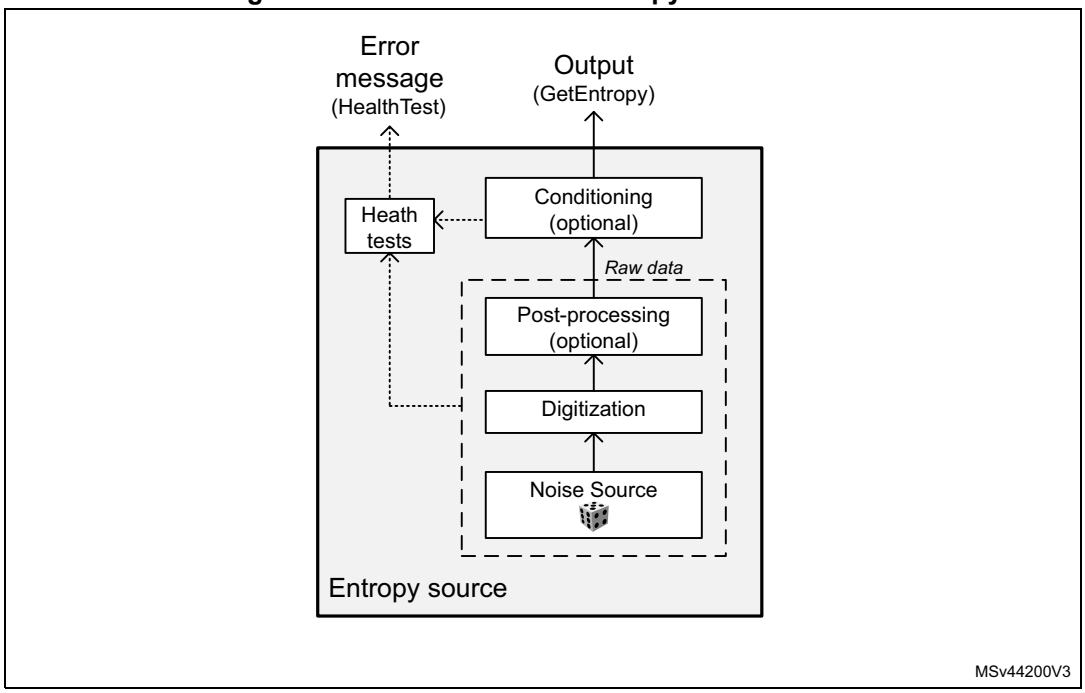
Signal name	Signal type	Description
rng_it	Digital output	RNG global interrupt request
rng_hclk	Digital input	AHB clock
rng_clk	Digital input	RNG dedicated clock, asynchronous to rng_hclk
rng_itamp_out	digital output	RNG internal tamper event signal to TAMP (XORed), triggered when an unexpected hardware fault occurs. When this signal is triggered, RNG stops delivering random samples, requiring a reset and a new initialization to be usable again.

40.3.3 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 322*. 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 322. 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 40.3.8: RNG low-power usage*.
- The XORing of all the noise sources into a single analog output.
- A sampling stage of this output clocked by a dedicated clock input (rng_clk with integrated divider), delivering a 1-bit raw data output.

This noise source sampling is independent to the AHB interface clock frequency (rng_hclk), with a possibility for the software to decrease the sampling frequency by using the integrated divider.

Note: In Section 40.6: 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 40.6.2*) a normalization debiasing is applied, that is half of the bits are taken from the sampled noise source, half of the bits are taken from inverted sampled noise source.

Conditioning

The conditioning component in the RNG is a deterministic function that increases the entropy rate of the resulting fixed-length bitstrings output (128-bit). The NIST SP800-90B target is full entropy on the output (128-bit).

The times required between two random number generations, and between the RNG initialization and availability of first sample are described in *Section 40.5: 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 40.5: RNG processing time*.

Whenever a random number is available through the RNG_DR register the DRDY flag transitions from 0 to 1. This flag remains high until output buffer becomes empty after reading four words from the RNG_DR register.

Note: When interrupts are enabled an interrupt is generated when this data ready flag transitions from 0 to 1. Interrupt is then cleared automatically by the RNG as explained above.

Health checks

This component ensures that the entire entropy source (with its noise source) starts then operates as expected, obtaining assurance that failures are caught quickly and with a high probability and reliability.

The RNG implements the following health check features in accordance with NIST SP800-90B. The described thresholds correspond to the value recommended for register RNG_HTCR (configuration A in *Section 40.6.2*).

1. Start-up health tests, performed after reset and before the first use of the RNG as entropy source
   - Repetition count test, flagging an error when the noise source has provided more than 38 consecutive bits at a constant value (0 or 1).
   - Adaptive proportion test running on a window of 1024 consecutive bits: the RNG verifies that the first bit on the outputs of the noise source is not repeated more than 686 times.
   - Known-answer tests, to verify the conditioning stage.
2. Continuous health tests, running indefinitely on the outputs of the noise source
   - Repetition count test, similar to the one running in start-up tests.
   - Adaptive proportion test, similar to the one running in start-up tests.
3. Vendor specific continuous tests
   - Transition count test, flagging an error when the noise source has delivered more than 32 consecutive occurrences of 2-bit patterns (01 or 10).
   - Real-time “too slow” sampling clock detector, flagging an error when one RNG clock cycle (before divider) is smaller than AHB clock cycle divided by 32.
4. On-demand test of digitized noise source (raw data)
   - Supported by restarting the entropy source and re-running the startup tests (see software reset sequence in *Section 40.3.4: 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 40.3.7: 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 40.6: RNG entropy source validation for details.

40.3.4 RNG initialization

The RNG simplified state machine is pictured on *Figure 323*.

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 40.5: RNG processing time*.

Figure 323. RNG initialization overview

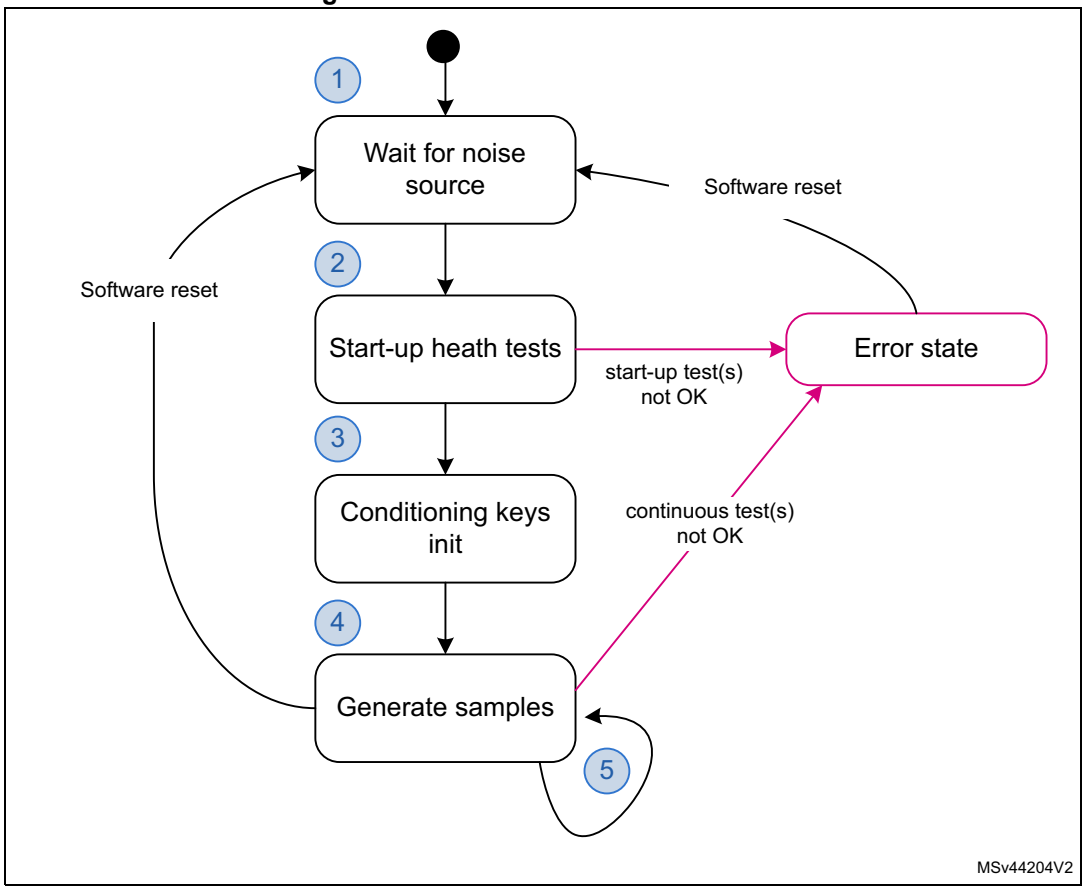


Figure 323 also highlights a possible software reset sequence, implemented by:

1. Writing bits RNGEN = 0 and CONDRST = 1 in the RNG_CR register with the same RNG configuration and a new CLKDIV if needed.
2. Then writing RNGEN = 1 and CONDRST = 0 in the RNG_CR register.
3. Wait for random number to be ready, after initialization completes

Note: When RNG peripheral is reset through RCC (hardware reset), the RNG configuration for optimal randomness is lost in RNG registers. Software reset with CONFIGLOCK set preserves the RNG configuration.

40.3.5 RNG operation

Normal operations

To run the RNG using interrupts, the following steps are recommended:

1. Consult *Section 40.6: RNG entropy source validation* and verify if a specific RNG configuration is required for the application.
   - If it is the case, write in the RNG_CR register the bit CONDRST = 1 together with the correct RNG configuration. Then perform a second write to the RNG_CR register with the bit CONDRST = 0, the interrupt enable bit IE = 1 and the RNG enable bit RNGEN = 1.
   - If it is not the case perform a write to the RNG_CR register with the interrupt enable bit IE = 1 and the RNG enable bit RNGEN = 1.
2. An interrupt is now generated when a random number is ready or when an error occurs. Therefore at each interrupt, check that:
   - No error occurred. The SEIS and CEIS bits must be set to 0 in the RNG_SR register.
   - A random number is ready. The DRDY bit must be set to 1 in the RNG_SR register.
   - If above two conditions are true the content of the RNG_DR register can be read up to four consecutive times. If valid data is available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of above conditions are false, the RNG_DR register must not be read. If an error occurred error recovery sequence described in *Section 40.3.7* must be used.

To run the RNG in polling mode following steps are recommended:

1. Consult *Section 40.6: RNG entropy source validation* and verify if a specific RNG configuration is required for the application.
   - If it is the case write in the RNG_CR register the bit CONDRST = 1 together with the correction RNG configuration. Then perform a second write to the RNG_CR register with the bit CONDRST = 0 and the RNG enable bit RNGEN = 1.
   - If it is not the case only enable the RNG by setting the RNGEN bit to 1 in the RNG_CR register.
2. Read the RNG_SR register and check that:
   - No error occurred (the SEIS and CEIS bits must be set to 0)
   - A random number is ready (the DRDY bit must be set to 1)
3. If above conditions are true read the content of the RNG_DR register up to four consecutive times. If valid data is available in the conditioning output buffer four

additional words can be read by the application (in this case the DRDY bit is still high). If one or both of above conditions are false, the RNG_DR register must not be read. If an error occurred error recovery sequence described in *Section 40.3.7* 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 40.6: RNG entropy source validation*. The gain in random number generation speed is summarized in *Section 40.5: 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 40.3.8: 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 40.3.3: Random number generation*.

40.3.6 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 40.6: RNG entropy source validation*.

*Note: When the CED bit in the RNG_CR register is set to 0, the RNG clock frequency before internal divider **must be higher** than AHB clock frequency divided by 32, otherwise the clock checker always flags a clock error (CECS = 1 in the RNG_SR register).*

See *Section 40.3.1: RNG block diagram* for details (AHB and RNG clock domains).

40.3.7 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 40.3.6: RNG clocking*) and then it must clear the CEIS bit interrupt flag. The CECS bit is automatically cleared when clocking condition is normal.

Note: The clock error has no impact on generated random numbers, that is the application can still read RNG_DR register.

CEIS is set only when CECS is set to 1 by RNG.

Noise source error detection

When a noise source (or seed) error occurs, the RNG stops generating random numbers and sets to 1 both SEIS and SECS bits to indicate that a seed error occurred. If a value is available in the RNG_DR register, it must not be used as it may not have enough entropy.

The following sequence must be used to fully recover from a seed error:

1. Software reset by writing CONDRST at 1 and at 0 (see bitfield description for details). This step is needed only if SECS is set. Indeed, when SEIS is set and SECS is cleared it means RNG performed the reset automatically (auto-reset).
2. If SECS was set in step 1 (no auto-reset) wait for CONDRST to be cleared in the RNG_CR register, then confirm that SEIS is cleared in the RNG_SR register. Otherwise just clear SEIS bit in the RNG_SR register.
3. If SECS was set in step 1 (no auto-reset) wait for SECS to be cleared by RNG. The random number generation is now back to normal.

Note: After a seed error RNG restarts generating random numbers when SECS is cleared.

When application sets ARDIS bit in RNG_CR register auto-reset is disabled. CONDRST must be used in step 1.

RNG tamper errors

When an unexpected error is found by the RNG an internal tamper event is triggered in TAMP peripheral, and the RNG stops delivering random data.

When this event occurs, secure application needs to reset the RNG peripheral either using the central reset management or the global SoC reset. Then a proper initialization of the RNG is required, again.

40.3.8 RNG low-power usage

If power consumption is a concern, the RNG can be disabled as soon as the DRDY bit is set to 1 by setting the RNGEN bit to 0 in the RNG_CR register. As the post-processing logic and the output buffer remain operational while RNGEN = 0 following features are available to software:

- If there are valid words in the output buffer four random numbers can still be read from the RNG_DR register.
- If there are valid bits in the conditioning output internal register four additional random numbers can be still be read from the RNG_DR register. If it is not the case RNG must be re-enabled by the application until the expected new noise source bits threshold is reached (128-bit in NIST mode) and a complete conditioning round is done.
  Four new random words are then available only if the expected number of conditioning round is reached (two if NISTC = 0). The overall time can be found in *Section 40.5: RNG processing time on page 1451*.

When disabling the RNG the user deactivates all the analog seed generators, whose power consumption is given in the datasheet electrical characteristics section. The user also gates all the logic clocked by the RNG clock. Note that this strategy is adding latency before a random sample is available on the RNG_DR register, because of the RNG initialization time.

If the RNG block is disabled during initialization (that is well before the DRDY bit rises for the first time), the initialization sequence resumes from where it was stopped when RNGEN bit is set to 1, unless the application resets the conditioning logic using CONDRST bit in the RNG_CR register.

40.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see *Section 40.3.7: Error management*
- Clock error, see *Section 40.3.7: Error management*

Dedicated interrupt enable control bits are available as shown in *Table 357*.

Table 357. 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.

40.5 RNG processing time

In recommended configuration A described in *Table 358*, the time between two sets of four 32-bit data is either:

- 206 x N AHB cycles if $f_{AHB} < f_{threshold}$ (conditioning stage is limiting), or
- 128 x N RNG cycles $f_{AHB} \geq f_{threshold}$ (noise source stage is limiting).

With $f_{threshold} = 1.6 \times f_{RNG}$, for instance 77 MHz if f_{RNG}= 48 MHz.Value N is 2.

Note: When CLKDIV is different from zero, f_{RNG} must take into account the internal divider ratio.

If configuration B is selected the performance figures become:

- 206 AHB cycles if $f_{AHB} < f_{threshold}$ or
- 32 RNG cycles $f_{AHB} \geq f_{threshold}$

with $f_{threshold} = 6.5 \times f_{RNG}$.

40.6 RNG entropy source validation

40.6.1 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.

40.6.2 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 358: RNG configurations*. Note that only configuration A can be certified NIST SP800-90B.

Table 358. RNG configurations

RNG Config.	RNG_CR bits						Loop number (N)	RNG_ HTCR register
	NISTC bit	RNG_ CONFIG1 [5:0]	CLKDIV [3:0]	RNG_ CONFIG2 [2:0]	RNG_ CONFIG3 [3:0]	CED bit		
A	0	0x0F	0x6(1)	0x0	0xD	0	2	0x0000 9AAE(2)
B	1	0x18	0x0	0x0	0x0	0	1	0x0000 A6BA

1. For NIST certification the noise source sampling must be 750 kHz or less. Hence, if the RNG clock is different from 48 MHz, this value of CLKDIV must be adapted. See CLKDIV bitfield description in *Section 40.7.1* for details.
2. Corresponds to 38 for repetition tests and 686 for adaptive tests. See *Health checks on page 1446* for details

40.6.3 Data collection

In order to run statistical tests it is required to collect samples from the entropy source at raw data level as well as at the output of the entropy source. For details on data collection and the running of statistical test suites refer to “STM32 microcontrollers random number generation validation using NIST statistical test suite” application note (AN4230) available from *www.st.com*.

Contact STMicroelectronics if above samples need to be retrieved for the product.

40.7 RNG registers

The RNG is associated with a control register, a data register and a status register.

40.7.1 RNG control register (RNG_CR)

Address offset: 0x000

Reset value: 0x0080 0D00

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]				ARDIS	Res.	CED	Res.	IE	RNGEN	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw		

Bit 31 **CONFIGLOCK**: RNG Config lock

0: Writes to the RNG_HTCR and RNG_CR configuration bits [29:4] are allowed.

1: Writes to the RNG_HTCR and RNG_CR configuration bits [29:4] are ignored until the next RNG reset.

This bitfield is set once: if this bit is set it can only be reset to 0 if RNG is reset.

Bit 30 **CONDRST**: Conditioning soft reset

Write 1 and then write 0 to reset the conditioning logic, clear all the FIFOs and start a new RNG initialization process, with RNG_SR cleared. Registers RNG_CR and RNG_HTCR are not changed by CONDRST.

This bit must be set to 1 in the same access that set any configuration bits [29:4]. In other words, when CONDRST bit is set to 1 correct configuration in bits [29:4] must also be written.

When CONDRST is set to 0 by software its value goes to 0 when the reset process is done. It takes about 2 AHB clock cycles + 2 RNG clock cycles.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:20 **RNG_CONFIG1[5:0]:** RNG configuration 1

Reserved to the RNG configuration (bitfield 1). Must be initialized using the recommended value documented in *Section 40.6: RNG entropy source validation*.

Writing any bit of RNG_CONFIG1 is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bits 19:16 **CLKDIV[3:0]**: Clock divider factor

This value used to configure an internal programmable divider (from 1 to 16) acting on the incoming RNG clock. These bits can be written only when the core is disabled (RNGEN = 0).

0x0: internal RNG clock after divider is similar to incoming RNG clock.

0x1: two RNG clock cycles per internal RNG clock.

0x2: 2^2 (= 4) RNG clock cycles per internal RNG clock.

...

0xF: 2^{15} RNG clock cycles per internal clock (for example. an incoming 48 MHz RNG clock becomes a 1.5 kHz internal RNG clock)

Writing these bits is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bits 15:13 **RNG_CONFIG2[2:0]:** RNG configuration 2
Reserved to the RNG configuration (bitfield 2). Refer to RNG_CONFIG1 bitfield for details.

Bit 12 **NISTC**: NIST custom
0: Hardware default values for NIST compliant RNG. In this configuration per 128-bit output two conditioning loops are performed and 256 bits of noise source are used.
1: Custom values for NIST compliant RNG. See *Section 40.6: 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 **ARDIS**: Auto reset disable
0: When a noise source error occurs RNG performs an automatic reset to clear SECS bit.
1: When a noise source error occurs application must reset RNG by writing CONDRST to 1 then to 0, in order to restart random number generation.
When auto-reset is enabled application still need to clear SEIS bit after a noise source error.
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 6 Reserved, must be kept at reset value.

Bit 5 **CED:** Clock error detection
0: Clock error detection is enable
1: Clock error detection is disable
The clock error detection cannot be enabled nor disabled on-the-fly when the RNG is enabled, that is to enable or disable CED the RNG must be disabled.
Writing this bit is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **IE:** Interrupt Enable
0: RNG Interrupt is disabled
1: RNG Interrupt is enabled. An interrupt is pending as soon as DRDY = 1, SEIS = 1 or CEIS = 1 in the RNG_SR register.

Bit 2 **RNGEN:** True random number generator enable
0: True random number generator is disabled. Analog noise sources are powered off and logic clocked by the RNG clock is gated.
1: True random number generator is enabled.

Bits 1:0 Reserved, must be kept at reset value.

40.7.2 RNG status register (RNG_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	Res.	Res.	SECS	CECS	DRDY
									rc_w0	rc_w0			r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SEIS:** Seed error interrupt status

This bit is set at the same time as SECS. It is cleared by writing 0 (unless CONDRST is used). Writing 1 has no effect.

0: No faulty sequence detected

1: At least one faulty sequence is detected. See SECS bit description for details.

An interrupt is pending if IE = 1 in the RNG_CR register.

Bit 5 **CEIS:** Clock error interrupt status

This bit is set at the same time as CECS. It is cleared by writing 0. Writing 1 has no effect.

0: The RNG clock is correct (fRNGCLK> fHCLK/32)

1: The RNG clock before internal divider is detected too slow (fRNGCLK< fHCLK/32)

An interrupt is pending if IE = 1 in the RNG_CR register.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **SECS:** Seed error current status

0: No faulty sequence has currently been detected. If the SEIS bit is set, this means that a faulty sequence was detected and the situation has been recovered.

1: At least one of the following faulty sequence has been detected:

- Run-time repetition count test failed (noise source has provided more than 24 consecutive bits at a constant value 0 or 1, or more than 32 consecutive occurrence of two bits patterns 01 or 10)
- Start-up or continuous adaptive proportion test on noise source failed.
- Start-up post-processing/conditioning sanity check failed.

Bit 1 **CECS:** Clock error current status

0: The RNG clock is correct (fRNGCLK> fHCLK/32). If the CEIS bit is set, this means that a slow clock was detected and the situation has been recovered.

1: The RNG clock is too slow (fRNGCLK< fHCLK/32).

Note: CECS bit is valid only if the CED bit in the RNG_CR register is set to 0.

Bit 0 **DRDY:** Data Ready

0: The RNG_DR register is not yet valid, no random data is available.

1: The RNG_DR register contains valid random data.

Once the output buffer becomes empty (after reading the RNG_DR register), this bit returns to 0 until a new random value is generated.

Note: The DRDY bit can rise when the peripheral is disabled (RNGEN = 0 in the RNG_CR register).

If IE=1 in the RNG_CR register, an interrupt is generated when DRDY = 1.

40.7.3 RNG data register (RNG_DR)

Address offset: 0x008

Reset value: 0x0000 0000

The RNG_DR register is a read-only register that delivers a 32-bit random value when read. The content of this register is valid when DRDY = 1 and value is not 0x0, even if RNGEN = 0.

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

Bits 31:0 **RNDATA[31:0]:** Random data

32-bit random data which are valid when DRDY = 1. When DRDY = 0 RNDATA value is zero.

When DRDY is set, it is recommended to always verify that RNG_DR is different from zero. Because when it is the case a seed error occurred between RNG_SR polling and RND_DR output reading (rare event).

40.7.4 RNG health test control register (RNG_HTCR)

Address offset: 0x010

Reset value: 0x0000 72AC

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 40.6: 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.

40.7.5 RNG register map

Table 359. 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]				ARDIS	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	1	1	0	1	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	0	0	0	0	0	1	1	1	0	0	1	0	1	0	1	0	1	1	0	0

Refer to *Section 2.3* for the register boundary addresses.

41 AES hardware accelerator (AES)

41.1 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).

41.2 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
- Hardware key sharing with side-channel resistant SAES peripheral (Shared-key mode), controlled by SAES

41.3 AES implementation

The devices have one AES and one SAES peripheral, implemented as shown in the following table.

Table 360. AES/SAES features

AES/SAES modes/features(1)	AES	SAES
ECB, CBC chaining	X	X
CTR, CCM, GCM chaining	X	-
AES 128-bit ECB encryption in cycles	51	528
DHUK and BHK key selection	-	X
Side-channel attacks resistance	-	X
Shared key between SAES and AES	X	

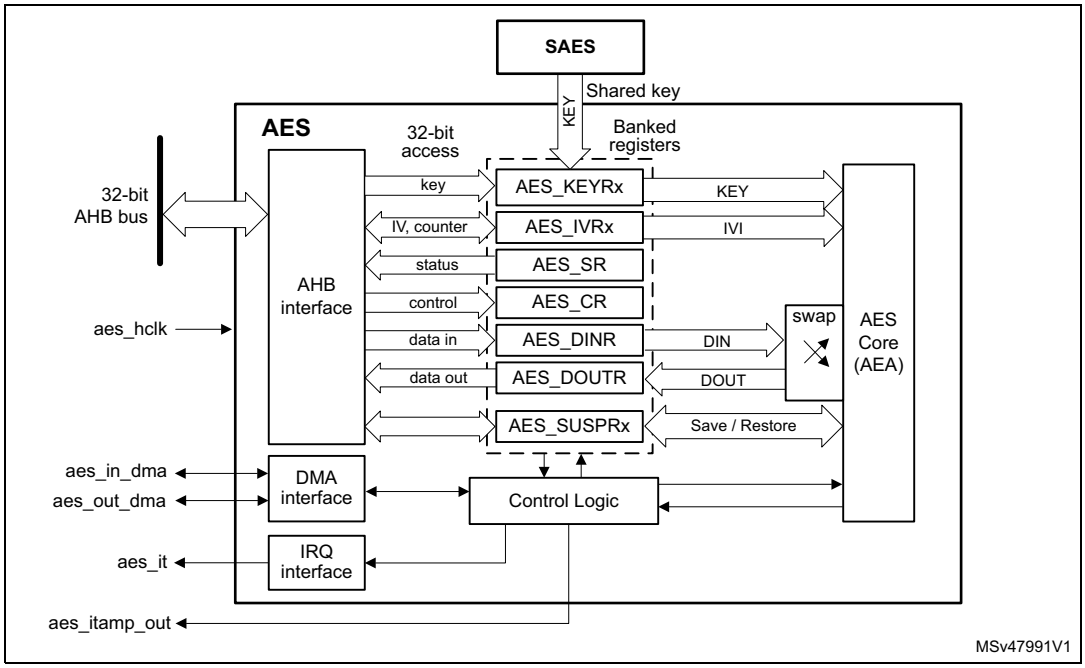
1. X = supported.

41.4 AES functional description

41.4.1 AES block diagram

Figure 324 shows the block diagram of AES.

Figure 324. AES block diagram



41.4.2 AES internal signals

Table 361 describes the user relevant internal signals interfacing the AES peripheral.

Table 361. 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
aes_itamp_out	Output	Tamper event signal to TAMP (XOR-ed), triggered when an unexpected hardware fault occurs. When this signal is triggered, AES automatically clears key registers. A reset is required for AES to be usable again.

41.4.3 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.

Special key operation is selected using the KMOD[1:0] bitfield of the AES_CR register. See *Section 41.4.13* for details.

Typical data processing

Typical usage of the AES is described in *Section 41.4.4: AES procedure to perform a cipher operation on page 1465*.

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)

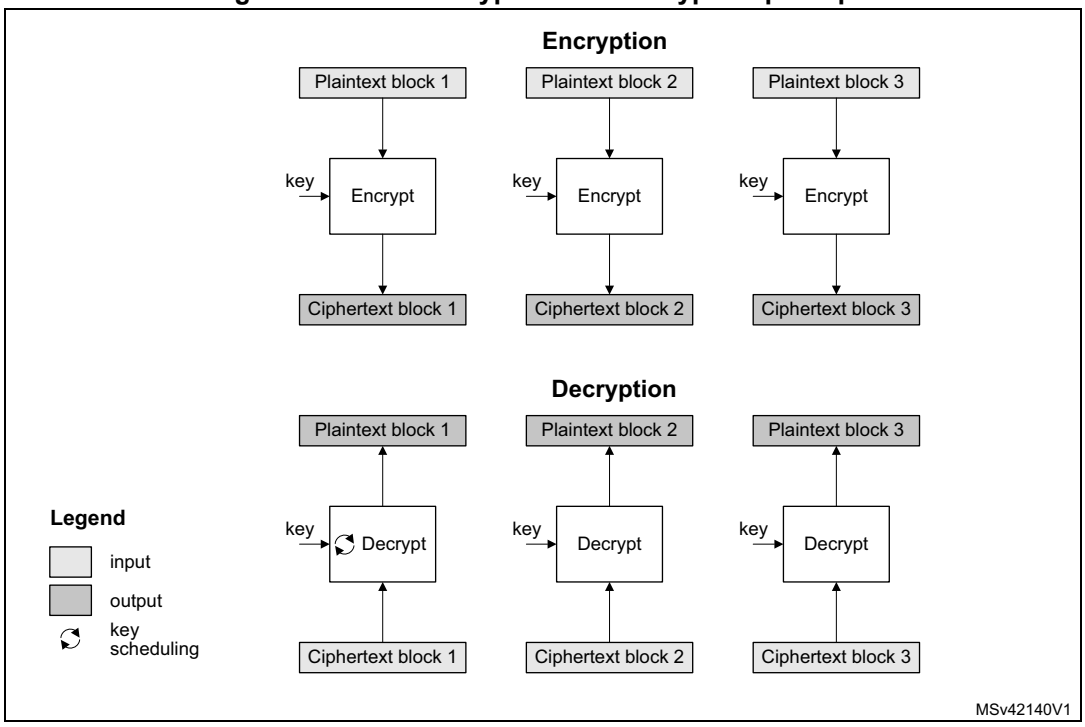
Note: The chaining mode may be changed only when AES is disabled (bit EN of the AES_CR register cleared).

Principle of each AES chaining mode is provided in the following subsections.

Detailed information is in dedicated sections, starting from *Section 41.4.8: AES basic chaining modes (ECB, CBC)*.

Electronic codebook (ECB) mode

Figure 325. ECB encryption and decryption principle

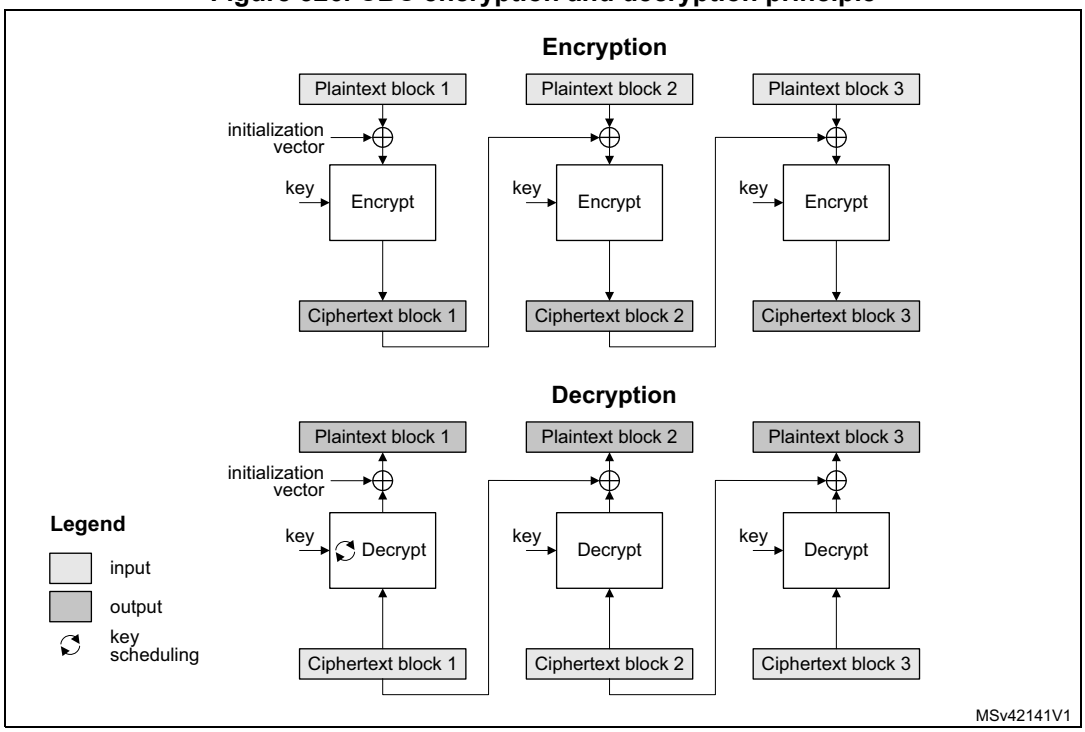


ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately.

Note: For decryption, a special key scheduling is required before processing the first block.

Cipher block chaining (CBC) mode

Figure 326. CBC encryption and decryption principle

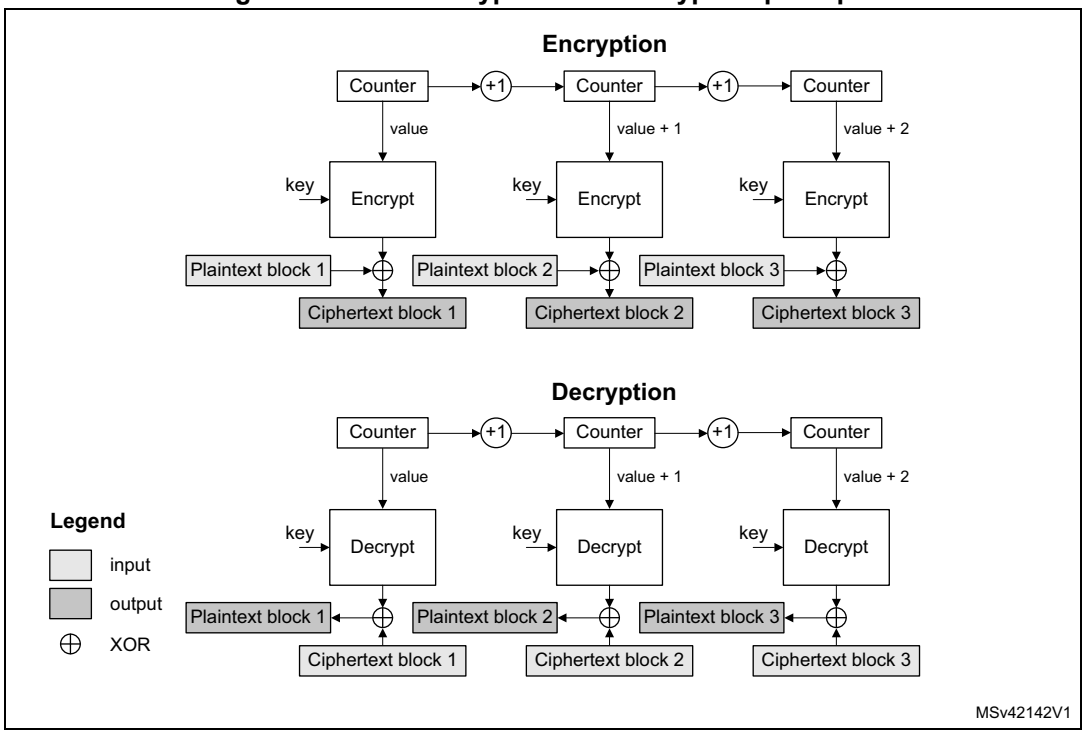


In CBC mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing.

Note: For decryption, a special key scheduling is required before processing the first block.

Counter (CTR) mode

Figure 327. CTR encryption and decryption principle

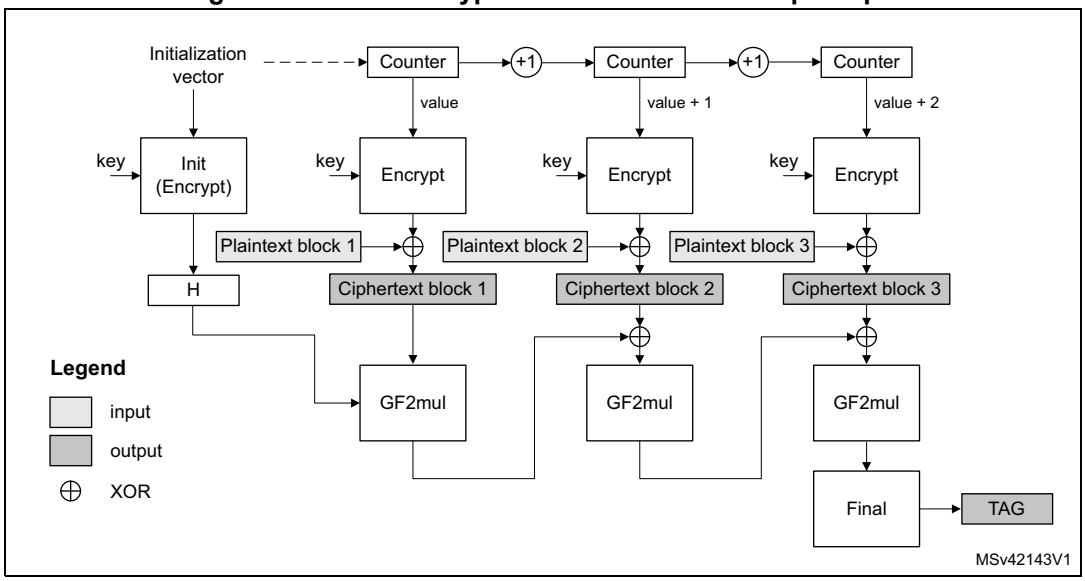


The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext as specified in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation*.

Note: Unlike with ECB and CBC modes, no key scheduling is required for the CTR decryption, since in this chaining scheme the AES core is always used in encryption mode for producing the key stream, or counter blocks.

Galois/counter mode (GCM)

Figure 328. GCM encryption and authentication principle

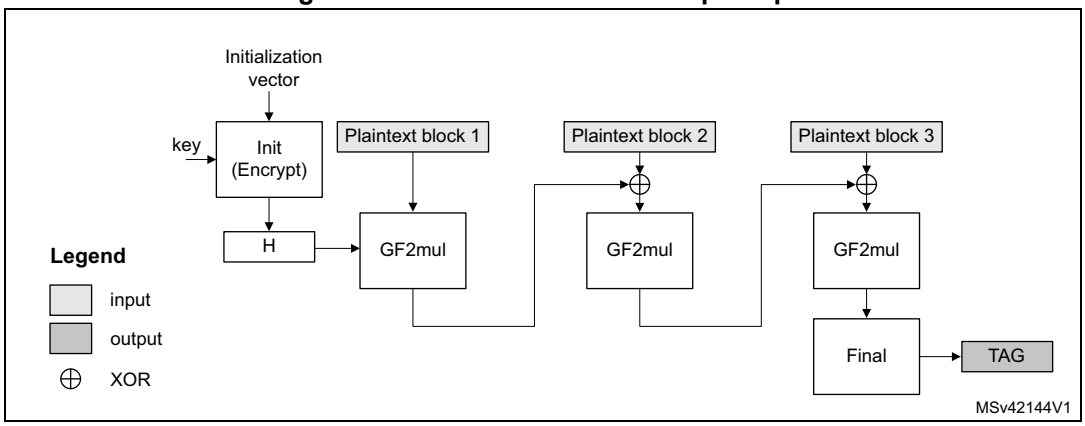


In Galois/counter mode (GCM), the plaintext message is encrypted while a message authentication code (MAC) is computed in parallel, thus generating the corresponding ciphertext and its MAC (also known as authentication tag). It is defined in NIST *Special Publication 800-38D, Recommendation for Block 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 329. GMAC authentication principle

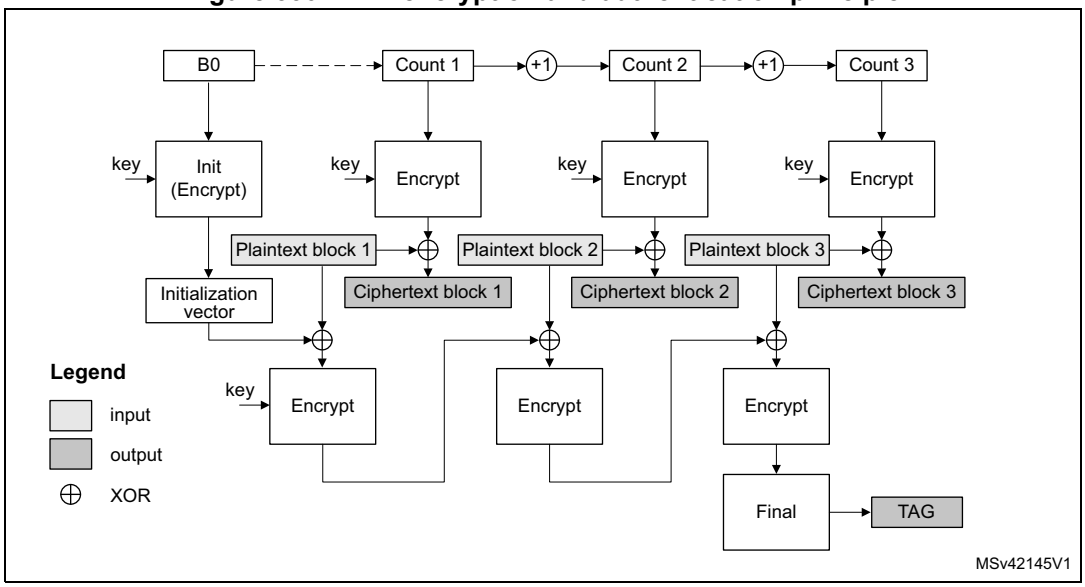


Galois message authentication code (GMAC) allows authenticating a message and generating the corresponding message authentication code (MAC). It is defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

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 330. CCM encryption and authentication principle



In Counter with cipher block chaining-message authentication code (CCM) mode, the plaintext message is encrypted while a message authentication code (MAC) is computed in parallel, thus generating the corresponding ciphertext and the corresponding MAC (also known as tag). It is described by NIST in *Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*.

CCM mode is based on AES in counter mode for confidentiality and it uses CBC for computing the message authentication code. It requires an initial value.

Like GCM, the CCM chaining mode can be applied on a message composed only by plaintext authenticated data (that is, only header, no payload). Note that this way of using CCM is not called CMAC (it is not similar to GCM/GMAC), and its use is not recommended by NIST.

41.4.4 AES procedure to perform a cipher operation

Introduction

A typical cipher operation is explained below. Detailed information is provided in sections starting from *Section 41.4.8: 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 41.4.5: 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. This step must be done before writing into key registers.
- Write a symmetric key into the AES_KEYRx registers (4 or 8 registers depending on the key size).

Note: AES sets KEYVALID in AES_SR when key information defined by KEYSIZE is loaded in AES_KEYRx.

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 when KMOD[1:0] = 00. For other KMOD[1:0] values refer to *Section 41.4.13*.

For ECB or CBC mode, refer to *Section 41.4.6: 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 CCF bit of the AES_ICR 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_IER 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 CCF bit of the AES_ICR 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 41.4.17: 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.

41.4.5 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. Select normal key mode by setting KMOD[1:0] to 00. For decryption with other KMOD[1:0] values, refer to *Section 41.4.13*.
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).

41.4.6 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 41.4.4: AES procedure to perform a cipher operation on page 1465* 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 41.4.14: AES data registers and data swapping for details).

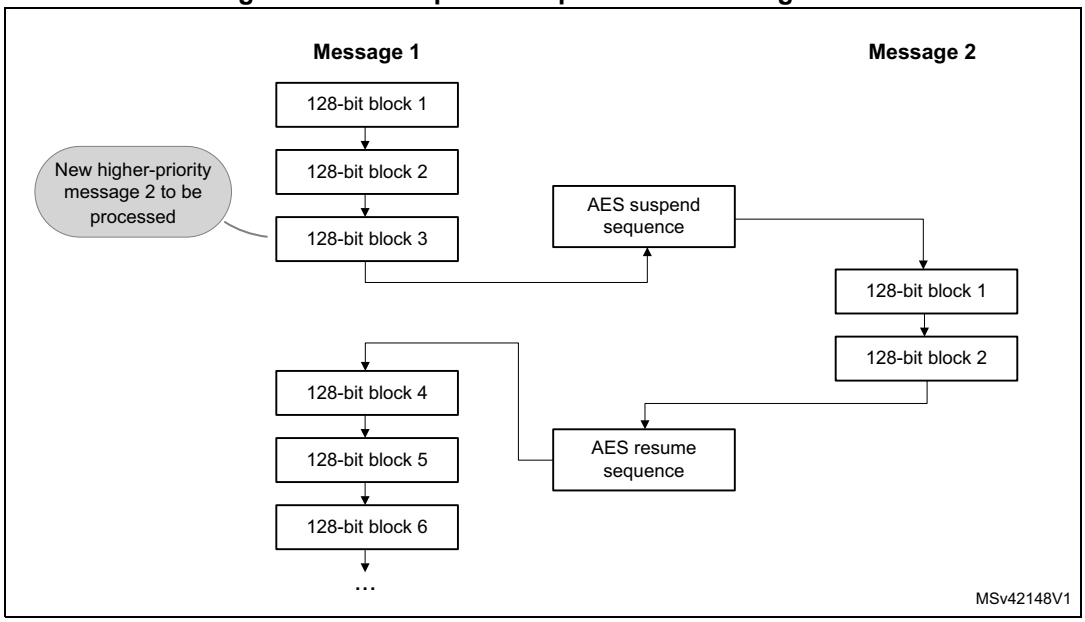
41.4.7 AES task suspend and resume

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 331 gives an example of suspend/resume operation: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

Figure 331. Example of suspend mode management



A detailed description of suspend/resume operations is in the sections dedicated to each AES mode.

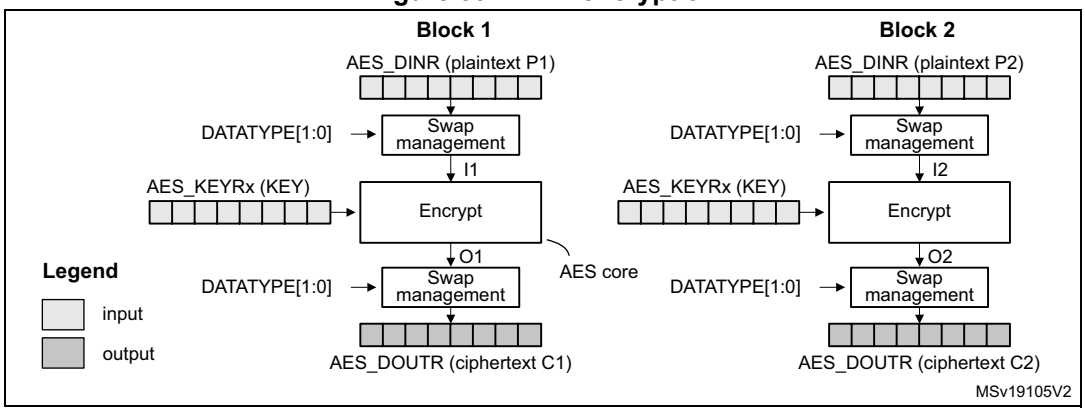
41.4.8 AES basic chaining modes (ECB, CBC)

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 332 illustrates the electronic codebook (ECB) encryption.

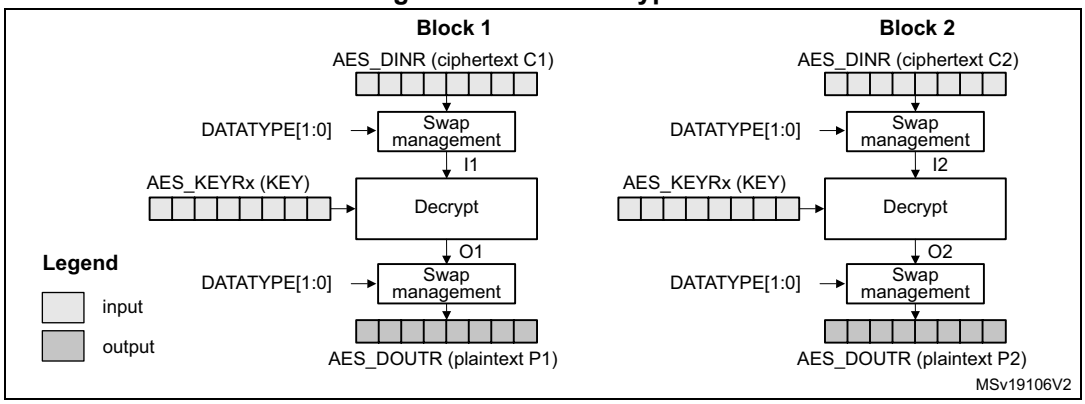
Figure 332. ECB encryption



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 333 illustrates the electronic codebook (ECB) decryption.

Figure 333. ECB decryption

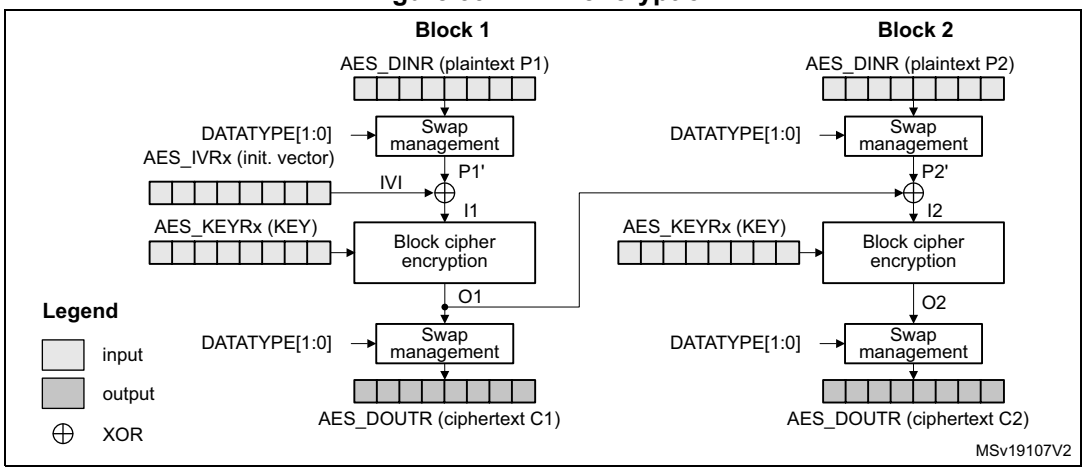


To perform an AES decryption in the ECB mode, the secret key has to be prepared by collecting the last-round encryption key (which requires to first execute the complete key schedule for encryption), and using it as the first-round key for the decryption of the ciphertext. This preparation is supported 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 334 illustrates the cipher block chaining (CBC) encryption.

Figure 334. CBC encryption

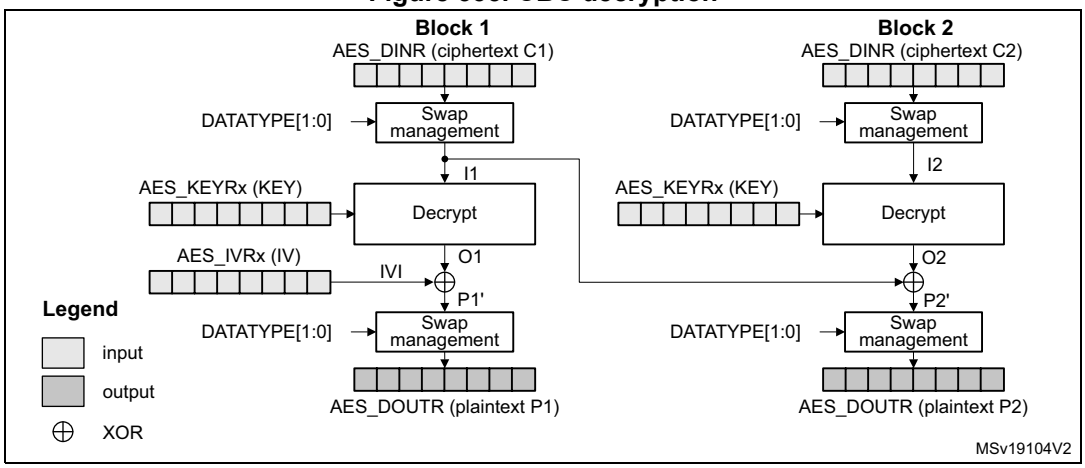


In CBC encrypt mode, the first plaintext input block, after bit/byte/half-word swapping (P1'), is XOR-ed with a 128-bit IVI bitfield (initialization vector and counter), producing the I1 input data for encrypt with the AES core, using a 128- or 256-bit key. The resulting 128-bit output block O1, after 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 41.4.6: AES ciphertext stealing and data padding*.

Figure 335 illustrates the cipher block chaining (CBC) decryption.

Figure 335. 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 41.4.6: AES ciphertext stealing and data padding*.

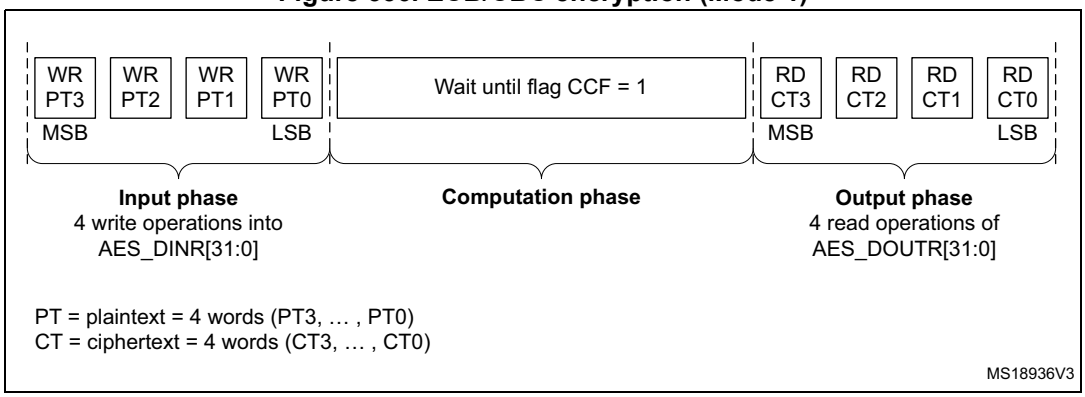
For more information on data swapping, refer to *Section 41.4.14: AES data registers and data swapping*.

ECB/CBC encryption sequence

The sequence of events to perform an ECB/CBC encryption (more detail in *Section 41.4.4*):

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. Select normal key mode by setting KMOD[1:0] to 00. For encryption with other KMOD[1:0] values, refer to *Section 41.4.13*.
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 336*.
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 336*. Then clear the CCF flag by setting the CCF bit of the AES_ICR register.
9. Repeat steps *6*-*7*-*8* to process all the blocks with the same encryption key.

Figure 336. ECB/CBC encryption (Mode 1)

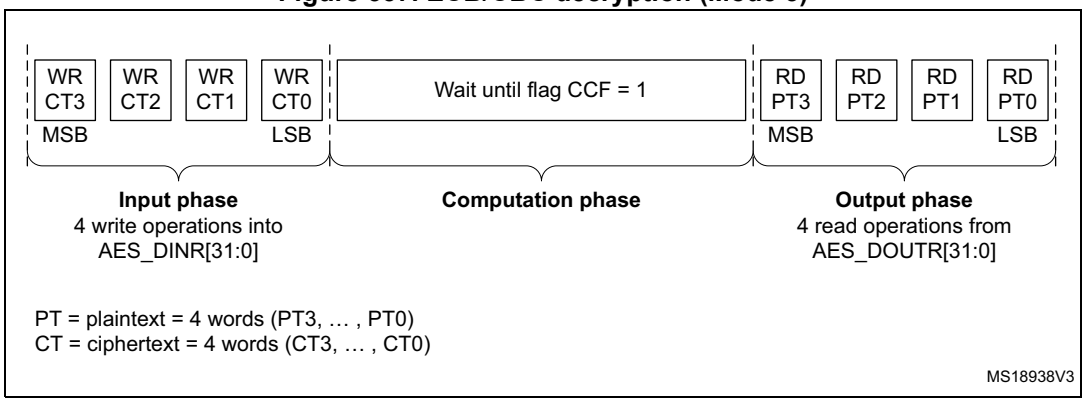


ECB/CBC decryption sequence

The sequence of events to perform an AES ECB/CBC decryption is as follows (More detail in *Section 41.4.4*). Select normal key mode by setting KMOD[1:0] to 00. For decryption with other KMOD[1:0] values, refer to *Section 41.4.13*.

1. Follow the steps described in *Section 41.4.5: 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 337*.
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 337*. Then clear the CCF flag by setting the CCF bit of the AES_ICR register.
8. Repeat steps *5*-*6*-*7* to process all the blocks encrypted with the same key.

Figure 337. 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 CCF bit of the AES_ICR register.
5. Save initialization vector registers (only required in CBC mode as AES_IVRx registers are altered during the data processing).
6. Disable the AES peripheral by clearing the bit EN of the AES_CR register.
7. Save the AES_CR register and clear the key registers if they are not needed, to process the higher priority message.
8. If DMA is used, save the DMA controller status (pointers for IN and OUT data transfers, number of remaining bytes, and so on).

To resume the processing of a message, proceed as follows:

1. If DMA is used, configure the DMA controller so as to complete the rest of the FIFO IN and FIFO OUT transfers.
2. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
3. Restore AES_CR register (with correct KEYSIZE) then restore AES_KEYRx registers.
4. Prepare the decryption key as described in *Section 41.4.5: 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.

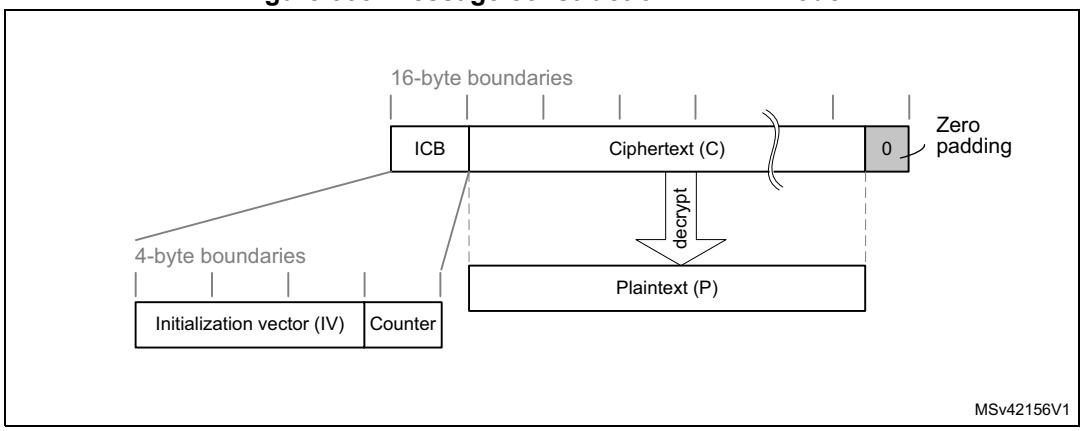
41.4.9 AES counter (CTR) mode

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 338*.

Figure 338. 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 339 and *Figure 340* 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 339. CTR encryption

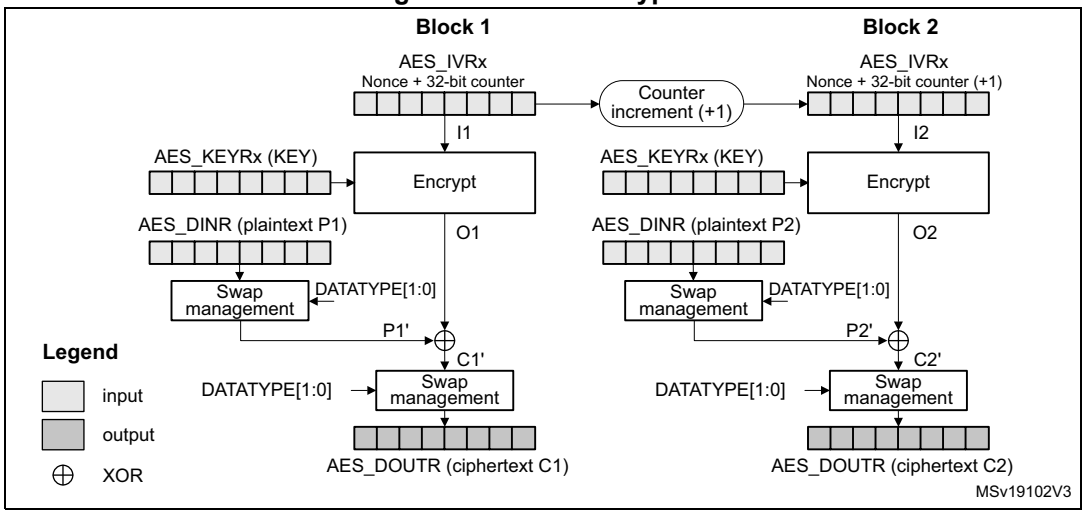
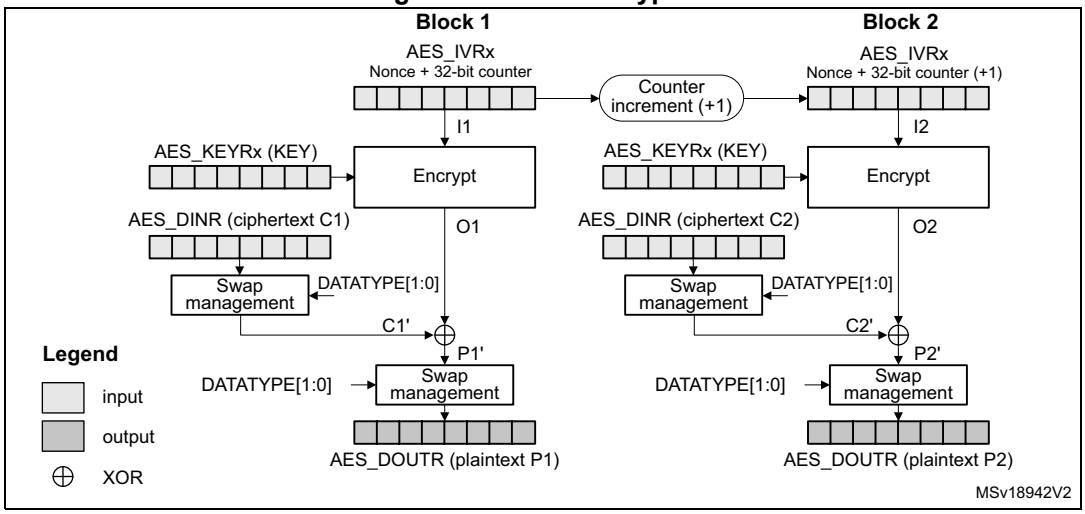


Figure 340. CTR decryption



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 362*.

Table 362. CTR mode initialization vector definition

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0} 32-bit counter = 0x0001

Unlike in CBC mode that uses the AES_IVRx registers only once when processing the first data block, in CTR mode AES_IVRx registers are used for processing each data block, and the AES peripheral increments the counter bits of the initialization vector (leaving the nonce bits unchanged).

CTR decryption does not differ from CTR encryption, since the core always encrypts the current counter block to produce the key stream that is then XOR-ed with the plaintext (CTR encryption) or ciphertext (CTR decryption) input. In CTR mode, the MODE[1:0] bitfield setting 01 (key derivation) is forbidden and all the other settings default to encryption mode.

The sequence of events to perform an encryption or a decryption in CTR chaining mode:

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
2. Select CTR chaining mode by setting to 010 the CHMOD[2:0] bitfield of the AES_CR register. Set MODE[1:0] bitfield to any value other than 01.
3. Initialize the AES_KEYRx registers, and load the AES_IVRx registers as described in *Table 362*.
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 41.4.4: 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 41.4.8: AES basic chaining modes (ECB, CBC)*.

Note: Like for CBC mode, the AES_IVRx registers must be reloaded during the resume operation.

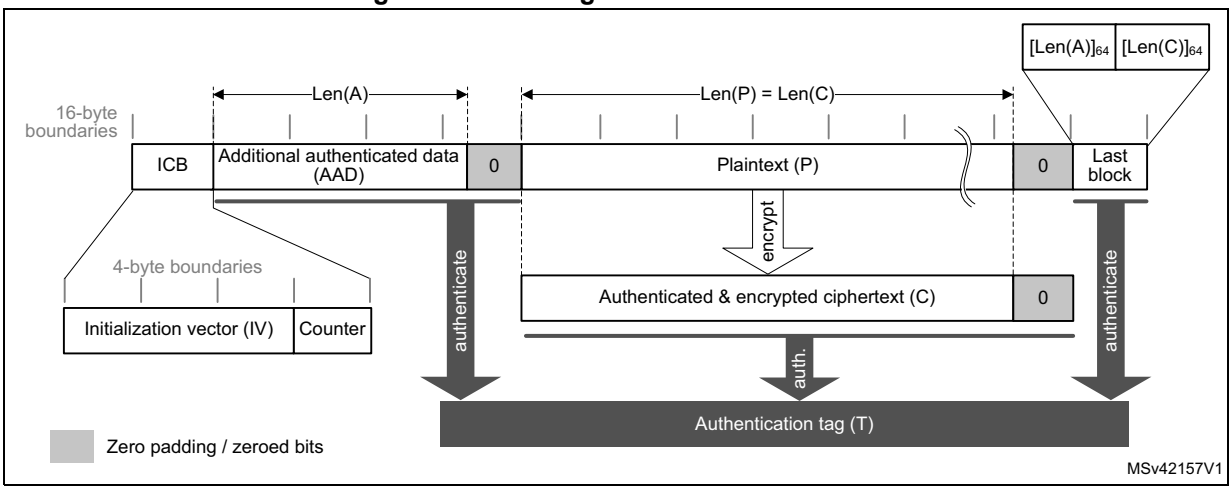
41.4.10 AES Galois/counter mode (GCM)

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 341*.

Figure 341. 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 363*.

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.

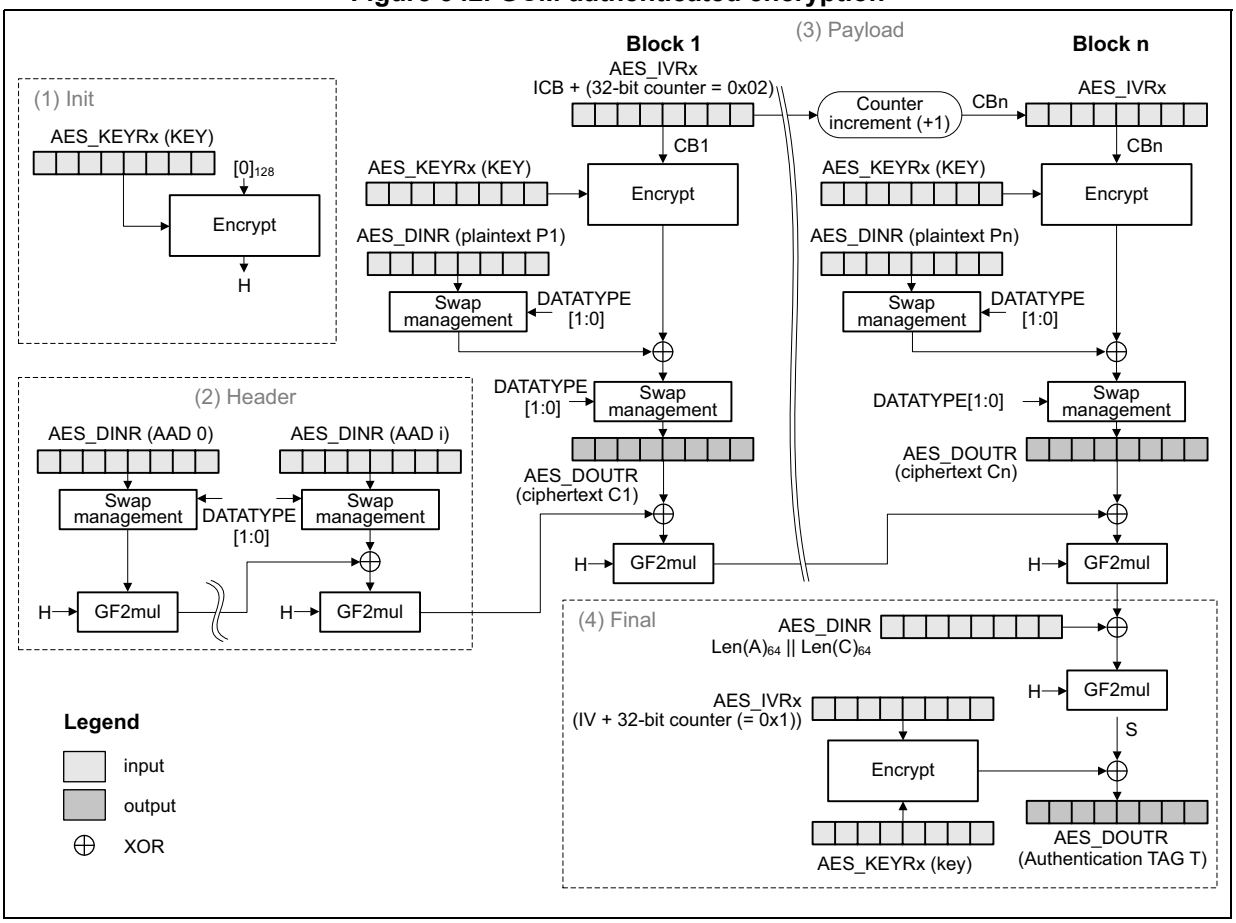
Table 363. GCM last block definition

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 342 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 342. 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 364*).

Table 364. Initialization of AES_IVRx registers in GCM mode

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
ICB[127:96]	ICB[95:64]	ICB[63:32]	ICB[31:0] 32-bit counter = 0x0002

Note: In this mode, the setting 01 of the MODE[1:0] bitfield (key derivation) is forbidden.

The authentication mechanism in GCM mode is based on a hash function called **GF2mul** that performs multiplication by a fixed parameter, called hash subkey (H), within a binary Galois field.

A GCM message is processed through the following phases, further described in next subsections:

- **Init phase**: AES prepares the GCM hash subkey (H).
- **Header phase**: AES processes the additional authenticated data (AAD), with hash computation only.
- **Payload phase**: AES processes the plaintext (P) with hash computation, counter block encryption and data XOR-ing. It operates in a similar way for ciphertext (C).
- **Final phase**: AES generates the authenticated tag (T) using the last block of the message.

GCM init phase

During this first step, the GCM hash subkey (H) is calculated and saved internally, to be used for processing all the blocks. The recommended sequence is:

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
2. Select GCM chaining mode, by setting to 011 the CHMOD[2:0] bitfield of the AES_CR register, and optionally, set the DATATYPE[1:0] bitfield.
3. Indicate the Init phase, by setting to 00 the GCMPH[1:0] bitfield of the AES_CR register.
4. Set the MODE[1:0] bitfield of the AES_CR register to 00 or 10. Although the bitfield is only used in payload phase, it is recommended to set it in the Init phase and keep it unchanged in all subsequent phases.
5. Initialize the AES_KEYRx registers with a key, and initialize AES_IVRx registers with the information as defined in *Table 364*.
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 CCF bit of the AES_ICR 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 41.4.4: 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 41.4.4: AES procedure to perform a cipher operation on page 1465*, 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 363*. 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 CCF bit of the AES_ICR 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 CCF bit of the AES_ICR register.
4. Save the AES_SUSPxR registers in the memory, where x is from 0 to 7.
5. In the payload phase, save the AES_IVRx registers as, during the data processing, they changed from their initial values. In the header phase, this step is not required.
6. Disable the AES peripheral, by clearing the EN bit of the AES_CR register.
7. Save the current AES configuration in the memory, excluding the initialization vector registers AES_IVRx. Key registers do not need to be saved as the original key value is known by the application.
8. If DMA is used, save the DMA controller status (pointers for IN data transfers, number of remaining bytes, and so on). In the payload phase, pointers for OUT data transfers must also be saved.

To resume the processing of a message, proceed as follows:

1. If DMA is used, configure the DMA controller in order to complete the rest of the FIFO IN transfers. In the payload phase, the rest of the FIFO OUT transfers must also be configured in the DMA controller.
2. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
3. Write the suspend register values, previously saved in the memory, back into their corresponding AES_SUSPxR registers, where x is from 0 to 7.
4. In the payload phase, write the initialization vector register values, previously saved in the memory, back into their corresponding AES_IVRx registers. In the header phase, write initial setting values back into the AES_IVRx registers.
5. Restore the initial setting values in the AES_CR and AES_KEYRx registers.
6. Enable the AES peripheral by setting the EN bit of the AES_CR register.

If DMA is used, enable AES DMA requests by setting the DMAINEN bit (and DMAOUTEN bit if in payload phase) of the AES_CR register.

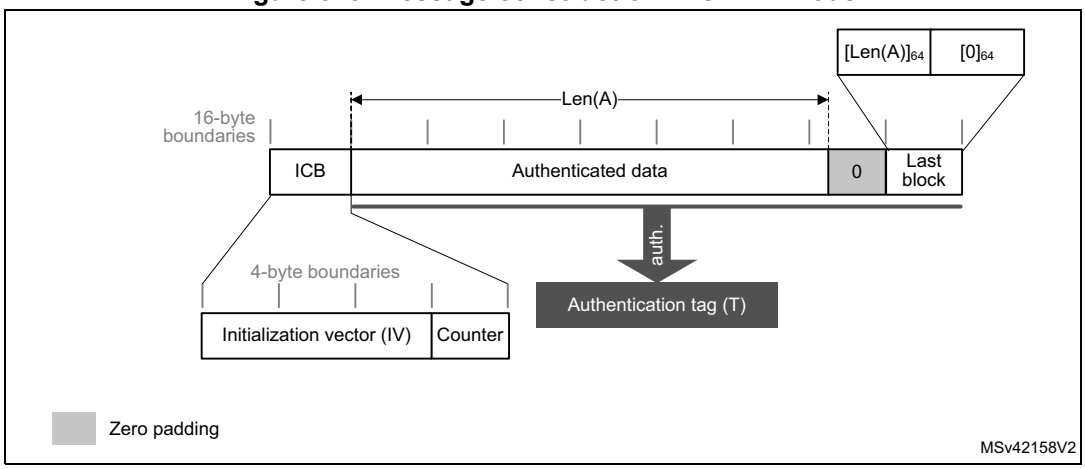
41.4.11 AES Galois message authentication code (GMAC)

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 343*.

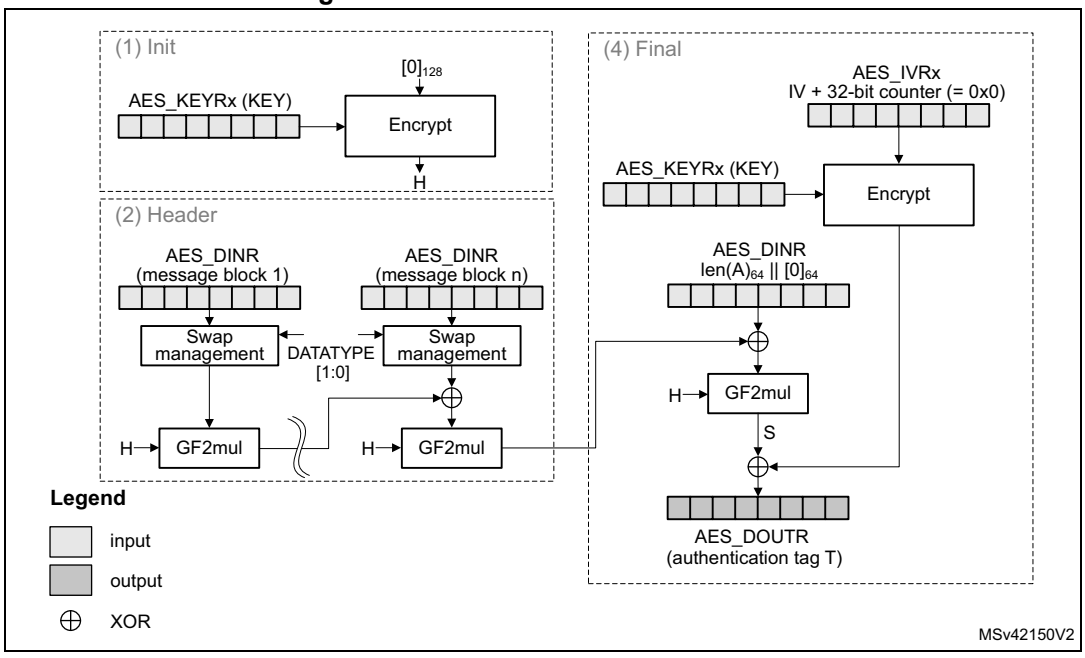
Figure 343. Message construction in GMAC mode



AES GMAC processing

Figure 344 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 344. 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.

41.4.12 AES counter with CBC-MAC (CCM)

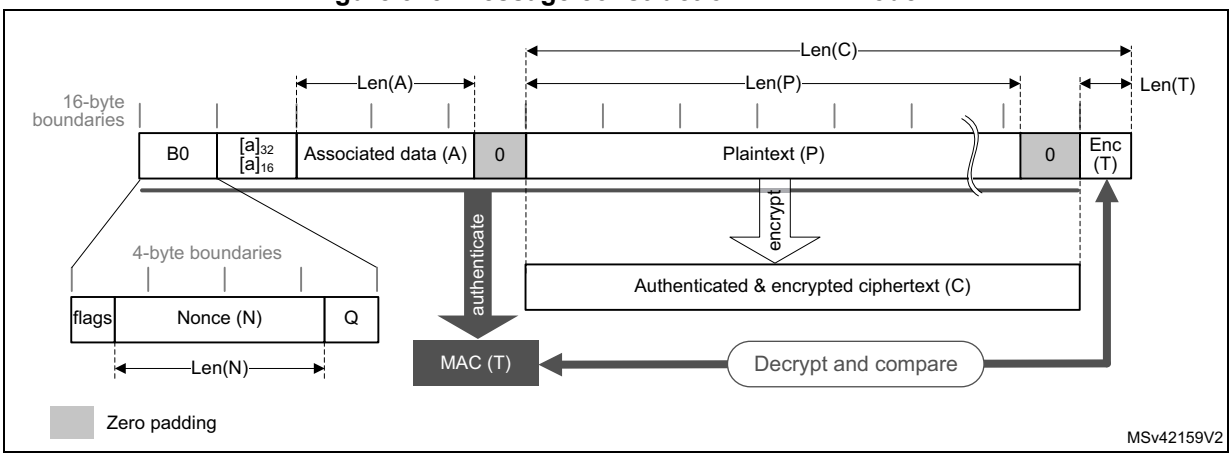
Overview

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 345*.

Figure 345. 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 345*). 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 345*).
- **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

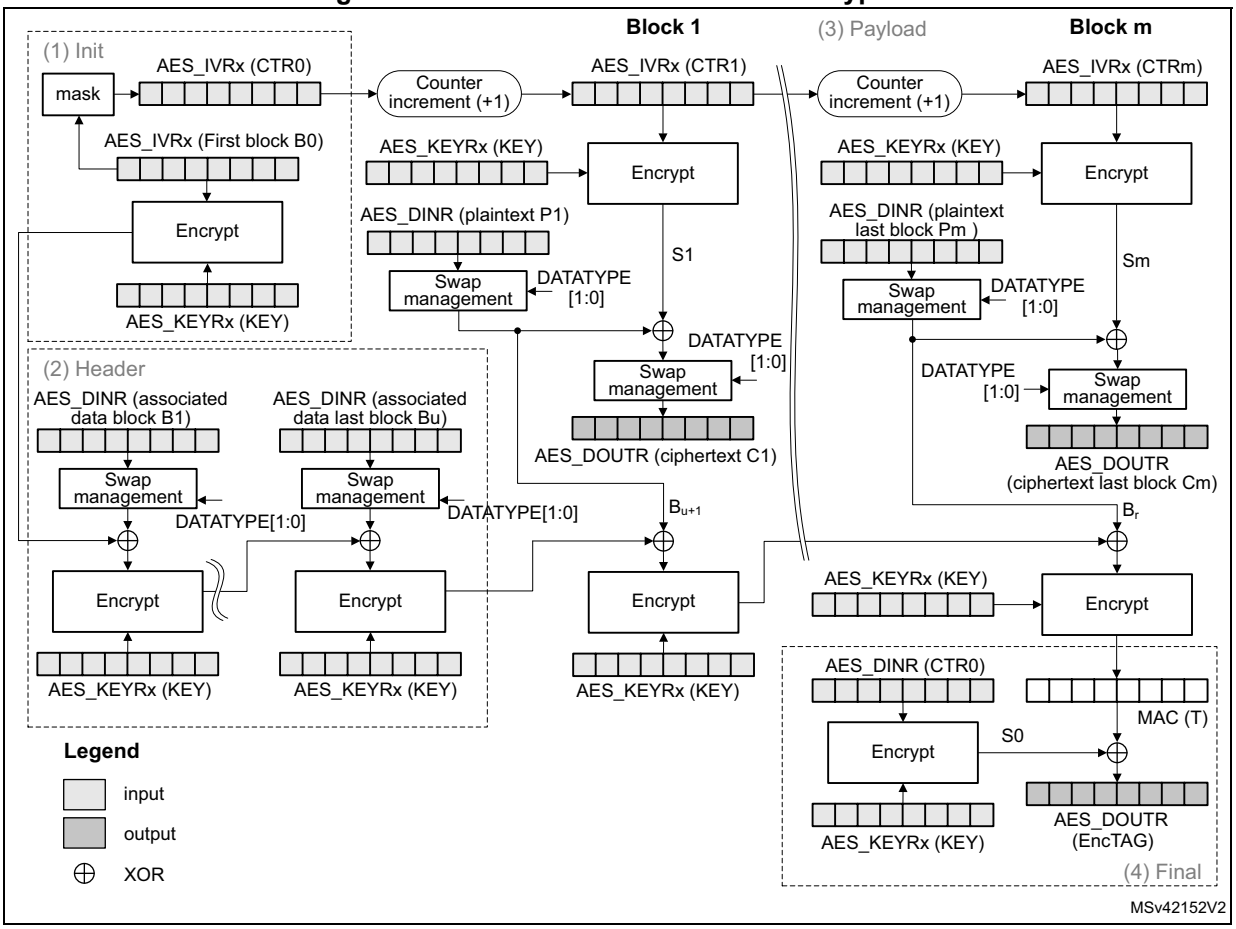
CTR1: 0710111213 141516 00000000 00000001

Generation of formatted input data blocks Bx (especially B0 and B1) must be managed by the application.

CCM processing

Figure 346 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 346. 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 365* 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 365. Initialization of AES_IVRx registers in CCM mode

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
B0[127:96]	B0[95:64]	B0[63:32]	B0[31:0]

Note: In this mode, the setting 01 of the MODE[1:0] bitfield (key derivation) is forbidden.

A CCM message is processed through the following phases, further described in next subsections:

- **Init phase**: AES processes the first block and prepares the first counter block.
- **Header phase**: AES processes associated data (A), with tag computation only.
- **Payload phase**: IP processes plaintext (P), with tag computation, counter block encryption, and data XOR-ing. It works in a similar way for ciphertext (C).
- **Final phase**: AES generates the message authentication code (MAC).

CCM Init phase

In this phase, the first block B0 of the CCM message is written into the AES_IVRx register. The AES_DOUTR register does not contain any output data. The recommended sequence is:

1. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
2. Select CCM chaining mode, by setting to 100 the CHMOD[2:0] bitfield of the AES_CR register, and optionally, set the DATATYPE[1:0] bitfield.
3. Indicate the Init phase, by setting to 00 the GCMPH[1:0] bitfield of the AES_CR register.
4. Set the MODE[1:0] bitfield of the AES_CR register to 00 or 10. Although the bitfield is only used in payload phase, it is recommended to set it in the Init phase and keep it unchanged in all subsequent phases.
5. Initialize the AES_KEYRx registers with a key, and initialize AES_IVRx registers with B0 data as described in *Table 365*.
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 CCF bit of the AES_ICR 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 41.4.4: 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 41.4.4: AES procedure to perform a cipher operation on page 1465*, 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 CCF bit of the AES_ICR 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 CCF bit of the AES_ICR register.
4. Save the AES_SUSPxR registers (where x is from 0 to 7) in the memory.
5. Save the AES_IVRx registers as, during the data processing, they changed from their initial values.
6. Disable the AES peripheral, by clearing the EN bit of the AES_CR register.
7. Save the current AES configuration in the memory, excluding the initialization vector registers AES_IVRx. Key registers do not need to be saved as the original key value is known by the application.
8. If DMA is used, save the DMA controller status (pointers for IN data transfers, number of remaining bytes, and so on). In the payload phase, pointers for OUT data transfers must also be saved.

To resume the processing of a message, proceed as follows:

1. If DMA is used, configure the DMA controller in order to complete the rest of the FIFO IN transfers. In the payload phase, the rest of the FIFO OUT transfers must also be configured in the DMA controller.
2. Disable the AES peripheral by clearing the EN bit of the AES_CR register.
3. Write the suspend register values, previously saved in the memory, back into their corresponding AES_SUSPxR registers (where x is from 0 to 7).
4. Write the initialization vector register values, previously saved in the memory, back into their corresponding AES_IVRx registers.
5. Restore the initial setting values in the AES_CR and AES_KEYRx registers.
6. Enable the AES peripheral by setting the EN bit of the AES_CR register.
7. If DMA is used, enable AES DMA requests by setting to 1 the DMAINEN bit (and DMAOUTEN bit if in payload phase) of the AES_CR register.

41.4.13 AES operation with shared keys

The AES peripheral can use the SAES peripheral as security co-processor. In this case, secure application prepares the key in the robust SAES peripheral. When ready, the AES application can load this prepared key through a dedicated hardware key bus.

Recommended sequences are described below and in SAES peripheral section *Section 42.4.10: SAES operation with shared keys*.

1. In SAES peripheral application encrypts the key to be shared, once, in Shared-key mode (KMOD[1:0] = 10).
2. Each time shared key is needed in AES peripheral application needs to decrypt it in SAES peripheral, selecting the same Shared-key mode (KMOD[1:0] = 10).
3. Once the shared key is decrypted and loaded in SAES_KEYRx registers it can be shared with the AES peripheral. Indeed, when shared key must be loaded in AES peripheral, application sets correct KEYSIZE and write KMOD[1:0] = 10 in AES_CR register. When KEYVALID is cleared, the key information is automatically transfered by hardware into AES_KEYRx, with BUSY set in AES_SR.
4. Once transfer is completed BUSY bit is cleared and KEYVALID bit is set in AES_SR register. If KEYVALID is not set when BUSY bit is cleared, or if a key error flag (KEIF) is set it means that an unexpected event occurred during the transfer (for example

hardware fault) or the KEYVALID bit in SAES_SR was cleared before the end of the transfer. When such errors occur, the whole SAES key sharing process must be restarted through the IPRST bits of control registers in both SAES and AES peripherals (see *Section 41.4.18* for details).

At that point AES is initialized with a valid, shared key. Application can proceed with the processing of data, setting KMOD[1:0] to 00.

Note: While KMOD[1:0] = 10 and BUSY = 1, if KEYSIZE in AES peripheral is different from the KEYSIZE in SAES peripheral the key sharing fails and KEIF is set in both peripherals.

41.4.14 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 41.4.17: AES DMA interface*.

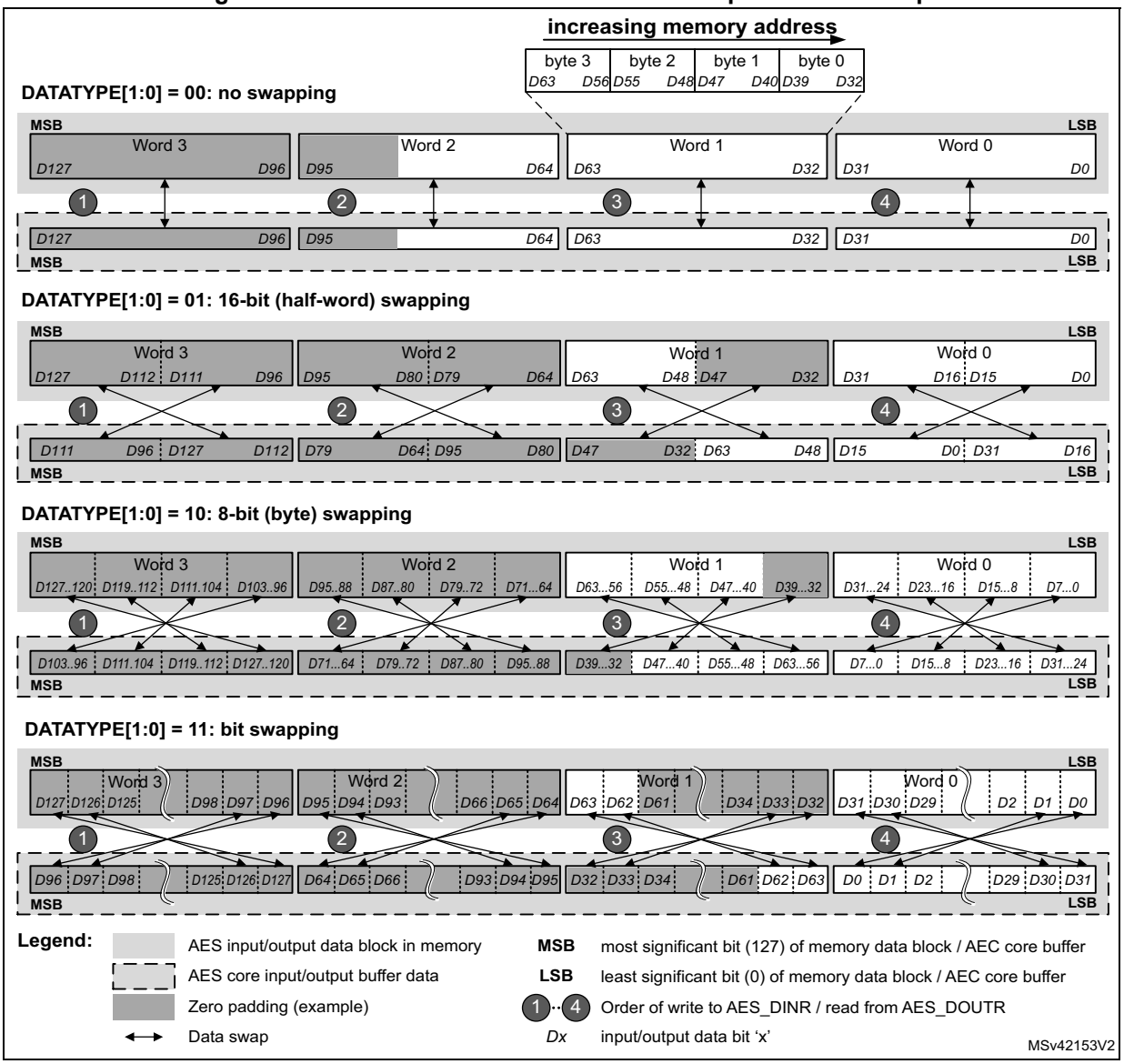
Data swapping

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 347* shows the construction of AES processing core input buffer data P127 to P0, from the input data entered through the AES_DINR register, or the construction of the output data available through the AES_DOUTR register, from the AES processing core output buffer data P127 to P0.

Figure 347. 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 347 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

41.4.15 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 366*.

Table 366. 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.

The entire key must be written before starting an AES computation. In normal key mode (KMOD[1:0] = 00), the AES_KEYRx (x = 0 to 3 for KEYSIZE = 0, x = 0 to 7 for KEYSIZE = 1) registers must always be written in either ascending or descending order.

Note: Initiating the key-loading sequence sets the BUSY flag and clears the KEYVALID flag. Once the amount of bits defined by KEYSIZE is transfered to the AES_KEYRx registers, BUSY is cleared, KEYVALID set and the EN bit becomes writable. If an error occurs, BUSY and KEYVALID are cleared and KEIF set (see Section 41.4.18: AES error management for details).
For additional information on key modes, refer to Section 41.4.13.

41.4.16 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.

41.4.17 AES DMA interface

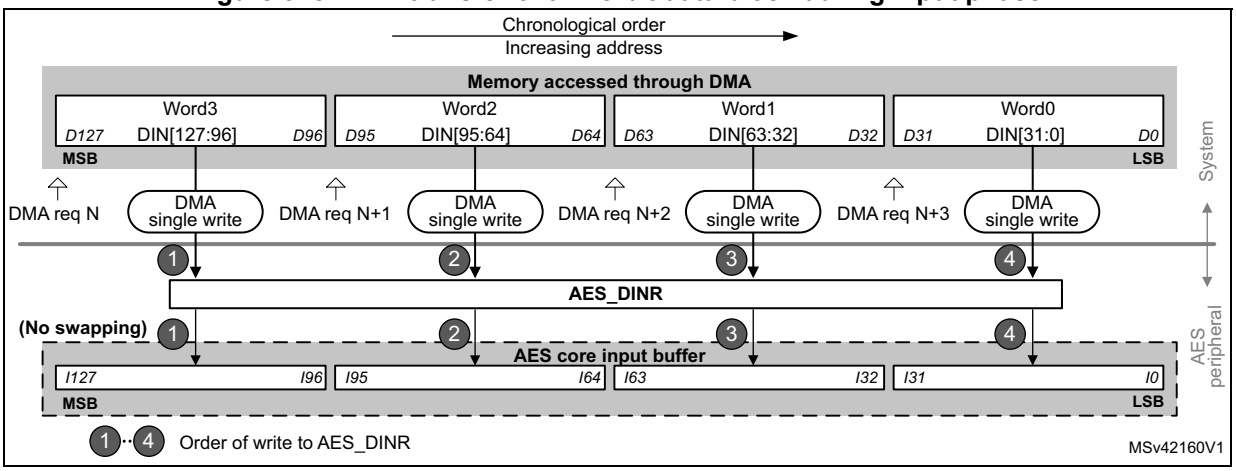
The AES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the AES_CR register.

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 348*.

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 41.4.4: AES procedure to perform a cipher operation.

Figure 348. DMA transfer of a 128-bit data block during input phase

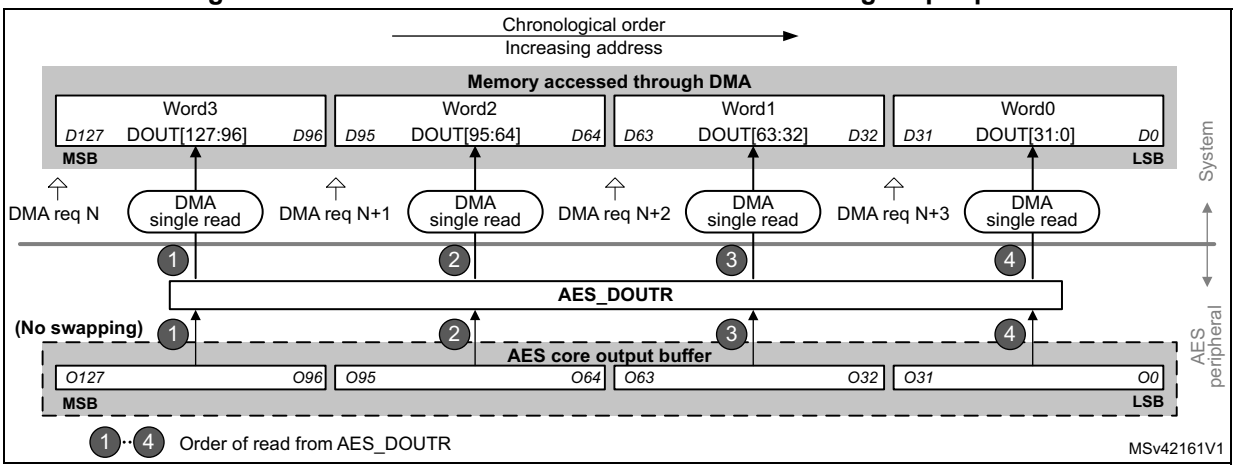


Data output using DMA

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 349*.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Figure 349. DMA transfer of a 128-bit data block during output phase



DMA operation in different operating modes

DMA operations are usable when Mode 1 (encryption) or Mode 3 (decryption) are selected via the MODE[1:0] bitfield of the register AES_CR. As in Mode 2 (key derivation) the AES_KEYRx registers must be written by software, enabling the DMA transfer through the DMAINEN and DMAOUTEN bits of the AES_CR register have no effect in that mode.

DMA single requests are generated by AES until it is disabled. So, after the data output phase at the end of processing of a 128-bit data block, AES switches automatically to a new data input phase for the next data block, if any.

When the data transferring between AES and memory is managed by DMA, the CCF flag has no use because the reading of the AES_DOUTR register is managed by DMA automatically at the end of the computation phase. The CCF flag must only be cleared when transiting back to data transferring managed by software. See *Section 41.4.4: AES procedure to perform a cipher operation*, subsection *Data append*, for details.

41.4.18 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 the RWEIF flag of the AES_ISR 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 RWEIE bit of the AES_IER register is set. For more details, refer to *Section 41.5: AES interrupts*.

The RDERR and RWEIF flag is cleared by setting the RWEIE bit of the AES_ICR register.

Write error flag (WDERR)

Unexpected write attempt of the AES_DINR register sets the WRERR flag of the AES_SR register and the RWEIF flag of the AES_ISR 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 RWEIE bit of the AES_IER register is set. For more details, refer to *Section 41.5: AES interrupts*.

The WRERR and RWEIF flag is cleared by setting the RWEIF bit of the AES_ICR register.

Key error interrupt flag (KEIF)

Failure to load a key into key registers, sets the KEIF flag of the AES_ISR register and clears the KEYVALID bit of the AES_SR register.

The KEIF flag is cleared with corresponding bit of the AES_ICR register. An interrupt is generated if the KEIE bit of the AES_IER register is set. For more details, refer to *Section 41.5: AES interrupts*.

The possible sources of key errors are:

- Key writing sequence error: an incorrect sequence has been detected when writing key registers. See *Section 41.4.15: AES key registers* for details.
- Key sharing size mismatch: error is triggered when KMOD[1:0] = 10 and application sets a KEYSIZE information in AES peripheral that does not match the KEYSIZE stored in SAES peripheral
- Key sharing error: the copy of key registers from SAES peripheral to AES failed. See *Section 41.4.13: AES operation with shared keys* for details.

Upon a key sharing error, reset both AES and SAES peripherals through the IPRST bit of their corresponding control register, then restart the key sharing sequence.

Note: For any key error, clear KEIF flag prior to re-configuring AES.

41.5 AES interrupts

Individual maskable interrupt sources generated by the AES peripheral signal the following events:

- computation completed
- read error
- write error
- key 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_IER register, and cleared by setting the corresponding bit of the AES_ICR register.

The status of each can be read from the AES_SR and AES_ISR registers.

Table 367 gives a summary of the interrupt sources, their event flags and enable bits.

Table 367. AES interrupt requests

Interrupt acronym	AES interrupt event	Event flag	Enable bit	Interrupt clear method
AES	computation completed flag	CCF	CCFIE	set CCF(1)
	read error flag	RDERR(2)	RWEIE	set RWEIF(1)
	write error flag	WRERR(2)		
	key error flag	KEIF	KEIE	set KEIF(1)

1. Bit of the AES_ICR register.
2. Flag of the AES_SR register, mirrored by the flag RWEIF of the AES_ISR register.

41.6 AES processing latency

The tables below summarize the latency to process a 128-bit block for each mode of operation.

Table 368. 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 369. 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).

41.7 AES registers

41.7.1 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
IPRST	Res.			Res.		KMOD[1:0]		NPBLB[3:0]				Res.	KEYSIZE	Res.	CHMOD[2]
rw						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.	GCMPH[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw

Bit 31 **IPRST**: AES peripheral software reset

Setting the bit resets the AES peripheral, putting all registers to their default values, except the IPRST bit itself. Hence, any key-relative data is lost. For this reason, it is recommended to set the bit before handing over the AES to a less secure application.

The bit must be low while writing any configuration registers.

Bits 30:26 Reserved, must be kept at reset value.

Bits 25:24 **KMOD[1:0]**: Key mode selection

The bitfield defines how the AES key can be used by the application:

00: Normal key

10: Shared key from SAES co-processor

Others: Reserved

With normal key selection, the key registers are freely usable, no specific usage or protection applies to AES_DIN and AES_DOUT registers.

With selection of shared key from SAES co-processor, the AES peripheral automatically loads its key registers with the data stored in the key registers of the SAES peripheral. The key value is available in key registers when BUSY bit is cleared and KEYVALID is set in the AES_SR register. Key error flag KEIF is set otherwise in the AES_ISR register.

The bitfield must be set only when KEYSIZE is correct, and when a shared key decryption sequence has been successfully completed in SAES co-processor.

Attempts to write the bitfield are ignored when the BUSY flag of AES_SR register is set, as well as 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 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 BUSY flag of AES_SR register is set, as well as 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).

Bits 10:7 Reserved, must be kept at reset value.

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 BUSY flag of AES_SR register is set, as well as 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 BUSY flag of AES_SR register is set, as well as 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 41.4.14: AES data registers and data swapping*.
Attempts to write the bitfield are ignored when the BUSY flag of AES_SR register is set, as well as 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.
The bit cannot be set as long as KEYVALID = 0.
Note: With KMOD[1:0] other than 00, use the IPRST bit rather than the bit EN.

41.7.2 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.	KEYVALID	Res.	Res.	Res.	BUSY	WRERR	RDERR	CCF
								r				r	r	r	r

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

Bit 7 **KEYVALID**: Key Valid flag

This bit is set by hardware when the amount of key information defined by KEYSIZE in AES_CR has been loaded in AES_KEYx key registers.

0: No valid key information is available in key registers. EN bit in AES_CR cannot be set.

1: Valid key information, defined by KEYSIZE in AES_CR, is loaded in key registers.

In normal mode when KEYSEL equals to zero, the application must write the key registers in the correct sequence, otherwise the KEIF flag of the AES_ISR register is set and KEYVALID stays at zero.

When KEYSEL is different from zero the BUSY flag is automatically set by AES. When key is loaded successfully, the BUSY flag is cleared and KEYVALID set. Upon an error, the KEIF flag of the AES_ISR register is set, the BUSY flag cleared and KEYVALID kept at zero.

When the KEIF flag is set, the application must clear it through the AES_ICR register, otherwise KEYVALID cannot be set. See the KEIF bit description for more details.

For more information on key loading, refer to *Section 41.4.15: AES key registers*.

Bits 6:4 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.

The flag is set when transferring a shared key from SAES peripheral.

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 RWEIF bit of the AES_ICR register.

Upon the flag setting, an interrupt is generated if enabled through the RWEIE bit of the AES_ICR 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 RWEIF bit of the AES_ICR register.

Upon the flag setting, an interrupt is generated if enabled through the RWEIE bit of the AES_ICR register.

The flag setting has no impact on the AES operation. Unexpected read returns zero.

Bit 0 **CCF**: Computation completed flag

This bit mirrors the CCF bit of the AES_ISR register.

41.7.3 AES data input register (AES_DINR)

Address offset: 0x08

Reset value: 0x0000 0000

Only 32-bit write access type is supported.

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

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 41.4.14: AES data registers and data swapping on page 1489*.

41.7.4 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 41.4.14: AES data registers and data swapping on page 1489*.

41.7.5 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). A special writing sequence is also required, as described in KEYVALID bit of the AES_SR register. Note that, if KMOD[1:0] = 10 (shared key), the key is directly loaded from SAES peripheral to AES_KEYRx registers (hence writes to key register is ignored and KEIF is set).

Refer to *Section 41.4.15: AES key registers on page 1491* for more details.

41.7.6 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.

41.7.7 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.

41.7.8 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.

41.7.9 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 41.4.16: AES initialization vector registers on page 1491* 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

41.7.10 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.

41.7.11 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.

41.7.12 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.

41.7.13 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.

41.7.14 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.

41.7.15 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.

41.7.16 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).

41.7.17 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.

41.7.18 AES interrupt enable register (AES_IER)

Address offset: 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	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	RWEIE	CCFIE
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIE:** Key error interrupt enable

This bit enables or disables (masks) the AES interrupt generation when KEIF (key error flag) is set.
0: Disabled (masked)
1: Enabled (not masked)

Bit 1 **RWEIE:** Read or write error interrupt enable

This bit enables or disables (masks) the AES interrupt generation when RWEIF (read and/or write error flag) is set.
0: Disabled (masked)
1: Enabled (not masked)

Bit 0 **CCFIE:** Computation complete flag interrupt enable

This bit enables or disables (masks) the AES interrupt generation when CCF (computation complete flag) is set.
0: Disabled (masked)
1: Enabled (not masked)

41.7.19 AES interrupt status register (AES_ISR)

Address offset: 0x304

Reset value: 0x0000 0000

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

15	14	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	RWEIF	CCF
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIF:** Key error interrupt flag

This read-only bit is set by hardware when key information failed to load into key registers.

0: No key error detected

1: Key information failed to load into key registers

Setting the corresponding bit of the AES_ICR register clears the KEIF and generates interrupt if the KEIE bit of the AES_IER register is set.

KEIF is triggered upon any of the following errors:

–AES_KEYRx register write does not respect the correct order. (For KEYSIZE = 0, AES_KEYR0 then AES_KEYR1 then AES_KEYR2 then AES_KEYR3 register, or reverse. For KEYSIZE = 1, AES_KEYR0 then AES_KEYR1 then AES_KEYR2 then AES_KEYR3 then AES_KEYR4 then AES_KEYR5 then AES_KEYR6 then AES_KEYR7, or reverse).

KEIF must be cleared by the application software, otherwise KEYVALID cannot be set.

Bit 1 **RWEIF:** Read or write error interrupt flag

This read-only bit is set by hardware when a RDERR or a WRERR error flag is set in the AES_SR register.

0: No read or write error detected

1: Read or write error detected (see AES_SR register for details)

RWEIF bit is cleared when application sets the corresponding bit of AES_ICR register. An interrupt is generated if the RWEIE bit has been previously set in the AES_IER register.

This flags has no meaning when key derivation mode is selected.

Bit 0 **CCF:** Computation complete 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 CCF bit of the AES_ICR register.

Upon the flag setting, an interrupt is generated if enabled through the CCFIE bit of the AES_IER register.

The flag is significant only when the DMAOUTEN bit is 0. It may stay high when DMA_EN is 1.

41.7.20 AES interrupt clear register (AES_ICR)

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	RWEIF	CCF
													w	w	w

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **KEIF:** Key error interrupt flag clear
Setting this bit clears the KEIF status bit of the AES_ISR register.

Bit 1 **RWEIF:** Read or write error interrupt flag clear
Setting this bit clears the RWEIF status bit of the AES_ISR register, and both RDERR and WRERR flags in the AES_SR register.

Bit 0 **CCF:** Computation complete flag clear
Setting this bit clears the CCF status bit of the AES_SR and AES_ISR registers.

41.7.21 AES register map

Table 370. 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	IPRST	Res.	Res.	Res.	Res.	Res.	KMOD[1]	KMOD[0]	NPBLB[3:0]				Res.	KEYSIZE	Res.	CHMOD[2]	Res.	GCMPH[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	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
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.	KEYVALID	Res.	Res.	Res.	BUSY	WRERR	RDERR	CCF
	Reset value																									0				0	0	0	0
0x008	AES_DINR	DIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	AES_DOUTR	DOUT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	AES_KEYR0	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	AES_KEYR1	KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	AES_KEYR2	KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	AES_KEYR3	KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	AES_IVR0	IVI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	AES_IVR1	IVI[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	AES_IVR2	IVI[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	AES_IVR3	IVI[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 370. AES register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x030	AES_KEYR4	KEY[159:128]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	AES_KEYR5	KEY[191:160]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	AES_KEYR6	KEY[223:192]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	AES_KEYR7	KEY[255:224]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	AES_SUSP0R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	AES_SUSP1R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	AES_SUSP2R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	AES_SUSP3R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	AES_SUSP4R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	AES_SUSP5R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x058	AES_SUSP6R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	AES_SUSP7R	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060-0x2FF	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.
0x300	AES_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	RWEIE	CCFIE
	Reset value																														0	0	0
0x304	AES_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	RWEIF	CCF
	Reset value																														0	0	0
0x308	AES_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	RWEIF	CCF
	Reset value																														0	0	0

Refer to *Section 2.3 on page 115* for the register boundary addresses.

42 Secure AES coprocessor (SAES)

42.1 Introduction

The secure AES coprocessor (SAES) 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. It incorporates a protection against side-channel attacks (SCA), including differential power analysis (DPA), certified SESIP and PSA security assurance level 3.

The peripheral supports ECB, and CBC chaining modes for key sizes of 128 or 256 bits, as well as special modes such as hardware secret key encryption/ decryption (Wrapped-key mode) and key sharing with faster AES peripheral (Shared-key mode).

SAES has the possibility to load by hardware STM32 hardware secret keys (boot hardware key BHK and derived hardware unique key DHUK), usable but not readable by application.

SAES 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). It is hardware-linked with the true random number generator (TRNG) and with the AES peripheral.

42.2 SAES 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
- 528 or 743 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)
  - Optional hardware loading of two hardware secret keys (BHK, DHUK) that can be XOR-ed together
- 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 SAES needs to process another message with a higher priority, then resume the original message
- Security context enforcement for keys
- Hardware secret key encryption/ decryption (Wrapped-key mode)
- Protection against side-channel attacks (SCA), incl. differential power analysis (DPA), certified SESIP and PSA security assurance level 3
- Hardware key sharing with faster AES peripheral (Shared-key mode), controlled by SAES

42.3 SAES implementation

The devices have one SAES and one AES peripheral, implemented as shown in the following table.

Table 371. AES/SAES features

AES/SAES modes/features(1)	AES	SAES
ECB, CBC chaining	X	X
CTR, CCM, GCM chaining	X	-
AES 128-bit ECB encryption in cycles	51	528
DHUK and BHK key selection	-	X
Side-channel attacks resistance	-	X
Shared key between SAES and AES	X	

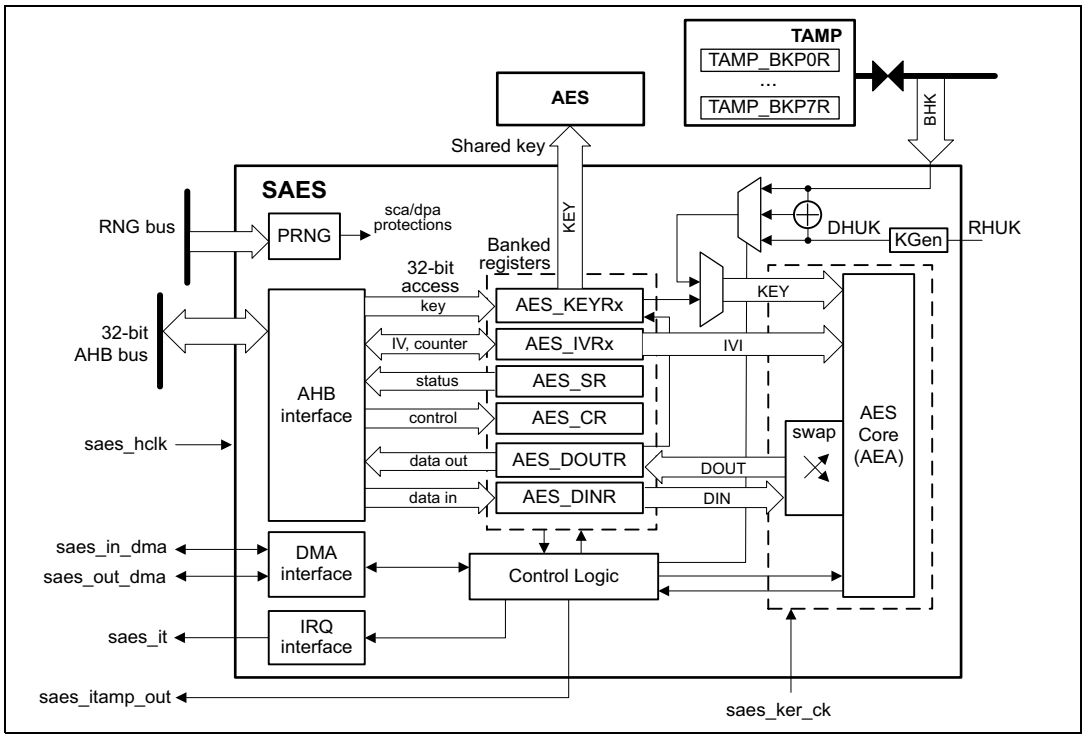
1. X = supported.

42.4 SAES functional description

42.4.1 SAES block diagram

Figure 350 shows the block diagram of SAES.

Figure 350. SAES block diagram



Note: AES represents the AES peripheral. The saes_ker_ck represents the rcc_shsi_ck clock signal.

42.4.2 SAES internal signals

Table 372 describes the user relevant internal signals interfacing the SAES peripheral.

Table 372. SAES internal input/output signals

Signal name	Signal type	Description
saes_hclk	Input	AHB bus clock
saes_it	Output	SAES interrupt request
saes_in_dma	Input/Output	Input DMA single request/acknowledge
saes_out_dma	Input/Output	Output DMA single request/acknowledge

Table 372. SAES internal input/output signals (continued)

Signal name	Signal type	Description
saes_itamp_out	Output	Tamper event signal to TAMP (XOR-ed), triggered when an unexpected hardware fault occurs. When this signal is triggered, SAES automatically clears key registers. A reset is required for SAES to be usable again.
rcc_shsi_ck	Input	Dedicated SHSI clock from RCC
RHUK	Input	256-bit root hardware unique key (non-volatile, unique per device and secret to software), used to internally compute the derived hardware unique key (DHUK)
BHK(1)	Input	256-bit boot hardware key (BHK) stored in tamper-resistant secure backup registers and written by a secure code during boot. Once written, this key cannot be read nor written by any application until the next product reset.

1. Connected to a set of backup registers in TAMP peripheral that are written, then read/write locked, by the application software (see *Section 42.4.12* for details).

42.4.3 SAES cryptographic core

Overview

The SAES cryptographic core consists of the following components:

- AES core algorithm (AEA)
- key input
- initialization vector (IV) input
- chaining algorithm logic (XOR)

The SAES core works on 128-bit data blocks (four words) with 128-bit or 256-bit key length. Depending on the chaining mode, the SAES requires zero or one 128-bit initialization vector IV.

The SAES features the following modes of operation:

- **Mode 1:**

  Plaintext encryption using a key stored in the SAES_KEYRx registers or read using a dedicated hardware bus
- **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 SAES_KEYRx registers. Now the SAES peripheral is ready to switch to Mode 3 for executing data decryption.
- **Mode 3:**

  Ciphertext decryption using a key stored in the SAES_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 SAES_CR register. It may be done only when the SAES peripheral is disabled.

Special key operation is selected using the KMOD[1:0] bitfield of the SAES_CR register. See *Section 42.4.10* and *Section 42.4.9* for details.

Typical data processing

Typical usage of the SAES is described in *Section 42.4.4: SAES procedure to perform a cipher operation on page 1515*.

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 SAES, selected through the CHMOD[2:0] bitfield of the SAES_CR register:

- Electronic code book (ECB)
- Cipher block chaining (CBC)

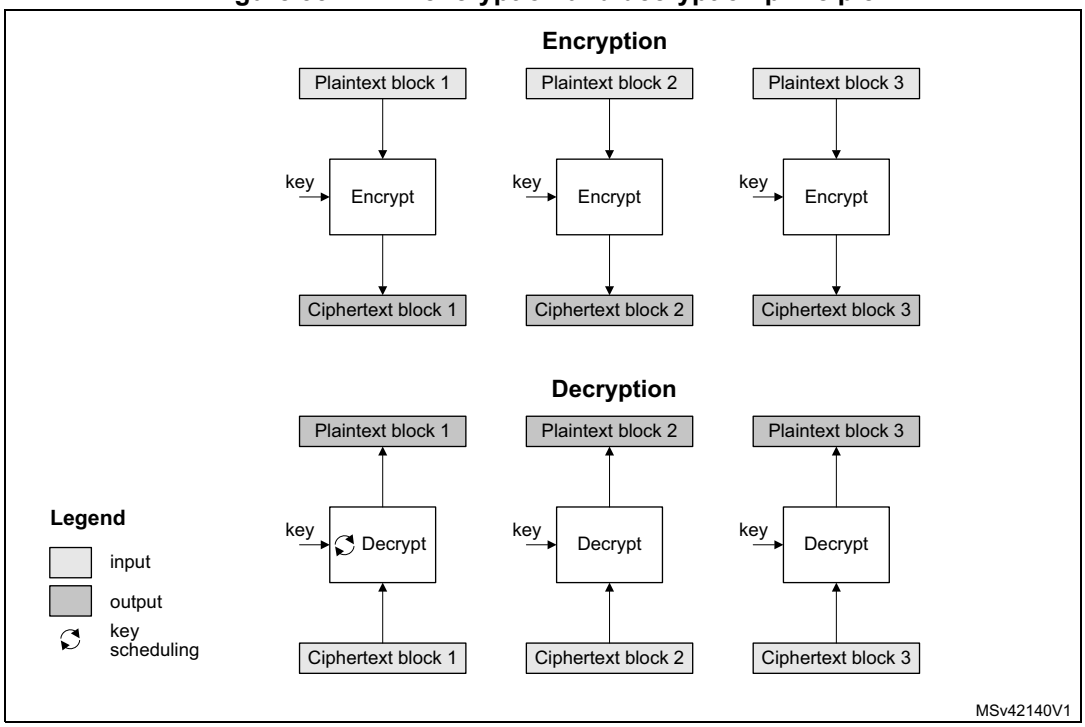
Note: The chaining mode may be changed only when SAES is disabled (bit EN of the SAES_CR register cleared).

Principle of each SAES chaining mode is provided in the following subsections.

Detailed information is in dedicated sections, starting from *Section 42.4.8: SAES basic chaining modes (ECB, CBC)*.

Electronic codebook (ECB) mode

Figure 351. ECB encryption and decryption principle

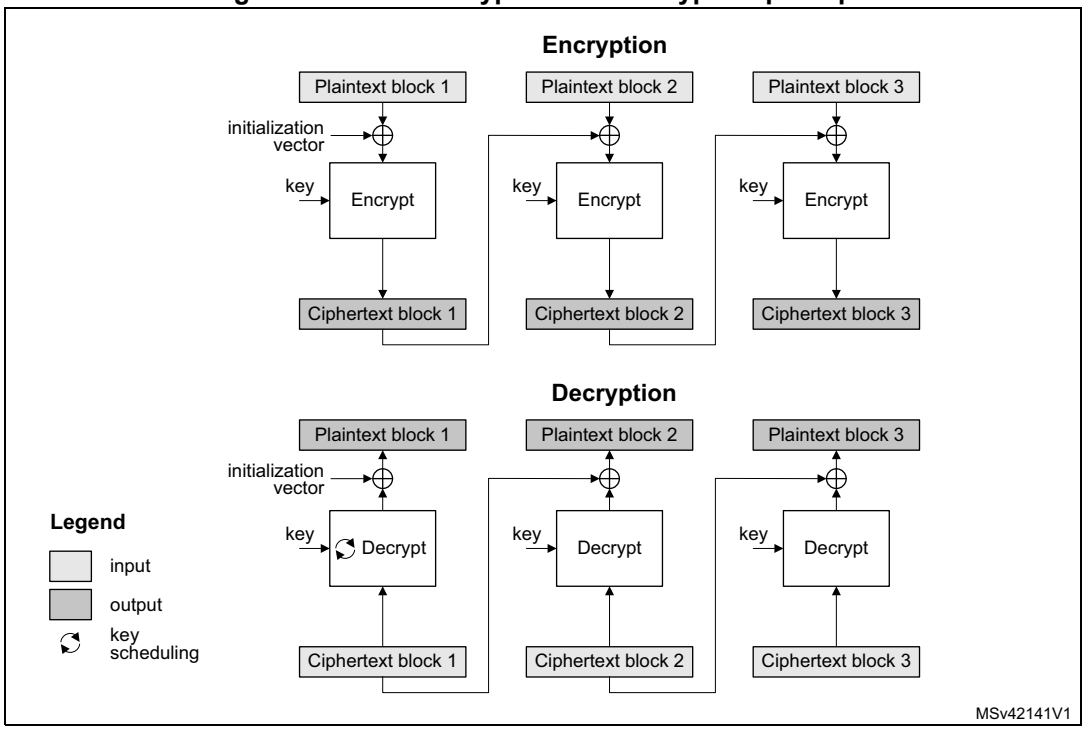


ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately.

Note: For decryption, a special key scheduling is required before processing the first block.

Cipher block chaining (CBC) mode

Figure 352. 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.

42.4.4 SAES procedure to perform a cipher operation

Introduction

A typical cipher operation is explained below. Detailed information is provided in sections starting from *Section 42.4.8: SAES basic chaining modes (ECB, CBC)*.

Initialization of SAES

To initialize SAES, first disable it by clearing the EN bit of the SAES_CR register. Then perform the following steps in any order (except KEYSIZE):

- Verify that BUSY = 0 in SAES_SR (no RNG random number fetch in progress), and RNGEIF = 0 in SAES_ISR (no random number fetching error flagged).
- Configure the SAES mode, by programming the MODE[1:0] bitfield of the SAES_CR register.
  - For encryption, select Mode 1 (MODE[1:0] = 00).
  - For decryption, select Mode 2 (MODE[1:0] = 01) then Mode 3 (MODE[1:0] = 10), as described in *Section 42.4.8: SAES basic chaining modes (ECB, CBC)*.
- Select the chaining mode, by programming the CHMOD[2:0] bitfield of the SAES_CR register. Select normal key mode by setting KMOD[1:0] to 00. For the other KMOD[1:0] values, refer to *Section 42.4.9* and *Section 42.4.10*.
- Configure the data type (1-, 8-, 16- or 32-bit), with the DATATYPE[1:0] bitfield in the SAES_CR register.
- When it is required (for example in CBC chaining mode), write the initialization vector into the SAES_IVRx registers.
- Configure the key size (128-bit or 256-bit), with the KEYSIZE bitfield of the SAES_CR register. This step must be done before writing into key registers or selecting a different key source using KEYSEL. If the key must not be shared with a different security context (different secure attribute), the KEYPROT bit of the SAES_CR register must also be set.
- Write a symmetric key into the SAES_KEYRx registers (4 or 8 registers depending on the key size). Alternatively, select a key source different from the key registers by setting KEYSEL[2:0] to a value different from 0x0. See *Section 42.4.12* for details.

Note: SAES sets KEYVALID in SAES_SR when key information defined by KEYSIZE is loaded in SAES_KEYRx.

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 when KMOD[1:0] = 00. For other KMOD[1:0] values refer to *Section 42.4.10* and *Section 42.4.9*.

Data append through polling

This method uses flag polling to control the data append through the following sequence:

1. Enable the SAES peripheral by setting the EN bit of the SAES_CR register.
2. Repeat the following sub-sequence until the payload is entirely processed:
   a) Write four input data words into the SAES_DINR register.
   b) Wait until the status flag CCF is set in the SAES_SR, then read the four data words from the SAES_DOUTR register.
   c) Clear the CCF flag, by setting the CCF bit of the SAES_ISR 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, refer to *Section 42.4.6: SAES ciphertext stealing and data padding*.
3. As it is the last block, follow the instructions in *Section 42.4.6: SAES ciphertext stealing and data padding,* then disable the SAES peripheral by clearing the EN bit of the SAES_CR register.

Note: Up to three wait cycles are automatically inserted between two consecutive writes to the SAES_DINR register, to allow sending the key to the AES co-processor.

Data append using interrupt

The method uses interrupt from the SAES peripheral to control the data append, through the following sequence:

1. Enable interrupts from SAES by setting the CCFIE bit of the SAES_IER register.
2. Enable the SAES peripheral by setting the EN bit of the SAES_CR register.
3. Write first four input data words into the SAES_DINR register.
4. Handle the data in the SAES interrupt service routine, upon interrupt:
   a) Read four output data words from the SAES_DOUTR register.
   b) Clear the CCF flag and thus the pending interrupt, by setting the CCF bit of the SAES_ISR 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, refer to *Section 42.4.6: SAES ciphertext stealing and data padding*. Then proceed with point 4e).
   d) If the data block just processed is the last block of the message, follow if needed *Section 42.4.6: SAES ciphertext stealing and data padding*, then disable the SAES peripheral by clearing the EN bit of the SAES_CR register and quit the interrupt service routine.
   e) Write next four input data words into the SAES_DINR register and quit the interrupt service routine.

Note: SAES is tolerant of delays between consecutive read or write operations, which allows, for example, an interrupt from another peripheral to be served between two SAES computations.

Data append using DMA

With this method, all the transfers and processing are managed by DMA and SAES. To use the method, proceed as follows:

1. If the last block to process is inferior to 128 bits, refer to *Section 42.4.6: SAES ciphertext stealing and data padding* to prepare the last four-word data block.
2. Configure the DMA controller so as to transfer the data to process from the memory to the SAES peripheral input and the processed data from the SAES peripheral output to the memory, as described in *Section 42.4.14: SAES DMA interface*. Configure the DMA controller so as to generate an interrupt on transfer completion.
3. Enable the SAES peripheral by setting the EN bit of the SAES_CR register
4. Enable DMA requests by setting the DMAINEN and DMAOUTEN bits of the SAES_CR register.
5. Upon DMA interrupt indicating the transfer completion, get the SAES-processed data from the memory.

Note: The CCF flag has no use with this method, because the reading of the SAES_DOUTR register is managed by DMA automatically, without any software action, at the end of the computation phase.

42.4.5 SAES 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.

Recommended method is to select the Mode 2 by setting to 01 the MODE[1:0] bitfield of the SAES_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. Verify that BUSY = 0 in SAES_SR (no RNG random number fetch in progress).
2. Disable the SAES peripheral by clearing the EN bit of the SAES_CR register.
3. Select Mode 2 by setting to 01 the MODE[1:0] bitfield of the SAES_CR. The CHMOD[2:0] bitfield is not significant in this case because this key derivation mode is independent of the chaining algorithm selected. Select normal key mode by setting KMOD[1:0] to 00. For decryption with other KMOD[1:0] values, refer to *Section 42.4.10* and *Section 42.4.9*.
4. Set key length to 128 or 256 bits, via KEYSIZE bit of SAES_CR register. If the key must not be shared with a different security context (different secure attribute), the KEYPROT bit of the SAES_CR register must also be set.
5. Write the SAES_KEYRx registers (128 or 256 bits) with encryption key, or, alternatively, select a key source different from the key registers, through KEYSEL[2:0]. Refer to *Section 42.4.12: SAES key registers* for details. Writes to the SAES_IVRx registers have no effect.
6. Enable the SAES peripheral, by setting the EN bit of the SAES_CR register.
7. Wait until the CCF flag is set in the SAES_SR register.
8. Clear the CCF flag. Derived key is available in AES core, ready to use for decryption.

Note: The SAES is disabled by hardware when the derivation key is available.

To restart a derivation key computation, repeat steps 5, 6, 7, and 8.

Note: The operation of the key preparation lasts 200 or 324 clock cycles, depending on the key size (128- or 256-bit).

42.4.6 SAES ciphertext stealing and data padding

When using SAES 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 SAES 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.

Note: Padding data are swapped in a similar way as normal data, according to the DATATYPE[1:0] field of the SAES_CR register (see Section 42.4.11: SAES data registers and data swapping for details).

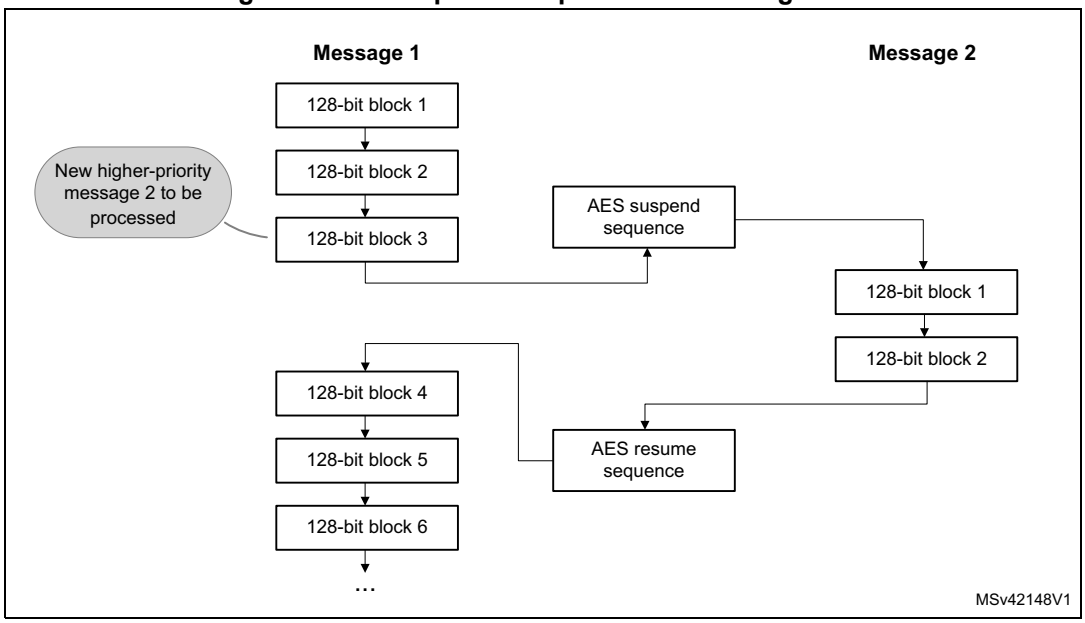
42.4.7 SAES task suspend and resume

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 SAES is enabled again to receive the next data block.

Figure 353 gives an example of suspend/resume operation: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

Figure 353. Example of suspend mode management



A detailed description of suspend/resume operations is in the sections dedicated to each SAES mode.

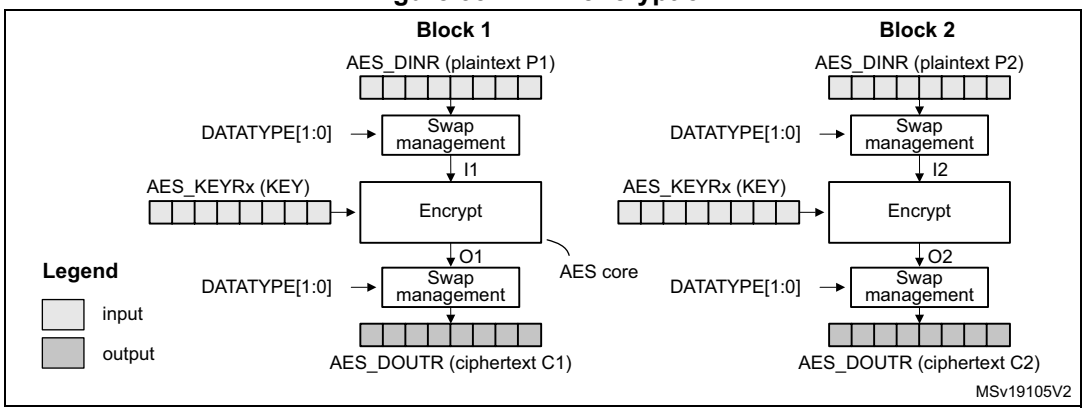
42.4.8 SAES basic chaining modes (ECB, CBC)

Overview

This section gives a brief explanation of the four basic operation modes provided by the SAES core: ECB encryption, ECB decryption, CBC encryption and CBC decryption. For detailed information, refer to the FIPS publication 197 from November 26, 2001.

Figure 354 illustrates the electronic codebook (ECB) encryption.

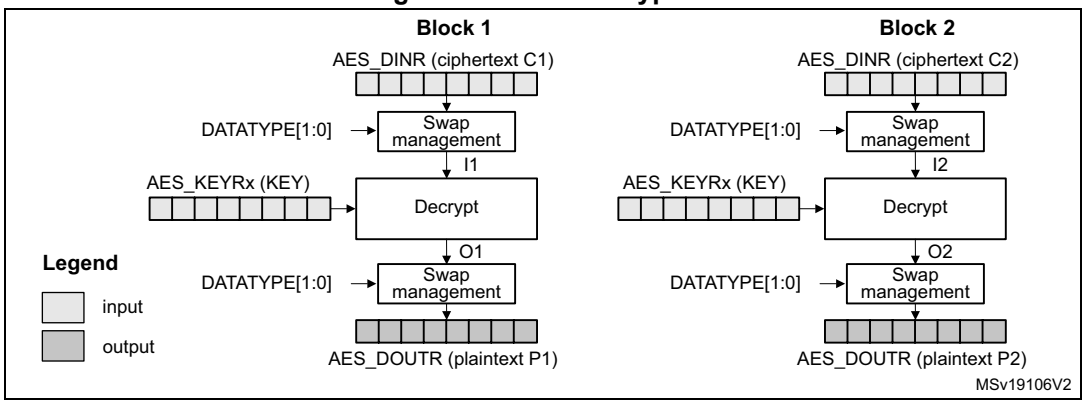
Figure 354. ECB encryption



In ECB encrypt mode, the 128-bit plaintext input data block Px in the SAES_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 SAES_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 355 illustrates the electronic codebook (ECB) decryption.

Figure 355. ECB decryption

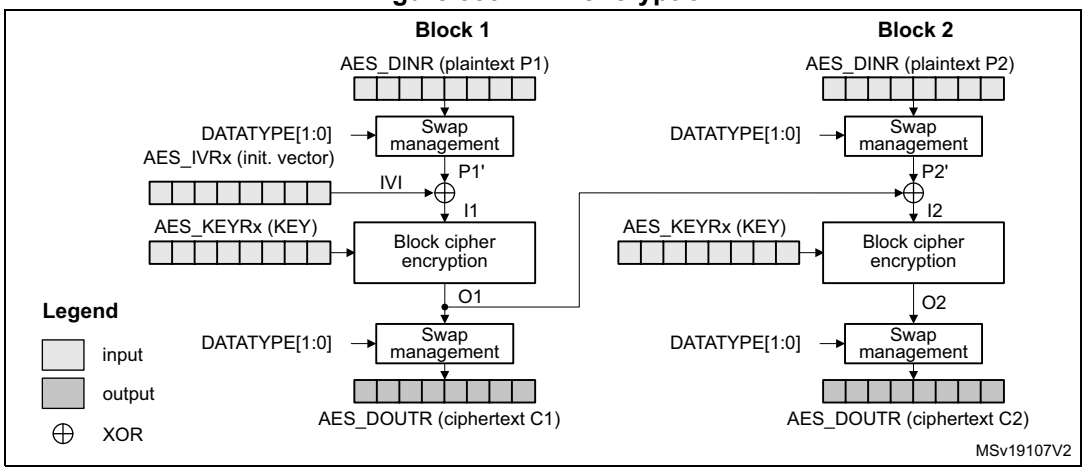


To perform an AES decryption in the ECB mode, the secret key has to be prepared by collecting the last-round encryption key (which requires to first execute the complete key schedule for encryption), and using it as the first-round key for the decryption of the ciphertext. This preparation is supported by the AES core.

In ECB decrypt mode, the 128-bit ciphertext input data block C1 in the SAES_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 SAES_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 356 illustrates the cipher block chaining (CBC) encryption.

Figure 356. CBC encryption

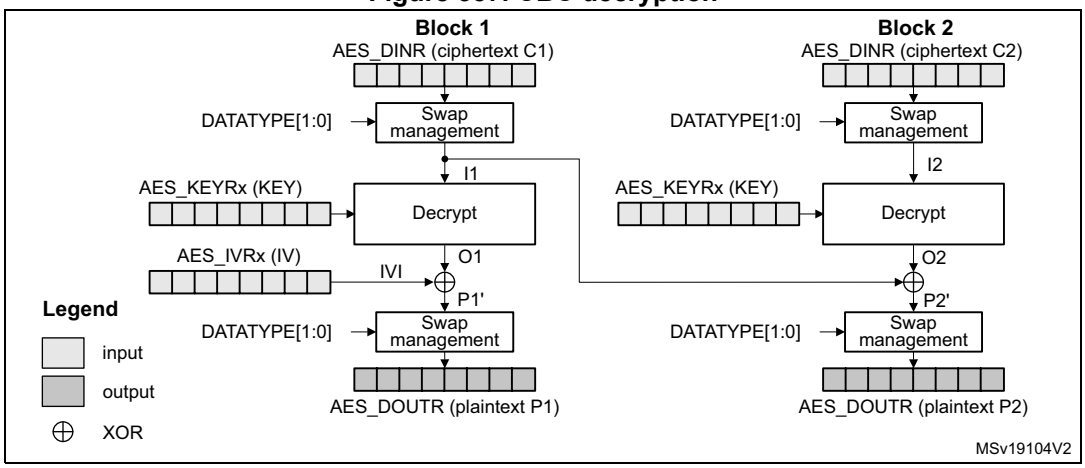


In CBC encrypt mode, the first plaintext input block, after bit/byte/half-word swapping (P1'), is XOR-ed with a 128-bit IVI bitfield (initialization vector and counter), producing the I1 input data for encrypt with the AES core, using a 128- or 256-bit key. The resulting 128-bit output block O1, after 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 42.4.6: SAES ciphertext stealing and data padding*.

Figure 357 illustrates the cipher block chaining (CBC) decryption.

Figure 357. 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 42.4.6: SAES ciphertext stealing and data padding*.

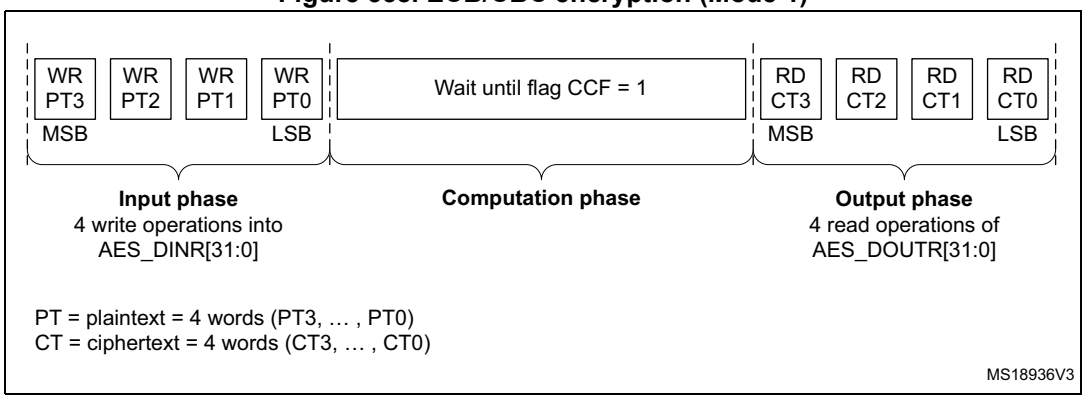
For more information on data swapping, refer to *Section 42.4.11: SAES data registers and data swapping*.

ECB/CBC encryption sequence

The sequence of events to perform an ECB/CBC encryption (more detail in *Section 42.4.4*):

1. Verify that BUSY = 0 in SAES_SR (no RNG random number fetch in progress).
2. Disable the SAES peripheral by clearing the EN bit of the SAES_CR register.
3. Select the Mode 1 by setting to 00 the MODE[1:0] bitfield of the SAES_CR register and select ECB or CBC chaining mode by setting the CHMOD[2:0] bitfield of the SAES_CR register to 000 or 001, respectively. Data type can also be defined, using DATATYPE[1:0] bitfield. Select normal key mode by setting KMOD[1:0] to 00. For encryption with other KMOD[1:0] values, refer to *Section 42.4.10* and *Section 42.4.9*.
4. Select 128- or 256-bit key length through the KEYSIZE bit of the SAES_CR register. If the key must not be shared with a different security context (different secure attribute), the KEYPROT bit of the SAES_CR register must also be set.
5. Write the SAES_KEYRx registers (128 or 256 bits) with encryption key. Alternatively, select a key source different from the key registers, through KEYSEL[2:0]. Refer to *Section 42.4.12: SAES key registers* for details. Fill the SAES_IVRx registers with the initialization vector data if CBC mode has been selected.
6. Enable the SAES peripheral by setting the EN bit of the SAES_CR register.
7. Write the SAES_DINR register four times to input the plaintext (MSB first), as shown in *Figure 358*.
8. Wait until the CCF flag is set in the SAES_SR register.
9. Read the SAES_DOUTR register four times to get the ciphertext (MSB first) as shown in *Figure 358*. Then clear the CCF flag by setting the CCF bit of the SAES_ISR register.
10. Repeat steps *7*-*8*-*9* to process all the blocks with the same encryption key.

Figure 358. ECB/CBC encryption (Mode 1)

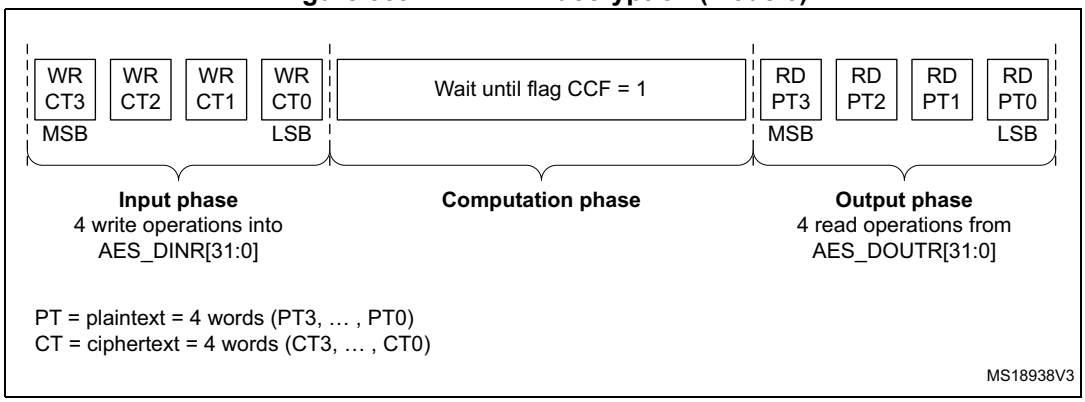


ECB/CBC decryption sequence

The sequence of events to perform an AES ECB/CBC decryption is as follows (More detail in *Section 42.4.4*). Select normal key mode by setting KMOD[1:0] to 00. For decryption with other KMOD[1:0] values, refer to *Section 42.4.10* and *Section 42.4.9*.

1. Follow the steps described in *Section 42.4.5: SAES 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 SAES_CR register and select ECB or CBC chaining mode by setting the CHMOD[2:0] bitfield of the SAES_CR register to 000 or 001, respectively. Data type can also be defined, using DATATYPE[1:0] bitfield. KEYSIZE and KMOD[1:0] bitfields must be kept as-is.
3. Write the SAES_IVRx registers with the initialization vector (required in CBC mode only).
4. Enable SAES by setting the EN bit of the SAES_CR register.
5. Write the SAES_DINR register four times to input the cipher text (MSB first), as shown in *Figure 359*.
6. Wait until the CCF flag is set in the SAES_SR register.
7. Read the SAES_DOUTR register four times to get the plain text (MSB first), as shown in *Figure 359*. Then clear the CCF flag by setting the CCF bit of the SAES_ISR register.
8. Repeat steps *5*-*6*-*7* to process all the blocks encrypted with the same key.

Figure 359. ECB/CBC decryption (Mode 3)



Suspend/resume operations in ECB/CBC modes

The following sequences are valid for normal key mode (KMOD[1:0] = 00).

To suspend the processing of a message, proceed as follows:

1. If DMA is used, stop the SAES DMA transfers to the IN FIFO by clearing the DMAINEN bit of the SAES_CR register.
2. If DMA is not used, read four times the SAES_DOUTR register to save the last processed block. If DMA is used, wait until the CCF flag is set in the SAES_SR register then stop the DMA transfers from the OUT FIFO by clearing the DMAOUTEN bit of the SAES_CR register.
3. If DMA is not used, poll the CCF flag of the SAES_SR register until it becomes 1 (computation completed).
4. Clear the CCF flag by setting the CCF bit of the SAES_ISR register.

5. Save initialization vector registers (only required in CBC mode as SAES_IVRx registers are altered during the data processing).
6. Disable the SAES peripheral by clearing the bit EN of the SAES_CR register.
7. Save the SAES_CR register and clear the key registers if they are not needed, to process the higher priority message.
8. If DMA is used, save the DMA controller status (pointers for IN and OUT data transfers, number of remaining bytes, and so on).

To resume the processing of a message, proceed as follows:

1. If DMA is used, configure the DMA controller so as to complete the rest of the FIFO IN and FIFO OUT transfers.
2. Disable the SAES peripheral by clearing the EN bit of the SAES_CR register.
3. Restore SAES_CR register (with correct KEYSIZE) then restore SAES_KEYRx registers. Alternatively, select a key source different from key registers, using KEYSEL[2:0]. Refer to *Section 42.4.12: SAES key registers* for details.
4. Prepare the decryption key as described in *Section 42.4.5: SAES decryption round key preparation* (only required for ECB or CBC decryption).
5. Restore SAES_IVRx registers using the saved configuration (only required in CBC mode).
6. Enable the SAES peripheral by setting the EN bit of the SAES_CR register.
7. If DMA is used, enable SAES DMA transfers by setting the DMAINEN and DMAOUTEN bits of the SAES_CR register.

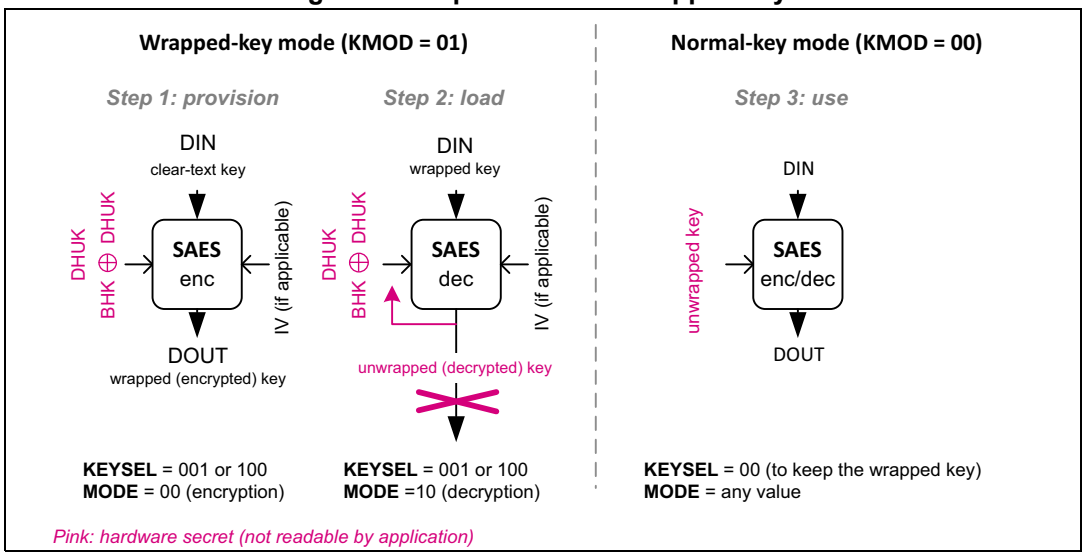
42.4.9 SAES operation with wrapped keys

SAES peripheral can wrap (encrypt) and unwrap (decrypt) application keys using hardware-secret key DHUK, XOR-ed or not with application key BHK. With this feature, AES keys can be made usable by application software without being exposed in clear-text (unencrypted).

Wrapped key sequences are too small to be suspended/resumed. SAES cannot unwrap a key using an unwrapped key.

Figure 360 summarizes the operation with wrapped keys. To protect the wrapped key, select DHUK by setting KEYSEL[2:0] to 001 or 100. Alternatively, select BHK by setting KEYSEL to 010 if the corresponding registers are read/write-locked in the TAMP peripheral.

Figure 360. Operation with wrapped keys



Note: DHUK value depends on privilege, KMOD[1:0], KEYSEL, CHMOD[2:0], and on whether SAES peripheral is secure or non-secure.

Encryption in Wrapped-key mode

Recommended sequence to wrap (that is, encrypt) a key is described below:

1. Verify that BUSY = 0 in SAES_SR (no RNG random number fetch in progress).
2. Disable the SAES peripheral by clearing the EN bit of the SAES_CR register.
3. In SAES_CR register, select the Mode 1 (encryption) by setting to 00 the MODE[1:0] bitfield, and select ECB or CBC chaining mode by setting CHMOD[2:0] bitfield to 000 or 001, respectively. Data type can be defined as 32-bit, with DATATYPE[1:0] bitfield set to 00. Key size must be properly configured using KEYSIZE bit, and KMOD[1:0] bitfield must be set as 01 (wrapped key). The KEYSIZE information is used both for the encryption key and for the key to be encrypted.
4. Write the SAES_IVRx registers with the initialization vector if CBC mode has been selected in previous step.
5. Select the DHUK by setting the KEYSEL[2:0] bitfield of the SAES_CR register to 001 or 100. Upon successful key loading BUSY bit is cleared and KEYVALID bit is set in SAES_SR register. Refer to *Section 42.4.12: SAES key registers* for detail on KEYSEL = 100 usage.
6. Enable SAES by setting the EN bit of the SAES_CR register.
7. Write the SAES_DINR register four times to input the key to encrypt (MSB first, see *Table 373 on page 1531*).
8. Wait until the CCF flag is set in the SAES_SR register.
9. Get the encrypted key (MSB first) by reading the SAES_DOUTR register four times. Then clear the CCF flag by setting the CCF bit in SAES_ICR register.
10. Repeat steps 6 to 8 if KEYSIZE is 256 bits.

Note: Encryption in Wrapped-key mode is only supported when ECB or CBC is selected through CHMOD[2:0].

Decryption in Wrapped-key mode

Recommended sequence to unwrap (i.e. decrypt) a wrapped key is described below:

1. Verify that BUSY = 0 in SAES_SR (no RNG random number fetch in progress).
2. Disable the SAES peripheral by clearing the EN bit of the SAES_CR register.
3. SAES_CR register, select the Mode 2 by setting to 01 the MODE[1:0] bitfield, and select ECB or CBC chaining mode by setting CHMOD[2:0] to 000 or 001, respectively. Key size must be properly configured using KEYSIZE bit, and KMOD[1:0] bitfield must be set as 01 (wrapped key). Data type selection with DATATYPE[1:0] bitfield must be the same as the one used during encryption (that is, 0x0). The KEYSIZE information is used both for the decryption key and for the key to be decrypted.

   If the decrypted key is not to share with a different security context (different security attribute), the KEYPROT bit of the SAES_CR register must also be set.
4. Select the DHUK by setting the KEYSEL[2:0] bitfield of the SAES_CR register to 001 or 100. Upon successful key loading, the SAES_SR register BUSY bit is cleared and KEYVALID bit set. Refer to *Section 42.4.12: SAES key registers* for detail on KEYSEL = 100 usage.
5. Enable SAES by setting the EN bit of the SAES_CR register.
6. Wait until the CCF flag is set in the SAES_SR register.
7. Clear the CCF flag. Decryption key is available in AES core, and SAES is disabled automatically.
8. In SAES_CR register select the Mode 3 by setting to 10 the MODE[1:0] bitfield.
9. Write the SAES_IVRx registers with the initialization vector if CBC mode has been selected in previous step.
10. Enable SAES by setting the EN bit of the SAES_CR register.
11. Write the SAES_DINR register four times to input the encrypted random key (MSB first, see *Table 373 on page 1531*).
12. Wait until the CCF flag is set in the SAES_SR register.
13. Clear the CCF flag, then repeat steps 10 and 11 if KEYSIZE is 256 bits.

When the decrypted key is loaded in key registers, KEYSEL[2:0] of the SAES_CR register is automatically cleared. Hence, after this sequence, the decrypted wrapped key is available in SAES_KEYRx registers, immediately usable by the application software for any AES operation (normal key mode).

Decrypted wrapped key can be shared with an application running in a different security context (different security attribute) if KEYPROT bit was cleared during step 2.

Note: When KMOD[1:0] = 01 (wrapped key) and MODE[1:0] = 10 (decryption) a read access to SAES_DOUTR register triggers a read error (RDERR).

When KEYSEL[2:0] = 001 (DHUK) or 100 (DHUK XOR BHK), the application software must use the same privilege, security, KMOD[1:0], CHMOD[2:0] and KEYSIZE context for encryption and decryption. Otherwise, the result is incorrect.

42.4.10 SAES operation with shared keys

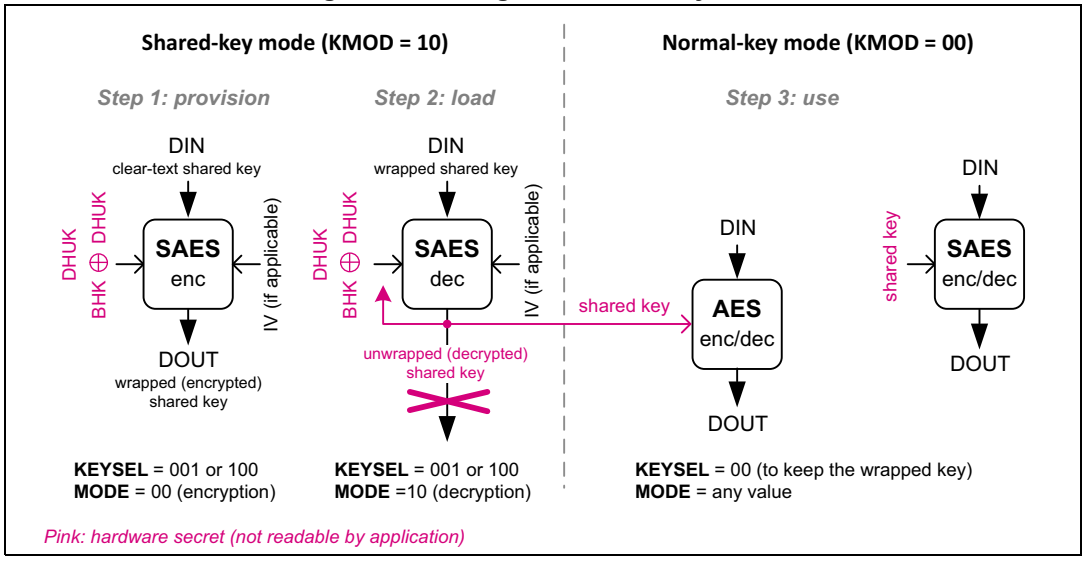
SAES peripheral can share application keys wrapped with hardware-secret key DHUK, XOR-ed or not with application key BHK . With this feature, the application software can make the AES keys available to the AES peripheral, without exposing them in clear-text (unencrypted).

Shared key sequences are too small to be suspended/resumed. SAES cannot unwrap a shared key using an unwrapped key.

Note: When a key stored in SAES is shared with AES, the protection given by KEYPROT bit is lost. The protection is detailed in Section 42.4.12: SAES key registers.

Figure 361 summarizes how to use Shared-key mode. To protect the shared key, DHUK must be selected, by setting KEYSEL[2:0] to 001 or 100. Alternatively, select BHK by setting KEYSEL to 010 if the corresponding registers are read/write-locked in the TAMP peripheral.

Figure 361. Usage of Shared-key mode



Note: DHUK value depends on privilege, KMOD[1:0], KSHAREID, KEYSEL, CHMOD[2:0], and on whether SAES peripheral is secure or non-secure.

In the step 3, AES represents the AES peripheral.

Encryption in Shared-key mode

Before SAES can share a key with the AES peripheral, the key must be encrypted once. The encryption sequence of a shared key is the same as for a wrapped key, with KMOD[1:0] set to 10 (shared key) and KSHAREID[1:0] kept at 00 in the step 3 in *Figure 361*. See *Encryption in Wrapped-key mode* for details.

Note: Encryption in Shared-key mode is only supported when ECB or CBC is selected through CHMOD[2:0].

Decryption in Shared-key mode

Each time SAES needs to share a key with the AES peripheral, shared encrypted key must be decrypted in SAES, then loaded by the AES. The overall sequence is described next.

Sequence in the SAES peripheral

The decryption sequence of a shared key is the same as for a wrapped key, with KMOD[1:0] set to 10 (shared key) and KSHAREID[1:0] kept at 00 in the step 3 in *Figure 361*. See *Decryption in Wrapped-key mode* for details.

Note: Instead of being shared, a decrypted shared key can be used directly in SAES as the KEYSEL[2:0] bitfield is automatically cleared. In this case, KMOD[1:0] must be set to 00 (normal key mode).

Sequence in the AES peripheral

Once the shared key is decrypted in SAES key registers, it can be shared with the AES peripheral, while SAES peripheral remains in key sharing state, that is, with KMOD[1:0] = 10 and KEYVALID = 1 in the SAES_SR register. The sequence in the AES key share target peripheral is as follows:

1. Initialize the AES processor to process some data
2. When the key must be loaded, set the same KEYSIZE as for the SAES peripheral and write the KMOD[1:0] bitfield of the AES_CR register to 10 (shared key). If the previous sequence in the SAES peripheral completed successfully, with KMOD[1:0] kept at 10 and KSHAREID[1:0] kept at 00, the SAES_KEYRx registers are automatically copied into the AES_KEYRx registers, with BUSY set in AES_SR.
3. Once the transfer is completed, the BUSY flag is cleared and KEYVALID set in the AES_SR register. If KEYVALID is not set when BUSY bit is cleared, or if a key error flag (KEIF) is set an unexpected event occurred during the transfer, such as a DPA error, a tamper event or the KEYVALID SAES flag was cleared before the end of the transfer. When such errors occur, the whole key sharing process starting from the SAES peripheral must be restarted, through the IPRST bits of both control registers.
4. As KEYVALID is set, the key share target peripheral is initialized with a valid, shared key. The application can proceed with the data processing of data, setting KMOD[1:0] to 00.

This sequence can be run multiple times (for example, to manage a suspend/resume situation), as long as SAES is unused and duly remains in key sharing state.

Note: in SAES peripheral, w*hen KMOD[1:0] = 10 (shared key) and MODE[1:0] = 10 (decryption), a read access to the SAES_DOUTR register triggers a read error (RDERR).*

When KEYSEL[2:0] = 001 (DHUK) or 100 (DHUK XOR BHK), the application software must use the same privilege, security, KMOD[1:0] / KSHAREID[1:0], CHMOD[2:0], and KEYSIZE context for encryption and decryption. Otherwise, the result is incorrect.

When KMOD[1:0] = 10 and BUSY = 1 in the AES peripheral and KEYSIZE value of AES and SAES differs, the key sharing fails and the KEIF flag is raised in both peripherals.

42.4.11 SAES data registers and data swapping

Data input and output

A 128-bit data block is entered into the SAES peripheral with four successive 32-bit word writes into the SAES_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 SAES peripheral with four successive 32-bit word reads from the SAES_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 SAES_DINR register or from SAES_DOUTR register is organized in big endian order, that is:

- the most significant byte of a word to write into SAES_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 SAES_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 SAES, 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 42.4.14: SAES DMA interface*.

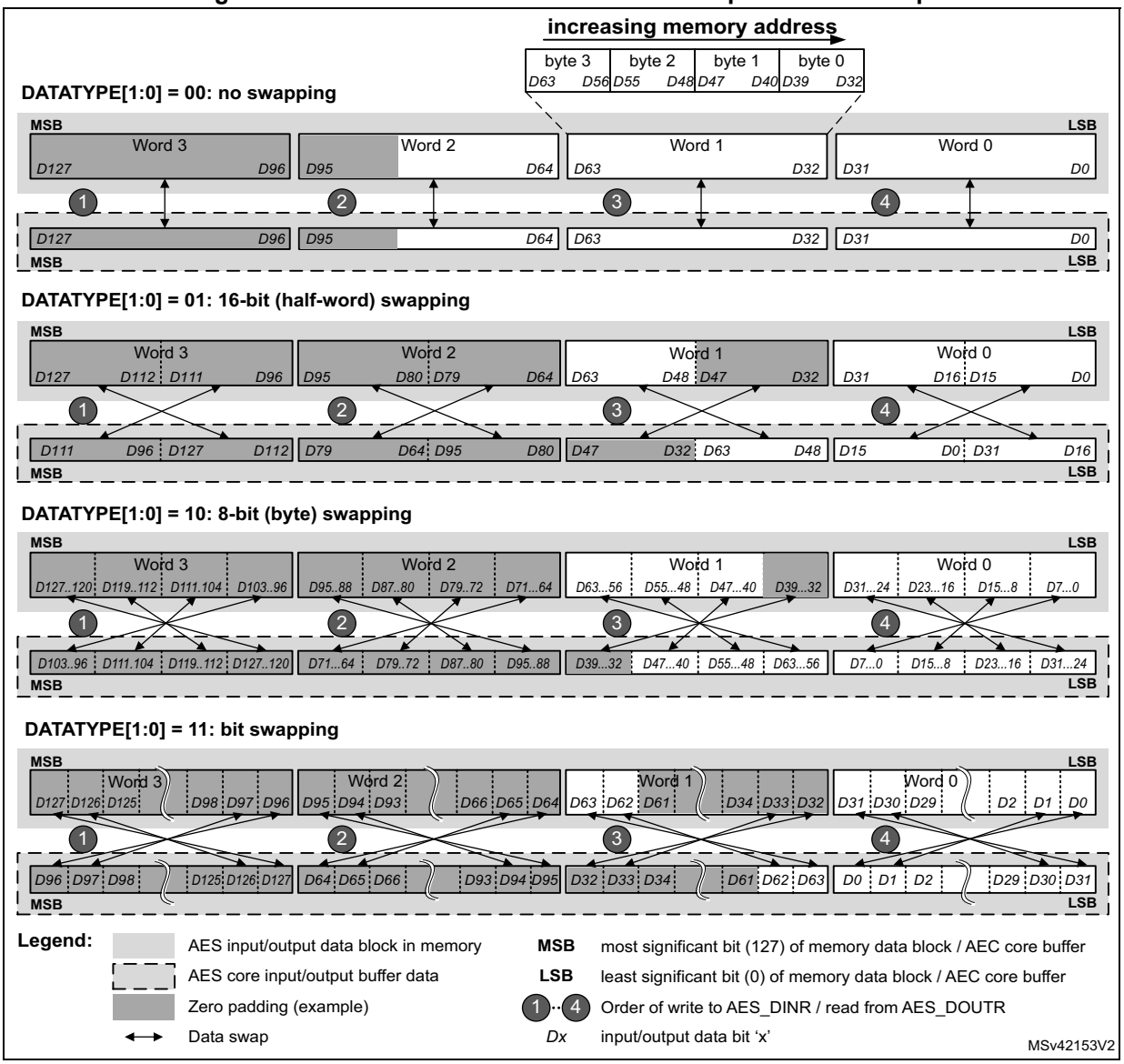
Data swapping

The SAES peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the SAES_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 SAES_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 SAES_CR register. The selection applies both to the input and the output of the AES core.

For different data swap types, *Figure 362* shows the construction of AES processing core input buffer data P127 to P0, from the input data entered through the SAES_DINR register, or the construction of the output data available through the SAES_DOUTR register, from the AES processing core output buffer data P127 to P0.

Figure 362. 128-bit block construction with respect to data swap



Note: The data in SAES key registers (SAES_KEYRx) and initialization registers (SAES_IVRx) are not sensitive to the swap mode selection.

Data padding

Figure 362 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

42.4.12 SAES key registers

The SAES_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 373*.

Table 373. Key endianness in SAES_KEYRx registers (128- or 256-bit key length)

SAES_KEYR 7 [31:0]	SAES_KEYR 6 [31:0]	SAES_KEYR 5 [31:0]	SAES_KEYR 4 [31:0]	SAES_KEYR 3 [31:0]	SAES_KEYR 2 [31:0]	SAES_KEYR 1 [31:0]	SAES_KEYR 0 [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]
TAMP_BKP7R [31:0]	TAMP_BKP6R [31:0]	TAMP_BKP5R [31:0]	TAMP_BKP4R [31:0]	TAMP_BKP3R [31:0]	TAMP_BKP2R [31:0]	TAMP_BKP1R [31:0]	TAMP_BKP0R [31:0]

The key for encryption or decryption may be written into these registers when the SAES peripheral is disabled, by clearing the EN bit and the KEYSEL[2:0] bitfield of the SAES_CR register.

The key registers are not affected by the data swapping controlled by DATATYPE[1:0] bitfield of the SAES_CR register.

The entire key must be written before starting an AES computation. In normal key mode (KMOD[1:0] = 00), with KEYSEL[2:0] = 000, the SAES_KEYRx (x = 0 to 3 for KEYSIZE = 0, x = 0 to 7 for KEYSIZE = 1) registers must always be written in either ascending or descending order.

With KEYSEL[2:0] set to 001, a derived hardware unique key (DHUK), computed inside the SAES engine from a non-volatile OTP-based root hardware unique key, is loaded directly into key registers, based on KEYSIZE information.

With KEYSEL[2:0] set to 010, the boot hardware key (BHK), stored in tamper-resistant secure backup registers, is entirely transferred into key registers upon a secure application performing a single read of all TAMP_BKPxR registers (x = 0 to 3 for KEYSIZE = 0, x = 0 to 7 for KEYSIZE = 1) in either ascending or descending order. Refer to *Table 373*.

With KEYSEL[2:0] set to 100, the XOR combination of DHUK and BHK is entirely transferred into key registers upon a secure application performing a single read of all TAMP_BKPxR registers (x = 0 to 3 for KEYSIZE = 0, x = 0 to 7 for KEYSIZE = 1) in either ascending or descending order. Refer to *Table 373*.

Repeated writing of KEYSEL[2:0] with the same non-zero value only triggers the loading of DHUK or BHK if KEYVALID = 0. The recommended method to clear KEYVALID is to set the IPRST bit in the SAES_CR register. Such method is required for example when switching from ECB decryption to ECB encryption, selecting the same BHK (KEYSEL[2:0] = 010).

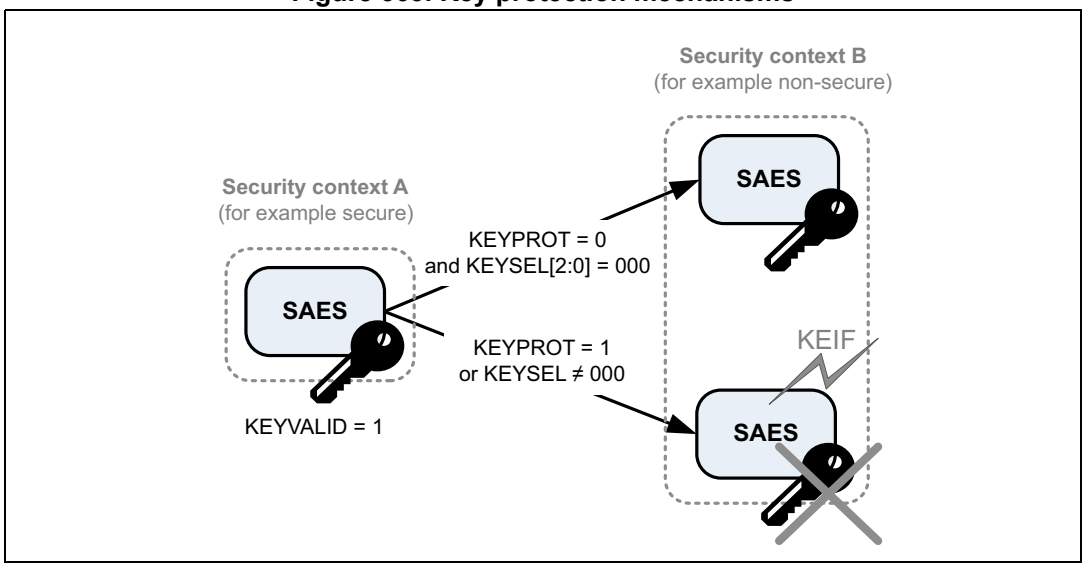
The KEYPROT bit of SAES_CR register must be set if the key to load in key registers must not be shared with an application executing in a different security context (that is, different security attribute). Setting KEYPROT and KEYVALID makes KEIF flag an error upon SAES access attempts by an application executing in a different security context than the one that loaded the key, as shown in *Figure 363*.

Note: KEYSEL[2:0] values different from zero (normal key) automatically protect the key registers.

DHUK, BHK and their XOR combination are not readable by any software (even secure).

Secure SAES uses secure DHUK. Non-secure SAES uses non-secure DHUK.

Figure 363. Key protection mechanisms



Note: Initiating the key-loading sequence sets the BUSY flag and clears the KEYVALID flag. Once the amount of bits defined by KEYSIZE is transfered to the SAES_KEYRx registers, BUSY is cleared, KEYVALID set and the EN bit becomes writable. If an error occurs, BUSY and KEYVALID are cleared and KEIF set (see Section 42.4.15: SAES error management for details).This holds for all KEYSEL values.
For additional information on key modes, refer to Section 42.4.10 and Section 42.4.9.

42.4.13 SAES initialization vector registers

The four SAES_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 (SAES_IVR0) to highest address (SAES_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 SAES_IVRx registers when the SAES peripheral is enabled have no effect to the register contents. For modifying the contents of the SAES_IVRx registers, the EN bit of the SAES_CR register must first be cleared.

Reading the SAES_IVRx registers returns the latest counter value (useful for managing suspend mode).

The SAES_IVRx registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield of the SAES_CR register.

42.4.14 SAES DMA interface

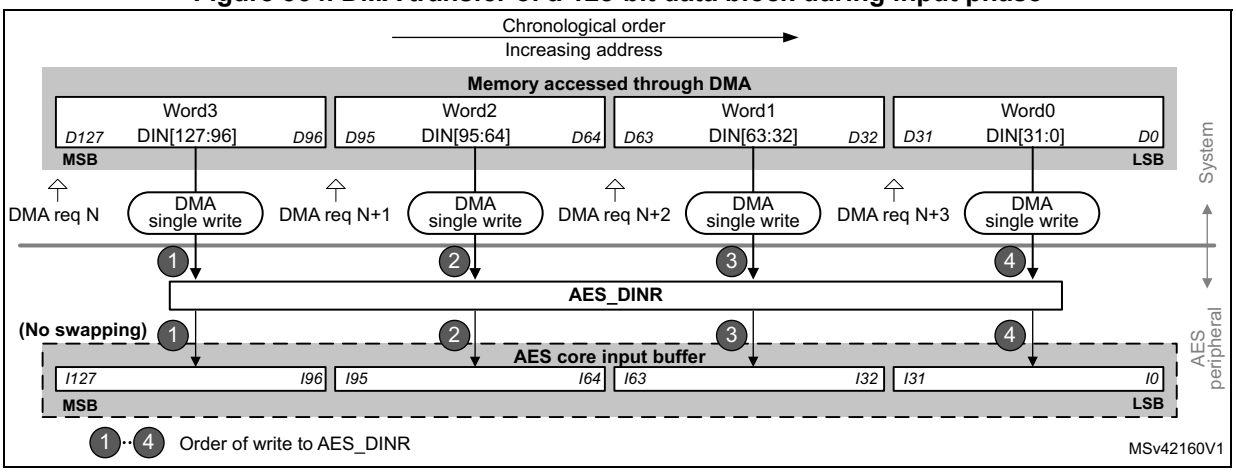
The SAES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the SAES_CR register.

Data input using DMA

Setting the DMAINEN bit of the SAES_CR register enables DMA writing into SAES. The SAES peripheral then initiates a DMA request during the input phase each time it requires to write a 128-bit block (quadruple word) to the SAES_DINR register, as shown in *Figure 364*.

Note: According to the algorithm and the mode selected, special padding / ciphertext stealing might be required. For details, refer to Section 42.4.6: SAES ciphertext stealing and data padding.

Figure 364. DMA transfer of a 128-bit data block during input phase

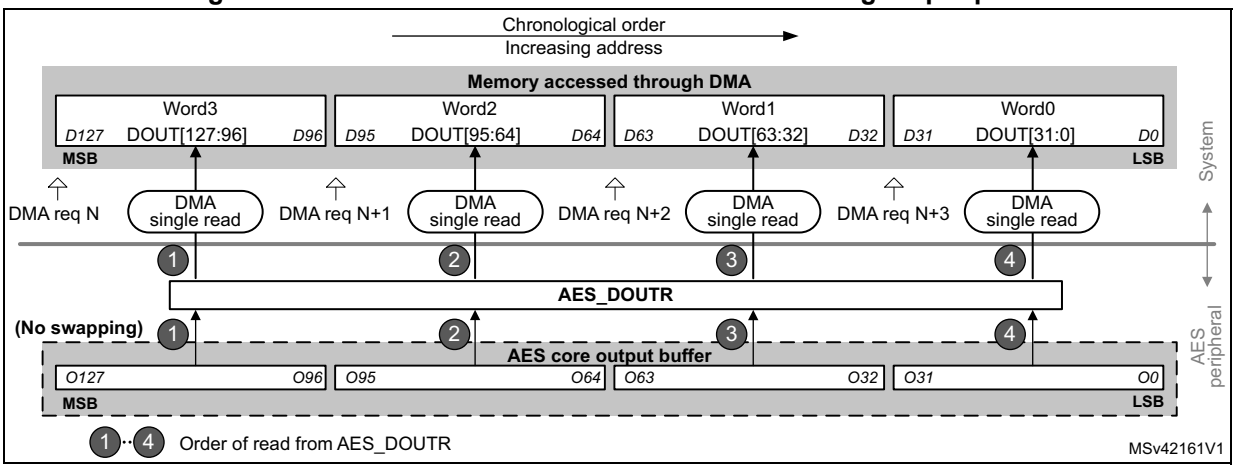


Data output using DMA

Setting the DMAOUTEN bit of the SAES_CR register enables DMA reading from SAES. The SAES peripheral then initiates a DMA request during the Output phase each time it requires to read a 128-bit block (quadruple word) to the SAES_DINR register, as shown in *Figure 365*.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Figure 365. 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 SAES_CR. As in Mode 2 (key derivation) the SAES_KEYRx registers must be written by software, enabling the DMA transfer through the DMAINEN and DMAOUTEN bits of the SAES_CR register have no effect in that mode.

DMA single requests are generated by SAES until it is disabled. So, after the data output phase at the end of processing of a 128-bit data block, SAES switches automatically to a new data input phase for the next data block, if any.

When the data transferring between SAES and memory is managed by DMA, the CCF flag has no use because the reading of the SAES_DOUTR register is managed by DMA automatically at the end of the computation phase. The CCF flag must only be cleared when transiting back to data transferring managed by software. See *Section 42.4.4: SAES procedure to perform a cipher operation*, subsection *Data append*, for details.

42.4.15 SAES error management

Unless indicated otherwise, SAES configuration can be changed at any moment by clearing the EN bit of the SAES_CR register.

Read error flag (RDERR)

Unexpected read attempt of the SAES_DOUTR register sets the RDERR flag of the SAES_SR register and the RWEIF flag of the SAES_ISR register, and returns zero.

RDERR is triggered during the computation phase or during the input phase.

Note: Unless indicated otherwise, SAES is not disabled upon a RDERR error detection and continues processing.

An interrupt is generated if the RWEIE bit of the SAES_IER register is set. For more details, refer to *Section 42.5: SAES interrupts*.

The RDERR and RWEIF flag is cleared by setting the RWEIE bit of the SAES_ISR register.

Write error flag (WDERR)

Unexpected write attempt of the SAES_DINR register sets the WRERR flag of the SAES_SR register and the RWEIF flag of the SAES_ISR register, and has no effect on the

SAES_DINR register. The WRERR is triggered during the computation phase or during the output phase.

Note: Unless indicated otherwise, SAES is not disabled after a WRERR error detection and continues processing.

An interrupt is generated if the RWEIE bit of the SAES_IER register is set. For more details, refer to *Section 42.5: SAES interrupts*.

The WRERR and RWEIF flag is cleared by setting the RWEIF bit of the SAES_ISR register.

Key error interrupt flag (KEIF)

Failure to load a key into key registers, or attempt to load a key while the key is protected, sets the KEIF flag of the SAES_ISR register and clears the KEYVALID bit of the SAES_SR register.

The KEIF flag is cleared with corresponding bit of the SAES_ISR register. An interrupt is generated if the KEIE bit of the SAES_IER register is set. For more details, refer to *Section 42.5: SAES interrupts*.

The possible sources of key errors are:

- Key protection error: while KEYVALID is set, if KEYPROT = 1 or KEYSEL is different from zero this error is triggered when an application executing in a security context different from the one used to load the key is detected accessing the SAES (that is, different security attribute).
- Key writing sequence error: an incorrect sequence has been detected when writing key registers. See *Section 42.4.12: SAES key registers* for details.
- Key sharing size mismatch: error is triggered when KMOD[1:0] = 10 and application sets a KEYSIZE information in AES peripheral that does not match the KEYSIZE stored in SAES peripheral
- Key sharing error: the copy of key registers from SAES peripheral to AES failed. See *Section 42.4.10: SAES operation with shared keys* for details.
- Hardware secret key loading error: the DHUK or BHK failed to load into SAES. KEYSEL = 001 (DHUK), 010 (BHK) or 100 (DHUK XOR BHK) is not functional.

Upon a key sharing error, reset both AES and SAES peripherals through the IPRST bit of their corresponding control register, then restart the key sharing sequence.

Upon a key selection error, clearing the KEIF flag automatically restarts the key selection process. Persisting problems (for example, RHUK load failing) may require a power-on reset.

Note: For any key error, clear KEIF flag prior to re-configuring SAES.

RNG error interrupt flag (RNGEIF)

SAES fetches random numbers from the RNG peripheral automatically after an IP reset triggered in the RCC. SAES cannot be used when RNGEIF is set.

An error detected while fetching a random number from RNG peripheral (due to, for example, bad entropy) sets the RNGEIF flag of the SAES_ISR register. The flag is cleared by setting the corresponding bit of the SAES_ICR register. An interrupt is generated if the

RNGEIE bit of the SAES_IER register is set. For more details, refer to *Section 42.5: SAES interrupts*.

Upon an RNG error:

- Verify that the RNG peripheral AHB clock is enabled and no noise source (or seed) error is pending in this peripheral.
- Clear the RNGEIF bit of the SAES_ISR register, or reset the peripheral by setting the IPRST bit of the SAES_CR register. The clearance of the BUSY flag in the SAES_SR register then indicates the completion of the random number fetch from RNG.

Note: To avoid RNGEIF errors, it is recommended to activate the RNG AHB clock each time SAES AHB clock is activated.

About DPA errors

An unexpected error triggers an SAES internal tamper event in the TAMP peripheral, and stops any SAES co-processor processing.

To resume normal operation, application must reset the SAES peripheral through RCC or global reset.

42.5 SAES interrupts

Individual maskable interrupt sources generated by the SAES peripheral signal the following events:

- computation completed
- read error
- write error
- key error
- RNG error

The individual sources are combined into the common interrupt signal saes_it that connects to NVIC (nested vectored interrupt controller). Each can individually be enabled/disabled, by setting/clearing the corresponding enable bit of the SAES_IER register, and cleared by setting the corresponding bit of the SAES_ISR register.

The status of each can be read from the SAES_SR and SAES_ISR registers.

Table 374 gives a summary of the interrupt sources, their event flags and enable bits.

Table 374. SAES interrupt requests

Interrupt acronym	SAES interrupt event	Event flag	Enable bit	Interrupt clear method
SAES	computation completed flag	CCF	CCFIE	set CCF(1)
	read error flag	RDERR(2)	RWEIE	set RWEIF(1)
	write error flag	WRERR(2)		
	key error flag	KEIF	KEIE	set KEIF(1)
	RNG error flag	RNGEIF	RNGEIE	set RNGEIF(1)

1. Bit of the SAES_ISR register.
2. Flag of the SAES_SR register, mirrored by the flag RWEIF of the SAES_ISR register.

42.6 SAES processing latency

The tables below summarize the latency to process a 128-bit block for each mode of operation.

Table 375. Processing latency for ECB, CBC

Key size	Mode of operation	Algorithm	Clock cycles(1)
128-bit	Mode 1: Encryption	ECB, CBC	**528**
	Mode 2: Key derivation	-	**200**
	Mode 3: Decryption	ECB, CBC	**528**
256-bit	Mode 1: Encryption	ECB, CBC	**743**
	Mode 2: Key derivation	-	**324**
	Mode 3: Decryption	ECB, CBC	**743**

1. SHSI clock from RCC (48 MHz with +/-15% jitter)

42.7 SAES registers

42.7.1 SAES control register (SAES_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPRST	KEYSEL[2:0]			KSHAREID[1:0]		KMOD[1:0]		Res.				KEYPROT	KEYSIZE	Res.	CHMOD[2]
rw	rw	rw	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.	DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
			rw	rw					rw	rw	rw	rw	rw	rw	rw

Bit 31 **IPRST**: SAES peripheral software reset

Setting the bit resets the SAES peripheral, putting all registers to their default values, except the IPRST bit itself and the SAES_DPACFG register. Hence, any key-relative data is lost. For this reason, it is recommended to set the bit before handing over the SAES to a less secure application. The bit must be low while writing any configuration registers.

Bits 30:28 **KEYSEL[2:0]**: Key selection

The bitfield defines the source of the key information to use in the AES cryptographic core.

000: Software key, loaded in key registers SAES_KEYx

001: Derived hardware unique key (DHUK)

010: Boot hardware key (BHK)

100: XOR of DHUK and BHK

111: Test mode key (256-bit hardware constant 0xA5A5...A5A5)

Others: Reserved (if used, unfreeze SAES with IPRST)

When KEYSEL is different from zero, selected key value is available in key registers when BUSY bit is cleared and KEYVALID is set in the SAES_SR register. Otherwise, the key error flag KEIF is set. Repeated writing of KEYSEL[2:0] with the same non-zero value only triggers the loading of DHUK or BHK if KEYVALID = 0.

When the application software changes the key selection by writing the KEYSEL[2:0] bitfield, the key registers are immediately erased and the KEYVALID flag cleared.

At the end of the decryption process, if KMOD[1:0] is other than zero, KEYSEL[2:0] is cleared.

With the bitfield value other than zero and KEYVALID set, the application cannot transfer the ownership of SAES with a loaded key to an application running in another security context (such as secure, non-secure). More specifically, when security of an access to any register does not match the information recorded by SAES, the KEIF flag is set.

Attempts to write the bitfield are ignored when the BUSY flag of SAES_SR register is set, as well as when the EN bit of the SAES_CR register is set before the write access and it is not cleared by that write access.

Bits 27:26 **KSHAREID[1:0]**: Key share identification

This bitfield defines, at the end of a decryption process with KMOD[1:0] = 10 (shared key), which target can read the SAES key registers using a dedicated hardware bus.

00: AES peripheral

Others: Reserved

Attempts to write the bitfield are ignored when the BUSY flag of SAES_SR register is set, as well as when the EN bit of the SAES_CR register is set before the write access and it is not cleared by that write access.

Bits 25:24 **KMOD[1:0]**: Key mode selection

The bitfield defines how the SAES key can be used by the application:

00: Normal key

01: Wrapped key

10: Shared key

Others: Reserved

With normal key selection, the key registers are freely usable, no specific usage or protection applies to SAES_DIN and SAES_DOUT registers.

With wrapped key selection, the key loaded in key registers can only be used to encrypt or decrypt AES keys. Hence, when a decryption is selected in Wrapped-key mode read-as-zero SAES_DOUT register is automatically loaded into SAES key registers after a successful decryption process.

With shared key selection, after a successful decryption process, SAES key registers are shared with the peripheral described in KSHAREID(1:0] bitfield. This sharing is valid only while KMOD[1:0] = 10 and KEYVALID = 1. When a decryption is selected, read-as-zero SAES_DOUT register is automatically loaded into SAES key registers after a successful decryption process.

With KMOD[1:0] other than zero, any attempt to configure the SAES peripheral for use by an application belonging to a different security domain (secure or non-secure) results in automatic key erasure and setting of the KEIF flag.

Attempts to write the bitfield are ignored when the BUSY flag of SAES_SR register is set, as well as when the EN bit of the SAES_CR register is set before the write access and it is not cleared by that write access.

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **KEYPROT**: Key protection

When set, hardware-based key protection is enabled.

0: When KEYVALID is set and KEYSEL = 0 application can transfer the ownership of the SAES, with its loaded key, to an application running in another security context (such as non-secure, secure).

1: When KEYVALID is set, key error flag (KEIF) is set when an access to any registers is detected, this access having a security context (for example, secure, non-secure) that does not match the one of the application that loaded the key.

Attempts to write the bit are ignored when the BUSY flag of SAES_SR register is set, as well as when the EN bit of the SAES_CR register is set before the write access and it is not cleared by that write access.

Bit 18 **KEYSIZE:** Key size selection

This bitfield defines the length of the key used in the SAES cryptographic core, in bits:

0: 128

1: 256

When KMOD[1:0] = 01 or 10 KEYSIZE also defines the length of the key to encrypt or decrypt.

Attempts to write the bit are ignored when the BUSY flag of SAES_SR register is set, as well as when the EN bit of the SAES_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 16 Reserved, must be kept at reset value.

Bits 15:13 Reserved, must be kept at reset value.

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 SAES 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 SAES 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).

Bits 10:7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

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)

others: Reserved

Attempts to write the bitfield are ignored when the BUSY flag of SAES_SR register is set, as well as when the EN bit of the SAES_CR register is set before the write access and it is not cleared by that write access.

Bits 4:3 **MODE[1:0]**: SAES operating mode

This bitfield selects the SAES 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 BUSY flag of SAES_SR register is set, as well as when the EN bit of the SAES_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 SAES_DINR register or read from the SAES_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 42.4.11: SAES data registers and data swapping*.

Attempts to write the bitfield are ignored when the BUSY flag of SAES_SR register is set, as well as when the EN bit of the SAES_CR register is set before the write access and it is not cleared by that write access.

Bit 0 **EN**: SAES enable

This bit enables/disables the SAES peripheral:

0: Disable

1: Enable

At any moment, clearing then setting the bit re-initializes the SAES peripheral.

This bit is automatically cleared by hardware upon the completion of the key preparation (Mode 2) .

The bit cannot be set as long as KEYVALID = 0.

Note: With KMOD[1:0] other than 00, use the IPRST bit rather than the bit EN.

42.7.2 SAES status register (SAES_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.	KEYVALID	Res.	Res.	Res.	BUSY	WRERR	RDERR	CCF
								r				r	r	r	r

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

Bit 7 **KEYVALID**: Key Valid flag

This bit is set by hardware when the amount of key information defined by KEYSIZE in SAES_CR has been loaded in SAES_KEYx key registers.

0: No valid key information is available in key registers. EN bit in SAES_CR cannot be set.

1: Valid key information, defined by KEYSIZE in SAES_CR, is loaded in key registers.

In normal mode when KEYSEL equals to zero, the application must write the key registers in the correct sequence, otherwise the KEIF flag of the SAES_ISR register is set and KEYVALID stays at zero.

When KEYSEL is different from zero the BUSY flag is automatically set by SAES. When key is loaded successfully, the BUSY flag is cleared and KEYVALID set. Upon an error, the KEIF flag of the SAES_ISR register is set, the BUSY flag cleared and KEYVALID kept at zero.

When the KEIF flag is set, the application must clear it through the SAES_ICR register, otherwise KEYVALID cannot be set. See the KEIF bit description for more details.

For more information on key loading, refer to *Section 42.4.12: SAES key registers*.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy

This flag indicates whether SAES is idle or busy during GCM payload **encryption** phase:

0: Idle

1: Busy

The flag is also set upon SAES initialization, upon fetching random number from the RNG, or upon transferring a shared key to a target peripheral.

Bit 2 **WRERR**: Write error

This flag indicates the detection of an unexpected write operation to the SAES_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 RWEIF bit of the SAES_ISR register.

Upon the flag setting, an interrupt is generated if enabled through the RWEIE bit of the SAES_ISR register.

The flag setting has no impact on the SAES operation. Unexpected write is ignored.

Bit 1 **RDERR**: Read error flag

This flag indicates the detection of an unexpected read operation from the SAES_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 RWEIF bit of the SAES_ISR register.

Upon the flag setting, an interrupt is generated if enabled through the RWEIE bit of the SAES_ISR register.

The flag setting has no impact on the SAES operation. Unexpected read returns zero.

Bit 0 **CCF**: Computation completed flag

This bit mirrors the CCF bit of the SAES_ISR register.

42.7.3 SAES data input register (SAES_DINR)

Address offset: 0x08

Reset value: 0x0000 0000

Only 32-bit write access type is supported.

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

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 SAES 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 SAES operating mode:

- **Mode 1** (encryption): plaintext
- **Mode 2** (key derivation): the bitfield is not used (SAES_KEYRx registers used for input if KEYSEL = 0)
- **Mode 3** (decryption): ciphertext

The data swap operation is described in *Section 42.4.11: SAES data registers and data swapping on page 1529*.

42.7.4 SAES data output register (SAES_DOUTR)

Address offset: 0x0C

Reset value: 0x0000 0000

Only 32-bit read access type is supported. Read when KMOD[1:0] = 01 or 10 while MODE[1:0] = 10 and EN = 1 triggers a read error.

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 SAES 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 SAES 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 42.4.11: SAES data registers and data swapping on page 1529*.

42.7.5 SAES key register 0 (SAES_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 SAES_KEYRx registers may be written only when KEYSIZE value is correct and when the SAES peripheral is disabled (EN bit of the SAES_CR register cleared). A special writing sequence is also required, as described in KEYVALID bit of the SAES_SR register. Note that, if KEYSEL is different from 0 and KEYVALID = 0, the key is directly loaded to SAES_KEYRx registers (hence writes to key register is ignored and KEIF is set).

Refer to *Section 42.4.12: SAES key registers on page 1531* for more details.

42.7.6 SAES key register 1 (SAES_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 SAES_KEYR0 register for description of the KEY[255:0] bitfield.

42.7.7 SAES key register 2 (SAES_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 SAES_KEYR0 register for description of the KEY[255:0] bitfield.

42.7.8 SAES key register 3 (SAES_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 SAES_KEYR0 register for description of the KEY[255:0] bitfield.

42.7.9 SAES initialization vector register 0 (SAES_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 42.4.13: SAES initialization vector registers on page 1532* for description of the IVI[127:0] bitfield.
The initialization vector is only used in chaining modes other than ECB.
The SAES_IVRx registers may be written only when the SAES peripheral is disabled

42.7.10 SAES initialization vector register 1 (SAES_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 SAES_IVR0 register for description of the IVI[128:0] bitfield.

42.7.11 SAES initialization vector register 2 (SAES_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 SAES_IVR0 register for description of the IVI[128:0] bitfield.

42.7.12 SAES initialization vector register 3 (SAES_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 SAES_IVR0 register for description of the IVI[128:0] bitfield.

42.7.13 SAES key register 4 (SAES_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 SAES_KEYR0 register for description of the KEY[255:0] bitfield.

42.7.14 SAES key register 5 (SAES_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 SAES_KEYR0 register for description of the KEY[255:0] bitfield.

42.7.15 SAES key register 6 (SAES_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 SAES_KEYR0 register for description of the KEY[255:0] bitfield.

42.7.16 SAES key register 7 (SAES_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 SAES_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).

42.7.17 SAES interrupt enable register (SAES_IER)

Address offset: 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	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.	RNGEIE	KEIE	RWEIE	CCFIE
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RNGEIE:** RNG error interrupt enable
This bit enables or disables (masks) the SAES interrupt generation when RNGEIF (RNG error flag) is set.
0: Disabled (masked)
1: Enabled (not masked)

Bit 2 **KEIE:** Key error interrupt enable
This bit enables or disables (masks) the SAES interrupt generation when KEIF (key error flag) is set.
0: Disabled (masked)
1: Enabled (not masked)

Bit 1 **RWEIE:** Read or write error interrupt enable
This bit enables or disables (masks) the SAES interrupt generation when RWEIF (read and/or write error flag) is set.
0: Disabled (masked)
1: Enabled (not masked)

Bit 0 **CCFIE:** Computation complete flag interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when CCF (computation complete flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

42.7.18 SAES interrupt status register (SAES_ISR)

Address offset: 0x304

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	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.	RNGEIF	KEIF	RWEIF	CCF
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RNGEIF:** RNG error interrupt flag

This read-only bit is set by hardware when an error is detected on RNG bus interface (e.g. bad entropy).

0: RNG bus is functional

1: Error detected on RNG bus interface (random seed fetching error)

RNGEIE bit is cleared when application sets the corresponding bit of SAES_ICR register. An interrupt is generated if the RNGEIE bit has been previously set in the SAES_IER register. Clearing this bit triggers the reload of a new random number from RNG peripheral.

Bit 2 **KEIF:** Key error interrupt flag

This read-only bit is set by hardware when key information failed to load into key registers or key register usage is forbidden.

0: No key error detected

1: Key information failed to load into key registers, or key register use is forbidden

Setting the corresponding bit of the SAES_ICR register clears the KEIF and generates interrupt if the KEIE bit of the SAES_IER register is set.

KEIF is triggered upon any of the following errors:

–SAES fails to load the DHUK (KEYSEL = 001 or 100).

–SAES fails to load the BHK (KEYSEL = 010 or 100) respecting the correct order.

–AES fails to load the key shared by SAES peripheral (KMOD = 10).

– When KEYVALID = 1 and (KEYPROT = 1 or KEYSEL is not 0x0), the security context of the application that loads the key (secure or non-secure) does not match the security attribute of the access to SAES_CR or SAES_DOUT. In this case, KEYVALID and EN bits are cleared.

–SAES_KEYRx register write does not respect the correct order. (For KEYSIZE = 0, SAES_KEYR0 then SAES_KEYR1 then SAES_KEYR2 then SAES_KEYR3 register, or reverse. For KEYSIZE = 1, SAES_KEYR0 then SAES_KEYR1 then SAES_KEYR2 then SAES_KEYR3 then SAES_KEYR4 then SAES_KEYR5 then SAES_KEYR6 then SAES_KEYR7, or reverse).

KEIF must be cleared by the application software, otherwise KEYVALID cannot be set.

Bit 1 **RWEIF:** Read or write error interrupt flag

This read-only bit is set by hardware when a RDERR or a WRERR error flag is set in the SAES_SR register.

0: No read or write error detected

1: Read or write error detected (see SAES_SR register for details)

RWEIF bit is cleared when application sets the corresponding bit of SAES_ICR register. An interrupt is generated if the RWEIE bit has been previously set in the SAES_IER register.

This flags has no meaning when key derivation mode is selected.

Bit 0 **CCF:** Computation complete 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 CCF bit of the SAES_ISR register.

Upon the flag setting, an interrupt is generated if enabled through the CCFIE bit of the SAES_IER register.

The flag is significant only when the DMAOUTEN bit is 0. It may stay high when DMA_EN is 1.

42.7.19 SAES interrupt clear register (SAES_ICR)

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RNGEIF:** RNG error interrupt flag clear

Application must set this bit to clear the RNGEIF status bit in SAES_ISR register.

Bit 2 **KEIF:** Key error interrupt flag clear

Setting this bit clears the KEIF status bit of the SAES_ISR register.

Bit 1 **RWEIF:** Read or write error interrupt flag clear

Setting this bit clears the RWEIF status bit of the SAES_ISR register, and both RDERR and WRERR flags in the SAES_SR register.

Bit 0 **CCF:** Computation complete flag clear

Setting this bit clears the CCF status bit of the SAES_SR and SAES_ISR registers.

42.7.20 SAES register map

Table 376. SAES 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	SAES_CR	IPRST	KEYSEL[2:0]			KSHAREID[1:0]		KMOD[1:0]		Res.				KEYPROT	KEYSIZE	Res.	CHMOD[2]	Res.	Res.		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	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
0x004	SAES_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.	KEYVALID	Res.	Res.	Res.	BUSY	WRERR	RDERR	CCF
	Reset value																									0				0	0	0	0
0x008	SAES_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	SAES_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	SAES_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	SAES_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	SAES_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	SAES_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	SAES_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	SAES_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	SAES_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	SAES_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	SAES_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	SAES_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	SAES_KEYR6	KEY[223:192]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	SAES_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
0x040-0x2FF	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 376. SAES 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
0x300	SAES_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.	RNGEIE	KEIE	RWEIE	CCFIE
	Reset value																													0	0	0	0
0x304	SAES_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.	RNGEIF	KEIF	RWEIF	CCF
	Reset value																													0	0	0	0
0x308	SAES_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.	RNGEIF	KEIF	RWEIF	CCF
	Reset value																													0	0	0	0

Refer to *Section 2.3 on page 115* for the register boundary addresses.

43 Hash processor (HASH)

43.1 Introduction

The hash processor is a fully compliant implementation of the secure hash algorithm (SHA-1, SHA2-224, SHA2-256), 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.

43.2 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, SHA2-224, SHA2-256, and MD5
  - 82 (respectively 66) clock cycles for processing one 512-bit block of data using SHA-1 (respectively SHA2-256) 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.
- Single or fixed DMA burst transfers of four words

- Interruptible message digest computation, on a per-block basis
  - Re-loadable digest registers
  - Hashing computation suspend/resume mechanism, including DMA

43.3 HASH implementation

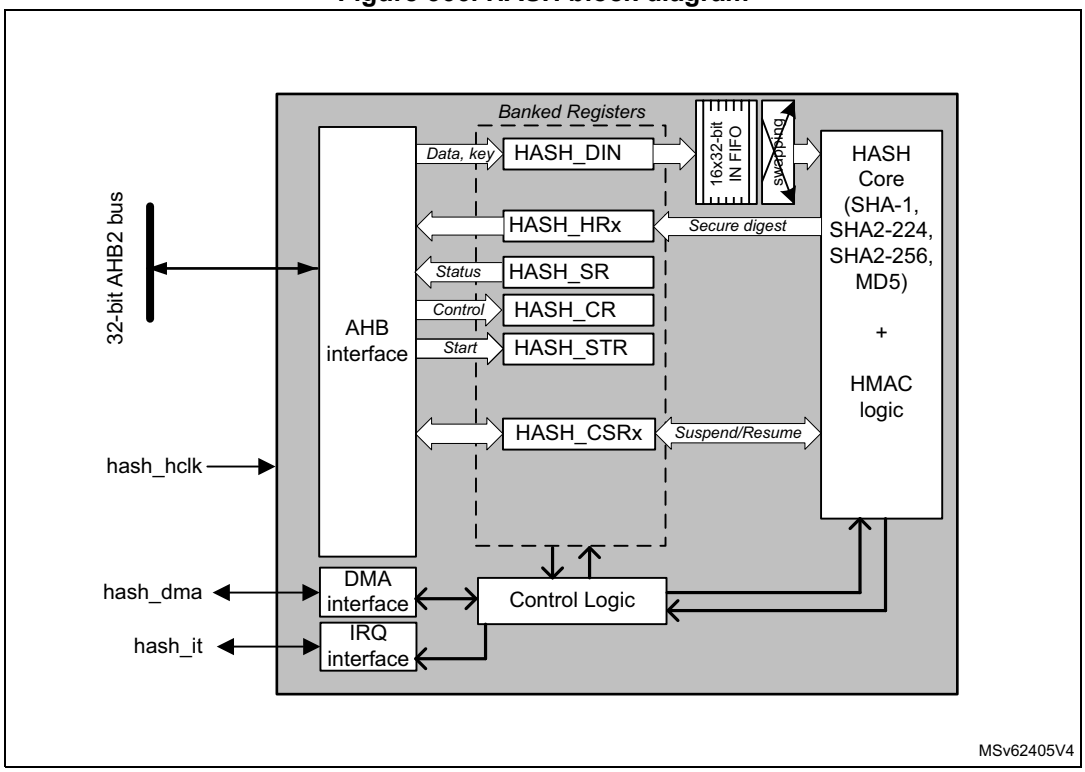
The devices have a single instance of HASH peripheral.

43.4 HASH functional description

43.4.1 HASH block diagram

Figure 366 shows the block diagram of the hash processor.

Figure 366. HASH block diagram



43.4.2 HASH internal signals

Table 377 describes a list of useful to know internal signals available at HASH level, not at product level (on pads).

Table 377. 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

43.4.3 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 SHA2-224 and SHA2-256, 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.

43.4.4 Message data feeding

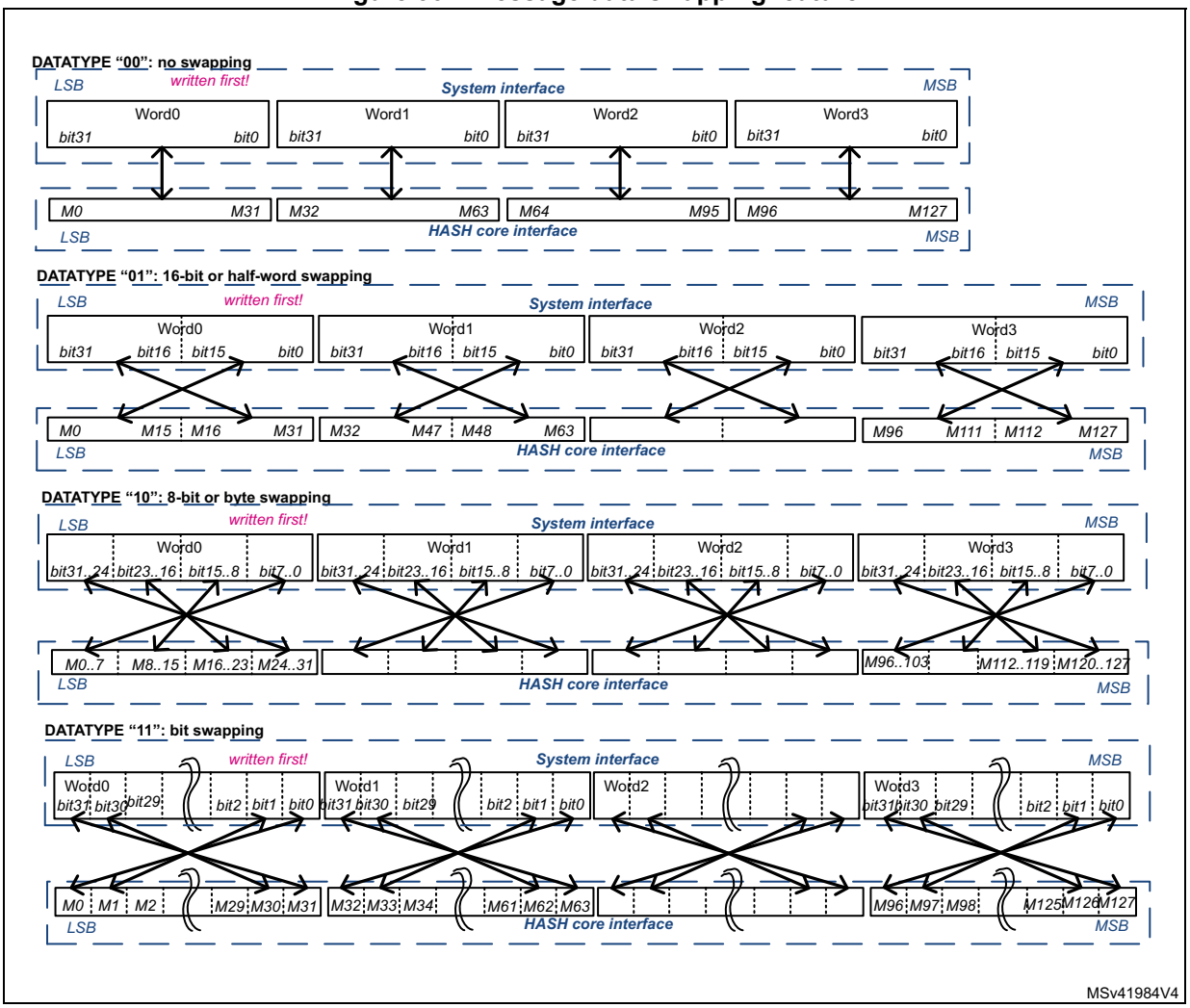
The message (or data file) to be processed by the HASH should be considered as a bit string. Per FIPS PUB 180-4 standard this message bit string grows from left to right, with hexadecimal words expressed in “big-endian” convention, so that within each word, the most significant bit is stored in the left-most 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 367* 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 367. Message data swapping feature



43.4.5 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 43.4.4: 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 43.4.7: HMAC operation* for details.
   c) 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, the block size being defined the NBWE bit of HASH_CR. 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 43.4.6* 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 43.4.6* for details).

c) When data are filled by DMA 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 by DMA 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 378*.

Table 378. 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
SHA2-224	HASH_H0 to HASH_H6	HASH_H0[31]	224
SHA2-256	HASH_H0 to HASH_H7		256

For more information about HMAC detailed instructions, refer to *Section 43.4.7: HMAC operation*.

43.4.6 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 43.4.5: 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
```

43.4.7 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.

Note: HMAC mode of the hash processor can be used with all supported algorithms.

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 43.4.4: 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 43.4.4: 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_HRx) 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 43.6: 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:

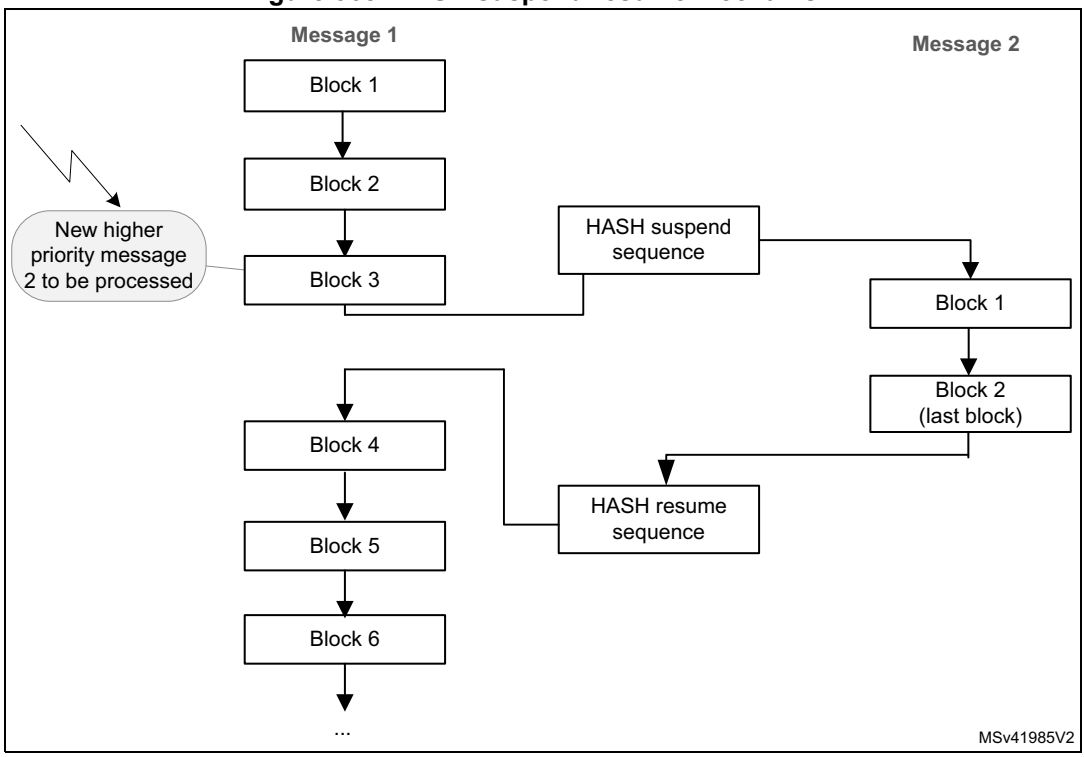
   HASH_HR0 = 0x5FD596EE
   HASH_HR1 = 0x78D5553C
   HASH_HR2 = 0x8FF4E72D
   HASH_HR3 = 0x266DFD19
   HASH_HR4 = 0x2366DA29

43.4.8 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 368*.

Figure 368. 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 hereafter.

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 the number of words defined in NBWE:

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.

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. In this sequence the DMA channel allocated to the hash peripheral remains allocated to the processing of message 1 (see *Figure 368*).

To suspend the processing of a message using DMA, proceed as follows:

1. Clear the DMAE bit to disable the DMA interface. The hash peripheral automatically fetches enough data using the DMA to complete the current input block and launch a hash process.
2. Wait until the last DMA transfer is complete (DMAS = 0 in HASH_SR).
3. Disable the DMA channel.
4. In Polling or Interrupt mode (recommended), wait until the hash processor is ready (no block is being processed), that is wait for DINIS = 1 in HASH_SR. If DCIS is also set in this register 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
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.

43.4.9 HASH DMA interface

The HASH supports both single and fixed DMA burst transfers of four words.

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 43.4.5: 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.

43.4.10 HASH error management

No error flags are generated by the hash processor.

43.5 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 379* gives a summary of the available features.

Table 379. 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

43.6 HASH processing time

Table 380 summarizes the time required to process an intermediate block for each mode of operation.

Table 380. Processing time (in clock cycle)

Mode of operation	FIFO load(1)	Computation phase	Total
MD5	16	50	**66**
SHA-1	16	66	**82**
SHA2-224	16	50	**66**
SHA2-256			

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

43.7 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.

43.7.1 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.	ALGO[1:0]		LKEY
													rw	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]				Res.	MODE	DATATYPE[1:0]		DMAE	INIT	Res.	Res.
		rw	r	r	r	r	r		rw	rw	rw	rw	rw		

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

Bits 18:17 **ALGO[1:0]**: Algorithm selection

These bits select the hash algorithm.
00: SHA-1
01: MD5
10: SHA2-224
11: SHA2-256
This selection is only taken into account when the INIT bit is set. Changing this bitfield during a computation has no effect.
When ALGO bitfield is updated and INIT bit is set, NBWE in HASH_SR is automatically updated to 0x11.

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 and MODE bits are both set. 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

Refer to DINNE bit of HASH_SR for the description.
This bit is read-only.

Bits 11:8 **NBW[3:0]:** Number of words already pushed
Refer to NBWP[3:0] bitfield of HASH_SR for the description.
This bitfield is read-only.

Bit 7 Reserved, must be kept at reset value.

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 ongoing 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.

43.7.2 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 zeros.

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]															
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
DATAIN[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

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 zeros.

43.7.3 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].

43.7.4 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 SHA2-224 digest result.
- HASH_HR0 to HASH_HR7 registers return the SHA2-256 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 returns zeros.

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 43.7.4: 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 43.7.4: 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 43.7.4: HASH digest registers* introduction.

43.7.5 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

43.7.6 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.	NBWE[4:0]				
											r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DINNE	Res.	NBWP[4:0]					Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
r		r	r	r	r	r						r	r	rc_w0	rc_w0

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **NBWE[4:0]:** Number of words expected

This bitfield reflects the number of words in the message that must be pushed into the FIFO to trigger a partial computation. NBWE is decremented by 1 when a write access is performed to the HASH_DIN register.

NBWE is set to the expected block size +1 in words (0x11) when INIT bit is set in HASH_CR, and it is set to the expected block size when partial digest calculation ends.

Bit 15 **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

Bit 14 Reserved, must be kept at reset value.

Bits 13:9 **NBWP[4:0]:** Number of words already pushed

This bitfield is the exact number of words in the message that have already been pushed into the FIFO. NBWP is incremented by one when a write access is performed to the HASH_DIN register.

When a digest calculation starts, NBWP is updated to NBWP- block size (in words), and NBWP goes to zero when the INIT bit is written to 1.

Bits 8:4 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.

43.7.7 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 43.7.7: HASH context swap registers* introduction.

43.7.8 HASH register map

Table 381 gives the summary HASH register map and reset values.

Table 381. 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]	ALGO0	LKEY	Res.	Res.	MDMAT	DINNE	NBW[3:0]				Res.	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.	DCIE	DINIE	
	Reset value																														0	0	
0x24	HASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBWE4	NBWE3	NBWE2	NBWE1	NBWE0	DINNE	Res.	NBWP4	NBWP3	NBWP2	NBWP1	NBWP0	Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
	Reset value												0	0	0	0	0	0		0	0	0	0	0						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 381. 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 115* for the register boundary addresses.

44 On-the-fly decryption engine (OTFDEC)

This section only applies to STM32U585 devices.

44.1 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.

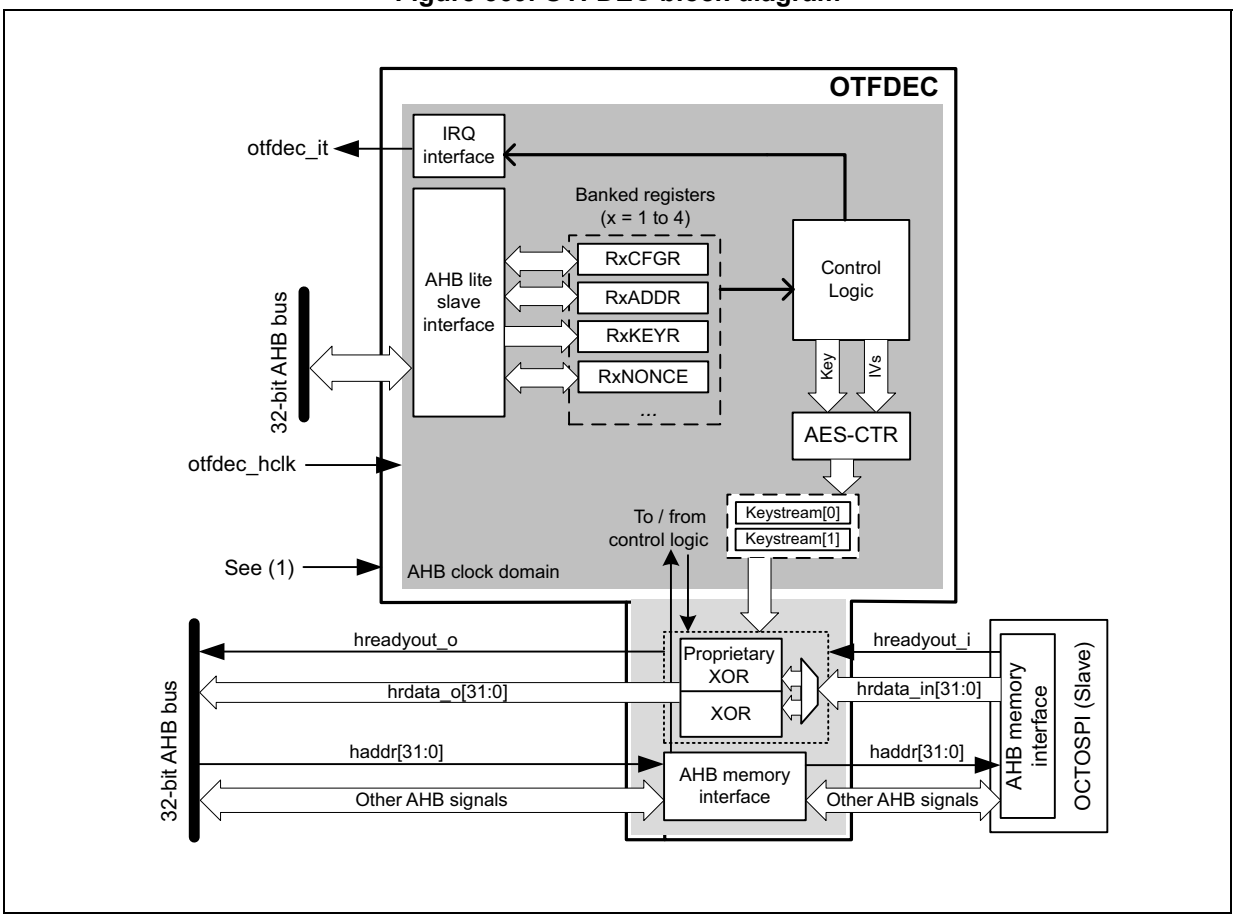
44.2 OTFDEC main features

- 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)
- Privileged-aware 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

44.3 OTFDEC functional description

44.3.1 OTFDEC block diagram

Figure 369. OTFDEC block diagram



1. otfdec_tzen

44.3.2 OTFDEC internal signals

Table 382 describes a list of useful to know internal signals available at OTFDEC level, not at the product level (on pads).

Table 382. 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.

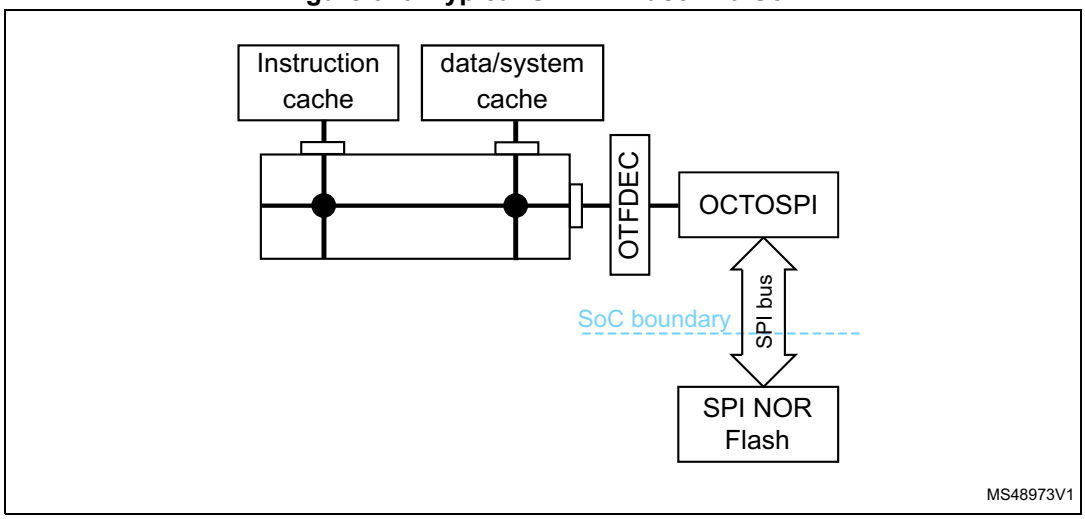
- TZEN = 1: TrustZone security is enabled in the product.
- TZEN = 0: TrustZone security is disabled in the product.

44.3.3 OTFDEC on-the-fly decryption

Introduction

Typical usage for OTFDEC is shown on *Figure 370*.

Figure 370. 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 access to most registers can be made privileged-only by setting PRIV bit in OTFDEC_PRIVCFGR register. 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 44.5.3: 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.

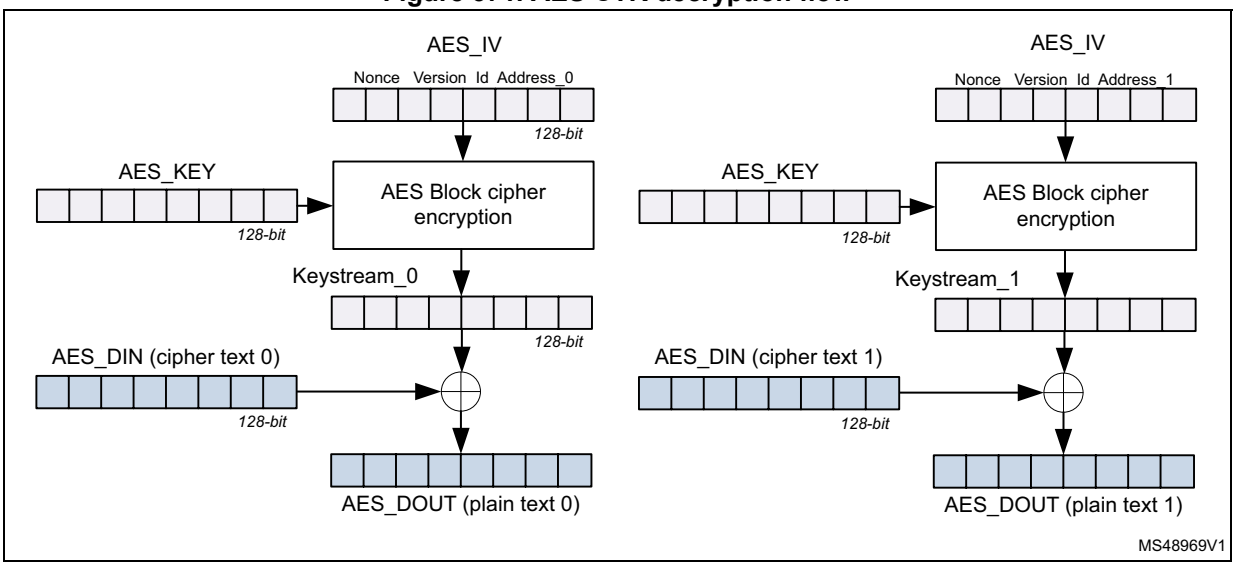
Note: Although OTFDEC does not prevent region overlapping, it is not a valid programming and it must be avoided by application software.

OTFDEC can decrypt incremental or wrap bursts only if they do not cross the 4096-byte aligned address boundaries.

44.3.4 OTFDEC usage of AES in counter mode decryption

Figure 371 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 371. 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).

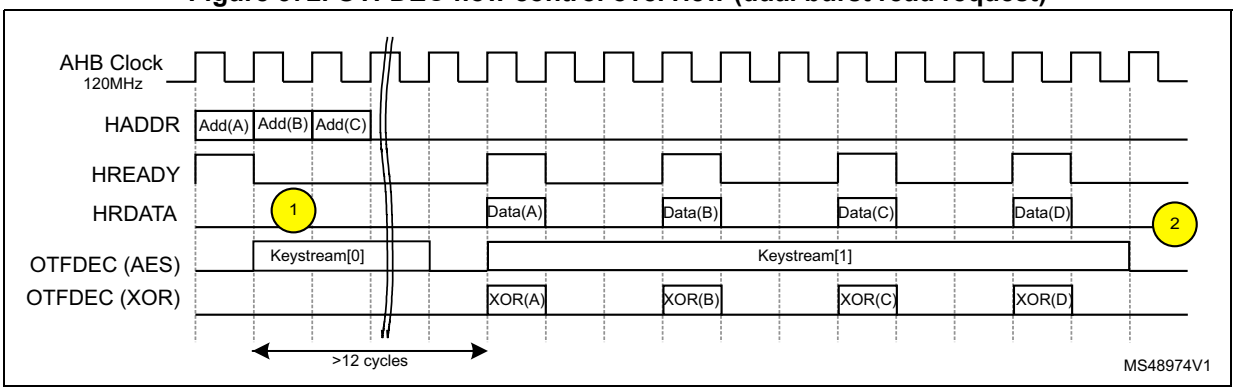
When the read request is not within an encrypted region, or the decryption is not enabled in this region, the AHB data is not changed.

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).

44.3.5 Flow control management

Figure 372 shows how OTFDEC manages one INCR4 AHB burst that corresponds to one 128-bit AES data block.

Figure 372. 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).

44.3.6 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 44.4*).

Note: After a key error, OTFDEC keys must be properly initialized again, and a reset of OTFDEC may be needed if registers are locked.

44.4 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 44.3.6*
- Illegal write to a region configuration while CONFIGLOCK = 1 (SEIF flag), see *Section 44.3.6*
- 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 44.3.6*.

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 383*. Status of the interrupt event is found in OTFDEC_ISR, and this event can be cleared using OTFDEC_ICR.

Table 383. 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.

44.5 OTFDEC application information

44.5.1 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).

44.5.2 OTFDEC and power management

Each time OTFDEC is reset, the correct key loading sequence described in *Section 44.5.1* 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.

44.5.3 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 44.3.4: OTFDEC usage of 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 44.5.1: 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.

44.5.4 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;
}
```

44.6 OTFDEC registers

44.6.1 OTFDEC control register (OTFDEC_CR)

Address offset: 0x0

Reset value: 0x0000 0000

Non-secure 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).

44.6.2 OTFDEC privileged access control configuration register (OTFDEC_PRIVCFGR)

Address offset: 0x10

Reset value: 0x0000 0000

Non-secure 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.

44.6.3 OTFDEC region x configuration register (OTFDEC_RxCFGR)

Address offset: 0x20 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Non-secure 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
REG_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 **REG_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 44.5.4*.

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.

44.6.4 OTFDEC region x start address register (OTFDEC_RxSTARTADDR)

Address offset: 0x24 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Non-secure 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
REG_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
REG_START_ADDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **REG_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 .

44.6.5 OTFDEC region x end address register (OTFDEC_RxENDADDR)

Address offset: 0x28 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0FFF

Non-secure 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
REG_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
REG_END_ADDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **REG_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.

44.6.6 OTFDEC region x nonce register 0 (OTFDEC_RxNONCER0)

Address offset: 0x2C + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Non-secure 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
REG_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
REG_NONCE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **REG_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.

44.6.7 OTFDEC region x nonce register 1 (OTFDEC_RxNONCER1)

Address offset: 0x30 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Non-secure 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
REG_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
REG_NONCE[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **REG_NONCE[63:32]**: Region nonce, bits [63:32]

Refer to the OTFDEC_RxNONCER0 register for description of the NONCE[63:0] bitfield.

44.6.8 OTFDEC region x key register 0 (OTFDEC_RxKEYR0)

Address offset: 0x34 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Non-secure 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
REG_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
REG_KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **REG_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.*

44.6.9 OTFDEC region x key register 1 (OTFDEC_RxKEYR1)

Address offset: 0x38 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Non-secure 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
REG_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
REG_KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **REG_KEY[63:32]**: Region key, bits [63:32]
Refer to the OTFDEC_RxKEYR0 register for description of the KEY[127:0] bitfield.

44.6.10 OTFDEC region x key register 2 (OTFDEC_RxKEYR2)

Address offset: 0x3C + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Non-secure 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
REG_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
REG_KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **REG_KEY[95:64]**: Region key, bits [95:64]
Refer to the OTFDEC_RxKEYR0 register for description of the KEY[127:0] bitfield.

44.6.11 OTFDEC region x key register 3 (OTFDEC_RxKEYR3)

Address offset: 0x40 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Non-secure 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
REG_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
REG_KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **REG_KEY[127:96]**: Region key, bits [127:96]

Refer to the OTFDEC_RxKEYR0 register for description of the KEY[127:0] bitfield.

44.6.12 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.

44.6.13 OTFDEC interrupt clear register (OTFDEC_ICR)

Address offset: 0x304

Reset value: 0x0000 0000

Non-secure 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.

44.6.14 OTFDEC interrupt enable register (OTFDEC_IER)

Address offset: 0x308

Reset value: 0x0000 0000

Non-secure 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).

44.6.15 OTFDEC register map

Table 384. 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
0x00	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
0x04-0x0C	Reserved	Reserved																															
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

Table 384. 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
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
0x44 - 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
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
0x74 - 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

Table 384. 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
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
0xA4 - 0xAC	Reserved	Reserved																															
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																															

Table 384. 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
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.		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.		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.		KEIE	XONEIE	SEIE
	Reset value																														0	0	0

Refer to *Section 2.3* for the register boundary addresses.

45 Public key accelerator (PKA)

This section only applies to STM32U585 devices.

45.1 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.

When manipulating secrets, the PKA incorporates a protection against side-channel attacks (SCA), including differential power analysis (DPA), certified SESIP and PSA security assurance level 3.

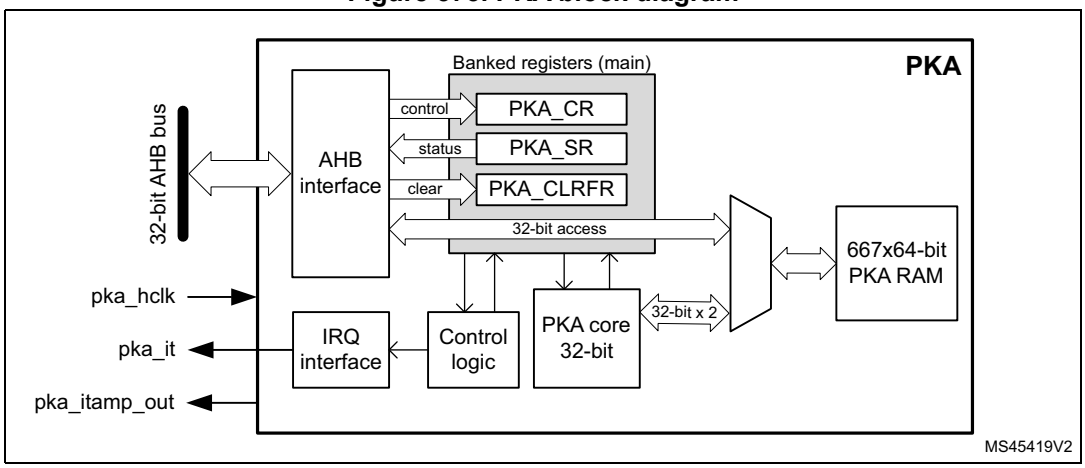
45.2 PKA main features

- 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, complete addition, double base ladder, projective to affine
  - ECDSA signature generation and verification
- Capability to handle operands up to 4160 bits for RSA/DH and 640 bits for ECC.
- When manipulating secrets: protection against side-channel attacks (SCA), including differential power analysis (DPA), certified SESIP and PSA security assurance level 3.
  - Applicable to modular exponentiation, ECC scalar multiplication and ECDSA signature generation.
- 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 an AHB bus error is generated, and write accesses are ignored).

45.3 PKA functional description

45.3.1 PKA block diagram

Figure 373. PKA block diagram



45.3.2 PKA internal signals

Table 385 lists the internal signals available at the PKA level, not necessarily available on product bonding pads.

Table 385. 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
pka_itamp_out	Digital output	PKA internal tamper event signal to TAMP (XOR-ed), triggered when an unexpected fault occurs while PKA manipulates secrets, or when the programmed input point is not found on the input curve (ECDSA signature and ECC scalar multiplication only). This signal is asserted as soon as a fault is detected. When asserted read access to PKA registers are reset to 0 and writes are ignored. The signal is de-asserted when PKA memory is cleared.

45.3.3 PKA reset and clocks

PKA is clocked on the AHB bus clock. When the PKA peripheral reset signal is released PKA_RAM is cleared automatically, taking 667 clock cycles. During this time the setting of bit EN in PKA_CR register 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).

45.3.4 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 4160 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 can be found in *Section 45.5.1: Supported elliptic curves*.

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.

A memory of 5336 bytes (667 words of 64 bits) called PKA RAM is used to provide initial data to the PKA, and to hold 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 386* and *Table 387*, respectively, for integer arithmetic functions and prime field (Fp) elliptic curve functions.

Table 386. PKA integer arithmetic functions list

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x01	000001	Montgomery parameter computation R2 mod n	*Section 45.4.2*
0x0E	001110	Modular addition (A+B) mod n	*Section 45.4.3*
0x0F	001111	Modular subtraction (A-B) mod n	*Section 45.4.4*
0x10	010000	Montgomery multiplication (AxB) mod n	*Section 45.4.5*
0x00	000000	Modular exponentiation A^e mod n	*Section 45.4.6*
0x02	000010	Modular exponentiation A^e mod n (fast mode)	
0x03	000011	Modular exponentiation A^e mod n (protected)	*Section 45.4.6*
0x08	001000	Modular inversion A^{-1} mod n	*Section 45.4.7*
0x0D	001101	Modular reduction A mod n	*Section 45.4.8*
0x09	001001	Arithmetic addition A+B	*Section 45.4.9*
0x0A	001010	Arithmetic subtraction A-B	*Section 45.4.10*
0x0B	001011	Arithmetic multiplication AxB	*Section 45.4.11*
0x0C	001100	Arithmetic comparison (A=B, A>B, A<B)	*Section 45.4.12*
0x07	000111	RSA CRT exponentiation	*Section 45.4.13*

Table 387. 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 45.4.14*
0x20	100000	ECC scalar multiplication kP (protected)	*Section 45.4.15*
0x23	100011	ECC complete addition	*Section 45.4.18*
0x24	100100	ECDSA sign (protected)	*Section 45.4.16*
0x26	100110	ECDSA verification	*Section 45.4.17*
0x27	100111	ECC double base ladder	*Section 45.4.19*
0x2F	101111	ECC projective to affine	*Section 45.4.20*

Each of these operating modes has an associated code that has to be written to the MODE field in the PKA_CR register. If the application selects any value that is not documented below the write to MODE bitfield is ignored, and an operation error (OPERRF) is triggered. When this happens a new operation must be selected after the error is cleared.

Some operations in *Table 386* and *Table 387* are indicated as protected. Those operations are used when manipulating secret keys (modular exponentiation for RSA decryption, scalar multiplication and signature for ECC). Those secrets (protected against side channel attacks) are automatically erased from PKA RAM at the end of the protected operations (BUSY goes low).

Caution: For security reason it is very important to select protected modular exponentiation (MODE=0x3) when performing RSA decryption.

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 45.5.2: Computation times*.

The only operation using fast mode is modular exponentiation (MODE=0x02).

45.3.5 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 the 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 the 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 coprimes.
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 *d*, follows the steps indicated below:

1. Convert the ciphertext C to an integer ciphertext representative c.
2. If necessary retrieve the prime factors (p, q) using (n, e, d) information, then compute phi = (p - 1) * (q - 1). Refer to NIST SP800-56B Appendix C for details.
3. 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) $m = m_2 + h\ q$
4. Convert the integer message representative *m* to an encoded message EM.
5. 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.

Note: For the decryption process protected version of modular exponentiation (MODE=0x3) is strongly recommended for security reason. For encryption process MODE=0x3 cannot be used, as it requires the knowledge of the private key.

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 45.5.1: 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, goes through the following steps.

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 $w = s^{-1} \bmod n$
5. calculate $u_1 = zw \bmod n$ and $u_2 = rw \bmod n$
6. calculate the curve point $(x_1, y_1) = u_1 \times G + u_2 \times Q_A$
7. the signature is valid if $r = x_1 \pmod n$, it is invalid otherwise.

Steps 4 to 7 are accelerated by PKA using *ECDSA verification*.

45.3.6 PKA procedure to perform an operation

Enabling/disabling PKA

Setting the EN bit to 1 in PKA_CR register enables the PKA peripheral. The PKA becomes available when INITOK bit is set in PKA_SR. 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.

Note: When setting EN bit in PKA_CR make sure that the value of MODE bitfield corresponds to an authorized PKA operation (see OPERRF in Section 45.3.7).

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, with the exception of the PKA modes 0x03, 0x20 and 0x24. For those operations the PKA memory is cleared after abort, making the memory unavailable for 667 cycles. During this clearing time only PKA registers can be accessed, with writes to EN bits ignored.

If INITOK bit stays at 0 make sure the RNG peripheral is clocked and properly initialized, then try to enable PKA again.

Data formats

The format of the input data and the results in the PKA RAM are specified, for each operation, in *Section 45.4*.

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.

Selecting an illegal or unknown operation in step *2* triggers an OPERRF error, and step *3* (PROCENDF=1) never happens. See *Section 45.3.7* for details.

Using precomputed Montgomery parameters (PKA Fast mode)

As explained in *Section 45.3.4*, 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 45.4.5*). This is know as “Fast mode”.

To manage the usage of Fast mode it is recommended to follow the procedure described below:

1. Load in PKA RAM the modulus size and value information. Such information is compiled in *Section 45.5.1*.
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 45.4*.

45.3.7 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.
- The selected operating mode using MODE bitfield is not listed in PKA operating modes (or in bitfield description). In this case the operation error flag (OPERRF) is set in the PKA_SR register, and write to MODE bitfield is ignored.

For each error flag above PKA generates an interrupt if the application sets the corresponding bit in PKA_CR register (see *Section 45.6* for details).

ADDRERRF, OPERRF 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.

OPERRF error must be cleared using OPERRFC bit in PKA_CLRFR before a new operation is written in PKA_CR register.

45.4 PKA operating modes

45.4.1 Introduction

The various operations supported by PKA are described in the following subsections, defining the format of the input data and of the results, both stored in the PKA RAM.

Warning: Validity of all input parameters to the PKA must be checked before starting any PKA operation, as PKA assumes that all input parameters are valid and consistent with each other.

Input parameters must not exceed the specified operand size in the operation tables.

The following information applies to all PKA operations.

- PKA core processes 64-bit words in its RAM. Hence hereafter all word size is 64-bit
- When an element is written as input in the PKA RAM, an additional word with all bits equal to zero has to be added after the most significant input word. This rule does not apply if the operand has a fixed size of 1.
- All reported RAM storage addresses refer to the least significant word of the data, and to obtain the actual address to use application must add to the indicated offset the base address of the PKA.
- Supported operand “Size” are:
  - ROS (RSA operand Size): data size is (*rsa_size* / 64 + 1) words, with *rsa_size* equal to the chosen modulus length in bits. For example, when computing RSA with an operand size of 1024 bits, ROS is equal to 17 words, or 1088 bits.
  - EOS (ECC operand Size): data size is (*ecc_size* / 64 + 1) words, with *ecc_size* equal to the chosen prime modulus length in bits. For example, when computing ECC with an operand size of 192 bits, EOS is equal to 4 words, or 256 bits.
  - ROS and EOS values include the required additional all 0 word.
- Unless indicated otherwise, all operands in the tables are integers.

Note: Fractional results for above formulas must be rounded up to the nearest integer since PKA core processes 64-bit words.

Note: The maximum ROS is 66 words (4160-bit max exponent size), while the maximum EOS is 11 words (640-bit max operand size).

As a first example (and to better understand the endianess in PKA memory), to prepare the operation *ECC Fp scalar multiplication*, when the application writes the x coordinate of point P for an ECC P256 curve (EOS = 5 words), the least significant bit must be placed in bit 0 at address offset 0x578, and the most significant bit in bit 63 at address offset 0x590. Then, as mentioned above, the application must write the empty word 0x0000000000000000 at address offset 0x598.

As a second example, still to prepare the operation ECC Fp scalar multiplication, when the application need to write the information a = -3, on a curve with a modulus length of 224 bits (i.e. four 64-bit words, rounded up, plus one) following data must be written in PKA memory:

```
@RAM+410  0x0000000000000001  /* curve coefficient 'a' sign without extra word */
@RAM+418  0x0000000000000011  /* value of |a| LSB
@RAM+420  0x0000000000000000  ...
@RAM+428  0x0000000000000000  ...
@RAM+430  0x0000000000000000  value of |a| MSB */
@RAM+438  0x0000000000000000  /* additional all 0 word */
```

45.4.2 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. This operation can be very useful when fast mode operation is used, because in this case the Montgomery parameter is passed as input, saving the time for its computation.

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 388*.

Table 388. Montgomery parameter computation

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x01	PKA_CR	6 bits
	Modulus length	(in bits, 0 ≤ value < 4160)	RAM@0x408	64 bits
	Modulus value n	(odd integer only, n < 2^{4160})	RAM@0x1088	ROS
OUT	Result: R^2 mod n	-	RAM@0x620	

45.4.3 Modular addition

Modular addition operation consists in the computation of A + B mod *n*. Operation instructions are summarized in *Table 389*.

Table 389. Modular addition

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0E	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand A	(0 ≤ A < n)	RAM@0xA50	ROS
	Operand B	(0 ≤ B < n)	RAM@0xC68	
	Modulus value n	(n < 2^{4160})	RAM@0x1088	
OUT	Result: A+B mod n	(0 ≤ result < n)	RAM@0xE78	

45.4.4 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 390*.

Table 390. Modular subtraction

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0F	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand A	(0 ≤ A < n)	RAM@0xA50	ROS
	Operand B	(0 ≤ B < n)	RAM@0xC68	
	Modulus value n	(n < 2^{4160})	RAM@0x1088	
OUT	Result: A-B mod n	(0 ≤ result < n)	RAM@0xE78	

45.4.5 Modular and Montgomery multiplication

To be more efficient when performing a sequence of multiplications the PKA accelerates multiplication which has at least one input in the Montgomery domain. The two main uses of this operation are:

- Map a value from natural domain to Montgomery domain and vice-versa
- Perform a modular multiplication A x B mod *n*

The method to perform above operations are described below. Note that “x” function is this operation, and A, B, C operands are in the natural domain.

1. Inward (or outward) conversion into (or from) Montgomery domain
   a) Let us 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 us 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 391*.

Table 391. Montgomery multiplication

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x10	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand A	(0 ≤ A < n)	RAM@0xA50	ROS
	Operand B	(0 ≤ B < n)	RAM@0xC68	
	Modulus value n	(odd integer only, n < 2^{4160})	RAM@0x1088	
OUT	Result: AxB mod n(1)	-	RAM@0xE78	

1. Result in Montgomery domain or in natural domain, depending upon the inputs nature (see examples *2* and *3*).

45.4.6 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*.

RSA operation involving public information (RSA encryption) can use the normal or fast mode detailed on *Table 392* and *Table 393*. RSA operation involving secret information (RSA decryption) must use the protected mode detailed on *Table 394*, for security reason.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware.

Note: When MODE=0x03 if the error output is different from 0xD60D all the memory content is cleared by PKA to avoid leaking information about the private key.

Operation instructions for modular exponentiation are summarized in *Table 392* (normal mode), *Table 393* (fast mode) and in *Table 394* (protected mode). Fast mode usage is explained in *Section 45.3.6*.

Table 392. 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	64 bits
	Operand length	(in bits, not null)	RAM@0x408	
IN/OUT	Operand A (base of exponentiation)	(0 ≤ A < n)	RAM@0xC68	ROS
IN	Exponent e	(0 ≤ e < n)	RAM@0xE78	
	Modulus value n	(odd integer only, n < 2^{4160})	RAM@0x1088	
OUT	Result: A^e mod n	(0 ≤ result < n)	RAM@0x838	

Table 393. 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	64 bits
	Operand length	(in bits, not null)	RAM@0x408	
IN/OUT	Operand A (base of exponentiation)	(0 ≤ A < n)	RAM@0xC68	ROS
IN	Exponent e	(0 ≤ e < n)	RAM@0xE78	
	Modulus value n	(odd integer only, n < 2^{4160})	RAM@0x1088	
IN/OUT	Montgomery parameter R2 mod n	(mandatory)	RAM@0x620	
OUT	Result: A^e mod n	(0 ≤ result < n)	RAM@0x838	

Table 394. Modular exponentiation (protected mode)

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x03	PKA_CR	6 bits
	Exponent e length	(in bits, not null)	RAM@0x400	64 bits
	Modulus or operand length	(in bits, not null)	RAM@0x408	
	Operand A (base of exponentiation)	(0 ≤ A < n)	RAM@0x16C8	ROS
	Exponent e	(0 ≤ e < n)	RAM@0x14B8	
	Modulus value n	(odd integer only, n < 2^{4096})	RAM@0x0838	
	Phi value(1)	-	RAM@0x0C68	
OUT	Result: A^e mod n	(0 ≤ result < n)	RAM@0x838	
ERROR	Error A^e mod n	– No errors: 0xD60D – Errors: 0xCBC9	RAM@0x1298	64 bits

1. Euler totient function of n^1 with phi = (p - 1) * (q - 1), where p and q are prime factors of modulus n (see NIST SP800-56B Appendix C or *RSA encryption/decryption principle* for details). As optimization it is recommended to keep phi information as part of the key pair generation. Alternative is to store the key as (*p*, *q*, *e*, *d*) instead of (*N*, *e*, *d*, *phi*), in this case, to derive *N* and *phi* using PKA arithmetic multiplier: *N* = *p* * *q*, and *phi* = (*p* - 1) * (*q* - 1).

45.4.7 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 greatest 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 395*.

Table 395. Modular inversion

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x08	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand A	(0 ≤ A < n)	RAM@0xA50	ROS
	Modulus value n	(odd integer only, n < 2^{4160})	RAM@0xC68	
OUT	Result: A^{-1} mod n	0 < result < n	RAM@0xE78	

45.4.8 Modular reduction

Modular reduction operation consists in the computation of the remainder of A divided by n. Operation instructions are summarized in *Table 396*.

Table 396. Modular reduction

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0D	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x400	64 bits
	Modulus length	(in bits, 8 < value < 4160)	RAM@0x408	
	Operand A	(0 ≤ A < 2n < 2^{4160})	RAM@0xA50	ROS
	Modulus value n	(odd integer only, n < 2^{4160})	RAM@0xC68	
OUT	Result A mod n	(0 < result < n)	RAM@0xE78	

45.4.9 Arithmetic addition

Arithmetic addition operation consists in the computation of A + B. Operation instructions are summarized in *Table 397*.

Table 397. Arithmetic addition

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x09	PKA_CR	6 bits
	Operand length M	(in bits, not null)	RAM@0x408	64 bits
	Operand A	(0 ≤ A < 2^M)	RAM@0xA50	ROS
	Operand B	(0 ≤ B < 2^M)	RAM@0xC68	
OUT	Result: A+B	(0 ≤ result < 2^{M+1})	RAM@0xE78	ROS + 1

45.4.10 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

For the last two bullets the 32-bit word following the most significant word of the output equals 0xFFFF FFFF, as result is negative.

Operation instructions are summarized in *Table 398*.

Table 398. Arithmetic subtraction

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0A	PKA_CR	6 bits
	Operand length M	(in bits, not null)	RAM@0x408	64 bits
	Operand A	($0 \leq < A < 2^M$)	RAM@0xA50	ROS
	Operand B	($0 \leq < B < 2^M$)	RAM@0xC68	
OUT	Result: A-B	($0 \leq$ result $< 2^M$)	RAM@0xE78	

45.4.11 Arithmetic multiplication

Arithmetic multiplication operation consists in the computation of AxB. Operation instructions are summarized in *Table 399*.

Table 399. Arithmetic multiplication

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0B	PKA_CR	6 bits
	Operand length M	(in bits, not null)	RAM@0x408	64 bits
	Operand A	($0 \leq A < 2^M$)	RAM@0xA50	ROS
	Operand B	($0 \leq B < 2^M$)	RAM@0xC68	
OUT	Result: AxB	($0 \leq$ result $< 2^M$)	RAM@0xE78	2xROS

45.4.12 Arithmetic comparison

Arithmetic comparison operation consists in the following computation:

- If A = B then result = 0xED2C
- If A > B then result = 0x7AF8
- If A < B then result = 0x916A

Operation instructions for arithmetic comparison are summarized in *Table 400*.

Table 400. Arithmetic comparison

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x0C	PKA_CR	6 bits
	Operand length M	(in bits, not null)	RAM@0x408	64 bits
	Operand A	($0 \leq A < 2^M$)	RAM@0xA50	ROS
	Operand B	($0 \leq B < 2^M$)	RAM@0xC68	
OUT	Result A?B	0xED2C, 0x7AF8 or 0x916A	RAM@0xE78	64 bits

45.4.13 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:

- p and q are precomputed primes, stored as part of the private key
- $d_P = d \bmod (p-1)$
- $d_Q = d \bmod (q-1)$ and
- $q_{inv} = q^{-1} \bmod p$

These values allow the recipient to compute the exponentiation $m = A^d \pmod{pq}$ more efficiently as follows:

- $m_1 = A^{dP} \bmod p$
- $m_2 = A^{dQ} \bmod p$
- $h = q_{inv} (m_1 - m_2) \bmod p$, with $m_1 > m_2$
- $m = m_2 + hq$

Operation instructions for computing CRT exponentiation A^d mod pq are summarized in *Table 401*.

Table 401. CRT exponentiation

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x07	PKA_CR	6 bits
IN	Operand length	(in bits, not null)	RAM@0x408	64 bits
IN	Operand d_P	$(0 \le d_P < 2^{M/2})$	RAM@0x730	ROS / 2
	Operand d_Q	$(0 \le d_Q < 2^{M/2})$	RAM@0xE78	
	Operand q_{inv}	$(0 \le q_{inv} < 2^{M/2})$	RAM@0x948	
	Prime p(1)	$(0 \le p < 2^{M/2})$	RAM@0xB60	
	Prime q(1)	$(0 \le q < 2^{M/2})$	RAM@0x1088	
IN	Operand A	$(0 \le A < 2^{M/2})$	RAM@0x12A0	ROS
OUT	Result: $A^d \bmod pq$	$(0 \le \text{result} < pq)$	RAM@0x838	

1. Must be different from 2.

45.4.14 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 402*.

Table 402. 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@0x408	64 bits
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x410	
	Curve coefficient |a|	(absolute value, |a| < p)	RAM@0x418	EOS
	Curve coefficient *b*	(|b| < p)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, 0 < p < 2640)	RAM@0x470	
	Point P coordinate *x*	(x < *p*)	RAM@0x578	
	Point P coordinate *y*	(y < *p*)	RAM@0x5D0	
	Montgomery parameter R2 mod n	-	RAM@0x4C8	
OUT	Result: point P on curve	– 0xD60D: point on curve – 0xA3B7: point not on curve – 0xF946: x or y coordinate is not smaller than modulus p	RAM@0x680	64 bits

45.4.15 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 403*.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events, a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware. PKA tamper is also triggered when the programmed input point is not found on the input ECC curve. PKA operation "Point on elliptic curve" can be used to avoid this.

Table 403. ECC Fp scalar multiplication

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x20	PKA_CR	6 bits
IN	Curve prime order *n* length	(in bits, not null,)	RAM@0x400	64 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	
	Curve coefficient *a* sign	– 0x0: positive – 0x1: negative	RAM@0x410	

Table 403. ECC Fp scalar multiplication (continued)

Parameters with direction		Value (Note)	Storage	Size
IN	Curve coefficient |*a*|	(absolute value, |a| < p)	RAM@0x418	EOS
	Curve coefficient *b*	(positive integer)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x1088	
	Scalar multiplier k	($0 \leq k < 2^{640}$)	RAM@0x12A0	
	Point P coordinate x_P	($x < p$)	RAM@0x578	
	Point P coordinate y_P	($y < p$)	RAM@0x470	
	Curve prime order *n*	(integer prime)	RAM@0xF88	
OUT	Result: k x P coordinate *x'*	(result < *p*)	RAM@0x578	
	Result: k x P coordinate *y'*	(result < *p*)	RAM@0x5D0	
ERROR	Error k x P	– No errors: 0xD60D – Errors: 0xCBC9	RAM@0x680	64 bits

When performing this operation the following special cases must be noted:

- For k = 0 this function returns a point at infinity (0, 0) if curve parameter b is nonzero, (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| must 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).

Note: If the error output is different from 0xD60D all the memory content is cleared by PKA to avoid leaking information about the private key.

45.4.16 ECDSA sign

ECDSA signing operation (outlined in *Section 45.3.5*) is summarized in *Table 404* (input parameters) and in *Table 405* (output parameters).

The application has to check if the output error is equal to 0xD60D, if it is different a new k must be generated and the ECDSA sign operation must be repeated.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware. PKA tamper is also triggered when the programmed input point is not found on the input ECC curve. PKA operation "Point on elliptic curve" can be used to avoid this.

Table 404. ECDSA sign - Inputs

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x24	PKA_CR	6 bits
	Curve prime order n length (*nlen*)	(in bits, not null)	RAM@0x400	64 bits
	Curve modulus p length	(in bits, 8 < value < 640)	RAM@0x408	
	Curve coefficient a sign	0x0: positive 0x1: negative	RAM@0x410	
	Curve coefficient |a|	(absolute value, |a| < p)	RAM@0x418	EOS
	Curve coefficient b	(positive integer)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x1088	
	Integer k(1)	($0 \leq k < 2^{640}$)	RAM@0x12A0	
	Curve base point G coordinate x	(x < *p*)	RAM@0x578	
	Curve base point G coordinate y	(y < *p*)	RAM@0x470	
	Hash of message z	(hash size equal to *nlen*)(2)	RAM@0xFE8	
	Private key d	(0 < d)	RAM@0xF28	
	Curve prime order n	(integer prime)	RAM@0xF88	

1. This integer is usually a cryptographically secure random number, but in some cases k can be deterministically generated.
2. Padding with zeroes or hash truncation must be used to have the hash parameter size equal to the curve prime order n length.

Table 405. ECDSA sign - Outputs

Parameters with direction		Value (Note)	Storage	Size
OUT	Signature part *r*	(0 < *r* < n)	RAM@0x730	EOS
	Signature part *s*	(0 < *s* < n)	RAM@0x788	
ERROR	Result of signature	– 0xD60D: successful computation, no error – 0xCBC9: failed computation – 0xA3B7: signature part r is equal to 0 – 0xF946: signature part s is equal to 0	RAM@0xFE0	64 bits

Note: If the error output equals 0xD60D or 0xCBC9 all the memory content is cleared by PKA to avoid leaking information about the private key. If error output equals 0xA3B7 or 0xF946 PKA memory content is partially erased, keeping the error code readable.

Extended ECDSA support

PKA also supports extended ECDSA signature, for which the inputs and the outputs are the same as ECDSA signature (*Table 404* and *Table 405*, respectively), with the addition of the coordinates of the point kG. This extra output is defined in *Table 406*.

Table 406. 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@0x1400	EOS
	Curve point kG coordinate y_1	($0 \le y_1 < p$)	RAM@0x1458	

45.4.17 ECDSA verification

ECDSA verification operation (outlined in *Section 45.3.5*) is summarized in *Table 407* (input parameters) and *Table 408* (output parameters).

The application has to check if the output error is equal to 0xD60D, if different the signature is not verified.

Table 407. ECDSA verification - Inputs

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x26	PKA_CR	6 bits
	Curve prime order n length (*nlen*)	(in bits, not null)	RAM@0x408	64 bits
	Curve modulus p length	(in bits, not null, 8 < value < 640)	RAM@0x4C8	
	Curve coefficient a sign	0x0: positive 0x1: negative	RAM@0x468	
	Curve coefficient |a|	(absolute value, |a| < p)	RAM@0x470	EOS
	Curve modulus value *p*	(odd integer prime, $0< p < 2^{640}$)	RAM@0x4D0	
	Curve base point G coordinate x	($x < p$)	RAM@0x678	
	Curve base point G coordinate y	($y < p$)	RAM@0x6D0	
	Public-key curve point Q coordinate x_Q	($x_Q < p$)	RAM@0x12F8	
	Public-key curve point Q coordinate y_Q	($y_Q < p$)	RAM@0x1350	
	Signature part *r*	($0 < r < n$)	RAM@0x10E0	
	Signature part *s*	($0 < s < n$)	RAM@0xC68	
	Hash of message *z*	(hash size equal to *nlen*)(1)	RAM@0x13A8	
	Curve prime order *n*	(integer prime)	RAM@0x1088	

1. Padding with zeroes or hash truncation must be used to have the hash parameter size equal to the curve prime order n length.

Table 408. ECDSA verification - Outputs

Parameters with direction		Value (Note)	Storage	Size
OUT	Result: ECDSA verify	– 0xD60D: valid signature – 0xA3B7: invalid signature	RAM@0x5D0	64 bits

45.4.18 ECC complete addition

ECC complete addition computes the addition of two given points on an elliptic curve.

Operation instructions are summarized in *Table 409*.

Note: The two input points and the resulting point are represented in Jacobian coordinates (X,Y,Z). To input a point in affine coordinates (x,y) conversion (X,Y,Z) = (x,y,1) can be used. To convert resulting point to Jacobian coordinates conversion (x,y)= (X/Z^2,Y/Z^3) can be used.

Table 409. ECC complete addition

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x23	PKA_CR	6 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	64 bits
	Curve coefficient *a* sign	– 0x0: positive 0x1: negative	RAM@0x410	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	EOS
	Curve coefficient |*a*|	(absolute value, |a| < p)	RAM@0x418	
	First point P coordinate X	(x < *p*)	RAM@0x628	
	First point P coordinate Y	(y < *p*)	RAM@0x680	
	First point P coordinate Z	(z < *p*)	RAM@0x6D8	
	Second point Q coordinate X	(x < *p*)	RAM@0x730	
	Second point Q coordinate Y	(y < *p*)	RAM@0x788	
	Second point Q coordinate Z	(z < *p*)	RAM@0x7E0	
OUT	Result coordinate X	(x < *p*)	RAM@0xD60	
	Result coordinate Y	(y < *p*)	RAM@0xDB8	
	Result coordinate Z	(z < *p*)	RAM@0xE10	

45.4.19 ECC double base ladder

ECC double base ladder operation consists in the computation of k*P+m*Q, where (P,Q) are two points on an elliptic curve and (k,m) are two scalars. Operation instructions are summarized in *Table 410*.

If the resulting point is the point at infinity (error code 0xA3B7), resulting coordinate equals (0, 0).

Note: The two input points are represented in Jacobian coordinates (X, Y , Z). To input a point in affine coordinates (x, y) conversion (X, Y, Z) = (x, y, 1) can be used. The result is represented in affine coordinates (x ,y)

Table 410. ECC double base ladder

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x27	PKA_CR	6 bits
	Curve prime order *n* length	(in bits, not null)	RAM@0x400	64 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	
	Curve coefficient *a* sign	– 0x0: positive – 0x1: negative	RAM@0x410	EOS
	Curve coefficient |*a*|	(absolute value, |a| < p)	RAM@0x418	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	
	Integer k	($0 < k < 2^{640}$)	RAM@0x520	
	Integer m	($0 < m < 2^{640}$)	RAM@0x578	
	First point P coordinate X	(X < *p*)	RAM@0x628	
	First point P coordinate Y	(Y < *p*)	RAM@0x680	
	First point P coordinate Z	(Z < *p*)	RAM@0x6D8	
	Second point Q coordinate X	(X < *p*)	RAM@0x730	
	Second point Q coordinate Y	(Y < *p*)	RAM@0x788	
	Second point Q coordinate Z	(Z < *p*)	RAM@0x7E0	
OUT	Result coordinate x	(x < *p*)	RAM@0x578	EOS
	Result coordinate y	(y < *p*)	RAM@0x5D0	
	Error code	– Point not at infinity: 0xD60D – Point at infinity: 0xA3B7	RAM@0x520	64 bits

45.4.20 ECC projective to affine

ECC projective to affine operation computes the conversion between the representation of a point P in homogeneous projective coordinates and the representation of the point P in affine coordinates. Namely, if the point is represented by the triple (X, Y, Z), it computes the affine coordinates (x, y) = (X/Y, Y/Z).

All the operations are performed modulo the modulus p of the curve, which the point belongs to. If the resulting point is the point at infinity (error code 0xA3B7), resulting coordinate equals (0,0).

Operation instructions are summarized in *Table 411*.

Table 411. ECC projective to affine

Parameters with direction		Value (Note)	Storage	Size
IN	MODE	0x2F	PKA_CR	6 bits
	Curve modulus *p* length	(in bits, 8 < value < 640)	RAM@0x408	64 bits
	Curve modulus value p	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	EOS
	Point P coordinate X (projective)	(X < *p*)	RAM@0xD60	
	Point P coordinate Y (projective)	(Y < *p*)	RAM@0xDB8	
	Point P coordinate Z (projective)	(Z < *p*)	RAM@0xE10	
	Montgomery parameter R2 mod n	-	RAM@0x4C8	
OUT	Point P coordinate x (affine)	(x < *p*)	RAM@0x578	
	Point P coordinate y (affine)	(y < *p*)	RAM@0x5D0	
ERROR	Error code	– Point not at infinity: 0xD60D – Point at infinity: 0xA3B7	RAM@0x680	64 bits

45.5 Example of configurations and processing times

45.5.1 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 412 summarizes the family of curves supported by PKA for ECC operations.

Table 412. 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			
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	

45.5.2 Computation times

The following tables summarize the PKA computation times, expressed in AHB clock cycles.

Table 413. Modular exponentiation

Exponent length (in bits)	Mode	Modulus length (in bits)			
		1024	2048	3072	4096
3	Normal	124600	491000	684000	1133200
	Fast	22700	82000	178000	311000
17	Normal	135700	531400	772400	1288000
	Fast	33800	122500	266500	465800
2^{16} + 1	Normal	180000	693700	1126200	1907200
	Fast	78200	284700	620400	1085000
1024	Protected	9958000	-	-	-
	Normal	5850000	-	-	-
	Fast	5748000	-	-	-
	CRT(1)	1775000	-	-	-
2048	Protected	-	63886000	-	-
	Normal	-	42240000	-	-
	Fast	-	41832000	-	-
	CRT(1)	-	11670000	-	-
3072	Protected	-	-	199403000	-
	Normal	-	-	136830000	-
	Fast	-	-	136325000	-
	CRT(1)	-	-	36886000	-
4096	Protected	-	-	-	454318000
	Normal	-	-	-	316000000
	Fast	-	-	-	315226000
	CRT(1)	-	-	-	84577000

1. CRT stands for chinese remainder theorem optimization (MODE bitfield= 0x07).

Table 414. ECC scalar multiplication(1)

Modulus length (in bits)							
160	192	256	320	384	512	521	640
-	1590000	3083000	5339000	8518000	17818000	21053000	31826000

1. These times depend on the number of “1”s included in the scalar parameter, and include the computation of Montgomery parameter R2.

Table 415. ECDSA signature average computation time(1) (2)

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
-	1500000	2744000	4579000	7184000	14455000	16685000	24965000

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 416. ECDSA verification average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
1011000	1495000	2938000	5014000	7979000	16804000	19254000	29582000

Table 417. ECC double base ladder average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
967000	1419000	2768000	4784000	7547000	15854000	18257000	28257000

Table 418. ECC projective to affine average computation times

Modulus length (in bits)						
160	**192**	**256**	**320**	**384**	**512**	**640**
47600	78000	148300	253000	419000	838400	1049300

Table 419. ECC complete addition average computation times

Modulus length (in bits)						
160	**192**	**256**	**320**	**384**	**512**	**640**
10000	12000	18000	26000	39000	53000	89000

Table 420. Point on elliptic curve Fp check average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
3400	4200	6100	8300	10900	17200	-	-

Table 421. Montgomery parameters average computation times(1)

Modulus length (in bits)							
192	**256**	**320**	**512**	**1024**	**2048**	**3072**	**4096**
8600	8710	11870	17000	102000	410000	506000	822000

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.

45.6 PKA interrupts

There are four individual maskable interrupt sources generated by the public key accelerator, signaling the following events:

1. PKA unsupported operation error (OPERRF), see *Section 45.3.7*
2. Access to unmapped address (ADDRERRF), see *Section 45.3.7*
3. PKA RAM access while PKA operation is in progress (RAMERRF), see *Section 45.3.7*
4. PKA end of operation (PROCENDF).

The four 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 422 gives a summary of the available features.

Table 422. PKA interrupt requests

Acronym	Event	Event flag	Enable control bit	Clear method
PKA	Unsupported operation	OPERRF	OPERRIE	Set OPERRFC bit
	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

45.7 PKA registers

45.7.1 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.	OP ERRIE	ADDR ERRIE	RAM ERRIE	Res.	PROC ENDIE	Res.
										rw	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 31:22 Reserved, must be kept at reset value.

Bit 21 **OPERRIE**: Operation error interrupt enable
0: No interrupt is generated when OPERRF flag is set in PKA_SR.
1: An interrupt is generated when OPERRF flag is set in PKA_SR.

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)
000011: Modular exponentiation (protected, used when manipulating secrets)
100000: Montgomery parameter computation then ECC scalar multiplication
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
100011: ECC complete addition
100111: ECC double base ladder
101111: ECC projective to affine

When an operation not listed here is written by the application with EN bit set, OPERRF bit is set in PKA_SR register, and the write to MODE bitfield is ignored.

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.

When an illegal operation is selected while START bit is set no operation is started, and OPERRF bit is set in PKA_SR.

Note: START is ignored if PKA is busy.

Bit 0 **EN**: PKA enable.

0: Disable PKA

1: Enable PKA.PKA becomes functional when INITOK is set by hardware in PKA_SR. When an illegal operation is selected while EN=1 OPERRF bit is set in PKA_SR. See PKA_CR.MODE bitfield for details.

Note: When EN=0 PKA RAM can still be accessed by the application.

45.7.2 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.	OP ERRF	ADDR ERRF	RAM ERRF	Res.	PROC ENDF	BUSY
										r	r	r		r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INITOK
															r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **OPERRF**: Operation error flag

0: No event error

1: An illegal or unknown operation has been selected in PKA_CR register

This bit is cleared using OPERRFC bit in PKA_CLRFR.

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, it 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 15:1 Reserved, must be kept at reset value.

Bit 0 **INITOK**: PKA initialization OK

This bit is asserted when PKA initialization is complete. When RNG is not able to output proper random numbers INITOK stays at 0.

0: PKA is not initialized correctly. START bit cannot be set.

1: PKA is initialized correctly and can be used normally.

45.7.3 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.	OP ERRFC	ADDR ERRFC	RAM ERRFC	Res.	PROC ENDFC	Res.
										w	w	w		w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **OPERRFC**: Clear operation error flag
- 0: No action
- 1: Clear the OPERRF flag in PKA_SR

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.

45.7.4 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 5336 bytes (max word offset: 0x14D0)

Note: PKA RAM cannot be used just after a PKA reset or a product reset, as described in Section 45.3.3: PKA reset and clocks.

45.7.5 PKA register map

Table 423. 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.	OPERRIE	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	0
0x004	PKA_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRF	ADDRERRF	RAMERRF	Res.	PROCENDF	BUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INITOK
	Reset value											0	0	0		0	0																0
0x008	PKA_CLRFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRFC	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		0																	

Refer to *Section 2.3 on page 115* for the register boundary addresses.

46 Advanced-control timers (TIM1/TIM8)

46.1 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 46.3.30: Timer synchronization*.

46.2 TIM1/TIM8 main features

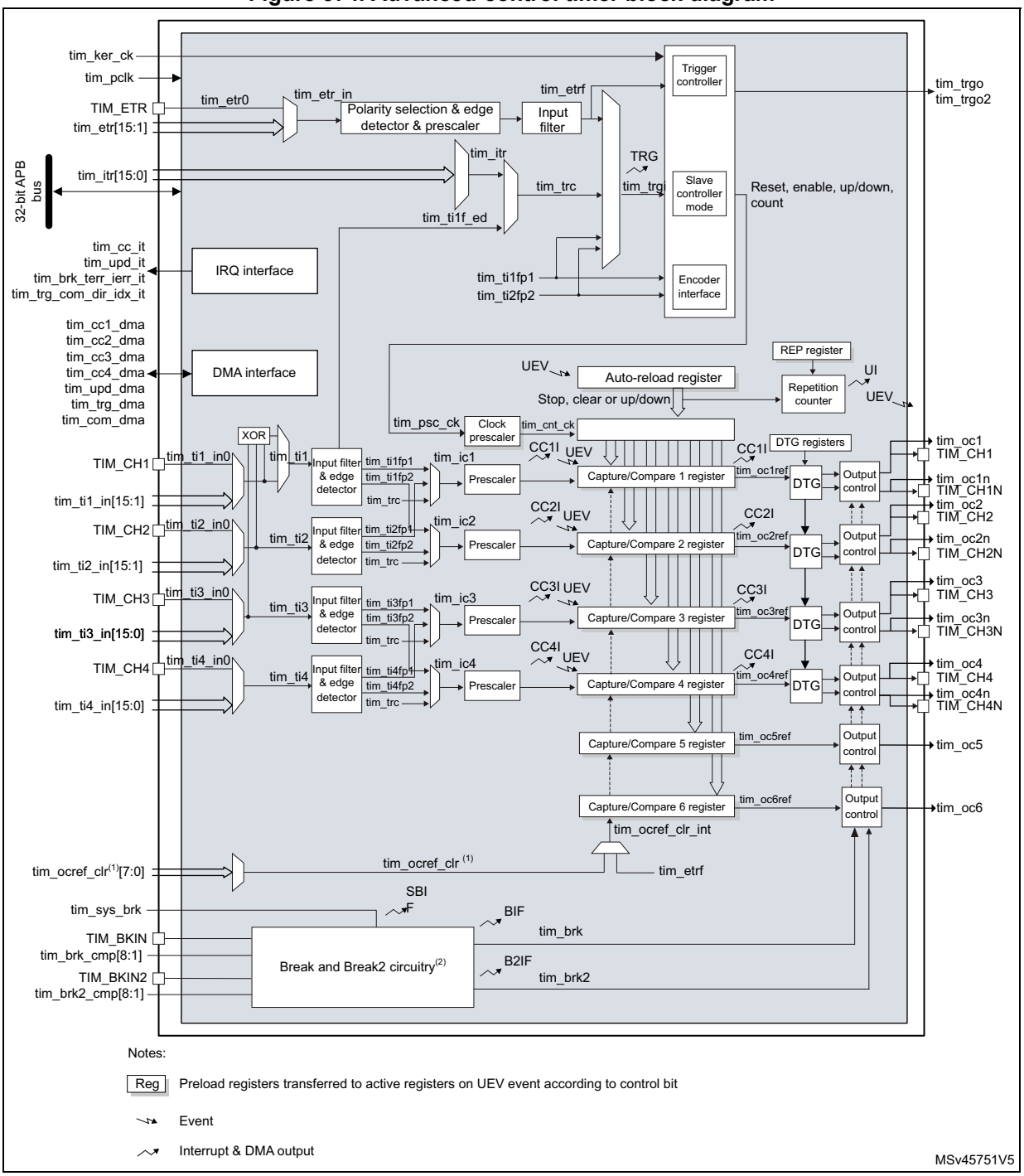
TIM1/TIM8 timer features include:

- 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

46.3 TIM1/TIM8 functional description

46.3.1 Block diagram

Figure 374. Advanced-control timer block diagram



1. This feature is not available on all timers, refer to *Section 46.3.2: TIM1/TIM8 pins and internal signals*.
2. See *Figure 421: Break and Break2 circuitry overview* for details.

46.3.2 TIM1/TIM8 pins and internal signals

The tables in this section summarize the TIM inputs and outputs

Table 424. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2 TIM_CH3 TIM_CH4	Input/Output	Timer multi-purpose channels. Each channel can be used for capture, compare or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock), external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.
TIM_CH1N TIM_CH2N TIM_CH3N TIM_CH4N	Output	Timer complementary outputs, derived from TIM_CHx outputs with the possibility to have deadtime insertion.
TIM_ETR	Input	External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
TIM_BKIN TIM_BKIN2	Input / Output	Break and Break2 inputs. These inputs can also be configured in bidirectional mode.

Table 425. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0] tim_ti3_in[15:0] tim_ti4_in[15:0]	Input	Internal timer inputs bus. The tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_etr[15:0]	Input	External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulsewidth control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
tim_itr[15:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo/tim_trgo2	Output	Internal trigger outputs. These triggers are used by other timers and /or other peripherals.

Table 425. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_ocref_clr[7:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.
tim_brk_cmp[8:1]	Input	Break input for internal signals
tim_brk2_cmp[8:1]	Input	Break2 input for internal signals
tim_sys_brk[n:0]	Input	System break input. This input gathers the MCU's system level errors.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock
tim_cc_it	Output	Timer capture/compare interrupt
tim_upd_it	Output	Timer update event interrupt
tim_brk_terr_ierr_it	Output	Timer break, break2, transition error and index error interrupt
tim_trg_com_dir_idx_it	Output	Timer trigger, commutation, direction and index interrupt
tim_cc1_dma tim_cc2_dma tim_cc3_dma tim_cc4_dma	Output	Timer capture / compare 1..4 dma requests
tim_upd_dma	Output	Timer update dma request
tim_trg_dma	Output	Timer trigger dma request
tim_com_dma	Output	Timer commutation dma request

Table 426, *Table 427*, *Table 428* and *Table 429* list the sources connected to the tim_ti[4:1] input multiplexers.

Table 426. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources	
	TIM1	TIM8
tim_ti1_in0	TIM1_CH1	TIM8_CH1
tim_ti1_in1	comp1_out	comp1_out
tim_ti1_in2	comp2_out	comp2_out
tim_ti1_in[15:3]	Reserved	

Table 427. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources	
	TIM1	TIM8
tim_ti2_in0	TIM1_CH2	TIM8_CH2
tim_ti2_in[15:1]	Reserved	

Table 428. Interconnect to the tim_ti3 input multiplexer

tim_ti3 inputs	Sources	
	TIM1	TIM8
tim_ti3_in0	TIM1_CH3	TIM8_CH3
tim_ti2_in[15:1]	Reserved	

Table 429. Interconnect to the tim_ti4 input multiplexer

tim_ti4 inputs	Sources	
	TIM1	TIM8
tim_ti4_in0	TIM1_CH4	TIM8_CH4
tim_ti4_in[15:1]	Reserved	

Table 430 lists the internal sources connected to the tim_etr input multiplexer.

Table 430. TIMx internal trigger connection

TIMx	TIM1	TIM8
tim_itr0	Reserved	tim1_trgo
tim_itr1	tim2_trgo	tim2_trgo
tim_itr2	tim3_trgo	tim3_trgo
tim_itr3	tim4_trgo	tim4_trgo
tim_itr4	tim5_trgo	tim5_trgo
tim_itr5	tim8_trgo	Reserved
tim_itr6	tim15_trgo	tim15_trgo
tim_itr7	tim16_oc1	tim16_oc1
tim_itr8	tim17_oc1	tim17_oc1
tim_itr[15:9]	Reserved	

Table 431 lists the internal sources connected to the tim_etr input multiplexer.

Table 431. Interconnect to the tim_etr input multiplexer

Timer external trigger input signal	Timer external trigger signals assignment	
	TIM1	TIM8
tim_etr0	TIM1_ETR	TIM8_ETR
tim_etr1	comp1_out	comp1_out
tim_etr2	comp2_out	comp2_out
tim_etr3	MSIK	MSIK
tim_etr4	HSI	HSI
tim_etr5	MSIS	MSIS
tim_etr6	Reserved	
tim_etr7		
tim_etr8	adc1_awd1	adc1_awd1
tim_etr9	adc1_awd2	adc1_awd2
tim_etr10	adc1_awd3	adc1_awd3
tim_etr11	adc4_awd1	adc4_awd1
tim_etr12	adc4_awd2	adc4_awd2
tim_etr13	adc4_awd3	adc4_awd3
tim_etr[15:14]	Reserved	

Table 432, *Table 433* and *Table 433* list the sources connected to the tim_brk and tim_brk2inputs.

Table 432. Timer break interconnect

tim_brk inputs	TIM1	TIM8
TIM_BKIN	TIM1_BKIN pin	TIM8_BKIN pin
tim_brk_cmp1	comp1_out	comp1_out
tim_brk_cmp2	comp2_out	comp2_out
tim_brk_cmp3	Reserved	
tim_brk_cmp4		
tim_brk_cmp5		
tim_brk_cmp6		
tim_brk_cmp7	mdf1_break0	mdf1_break2
tim_brk_cmp8	Reserved	

Table 433. Timer break2 interconnect

tim_brk2 inputs	TIM1	TIM8
TIM_BKIN2	TIM1_BKIN2 pin	TIM8_BKIN2 pin
tim_brk2_cmp1	comp1_out	comp1_out
tim_brk2_cmp2	comp2_out	comp2_out
tim_brk2_cmp3	Reserved	
tim_brk2_cmp4		
tim_brk2_cmp5		
tim_brk2_cmp6		
tim_brk2_cmp7	mdf1_break1	mdf1_break3
tim_brk2_cmp8	Reserved	

Table 434. System break interconnect

tim_sys_brk inputs	TIM1 / TIM8 / TIM20	Enable bit in SYSCFG_CFGR2 register
tim_sys_brk0	Cortex®-M33 LOCKUP	CLL
tim_sys_brk1	Programmable Voltage Detector (PVD)	PVDL
tim_sys_brk2	SRAM double ECC error	SPL
tim_sys_brk3	Flash double ECC error	ECCL
tim_sys_brk4	Clock Security System (CSS)	None (always enabled)

Table 435 lists the internal sources connected to the tim_ocref_clr input multiplexer.

Table 435. Interconnect to the ocref_clr input multiplexer

Timer OCREF clear signal	Timer OCREF clear signals assignment	
	TIM1	TIM8
tim_ocref_clr0	comp1_out	comp1_out
tim_ocref_clr1	comp2_out	comp2_out
tim_ocref_clr2	Reserved	
tim_ocref_clr3		
tim_ocref_clr4		
tim_ocref_clr5		
tim_ocref_clr6		
tim_ocref_clr7		

46.3.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software, even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note: The counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler divides the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 375 and *Figure 376* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 375. Counter timing diagram with prescaler division change from 1 to 2

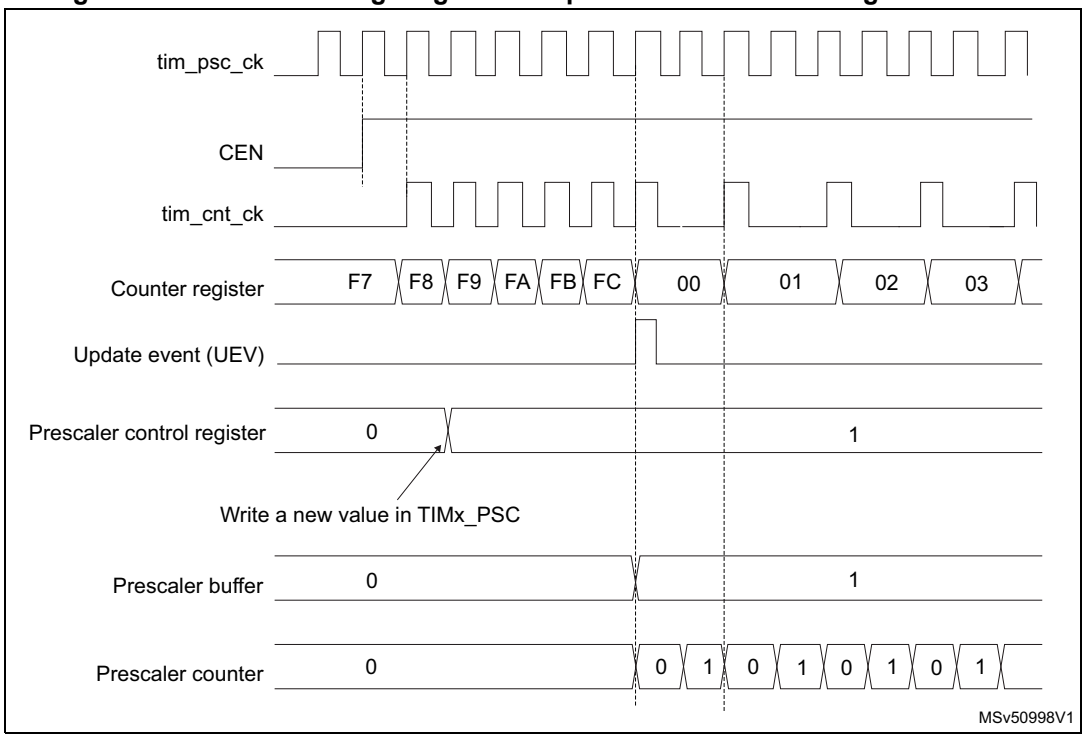
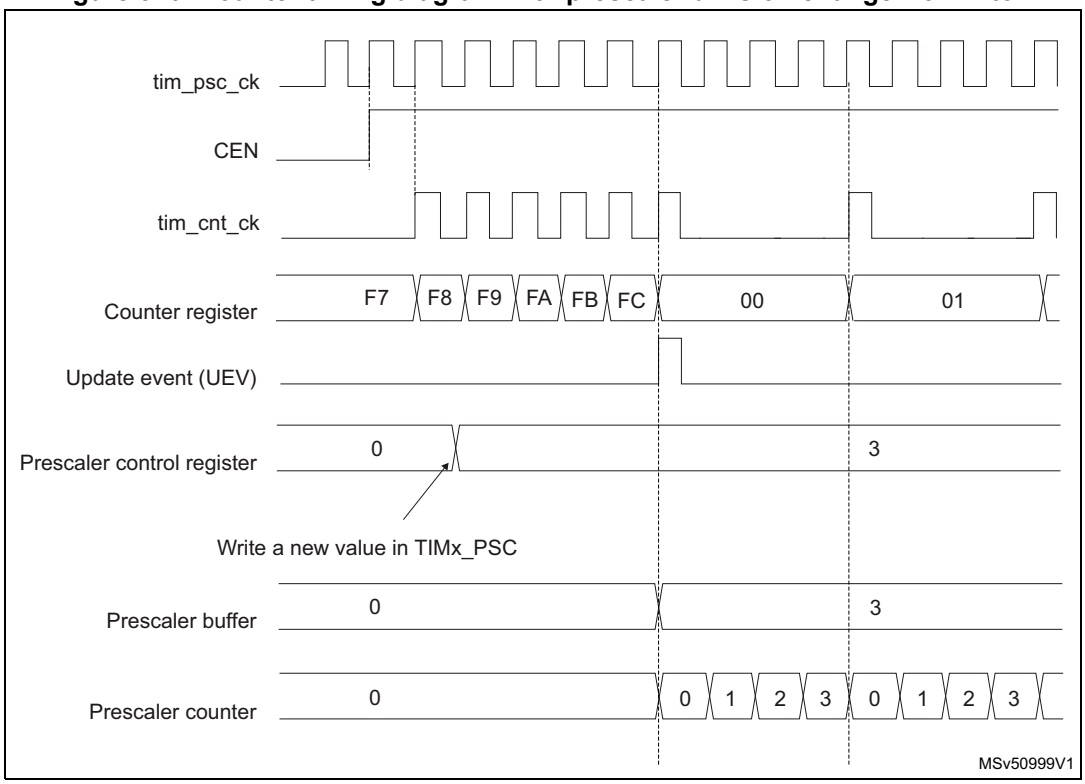


Figure 376. Counter timing diagram with prescaler division change from 1 to 4



46.3.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 377. Counter timing diagram, internal clock divided by 1

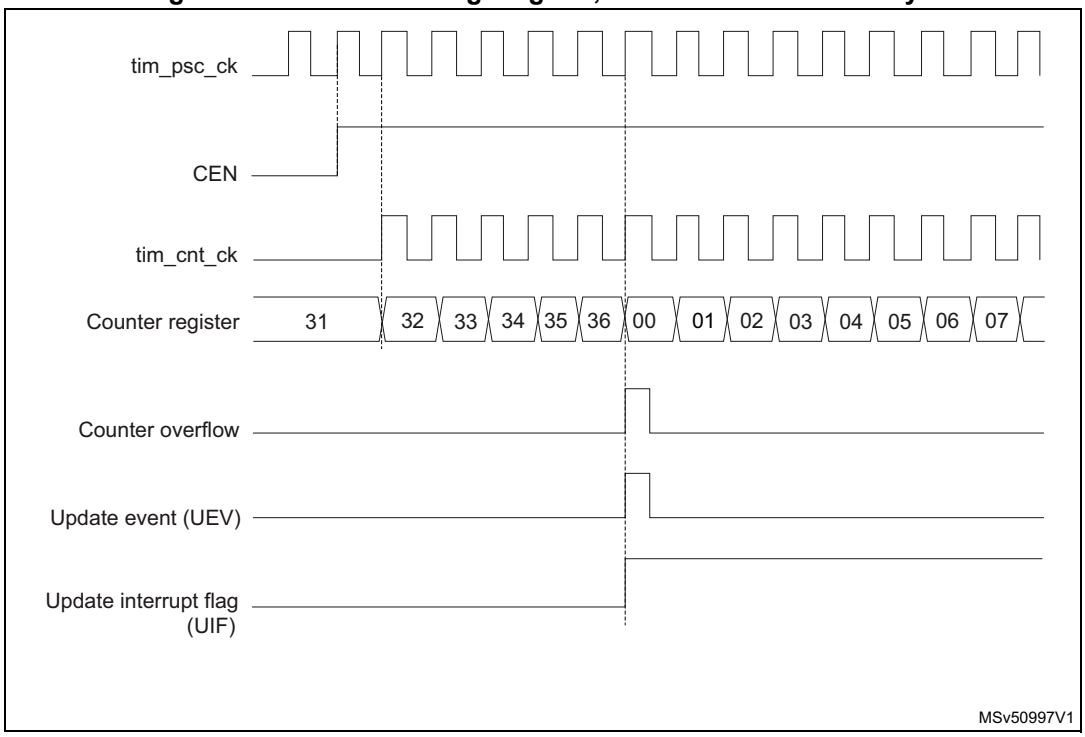


Figure 378. Counter timing diagram, internal clock divided by 2

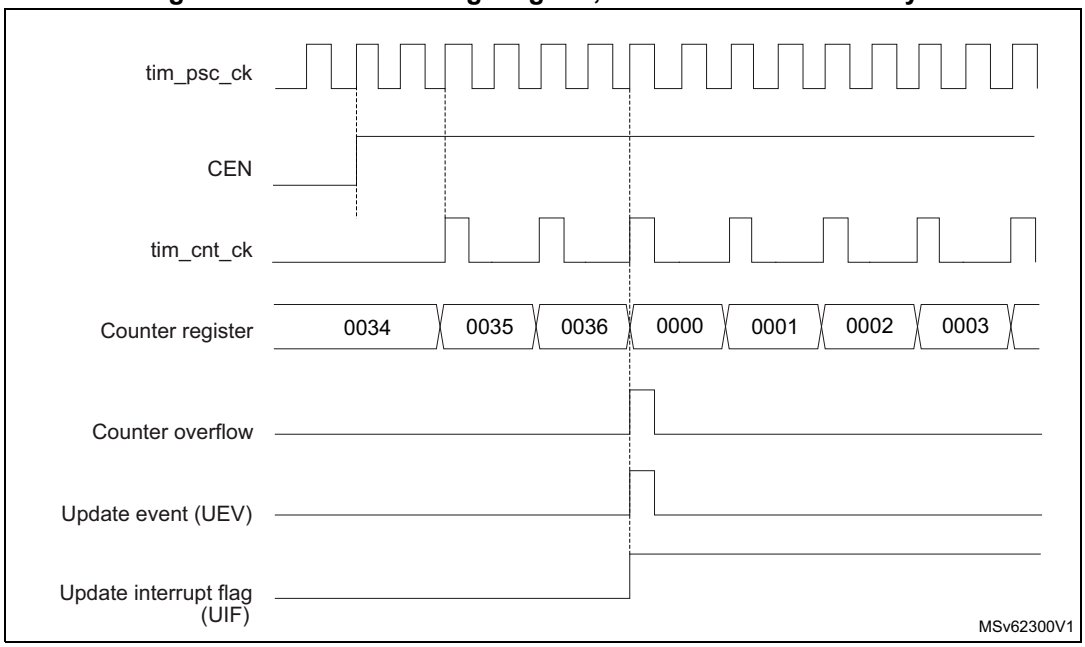


Figure 379. Counter timing diagram, internal clock divided by 4

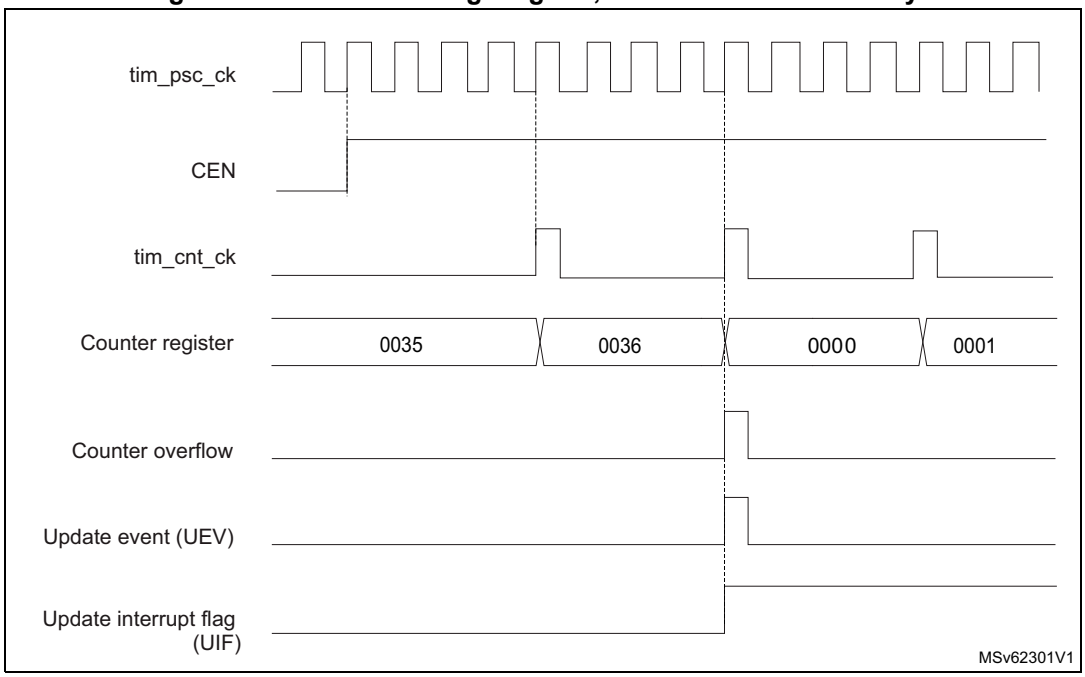


Figure 380. Counter timing diagram, internal clock divided by N

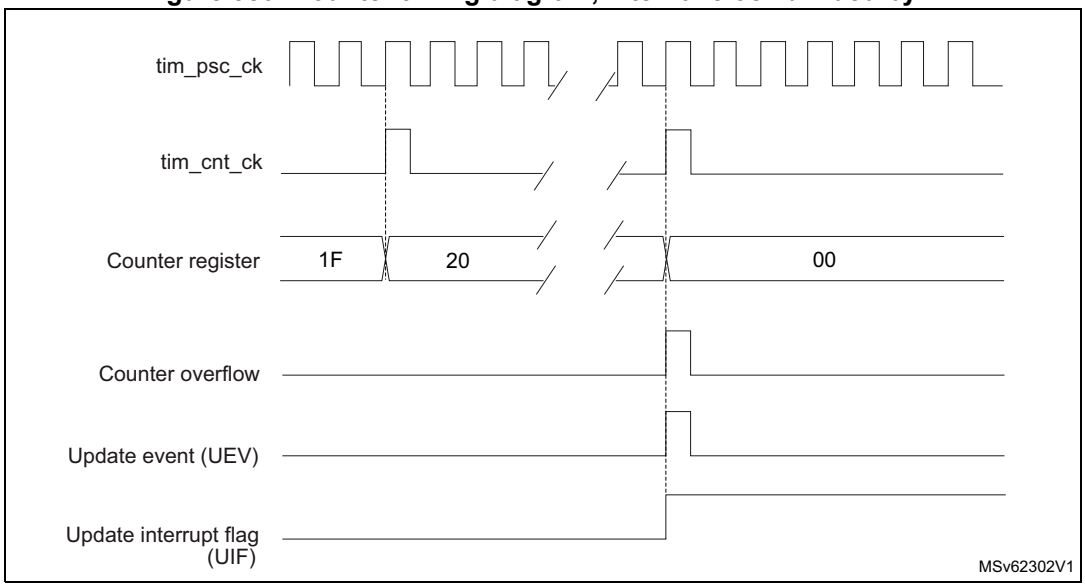


Figure 381. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

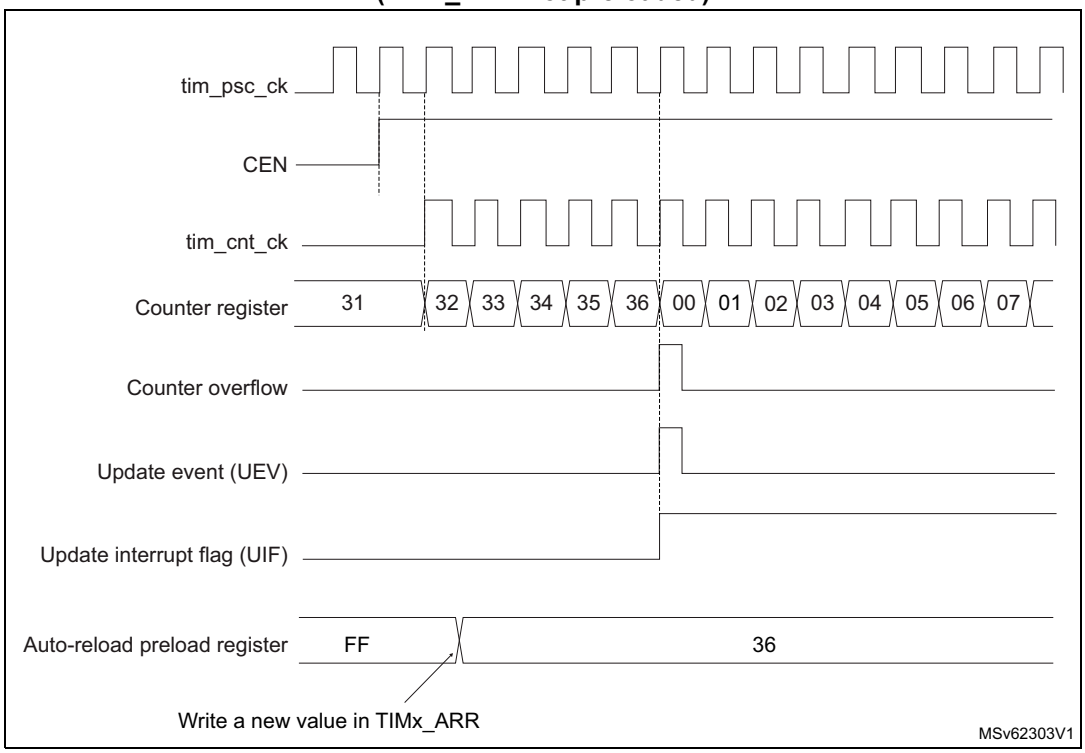
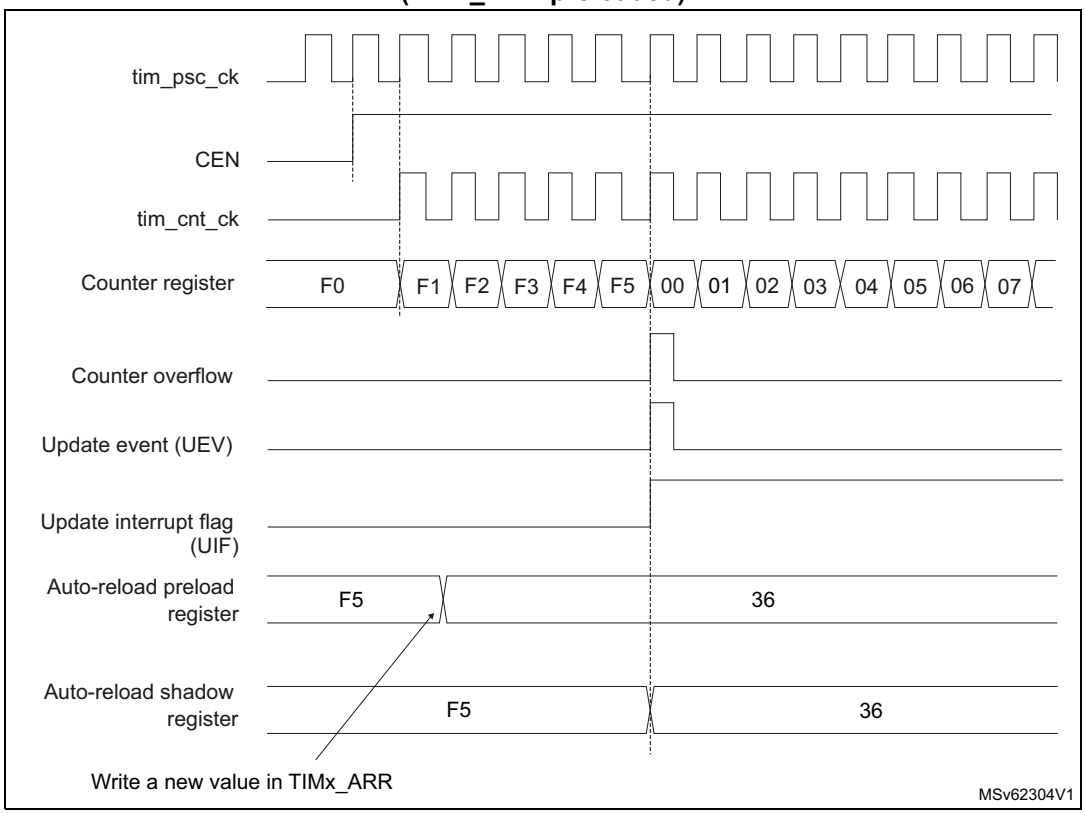


Figure 382. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



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 383. Counter timing diagram, internal clock divided by 1

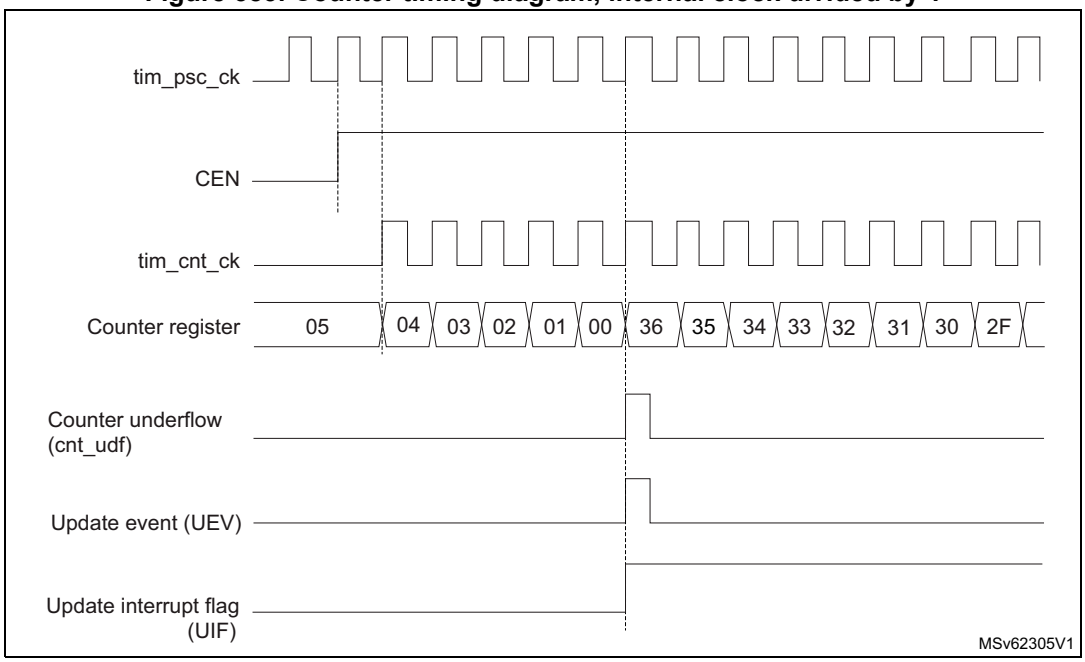


Figure 384. Counter timing diagram, internal clock divided by 2

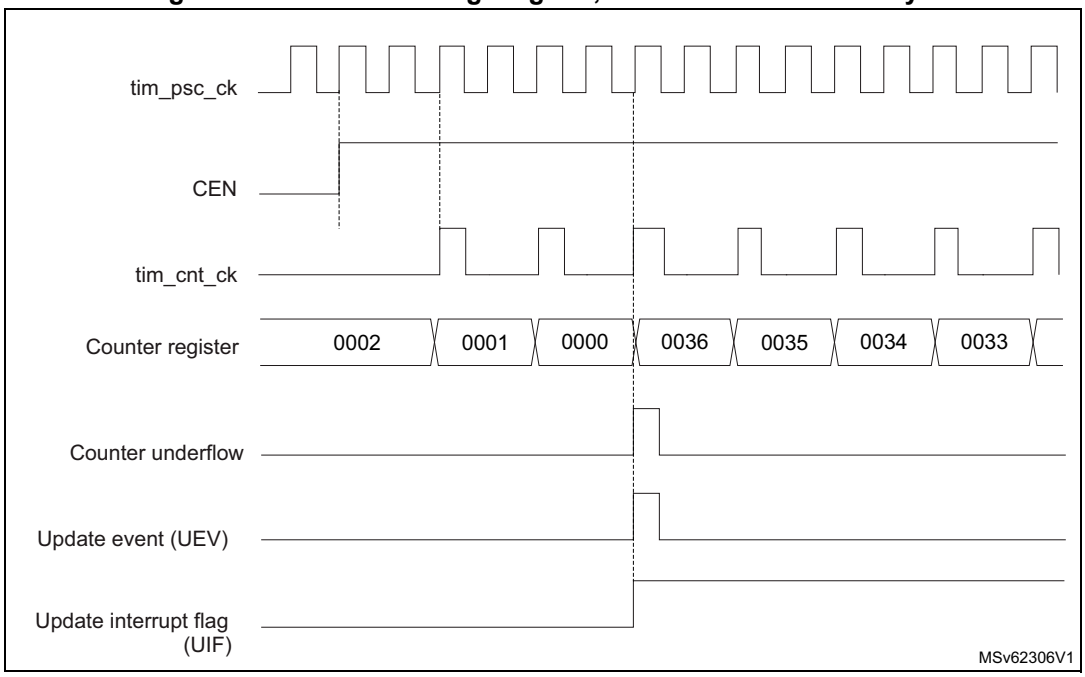


Figure 385. Counter timing diagram, internal clock divided by 4

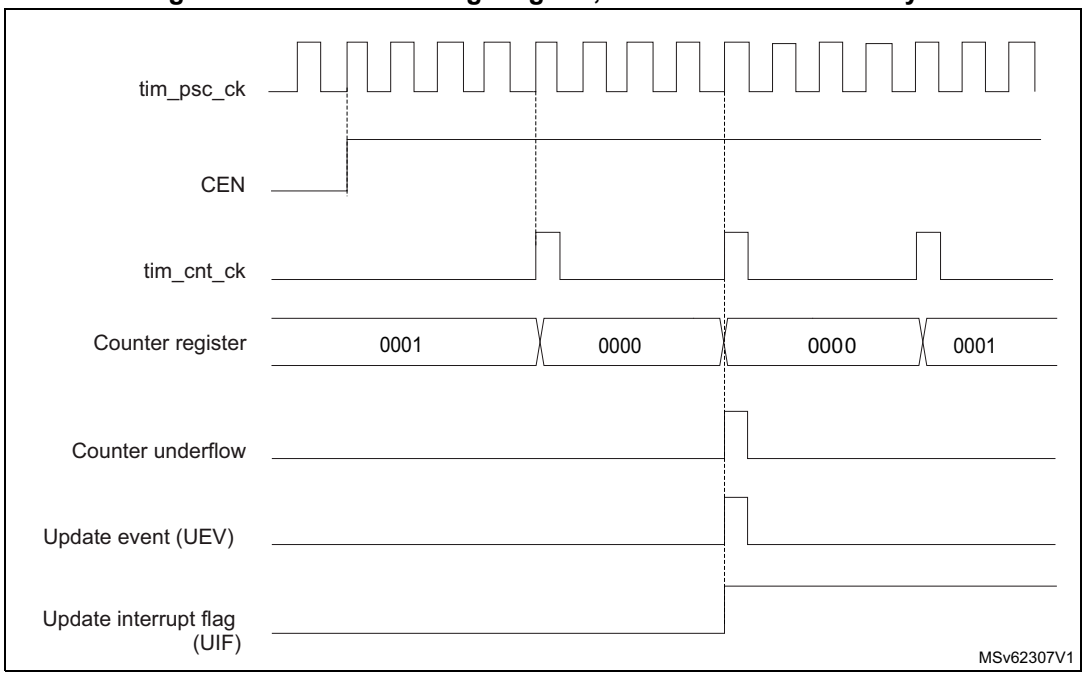


Figure 386. Counter timing diagram, internal clock divided by N

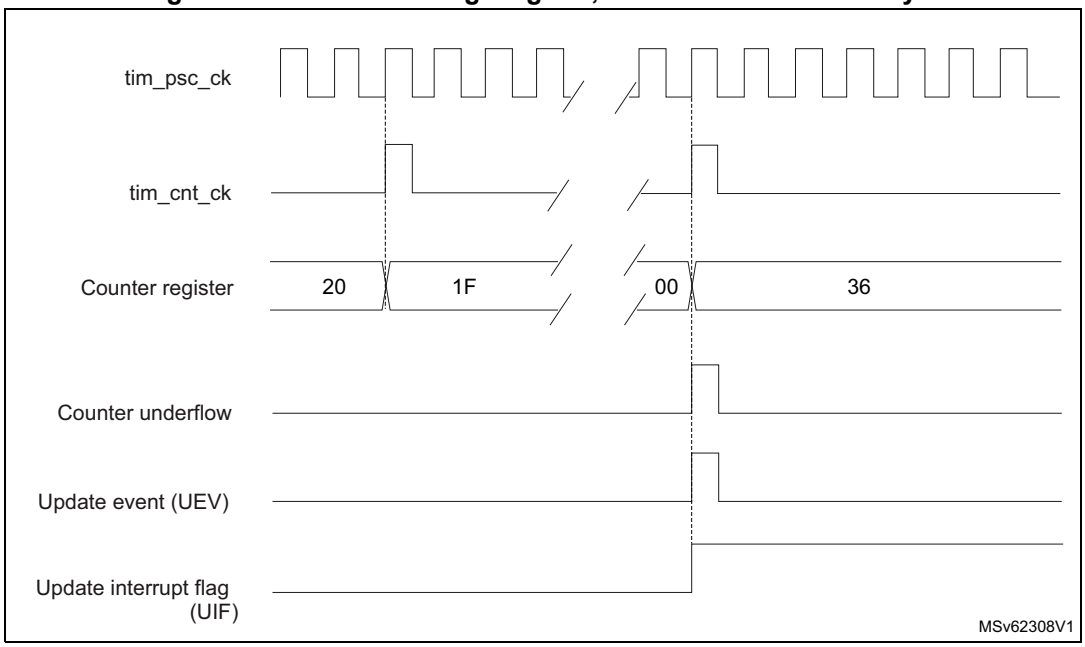
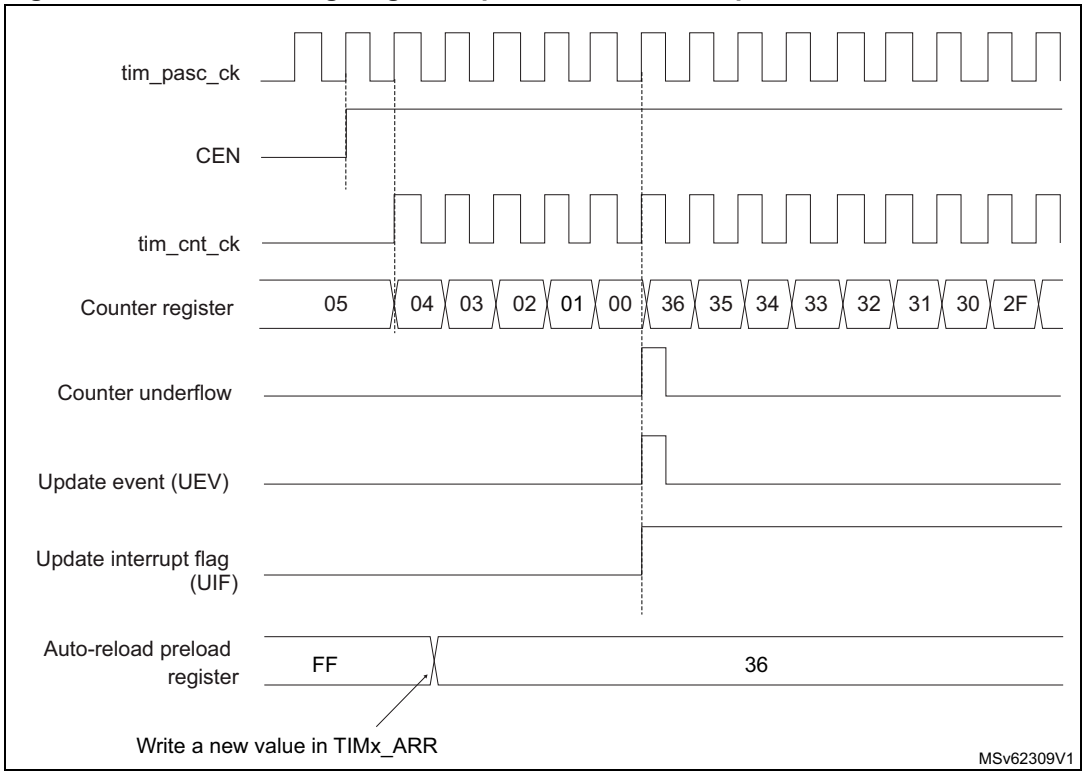


Figure 387. Counter timing diagram, update event when repetition counter is not used



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.

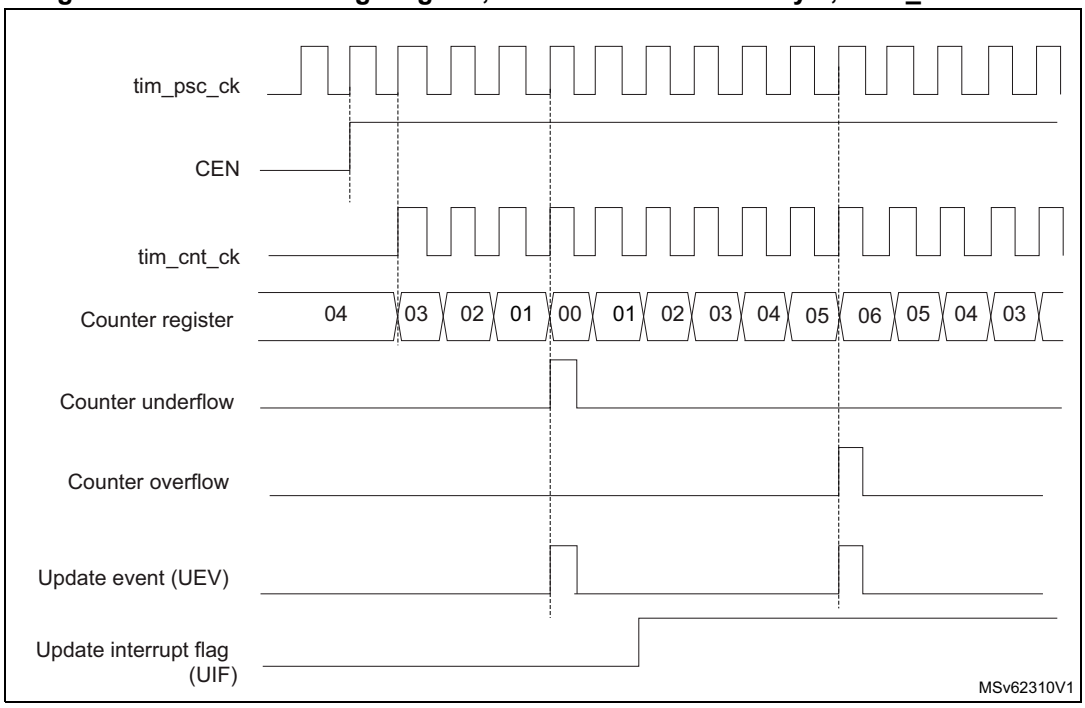
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on 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 388. 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 46.6: TIM1/TIM8 registers*).

Figure 389. Counter timing diagram, internal clock divided by 2

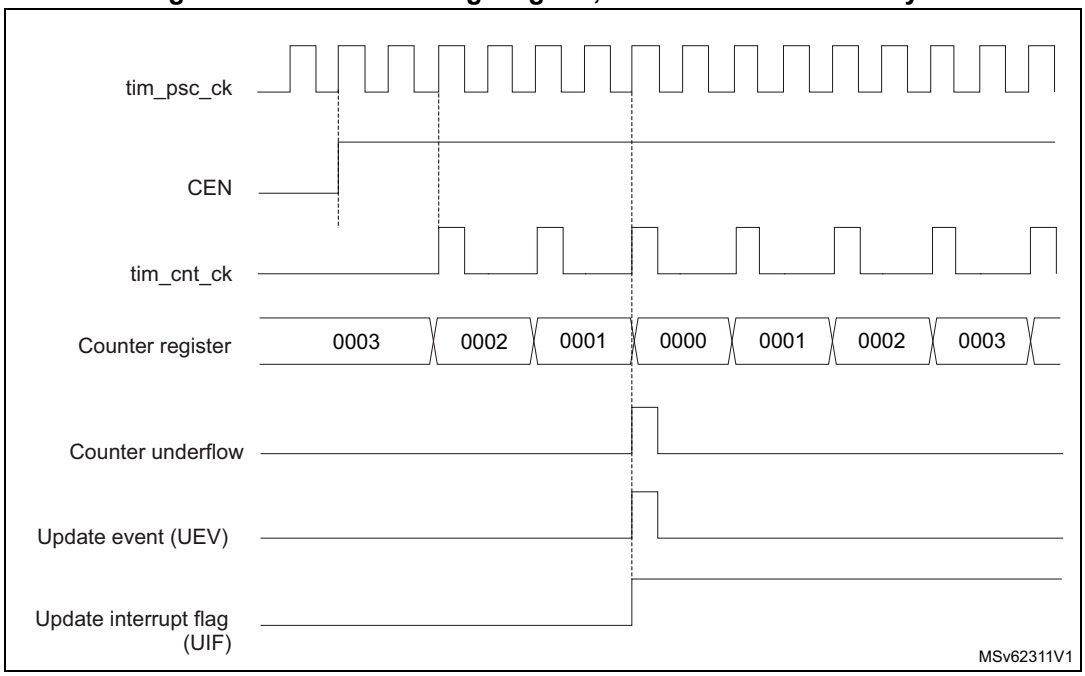


Figure 390. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36

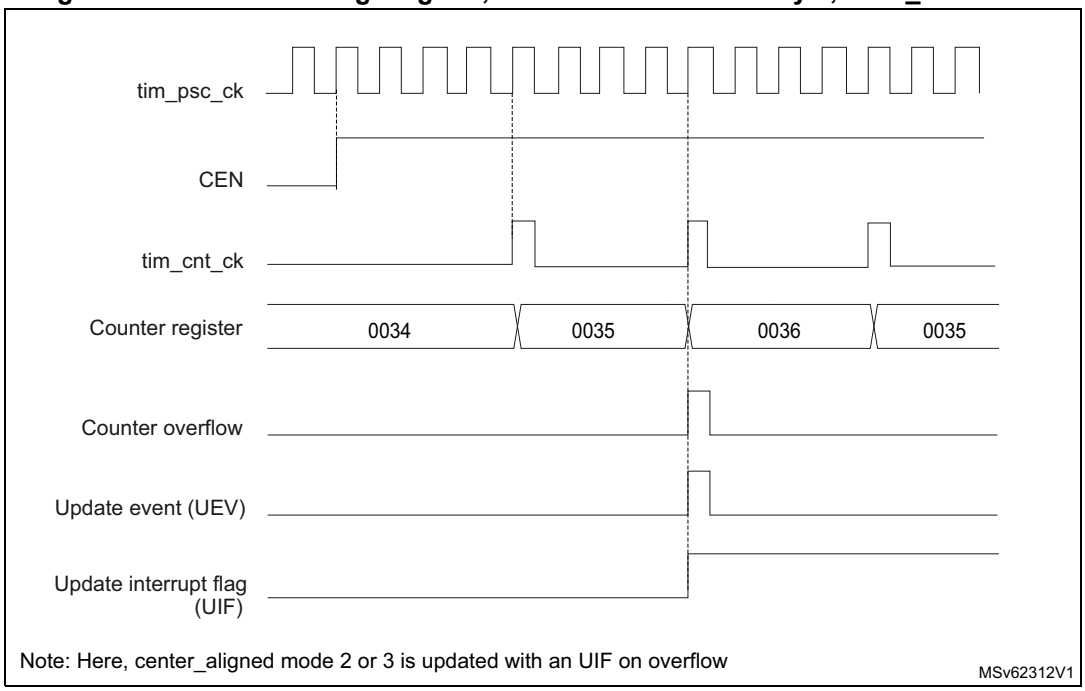


Figure 391. Counter timing diagram, internal clock divided by N

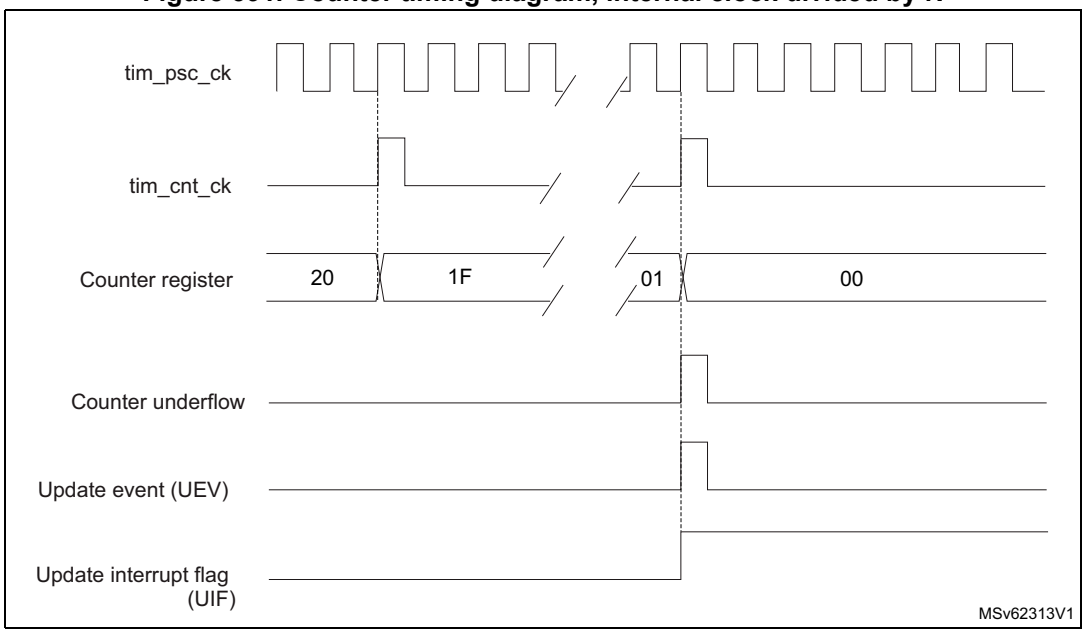


Figure 392. Counter timing diagram, update event with ARPE=1 (counter underflow)

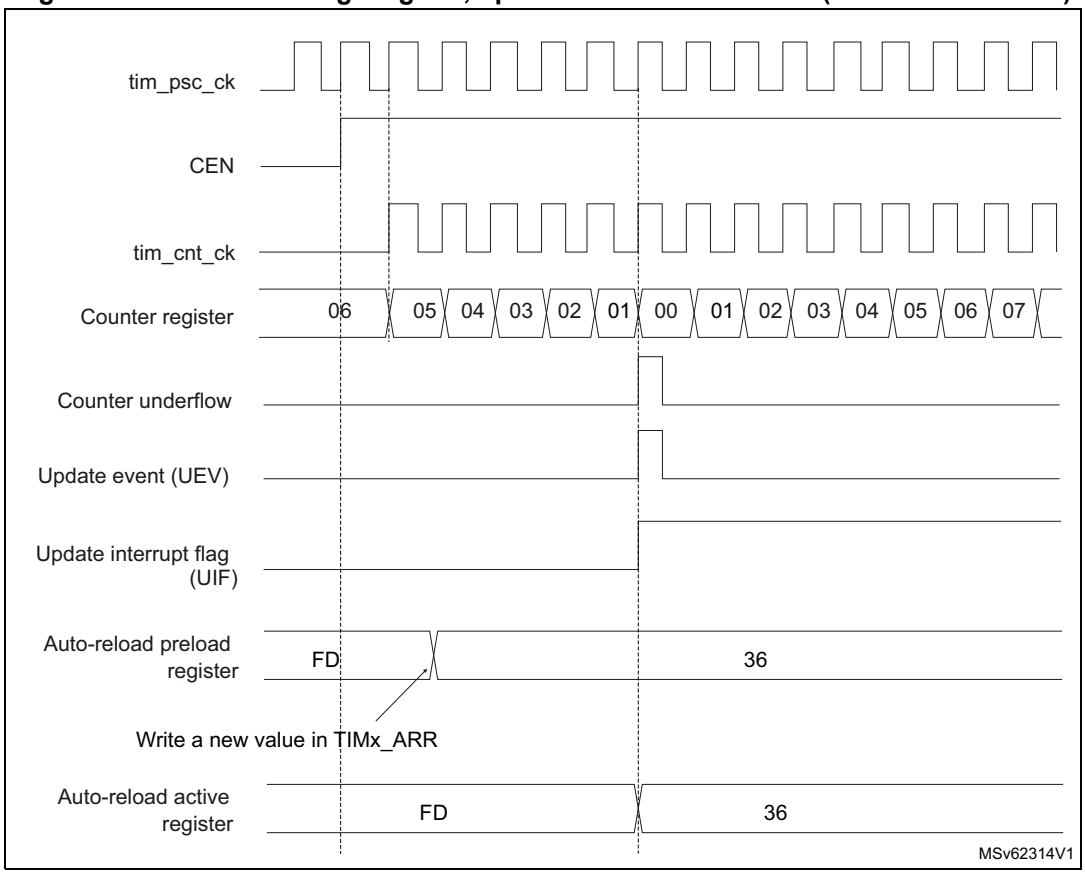
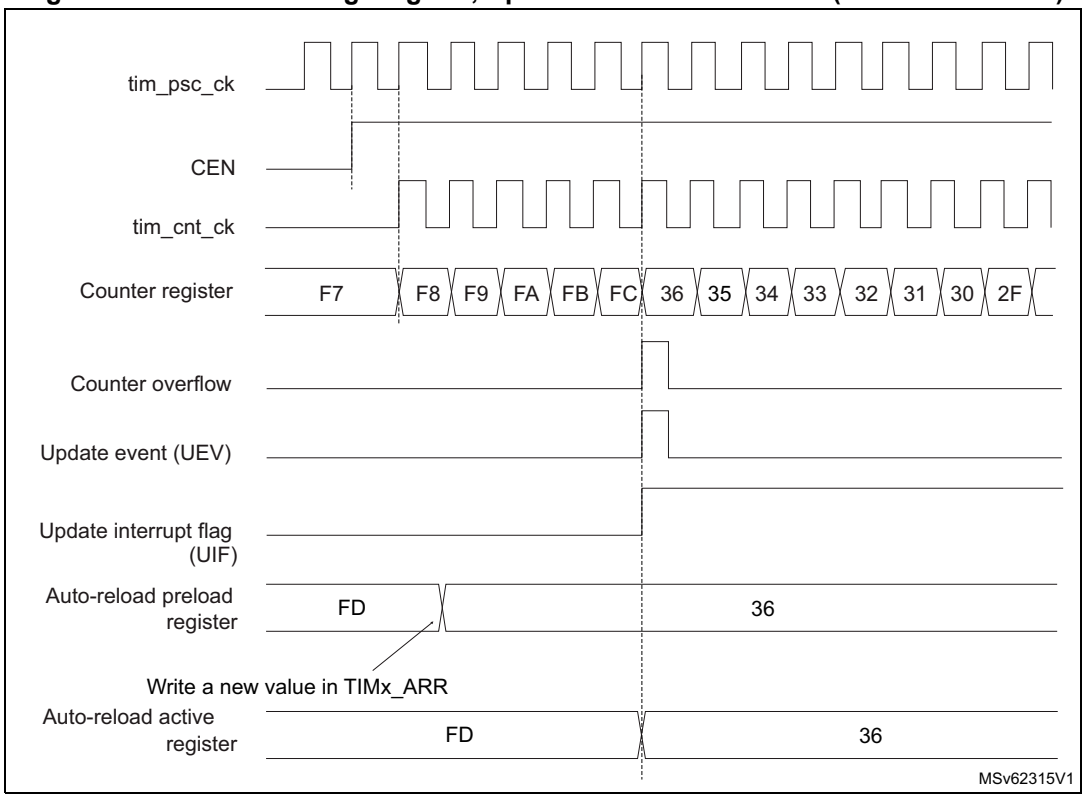


Figure 393. Counter timing diagram, Update event with ARPE=1 (counter overflow)



46.3.5 Repetition counter

Section 46.3.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented:

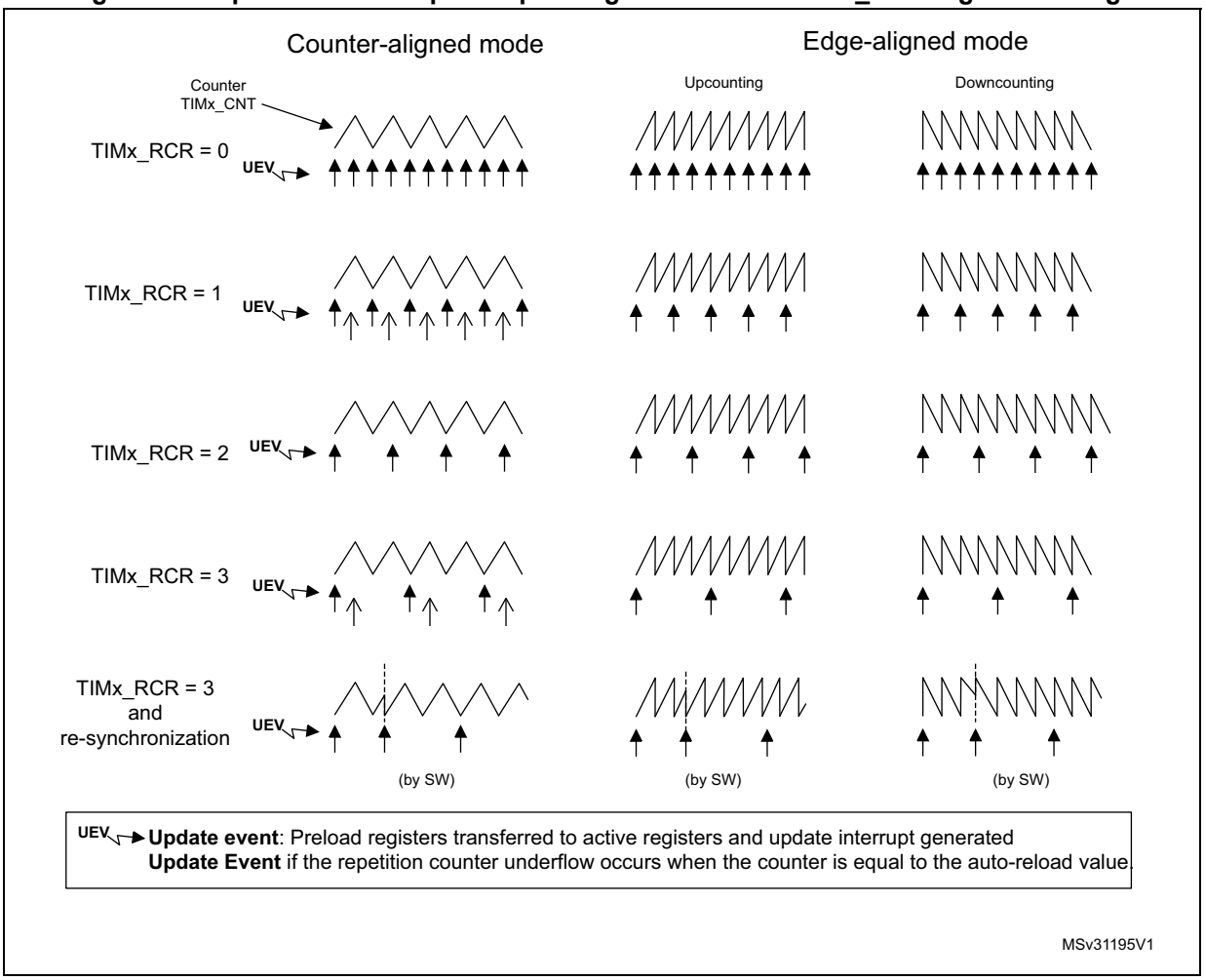
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2xT_{ck}$, due to the symmetry of the pattern.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 394*). 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 394. Update rate examples depending on mode and TIMx_RCR register settings



46.3.6 External trigger input

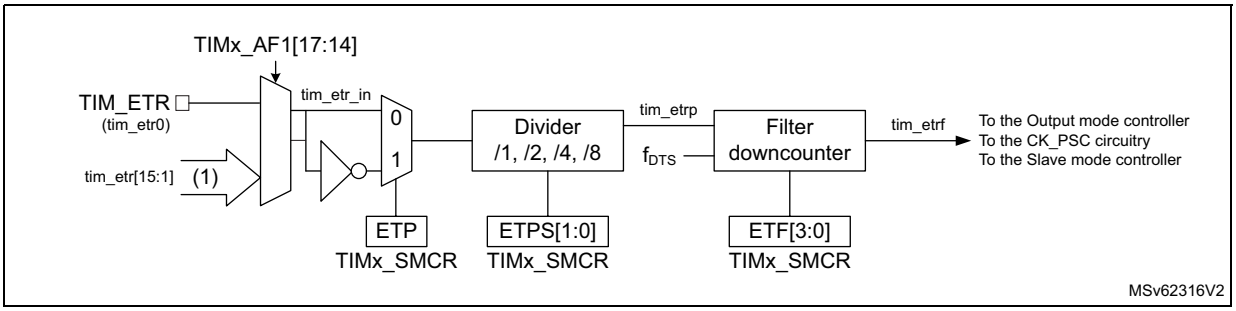
The timer features an external trigger input tim_etr_in. It can be used as:

- external clock (external clock mode 2, see *Section 46.3.7*)
- trigger for the slave mode (see *Section 46.3.30*)
- PWM reset input for cycle-by-cycle current regulation (see *Section 46.3.9*)

Figure 395 below describes the tim_etr_in input conditioning. The input polarity is defined with the ETP bit in TIMxSMCR register. The trigger can be prescaled with the divider programmed by the ETPS[1:0] bitfield and digitally filtered with the ETF[3:0] bitfield. The resulting signal (tim_etrf) is available for three purposes: as an external clock, to condition

the output (typically to reset a PWM output for a current limitation), and as a trigger for the Slave mode controller.

Figure 395. External trigger input block



The tim_etr_in input comes from multiple sources: input pins (default configuration), or internal sources. The selection is done with the ETRSEL[3:0] bitfield in the TIMx_AF1 register.

Refer to *Section 46.3.2: TIM1/TIM8 pins and internal signals* for the list of sources connected to the etr_in input in the product.

46.3.7 Clock selection

The counter clock can be provided by the following clock sources:

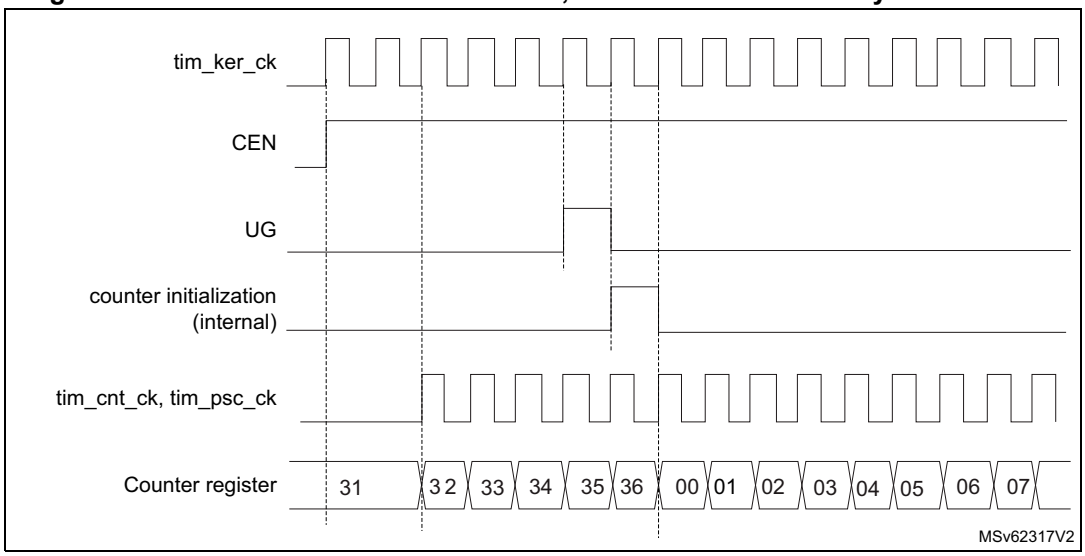
- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2)
- External clock mode2: external trigger input (tim_etr_in)
- Encoder mode

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS=000), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 396 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

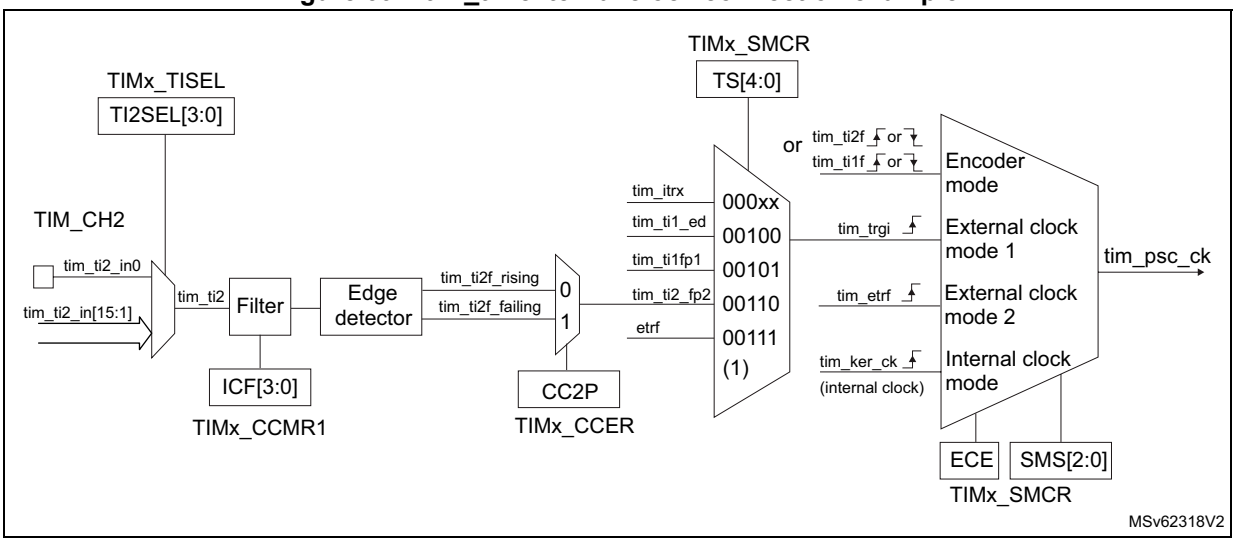
Figure 396. Control circuit in normal mode, internal clock divided by 1



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 397. tim_ti2 external clock connection example



1. Codes ranging from 01000 to 11111 are reserved.

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

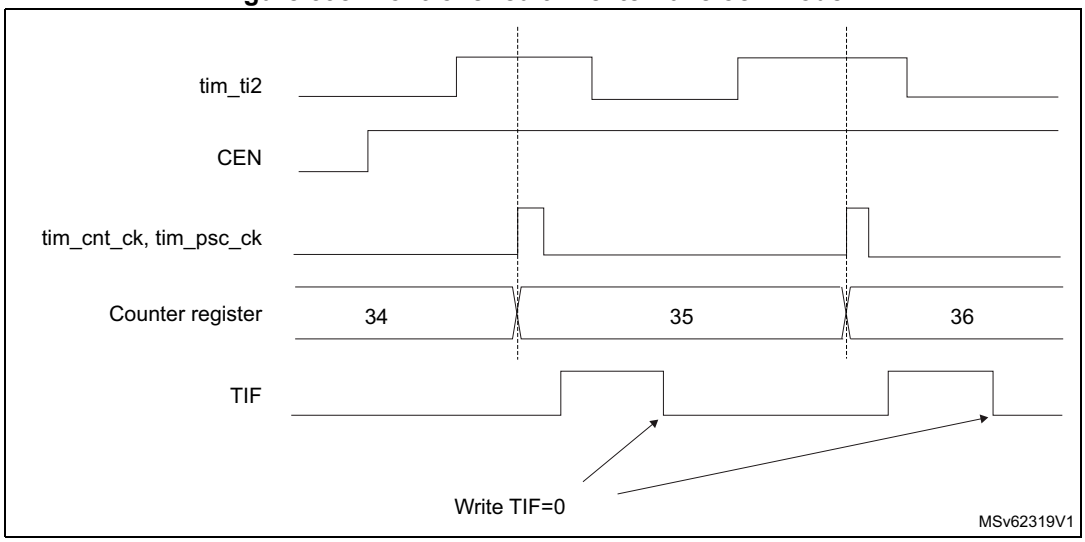
1. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
3. Select rising edge polarity by writing CC2P=0 and CC2NP=0 in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
5. Select tim_ti2 as the trigger input source by writing TS=00110 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 398. Control circuit in external clock mode 1



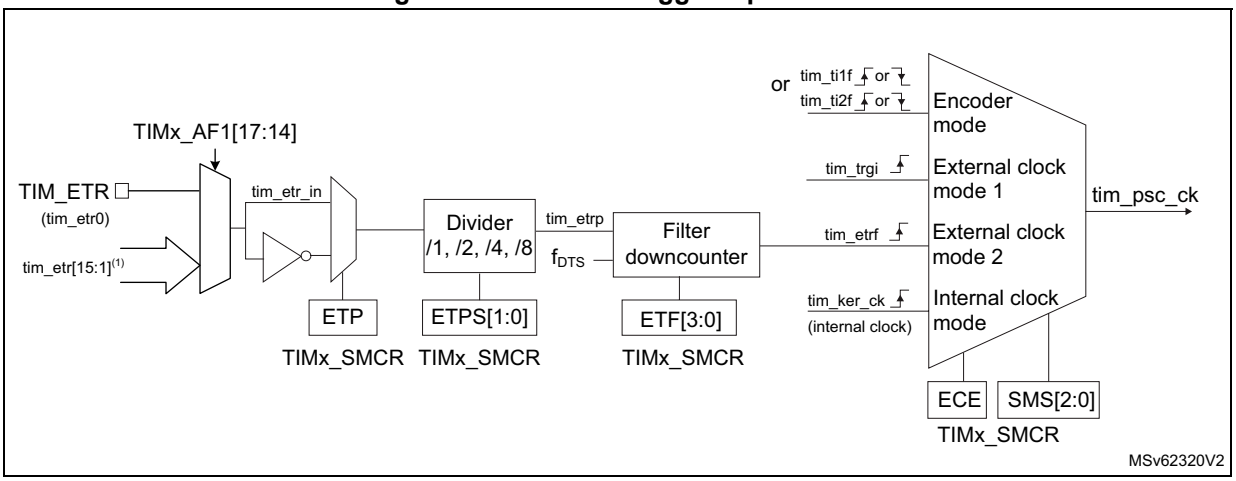
External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter counts at each rising or falling edge on the external trigger input tim_etr_in.

The *Figure 399* gives an overview of the external trigger input block.

Figure 399. External trigger input block



1. Refer to *Section 46.3.2: TIM1/TIM8 pins and internal signals*.

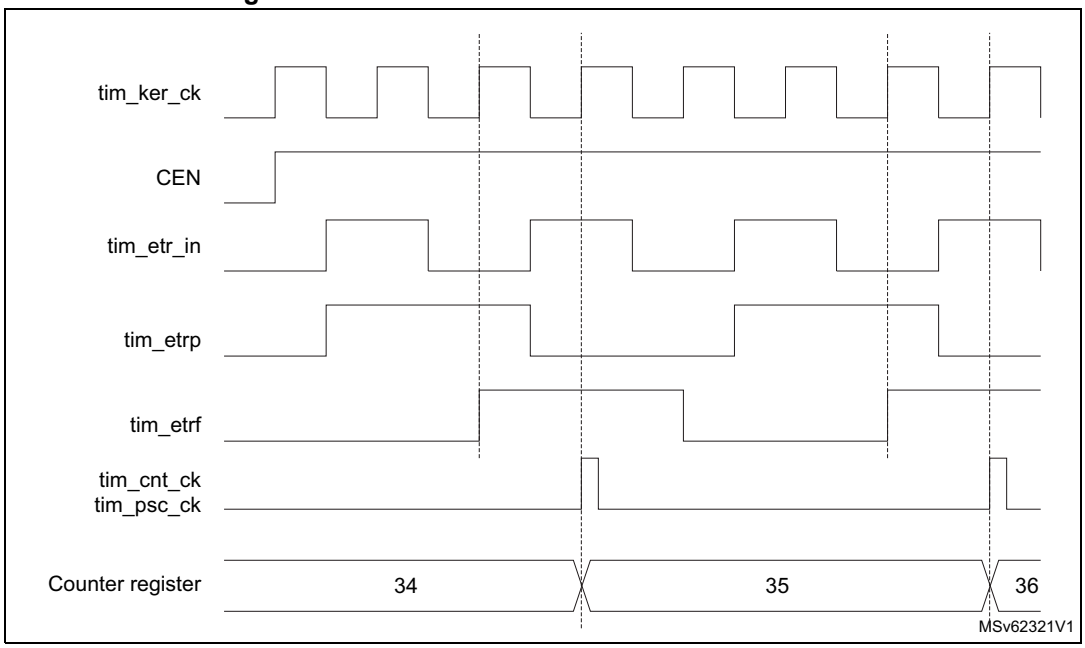
For example, to configure the upcounter to count each 2 rising edges on tim_etr_in, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
2. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register
3. Select rising edge detection on the tim_etr_in input by writing ETP=0 in the TIMx_SMCR register
4. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
5. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 tim_etr_in rising edges.

The delay between the rising edge on tim_etr_in and the actual clock of the counter is due to the resynchronization circuit on the tim_etrp signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most ¼ of tim_ker_ck frequency. When the ETRP signal is faster, the user should apply a division of the external signal by a proper ETPS prescaler setting.

Figure 400. Control circuit in external clock mode 2



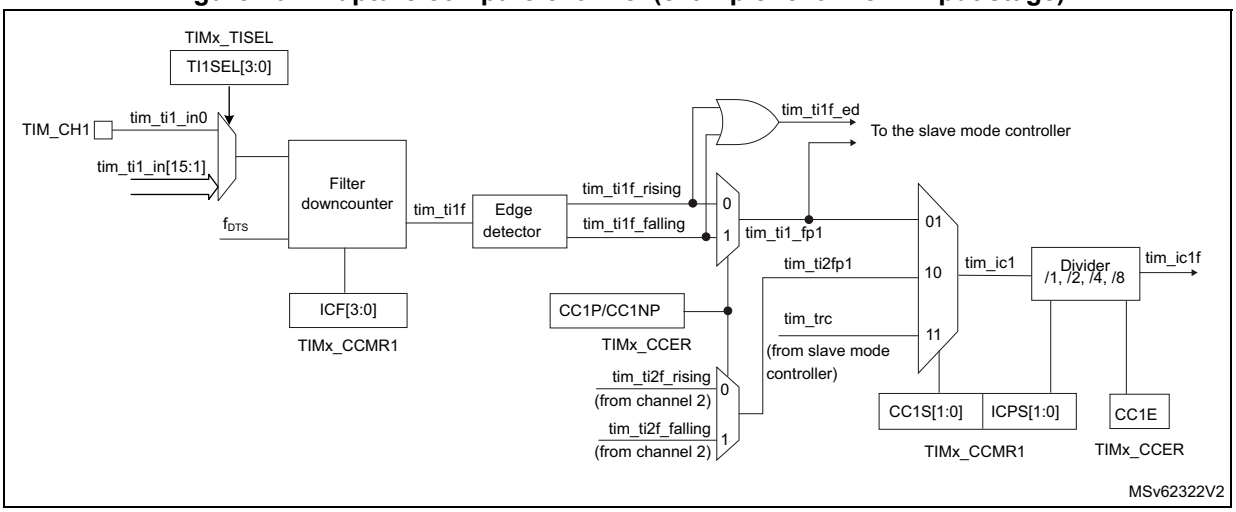
46.3.8 Capture/compare channels

Each capture/compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing, and prescaler, except for channels 5 and 6) and an output stage (with comparator and output control).

Figure 401 to *Figure 404* give an overview of one capture/compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 401. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 402. Capture/compare channel 1 main circuit

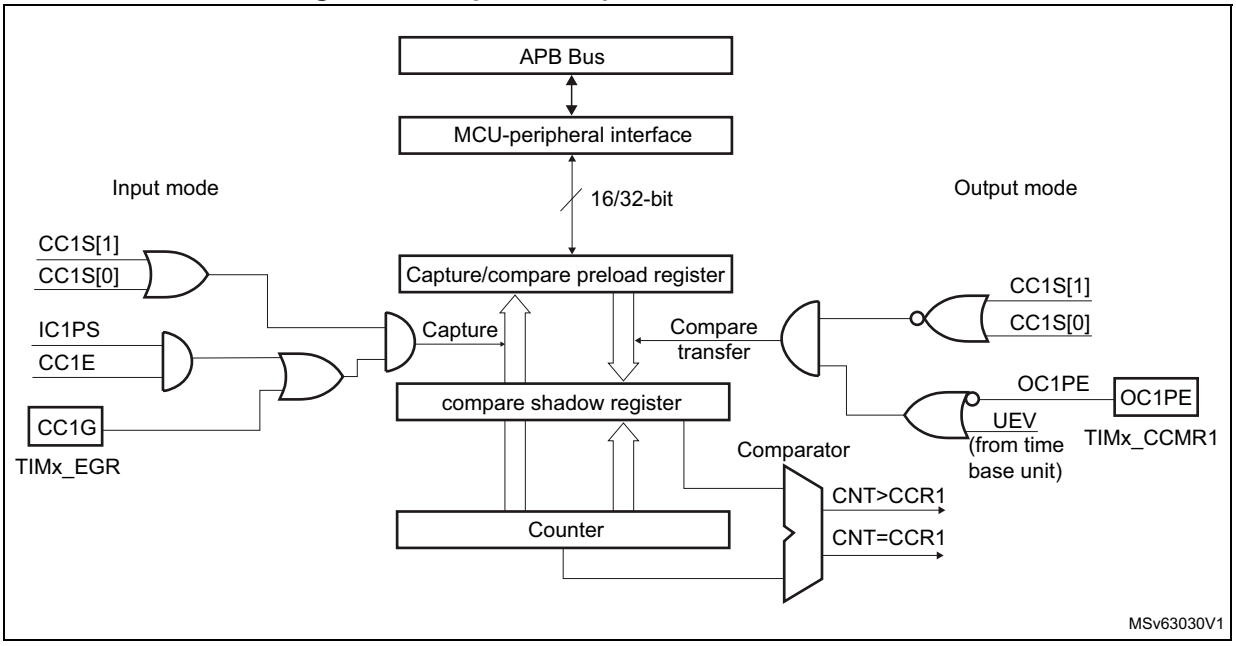
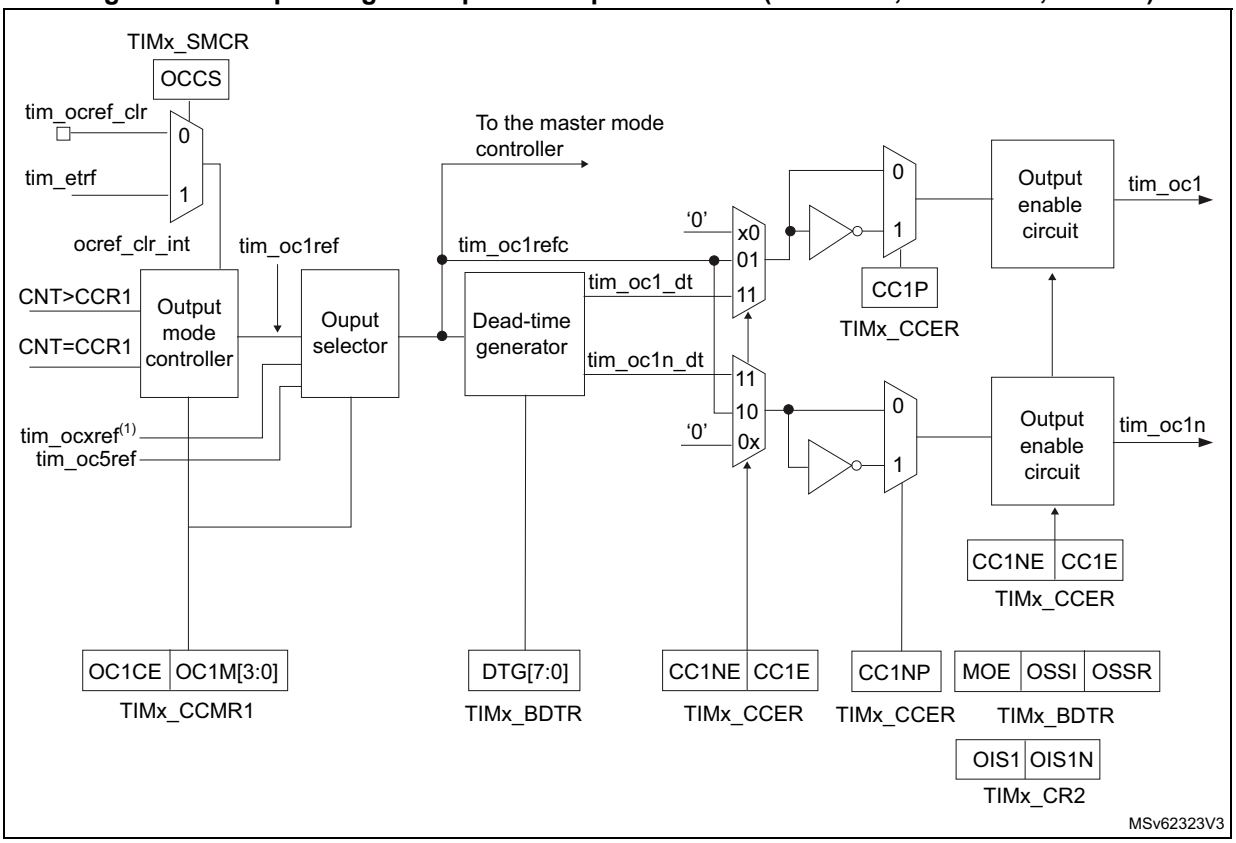
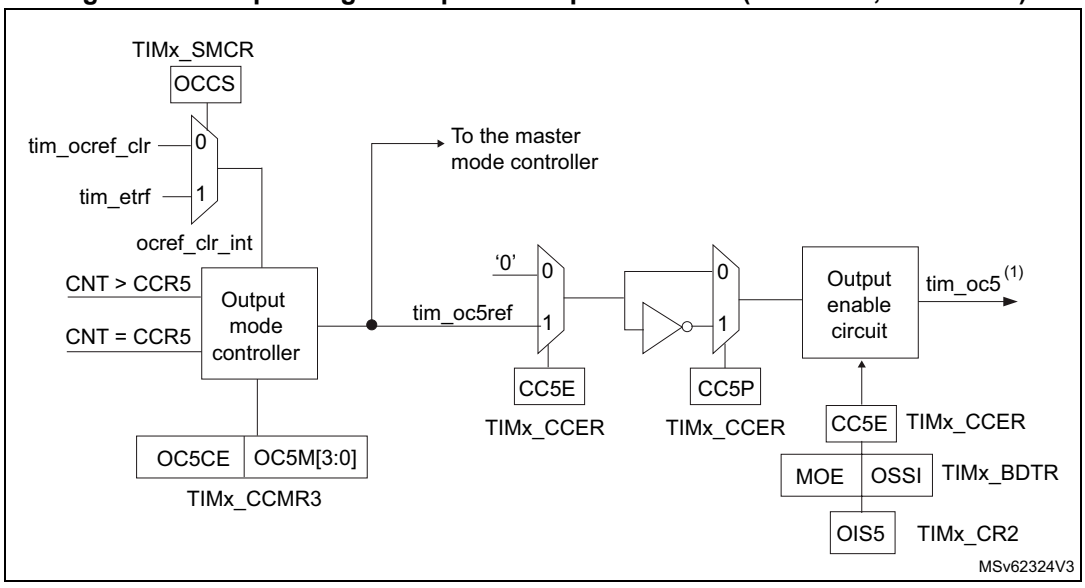


Figure 403. Output stage of capture/compare channel (channel 1, idem ch. 2, 3 and 4)



1. tim_ocxref, where x is the rank of the complementary channel

Figure 404. 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.

46.3.9 Input capture mode

In Input capture mode, the capture/compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

- Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
- Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on tim_ti1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
- Select the edge of the active transition on the tim_ti1 channel by writing CC1P and CC1NP bits to 0 in the TIMx_CCER register (rising edge in this case).
- Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx_CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

46.3.10 PWM input mode

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges)
- The TIM_CCR2 register holds the pulsewidth (interval between two consecutive rising and falling edges

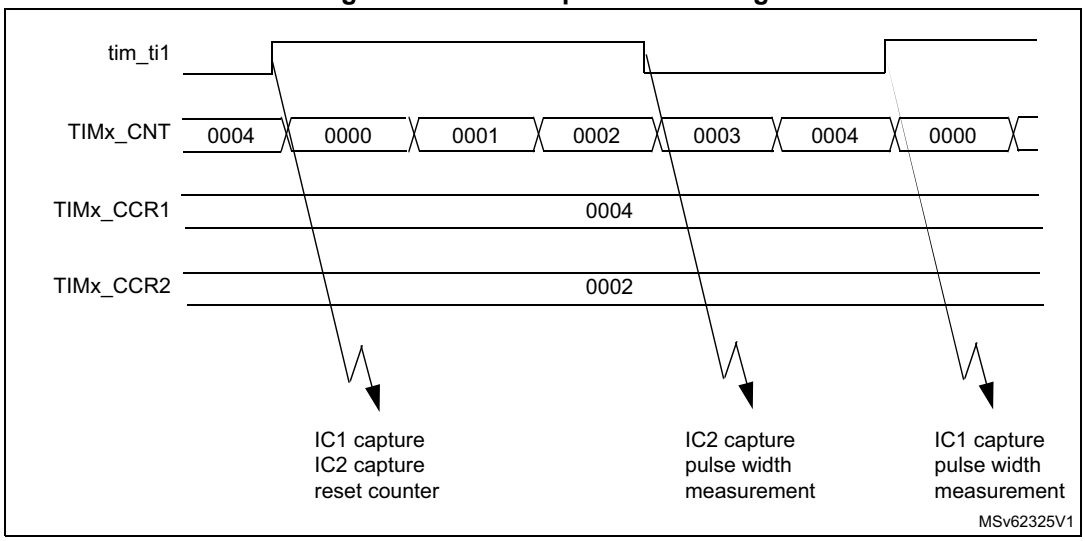
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tixfp1 input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two tim_tixfp signals is selected as trigger input and the slave mode controller is configured in reset mode.

The period and the pulsewidth of a PWM signal applied on tim_ti1 can be measured using the following procedure:

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to ‘0’ (active on rising edge).
- Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP=’10’ (active on falling edge).
- Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.

Figure 405. PWM input mode timing



46.3.11 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_ocxn) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, user just needs to write 0101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP=0 (tim_ocx active high) => tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 0100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

46.3.12 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed. Channels 1 to 4 can be output, while channel 5 and 6 are only available inside the microcontroller (for instance, for compound waveform generation or for ADC triggering).

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=0000), be set

active (OCxM=0001), be set inactive (OCxM=0010) or can toggle (OCxM=0011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

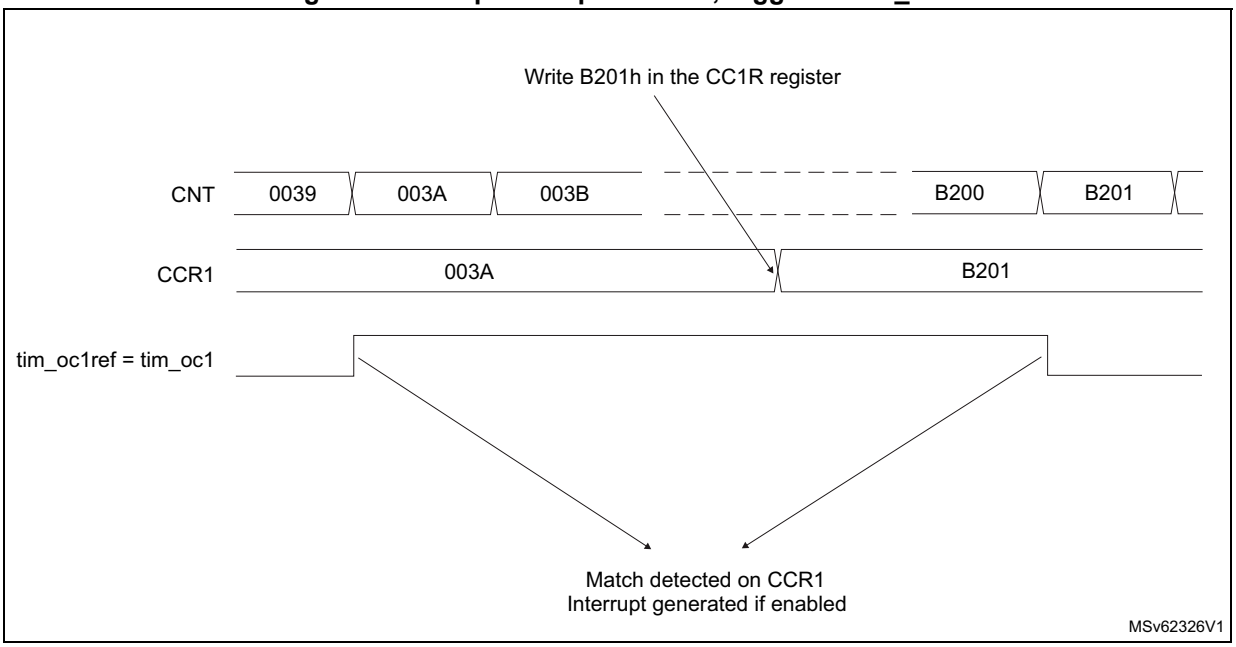
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 406*.

Figure 406. Output compare mode, toggle on tim_oc1



46.3.13 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing ‘0110’ (PWM mode 1) or ‘0111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

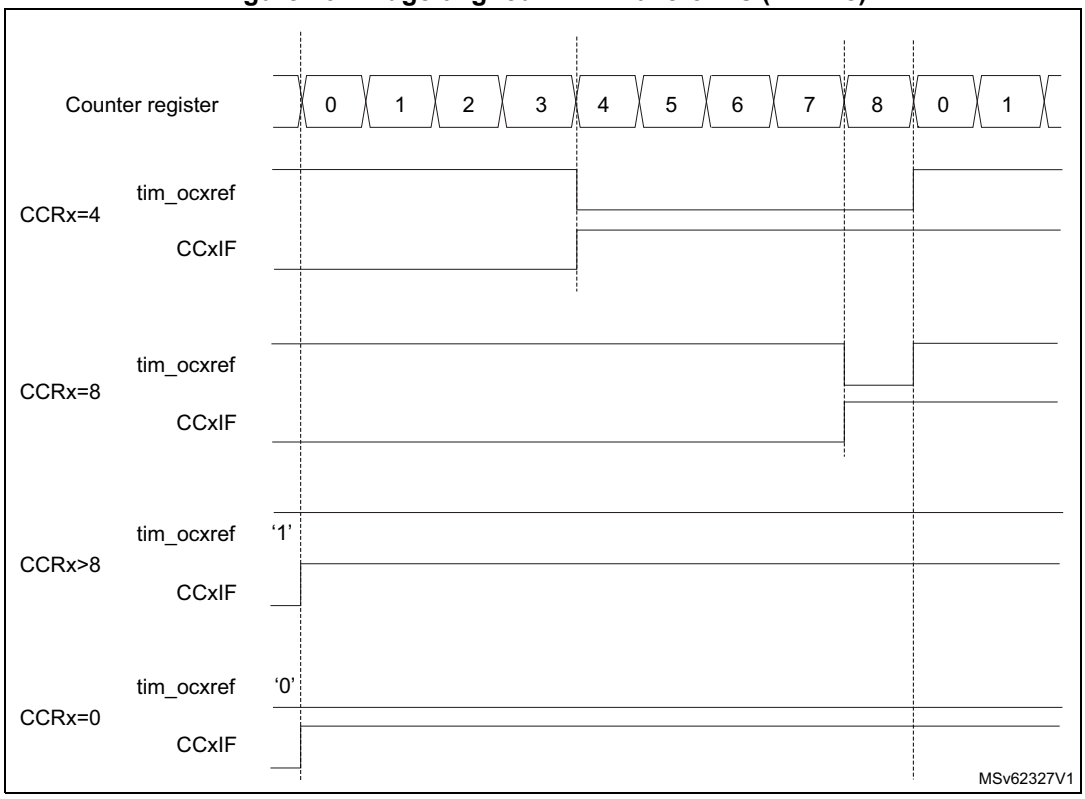
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to the *Upcounting mode on page 1639*.

  In the following example, we consider PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then tim_ocxref is held at ‘1’. If the compare value is 0 then tim_ocxref is held at ‘0’. *Figure 407* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 407. 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 1643*

  In PWM mode 1, the reference signal tim_ocxref is low as long as TIMx_CNT > TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then tim_ocxref is held at ‘1’. 0% PWM is not possible in this mode.

PWM center-aligned mode

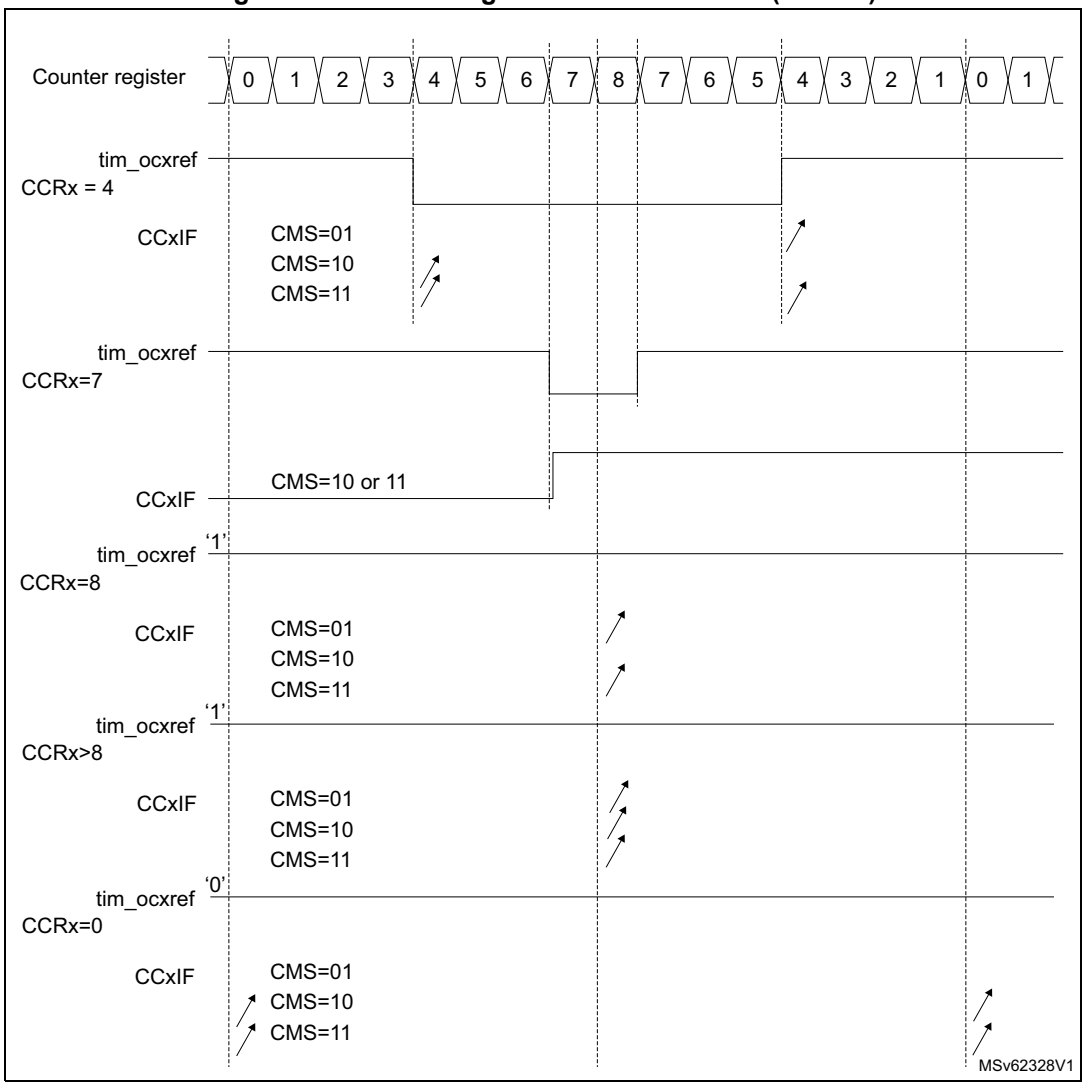
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00’ (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit

(DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to the *Center-aligned mode (up/down counting) on page 1646*.

Figure 408 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 408. Center-aligned PWM waveforms (ARR=8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit

in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

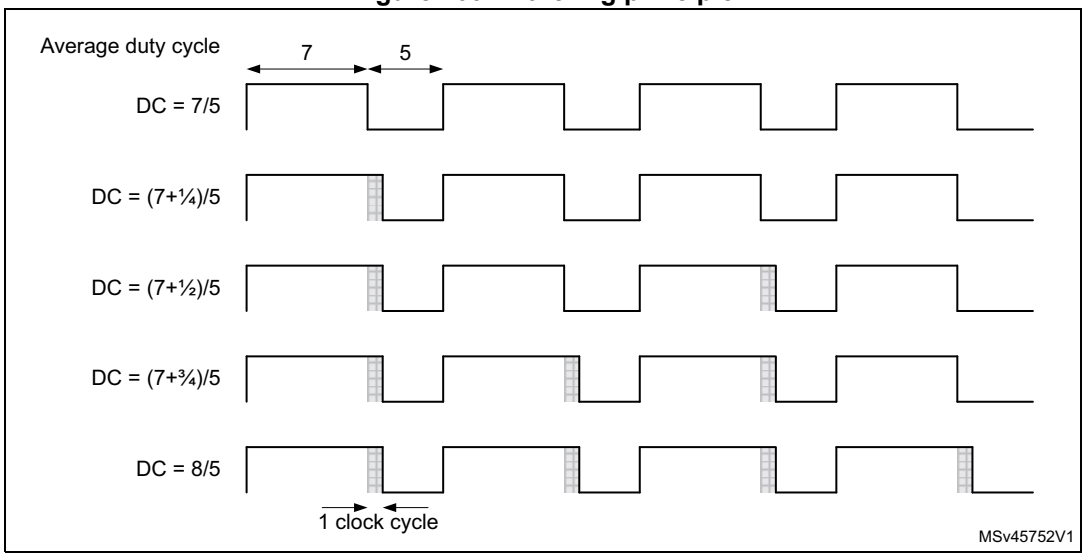
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the auto-reload value is written in the counter (TIMx_CNT > TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. The *Figure 409* below presents the dithering principle applied to 4 consecutive PWM cycles.

Figure 409. Dithering principle



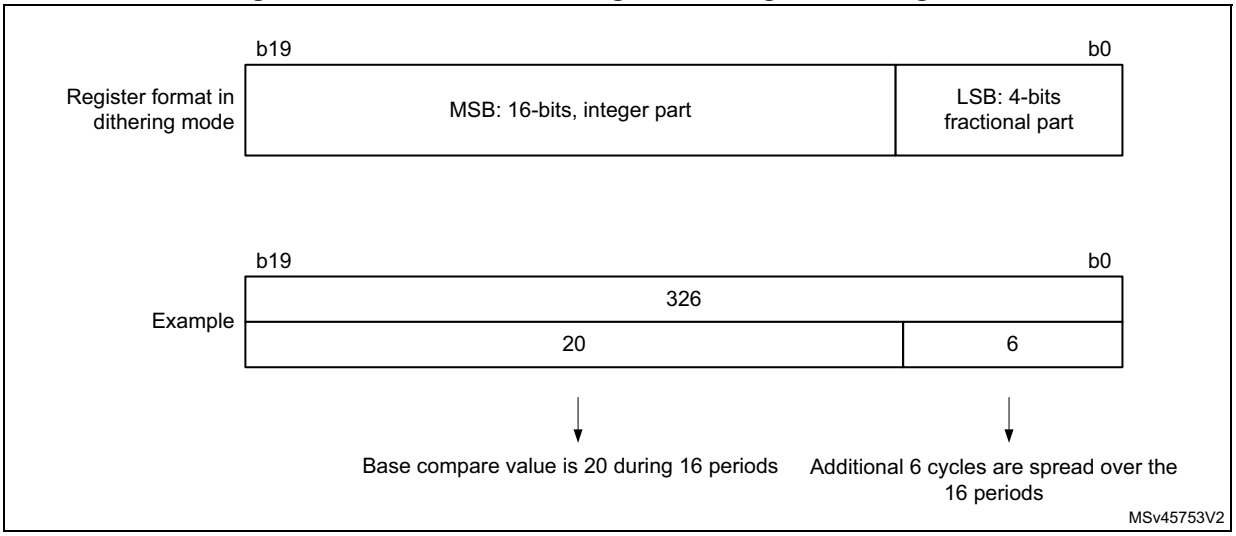
When the dithering mode is enabled, the register coding is changed as follows (see *Figure 410* for example):

- the 4 LSBs are coding for the enhanced resolution part (fractional part)
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The ARR[3:0] bits must be reset
3. The DITHEN bit must be reset

4. *The CCIF flags must be cleared*
5. *The CEN bit can be set (eventually with ARPE = 1).*

Figure 410. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

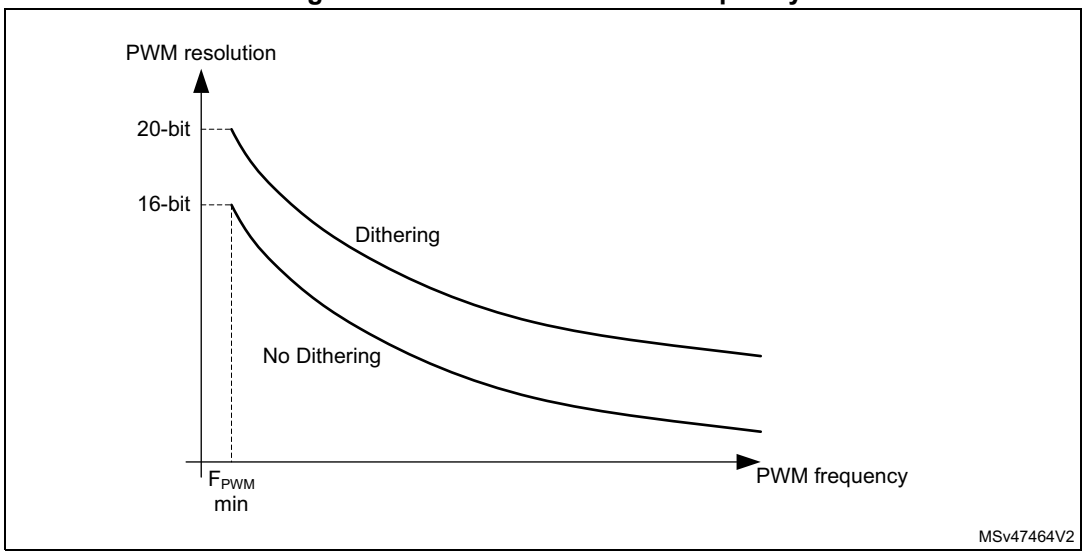
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

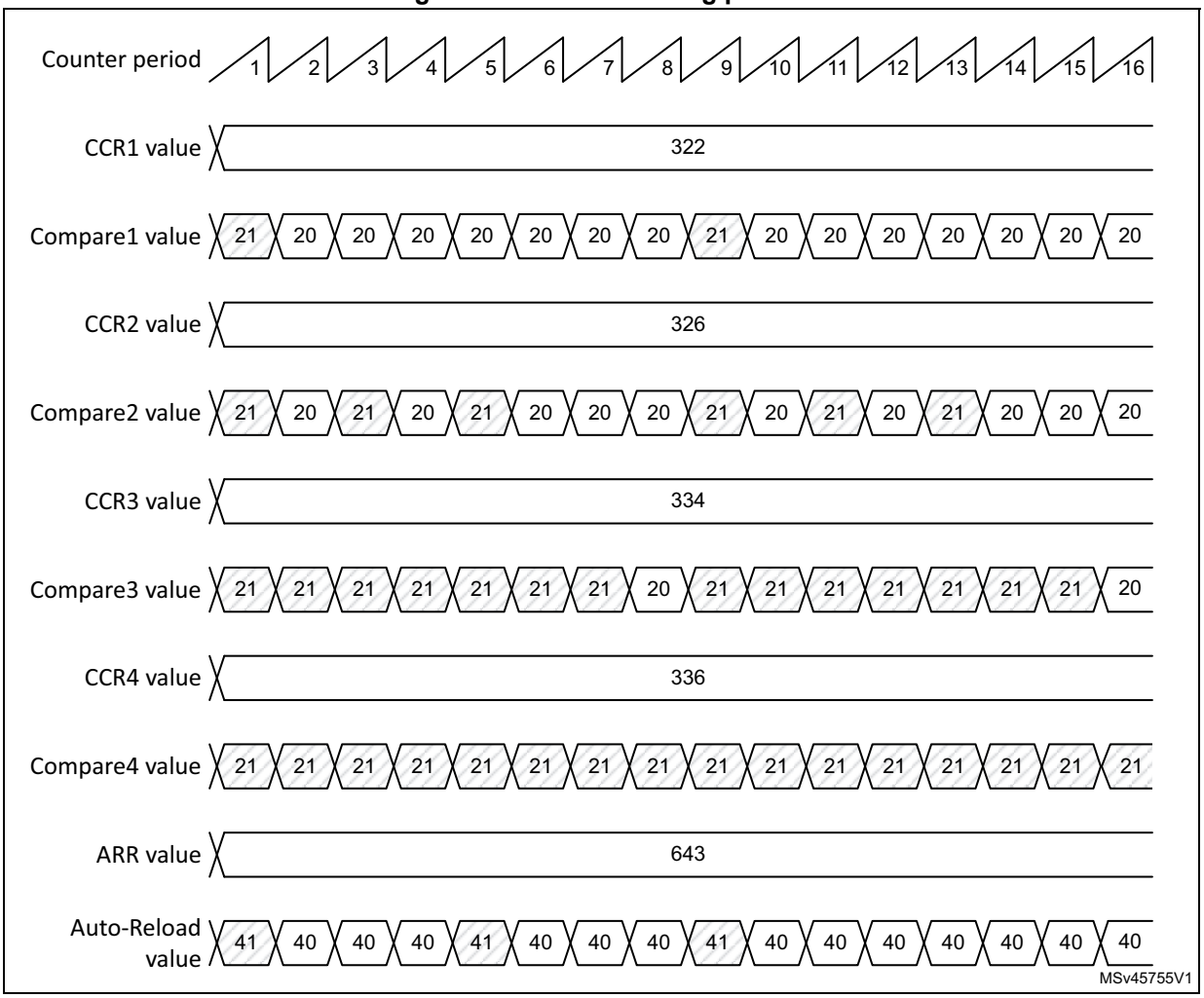
As shown on the *Figure 411* below, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 411. PWM resolution vs frequency



The duty cycle and / or period changes are spread over 16 consecutive periods, as described in the *Figure 412* below.

Figure 412. PWM dithering pattern



The auto-reload and compare values increments are spread following specific patterns described in the *Table 436* below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 436. CCR and ARR register change dithering pattern

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-

Table 436. CCR and ARR register change dithering pattern (continued)

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIMx_CR1 register are not equal to '00'). In this case, the dithering pattern is applied over 8 consecutive PWM periods, considering the up and down counting phases as shown in the *Figure 413* below.

Figure 413. Dithering effect on duty cycle in center-aligned PWM mode

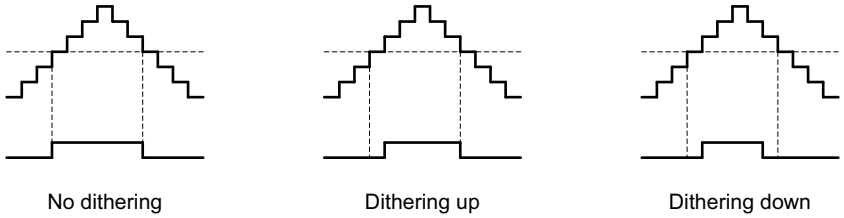


Table 437 below shows how the dithering pattern is added in center-aligned PWM mode.

Table 437. CCR register change dithering pattern in center-aligned PWM mode

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-

Table 437. CCR register change dithering pattern in center-aligned PWM mode (continued)

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

46.3.14 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx register. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

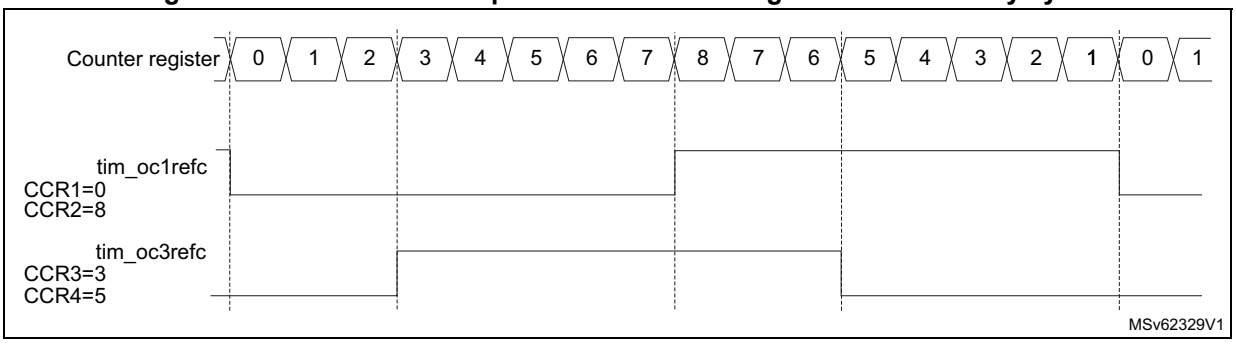
Asymmetric PWM mode can be selected independently on two channel (one tim_ocx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if an tim_oc1refc signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the tim_oc2ref signal on channel 2, or an tim_oc2refc signal resulting from asymmetric PWM mode 1.

Figure 414 represents an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 2). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.

Figure 414. Generation of 2 phase-shifted PWM signals with 50% duty cycle



46.3.15 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

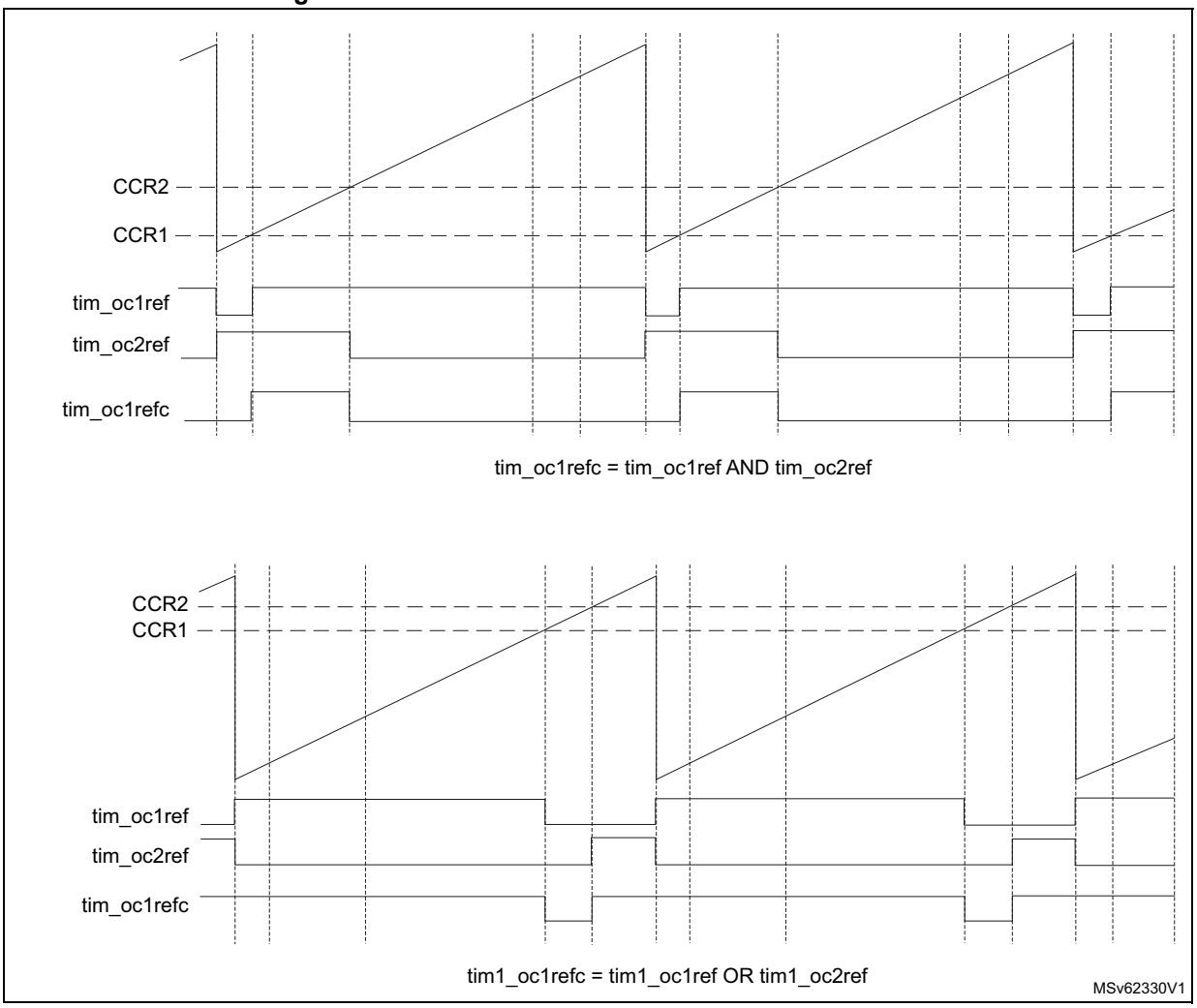
When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 415 represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
- Channel 3 is configured in Combined PWM mode 2,
- Channel 4 is configured in PWM mode 1.

Figure 415. Combined PWM mode on channel 1 and 3



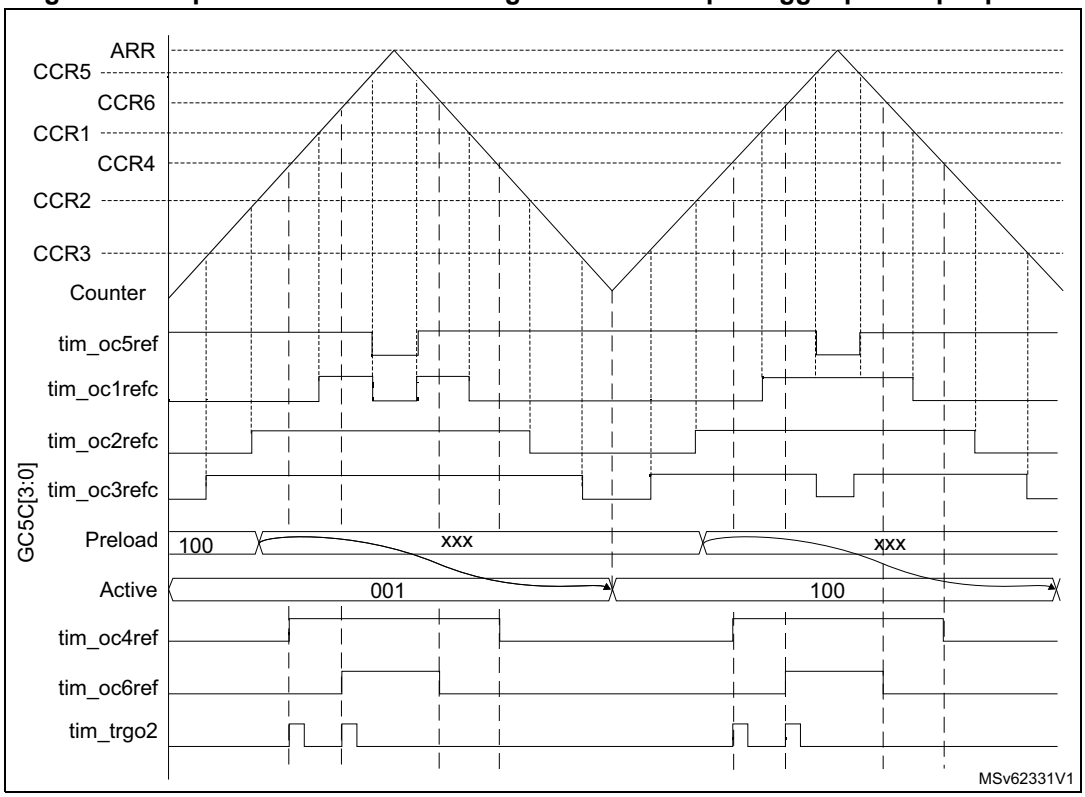
46.3.16 Combined 3-phase PWM mode

Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses. The tim_oc5ref signal is used to define the resulting combined signal. The 3-bits GC5C[3:1] in the TIMx_CCR5 allow selection on which reference signal the tim_oc5ref is combined. The resulting signals, tim_ocxrefc, are made of an AND logical combination of two reference PWMs:

- If GC5C1 is set, tim_oc1refc is controlled by TIMx_CCR1 and TIMx_CCR5
- If GC5C2 is set, tim_oc2refc is controlled by TIMx_CCR2 and TIMx_CCR5
- If GC5C3 is set, tim_oc3refc is controlled by TIMx_CCR3 and TIMx_CCR5

Combined 3-phase PWM mode can be selected independently on channels 1 to 3 by setting at least one of the 3-bits GC5C[3:1].

Figure 416. 3-phase combined PWM signals with multiple trigger pulses per period



The tim_trgo2 waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Refer to *Section 46.3.31: ADC triggers* for more details.

46.3.17 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 tim_ocx or complementary tim_ocxn) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals tim_ocx and tim_ocxn are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to *Table 445: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature on page 1748* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a

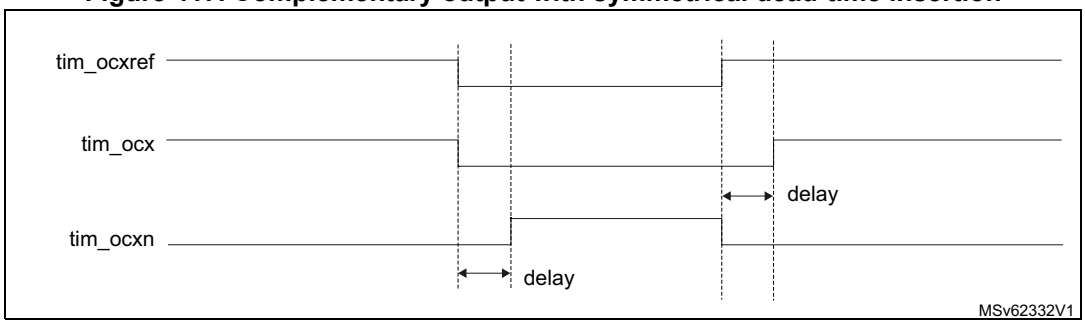
reference waveform tim_ocxref, it generates 2 outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_ocx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The tim_ocxn output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (tim_ocx or tim_ocxn) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal tim_ocxref. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

Figure 417. Complementary output with symmetrical dead-time insertion



The DTAE bit in the TIMx_DTR2 is used to differentiate the deadtime values for rising and falling edges of the reference signal, as shown on *Figure 418*.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bitfield in the TIMx_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bitfield in the TIMx_DTR2 register. The DTAE bit must be written before enabling the counter and must not be modified while CEN=1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bitfield DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set, in the TIMX_DTR2 register. The preload value is loaded in the active register on the next update event.

Note: If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.

Figure 418. Asymmetrical deadtime

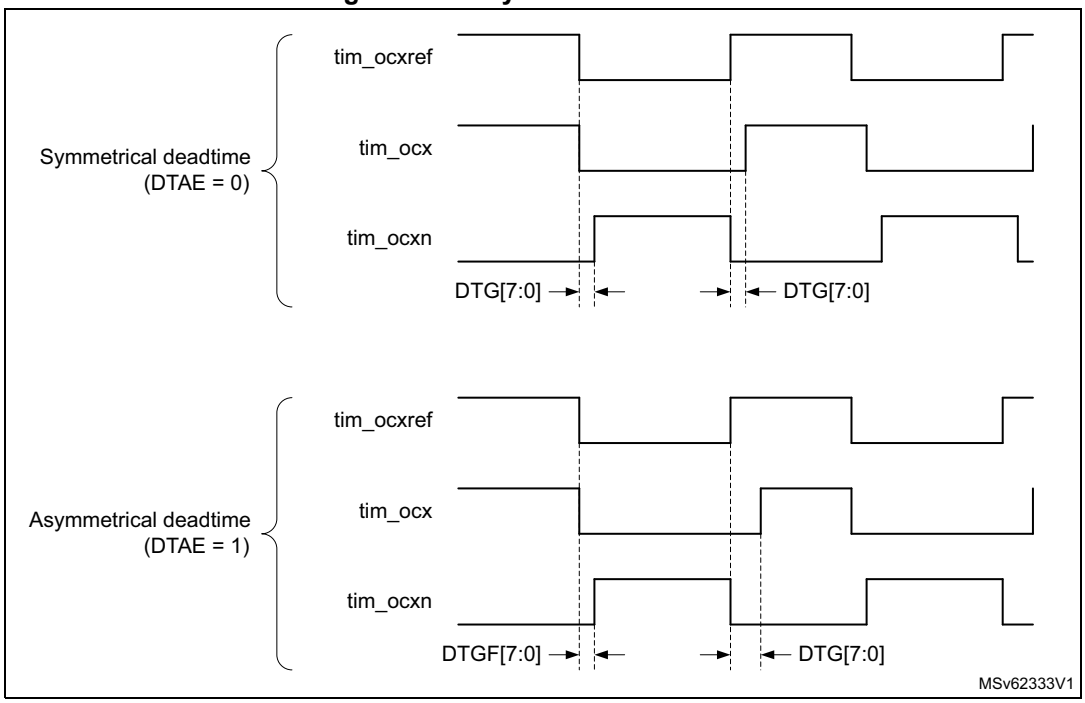


Figure 419. Dead-time waveforms with delay greater than the negative pulse

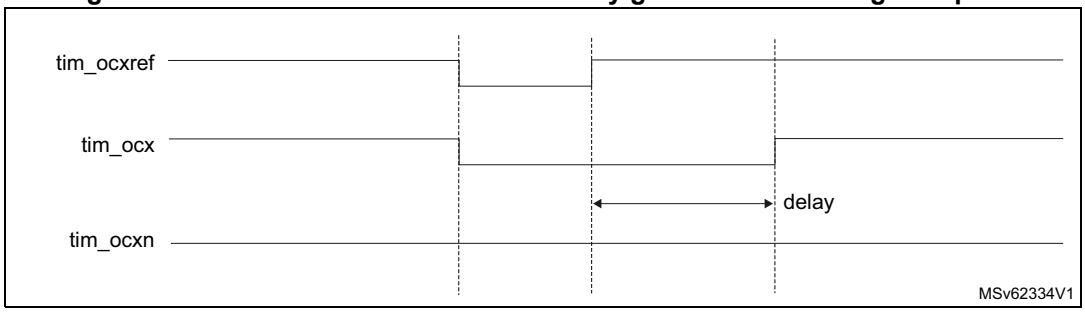
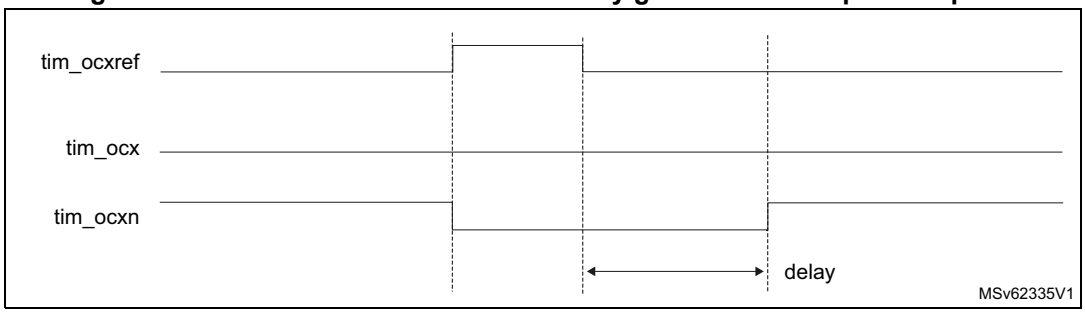


Figure 420. Dead-time waveforms with delay greater than the positive pulse



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 46.6.20: TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)* for delay calculation.

Re-directing tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be re-directed to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This is used to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only tim_ocxn is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP=0 then tim_ocxn=tim_ocxref. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE=CCxNE=1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

46.3.18 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The two break inputs are usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state. A number of internal MCU events can also be selected to trigger an output shut-down.

The break features two channels. A break channel which gathers both system-level fault (clock failure, ECC / parity errors,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration. A break2 channel which only includes application faults and is able to force the outputs to an inactive state.

The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx_BDTR register is used to enable /disable the outputs by software and is reset in case of break or break2 event.
- the OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- the OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 445: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature on page 1748* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break functions can be enabled by setting the BKE and BK2E bits in the TIMx_BDTR register. The break input polarities can be selected by configuring the BKP and BK2P bits in the same register. BKEx and BKPx can be modified at the same time. When the BKEx and BKPx bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous

and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The sources for break (tim_brk) channel are:

- External sources connected to one of the TIMx_BKIN pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources:
  - coming from a tim_brk_cmpx input (refer to *Section 46.3.2: TIM1/TIM8 pins and internal signals* for product specific implementation)
  - coming from a system break request (refer to *Section 46.3.2: TIM1/TIM8 pins and internal signals* for product specific implementation)

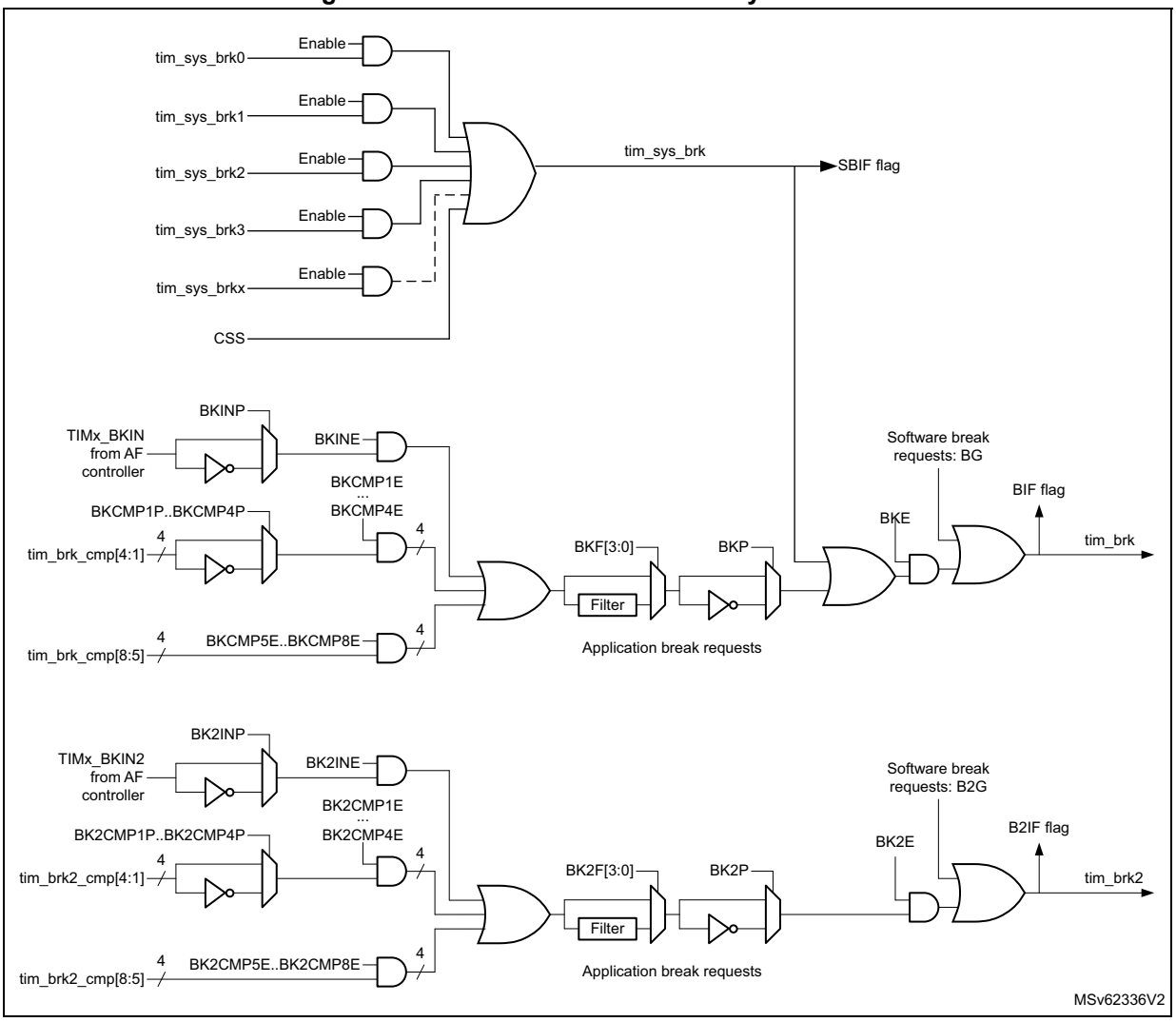
The sources for break2 (tim_brk2) are:

- External sources connected to one of the TIMx_BKIN2 pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources coming from a tim_brk2_cmpx input (refer to *Section 46.3.2: TIM1/TIM8 pins and internal signals* for product specific implementation)

Break events can also be generated by software using BG and B2G bits in the TIMx_EGR register.

All sources are ORed before entering the timer tim_brk or tim_brk2 inputs, as per *Figure 421* below.

Figure 421. Break and Break2 circuitry overview



Note: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When one of the breaks occurs (selected level on one of the break inputs):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the GPIO controller (selected by the OSSI bit). This feature is enabled even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE=0. If OSSI=0, the timer releases the output control (taken over by the GPIO controller), otherwise the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, tim_ocx and tim_ocxn cannot be driven to

their active level together. Note that because of the resynchronization on MOE, the dead-time duration is slightly longer than usual (around 2 tim_ker_ck clock cycles).

  - If OSSI=0, the timer releases the output control (taken over by the GPIO controller which forces a Hi-Z state), otherwise the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

- The break status flag (SBIF, BIF and B2IF bits in the TIMx_SR register) is set. An interrupt is generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event (UEV). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it to '1' again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

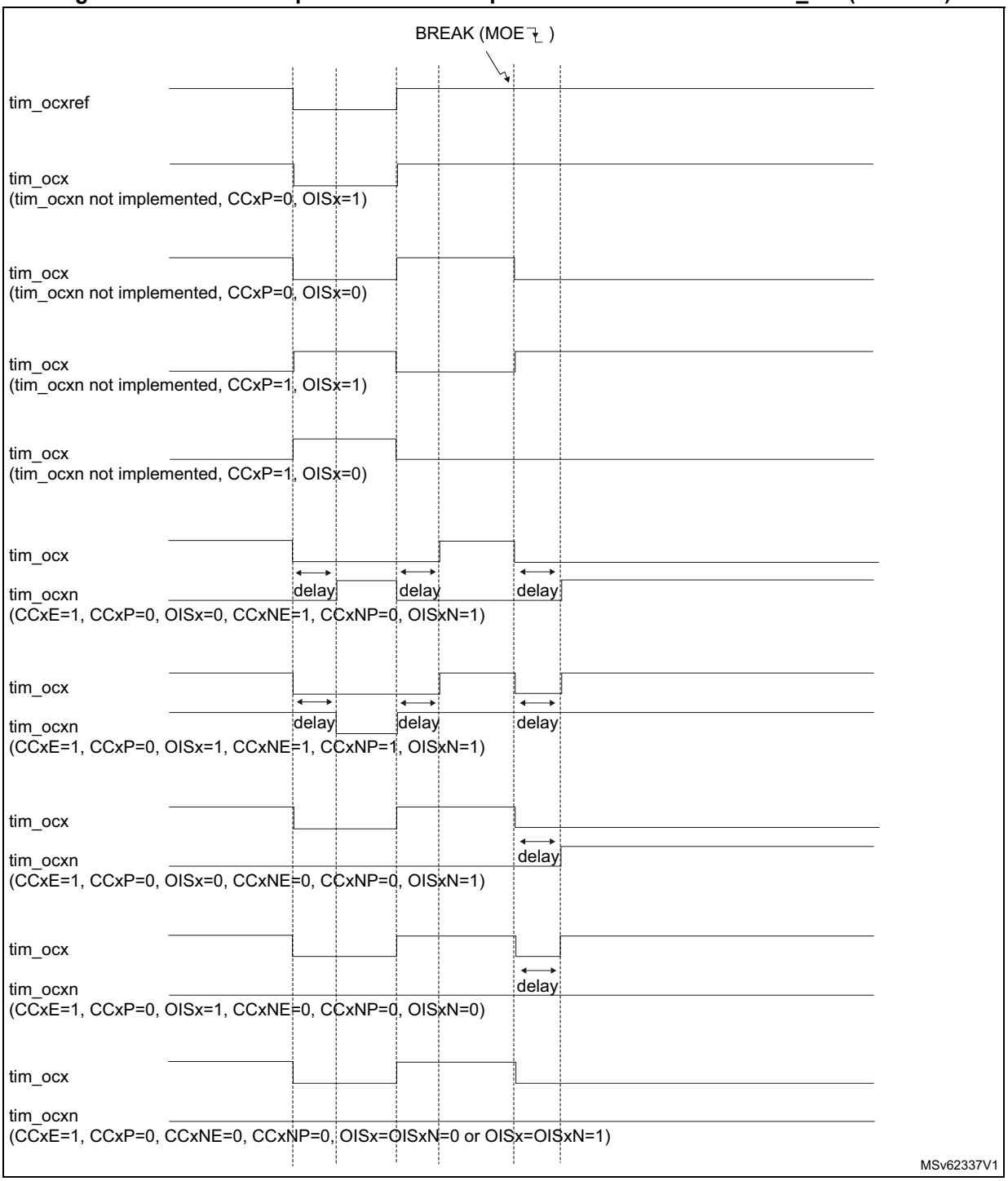
Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value. If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.

Note: The break inputs are active on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF and B2IF cannot be cleared.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It is used to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_ocxn polarities and state when disabled, OCxM configurations, break enable and polarity). The application can choose from 3 levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to *Section 46.6.20: TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)*. The LOCK bits can be written only once after an MCU reset.

Figure 422 shows an example of behavior of the outputs in response to a break.

Figure 422. Various output behavior in response to a break event on tim_brk (OSSI = 1)



The two break inputs have different behaviors on timer outputs:

- The tim_brk input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- tim_brk2 can only disable (inactive state) the PWM outputs.

The tim_brk has a higher priority than tim_brk2 input, as described in *Table 438*.

Note: tim_brk2 must only be used with OSSR = OSSI = 1.

Table 438. Behavior of timer outputs versus tim_brk/tim_brk2 inputs

tim_brk	tim_brk2	Timer outputs state	Typical use case	
			tim_ocxn output (low side switches)	tim_ocx output (high side switches)
Active	X	– Inactive then forced output state (after a deadtime) – Outputs disabled if OSSI = 0 (control taken over by GPIO logic)	ON after deadtime insertion	OFF
Inactive	Active	Inactive	OFF	OFF

Figure 423 gives an example of tim_ocx and tim_ocxn output behavior in case of active signals on tim_brk and tim_brk2 inputs. In this case, both outputs have active high polarities (CCxP = CCxNP = 0 in TIMx_CCER register).

Figure 423. PWM output state following tim_brk and tim_brk2 assertion (OSSI=1)

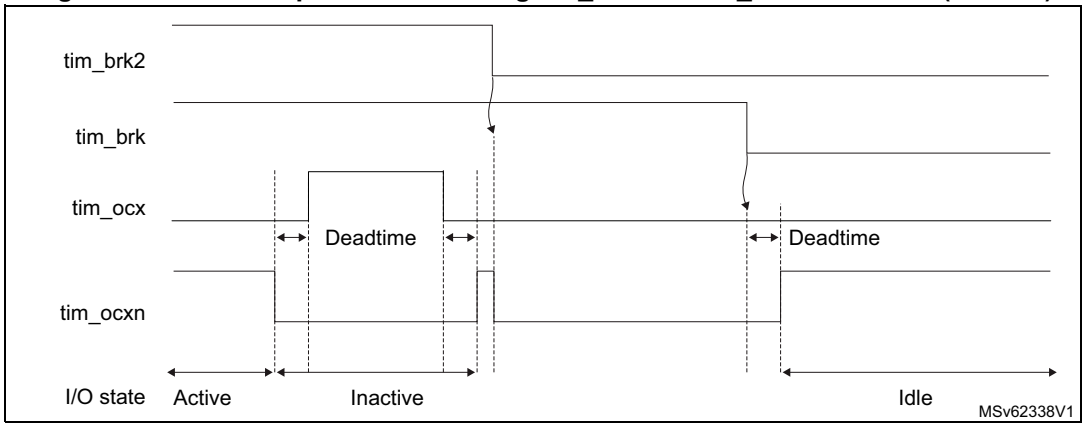
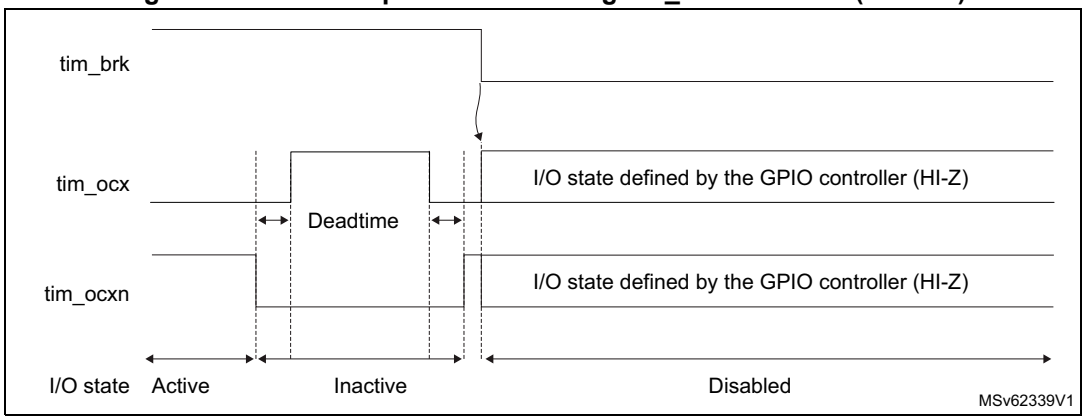


Figure 424. PWM output state following tim_brk assertion (OSSI=0)



46.3.19 Bidirectional break inputs

The TIM1/TIM8 are featuring bidirectional break I/Os, as represented on *Figure 425*.

This provides support for:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin
- Internal break sources and multiple external open drain sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged

The tim_brk and tim_brk2 inputs are configured in bidirectional mode using the BKBID and BK2BID bits in the TIMxBDTR register. The BKBID programming bits can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode is available for both the tim_brk and tim_brk2 inputs, and require the I/O to be configured in open-drain mode with active low polarity (using BKINP, BKP, BK2INP and BK2P bits). Any break request coming either from system (e.g. CSS), from on-chip peripherals or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software events (BG and B2G) also cause the break I/O to be forced to '0' to indicate to the external components that the timer is entered in break state. However, this is valid only if the break is enabled (BKE or B2KE = 1). When a software break event is generated with BKE or B2KE = 0), the outputs are put in safe state and the break flag is set, but there is no effect on the TIMx_BKIN and TIMx_BKIN2 I/Os.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM (BK2DSRM) bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM (BK2DSRM) bit is set and the open drain control is released. This prevents the PWM output to be re-started as long as the break condition is present.
- The BKDSRM (BK2DSRM) bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 439*).

Table 439. Break protection disarming conditions

MOE	BKBID (BK2BID)	BKDSRM (BK2DSRM)	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and re-arming break circuitry

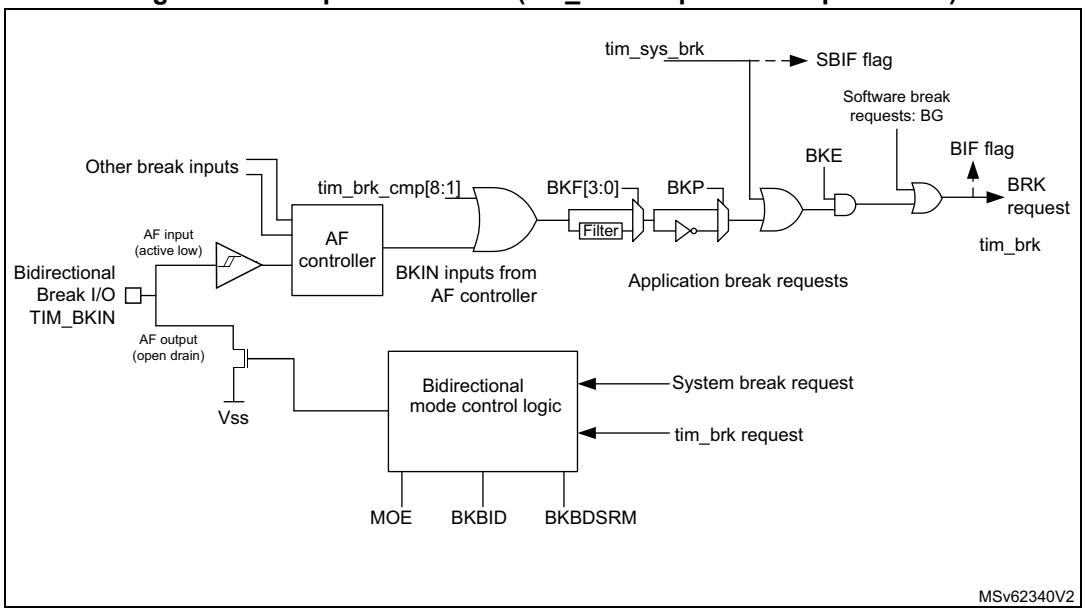
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break (break2) event:

- The BKDSRM (BK2DSRM) bit must be set to release the output control
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before re-arming)
- The software must poll the BKDSRM (BK2DSRM) bit until it is cleared by hardware (when the application break condition disappears)

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 425. Output redirection (tim_brk2 request not represented)



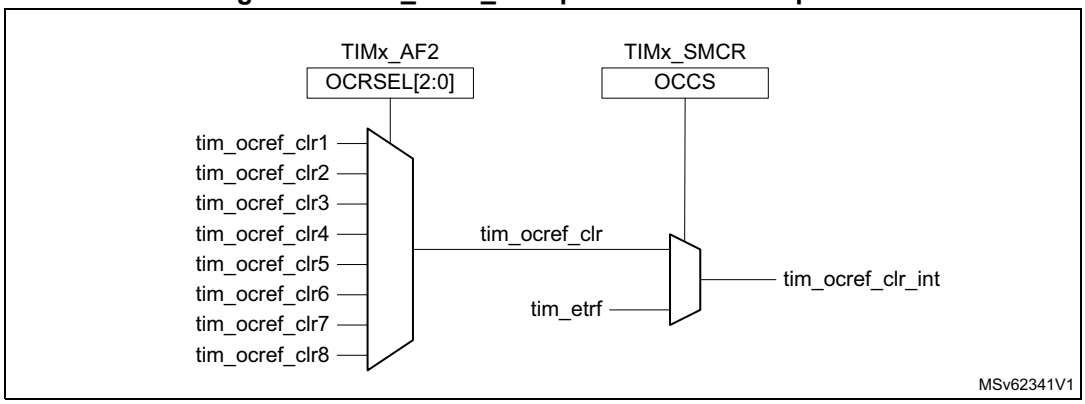
46.3.20 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next update event (UEV) occurs. This function can only

be used in Output compare and PWM modes. It does not work in Forced mode. tim_ocref_clr_int input can be selected between the tim_ocref_clr input and tim_etrf (tim_etr_in after the filter) by configuring the OCCS bit in the TIMx_SMCR register.

The tim_ocref_clr input can be selected among several inputs, using the OCRSEL[2:0] bitfield in the TIMx_AF2 register, as shown on the *Figure 426* below. Refer to *Section 46.3.2: TIM1/TIM8 pins and internal signals* for a list of sources available in the product.

Figure 426. tim_ocref_clr input selection multiplexer

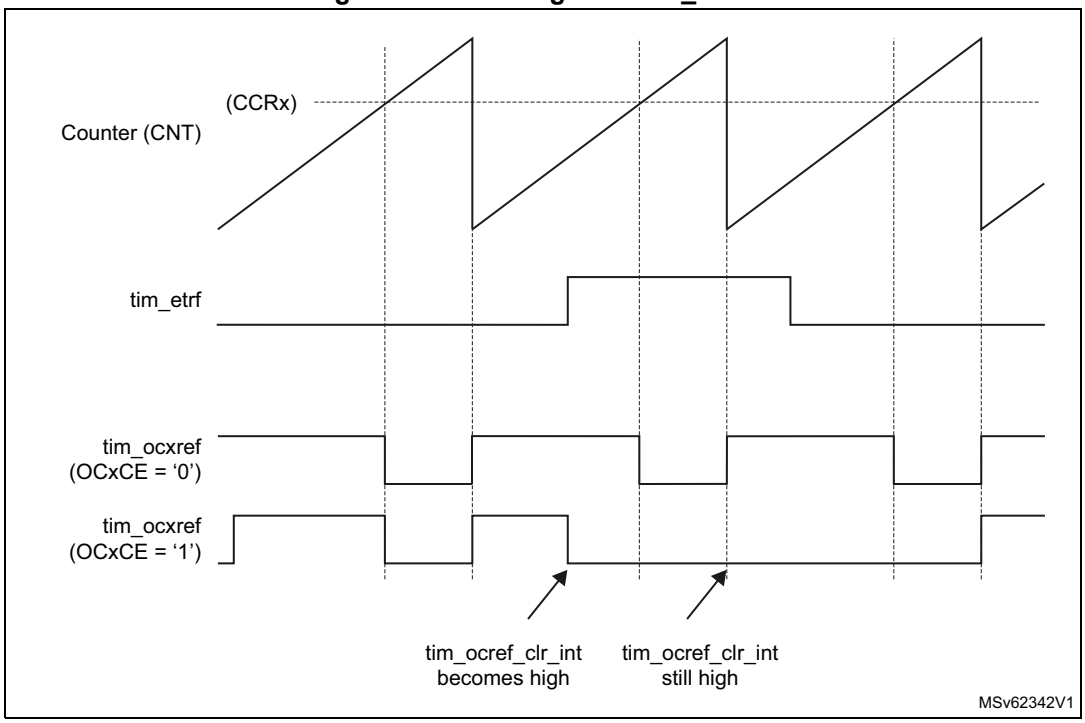


When tim_etrf is chosen, tim_etr_in must be configured as follows:

1. The External Trigger Prescaler must be kept off: bits ETPS[1:0] of the TIMx_SMCR register set to ‘00’.
2. The external clock mode 2 must be disabled: bit ECE of the TIMx_SMCR register set to ‘0’.
3. The External Trigger Polarity (ETP) and the External Trigger Filter (ETF) can be configured according to application needs (as per polarity of the source connected to the trigger and eventual need to remove noise using the filter).

Figure 427 shows the behavior of the tim_ocxref signal when the tim_etrf Input becomes High, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

Figure 427. Clearing TIMx tim_ocxref



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), then tim_ocxref is enabled again at the next counter overflow.

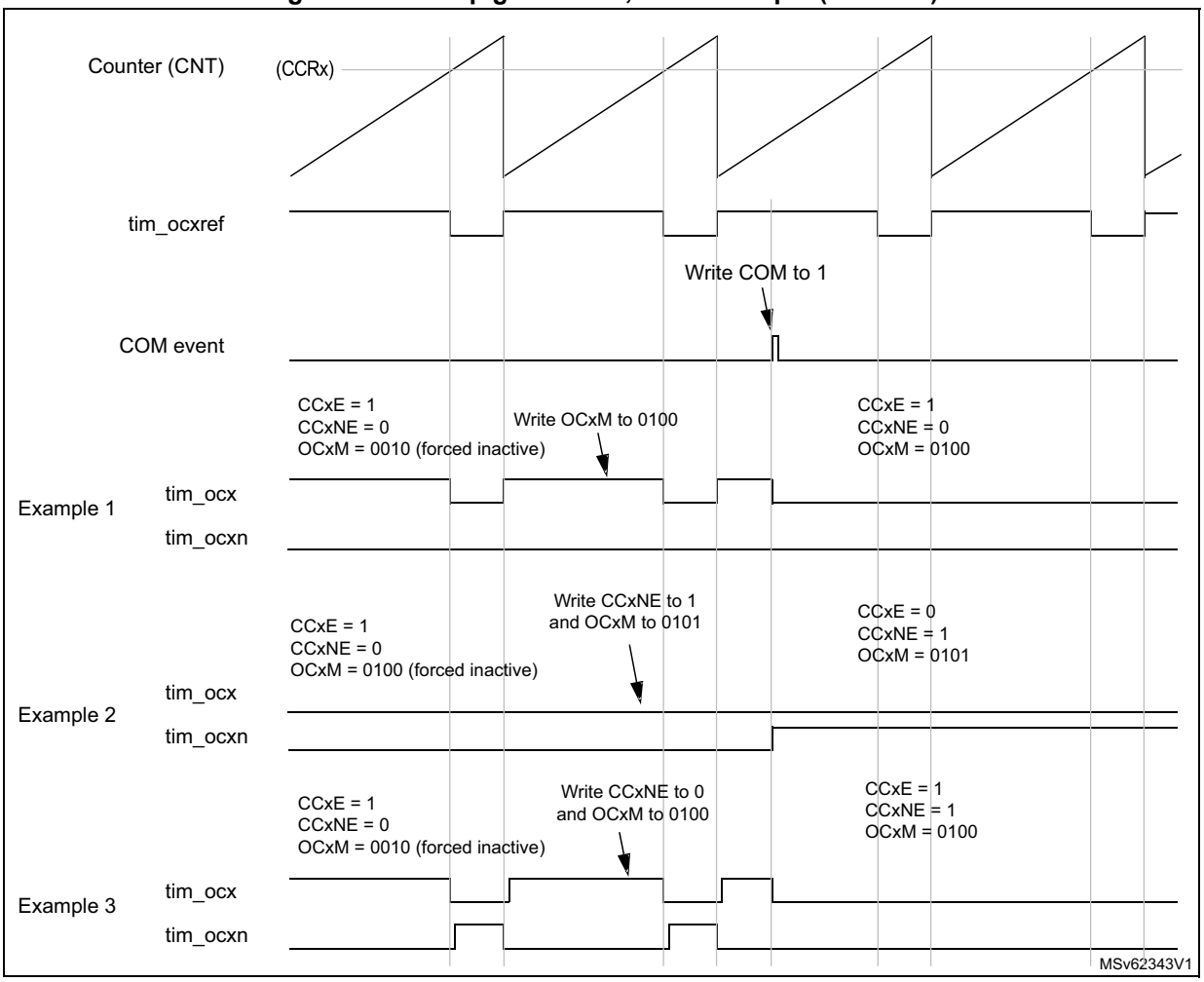
46.3.21 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The *Figure 428* describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 428. 6-step generation, COM example (OSSR=1)



46.3.22 One-pulse mode

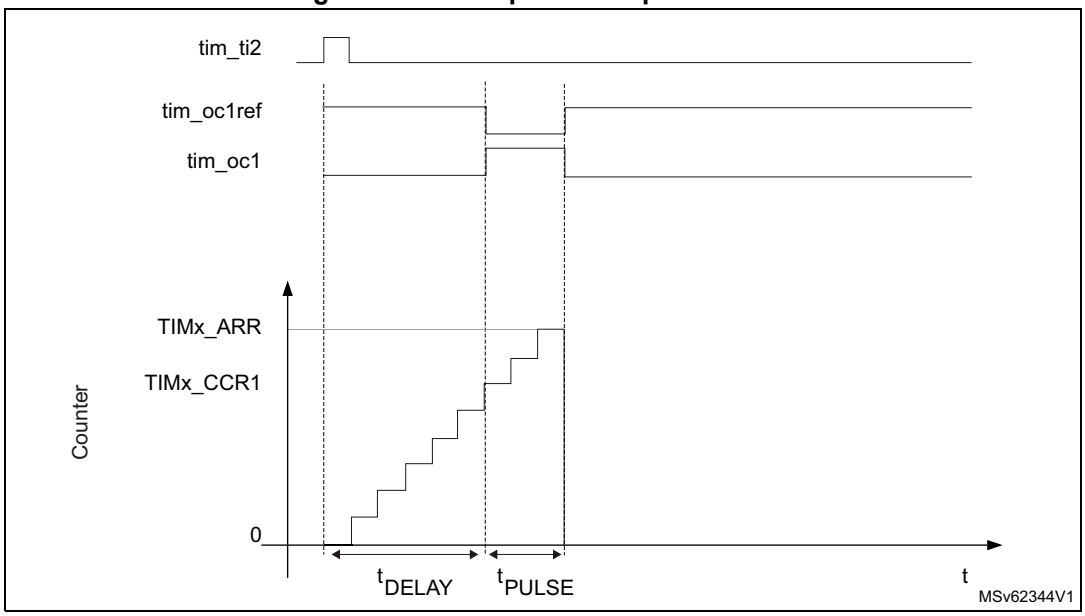
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 429. Example of one pulse mode.



For example one may want to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Let's use tim_ti2fp2 as trigger 1:

- Map tim_ti2fp2 to tim_ti2 by writing CC2S='01' in the TIMx_CCMR1 register.
- tim_ti2fp2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
- Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS=00110 in the TIMx_SMCR register.
- tim_ti2fp2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to '0' in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: tim_ocx fast enable:

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

46.3.23 Retriggerable One-pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one-pulse mode described in *Section 46.3.22*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

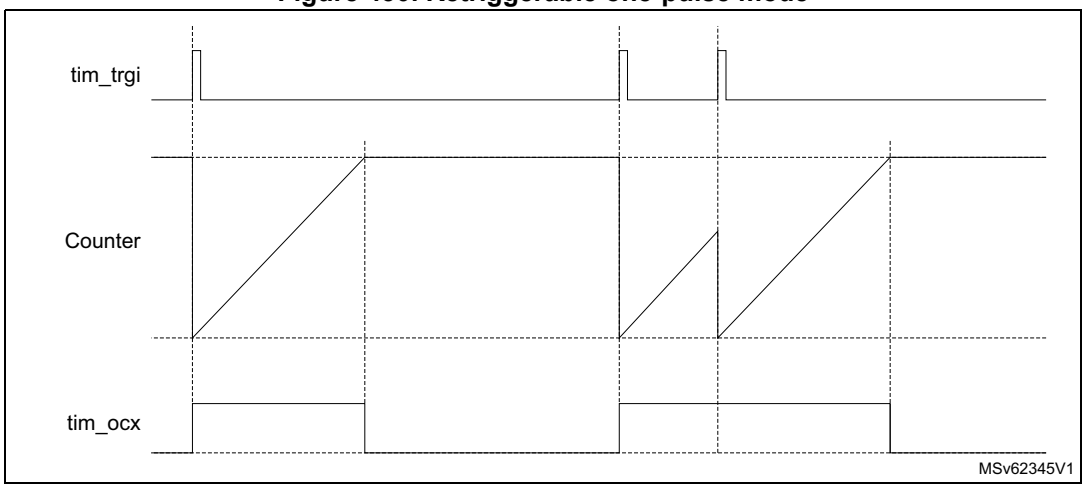
The timer must be in Slave mode, with the bits SMS[3:0] = ‘1000’ (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to ‘1000’ or ‘1001’ for retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 430. Retriggerable one-pulse mode

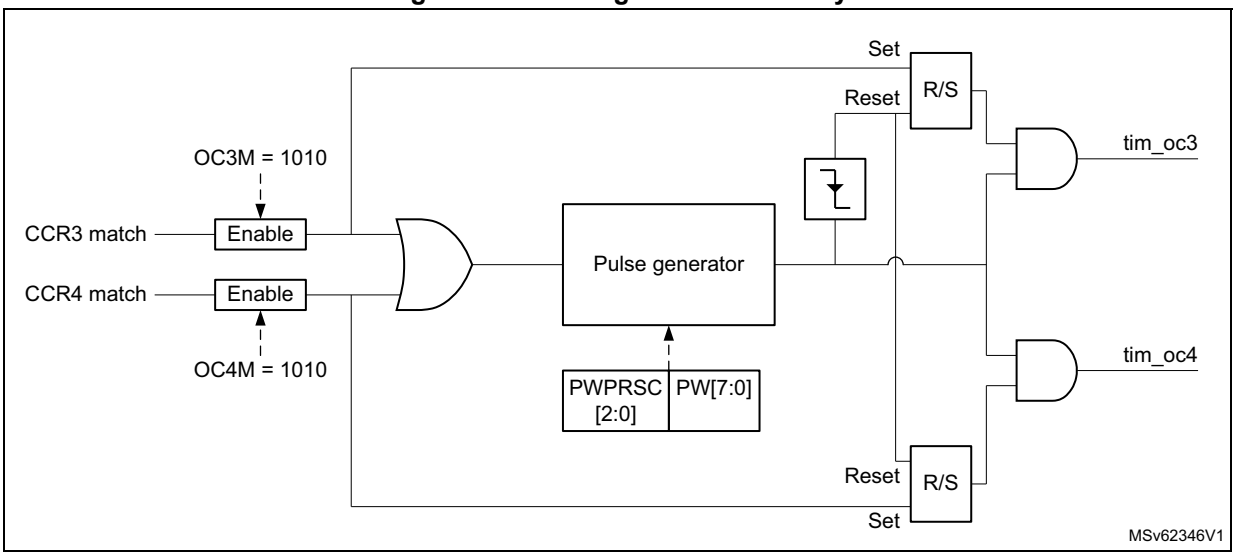


46.3.24 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulsewidth generated when the counter value equals a given compare value, for debugging or synchronization purposes.

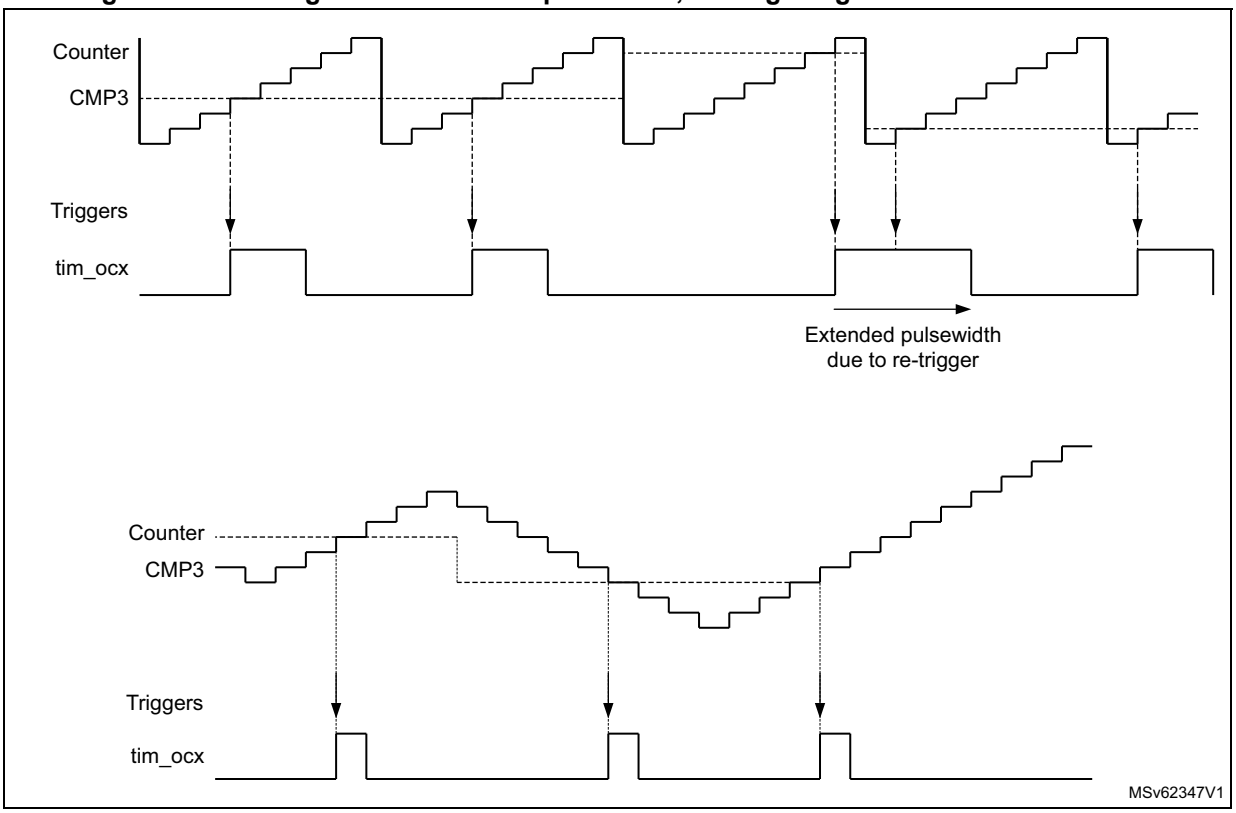
This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on the *Figure 431* below.

Figure 431. Pulse generator circuitry



The *Figure 432* below shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 432. Pulse generation on compare event, for edge-aligned and encoder modes



This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bit fields in TIMx_CCMR2 register.

The pulsewidth is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

t_{PW} = PW[7:0] x t_{PWG}

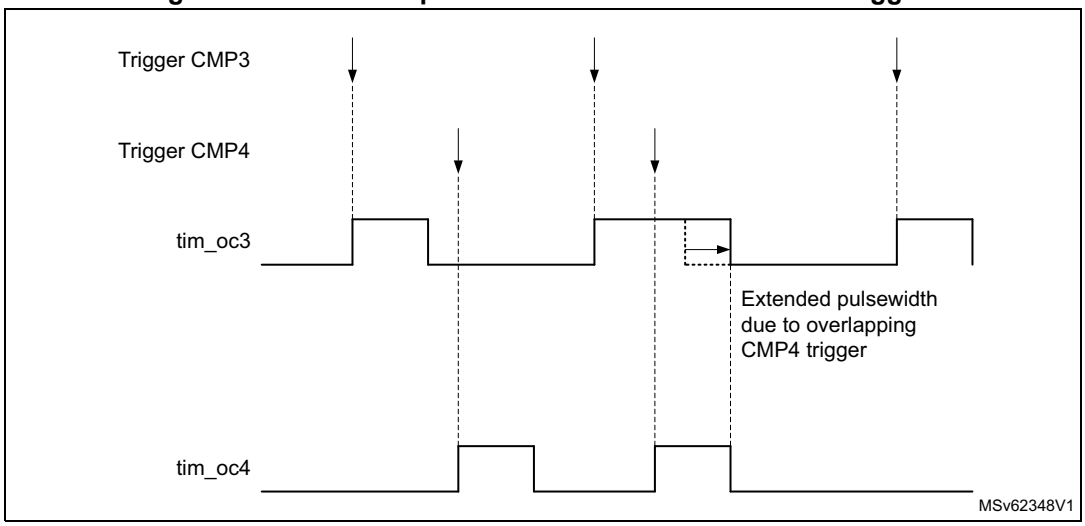
where $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim\_ker\_ck}$.

gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is ongoing, causes the pulse to be extended.

Note: If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the 1st arriving trigger is extended (because of the re-trigger), while the pulse width of the last arriving trigger is correct (as shown on the Figure 433 below).

Figure 433. Extended pulsewidth in case of concurrent triggers



46.3.25 Encoder interface mode

Quadrature encoder

To select Encoder Interface mode write SMS='0001' in the TIMx_SMCR register if the counter is counting on tim_ti1 edges only, SMS='0010' if it is counting on tim_ti2 edges only and SMS='0011' if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs tim_ti1 and tim_ti2 are used to interface to an quadrature encoder. Refer to *Table 440*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1=tim_ti1 if not filtered and not inverted, tim_ti2fp2=tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to '1'). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.

Table 440. Counting direction versus encoder signals (CC1P = CC2P = 0)

Active edge	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	1110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	1111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	0001	High	Down	Up	No count	No count
		Low	Up	Down	No count	No count
Counting on tim_ti2 only x2 mode	0010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	0011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to the external trigger input and trigger a counter reset.

The *Figure 434* gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S='01' (TIMx_CCMR1 register, tim_ti1fp1 mapped on tim_ti1).
- CC2S='01' (TIMx_CCMR2 register, tim_ti1fp2 mapped on tim_ti2).
- CC1P='0' and CC1NP='0' (TIMx_CCER register, tim_ti1fp1 non-inverted, tim_ti1fp1=tim_ti1).
- CC2P='0' and CC2NP='0' (TIMx_CCER register, tim_ti1fp2 non-inverted, tim_ti1fp2= tim_ti2).
- SMS='011' (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN='1' (TIMx_CR1 register, Counter enabled).

Figure 434. Example of counter operation in encoder interface mode.

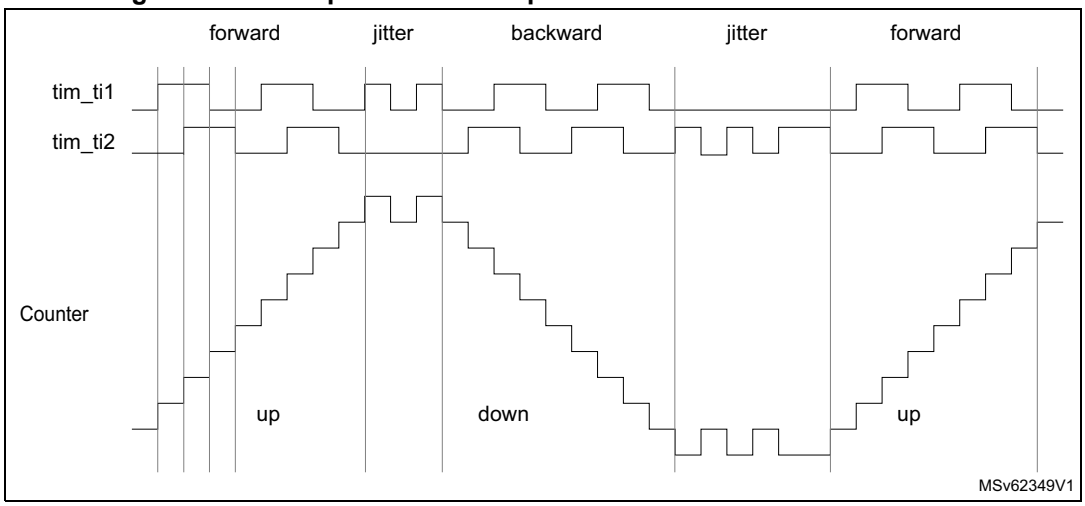
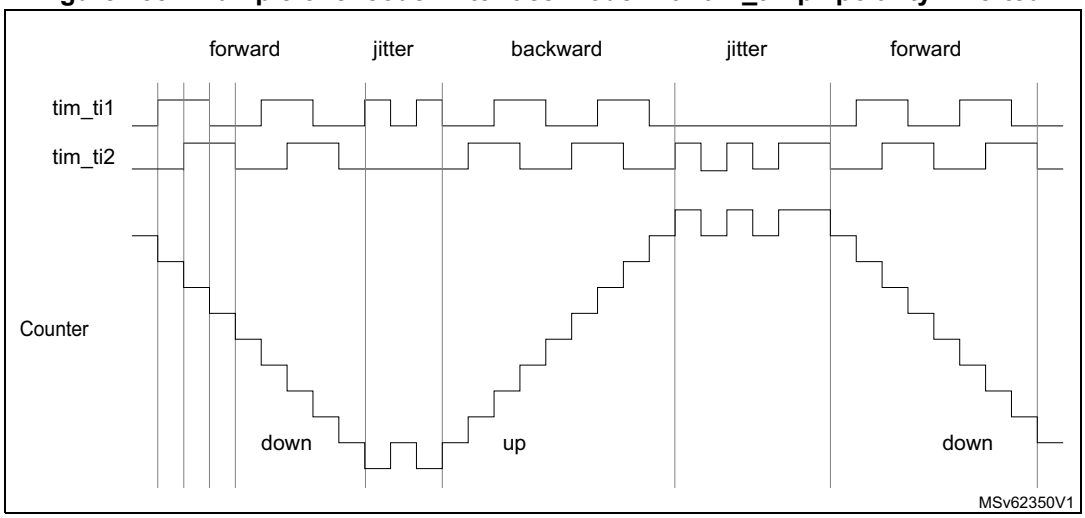


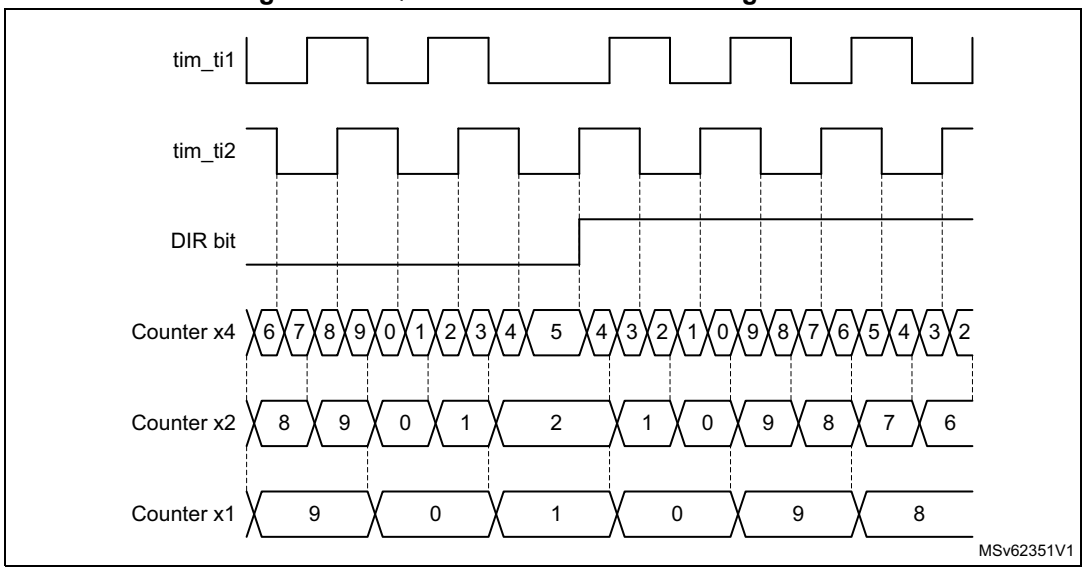
Figure 435 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P='1').

Figure 435. Example of encoder interface mode with tim_ti1fp1 polarity inverted.



The *Figure 436* below shows the timer counter value during a speed reversal, for various counting modes.

Figure 436. Quadrature encoder counting modes



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support other types of encoders.

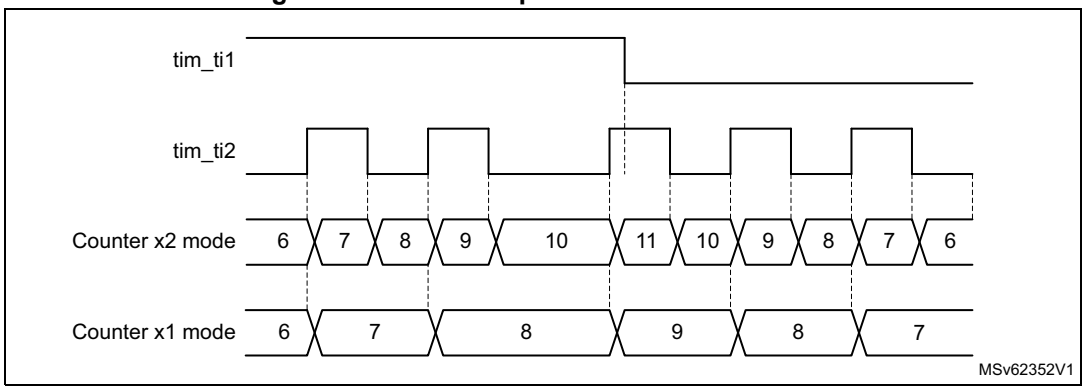
In the "clock plus direction" mode shown on *Figure 437*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1010: x2 mode, the counter is updated on both rising and falling edges of the clock
- 1011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 437. Direction plus clock encoder mode



Directional Clock encoder mode

In the “directional clock” mode on *Figure 438*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock line. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 438* below) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 439* below).
- 1101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 438* below), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 439* below).

Figure 438. Directional clock encoder mode (CC1P = CC2P = 0)

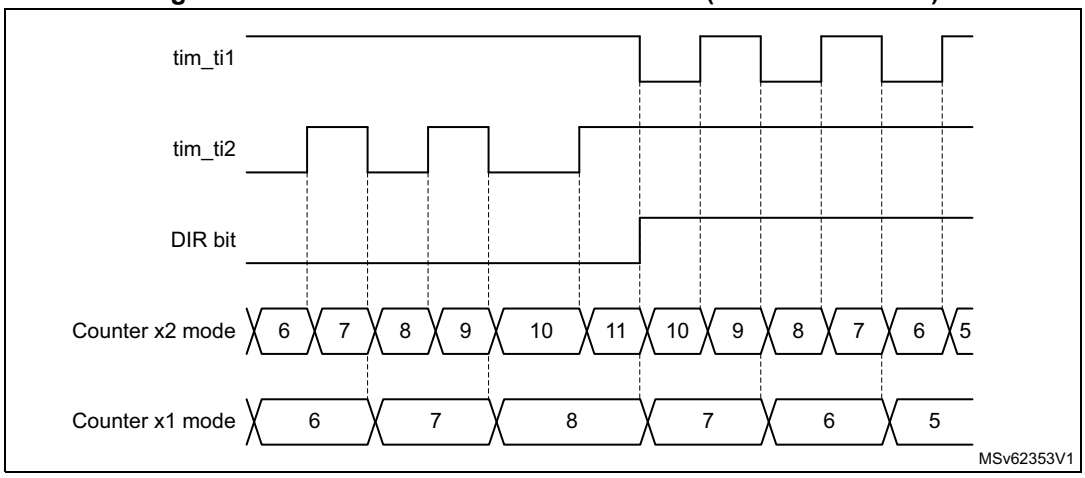
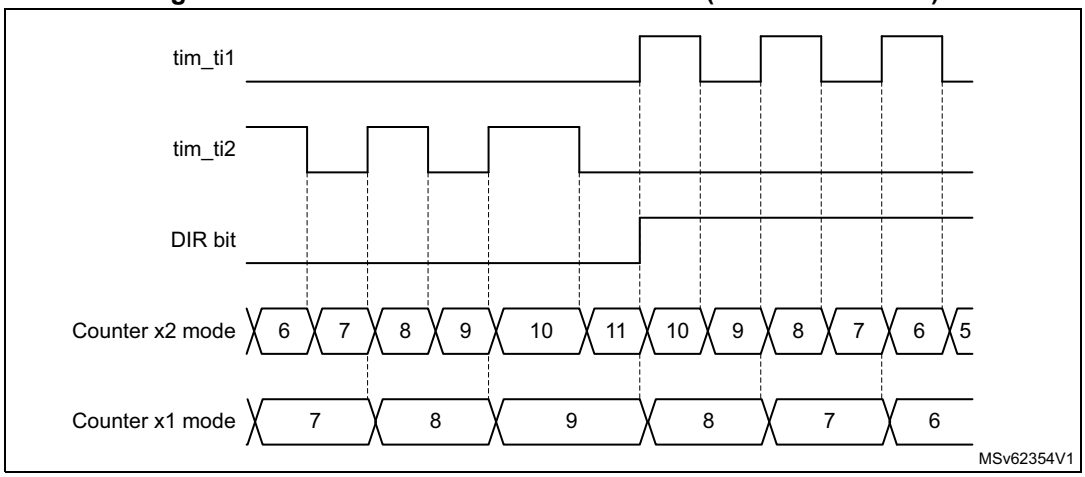


Figure 439. Directional clock encoder mode (CC1P = CC2P = 1)



The *Table 441* here-below details how the directional clock mode operates, for any input transition.

Table 441. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP=0	1100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP=1	1100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP=0	1101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP=1	1101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Index Input

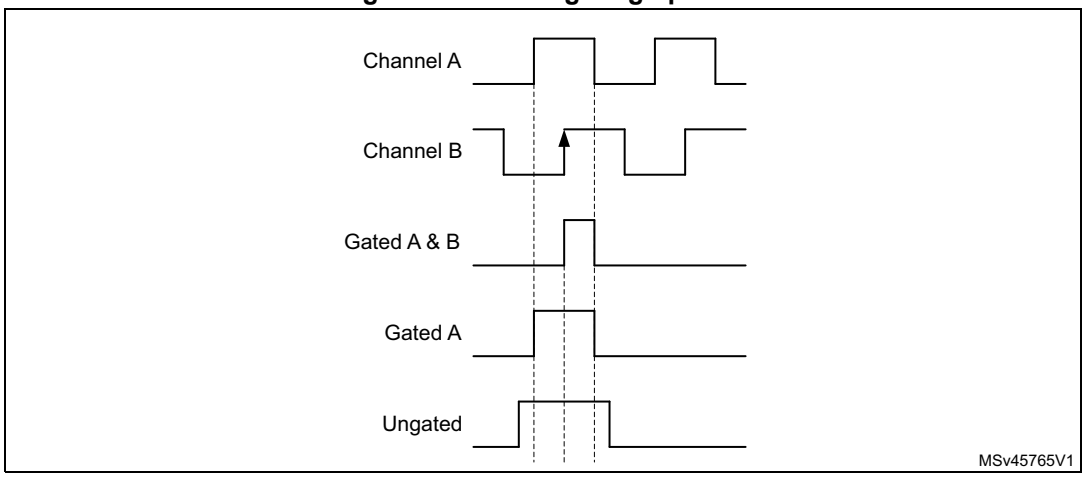
The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The Index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIMX_ECR register. The IE bit must be set only in encoder mode, when the SMS[3:0] bitfield has the following values: 0001, 0010, 011, 1010, 1011, 1100, 1101, 1110, 1111.

Commercially available encoders are proposed with several options for index pulse conditioning, as per the *Figure 440* below:

- gated with A and B: the pulsewidth is 1/4 of one channel period, aligned with both A and B edges
- gated with A (or gated with B): the pulsewidth is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B)
- ungated: the pulsewidth is up to one channel period, without any alignment to the edges

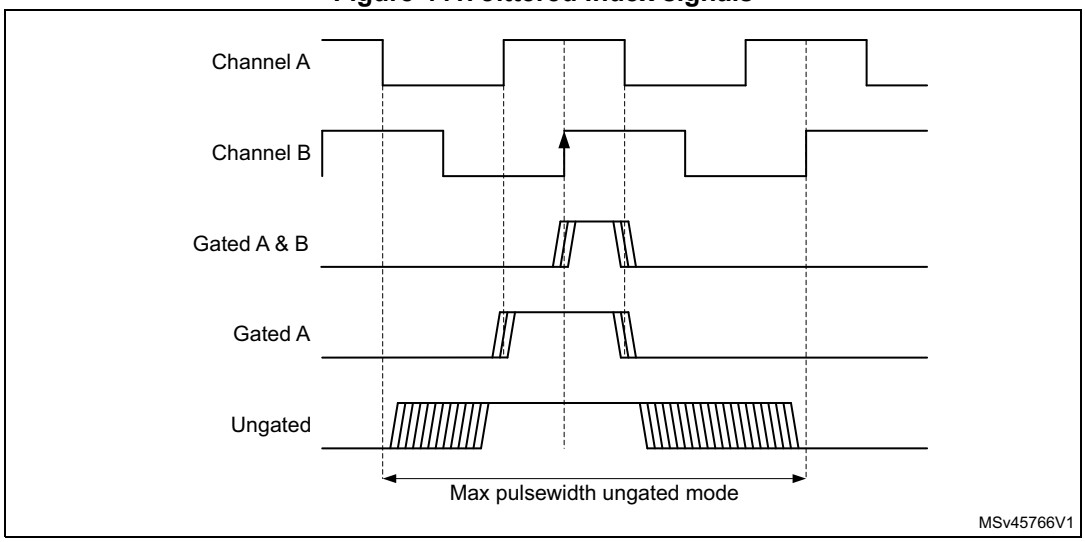
Figure 440. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as show on *Figure 441* below.

In ungated mode, the signal must be strictly below 2 encoder periods. If the pulsewidth is greater or equal to 2 encoder period, the counter is reset multiple times.

Figure 441. Jittered Index signals



The timer supports the 3 gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (i.e. channel A and channel B

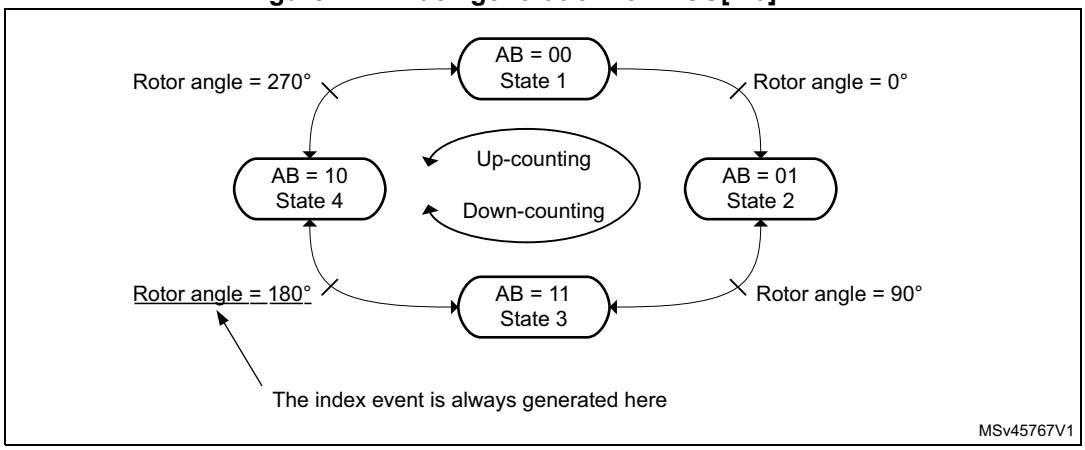
state combination) the index must be synchronized, using the IPOS[1:0] bitfield in the TIMx_ECR register.

The Index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0).
- The counter is set to TIMx_ARR when down counting.

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. The *Figure 442* below shows at which position is the index generated, for a simplistic example (an encoder providing 4 edges par mechanical rotation).

Figure 442. Index generation for IPOS[1:0] = 11



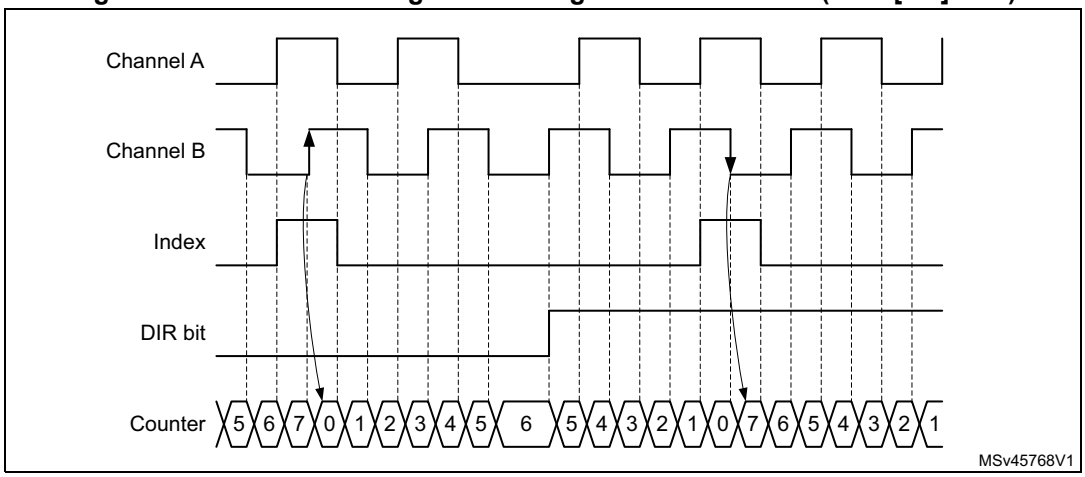
The *Figure 443* below presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is '11' (ChA=1, ChB=1), when up-counting (DIR bit = 0).
- Counter set to TIMx_ARR when exiting the '11' state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

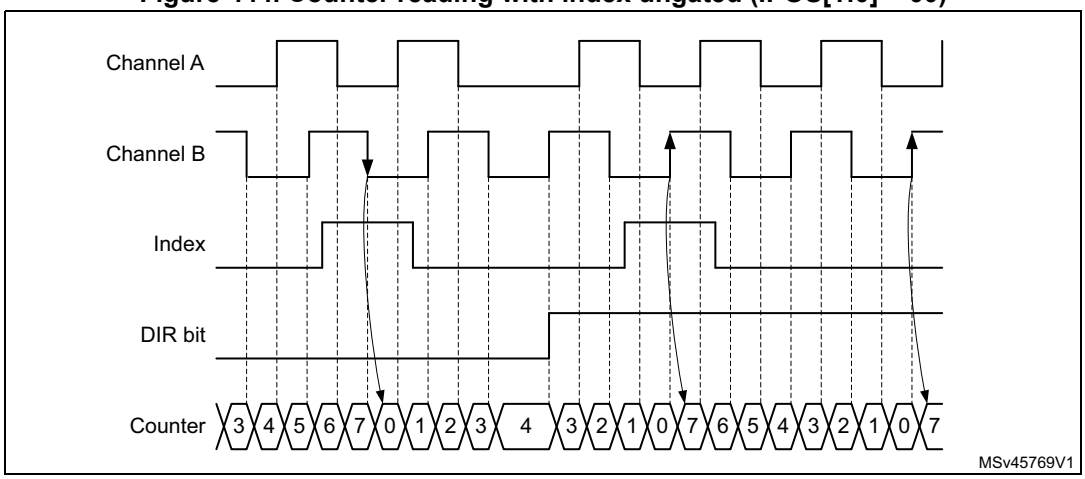
The arrows are indicating on which transition is the index event interrupt generated.

Figure 443. Counter reading with index gated on channel A (IPOS[1:0] = 11)



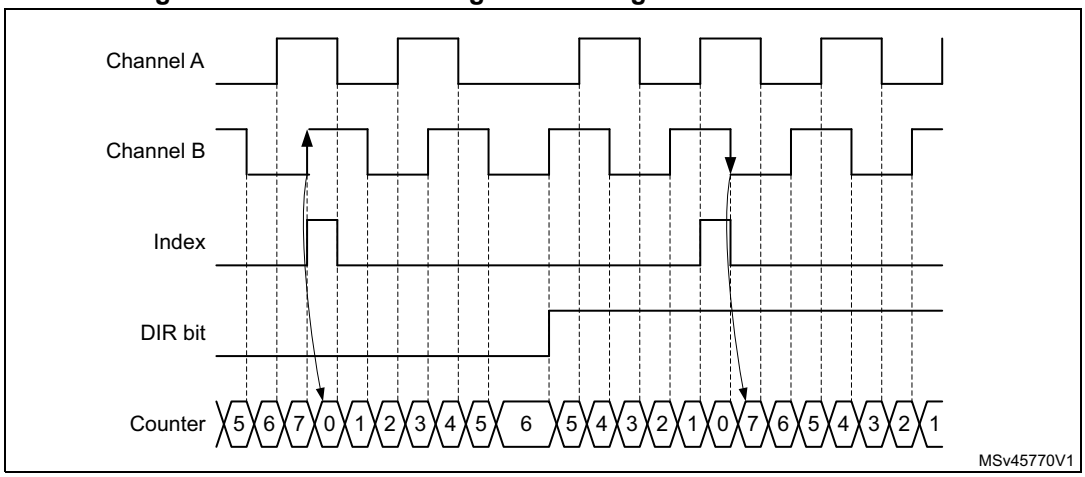
The *Figure 444.* below presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 444. Counter reading with index ungated (IPOS[1:0] = 00)



The *Figure 445.* below shows how the ‘gated on A & B’ mode is handled, for various pulse alignment scenario. The arrows are indicating on which transition is the index event generated.

Figure 445. Counter reading with index gated on channel A and B



The *Figure 446* and *Figure 447* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 446. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

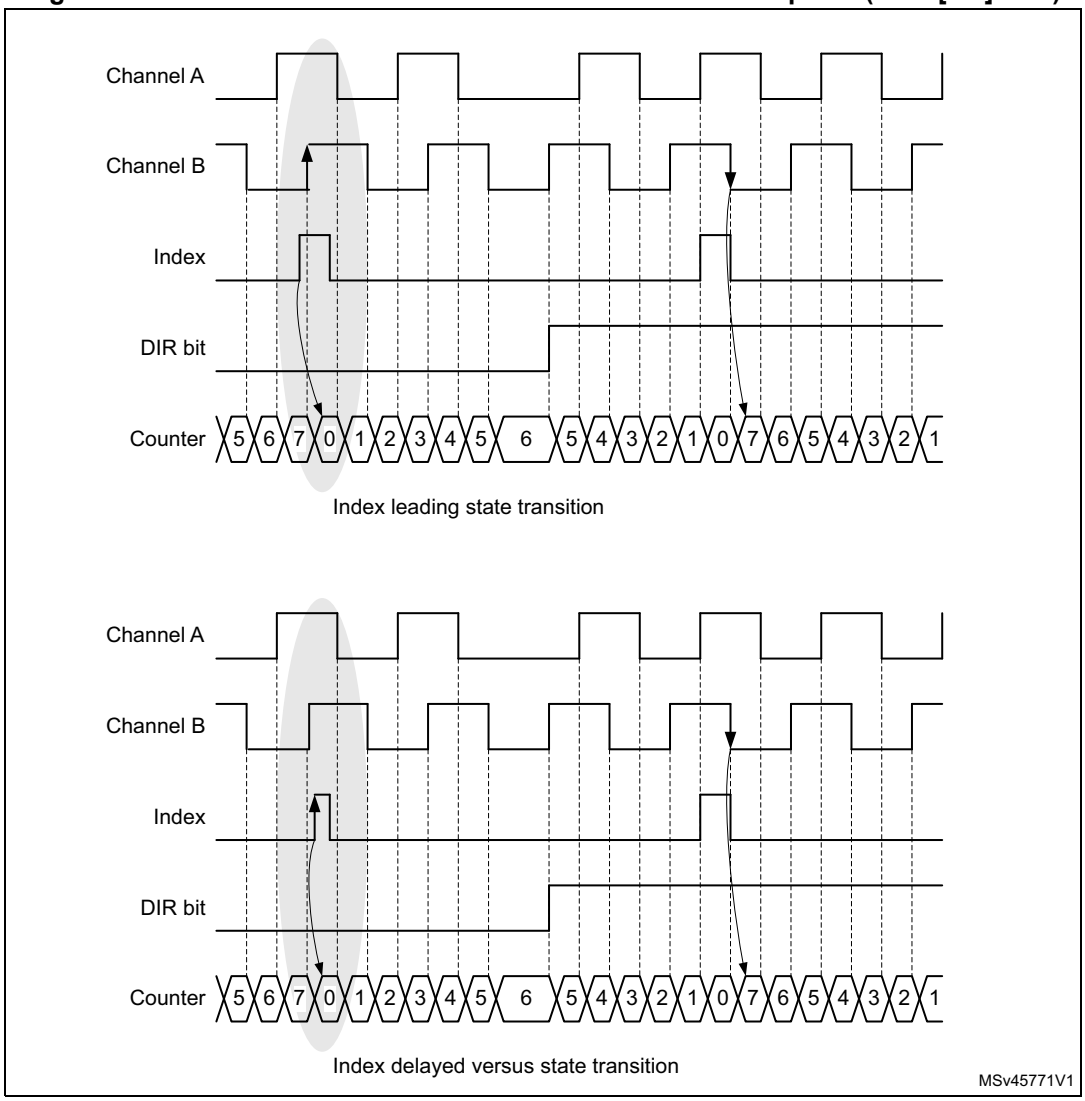
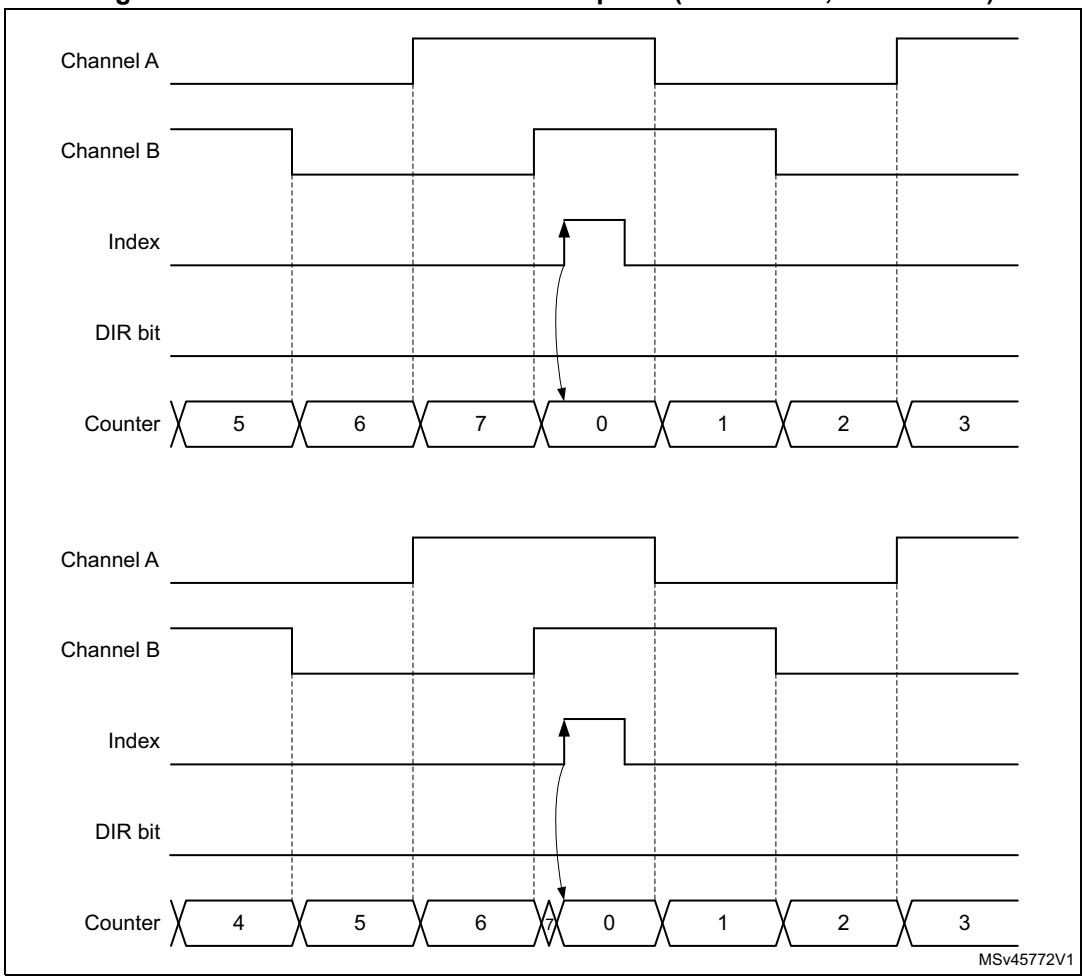
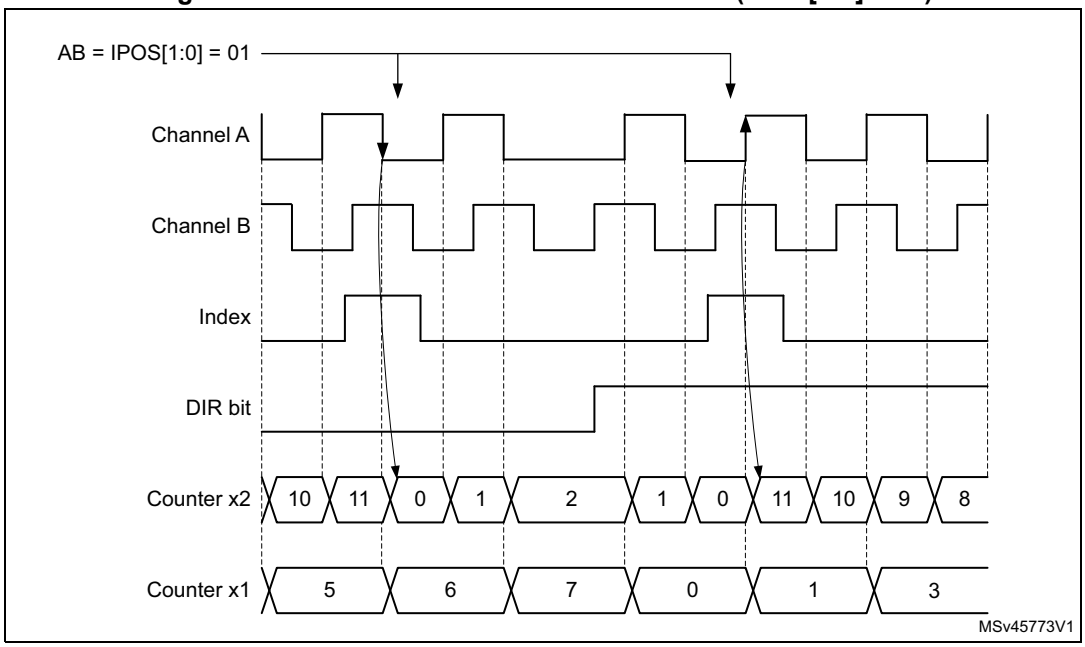


Figure 447. Counter reset Narrow index pulse (closer view, ARR = 0x07)



The *Figure 448* below shows how the index is managed in x1 and x2 modes.

Figure 448. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)



Directional index sensitivity

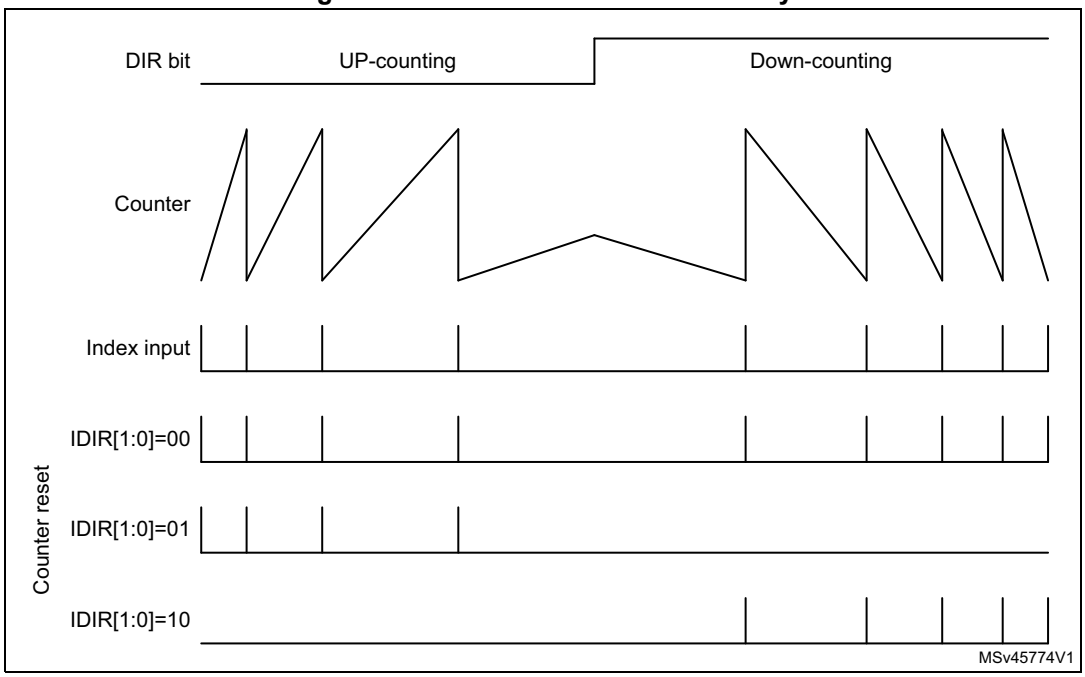
The IDIR[1:0] bitfield in the TIMx_ECR register allows the index to be active only in a selected counting direction.

The *Figure 449* below shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

Note: The IDR[1:0] bitfield must be written when IE bit is reset (index mode disabled).

Note: The directional index sensitivity is not supported in clock + direction mode. When SMS[3:0] = 1010 or 1011, the IDIR[1:0] must be set to 00.

Figure 449. Directional index sensitivity

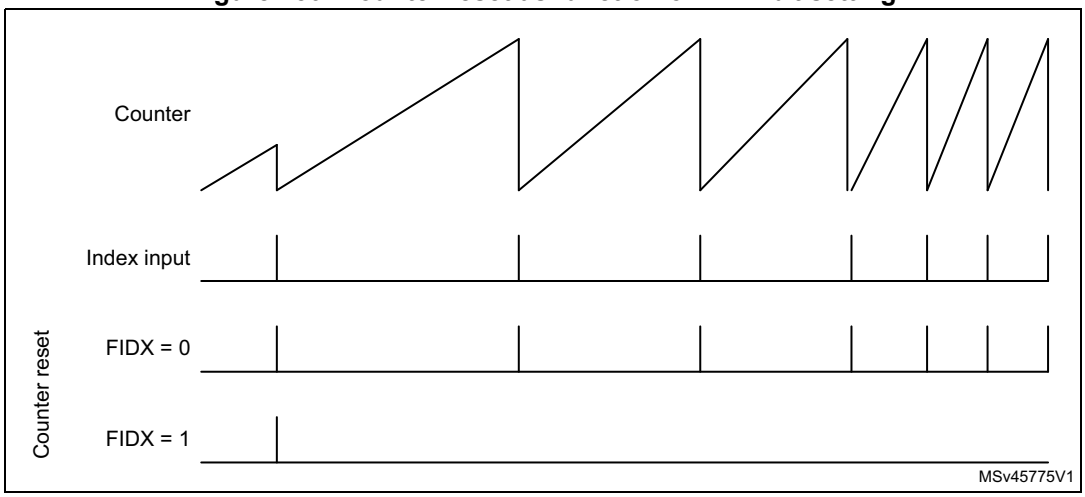


Special first index event management

The FIDX bit in the TIMx_ECR register allows the Index to be taken only once, as shown on the *Figure 450* below. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be re-armed by writing the FIDX bit to 0 and setting it again to 1.

Note: When FIDX = 1, the index can be issued twice (IDXF flag set) if the direction changes at position 0 (index active).

Figure 450. Counter reset as function of FIDX bit setting



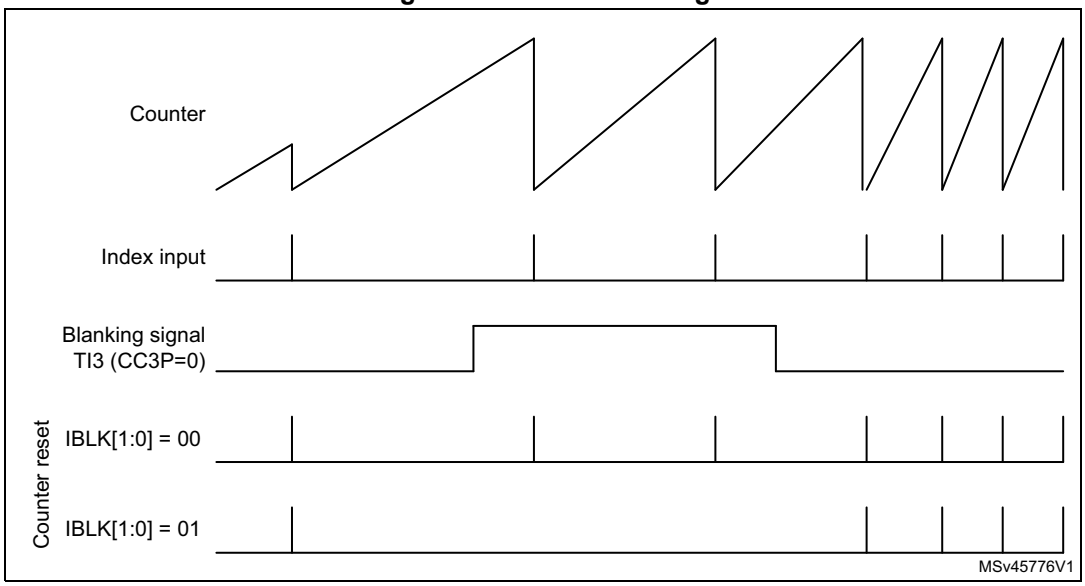
Index blanking

The Index event can be blanked using the tim_ti3 or tim_ti4 inputs. During the blanking window, the index events are no longer resetting the counter, as shown on the *Figure 451* below.

This mode is enabled using the IBLK[1:0] bitfield in the TIMx_ECR register, as following:

- IBLK[1:0] = 00: Index signal always active
- IBLK[1:0] = 01: Index signal blanking on tim_ti3 input
- IBLK[1:0] = 10: Index signal blanking on tim_ti4 input

Figure 451. Index blanking



Index management in non-quadrature mode

The *Figure 452* and *Figure 453* below detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[3:0] bitfield is equal to 1010, 1011, 1100, 1101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as following:

- IPOS[0] = 0: Index is detected on clock low level
- IPOS[0] = 1: Index is detected on clock high level

The IPOS[1] bit is not-significant.

Figure 452. Index behavior in clock + direction mode, IPOS[0] = 1

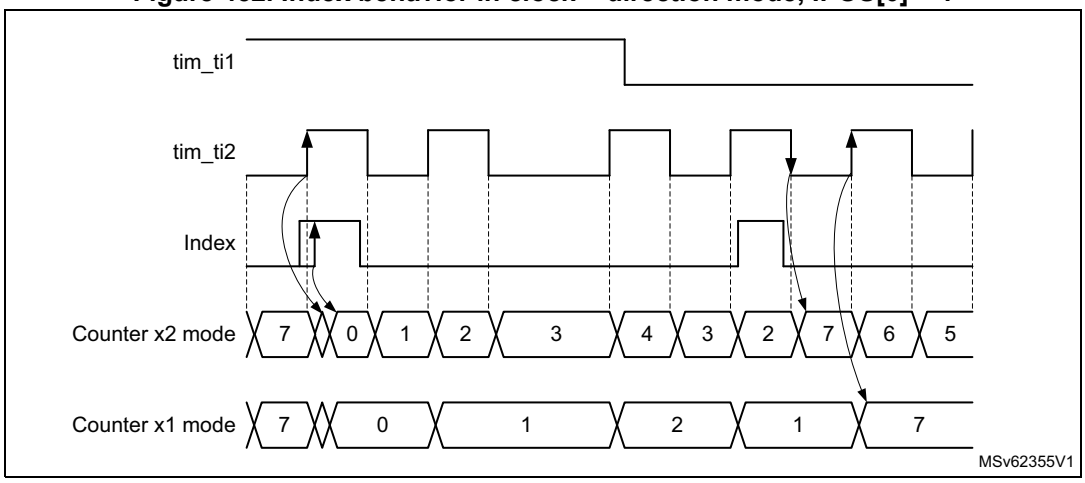
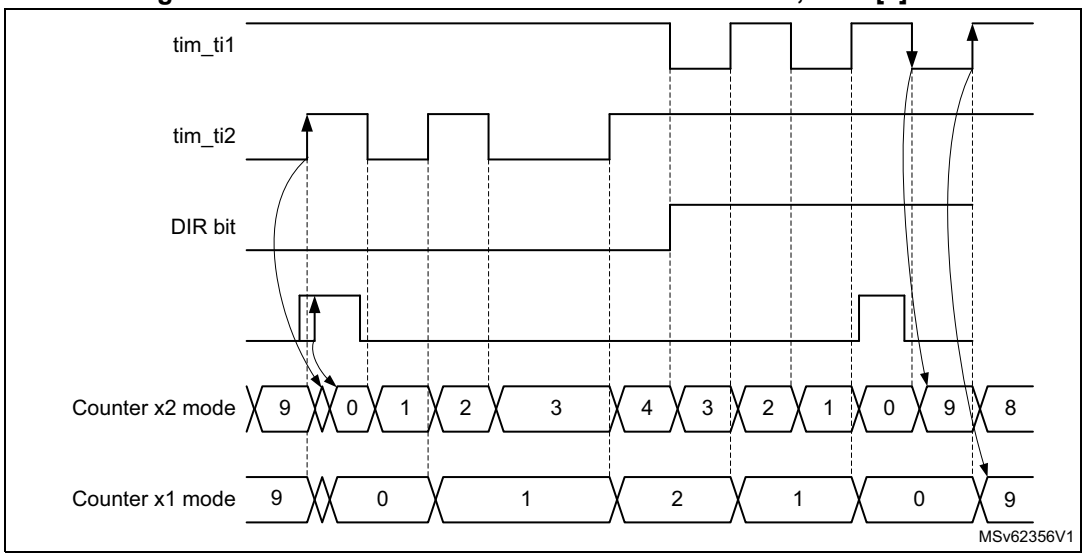


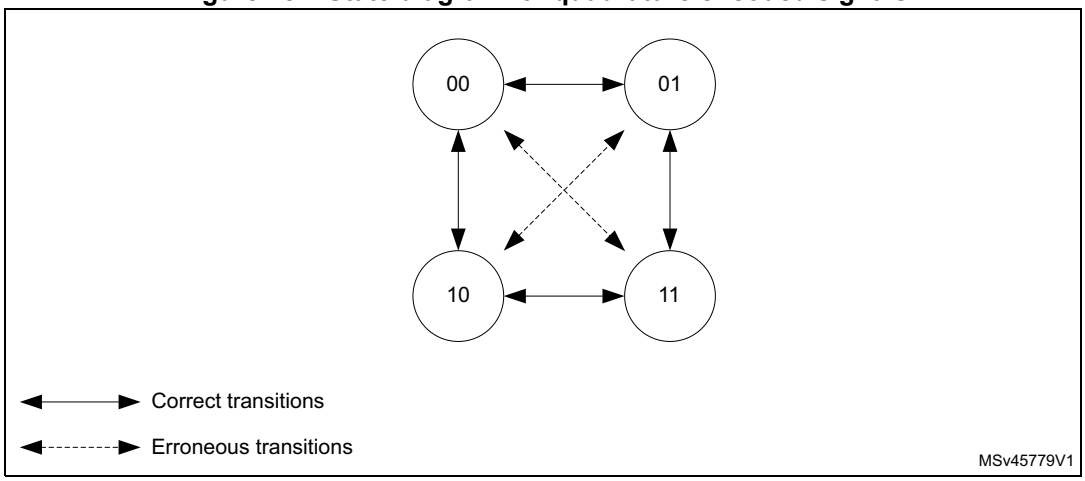
Figure 453. Index behavior in directional clock mode, IPOS[0] = 1



Encoder error management

For encoder configurations where 2 quadrature signals are available, it is possible to detect transition errors. The reading on the 2 inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on the *Figure 454.* below. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIMx_SR status register. A transition error interrupt is generated if the TERRIE bit is set in the TIMx_DIER register.

Figure 454. State diagram for quadrature encoded signals



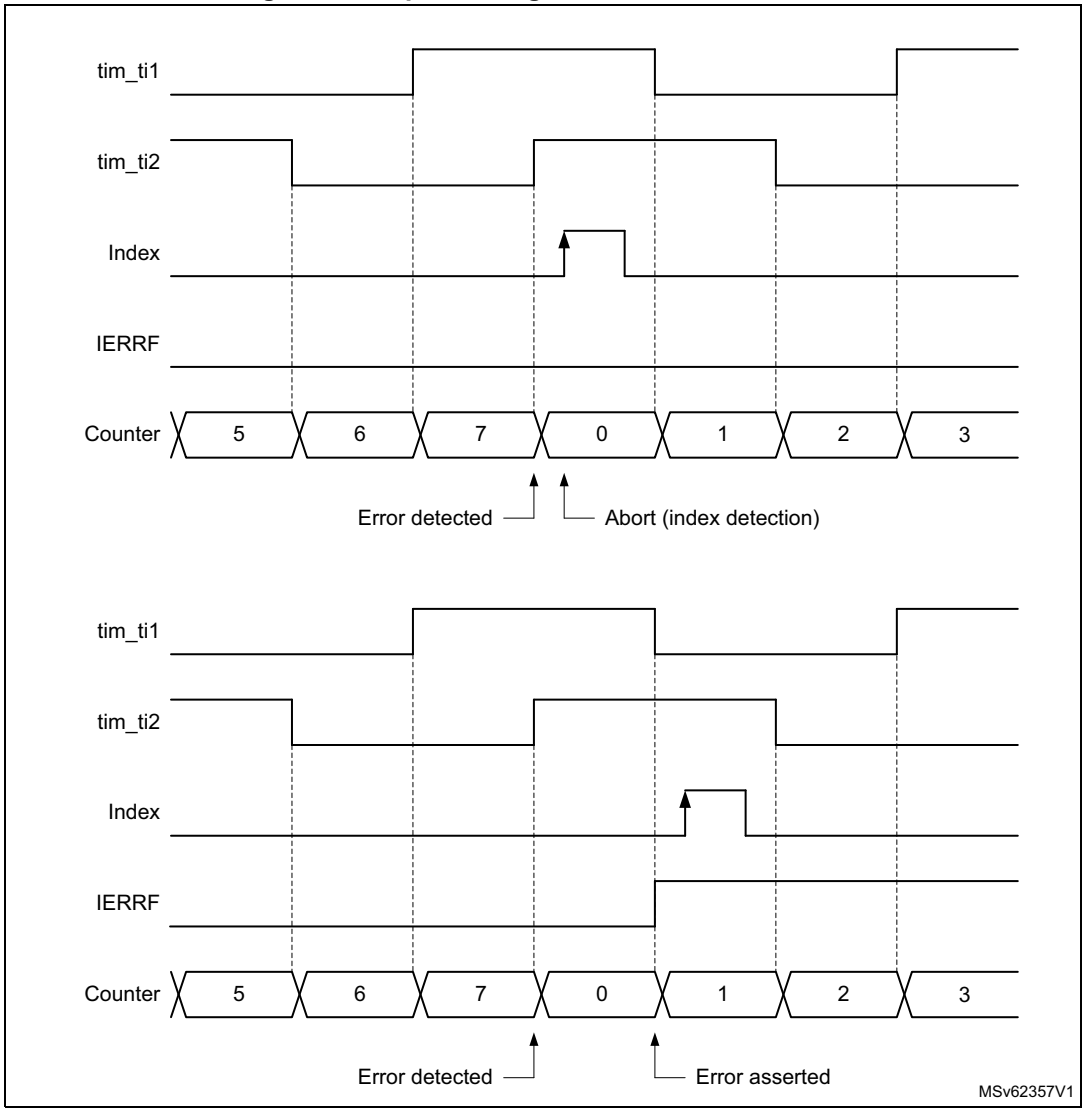
For encoder having an Index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The Index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIMx_ARR to 0 or decremented from 0 to TIMxARR value without any index event, this is reported as an Index position error.

The overflow threshold is programmed using the TIMx_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIMx_ARR = 3999 + 1 = 4000.

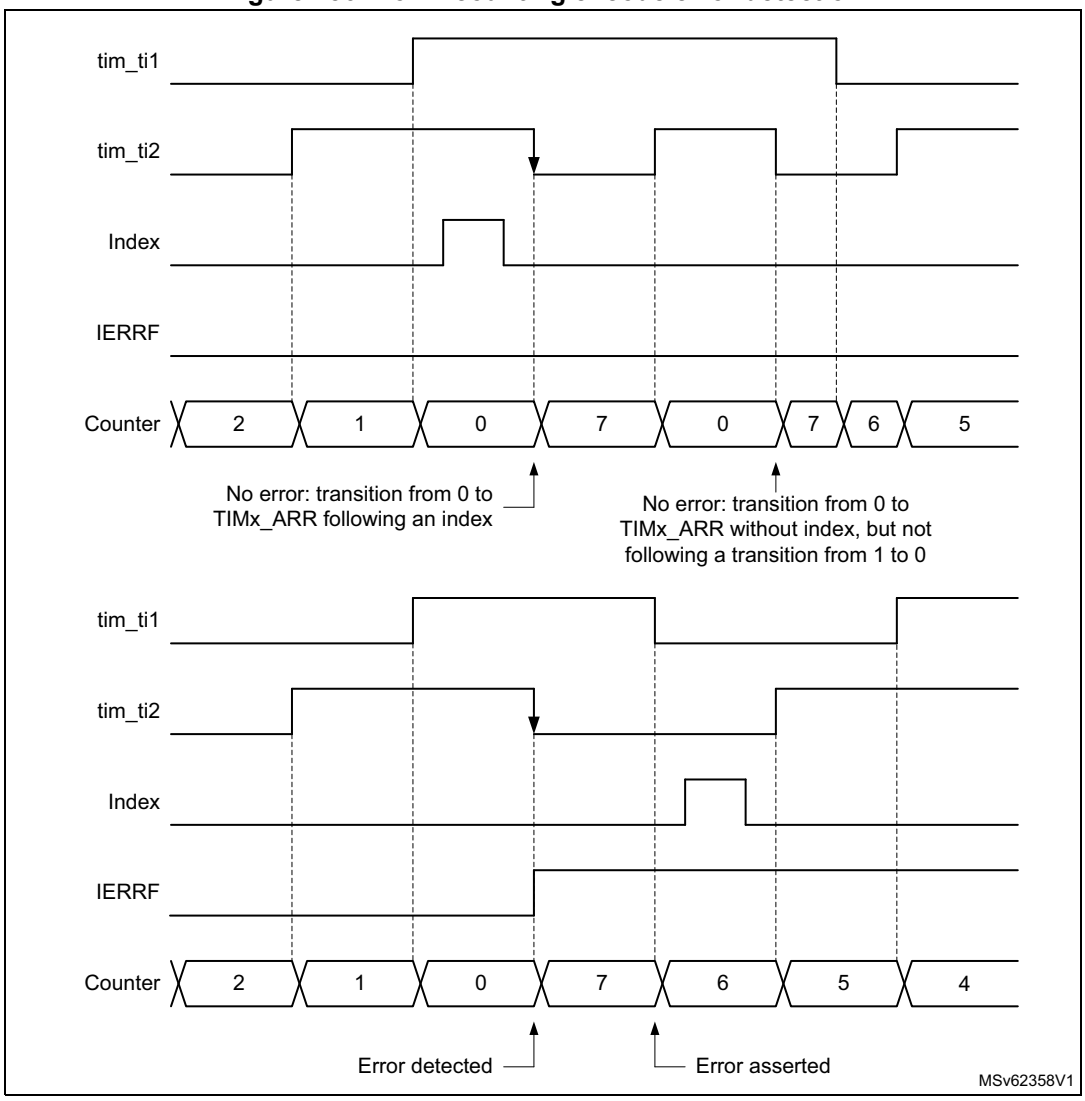
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is cope with narrow index pulses in gated A and B mode, as shown on *Figure 455* below.

Figure 455. Up-counting encoder error detection



In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 456* below, to avoid any false error detection in case the encoder dithers between TIMx_ARR and 0 immediately after the index detection.

Figure 456. Down-counting encode error detection



An index error sets the IERRF interrupt flag in the TIMx_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIMx_DIER register.

Functional encoder Interrupts

The following interrupts are also available in encoder mode

- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIMx_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIMx_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIMx_DIER register.
- Index event: the Index event sets the IDXF interrupt flag in the TIMx_SR status register. An Index interrupt is generated if the IDXIE bit is set in the TIMx_DIER register.

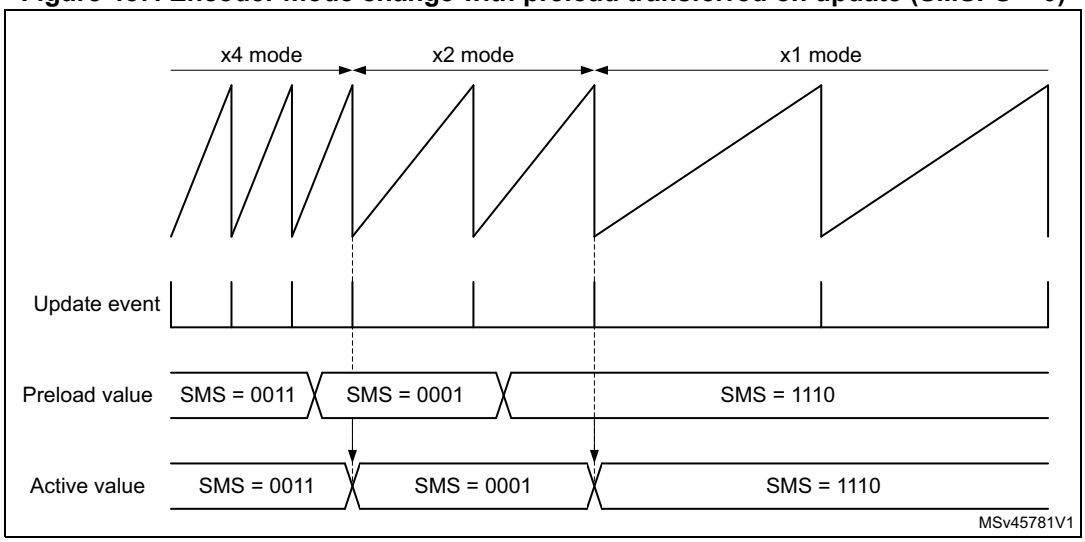
Slave mode selection preload for run-time encoder mode update

It may be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the Update interrupt rate, by switching from x4 to x2 to x1 mode, as show on the *Figure 457* below.

For this purpose, the SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value can be selected with the SMSPS bit in the TIMx_SMCR register.

- SMSPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when upcounting, and underflows when downcounting. This mode must be used only when index is disabled (bit IE = 0)
- SMSPS = 1: the transfer is triggered by the Index event

Figure 457. Encoder mode change with preload transferred on update (SMSPS = 0)



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIMx_CR2 register.It is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

46.3.26 Direction bit output

Its is possible to output a direction signal out of the timer, on the tim_oc3n and tim_oc4 output signals (copy of the DIR bit in the TIMx_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bit field to 1011 in the TIMx_CCMR2 register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

46.3.27 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. In particular cases, it can ease the calculations by avoiding race conditions, caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

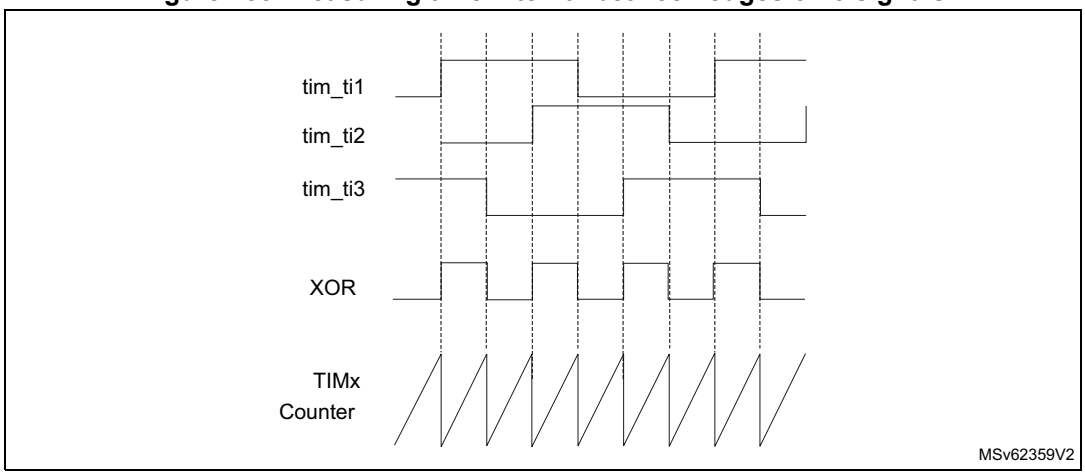
There is no latency between the UIF and UIFCPY flags assertion.

46.3.28 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins tim_ti1, tim_ti2 and tim_ti3.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is convenient to measure the interval between edges on two input signals, as per *Figure 458* below.

Figure 458. Measuring time interval between edges on 3 signals



46.3.29 Interfacing with Hall sensors

This is done using the advanced-control timers to generate PWM signals to drive the motor and another timer TIMx referred to as "interfacing timer" in *Figure 459*. The "interfacing timer" captures the 3 timer input pins (tim_ti1, tim_ti2 and tim_ti3) connected through a XOR to the tim_ti1 input channel (selected by setting the TI1S bit in the TIMx_CR2 register).

The slave mode controller is configured in reset mode; the slave input is tim_ti1f_ed. Thus, each time one of the 3 inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the "interfacing timer", capture/compare channel 1 is configured in capture mode, capture signal is tim_trc (See *Figure 401: Capture/compare channel (example: channel 1*

input stage) on page 1657). The captured value, which corresponds to the time elapsed between 2 changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (by triggering a COM event). The advanced-control timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer through the tim_trgo output.

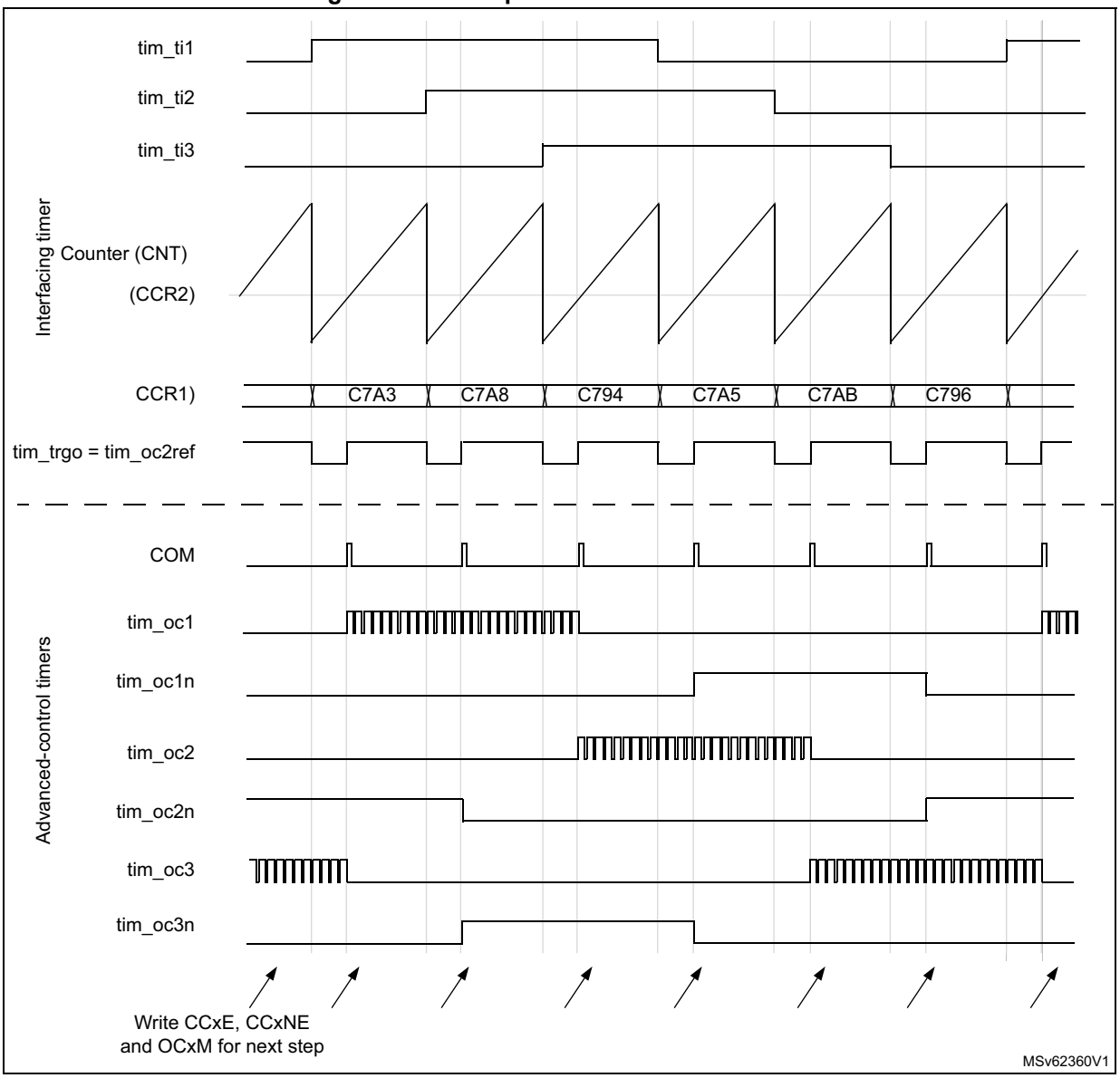
Example: one wants to change the PWM configuration of the advanced-control timer after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure 3 timer inputs ORed to the tim_ti1 input channel by writing the TI1S bit in the TIMx_CR2 register to ‘1’,
- Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the tim_ti1 change. Set the prescaler to get a maximum counter period longer than the time between 2 changes on the sensors,
- Program the channel 1 in capture mode (tim_trc selected): write the CC1S bits in the TIMx_CCMR1 register to ‘01’. The digital filter can also be programmed if needed,
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to ‘111’ and the CC2S bits to ‘00’ in the TIMx_CCMR1 register,
- Select tim_oc2ref as trigger output on tim_trgo: write the MMS bits in the TIMx_CR2 register to ‘101’,

In the advanced-control timer, the right tim_itrx input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC=1 in the TIMx_CR2 register) and the COM event is controlled by the trigger input (CCUS=1 in the TIMx_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of tim_oc2ref).

The *Figure 459* describes this example.

Figure 459. Example of Hall sensor interface



46.3.30 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 47.4.23: Timer synchronization* for details. They can be synchronized in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

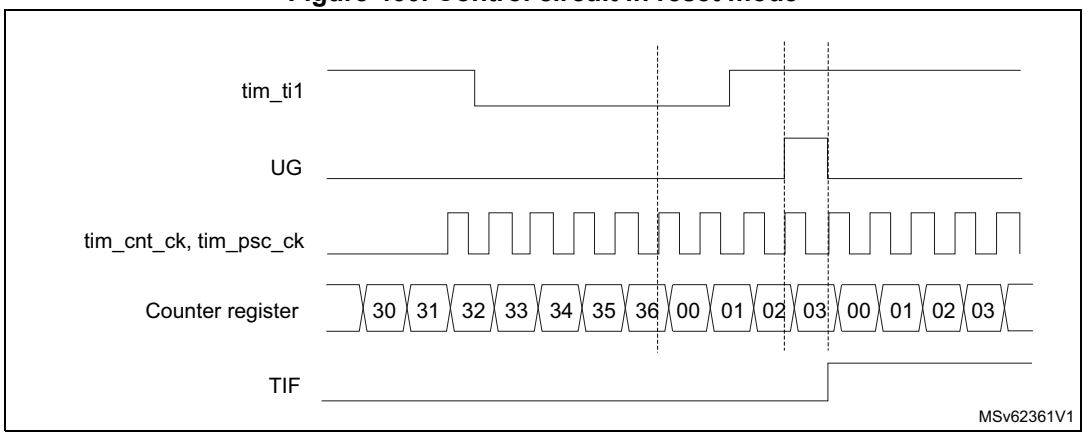
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

- Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.
- Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 460. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

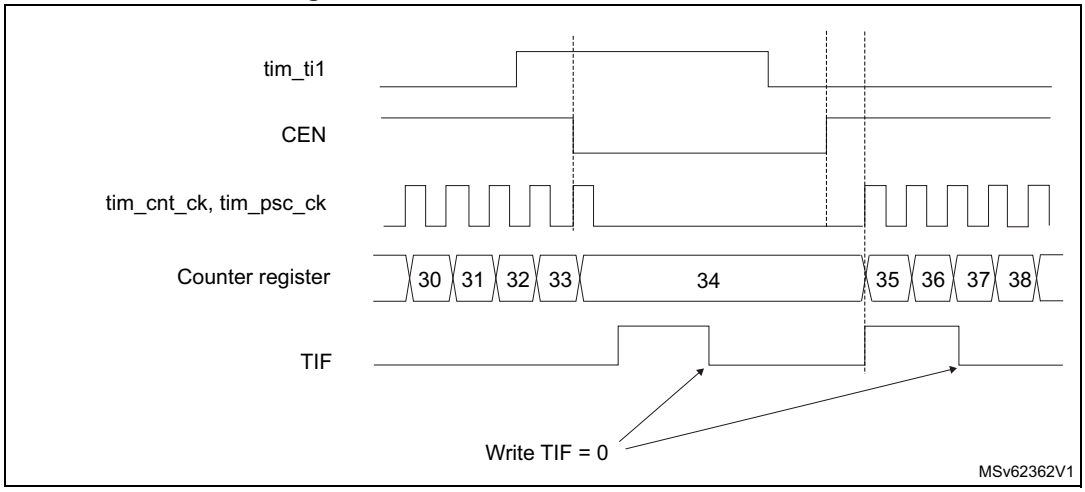
In the following example, the upcounter counts only when tim_ti1 input is low:

- Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.
- Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 461. Control circuit in Gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

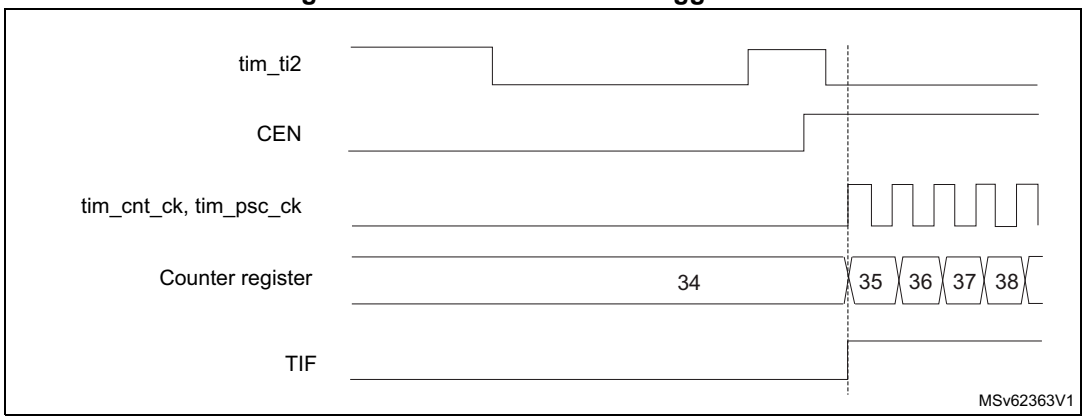
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

- Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select tim_ti2 as the input source by writing TS=00110 in TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 462. Control circuit in trigger mode



Slave mode: Combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for One-pulse mode.

Slave mode: Combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the tim_etr_in signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select tim_etr_in as tim_trgi through the TS bits of TIMx_SMCR register.

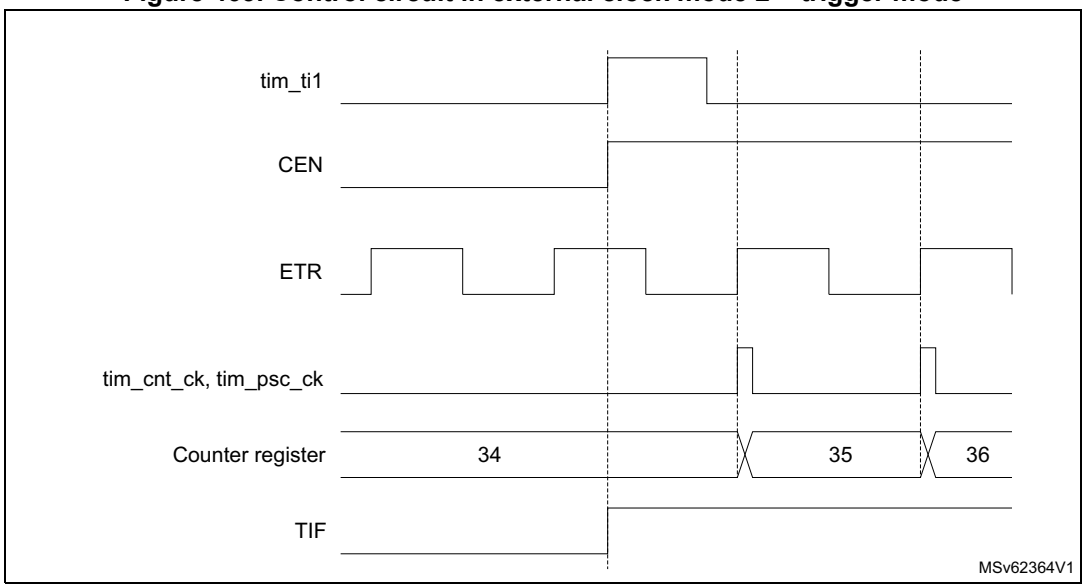
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS=00: prescaler disabled
   - ETP=0: detection of rising edges on tim_etr_in and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F=0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S=01in TIMx_CCMR1 register to select only the input capture source
   - CC1P=0 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.

A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.

Figure 463. Control circuit in external clock mode 2 + trigger mode



Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

46.3.31 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events. It is also possible to generate a pulse issued by internal edge detectors, such as:

- Rising and falling edges of OC4ref
- Rising edge on OC5ref or falling edge on OC6ref

The triggers are issued on the tim_trgo2 internal line which is redirected to the ADC. There is a total of 16 possible events, which can be selected using the MMS2[3:0] bits in the TIMx_CR2 register.

An example of an application for 3-phase motor drives is given in *Figure 416 on page 1675*.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Note: The clock of the ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the timer.

46.3.32 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR register define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 46.6.29: TIMx DMA control register (TIMx_DCR)(x = 1, 8)* for details).

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 and DBSS = 1.
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.

46.3.33 TIM1/TIM8 DMA requests

The TIM1/TIM8 can generate a DMA request, as shown in the table below.

Table 442. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_cc2_dma	Capture/compare 2	CC2DE
tim_cc3_dma	Capture/compare 3	CC3DE
tim_cc4_dma	Capture/compare 4	CC4DE
tim_com_dma	Commutation (COM)	COMDE
tim_trg_dma	Trigger	TDE

46.3.34 Debug mode

When the microcontroller enters debug mode (Cortex®-M33 core halted), the TIMx counter can either continue to work normally or stop, depending on DBG_TIMx_STOP configuration bit in DBG module.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0), typically to force a Hi-Z.

For more details, refer to section Debug support (DBG).

46.4 TIM1/TIM8 low-power modes

Table 443. Effect of low-power modes on TIM1/TIM8

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

46.5 TIM1/TIM8 interrupts

The TIM1/TIM8 can generate multiple interrupts, as shown in *Table 444*.

Table 444. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM_UP	Update	UIF	UIE	write 0 in UIF	Yes	No
TIM_CC	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
	Capture/compare 2	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Capture/compare 3	CC3IF	CC3IE	write 0 in CC3IF	Yes	No
	Capture/compare 4	CC4IF	CC4IE	write 0 in CC4IF	Yes	No
TIM_TRG_COM	Commutation (COM)	COMIF	COMIE	write 0 in COMIF	Yes	No
	Trigger	TIF	TIE	write 0 in TIF	Yes	No
TIM_DIR_IDX	Index	IDXF	IDXIE	write 0 in IDXF	Yes	No
	Direction	DIRF	DIRIE	write 0 in DIRF	Yes	No
TIM_BRK	Break	BIF	BIE	write 0 in BIF	Yes	No
	Break2	B2IF		write 0 in B2IF	Yes	No
	System Break	SBIF		write 0 in SBIF	Yes	No
TIM_IERR	Index Error	IERRF	IERRIE	write 0 in IERRF	Yes	No
TIM_TER	Transition Error	TERRF	TERRIE	write 0 in TERRF	Yes	No

46.6 TIM1/TIM8 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

46.6.1 TIMx control register 1 (TIMx_CR1)(x = 1, 8)

Address offset: 0x000

Reset value: 0x0000

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

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bit-field indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (tim_etr_in, tim_tix),

00: $t_{DTS}=t_{tim\_ker\_ck}$
01: $t_{DTS}=2*t_{tim\_ker\_ck}$
10: $t_{DTS}=4*t_{tim\_ker\_ck}$
11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter
Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

46.6.2 TIMx control register 2 (TIMx_CR2)(x = 1, 8)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	MMS[3]	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5
						rw		rw	rw	rw	rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OIS4N	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 24 Reserved, must be kept at reset value.

Bits 23:20 **MMS2[3:0]**: Master mode selection 2

These bits allow the information to be sent to ADC for synchronization (tim_trgo2) to be selected. The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on tim_trgo2 is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo2). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between the CEN control bit and the trigger input when configured in Gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo2, except if the Master/Slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - the update event is selected as trigger output (tim_trgo2). For instance, a master timer can then be used as a prescaler for a slave timer.

0011: **Compare pulse** - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (tim_trgo2).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo2)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo2)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo2)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo2)

1000: **Compare** - tim_oc5refc signal is used as trigger output (tim_trgo2)

1001: **Compare** - tim_oc6refc signal is used as trigger output (tim_trgo2)

1010: **Compare Pulse** - tim_oc4refc rising or falling edges generate pulses on tim_trgo2

1011: **Compare pulse** - tim_oc6refc rising or falling edges generate pulses on tim_trgo2

1100: **Compare pulse** - tim_oc4refc or tim_oc6refc rising edges generate pulses on tim_trgo2

1101: **Compare pulse** - tim_oc4refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2

1110: **Compare pulse** - tim_oc5refc or tim_oc6refc rising edges generate pulses on tim_trgo2

1111: **Compare pulse** - tim_oc5refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **OIS6**: Output idle state 6 (tim_oc6 output)
Refer to OIS1 bit

Bit 17 Reserved, must be kept at reset value.

Bit 16 **OIS5**: Output idle state 5 (tim_oc5 output)
Refer to OIS1 bit

Bit 15 **OIS4N**: Output idle state 4 (tim_oc4n output)
Refer to OIS1N bit

Bit 14 **OIS4**: Output idle state 4 (tim_oc4 output)
Refer to OIS1 bit

Bit 13 **OIS3N**: Output idle state 3 (tim_oc3n output)
Refer to OIS1N bit

Bit 12 **OIS3**: Output idle state 3 (tim_oc3n output)
Refer to OIS1 bit

Bit 11 **OIS2N**: Output idle state 2 (tim_oc2n output)
Refer to OIS1N bit

Bit 10 **OIS2**: Output idle state 2 (tim_oc2 output)
Refer to OIS1 bit

Bit 9 **OIS1N**: Output idle state 1 (tim_oc1n output)
0: tim_oc1n=0 after a dead-time when MOE=0
1: tim_oc1n=1 after a dead-time when MOE=0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **OIS1**: Output idle state 1 (tim_oc1 output)
0: tim_oc1=0 (after a dead-time) when MOE=0
1: tim_oc1=1 (after a dead-time) when MOE=0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **TI1S**: tim_ti1 selection
0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input
1: tim_ti1_in[15:0], tim_ti2_in[15:0] and tim_ti3_in[15:0] multiplexers outputs are XORed and connected to the tim_ti1 input

Bits 25, 6:4 **MMS[3:0]**: Master mode selection

These bits select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

0011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)

1000: **Encoder Clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

Other codes reserved

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only

1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on tim_trgi

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on tim_trgi, depending on the CCUS bit).

Note: This bit acts only on channels that have a complementary output.

46.6.3 TIMx slave mode control register (TIMx_SMCR)(x = 1, 8)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
						rw	rw			rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **SMSPS**: SMS preload source

This bit selects whether the events that triggers the SMS[3:0] bitfield transfer from preload to active

0: The transfer is triggered by the Timer's Update event

1: The transfer is triggered by the Index event

Bit 24 **SMSPE**: SMS preload enable

This bit selects whether the SMS[3:0] bitfield is preloaded

0: SMS[3:0] bitfield is not preloaded

1: SMS[3:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **TS[4:3]**: Trigger selection - bit 4:3

Refer to TS[2:0] description - bits 6:4

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

Bit 15 **ETP**: External trigger polarity

This bit selects whether tim_etr_in or tim_$\overline{\text{etr_in}}$ is used for trigger operations

0: tim_etr_in is non-inverted, active at high level or rising edge.

1: tim_etr_in is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the tim_etrf signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_etrf (SMS=111 and TS=00111).

It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tim_trgi must not be connected to tim_etrf in this case (TS bits must not be 00111).

If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is tim_etrf.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal tim_etrp frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etr_in.

00: Prescaler OFF
01: tim_etr_in frequency divided by 2
10: tim_etr_in frequency divided by 4
11: tim_etr_in frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample tim_etrp signal and the length of the digital filter applied to tim_etrp. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bit 7 **MSM:** Master/slave mode

0: No action
1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.

Bits 6:4 **TS[2:0]:** Trigger selection

This bitfield is combined with TS[4:3] bits.

This bit-field selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (tim_itr0)
00001: Internal Trigger 1 (tim_itr1)
00010: Internal Trigger 2 (tim_itr2)
00011: Internal Trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: External Trigger input (tim_etrf)
01000: Internal Trigger 4 (tim_itr4)
01001: Internal Trigger 5 (tim_itr5)
01010: Internal Trigger 6 (tim_itr6)
01011: Internal Trigger 7 (tim_itr7)
01100: Internal Trigger 8 (tim_itr8)
01101: Internal Trigger 9 (tim_itr9)
01110: Internal Trigger 10 (tim_itr10)
01111: Internal trigger 11 (tim_itr11)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
Others: Reserved

See *Table 430: TIMx internal trigger connection* for more details on tim_itrx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS:** OCREF clear selection

This bit is used to select the OCREF clear source.

0: tim_ocref_clr_int is connected to the tim_ocref_clr input
1: tim_ocref_clr_int is connected to tim_etrf

Bits 16, 2:0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

0000: Slave mode disabled - if CEN = ‘1’ then the prescaler is clocked directly by the internal clock.

0001: Quadrature encoder mode 1, x2 mode- Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

0010: Quadrature encoder mode 2, x2 mode - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

0011: Quadrature encoder mode 3, x4 mode - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

0100: Reset Mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

1010: Encoder mode: Clock plus direction, x2 mode.

1011: Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P

1100: Encoder mode: Directional Clock, x2 mode.

1101: Encoder mode: Directional Clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

1110: Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

1111: Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS=00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

46.6.4 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 1, 8)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 23 **TERRIE**: Transition error interrupt enable
0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 **IERRIE**: Index error interrupt enable
0: Index error interrupt disabled
1: Index error interrupt enabled

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction Change interrupt disabled
1: Direction Change interrupt enabled

Bit 20 **IDXIE**: Index interrupt enable
0: Index interrupt disabled
1: Index Change interrupt enabled

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable
0: COM DMA request disabled
1: COM DMA request enabled

Bit 12 **CC4DE**: Capture/compare 4 DMA request enable
0: CC4 DMA request disabled
1: CC4 DMA request enabled

Bit 11 **CC3DE**: Capture/compare 3 DMA request enable
0: CC3 DMA request disabled
1: CC3 DMA request enabled

Bit 10 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bit 4 **CC4IE**: Capture/compare 4 interrupt enable
0: CC4 interrupt disabled
1: CC4 interrupt enabled

Bit 3 **CC3IE**: Capture/compare 3 interrupt enable
0: CC3 interrupt disabled
1: CC3 interrupt enabled

Bit 2 **CC2IE**: Capture/compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

46.6.5 TIMx status register (TIMx_SR)(x = 1, 8)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	CC6IF	CC5IF
								rc_w0	rc_w0	rc_w0	rc_w0			rc_w0	rc_w0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

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

Bit 23 **TERRF**: Transition error interrupt flag
This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to ‘0’.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 **IERRF**: Index error interrupt flag
This flag is set by hardware when an index error is detected. It is cleared by software by writing it to ‘0’.
0: No index error has been detected.
1: An index error has been detected

Bit 21 **DIRF**: Direction change interrupt flag
This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIMx_CR is changing). It is cleared by software by writing it to ‘0’.
0: No direction change
1: Direction change

Bit 20 **IDXF**: Index interrupt flag
This flag is set by hardware when an index event is detected. It is cleared by software by writing it to ‘0’.
0: No index event occurred.
1: An index event has occurred

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC6IF**: Compare 6 interrupt flag
Refer to CC1IF description
Note: Channel 6 can only be configured as output.

Bit 16 **CC5IF**: Compare 5 interrupt flag
Refer to CC1IF description
Note: Channel 5 can only be configured as output.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **SBIF**: System break interrupt flag
This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.
This flag must be reset to re-start PWM operation.
0: No break event occurred.
1: An active level has been detected on the system break input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 12 **CC4OF**: Capture/compare 4 overcapture flag
Refer to CC1OF description

Bit 11 **CC3OF**: Capture/compare 3 overcapture flag
Refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.

0: No overcapture has been detected.

1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 **B2IF**: Break 2 interrupt flag

This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is not active.

0: No break event occurred.

1: An active level has been detected on the break 2 input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred.

1: An active level has been detected on the break input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred.

1: Trigger interrupt pending.

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on COM event (when capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.

0: No COM event occurred.

1: COM interrupt pending.

Bit 4 **CC4IF**: Capture/compare 4 interrupt flag

Refer to CC1IF description

Bit 3 **CC3IF**: Capture/compare 3 interrupt flag

Refer to CC1IF description

Bit 2 **CC2IF**: Capture/compare 2 interrupt flag
Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in downcounting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
–At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.
–When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.
–When CNT is reinitialized by a trigger event (refer to *Section 46.6.3: TIMx slave mode control register (TIMx_SMCR)(x = 1, 8)*), if URS=0 and UDIS=0 in the TIMx_CR1 register.

46.6.6 TIMx event generation register (TIMx_EGR)(x = 1, 8)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
							w	w	w	w	w	w	w	w	w

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

Bit 8 **B2G**: Break 2 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break 2 event is generated. MOE bit is cleared and B2IF flag is set. Related interrupt can occur if enabled.

Bit 7 **BG**: Break generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/compare control update generation

This bit can be set by software, it is automatically cleared by hardware

0: No action

1: CCxE, CCxNE and OCxM bits update (providing CCPC bit is set)

Note: This bit acts only on channels having a complementary output.

Bit 4 **CC4G**: Capture/compare 4 generation

Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation

Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation

Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

46.6.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 1, 8)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode:

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

46.6.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 1, 8)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE:** Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bit 7 **OC1CE:** Output compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr_int signal

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (tim_oc1ref=‘0’) as long as TIMx_CNT>TIMx_CCR1 else active (tim_oc1ref=’1’).

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Asymmetric PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=’00’ (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Note: On channels having a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

46.6.9 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 1, 8)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 3 in input capture mode and channel 4 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

46.6.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 1, 8)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 3 in input capture mode and channel 4 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode
Refer to OC3M[3:0] bit description

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bit 7 **OC3CE:** Output compare 3 clear enable

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode

These bits define the behavior of the output reference signal tim_oc3ref from which tim_oc3 and tim_oc3n are derived. tim_oc3ref is active high whereas tim_oc3 and tim_oc3n active level depends on CC3P and CC3NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR3 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 3 to active level on match. tim_oc3ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0010: Set channel 3 to inactive level on match. tim_oc3ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0011: Toggle - tim_oc3ref toggles when TIMx_CNT=TIMx_CCR3.

0100: Force inactive level - tim_oc3ref is forced low.

0101: Force active level - tim_oc3ref is forced high.

0110: PWM mode 1 - In upcounting, channel 3 is active as long as TIMx_CNT<TIMx_CCR3 else inactive. In downcounting, channel 3 is inactive (tim_oc3ref=‘0’) as long as TIMx_CNT>TIMx_CCR3 else active (tim_oc3ref=’1’).

0111: PWM mode 2 - In upcounting, channel 3 is inactive as long as TIMx_CNT<TIMx_CCR3 else active. In downcounting, channel 3 is active as long as TIMx_CNT>TIMx_CCR3 else inactive.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Pulse on compare: a pulse is generated on tim_oc3ref upon CCR3 match event, as per PWPRSC[2:0] and PW[7:0] bitfields programming in TIMxECR.

1011: Direction output. The tim_oc3ref signal is overridden by a copy of the DIR bit.

1100: Combined PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc is the logical OR between tim_oc3ref and tim_oc4ref.

1101: Combined PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc is the logical AND between tim_oc3ref and tim_oc4ref.

1110: Asymmetric PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=’00’ (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

On channels having a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC3M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3

10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4

11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

46.6.11 TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CC6P**: Capture/compare 6 output polarity
Refer to CC1P description

Bit 20 **CC6E**: Capture/compare 6 output enable
Refer to CC1E description

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC5P**: Capture/compare 5 output polarity
Refer to CC1P description

Bit 16 **CC5E**: Capture/compare 5 output enable
Refer to CC1E description

Bit 15 **CC4NP**: Capture/compare 4 complementary output polarity
Refer to CC1NP description

Bit 14 **CC4NE**: Capture/compare 4 complementary output enable
Refer to CC1NE description

Bit 13 **CC4P**: Capture/compare 4 output polarity
Refer to CC1P description

Bit 12 **CC4E**: Capture/compare 4 output enable
Refer to CC1E description

Bit 11 **CC3NP**: Capture/compare 3 complementary output polarity
Refer to CC1NP description

Bit 10 **CC3NE**: Capture/compare 3 complementary output enable
Refer to CC1NE description

Bit 9 **CC3P**: Capture/compare 3 output polarity
Refer to CC1P description

Bit 8 **CC3E**: Capture/compare 3 output enable
Refer to CC1E description

Bit 7 **CC2NP**: Capture/compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 **CC2NE**: Capture/compare 2 complementary output enable
Refer to CC1NE description

Bit 5 **CC2P**: Capture/compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/compare 1 complementary output polarity
CC1 channel configured as output:
0: tim_oc1n active high.
1: tim_oc1n active low.
CC1 channel configured as input:
This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1 and tim_ti2fp1. Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (channel configured as output).

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/compare 1 complementary output enable
0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 1 **CC1P**: Capture/compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP=1, CC1P=0: the configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 445* 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 445. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature

Control bits					Output states[1]	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	tim_ocx output state	tim_ocxn output state
1	X	X	0	0	Output disabled (not driven by the timer: Hi-Z) tim_ocx=0, tim_ocxn=0	
		0	0	1	Output disabled (not driven by the timer: Hi-Z) tim_ocx=0	tim_ocxref + Polarity tim_ocxn = tim_ocxref xor CCxNP
		0	1	0	tim_ocxref + Polarity tim_ocx=tim_ocxref xor CCxP	Output Disabled (not driven by the timer: Hi-Z) tim_ocxn=0
		X	1	1	OCREF + Polarity + dead-time	Complementary to OCREF (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) tim_ocx=CCxP	tim_ocxref + Polarity tim_ocxn = tim_ocxref x or CCxNP
		1	1	0	tim_ocxref + Polarity tim_ocx=tim_ocxref xor CCxP	Off-State (output enabled with inactive state) tim_ocxn=CCxNP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z).	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: tim_ocx=CCxP, tim_ocxn=CCxNP (if tim_brk or tim_brk2 is triggered). Then (this is valid only if tim_brk is triggered), if the clock is present: tim_ocx=OISx and tim_ocxn=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and tim_ocxn both in active state (may cause a short circuit when driving switches in half-bridge configuration). *Note: tim_brk2 can only be used if OSSI = OSSR = 1.*	
			1	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 tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and the GPIO registers.

46.6.12 TIMx counter (TIMx_CNT)(x = 1, 8)

Address offset: 0x024

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in the TIMxCR1 is reset, bit 31 is reserved and read at 0.

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

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

46.6.13 TIMx prescaler (TIMx_PSC)(x = 1, 8)

Address offset: 0x028

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency ($f_{tim\_cnt\_ck}$) is equal to $f_{tim\_psc\_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

46.6.14 TIMx auto-reload register (TIMx_ARR)(x = 1, 8)

Address offset: 0x02C

Reset value: 0x0000 FFFF

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

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

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

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 46.3.3: Time-base unit on page 1637* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

46.6.15 TIMx repetition counter register (TIMx_RCR)(x = 1, 8)

Address offset: 0x030

Reset value: 0x0000

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

Bits 15:0 **REP[15:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to:

– the number of PWM periods in edge-aligned mode

– the number of half PWM period in center-aligned mode.

46.6.16 TIMx capture/compare register 1 (TIMx_CCR1)(x = 1, 8)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input: CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

46.6.17 TIMx capture/compare register 2 (TIMx_CCR2)(x = 1, 8)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output: CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input: CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

46.6.18 TIMx capture/compare register 3 (TIMx_CCR3)(x = 1, 8)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 19:0 **CCR3[19:0]**: Capture/compare value

If channel CC3 is configured as output: CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc3 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR3[15:0]. The CCR3[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR3[19:4]. The CCR3[3:0] bitfield contains the dithered part.
If channel CC3 is configured as input: CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIMx_CCR3 register is read-only and cannot be programmed.
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR3[15:0]. The CCR3[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR3[19:4]. The CCR3[3:0] bits are reset.

46.6.19 TIMx capture/compare register 4 (TIMx_CCR4)(x = 1, 8)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 19:0 **CCR4[19:0]**: Capture/compare value

If channel CC4 is configured as output: CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[19:4]. The CCR4[3:0] bitfield contains the dithered part.

If channel CC4 is configured as input: CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIMx_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[19:4]. The CCR4[3:0] bits are reset.

46.6.20 TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	BK2BID	BKBID	BK2 DSRM	BK DSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the bits BKBID/BK2BID/BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.

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

Bit 29 **BK2BID**: Break2 bidirectional

Refer to BKBID description

Bit 28 **BKBID**: Break bidirectional

0: Break input tim_brk in input mode

1: Break input tim_brk in bidirectional mode

In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 **BK2DSRM**: Break2 disarm

Refer to BKDSRM description

Bit 26 **BKDSRM**: Break disarm

0: Break input tim_brk is armed

1: Break input tim_brk is disarmed

This bit is cleared by hardware when no break source is active.

The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 25 **BK2P**: Break 2 polarity

0: Break input tim_brk2 is active low

1: Break input tim_brk2 is active high

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 24 **BK2E**: Break 2 enable

This bit enables the complete break 2 protection (including all sources connected to bk_acth and BKIN sources, as per *Figure 421: Break and Break2 circuitry overview*).

0: Break2 function disabled

1: Break2 function enabled

Note: The BRKIN2 must only be used with OSSR = OSSI = 1.

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 23:20 **BK2F[3:0]**: Break 2 filter

This bit-field defines the frequency used to sample tim_brk2 input and the length of the digital filter applied to tim_brk2. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, tim_brk2 acts asynchronously
0001: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 19:16 **BKF[3:0]**: Break filter

This bit-field defines the frequency used to sample tim_brk input and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as one of the break inputs is active (tim_brk or tim_brk2). It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: In response to a break 2 event. OC and OCN outputs are disabled
In response to a break event or if MOE is written to 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register).
See OC/OCN enable description for more details (*Section 46.6.11: 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 tim_brk and tim_brk2 is active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low

1: Break input tim_brk is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

This bit enables the complete break protection (including all sources connected to bk_acth and BKIN sources, as per *Figure 421: 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 46.6.11: 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 46.6.11: 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 BKBID/BK2BID/BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.
DTG[7:5]=0xx => DT=DTG[7:0]x t_{dtg} with $t_{dtg}=t_{DTS}$.
DTG[7:5]=10x => DT=(64+DTG[5:0])xt_{dtg} with $T_{dtg}=2xt_{DTS}$.
DTG[7:5]=110 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=8xt_{DTS}$.
DTG[7:5]=111 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=16xt_{DTS}$.
Example if T_{DTS}=125ns (8MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 us to 31750 ns by 250 ns steps,
32 us to 63us by 1 us steps,
64 us to 126 us by 2 us steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

46.6.21 TIMx capture/compare register 5 (TIMx_CCR5)(x = 1, 8)

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[19:16]			
rw	rw	rw										rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR5[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **GC5C3**: Group channel 5 and channel 3
Distortion on channel 3 output:
0: No effect of tim_oc5ref on tim_oc3refc
1: tim_oc3refc is the logical AND of tim_oc3ref and tim_oc5ref
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR2).
Note: it is also possible to apply this distortion on combined PWM signals.

Bit 30 **GC5C2**: Group channel 5 and channel 2
Distortion on channel 2 output:
0: No effect of tim_oc5ref on tim_oc2refc
1: tim_oc2refc is the logical AND of tim_oc2ref and tim_oc5ref
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).
Note: it is also possible to apply this distortion on combined PWM signals.

Bit 29 **GC5C1**: Group channel 5 and channel 1
Distortion on channel 1 output:
0: No effect of oc5ref on oc1refc
1: oc1refc is the logical AND of oc1ref and oc5ref
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).
Note: it is also possible to apply this distortion on combined PWM signals.

Bits 28:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR5[19:0]**: Capture/compare 5 value
CCR5 is the value to be loaded in the actual capture/compare 5 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC5PE). Else the preload value is copied in the active capture/compare 5 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc5 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR5[15:0]. The CCR5[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR5[19:4]. The CCR5[3:0] bitfield contains the dithered part.

46.6.22 TIMx capture/compare register 6 (TIMx_CCR6)(x = 1, 8)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR6[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 19:0 **CCR6[19:0]**: Capture/compare 6 value

CCR6 is the value to be loaded in the actual capture/compare 6 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC6PE). Else the preload value is copied in the active capture/compare 6 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc6 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR6[15:0]. The CCR6[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR6[19:4]. The CCR6[3:0] bitfield contains the dithered part.

46.6.23 TIMx capture/compare mode register 3 (TIMx_CCMR3) (x = 1, 8)

Address offset: 0x050

Reset value: 0x0000 0000

Refer to the above CCMR1 register description. Channels 5 and 6 can only be configured in output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC6 CE	OC6M[2:0]			OC6 PE	OC6FE	Res.	Res.	OC5 CE	OC5M[2:0]			OC5PE	OC5FE	Res.	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw		

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC6CE**: Output compare 6 clear enable

Bits 24, 14:12 **OC6M[3:0]**: Output compare 6 mode

Bit 11 **OC6PE**: Output compare 6 preload enable

Bit 10 **OC6FE**: Output compare 6 fast enable

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **OC5CE:** Output compare 5 clear enable

Bits 16, 6:4 **OC5M[3:0]**: Output compare 5 mode

Bit 3 **OC5PE**: Output compare 5 preload enable

Bit 2 **OC5FE**: Output compare 5 fast enable

Bits 1:0 Reserved, must be kept at reset value.

46.6.24 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 1, 8)

Address offset: 0x054

Reset value: 0x0000 0000

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

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

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

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

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5]=0xx => DTF=DTGF[7:0]x t_{dtg} with $t_{dtg}=t_{DTS}$.

DTGF[7:5]=10x => DTF=(64+DTGF[5:0])xt_{dtg} with $T_{dtg}=2xt_{DTS}$.

DTGF[7:5]=110 => DTF=(32+DTGF[4:0])xt_{dtg} with $T_{dtg}=8xt_{DTS}$.

DTGF[7:5]=111 => DTF=(32+DTGF[4:0])xt_{dtg} with $T_{dtg}=16xt_{DTS}$.

Example if T_{DTS}=125ns (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 us to 31750 ns by 250 ns steps,

32 us to 63us by 1 us steps,

64 us to 126 us by 2 us steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

46.6.25 TIMx timer encoder control register (TIMx_ECR)(x = 1, 8)

Address offset: 0x058

Reset value: 0x0000 0000

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

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **PWPRSC[2:0]**: Pulse width prescaler

This bitfield sets the clock prescaler for the pulse generator, as following:

$t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim\_ker\_ck}$

Bits 23:16 **PW[7:0]**: Pulse width

This bitfield defines the pulse duration, as following:

$t_{PW} = PW[7:0] \times t_{PWG}$

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

Bits 7:6 **IPOS[1:0]**: Index positioning

In quadrature encoder mode (SMS[3:0] = 0001, 0010, 0011, 1110, 1111), this bit indicates in which AB input configuration the Index event resets the counter.

00: Index resets the counter when AB = 00
01: Index resets the counter when AB = 01
10: Index resets the counter when AB = 10
11: Index resets the counter when AB = 11

In directional clock mode or clock plus direction mode (SMS[3:0] = 1010, 1011, 1100, 1101), these bits indicates on which level the Index event resets the counter. In bidirectional clock mode, this applies for both clock inputs.

x0: Index resets the counter when clock is 0
x1: Index resets the counter when clock is 1

Note: IPOS[1] bit is not significant

Bit 5 **FIDX**: First index

This bit indicates if the first index only is taken into account

0: Index is always active
1: the first Index only resets the counter

Bits 4:3 **IBLK[1:0]**: Index blanking
This bit indicates if the Index event is conditioned by the tim_ti3 or tim_ti4 input
00: Index always active
01: Index disabled when tim_ti3 input is active, as per CC3P bitfield
10: Index disabled when tim_ti4 input is active, as per CC4P bitfield
11: Reserved

Bits 2:1 **IDIR[1:0]**: Index direction
This bit indicates in which direction the Index event resets the counter.
00: Index resets the counter whatever the direction
01: Index resets the counter when up-counting only
10: Index resets the counter when down-counting only
11: Reserved
Note: The IDR[1:0] bitfield must be written when IE bit is reset (index disabled).

Bit 0 **IE**: Index enable
This bit indicates if the Index event resets the counter.
0: Index disabled
1: Index enabled

46.6.26 TIMx timer input selection register (TIMx_TISEL)(x = 1, 8)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

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

Bits 27:24 **TI4SEL[3:0]**: Selects tim_ti4[15:0] input
0000: tim_ti4_in0: TIMx_CH4
0001: tim_ti4_in1
...
1111: tim_ti4_in15
Refer to *Section 46.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: Selects tim_ti3[15:0] input
0000: tim_ti3_in0: TIMx_CH2
0001: tim_ti3_in1
...
1111: tim_ti3_in15
Refer to *Section 46.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: Selects tim_ti2[15:0] input
0000: tim_ti2_in0: TIMx_CH2
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 46.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

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

Bits 3:0 **TI1SEL[3:0]**: Selects tim_ti1[15:0] input
0000: tim_ti1_in0: TIMx_CH1
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 46.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

46.6.27 TIMx alternate function option register 1 (TIMx_AF1)(x = 1, 8)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	BK CMP8E	BK CMP7E	BK CMP6E	BK CMP5E	BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: etr_in source selection
These bits select the etr_in input source.
0000: tim_etr0: TIMx_ETR input
0001: tim_etr1
...
1111: tim_etr15
Refer to *Section 46.3.2: TIM1/TIM8 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity
This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp4 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)
1: tim_brk_cmp4 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp3 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp2 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp1 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: TIMx_BKIN input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Section 46.3.2: TIM1/TIM8 pins and internal signals for product specific implementation.

46.6.28 TIMx alternate function register 2 (TIMx_AF2)(x = 1, 8)

Address offset: 0x064

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BK2CMP4P	BK2CMP3P	BK2CMP2P	BK2CMP1P	BK2INP	BK2CMP8E	BK2CMP7E	BK2CMP6E	BK2CMP5E	BK2CMP4E	BK2CMP3E	BK2CMP2E	BK2CMP1E	BK2INE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 18:16 **OCRSEL[2:0]**: ocref_clr source selection

These bits select the ocref_clr input source.

000: tim_ocref_clr0

001: tim_ocref_clr1

...

111: tim_ocref_clr7

Refer to *Section 46.3.2: TIM1/TIM8 pins and internal signals* for product specific information.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **BK2CMP4P**: tim_brk2_cmp4 input polarity

This bit selects the tim_brk2_cmp4 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp4 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp4 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 12 **BK2CMP3P**: tim_brk2_cmp3 input polarity

This bit selects the tim_brk2_cmp3 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp3 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp3 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 **BK2CMP2P**: tim_brk2_cmp2 input polarity

This bit selects the tim_brk2_cmp2 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp2 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BK2CMP1P**: tim_brk2_cmp1 input polarity

This bit selects the tim_brk2_cmp1 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp1 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp1 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BK2INP**: TIMx_BKIN2 input polarity

This bit selects the TIMx_BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.

0: TIMx_BKIN2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: TIMx_BKIN2 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BK2CMP8E**: tim_brk2_cmp8 enable

This bit enables the tim_brk2_cmp8 for the timer's tim_brk2 input. tim_brk2_cmp8 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp8 input disabled

1: tim_brk2_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **BK2CMP7E**: tim_brk2_cmp7 enable

This bit enables the tim_brk2_cmp7 for the timer's tim_brk2 input. tim_brk2_cmp7 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp7 input disabled

1: tim_brk2_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 6 **BK2CMP6E**: tim_brk2_cmp6 enable

This bit enables the tim_brk2_cmp6 for the timer's tim_brk2 input. tim_brk2_cmp6 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp6 input disabled

1: tim_brk2_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 5 **BK2CMP5E**: tim_brk2_cmp5 enable

This bit enables the tim_brk2_cmp5 for the timer's tim_brk2 input. tim_brk2_cmp5 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp5 input disabled

1: tim_brk2_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 4 **BK2CMP4E**: tim_brk2_cmp4 enable

This bit enables the tim_brk2_cmp4 for the timer's tim_brk2 input. tim_brk2_cmp4 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp4 input disabled

1: tim_brk2_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 3 **BK2CMP3E**: tim_brk2_cmp3 enable

This bit enables the tim_brk2_cmp3 for the timer's tim_brk2 input. tim_brk2_cmp3 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp3 input disabled

1: tim_brk2_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 2 **BK2CMP2E**: tim_brk2_cmp2 enable

This bit enables the tim_brk2_cmp2 for the timer's tim_brk2 input. tim_brk2_cmp2 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp2 input disabled

1: tim_brk2_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BK2CMP1E**: tim_brk2_cmp1 enable

This bit enables the tim_brk2_cmp1 for the timer's tim_brk2 input. tim_brk2_cmp1 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp1 input disabled

1: tim_brk2_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BK2INE**: TIMx_BKIN2 input enable

This bit enables the TIMx_BKIN2 alternate function input for the timer's tim_brk2 input. TIMx_BKIN2 input is 'ORed' with the other tim_brk2 sources.

0: TIMx_BKIN2 input disabled

1: TIMx_BKIN2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Section 46.3.2: TIM1/TIM8 pins and internal signals for product specific implementation.

46.6.29 TIMx DMA control register (TIMx_DCR)(x = 1, 8)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[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.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

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

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

0000: Reserved
0001: Update
0010: CC1
0011: CC2
0100: CC3
0101: CC4
0110: COM
0111: Trigger
Others: reserved

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
...
11010: 26 transfers

Example: Let us consider the following transfer: DBL = 7 bytes & DBA = TIM2_CR1.

–If DBL = 7 bytes and DBA = TIM2_CR1 represents the address of the byte to be transferred, the address of the transfer should be given by the following equation:

(TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data are copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

–If the DMA Data Size is configured in half-words, 16-bit data are transferred to each of the 7 registers.

–If the DMA Data Size is configured in bytes, the data are also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1
00001: TIMx_CR2
00010: TIMx_SMCR
...

46.6.30 TIMx DMA address for full transfer (TIMx_DMAR)(x = 1, 8)

Address offset: 0x3E0

Reset value: 0x0000 0000

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

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address (TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

46.6.31 TIMx register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

Table 446. TIMx register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																				0	0		0	0	0	0	0	0	0	0	0	0
0x004	**TIMx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	MMS[3]	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5	OIS4N	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]			CCDS	CCUS	Res.	CCPC
	Reset value							0		0	0	0	0		0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x008	**TIMx_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	ETP	ECE	ETP S [1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
	Reset value							0	0			0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value									0	0	0	0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	CC6IF	CC5IF	Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value									0	0	0	0			0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																								0	0	0	0	0	0	0	0	0

Table 446. TIMx register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	**TIMx_CCMR1** Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2 S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2 S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	**TIMx_CCMR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4 S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3 S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4 S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	**TIMx_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E	CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value											0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	**TIMx_CNT**	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	**TIMx_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	**TIMx_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x030	**TIMx_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	**TIMx_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	**TIMx_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	**TIMx_CCR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	**TIMx_CCR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	**TIMx_BDTR**	Res.	Res.	BK2BID	BKBID	BK2DSRM	BKDSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 446. TIMx register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x048	**TIMx_CCR5**	GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[19:0]																			
	Reset value	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	**TIMx_CCR6**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	**TIMx_CCMR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]	OC6CE	OC6M [2:0]			OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M [2:0]			OC5PE	OC5FE	Res.	Res.
	Reset value								0								0	0	0	0	0	0	0			0	0	0	0	0	0		
0x054	**TIMx_DTR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x058	**TIMx_ECR**	Res.	Res.	Res.	Res.	Res.	PWPR SC[2:0]			PW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS [1:0]		FIDX	IBLK [1:0]		IDIR [1:0]		IE
	Reset value						0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x05C	**TIMx_TISEL**	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x060	**TIMx_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res ETRSEL [3:0]				BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x064	**TIMx_AF2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL [2:0]			Res.	Res.	BK2CMP4P	BK2CMP3P	BK2CMP2P	BK2CMP1P	BK2INP	BK2CMP8E	BK2CMP7E	BK2CMP6E	BK2CMP5E	BK2CMP4E	BK2CMP3E	BK2CMP2E	BK2CMP1E	BK2INE
	Reset value														0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x068.. 0x3D8	Reserved	Res.																															
0x3DC	**TIMx_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	**TIMx_DMAR**	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3 on page 115* for the register boundary addresses.

47 General-purpose timers (TIM2/TIM3/TIM4/TIM5)

47.1 TIM2/TIM3/TIM4/TIM5 introduction

The general-purpose timers consist of a 16-bit or 32-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 47.4.23: Timer synchronization*.

47.2 TIM2/TIM3/TIM4/TIM5 main features

General-purpose TIMx timer features include:

- 16-bit or 32-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge- and Center-aligned modes)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

47.3 TIM2/TIM3/TIM4/TIM5 implementation

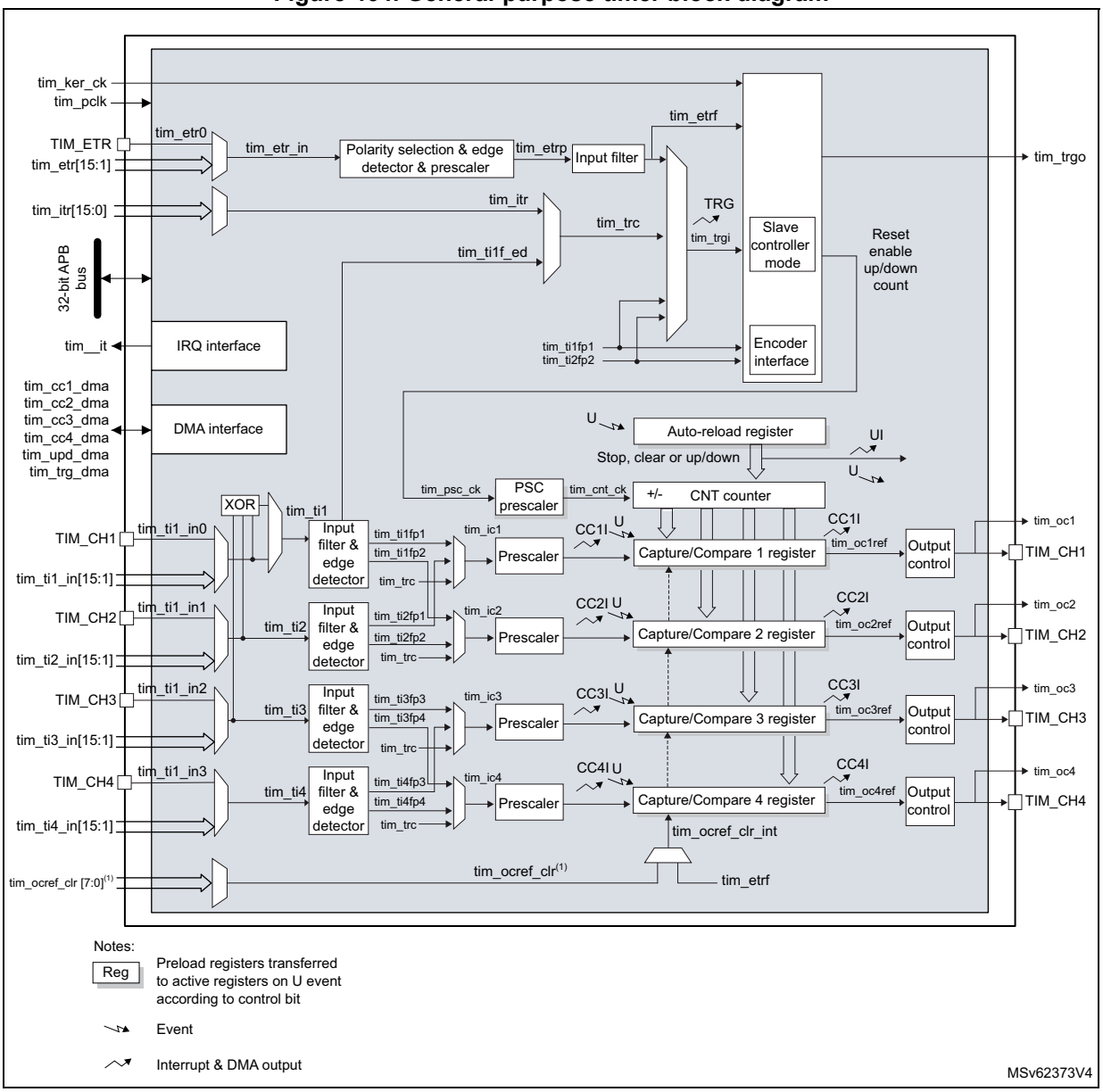
Table 447. STM32U575/585 general purpose timers

Timer instance	TIM2	TIM3	TIM4	TIM5
Resolution	32-bit	32-bit	32-bit	32-bit
OCREF clear selection	Yes	Yes	Yes	Yes
Sources	tim_etrf tim_ocref_clr[7:0]	tim_etrf tim_ocref_clr[7:0]	tim_etrf tim_ocref_clr[7:0]	tim_etrf tim_ocref_clr[7:0]

47.4 TIM2/TIM3/TIM4/TIM5 functional description

47.4.1 Block diagram

Figure 464. General-purpose timer block diagram



1. This feature is not available on all timers, refer to the *Section 47.3: TIM2/TIM3/TIM4/TIM5 implementation*.

47.4.2 TIM2/TIM3/TIM4/TIM5 pins and internal signals

Table 448 and *Table 449* in this section summarize the TIM inputs and outputs.

Table 448. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2 TIM_CH3 TIM_CH4	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) , external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.
TIM_ETR	Input	External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.

Table 449. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0] tim_ti3_in[15:0] tim_ti4_in[15:0]	Input	Internal timer inputs bus. The tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_etr[15:0]	Input	External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulse width control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
tim_itr[15:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_ocref_clr[7:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulse width control.
tim_pclk	Input	Timer APB clock.
tim_ker_ck	Input	Timer kernel clock

Table 449. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_it	Output	Global Timer interrupt, gathering capture/compare, update and break trigger requests.
tim_cc1_dma tim_cc2_dma tim_cc3_dma tim_cc4_dma	Output	Timer capture / compare 1..4 dma requests.
tim_upd_dma	Output	Timer update dma request.
tim_trg_dma	Output	Timer trigger dma request.

Table 450, *Table 451*, *Table 452* and *Table 453* are listing the sources connected to the tim_ti[4:1] input multiplexers.

Table 450. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti1_in0	TIM2_CH1	TIM3_CH1	TIM4_CH1	TIM5_CH1
tim_ti1_in1	comp1_out	comp1_out	comp1_out	LSI
tim_ti1_in2	comp2_out	comp2_out	comp2_out	LSE
tim_ti1_in3	Reserved	Reserved	Reserved	rtc_wut_trg
tim_ti1_in4				comp1_out
tim_ti1_in5				comp2_out
tim_ti1_in[15:6]	Reserved			

Table 451. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti2_in0	TIM2_CH2	TIM3_CH2	TIM4_CH2	TIM5_CH2
tim_ti2_in1	comp1_out	comp1_out	comp1_out	comp1_out
tim_ti2_in2	comp2_out	comp2_out	comp2_out	comp2_out
tim_ti2_in[15:3]	Reserved			

Table 452. Interconnect to the tim_ti3 input multiplexer

tim_ti3 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti3_in0	TIM2_CH3	TIM3_CH3	TIM4_CH3	TIM5_CH3
tim_ti3_in[15:1]	Reserved			

Table 453. Interconnect to the tim_ti4 input multiplexer

tim_ti4 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti4_in0	TIM2_CH4	TIM3_CH4	TIM4_CH4	TIM5_CH4
tim_ti4_in1	comp1_out	Reserved	Reserved	Reserved
tim_ti4_in2	comp2_out			
tim_ti4_in[15:3]	Reserved			

Table 454 lists the internal sources connected to the tim_itr input multiplexer.

Table 454. TIMx internal trigger connection

TIMx	TIM2	TIM3	TIM4	TIM5
tim_itr0	tim1_trgo	tim1_trgo	tim1_trgo	tim1_trgo
tim_itr1	Reserved	tim2_trgo	tim2_trgo	tim2_trgo
tim_itr2	tim3_trgo	Reserved	tim3_trgo	tim3_trgo
tim_itr3	tim4_trgo	tim4_trgo	Reserved	tim4_trgo
tim_itr4	tim5_trgo	tim5_trgo	tim5_trgo	Reserved
tim_itr5	tim8_trgo	tim8_trgo	tim8_trgo	tim8_trgo
tim_itr6	tim15_trgo	tim15_trgo	tim15_trgo	tim15_trgo
tim_itr7	tim16_oc1	tim16_oc1	tim16_oc1	tim16_oc1
tim_itr8	tim17_oc1	tim17_oc1	tim17_oc1	tim17_oc1
tim_itr9	Reserved	Reserved	Reserved	Reserved
tim_itr10				
tim_itr11	OTG FS SOF			
tim_itr[15:12]	Reserved			

Table 455 lists the internal sources connected to the tim_etr input multiplexer.

Table 455. Interconnect to the tim_etr input multiplexer

Timer external trigger input signal	Timer external trigger signals assignment			
	TIM2	TIM3	TIM4	TIM5
tim_etr0	TIM2_ETR	TIM3_ETR	TIM4_ETR	TIM5_ETR
tim_etr1	comp1_out	comp1_out	comp1_out	comp1_out
tim_etr2	comp2_out	comp2_out	comp2_out	comp2_out
tim_etr3	MSIK	MSIK	MSIK	MSIK
tim_etr4	HSI	HSI	HSI	HSI
tim_etr5	MSIS	MSIS	MSIS	MSIS
tim_etr6	Reserved	Reserved	Reserved	Reserved
tim_etr7				
tim_etr8	TIM3_ETR	TIM2_ETR	TIM3_ETR	TIM2_ETR
tim_etr9	TIM4_ETR	TIM4_ETR	TIM5_ETR	TIM3_ETR
tim_etr10	TIM5_ETR	Reserved	Reserved	Reserved
tim_etr11	LSE	adc1_awd1		
tim_etr12	Reserved	adc1_awd2		
tim_etr13		adc1_awd3		
tim_etr[15:14]	Reserved			

Table 456 lists the internal sources connected to the tim_ocref_clr input multiplexer.

Table 456. Interconnect to the tim_ocref_clr input multiplexer

Timer tim_ocref_clr signal	Timer tim_ocref_clr signals assignment			
	TIM2	TIM3	TIM4	TIM5
tim_ocref_clr0	comp1_out	comp1_out	comp1_out	comp1_out
tim_ocref_clr1	comp2_out	comp2_out	comp2_out	comp2_out
tim_ocref_clr[7:2]	Reserved			

47.4.3 Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC):
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 465 and *Figure 466* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 465. Counter timing diagram with prescaler division change from 1 to 2

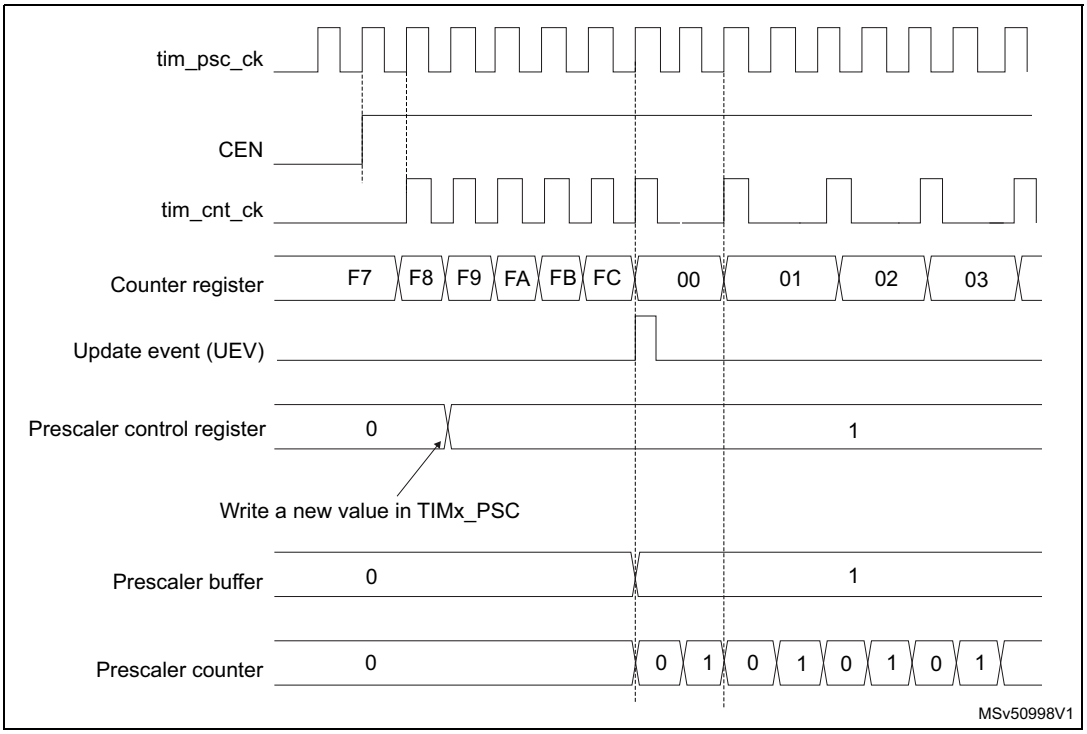
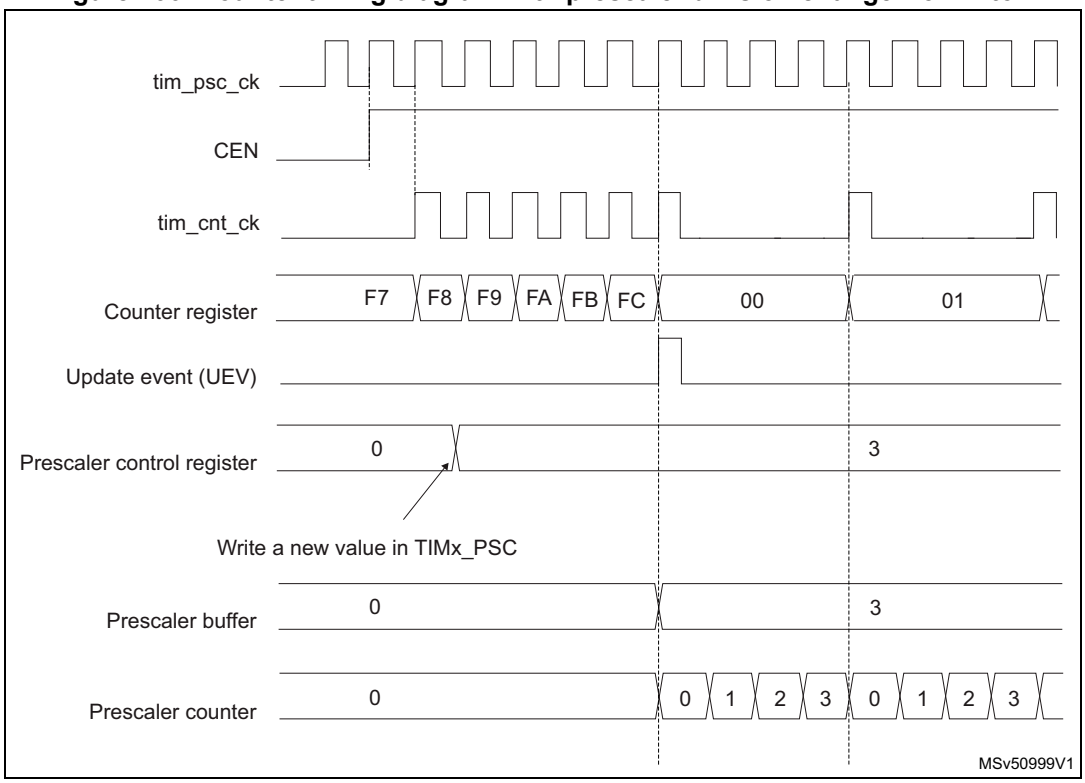


Figure 466. Counter timing diagram with prescaler division change from 1 to 4



47.4.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An Update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 467. Counter timing diagram, internal clock divided by 1

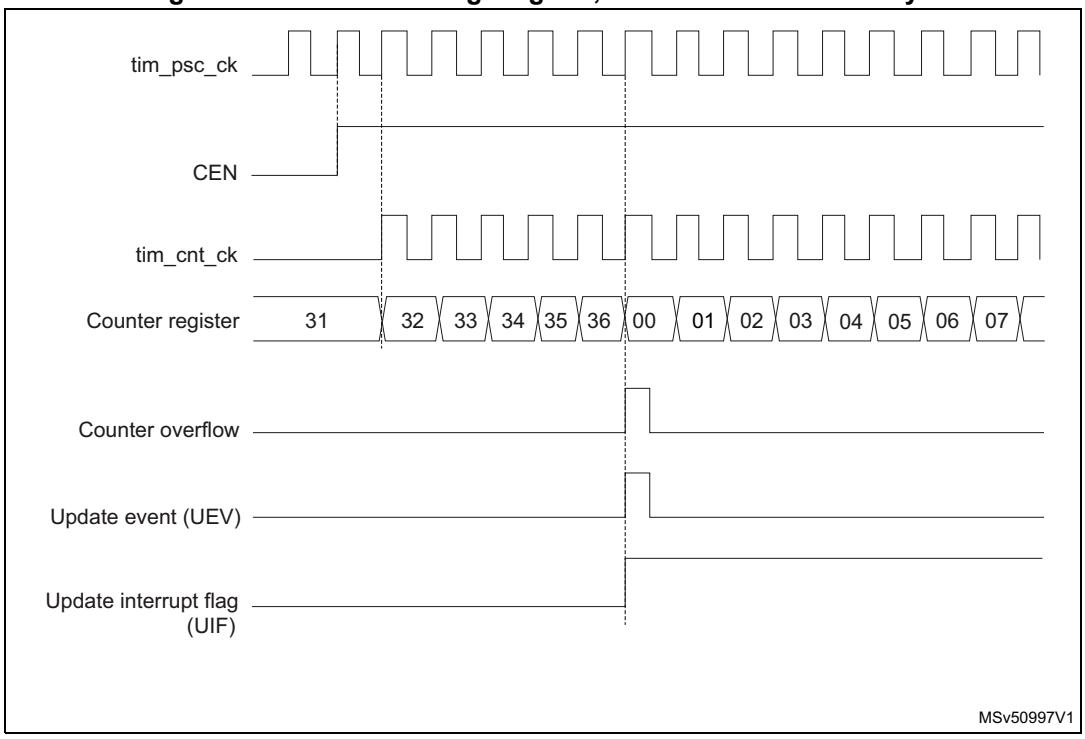


Figure 468. Counter timing diagram, internal clock divided by 2

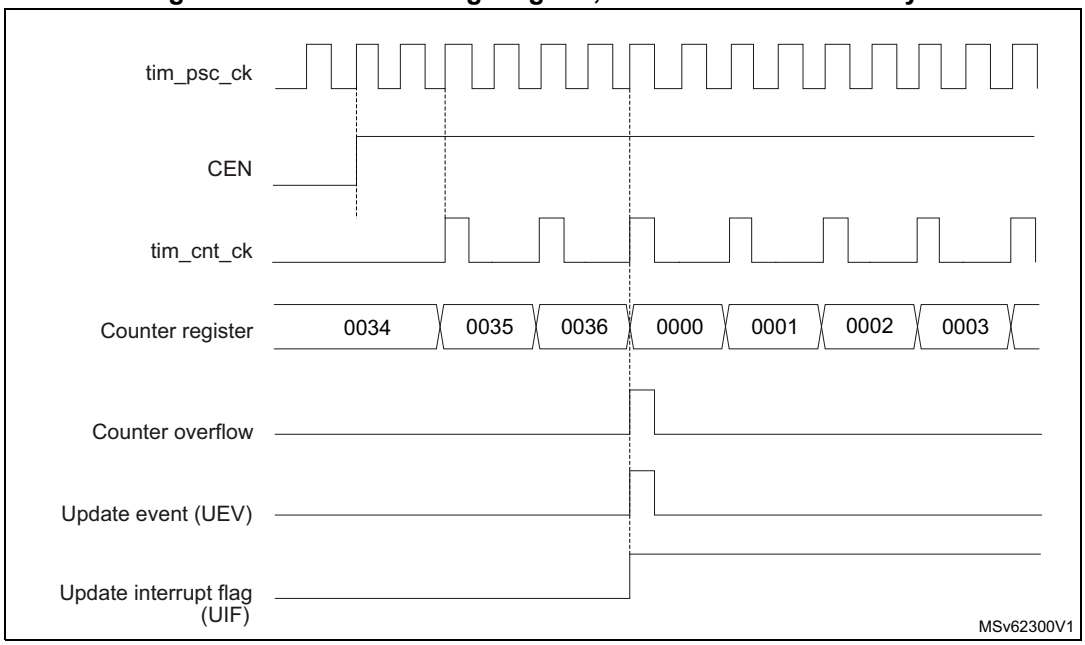


Figure 469. Counter timing diagram, internal clock divided by 4

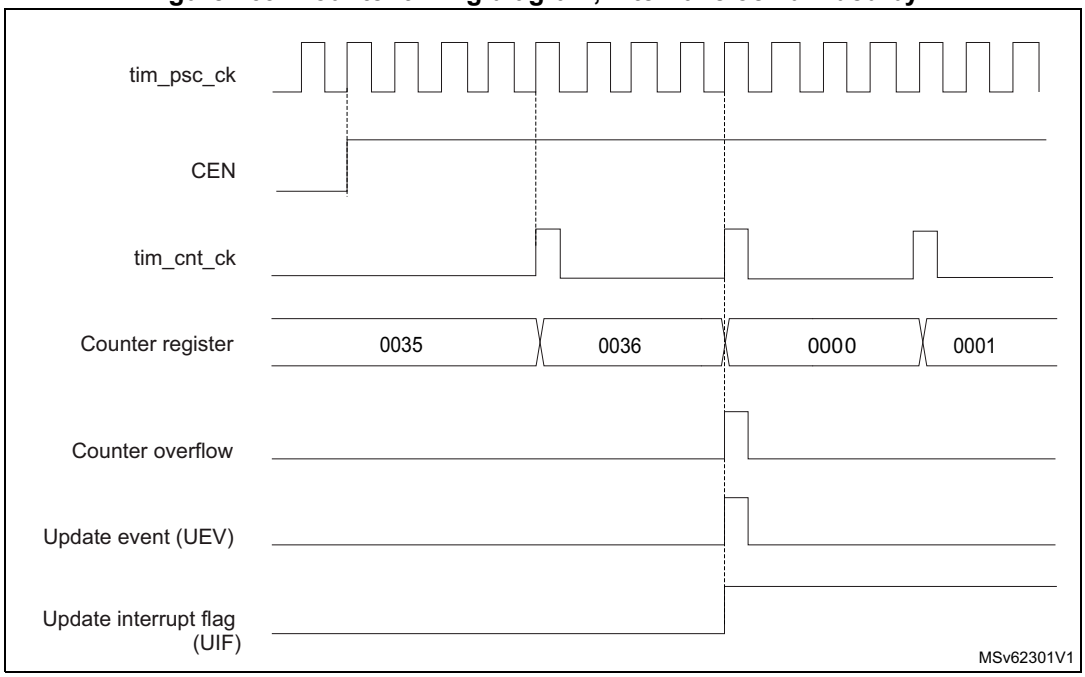


Figure 470. Counter timing diagram, internal clock divided by N

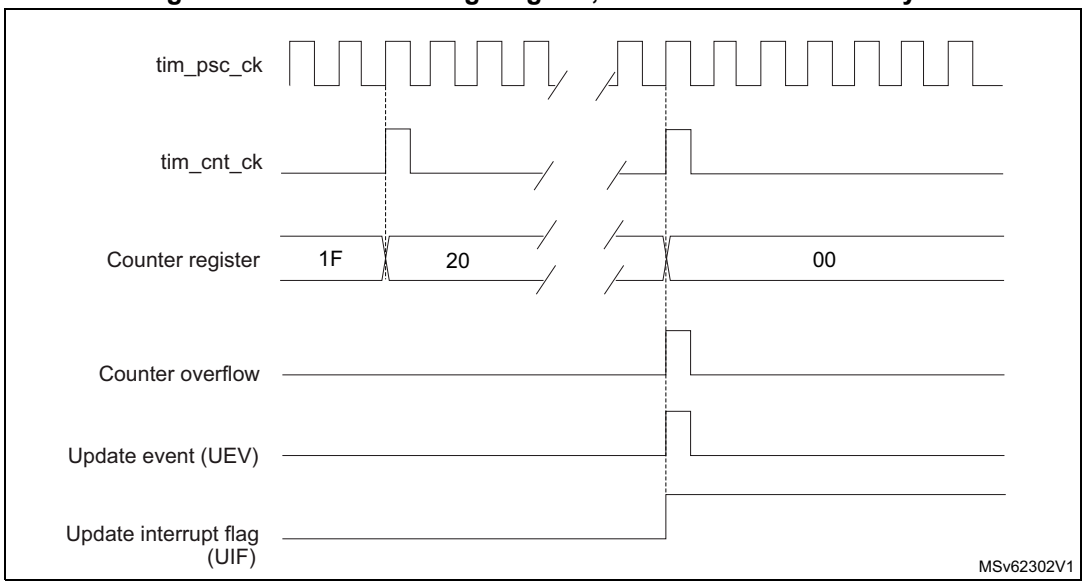


Figure 471. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)

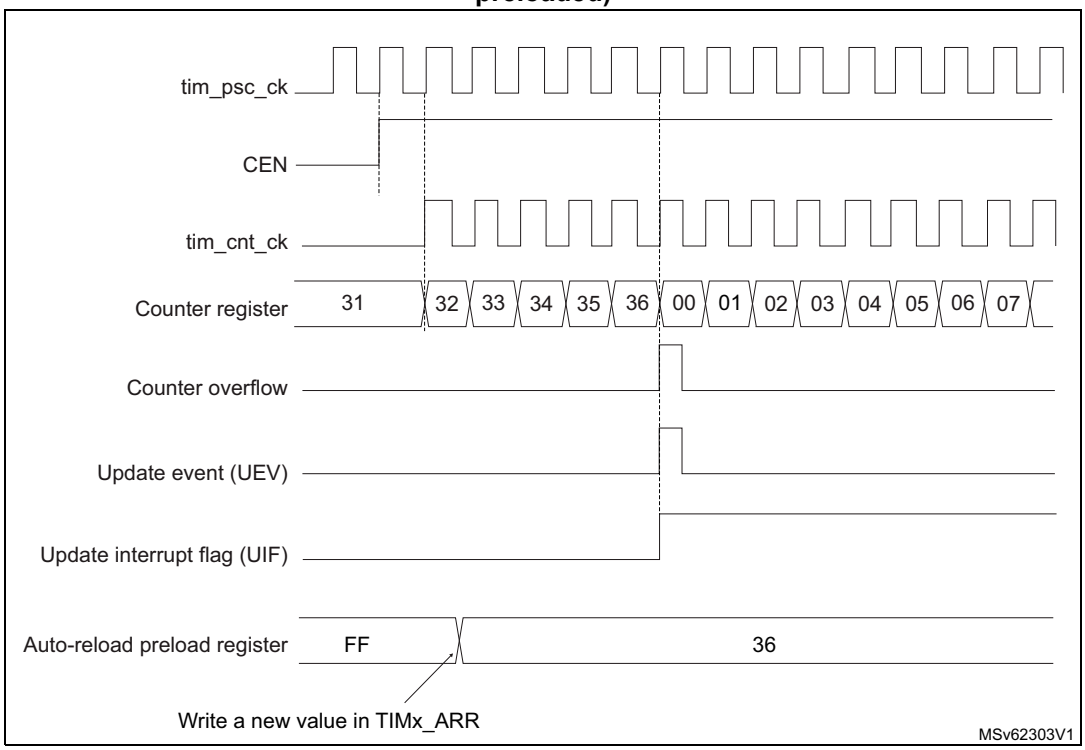
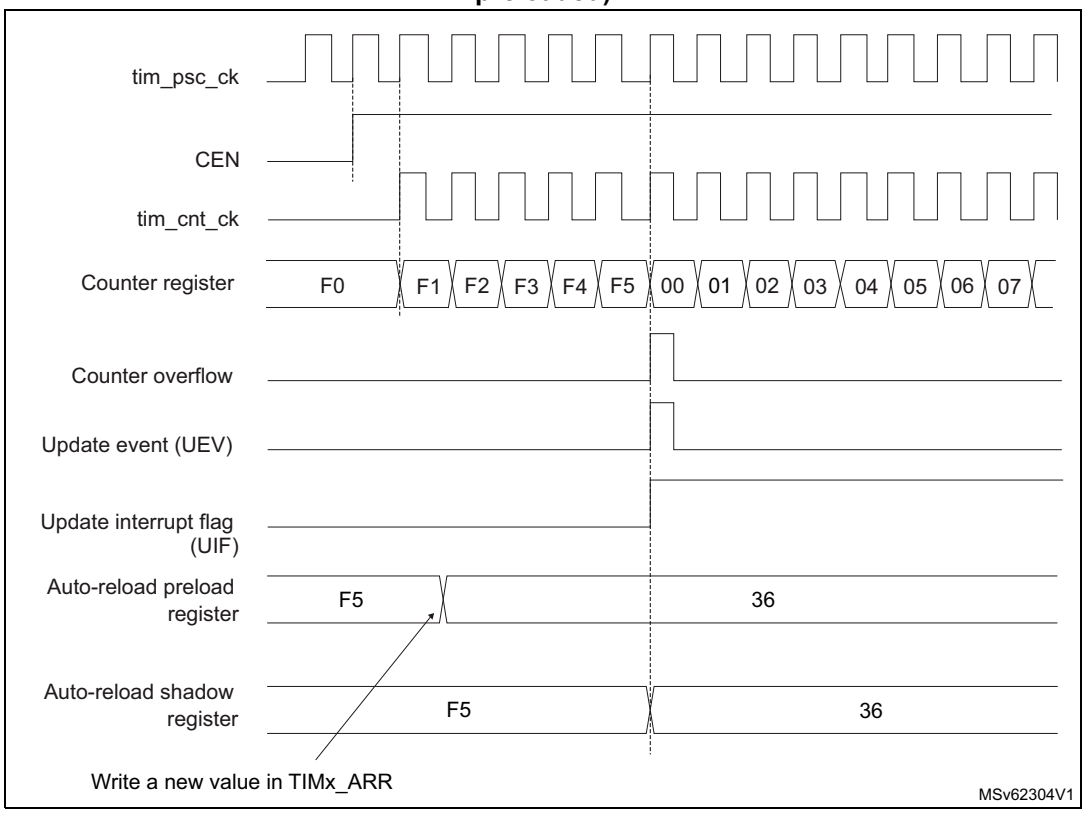


Figure 472. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded)



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 473. Counter timing diagram, internal clock divided by 1

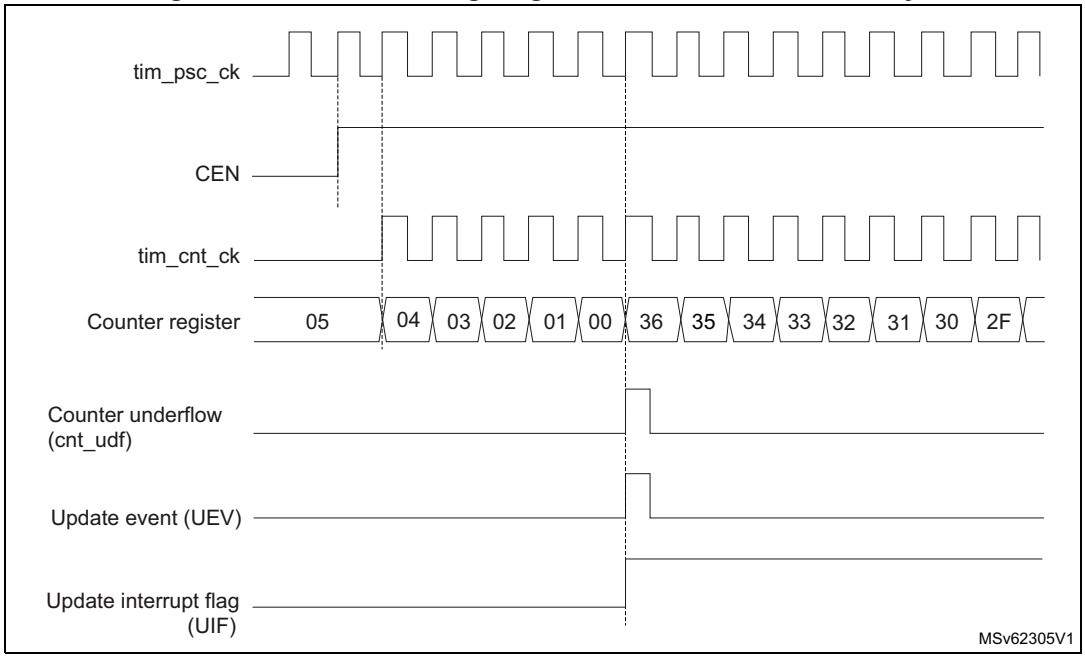


Figure 474. Counter timing diagram, internal clock divided by 2

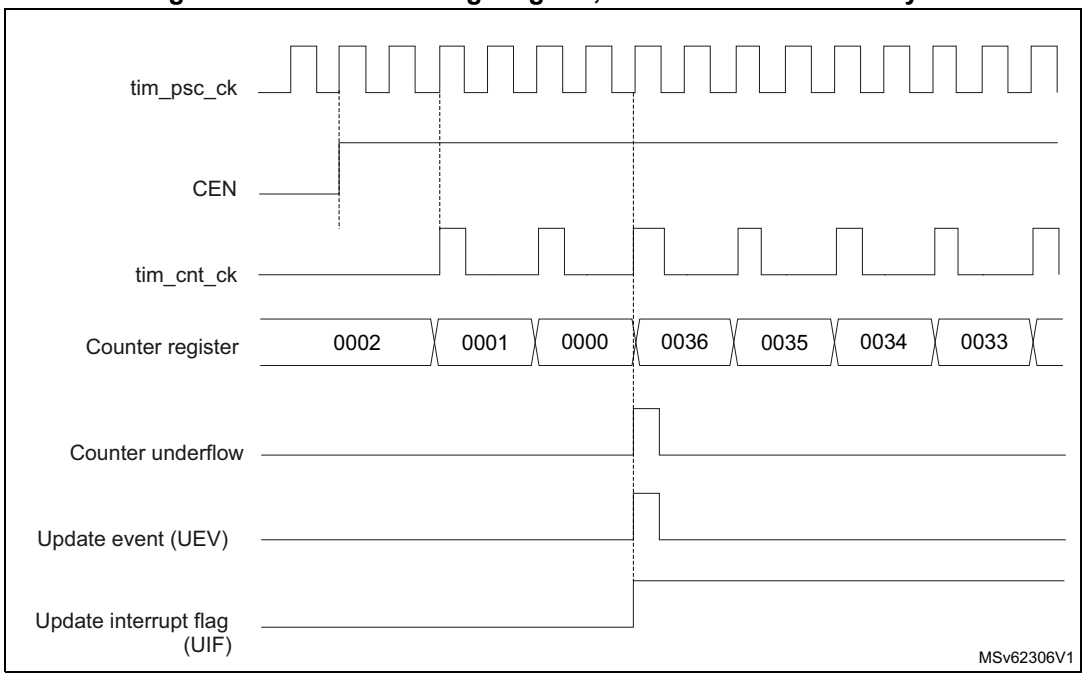


Figure 475. Counter timing diagram, internal clock divided by 4

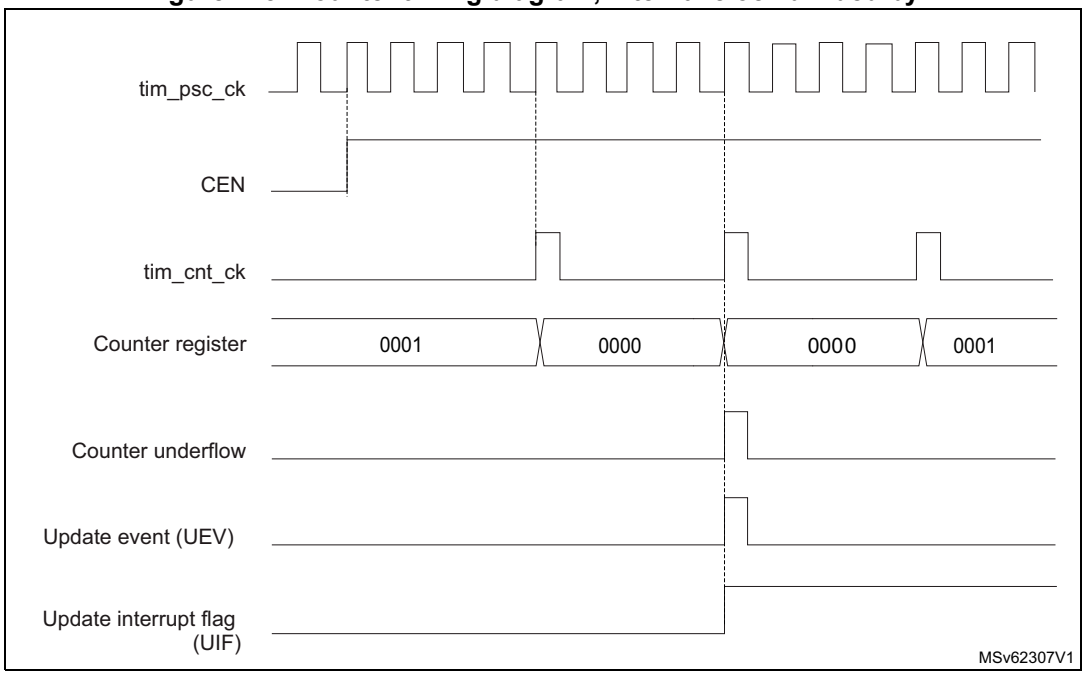


Figure 476. Counter timing diagram, internal clock divided by N

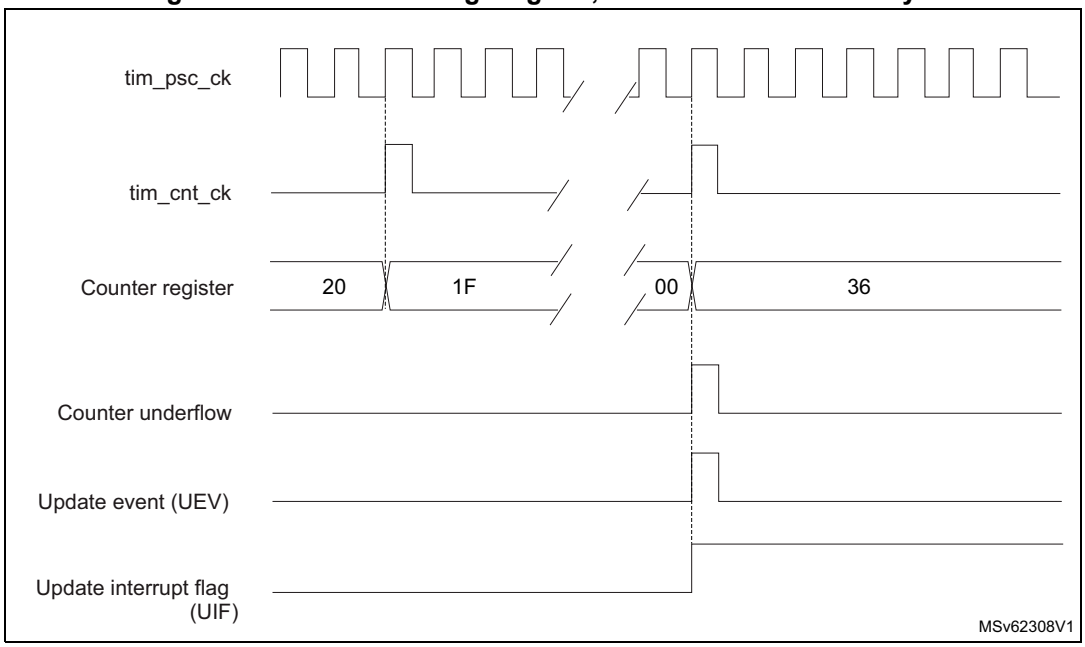
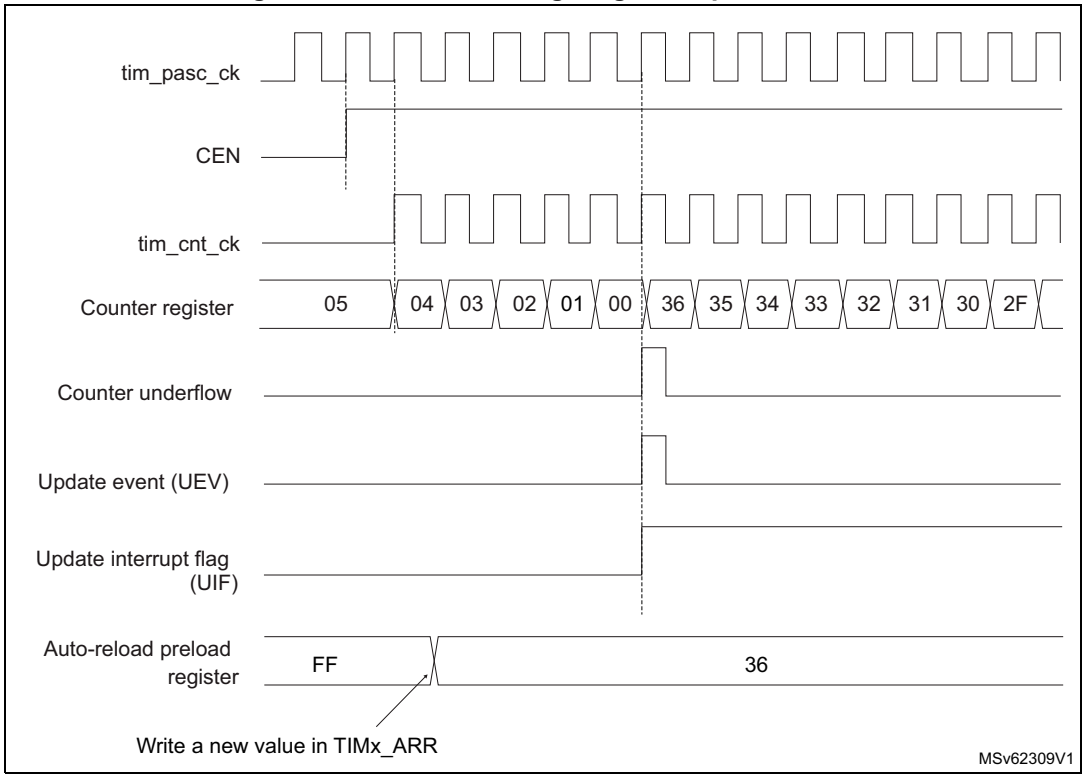


Figure 477. Counter timing diagram, Update event



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-

reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the direction bit (DIR from TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

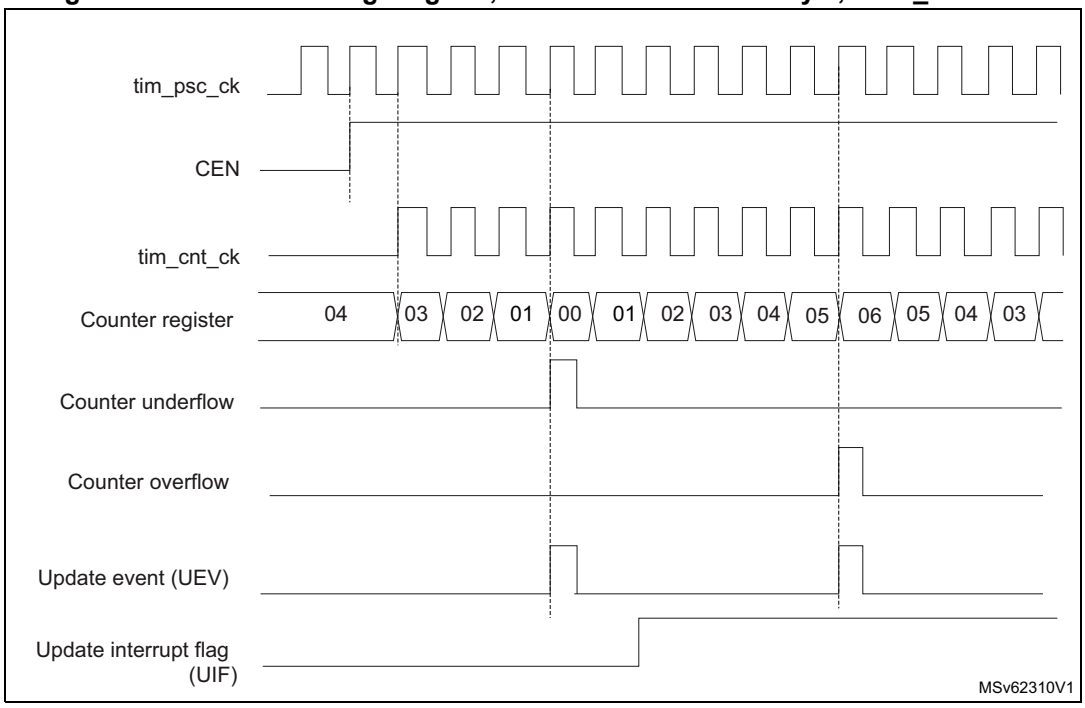
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 478. 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 47.5.1: TIMx control register 1 (TIMx_CR1)(x = 2 to 5) on page 1852*).

Figure 479. Counter timing diagram, internal clock divided by 2

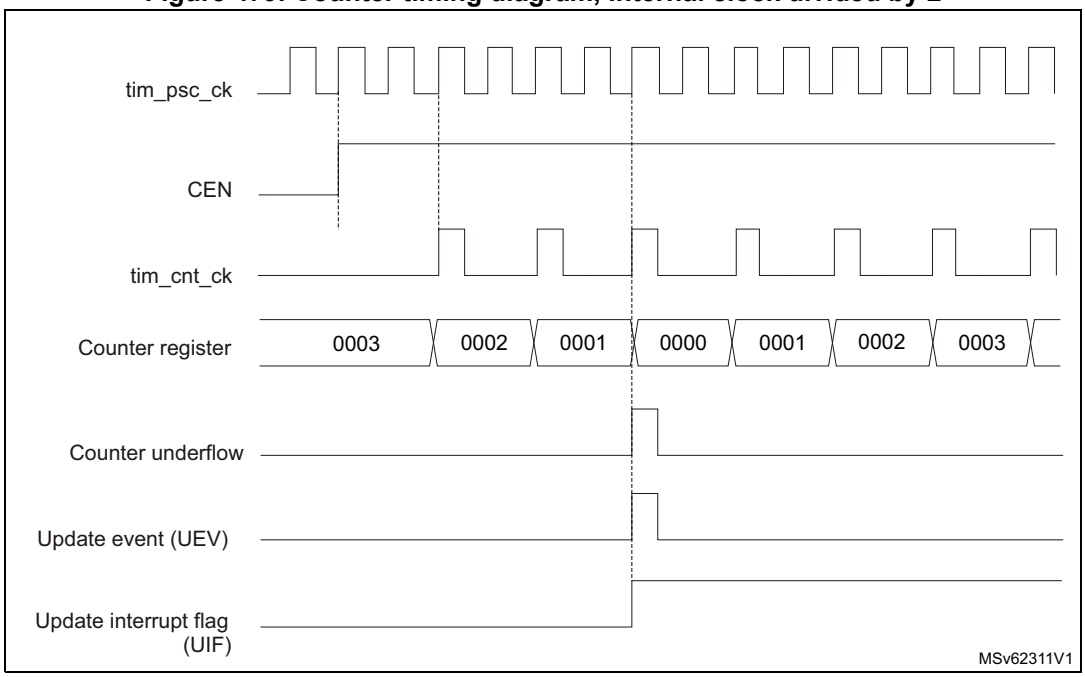
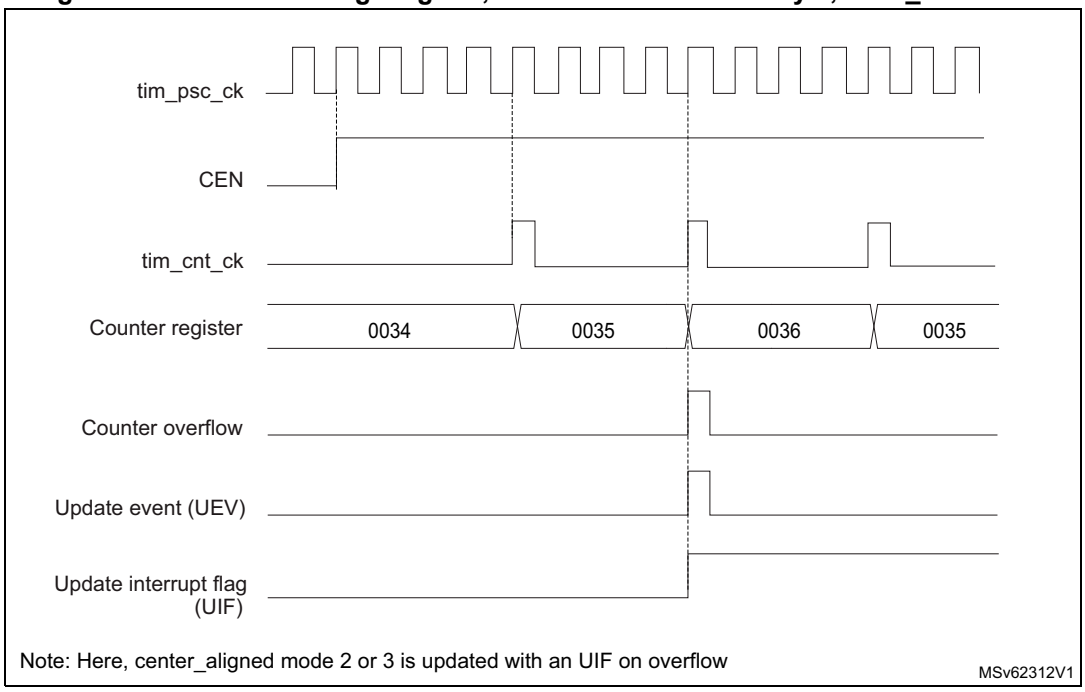


Figure 480. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36



1. Center-aligned mode 2 or 3 is used with an UIF on overflow.

Figure 481. Counter timing diagram, internal clock divided by N

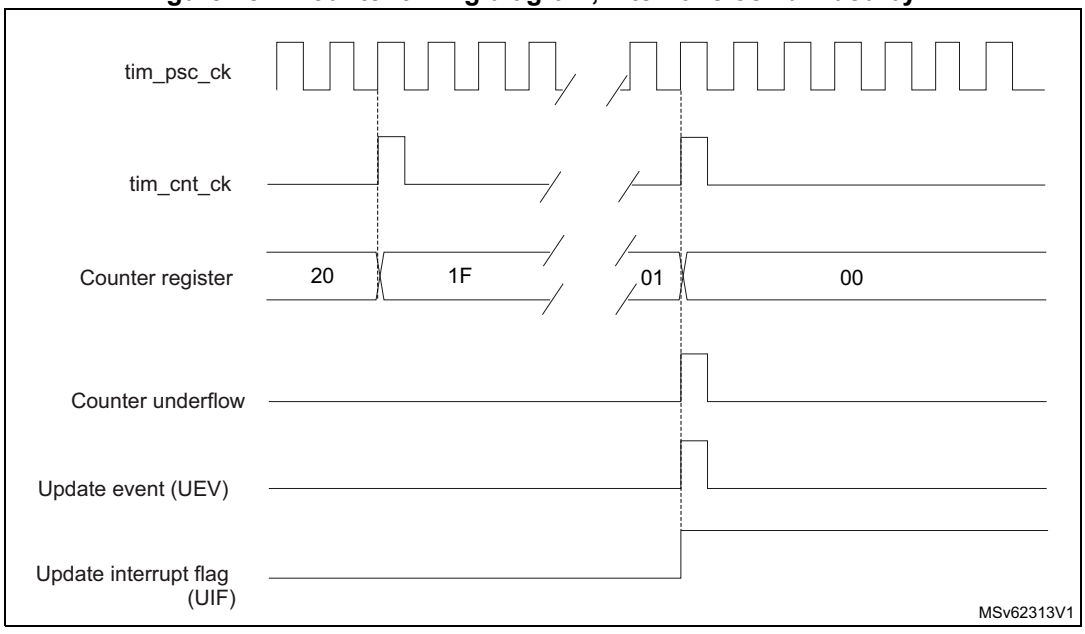


Figure 482. Counter timing diagram, Update event with ARPE=1 (counter underflow)

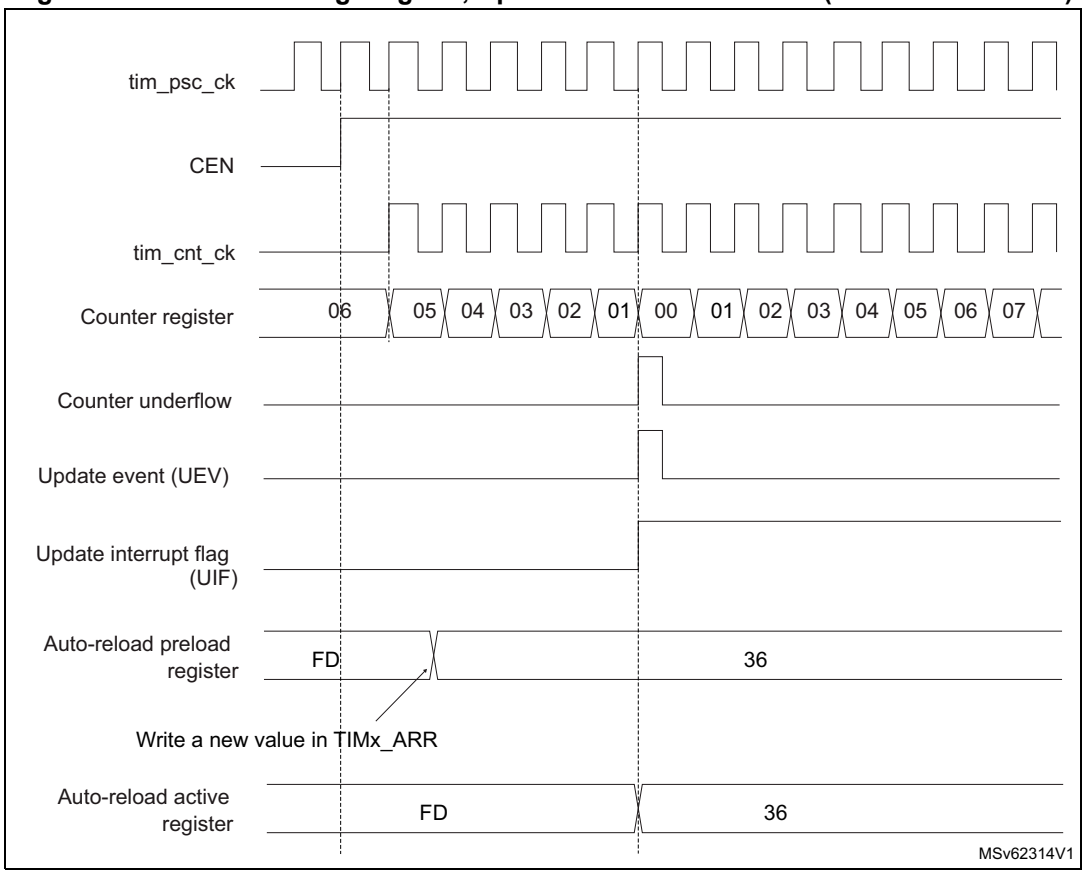
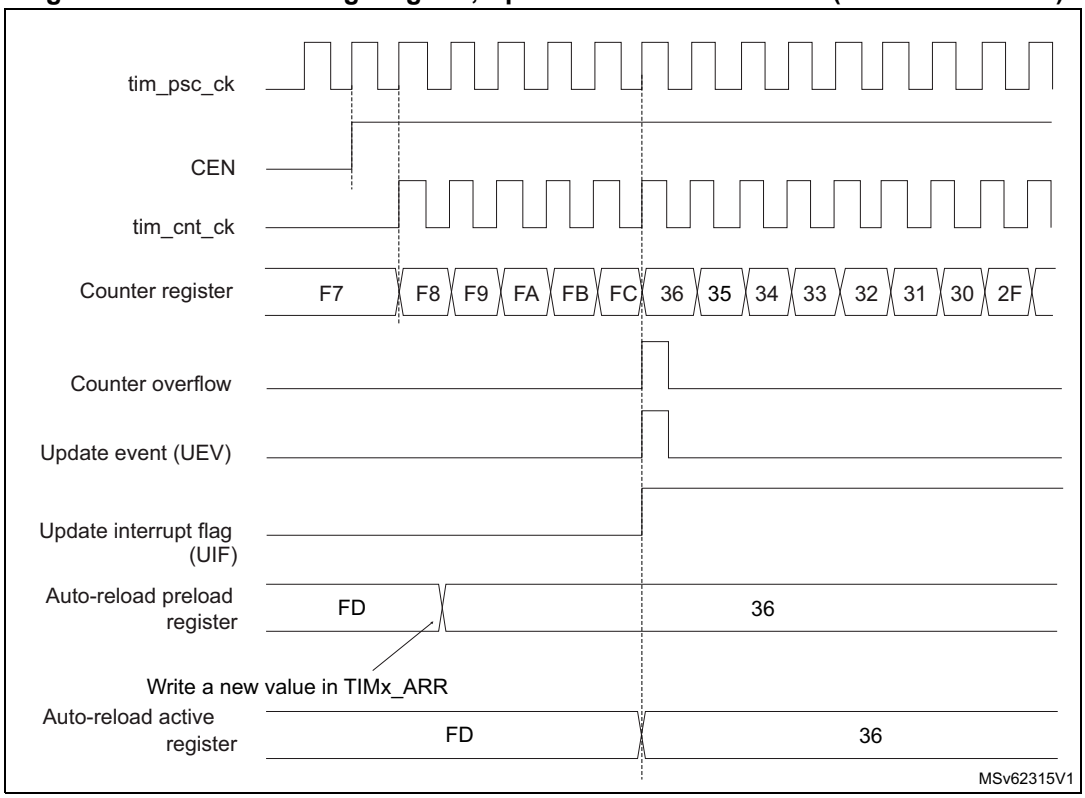


Figure 483. Counter timing diagram, Update event with ARPE=1 (counter overflow)



47.4.5 Clock selection

The counter clock can be provided by the following clock sources:

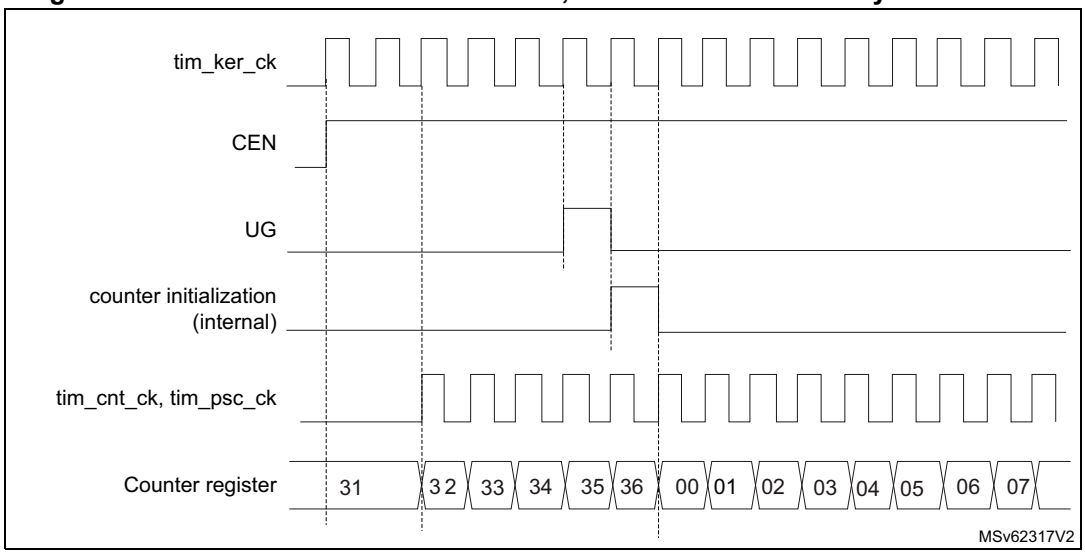
- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2)
- External clock mode2: external trigger input (tim_etr_in)
- Internal trigger inputs (tim_itr): using one timer as prescaler for another timer, for example, Timer 1 can be configured to act as a prescaler for Timer 2. Refer to *: Using one timer as prescaler for another timer on page 1844* for more details.

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS=000 in the TIMx_SMCR register), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 484 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

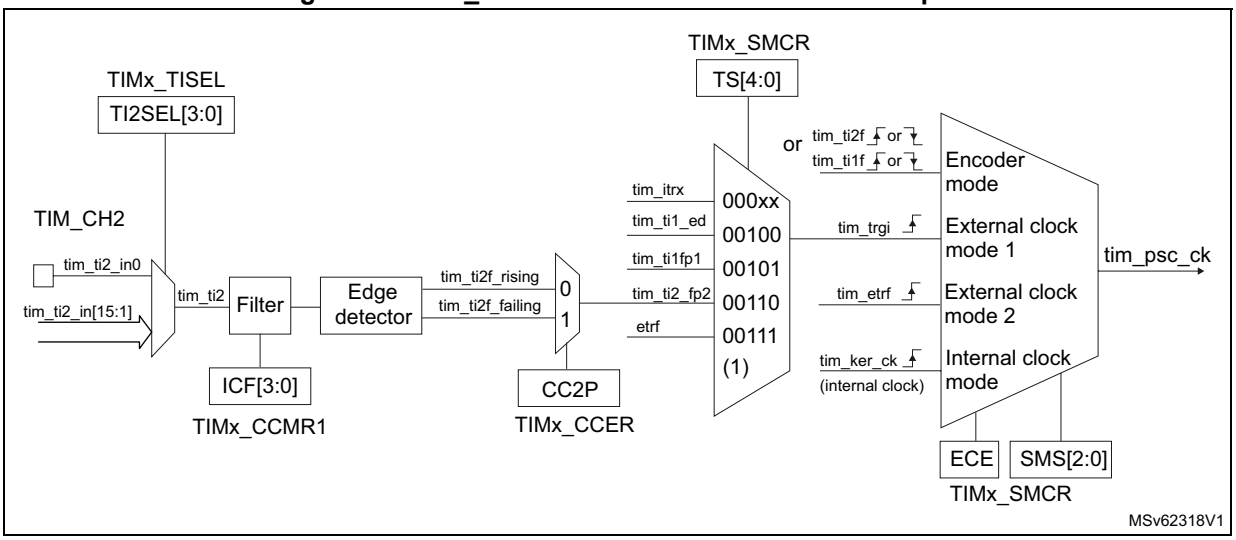
Figure 484. Control circuit in normal mode, internal clock divided by 1



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 485. tim_ti2 external clock connection example



1. Codes ranging from 01000 to 11111: tim_itr[15:0].

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S= ‘01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).

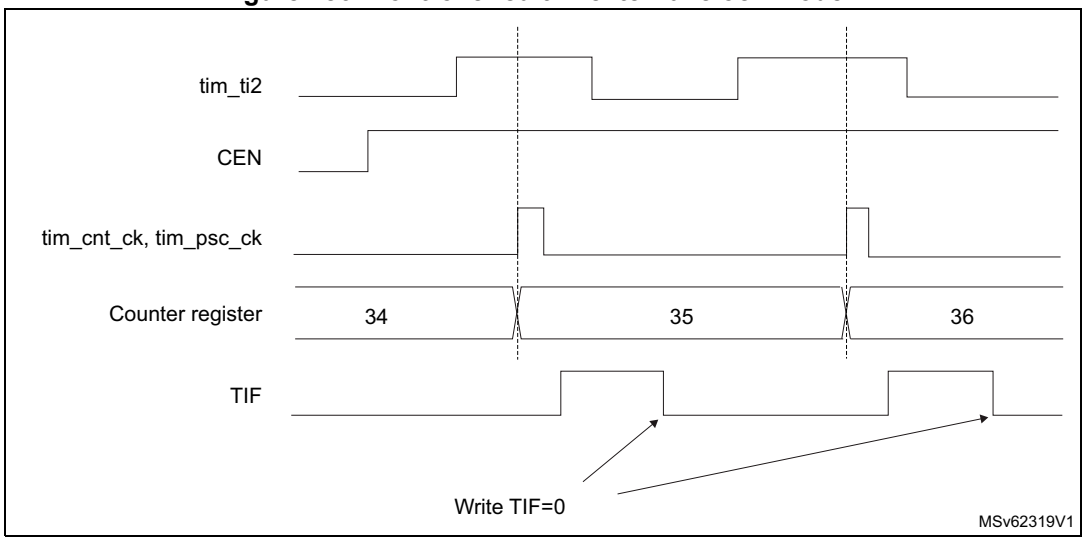
Note: The capture prescaler is not used for triggering, so it does not need to be configured.

4. Select rising edge polarity by writing CC2P=0 and CC2NP=0 and CC2NP=0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
6. Select tim_ti2 as the input source by writing TS=00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 486. Control circuit in external clock mode 1



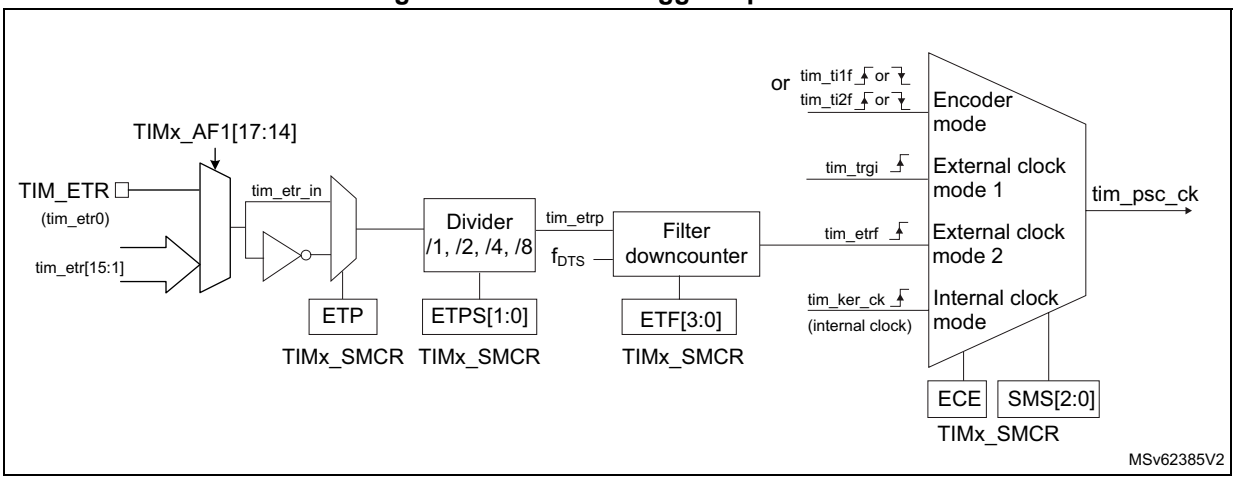
External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input tim_etr_in.

Figure 487 gives an overview of the external trigger input block.

Figure 487. External trigger input block



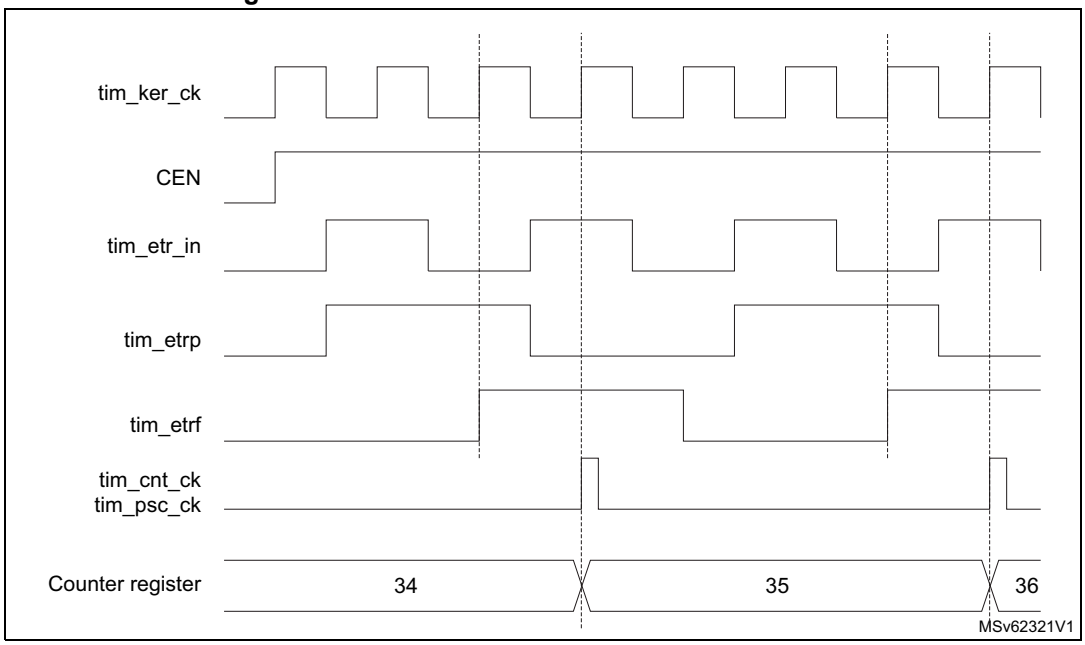
For example, to configure the upcounter to count each 2 rising edges on tim_etr_in, use the following procedure:

1. Select the proper tim_etr_in source (internal or external) with the ETRSEL[3:0] bits in the TIMx_AF1 register.
2. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
3. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register
4. Select rising edge detection on the tim_etr_in by writing ETP=0 in the TIMx_SMCR register
5. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 tim_etr_in rising edges.

The delay between the rising edge on tim_etr_in and the actual clock of the counter is due to the resynchronization circuit on the tim_etrp signal. As a consequence, the maximum frequency that can be correctly captured by the counter is at most ¼ of TIMxCLK frequency. When the ETRP signal is faster, the user should apply a division of the external signal by a proper ETPS prescaler setting.

Figure 488. Control circuit in external clock mode 2



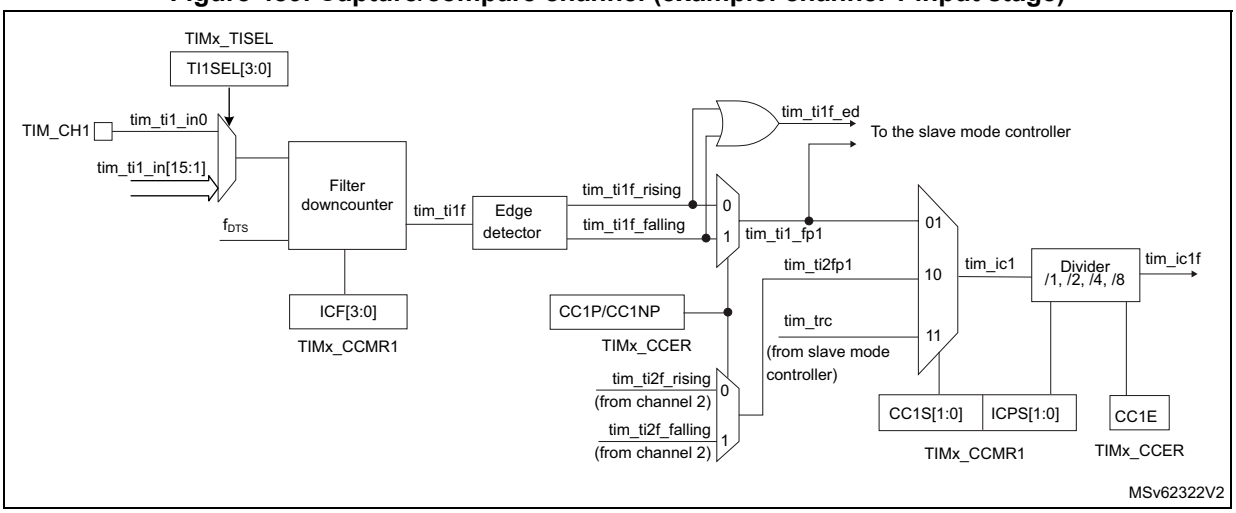
47.4.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figure gives an overview of one Capture/Compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 489. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 490. Capture/compare channel 1 main circuit

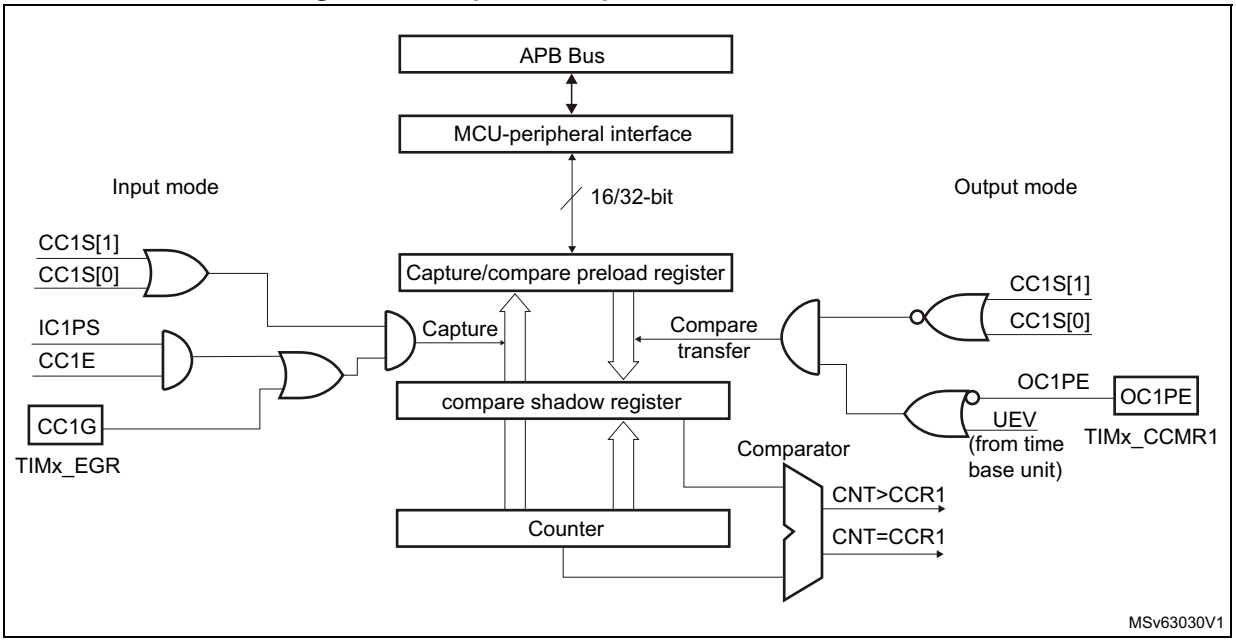
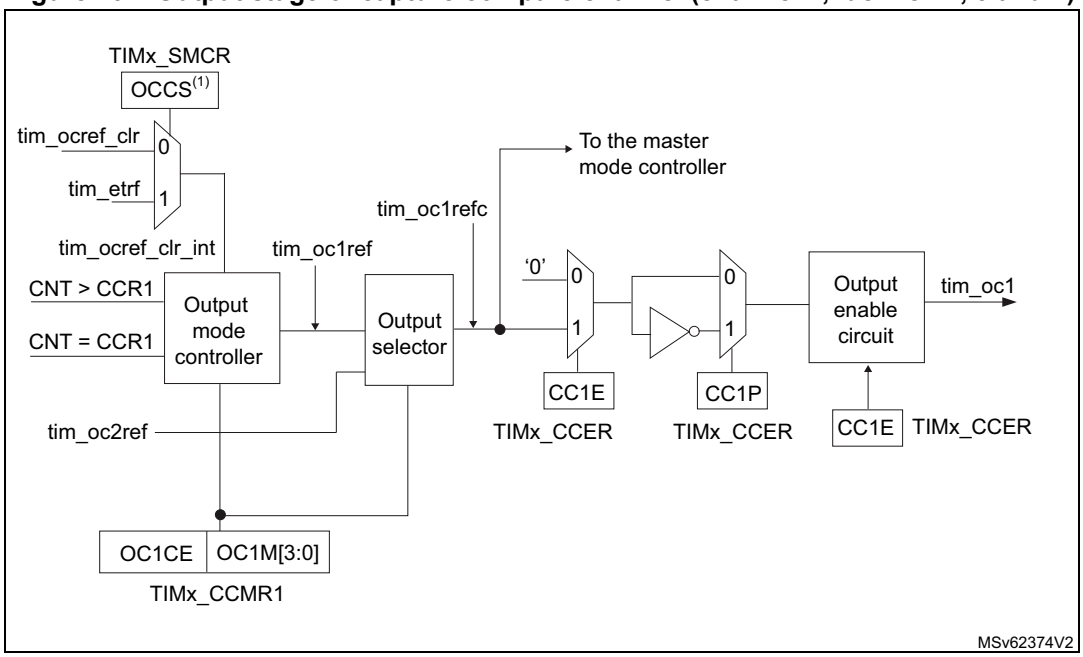


Figure 491. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4)



1. Available on some instances only. If not available, tim_etrf is directly connected to tim_ocref_clr_int.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

47.4.7 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_tix_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the needed input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on tim_ti1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by writing the CC1P and CC1NP and CC1NP bits to 000 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

47.4.8 PWM input mode

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges)
- The TIM_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges

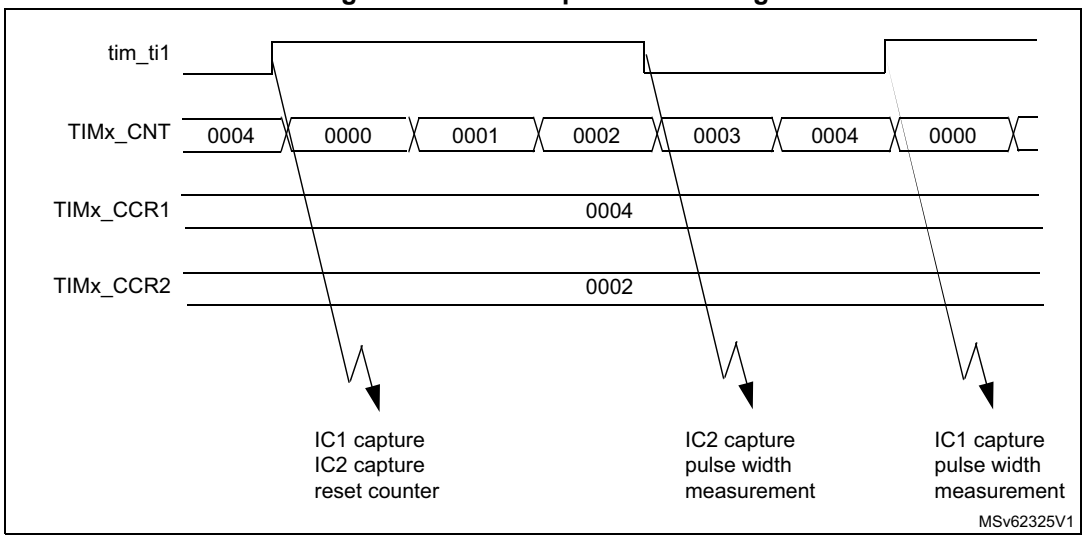
This mode is a particular case of input capture mode. The set-up procedure is similar wit hthe following differences:

- Two ICx signals are mapped on the same tim_tix input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

The period and the pulse width of a PWM signal applied on tim_ti1 can be measured using the following procedure:

1. Select the proper tim_tix_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P to ‘0’ and the CC1NP bit to ‘0’ (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P bit to ‘1’ and the CC2NP bit to ’0’ (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ‘1 in the TIMx_CCER register.

Figure 492. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only tim_ti1fp1 and tim_ti2fp2 are connected to the slave mode controller.

47.4.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

e.g.: CCxP=0 (tim_ocx active high) => tim_ocx is forced to high level.

tim_ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the Output Compare Mode section.

47.4.10 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP

bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

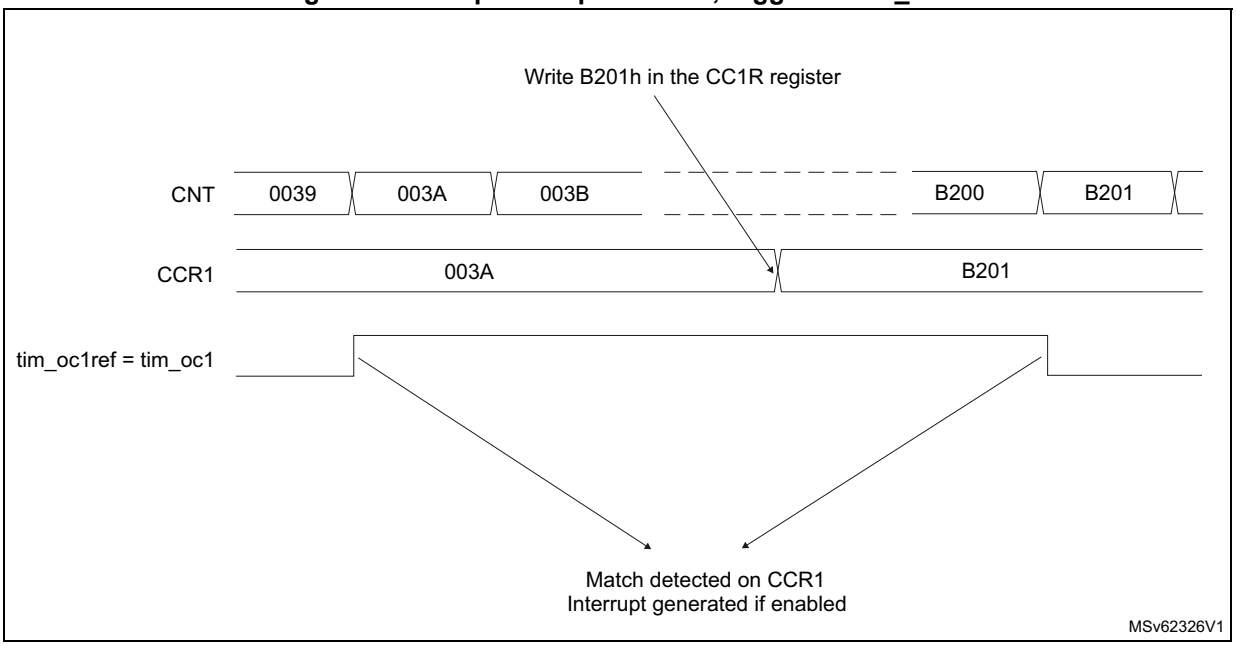
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example:
   a) Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx
   b) Write OCxPE = 0 to disable preload register
   c) Write CCxP = 0 to select active high polarity
   d) Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE=0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 493*.

Figure 493. Output compare mode, toggle on tim_oc1



47.4.11 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 110 (PWM mode 1) or ‘111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter). The tim_ocref_clr can be cleared by an external event through the tim_etr_in or the tim_oceref_clr signals. In this case the tim_ocref_clr signal is asserted only:

- After a compare match event
- When the output compare mode (OCxM bits in TIMx_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM=‘000) to one of the PWM modes (OCxM=‘110 or ‘111). This forces the PWM by software while the timer is running.

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

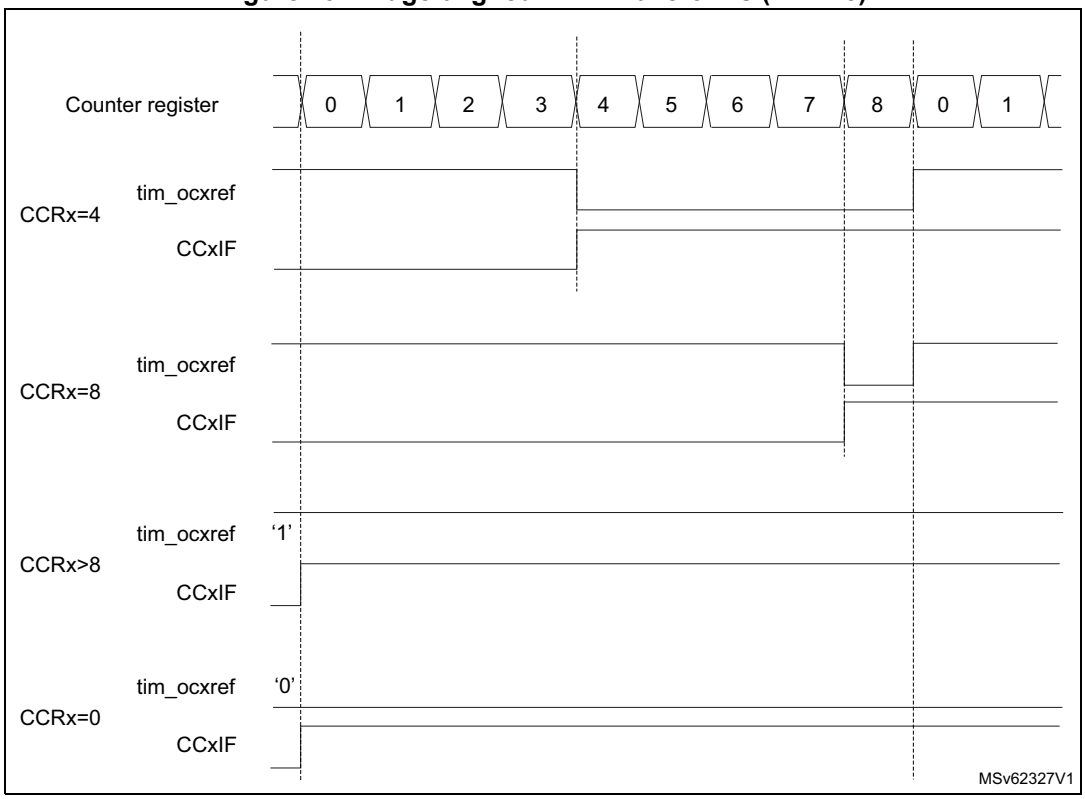
PWM edge-aligned mode

Upcounting configuration

Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Upcounting mode on page 1783*.

In the following example, we consider PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT <TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then tim_ocxref is held at ‘1. If the compare value is 0 then tim_ocxref is held at ‘0. *Figure 494* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 494. 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 1786*.

In PWM mode 1, the reference signal tim_ocxref is low as long as TIMx_CNT>TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then tim_ocxref is held at 100%. PWM is not possible in this mode.

PWM center-aligned mode

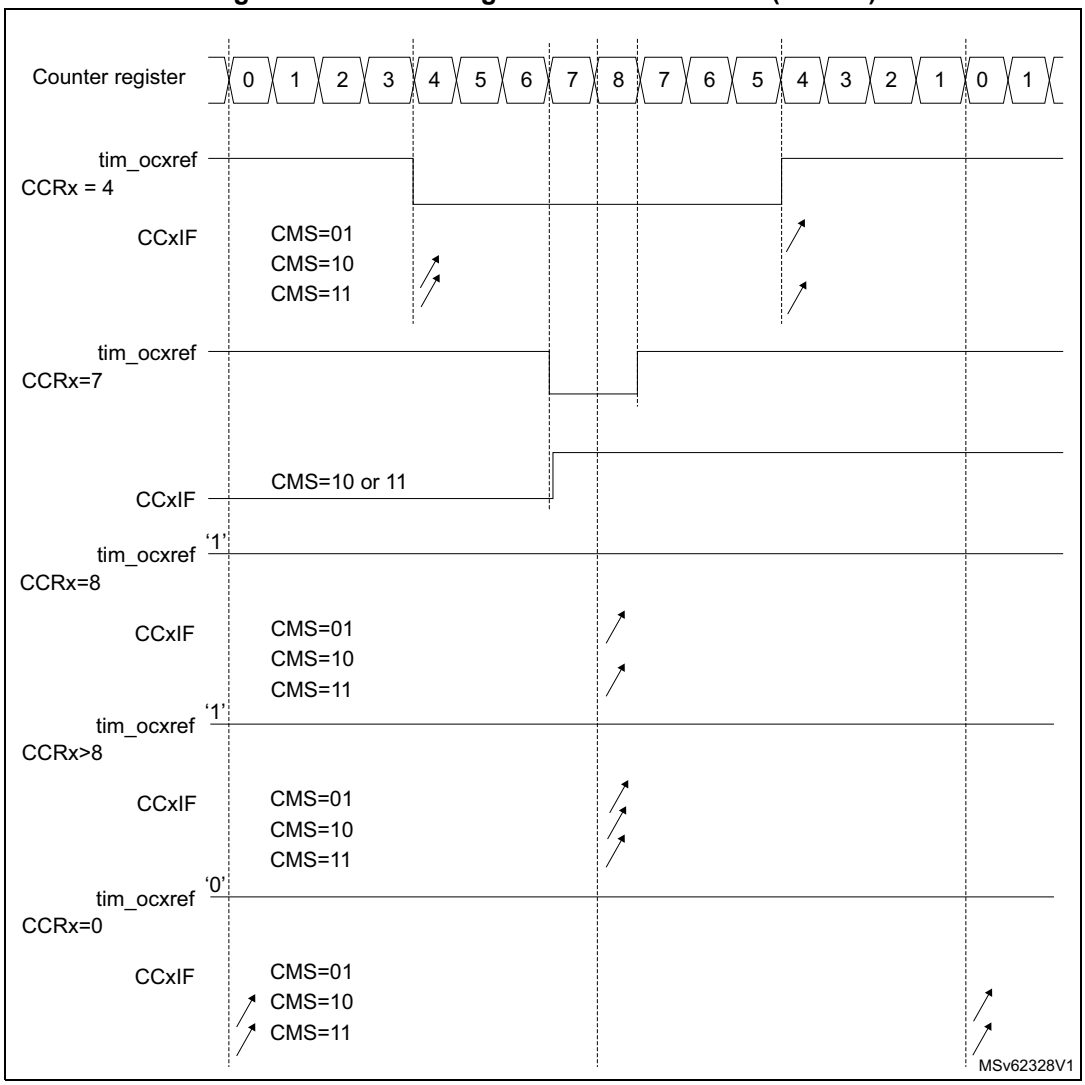
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit

(DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down counting) on page 1789*.

Figure 495 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 495. Center-aligned PWM waveforms (ARR=8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit

in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

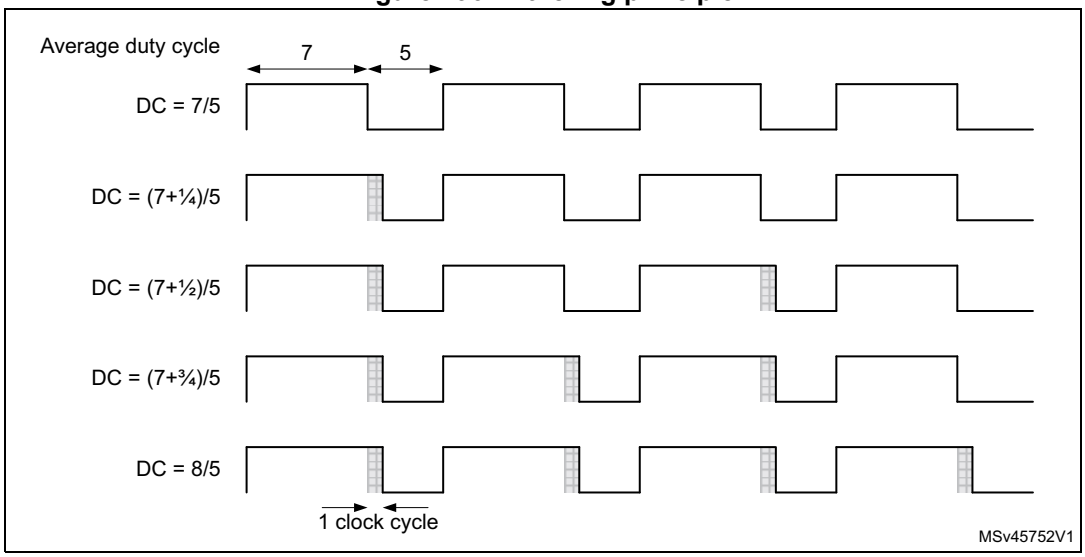
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the auto-reload value is written in the counter (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. The *Figure 496* below presents the dithering principle applied to 4 consecutive PWM cycles.

Figure 496. Dithering principle

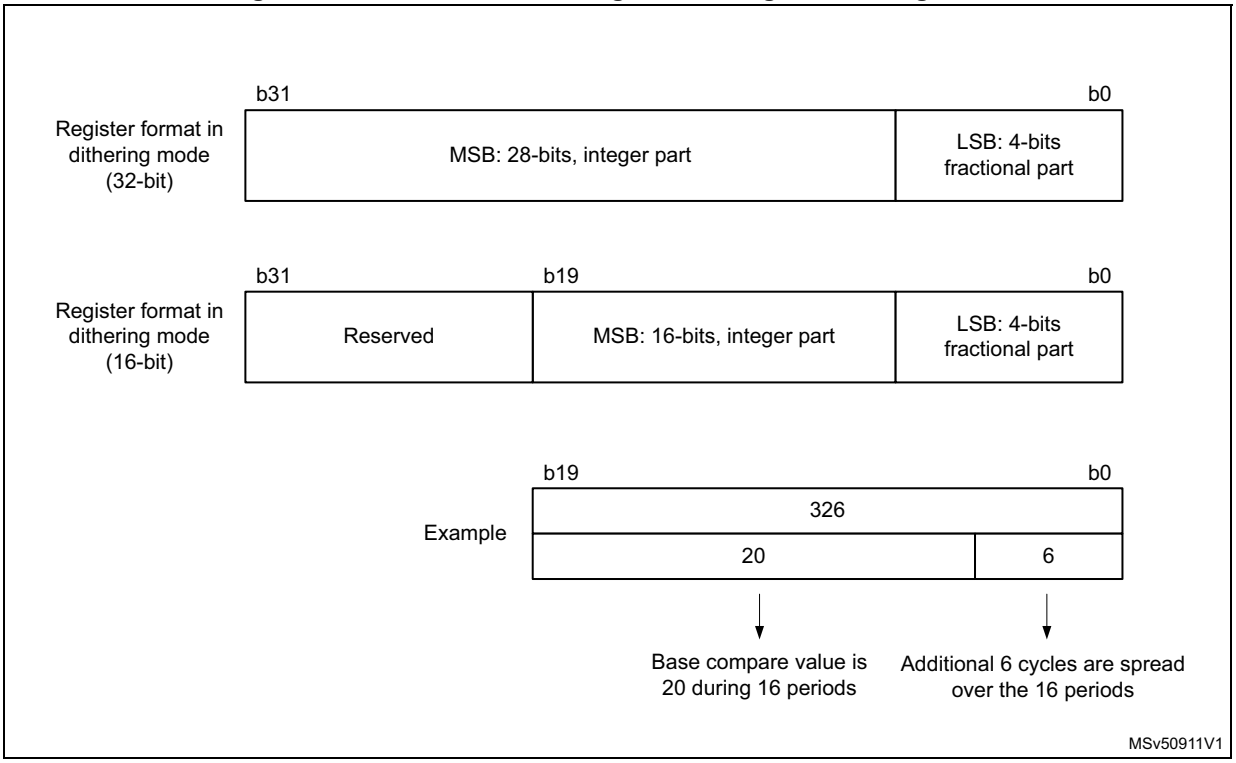


When the dithering mode is enabled, the register coding is changed as following (see *Figure 497* for example):

- The 4 LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted by 4 places and are coding for the base value. In 16-bit mode, the 16-bit format is maintained.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The ARR[3:0] bits must be reset
3. The DITHEN bit must be reset
4. The CCIF flags must be cleared
5. The CEN bit can be set (eventually with ARPE = 1)

Figure 497. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode (16-bit timer): } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

$$\text{Dithering mode (32-bit timer): } F_{pwmMin} = \frac{F_{Tim}}{268435454 + \frac{15}{16}}$$

Note: For 16-bit timers, the maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part). For 32-bit timers, the maximum TIMx_ARR and TIMxCCRy values are limited to

0xFFFFFFEF in dithering mode (corresponds to 264435454 for the integer part and 15 for the dithered part).

As shown on the *Figure 498* and *Figure 499* below, the dithering mode is used to increase the PWM resolution.

Figure 498. PWM resolution vs frequency (16-bit mode)

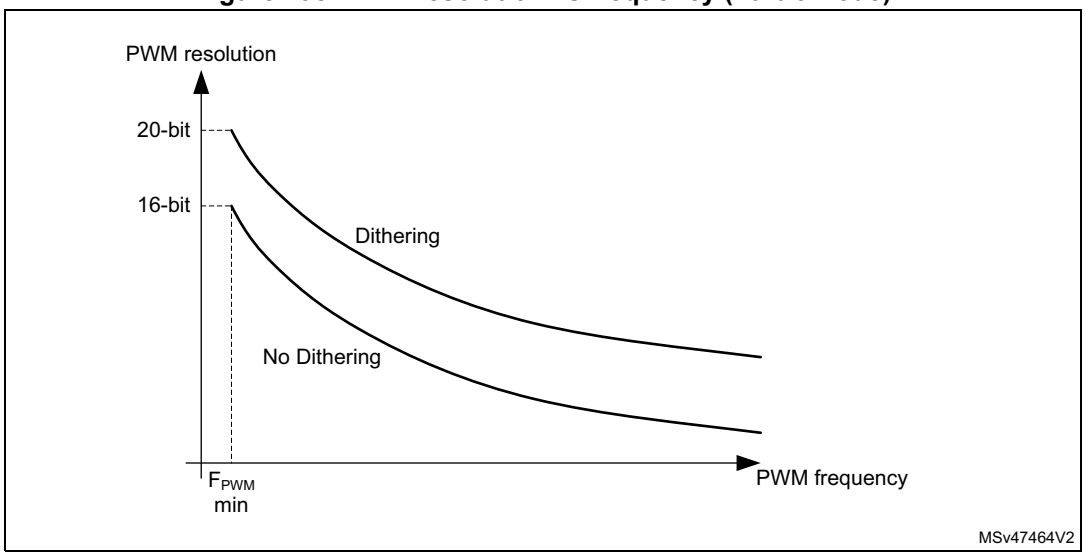
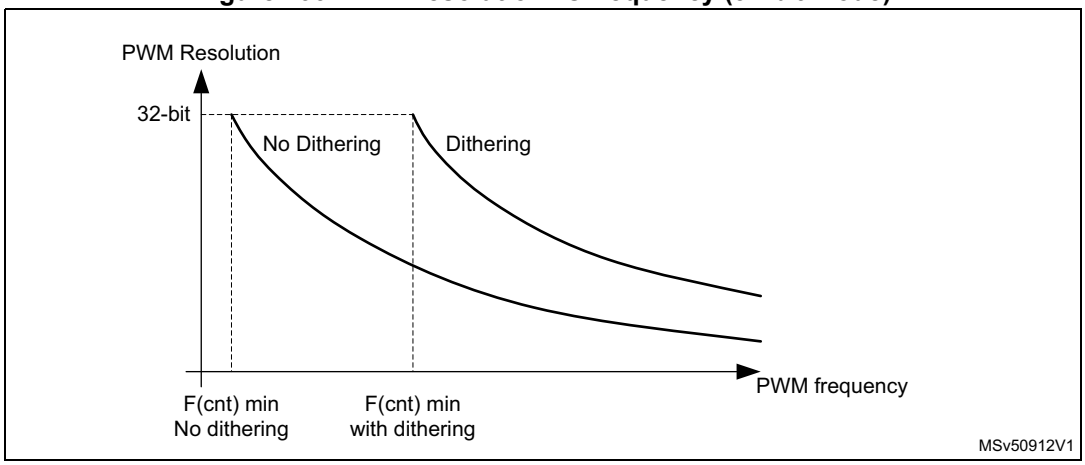
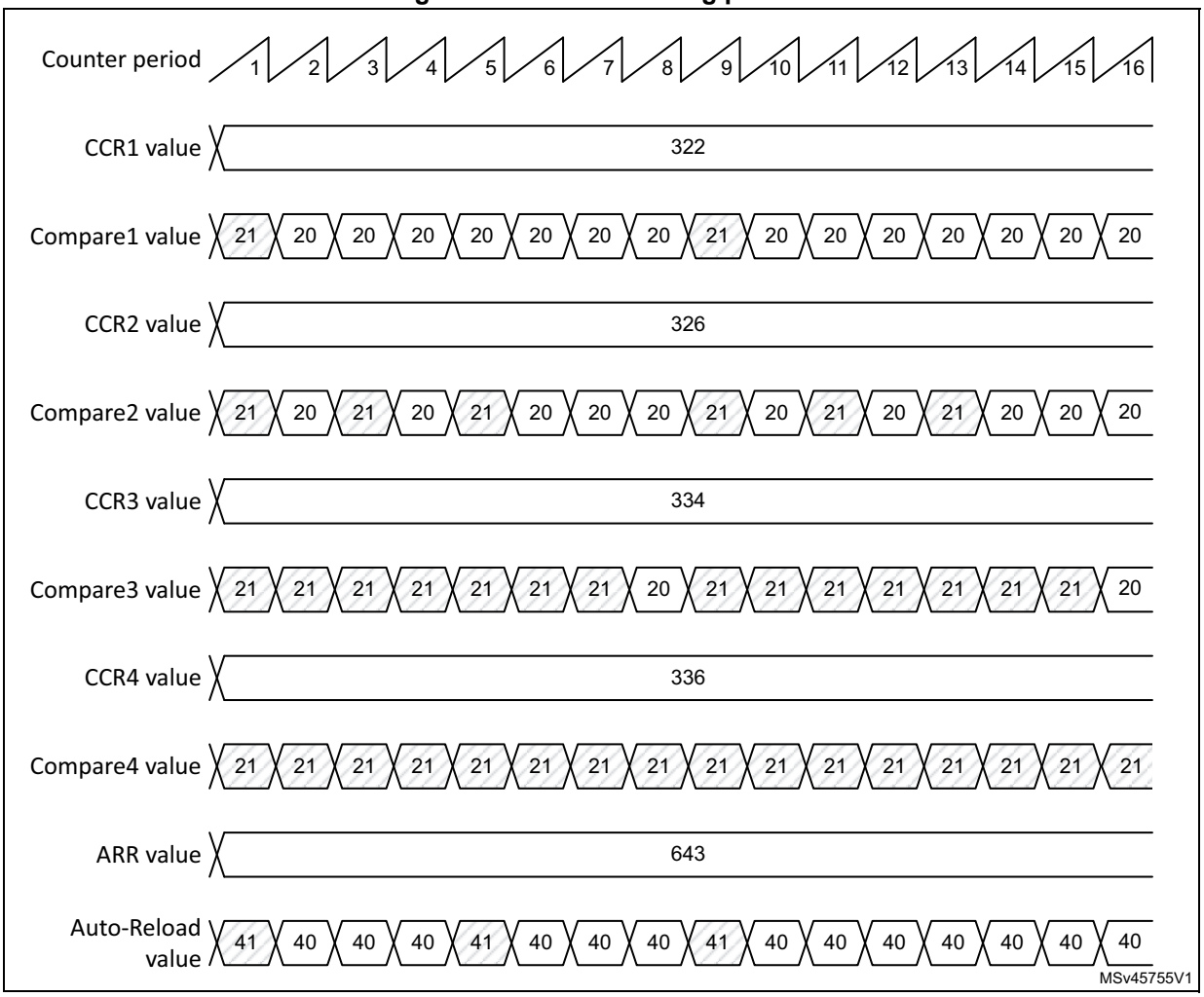


Figure 499. PWM resolution vs frequency (32-bit mode)



The duty cycle and / or period changes are spread over 16 consecutive periods, as described in the *Figure 500* below.

Figure 500. PWM dithering pattern



The auto-reload and compare values increments are spread following specific patterns described in the *Table 457* below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 457. CCR and ARR register change dithering pattern

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIMx_CR1 register are not equal to '00'). In this case, the dithering pattern is applied over 8 consecutive PWM periods, considering the up and down counting phases as shown in the *Figure 501* below.

Figure 501. Dithering effect on duty cycle in center-aligned PWM mode

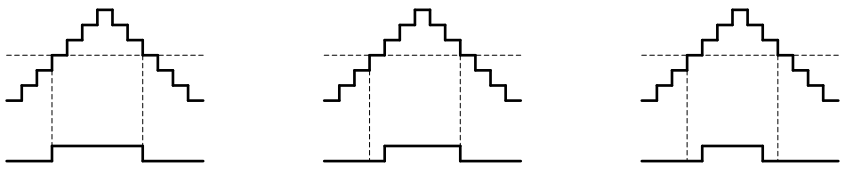


Table 458 below shows how the dithering pattern is added in center-aligned PWM mode.

Table 458. CCR register change dithering pattern in center-aligned PWM mode

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

47.4.12 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx registers. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

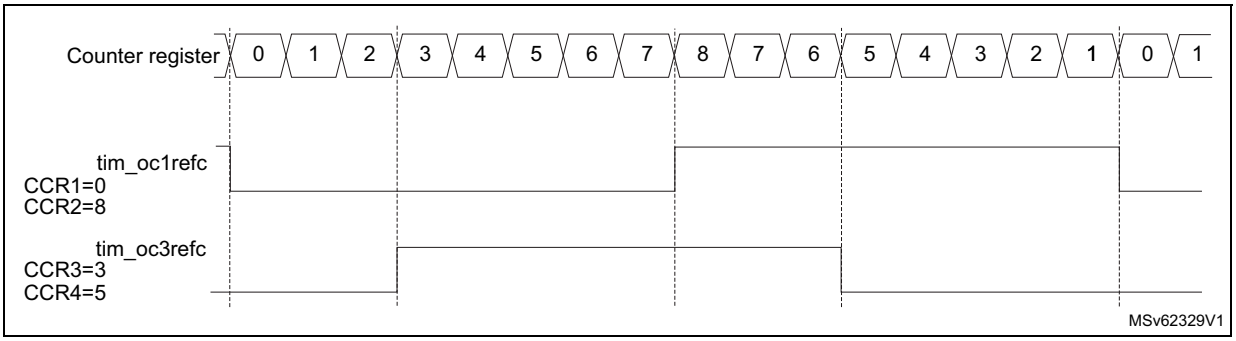
Asymmetric PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

When a given channel is used as asymmetric PWM channel, its secondary channel can also be used. For instance, if an tim_oc1refc signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the tim_oc2ref signal on channel 2, or an tim_oc2refc signal resulting from asymmetric PWM mode 2.

Figure 502 shows an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 2).

Figure 502. Generation of 2 phase-shifted PWM signals with 50% duty cycle



47.4.13 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

– tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2

– tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

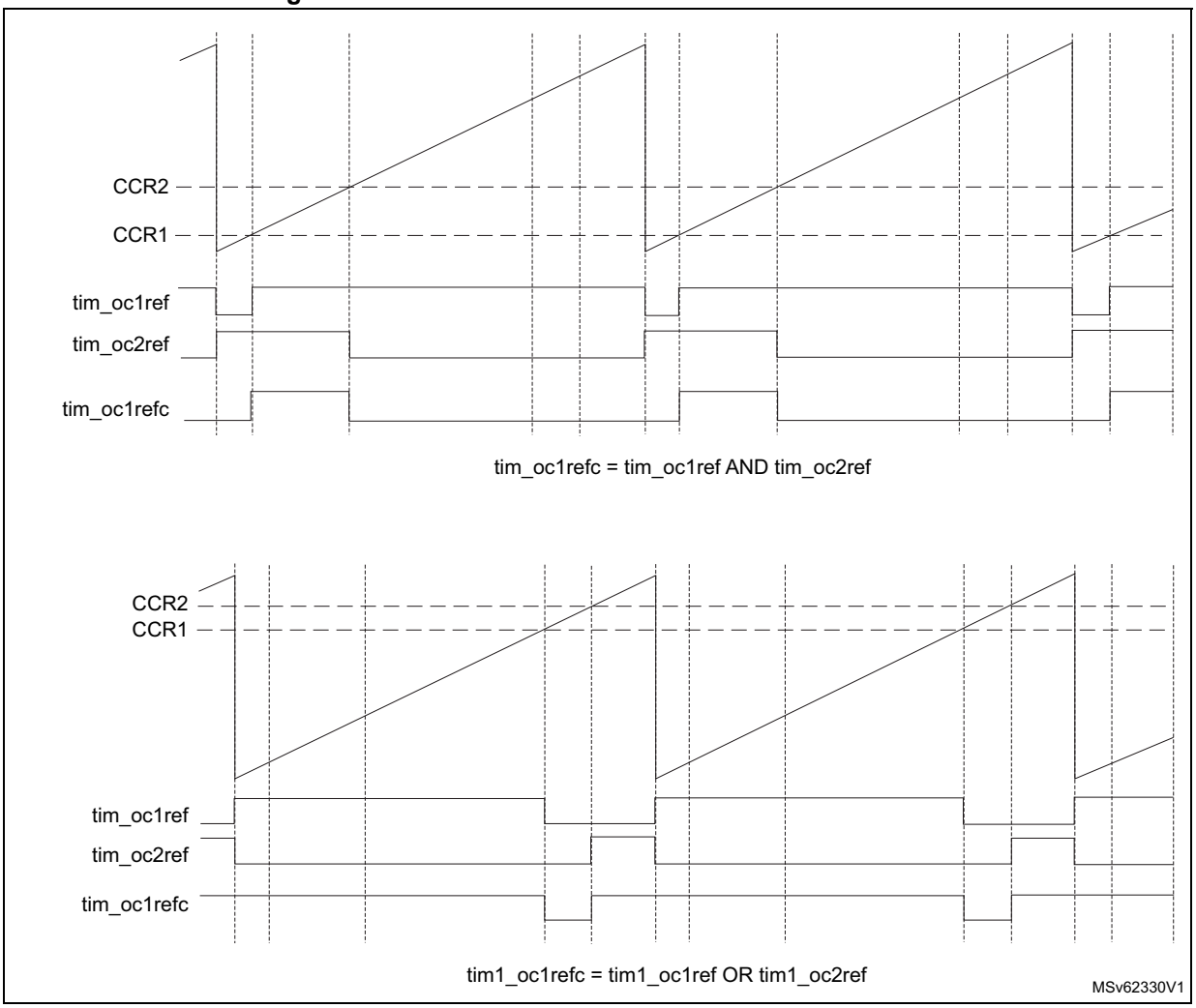
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 503 shows an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
- Channel 3 is configured in Combined PWM mode 2,
- Channel 4 is configured in PWM mode 1

Figure 503. Combined PWM mode on channels 1 and 3



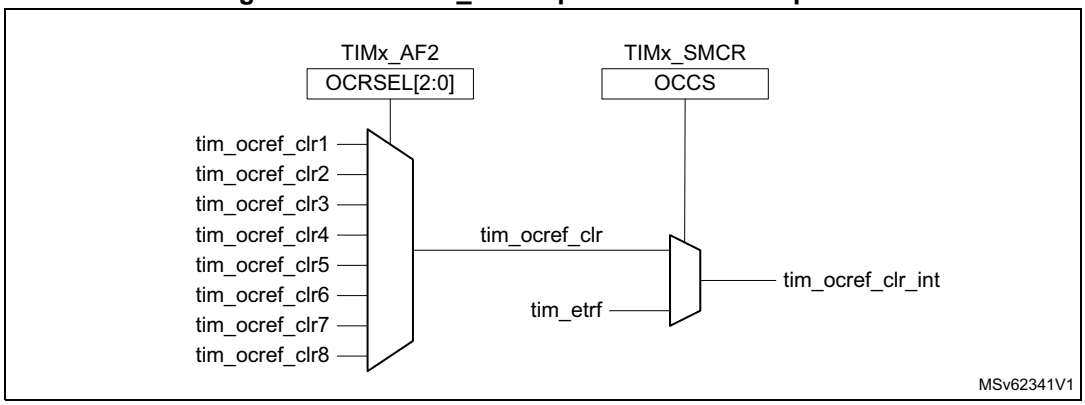
47.4.14 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next update event (UEV) occurs. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

The tim_ocref_clr_int source depends on the OCREF clear selection feature implementation, refer to *Section 47.3: TIM2/TIM3/TIM4/TIM5 implementation*.

If the OCREF clear selection feature is implemented, the tim_ocref_clr_int can be selected between the tim_ocref_clr input and the tim_etrf input (tim_etr_in after the filter) by configuring the OCCS bit in the TIMx_SMCR register. The tim_ocref_clr input can be selected among several tim_ocref_clr[7:0] inputs, using the OCRSEL[2:0] bitfield in the TIMx_AF2 register, as shown in *Figure 504* below.

Figure 504. OCREF_CLR input selection multiplexer



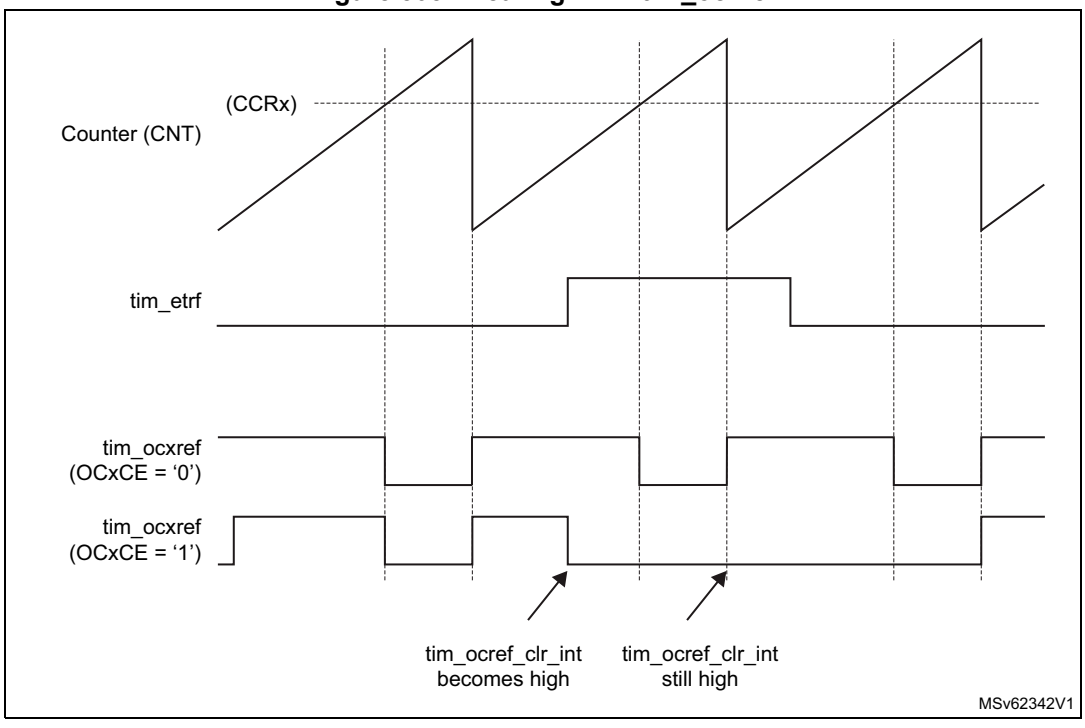
If the OCREF clear selection feature is not implemented, the tim_ocref_clr_int input is directly connected to the tim_etrf input.

For example, the tim_ocref_clr_int signal can be connected to the output of a comparator to be used for current handling. In this case, tim_etr_in must be configured as follows:

1. The external trigger prescaler should be kept off: bits ETPS[1:0] in the TIMx_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIM1_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application's needs.

Figure 505 shows the behavior of the tim_ocxref signal when the tim_etrf input becomes high, for both values of the OCxCE enable bit. In this example, the timer TIMx is programmed in PWM mode.

Figure 505. Clearing TIMx tim_ocxref



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), tim_ocxref is enabled again at the next counter overflow.

47.4.15 One-pulse mode

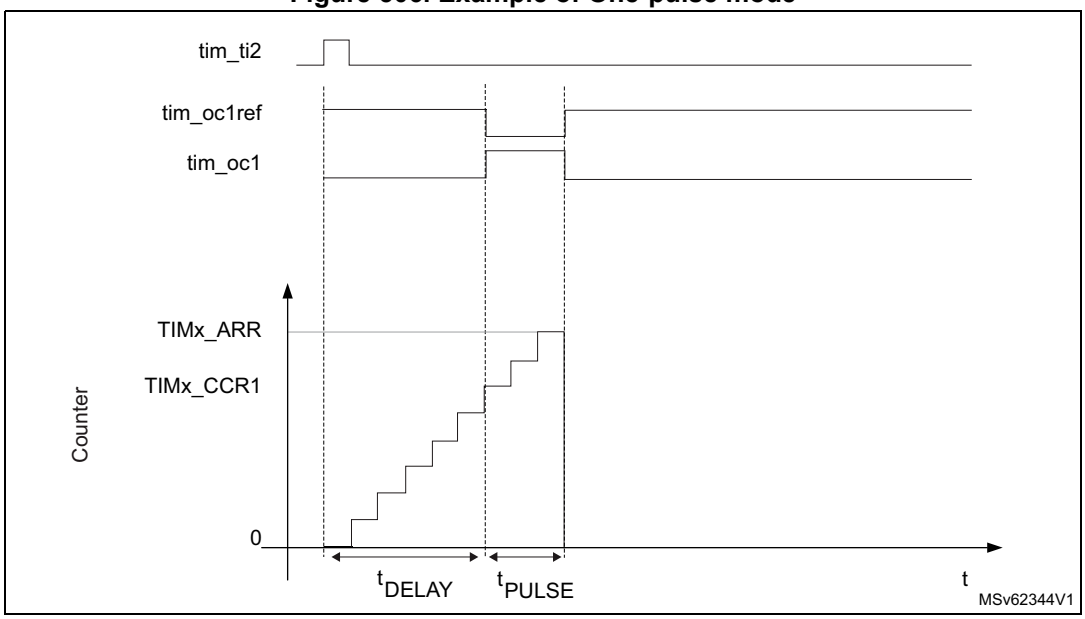
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

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 506. Example of One-pulse mode



For example one may want to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Let's use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map tim_ti2fp2 on tim_ti2 by writing CC2S=01 in the TIMx_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P=0 and CC2NP='0' in the TIMx_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS=00110 in the TIMx_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to '110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one want to build a waveform with a transition from '0 to '1 when a compare match occurs and a transition from '1 to '0 when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE=1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to '0 in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: tim_ocx fast enable:

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

47.4.16 Retriggerable one-pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one-pulse mode described in *Section 47.4.15*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for Retriggerable OPM mode 1 or 2.

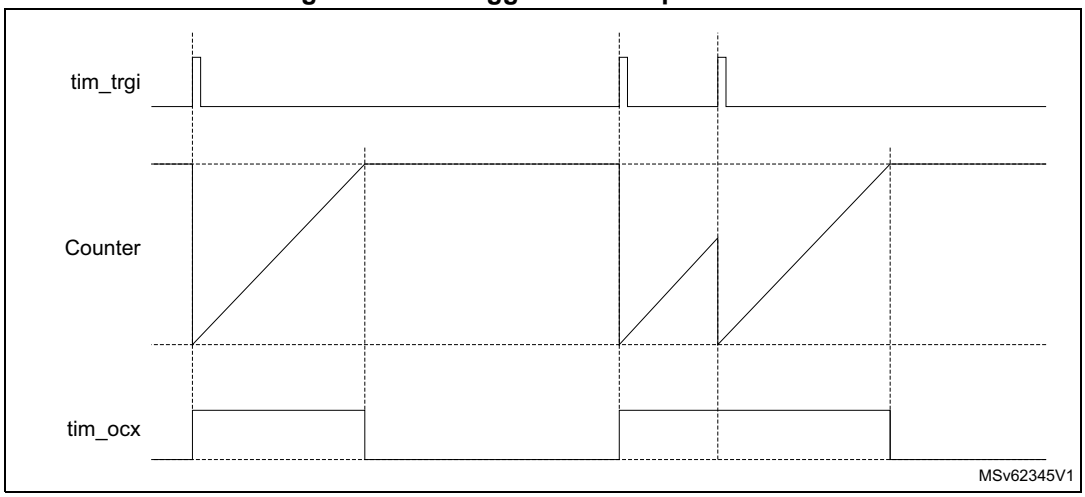
If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode CCRx must be above or equal to ARR.

Note: In Retriggerable one-pulse mode, the CCxIF flag is not significant.

The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 507. Retriggerable one-pulse mode

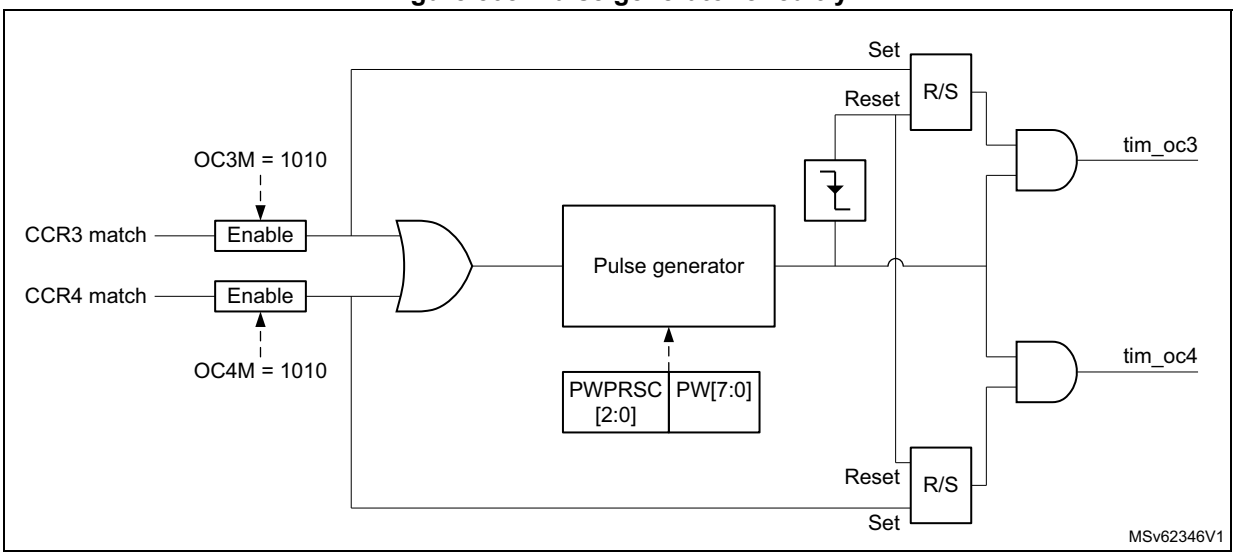


47.4.17 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulse width generated when the counter value equals a given compare value, for debugging or synchronization purposes.

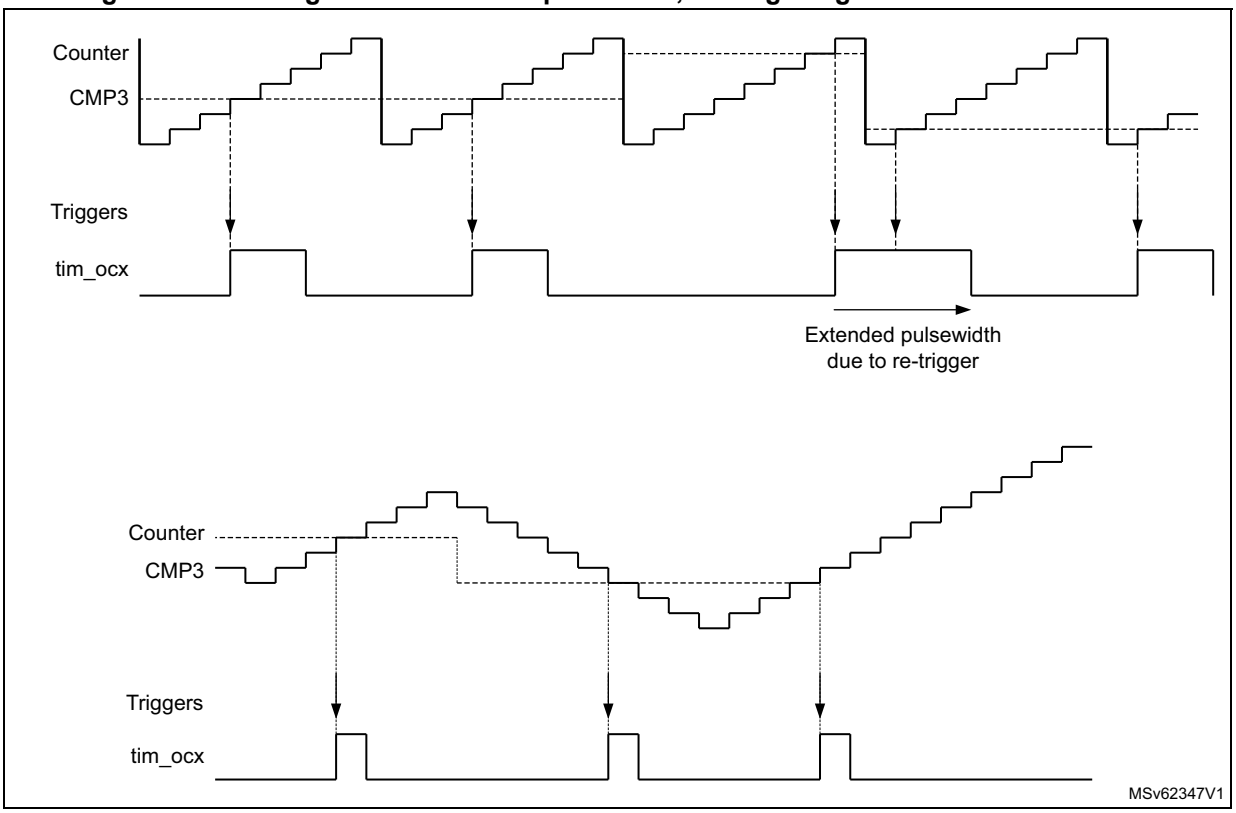
This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on the *Figure 508* below.

Figure 508. Pulse generator circuitry



The *Figure 509* below shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 509. Pulse generation on compare event, for edge-aligned and encoder modes



This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bit fields in TIMx_CCMR2 register.

The pulse width is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

t_{PW} = PW[7:0] x t_{PWG}

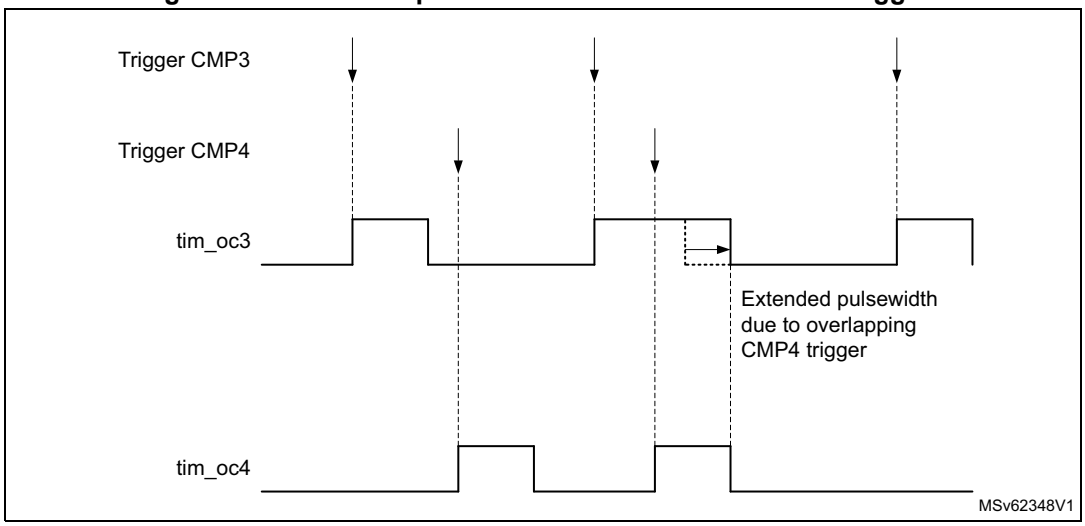
where $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim\_ker\_ck}$.

gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is ongoing, causes the pulse to be extended.

Note: If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the 1st arriving trigger is extended (because of the re-trigger), while the pulse width of the last arriving trigger is correct (as shown on the Figure 510 below).

Figure 510. Extended pulse width in case of concurrent triggers



47.4.18 Encoder interface mode

Quadrature encoder

To select Encoder Interface mode write SMS='0001 in the TIMx_SMCR register if the counter is counting on tim_ti1 edges only, SMS=0010 if it is counting on tim_ti2 edges only and SMS=0011 if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. CC1NP and CC2NP must be kept cleared. When needed, the input filter can be programmed as well.

The two inputs tim_ti1 and tim_ti2 are used to interface to an incremental encoder. Refer to *Table 459*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1=tim_ti1 if not filtered and not inverted, tim_ti2fp2=tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to '1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.

Table 459. Counting direction versus encoder signals(CC1P = CC2P = 0)

Active edge	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	1110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	1111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	0001	High	Down	Up	No count	No count
		Low	Up	Down	No count	Down
Counting on tim_ti2 only x2 mode	0010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	0011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to the external trigger input and trigger a counter reset.

Figure 511 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S= 01 (TIMx_CCMR1 register, tim_ti1fp1 mapped on tim_ti1)
- CC2S= 01 (TIMx_CCMR2 register, tim_ti2fp2 mapped on tim_ti2)
- CC1P and CC1NP = '0' (TIMx_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1=tim_ti1)
- CC2P and CC2NP = '0' (TIMx_CCER register, tim_ti2fp2 noninverted, tim_ti2fp2=tim_ti2)
- SMS= 011 (TIMx_SMCR register, both inputs are active on both rising and falling edges)
- CEN= 1 (TIMx_CR1 register, Counter is enabled)

Figure 511. Example of counter operation in encoder interface mode

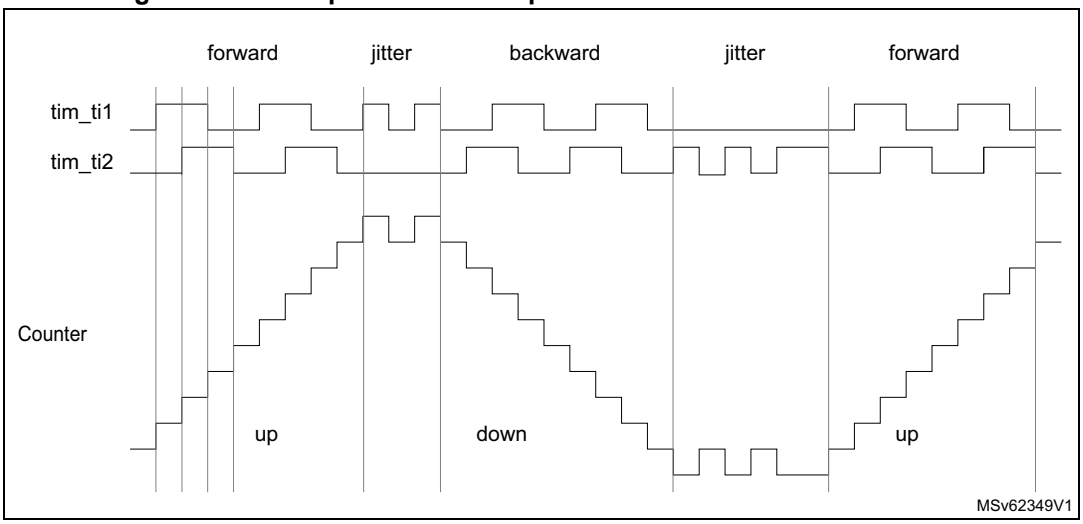
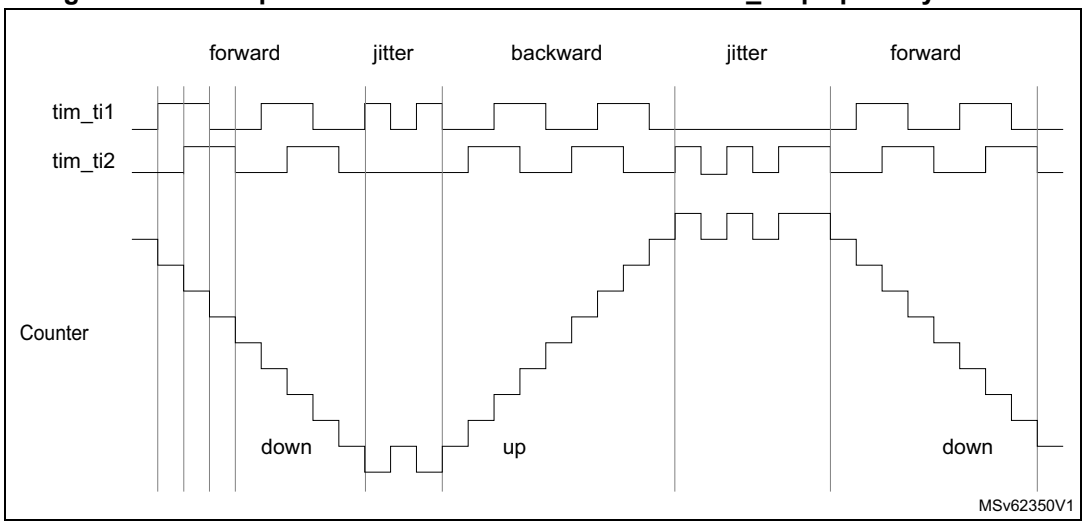


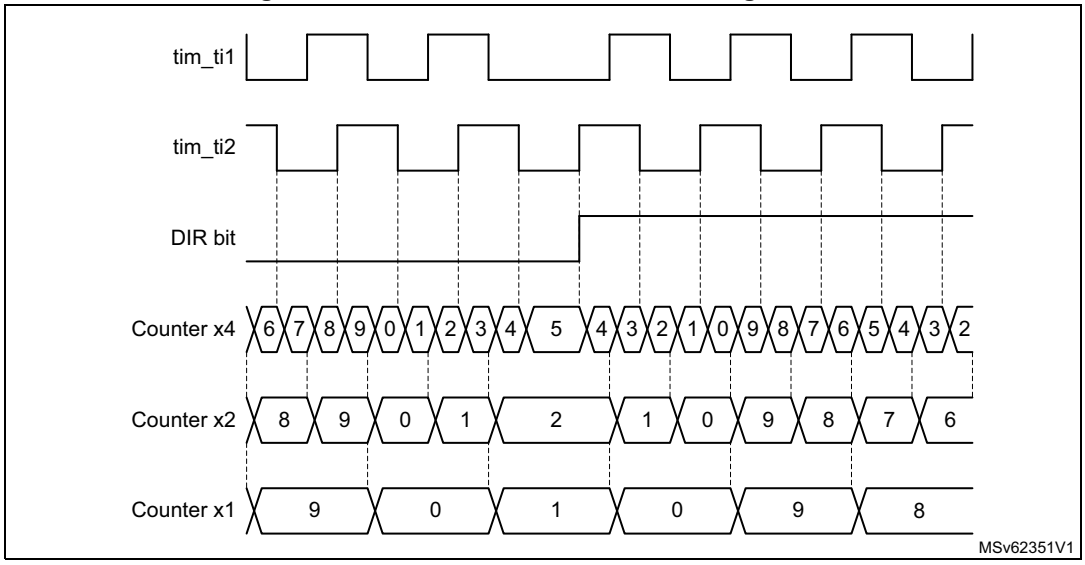
Figure 512 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P=1).

Figure 512. Example of encoder interface mode with tim_ti1fp1 polarity inverted



The *Figure 513* below shows the timer counter value during a speed reversal, for various counting modes.

Figure 513. Quadrature encoder counting modes



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support other types of encoders.

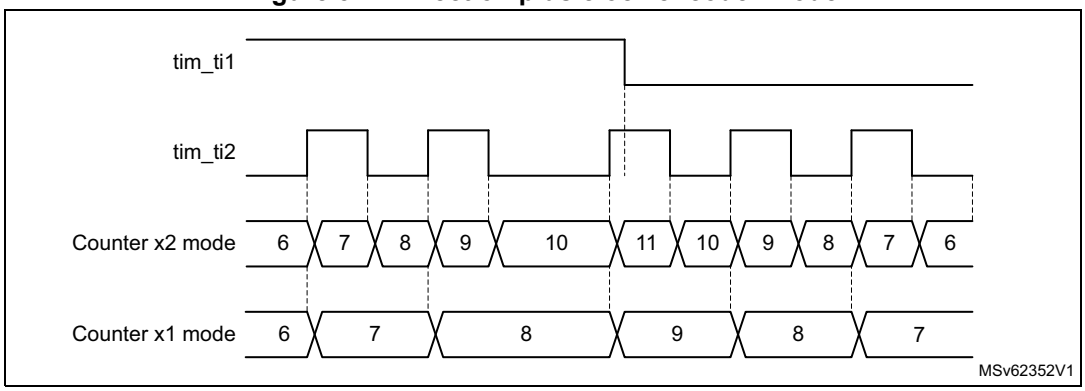
In the "clock plus direction" mode shown on *Figure 514*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1010: x2 mode, the counter is updated on both rising and falling edges of the clock
- 1011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 514. Direction plus clock encoder mode



Directional Clock encoder mode

In the “directional clock” mode on *Figure 515*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock line. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 515* below) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 516* below).
- 1101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 515* below), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 516* below).

Figure 515. Directional clock encoder mode (CC1P = CC2P = 0)

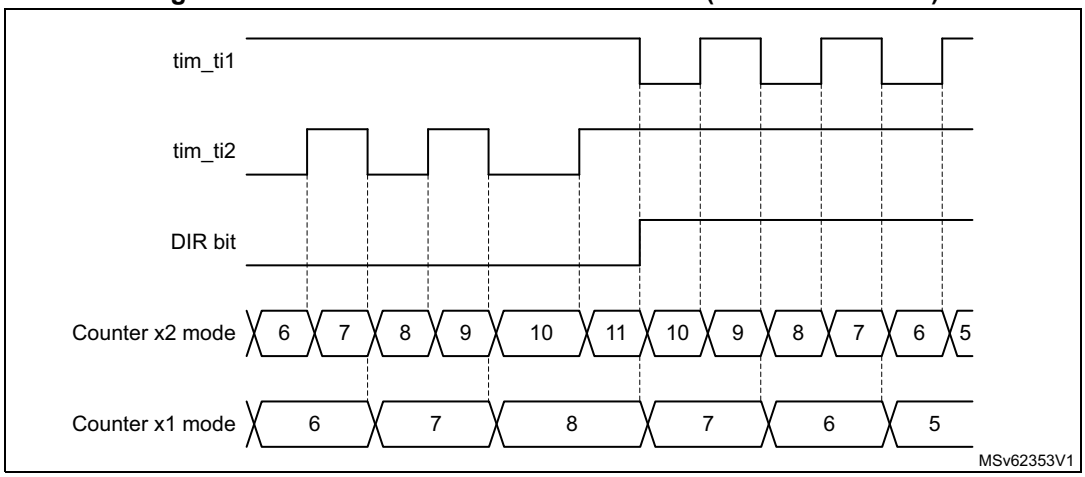
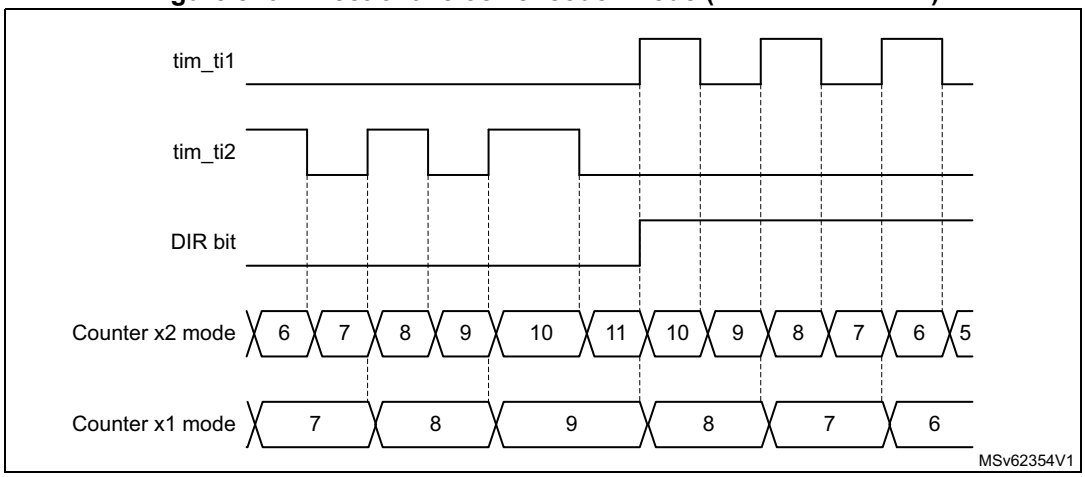


Figure 516. Directional clock encoder mode (CC1P = CC2P = 1)



The *Table 460* here-below details how the directional clock mode operates, for any input transition.

Table 460. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP=0	1100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP=1	1100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP=0	1101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP=1	1101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Index Input

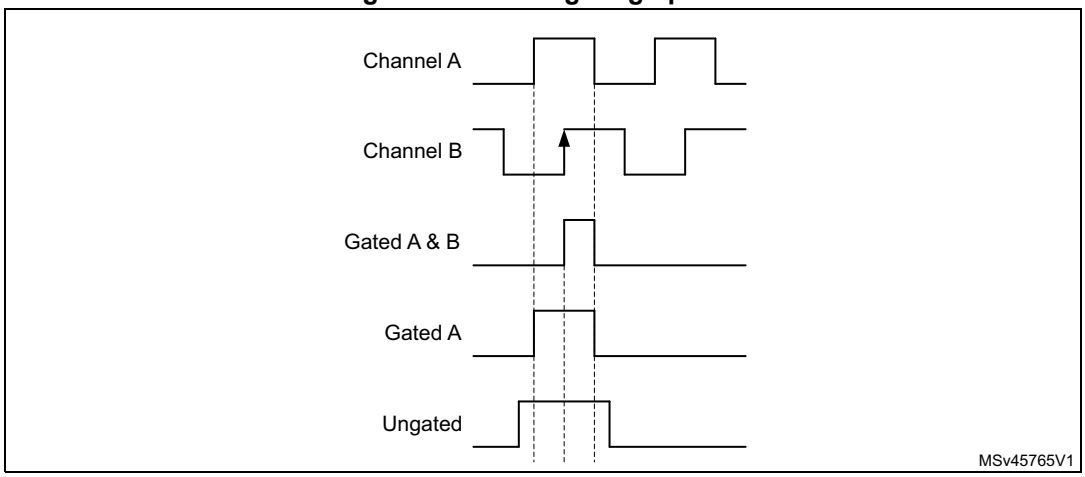
The counter can be reset by an Index signal coming from the encoder, indicating an absolute reference position. The Index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIMx_ECR register. The IE bit must be set only in encoder mode, when the SMS[3:0] bitfield has the following values: 0001, 0010, 011, 1010, 1011, 1100, 1101, 1110, 1111.

Commercially available encoders are proposed with several options for index pulse conditioning, as per the *Figure 517* below:

- gated with A and B: the pulse width is 1/4 of one channel period, aligned with both A and B edges
- gated with A (or gated with B): the pulse width is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B)
- ungated: the pulse width is up to one channel period, without any alignment to the edges

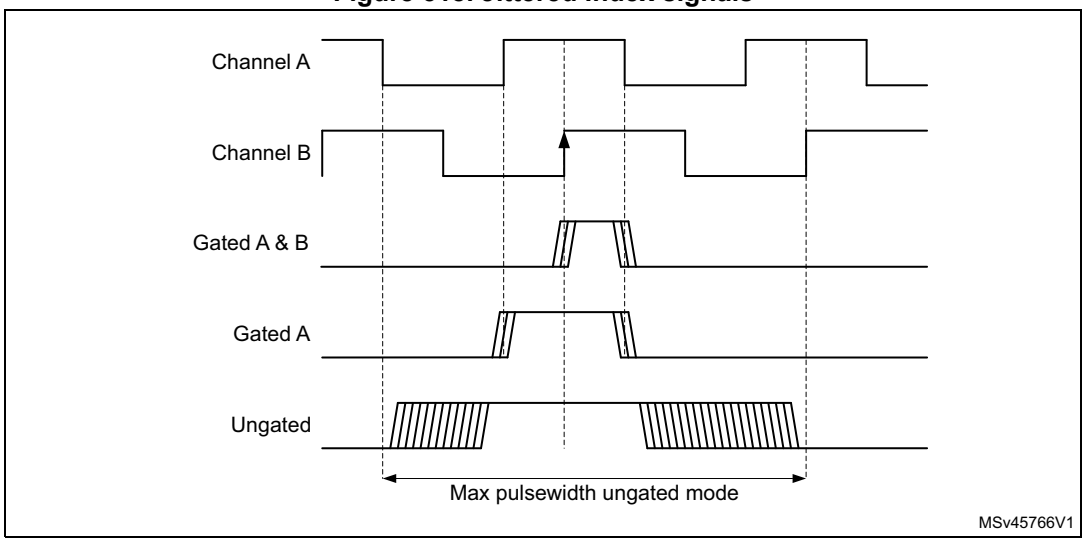
Figure 517. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as show on *Figure 518* below.

In ungated mode, the signal must be strictly below 2 encoder periods. If the pulse width is greater or equal to 2 encoder period, the counter is reset multiple times.

Figure 518. Jittered Index signals



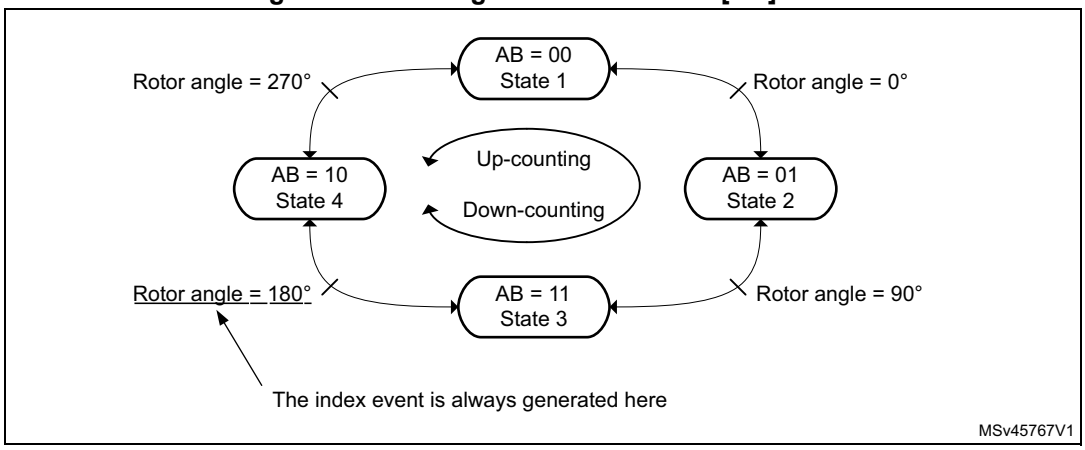
The timer supports the 3 gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (i.e. channel A and channel B state combination) the index must be synchronized, using the IPOS[1:0] bitfield in the TIMx_ECR register.

The Index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0)
- The counter is set to TIMx_ARR when down counting

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. The *Figure 519* below shows at which position is the index generated, for a simplistic example (an encoder providing 4 edges par mechanical rotation).

Figure 519. Index generation for IPOS[1:0] = 11



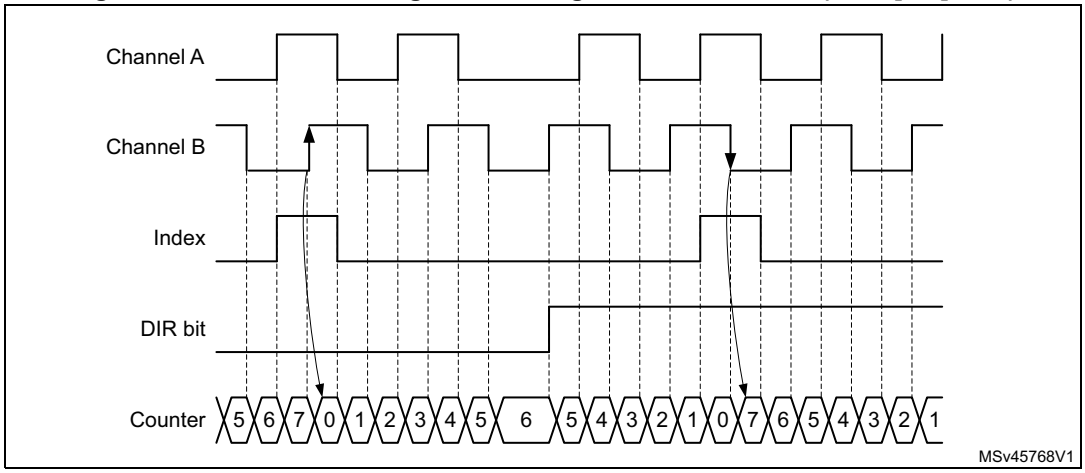
The *Figure 520* below presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is '11' (ChA=1, ChB=1), when up-counting (DIR bit = 0).
- Counter set to TIMx_ARR when exiting the '11' state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

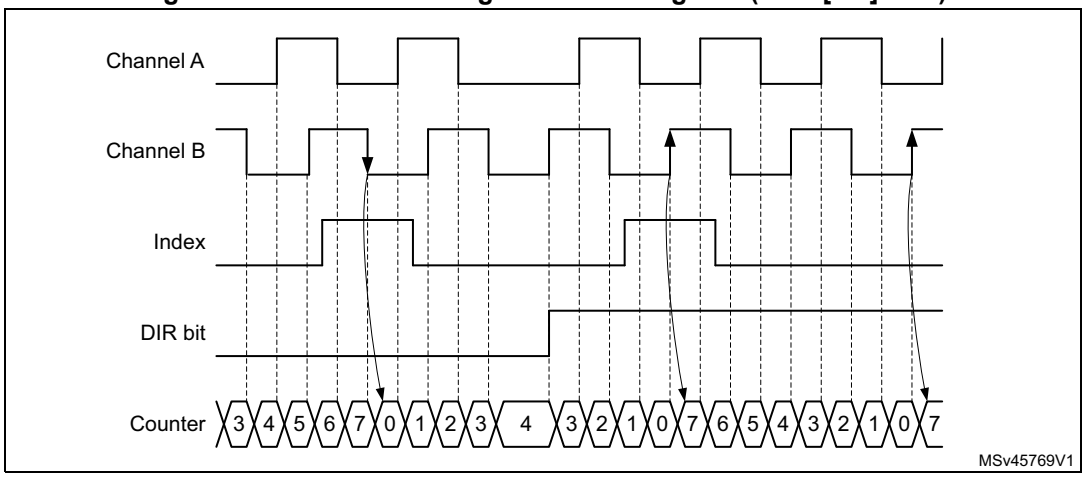
The arrows are indicating on which transition is the index event interrupt generated.

Figure 520. Counter reading with index gated on channel A (IPOS[1:0] = 11)



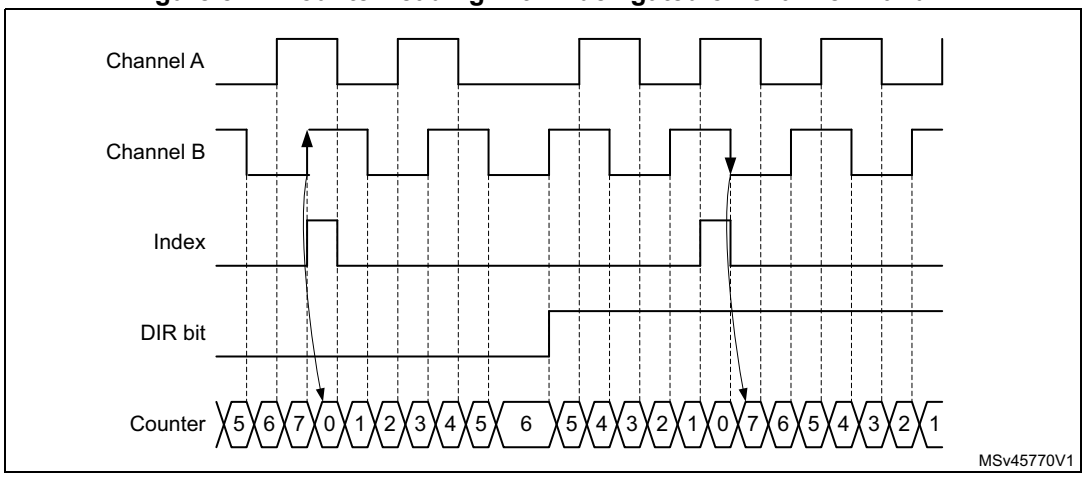
The *Figure 521* below presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 521. Counter reading with index ungated (IPOS[1:0] = 00)



The *Figure 522* below shows how the 'gated on A & B' mode is handled, for various pulse alignment scenario. The arrows are indicating on which transition is the index event generated.

Figure 522. Counter reading with index gated on channel A and B



The *Figure 523* and *Figure 524* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 523. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

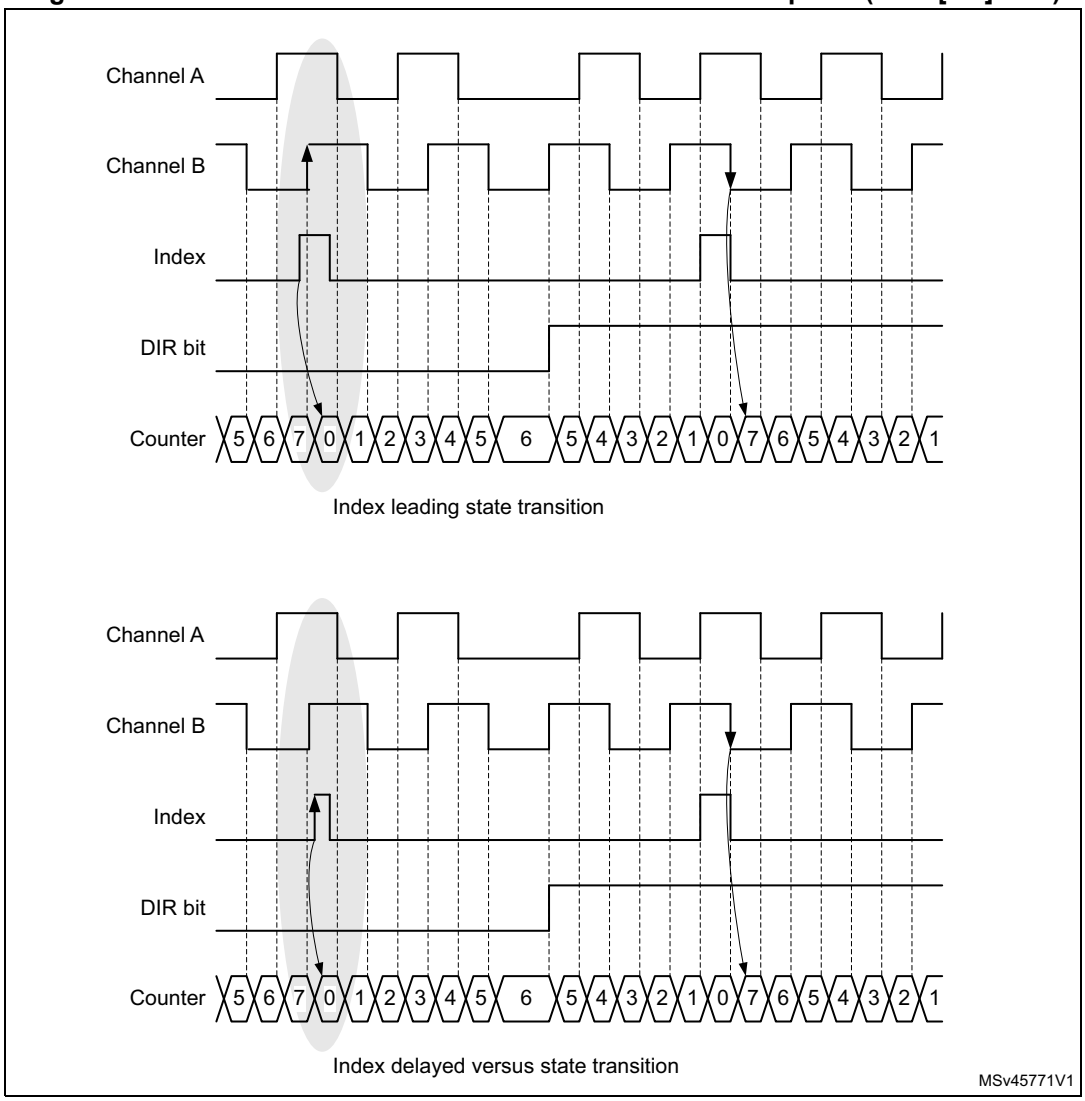
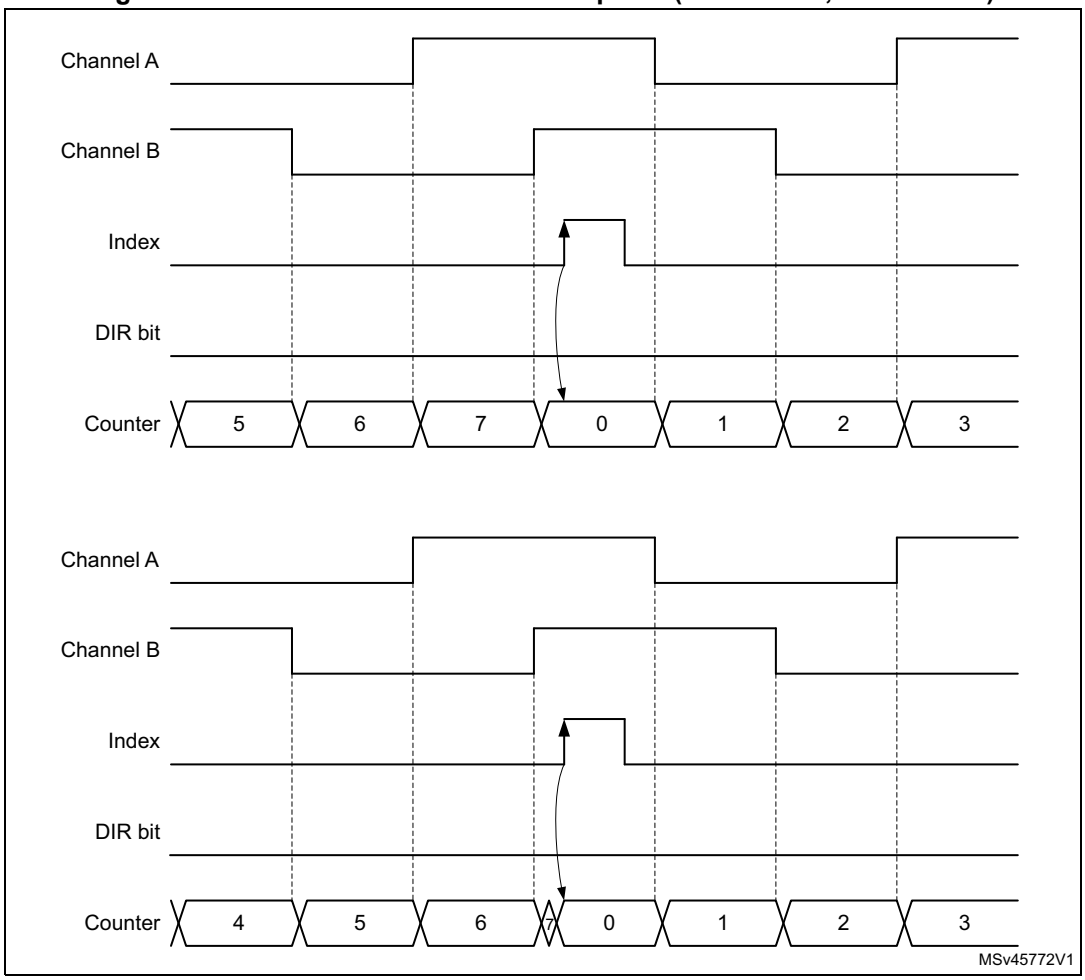
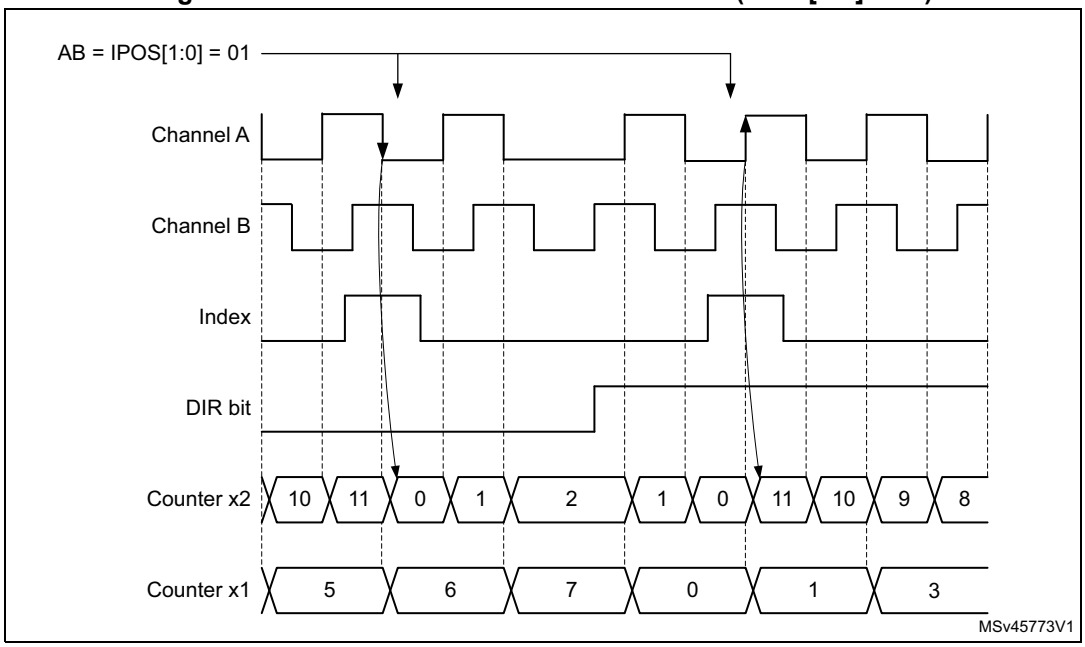


Figure 524. Counter reset Narrow index pulse (closer view, ARR = 0x07)



The *Figure 525* below shows how the index is managed in x1 and x2 modes.

Figure 525. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)



Directional index sensitivity

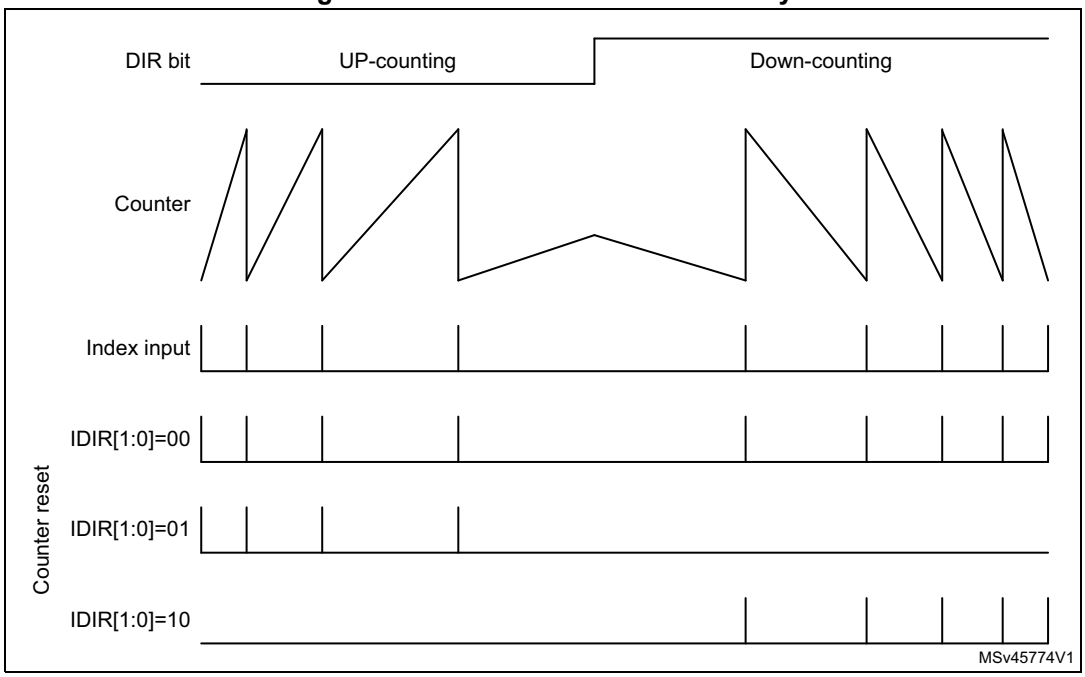
The IDIR[1:0] bitfield in the TIMx_ECR register allows the index to be active only in a selected counting direction.

The *Figure 526* below shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

Note: The IDR[1:0] bitfield must be written when IE bit is reset (index mode disabled).

Note: The directional index sensitivity is not supported in clock + direction mode. When SMS[3:0] = 1010 or 1011, the IDIR[1:0] must be set to 00.

Figure 526. Directional index sensitivity

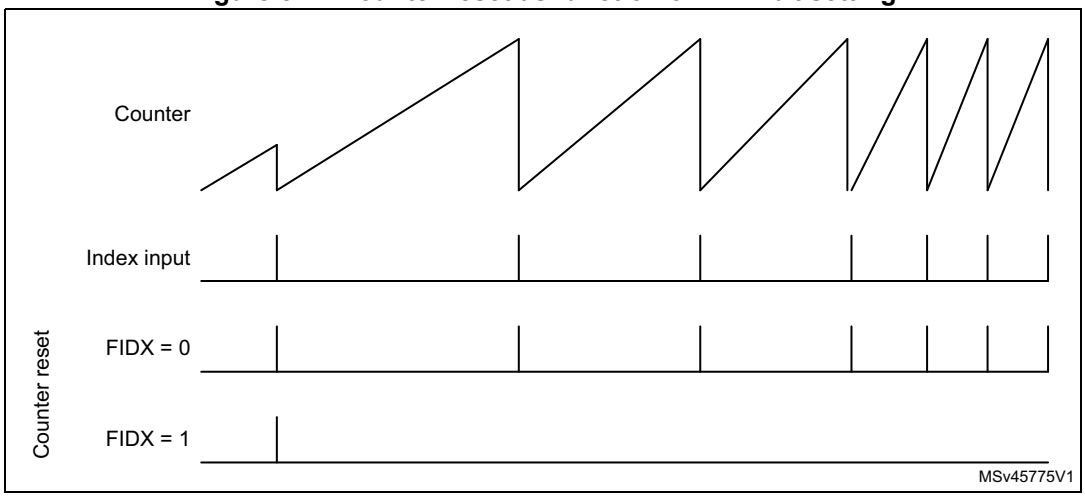


Special first index event management

The FIDX bit in the TIMx_ECR register allows the Index to be taken only once, as shown on the *Figure 527* below. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be re-armed by writing the FIDX bit to 0 and setting it again to 1.

Note: When FIDX = 1, the index can be issued twice (IDXF flag set) if the direction changes at position 0 (index active).

Figure 527. Counter reset as function of FIDX bit setting



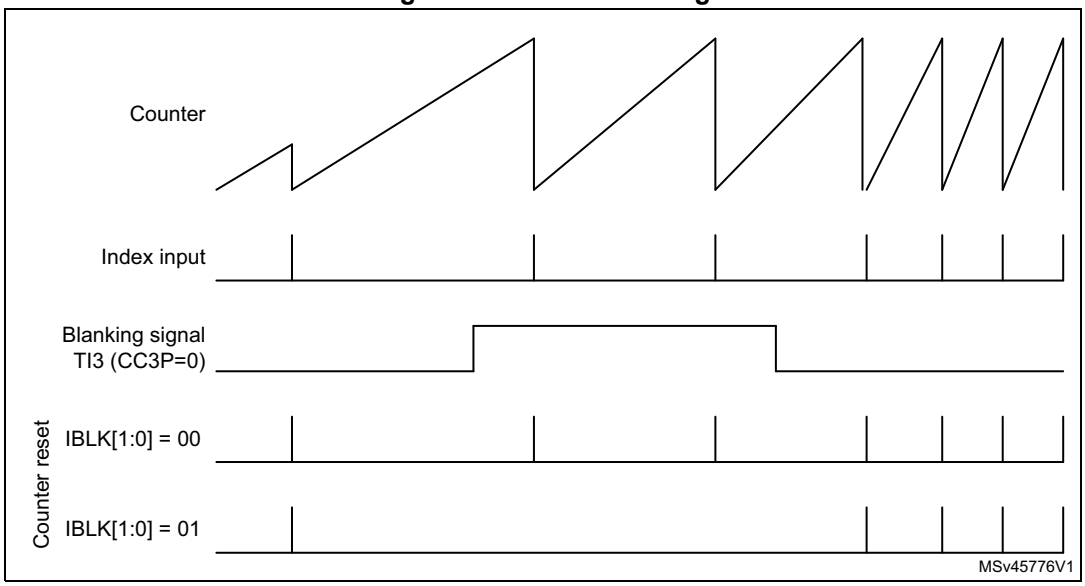
Index blanking

The Index event can be blanked using the tim_ti3 or tim_ti4 inputs. During the blanking window, the index events are no longer resetting the counter, as shown on the *Figure 528* below.

This mode is enabled using the IBLK[1:0] bitfield in the TIMx_ECR register, as following:

- IBLK[1:0] = 00: Index signal always active
- IBLK[1:0] = 01: Index signal blanking on tim_ti3 input
- IBLK[1:0] = 10: Index signal blanking on tim_ti4 input

Figure 528. Index blanking



Index management in non-quadrature mode

The *Figure 529* and *Figure 530* below detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[3:0] bitfield is equal to 1010, 1011, 1100, 1101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as following:

- IPOS[0] = 0: Index is detected on clock low level
- IPOS[0] = 1: Index is detected on clock high level

The IPOS[1] bit is not-significant.

Figure 529. Index behavior in clock + direction mode, IPOS[0] = 1

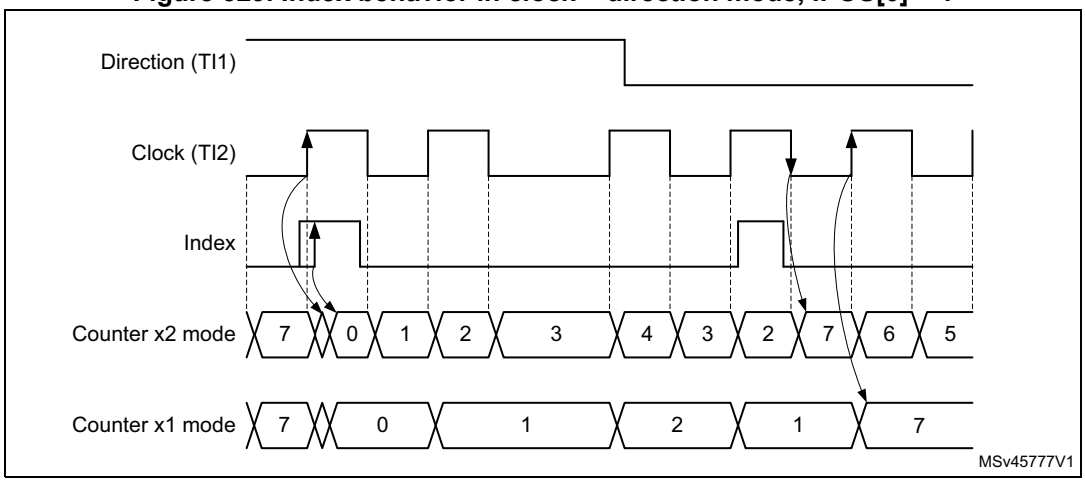
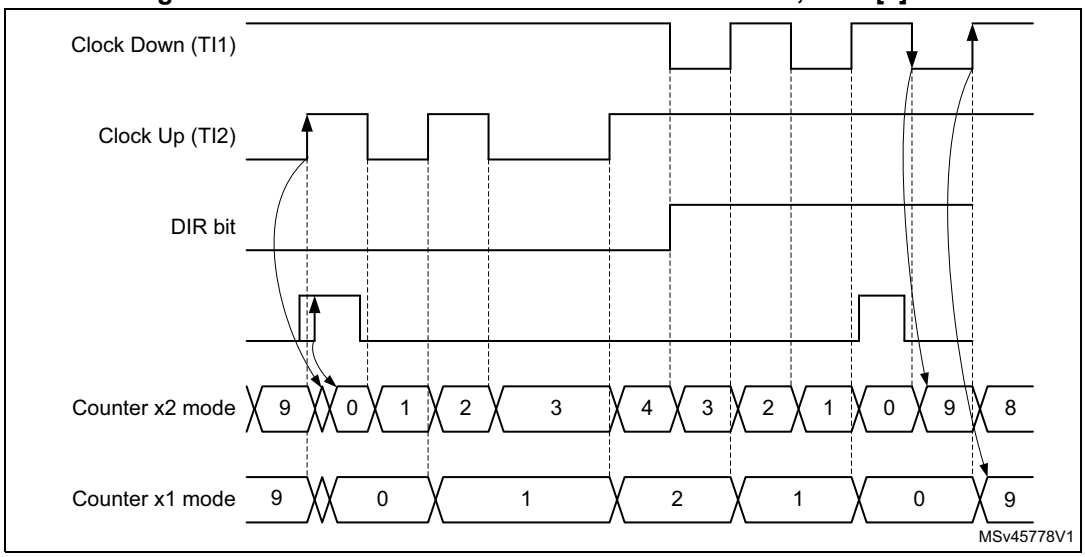


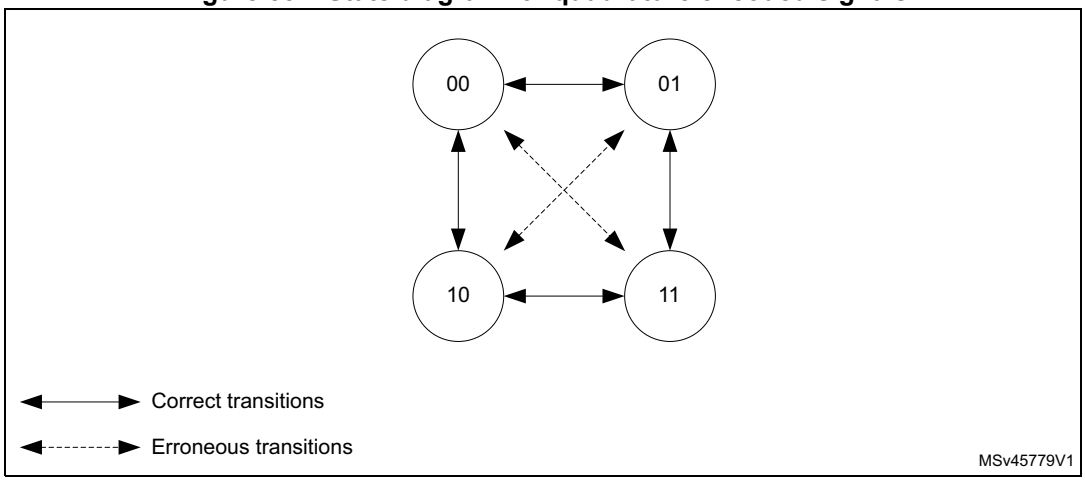
Figure 530. Index behavior in directional clock mode, IPOS[0] = 1



Encoder error management

For encoder configurations where 2 quadrature signals are available, it is possible to detect transition errors. The reading on the 2 inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on the *Figure 531.* below. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIMx_SR status register. A transition error interrupt is generated if the TERRIE bit is set in the TIMx_DIER register.

Figure 531. State diagram for quadrature encoded signals



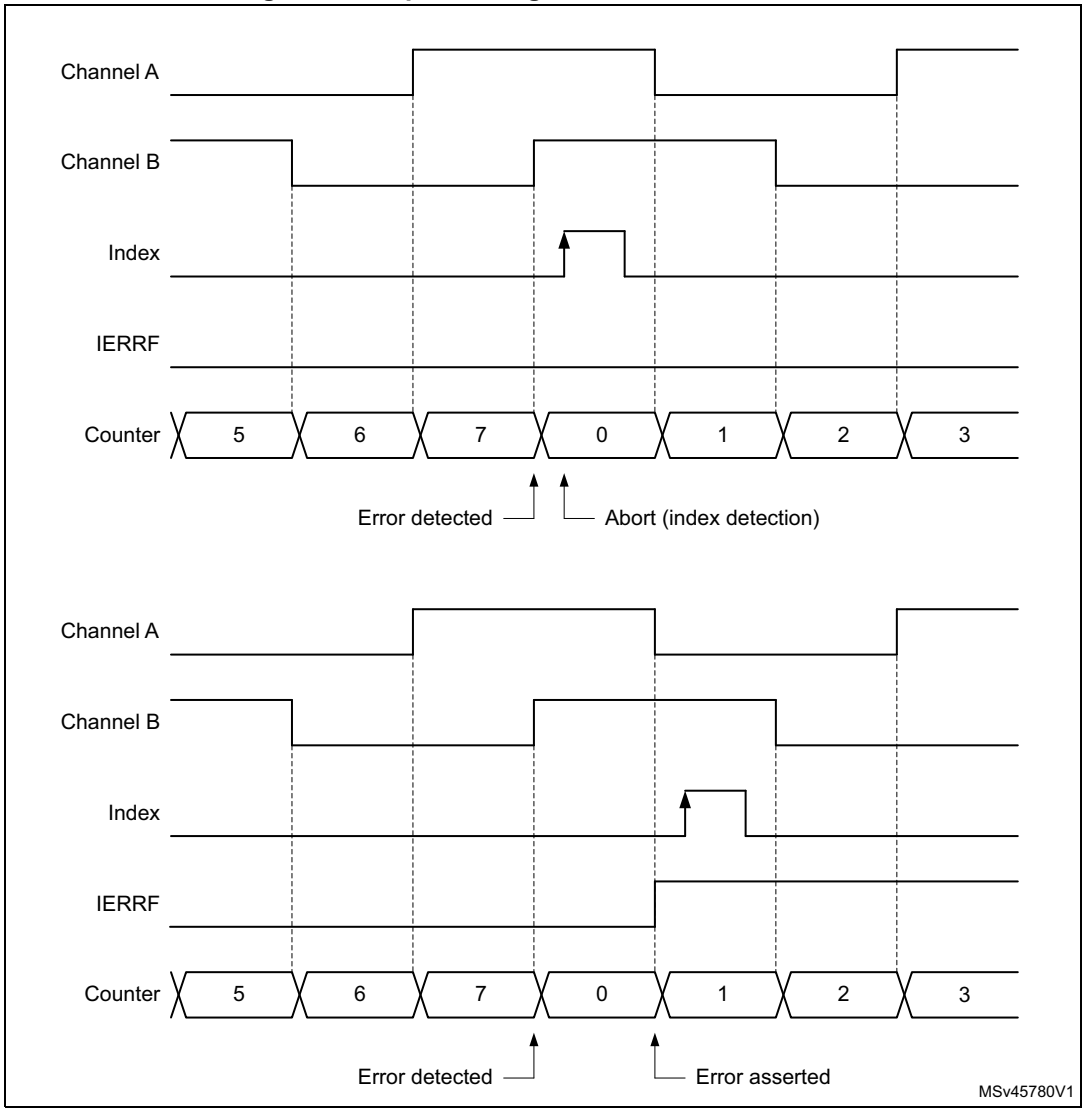
For encoder having an Index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The Index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIMx_ARR to 0 or decremented from 0 to TIMxARR value without any index event, this is reported as an Index position error.

The overflow threshold is programmed using the TIMx_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIMx_ARR = 3999 + 1 = 4000.

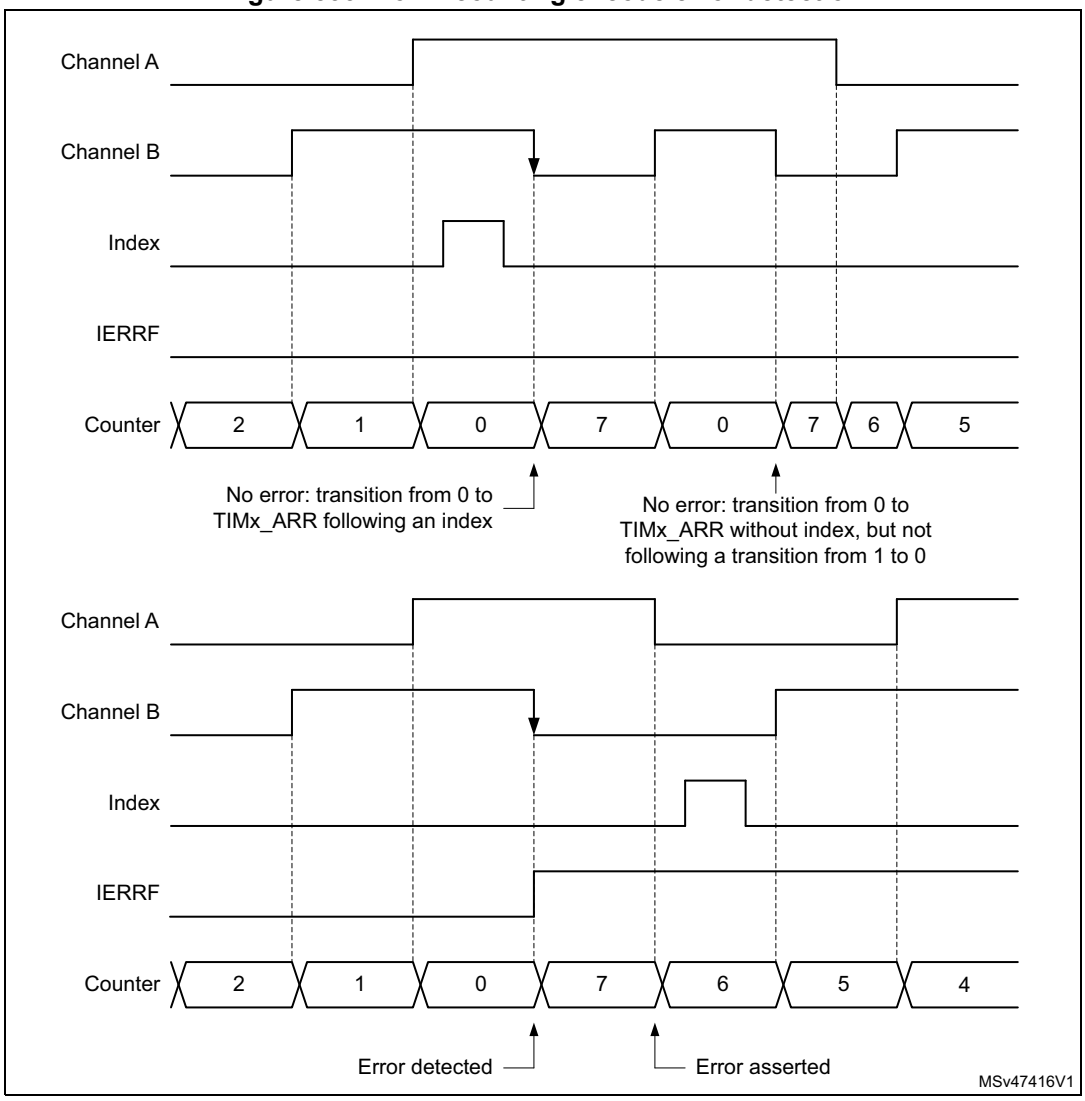
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is cope with narrow index pulses in gated A and B mode, as shown on *Figure 532* below.

Figure 532. Up-counting encoder error detection



In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 533* below, to avoid any false error detection in case the encoder dithers between TIMx_ARR and 0 immediately after the index detection.

Figure 533. Down-counting encode error detection



An index error sets the IERRF interrupt flag in the TIMx_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIMx_DIER register.

Functional encoder Interrupts

The following interrupts are also available in encoder mode

- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIMx_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIMx_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIMx_DIER register.
- Index event: the Index event sets the IDXF interrupt flag in the TIMx_SR status register. An Index interrupt is generated if the IDXIE bit is set in the TIMx_DIER register.

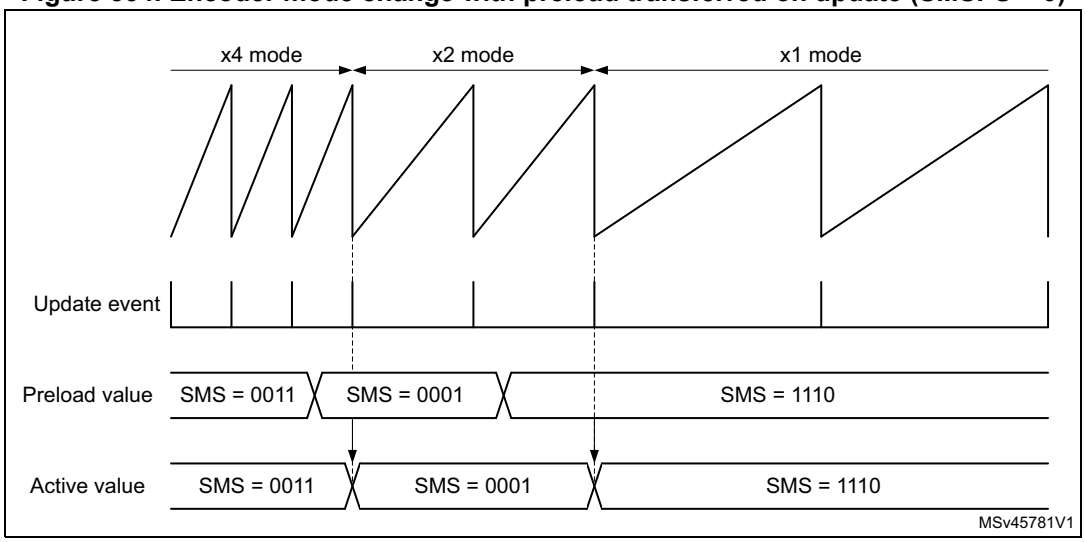
Slave mode selection preload for run-time encoder mode update

It may be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the Update interrupt rate, by switching from x4 to x2 to x1 mode, as show on the *Figure 534* below.

For this purpose, the SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value can be selected with the SMSPS bit in the TIMx_SMCR register.

- SMSPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when upcounting, and underflows when downcounting. This mode must be used only when index is disabled (bit IE = 0).
- SMSPS = 1: the transfer is triggered by the Index event.

Figure 534. Encoder mode change with preload transferred on update (SMSPS = 0)



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIMx_CR2 register.It is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

47.4.19 Direction bit output

Its is possible to output a direction signal out of the timer, on the tim_oc3 and tim_oc4 output signals (copy of the DIR bit in the TIMx_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bit field to 1011 in the TIMx_CCMR2 register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

47.4.20 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register's bit 31 (TIMxCNT[31]). This is used 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).

47.4.21 Timer input XOR function

The TI1S bit in the TIM1xx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins tim_ti1, tim_ti2 and tim_ti3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in *Section 46.3.29: Interfacing with Hall sensors on page 1712*.

47.4.22 Timers and external trigger synchronization

The TIMx Timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

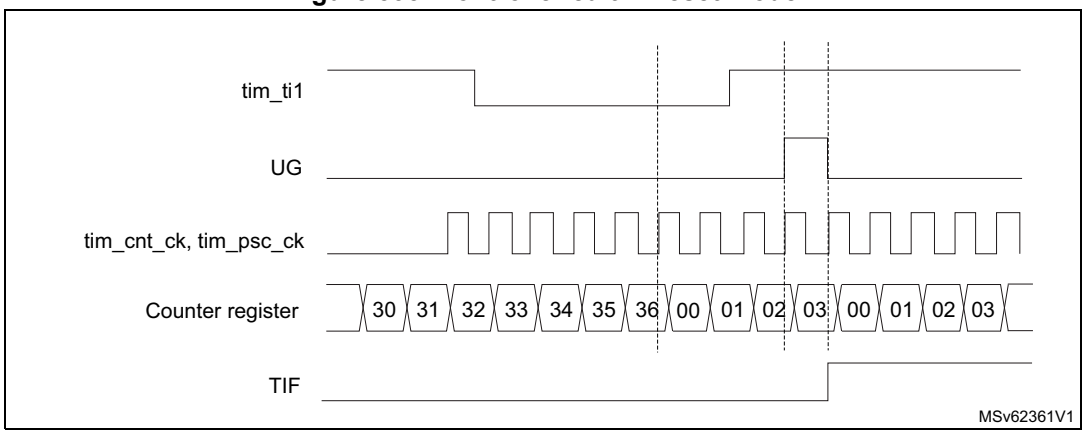
1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the

trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 535. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

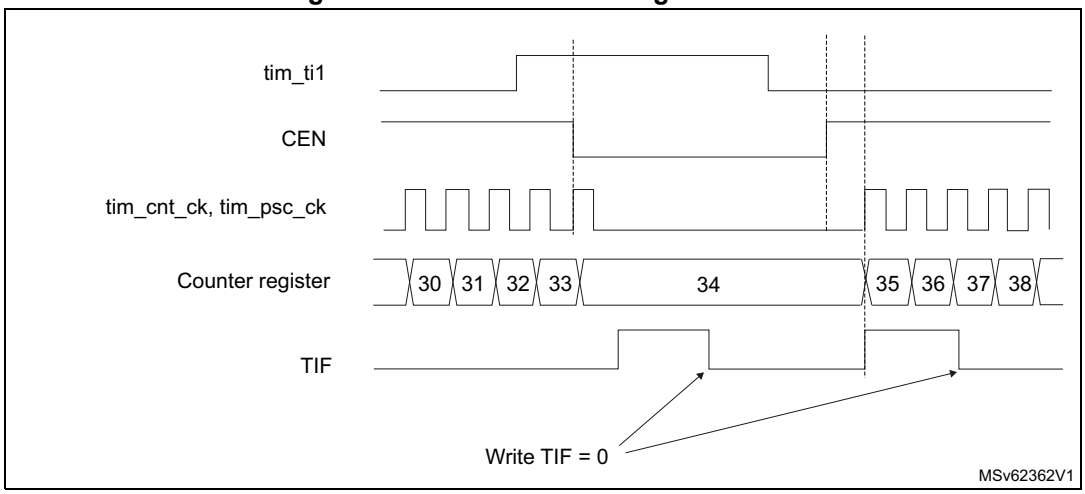
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 536. Control circuit in gated mode



Note: The configuration "CCxP=CCxNP=1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

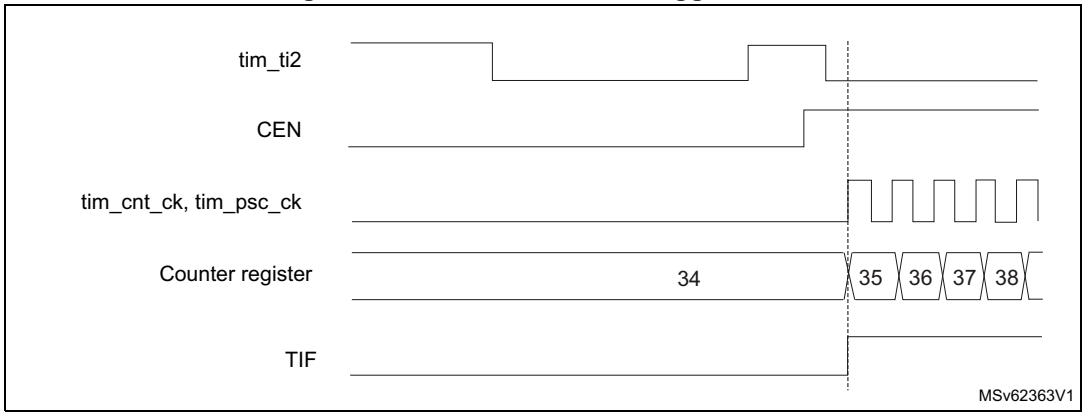
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. CC2S bits are selecting the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select tim_ti2 as the input source by writing TS=00110 in TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 537. Control circuit in trigger mode



Slave mode selection preload for run-time encoder mode update

The SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value is the update event (UEV) occurring when the counter overflows.

Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode – combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

Slave mode – external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the tim_etr_in signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode or trigger mode. It is recommended not to select tim_etr_in as tim_trgi through the TS bits of TIMx_SMCR register.

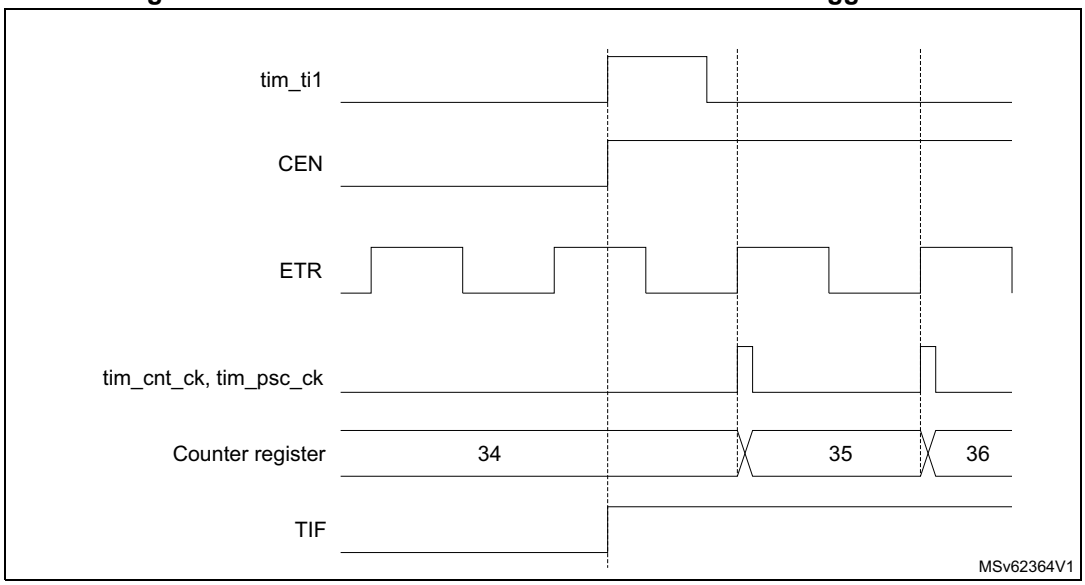
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS=00: prescaler disabled
   - ETP=0: detection of rising edges on tim_etr_in and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F=0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S=01in TIMx_CCMR1 register to select only the input capture source
   - CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.

A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.

Figure 538. Control circuit in external clock mode 2 + trigger mode



47.4.23 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one Timer is configured in Master Mode, it can reset, start, stop or clock the counter of another Timer configured in Slave Mode.

Figure 539 and *Figure 540* show examples of master/slave timer connections.

Figure 539. Master/Slave timer example

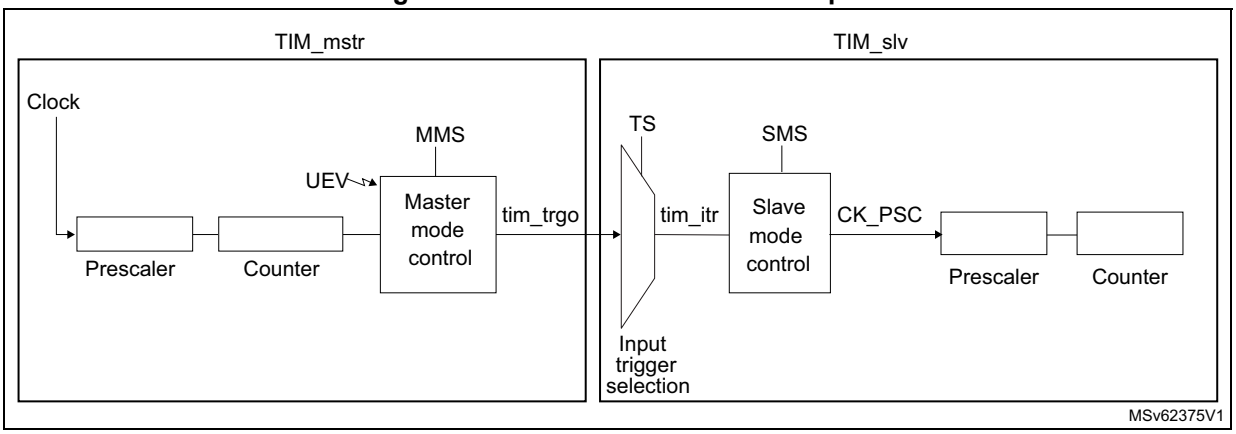
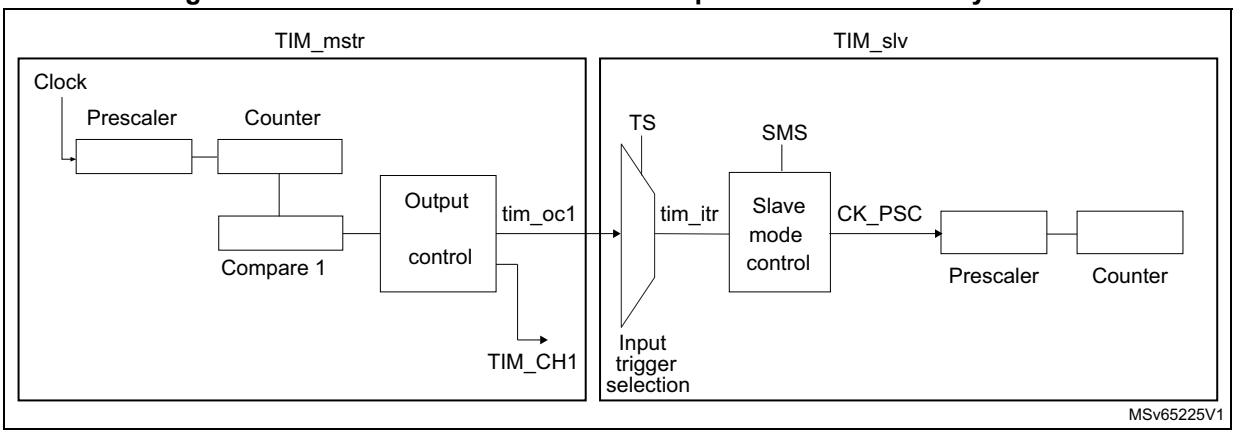


Figure 540. Master/slave connection example with 1 channel only timers



Note: The timers with one channel only (see Figure 540) do not feature a master mode. However, the tim_oc1 output signal can serve as trigger for slave timer (see TIMx internal trigger connection table in Section 47.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals). The tim_oc1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer detects the trigger. For instance, if the destination timer tim_ker_ck clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

Using one timer as prescaler for another timer

For example, TIM_mstr can be configured to act as a prescaler for TIM_slv. Refer to *Figure 539*. To do this:

1. Configure TIM_mstr in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS=010 is written in the TIM_mstr_CR2 register, a rising edge is output on tim_trgo each time an update event is generated.
2. To connect the tim_trgo output of TIM_mstr to TIM_slv, TIM_slv must be configured in slave mode using ITR2 as internal trigger. This is selected through the TS bits in the TIM_slv_SMCR register (writing TS=00010).
3. Then the slave mode controller must be put in external clock mode 1 (write SMS=111 in the TIM_slv_SMCR register). This causes TIM_slv to be clocked by the rising edge of the periodic TIM_mstr trigger signal (which correspond to the TIM_mstr counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: If tim_ocx is selected on TIM_mstr as the trigger output (MMS=1xx), its rising edge is used to clock the counter of TIM_slv.

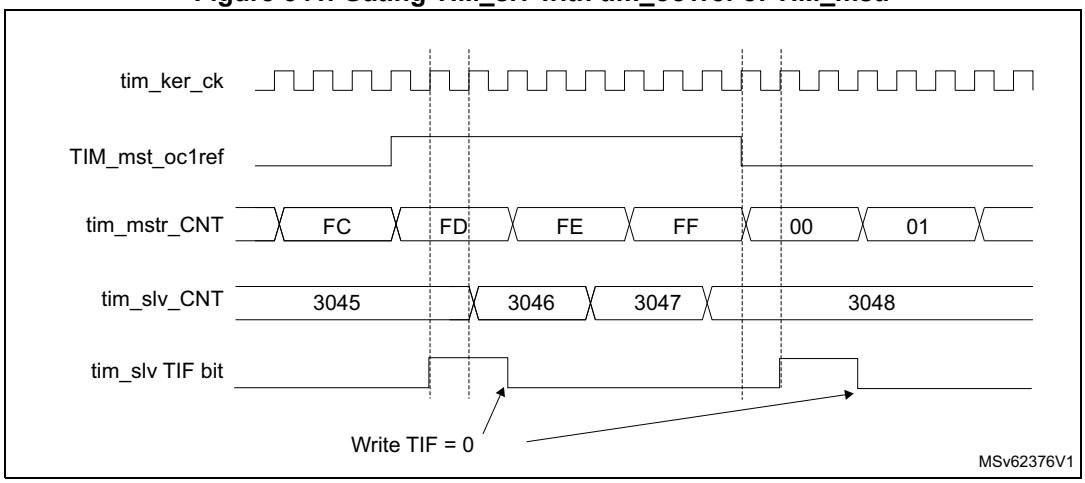
Using one timer to enable another timer

In this example, we control the enable of TIM_slv with the output compare 1 of TIM_mstr. Refer to *Figure 539* for connections. TIM_slv counts on the divided internal clock only when tim_oc1ref of TIM_mstr is high. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck ($f_{tim\_cnt\_ck} = f_{tim\_ker\_ck}/3$).

1. Configure TIM_mstr master mode to send its Output Compare 1 Reference (tim_oc1ref) signal as trigger output (MMS=100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS=00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS=101 in TIM_slv_SMCR register).
5. Enable TIM_slv by writing ‘1 in the CEN bit (TIM_slv_CR1 register).
6. Start TIM_mstr by writing ‘1 in the CEN bit (TIM_mstr_CR1 register).

Note: The slave timer counter clock is not synchronized with the master timer counter clock, this mode only affects the TIM_slv counter enable signal.

Figure 541. Gating TIM_slv with tim_oc1ref of TIM_mstr

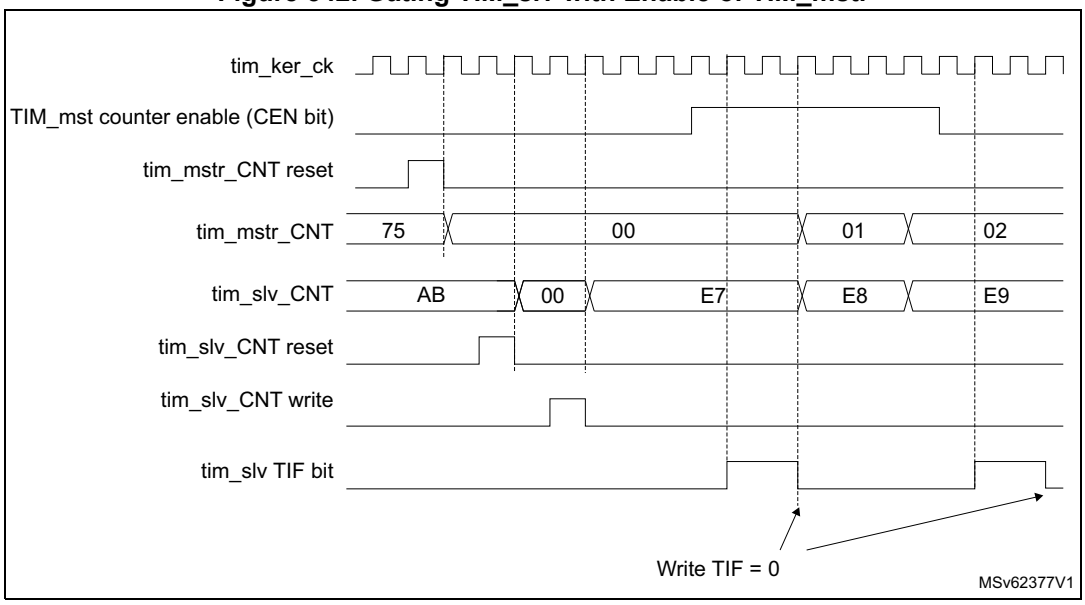


In the example in *Figure 541*, the TIM_slv counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM_mstr. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example (refer to *Figure 542*), we synchronize TIM_mstr and TIM_slv. TIM_mstr is the master and starts from 0. TIM_slv is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM_slv stops when TIM_mstr is disabled by writing ‘0 to the CEN bit in the TIM_mstr_CR1 register:

1. Configure TIM_mstr master mode to send its Output Compare 1 Reference (tim_oc1ref) signal as trigger output (MMS=100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS=00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS=101 in TIM_slv_SMCR register).
5. Reset TIM_mstr by writing ‘1 in UG bit (TIM_mstr_EGR register).
6. Reset TIM_slv by writing ‘1 in UG bit (TIM_slv_EGR register).
7. Initialize TIM_slv to 0xE7 by writing ‘0xE7’ in the TIM_slv counter (TIM_slv_CNT).
8. Enable TIM_slv by writing ‘1 in the CEN bit (TIM_slv_CR1 register).
9. Start TIM_mstr by writing ‘1 in the CEN bit (TIM_mstr_CR1 register).
10. Stop TIM_mstr by writing ‘0 in the CEN bit (TIM_mstr_CR1 register).

Figure 542. Gating TIM_slv with Enable of TIM_mstr

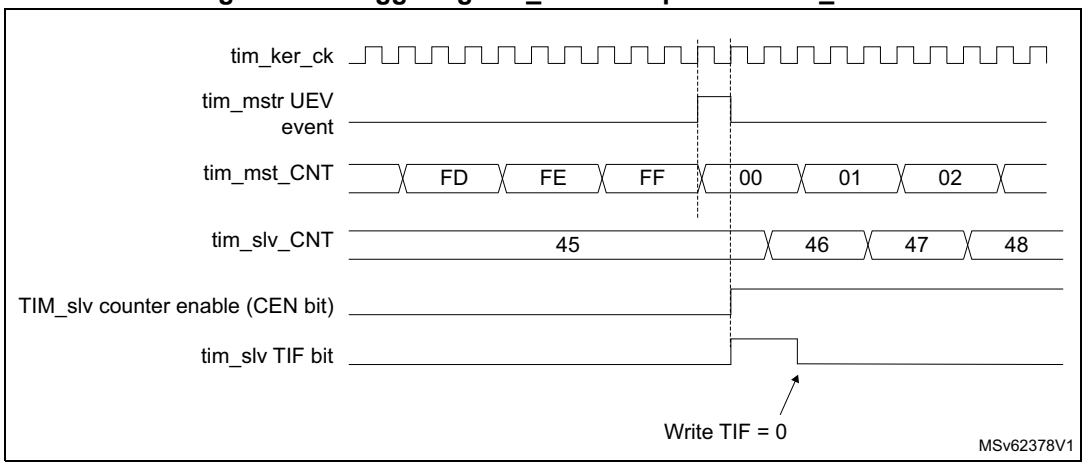


Using one timer to start another timer

In this example, we set the enable of TIM_slv with the update event of TIM_mstr. Refer to *Figure 539* for connections. TIM_slv starts counting from its current value (which can be non-zero) on the divided internal clock as soon as the update event is generated by TIM_mstr. When TIM_slv receives the trigger signal its CEN bit is automatically set and the counter counts until we write ‘0 to the CEN bit in the TIM_slv_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck ($f_{tim\_cnt\_ck} = f_{tim\_ker\_ck}/3$).

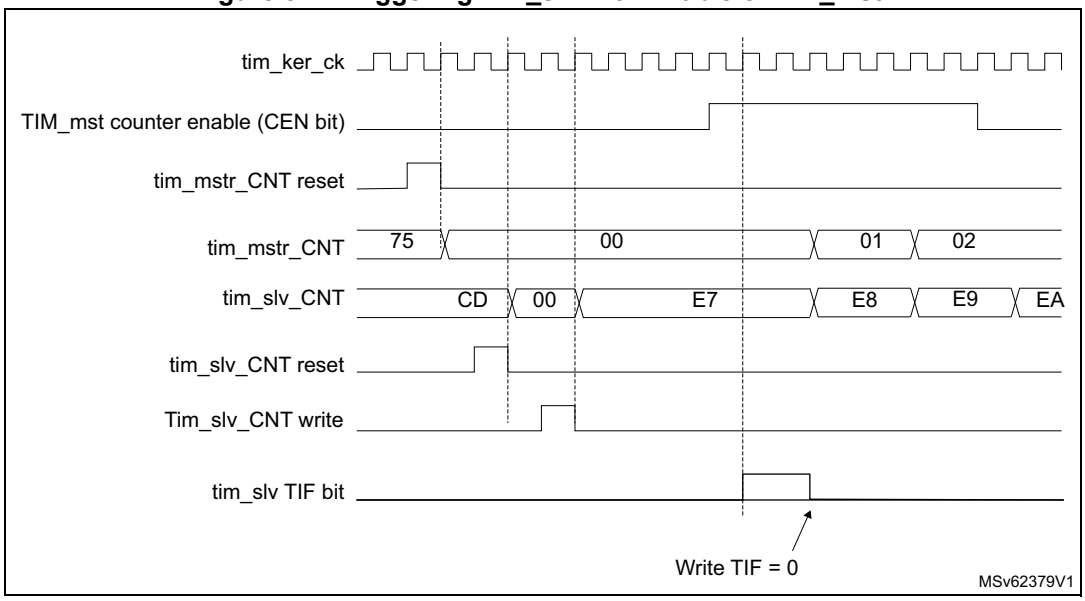
1. Configure TIM_mstr master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr period (TIM_mstr_ARR registers).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS=00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in trigger mode (SMS=110 in TIM_slv_SMCR register).
5. Start TIM_mstr by writing ‘1 in the CEN bit (TIM_mstr_CR1 register).

Figure 543. Triggering TIM_slv with update of TIM_mstr



As in the previous example, both counters can be initialized before starting counting. *Figure 544* shows the behavior with the same configuration as in *Figure 543* but in trigger mode instead of gated mode (SMS=110 in the TIM_slv_SMCR register).

Figure 544. Triggering TIM_slv with Enable of TIM_mstr



Starting 2 timers synchronously in response to an external trigger

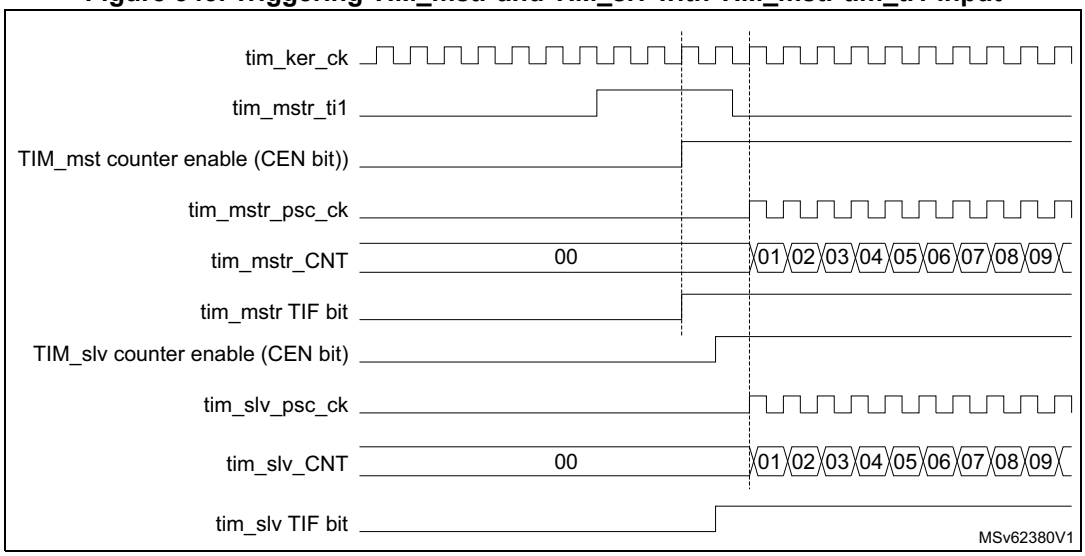
In this example, we set the enable of TIM_mstr when its tim_ti1 input rises, and the enable of TIM_slv with the enable of TIM_mstr. Refer to *Figure 539* for connections. To ensure the counters are aligned, TIM_mstr must be configured in Master/Slave mode (slave with respect to tim_ti1, master with respect to TIM_slv):

1. Configure TIM_mstr master mode to send its Enable as trigger output (MMS=001 in the TIM_mstr_CR2 register).
2. Configure TIM_mstr slave mode to get the input trigger from tim_ti1 (TS=00100 in the TIM_mstr_SMCR register).
3. Configure TIM_mstr in trigger mode (SMS=110 in the TIM_mstr_SMCR register).
4. Configure the TIM_mstr in Master/Slave mode by writing MSM=1 (TIM_mstr_SMCR register).
5. Configure TIM_slv to get the input trigger from TIM_mstr (TS=00000 in the TIM_slv_SMCR register).
6. Configure TIM_slv in trigger mode (SMS=110 in the TIM_slv_SMCR register).

When a rising edge occurs on tim_ti1 (TIM_mstr), both counters starts counting synchronously on the internal clock and both TIF flags are set.

Note: In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but an offset can easily be inserted between them by writing any of the counter registers (TIMx_CNT). One can see that the master/slave mode insert a delay between CNT_EN and CK_PSC on TIM_mstr.

Figure 545. Triggering TIM_mstr and TIM_slv with TIM_mstr tim_ti1 input



Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

47.4.24 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

47.4.25 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

The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 47.5.23: TIMx DMA control register (TIMx_DCR)(x = 2 to 5)* for details).

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 and DBSS = 1.
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.

47.4.26 TIM2/TIM3/TIM4/TIM5 DMA requests

The TIM2/TIM3/TIM4/TIM5 can generate a DMA requests, as shown in *Table 461*.

Table 461. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_cc2_dma	Capture/compare 2	CC2DE
tim_cc3_dma	Capture/compare 3	CC3DE
tim_cc4_dma	Capture/compare 4	CC4DE
tim_trg_dma	Trigger	TDE

Note: Some timer's DMA requests may not be connected to the DMA controller. Refer to the DMA section(s) for more details.

47.4.27 Debug mode

When the microcontroller enters debug mode (Cortex®-M33 core halted), the TIMx counter can either continues to work normally or stops.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

47.4.28 TIM2/TIM3/TIM4/TIM5 low-power modes

Table 462. Effect of low-power modes on TIM2/TIM3/TIM4/TIM5

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

47.4.29 TIM2/TIM3/TIM4/TIM5 interrupts

The TIM2/TIM3/TIM4/TIM5 can generate multiple interrupts, as shown in *Table 463*.

Table 463. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM_UP	Update	UIF	UIE	write 0 in UIF	Yes	No
TIM_CC	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
	Capture/compare 2	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Capture/compare 3	CC3IF	CC3IE	write 0 in CC3IF	Yes	No
	Capture/compare 4	CC4IF	CC4IE	write 0 in CC4IF	Yes	No
TIM_TRG	Trigger	TIF	TIE	write 0 in TIF	Yes	No
TIM_DIR_IDX	Index	IDXF	IDXIE	write 0 in IDXF	Yes	No
	Direction	DIRF	DIRIE	write 0 in DIRF	Yes	No
TIM_IERR	Index Error	IERRF	IERRIE	write 0 in IERRF	Yes	No
TIM_TER	Transition Error	TERRF	TERRIE	write 0 in TERRF	Yes	No

47.5 TIM2/TIM3/TIM4/TIM5 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

47.5.1 TIMx control register 1 (TIMx_CR1)(x = 2 to 5)

Address offset: 0x000

Reset value: 0x0000

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

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering Enable

0: Dithering disabled

1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.

1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bit-field indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_etr_in, tim_tix),

00: $t_{DTS} = t_{tim\_ker\_ck}$

01: $t_{DTS} = 2 \times t_{tim\_ker\_ck}$

10: $t_{DTS} = 4 \times t_{tim\_ker\_ck}$

11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered

1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).

01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.

10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.

11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 **DIR**: Direction

0: Counter used as upcounter

1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

47.5.2 TIMx control register 2 (TIMx_CR2)(x = 2 to 5)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
								rw	rw	rw	rw	rw			

Bits 31:26 Reserved, must be kept at reset value.

Bits 24:8 Reserved, must be kept at reset value.

Bit 7 **TI1S**: tim_ti1 selection

0: The tim_ti1_in[15:0] multiplexer output is to tim_ti1 input
1: The tim_ti1_in[15:0], tim_ti2_in[15:0] and tim_ti3_in[15:0] multiplexers outputs are XORed and connected to the tim_ti1 input. See also *Section 46.3.29: Interfacing with Hall sensors on page 1712*.

Bits 25, 6, 5, 4 **MMS[3:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter enable signal, CNT_EN, is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode.
When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

0011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)

1000: **Encoder Clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

Others: Reserved

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bits 2:0 Reserved, must be kept at reset value.

47.5.3 TIMx slave mode control register (TIMx_SMCR)(x = 2 to 5)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
						rw	rw			rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **SMSPS**: SMS preload source

This bit selects whether the events that triggers the SMS[3:0] bitfield transfer from preload to active

0: The transfer is triggered by the Timer's Update event

1: The transfer is triggered by the Index event

Bit 24 **SMSPE**: SMS preload enable

This bit selects whether the SMS[3:0] bitfield is preloaded

0: SMS[3:0] bitfield is not preloaded

1: SMS[3:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

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

Bit 15 **ETP**: External trigger polarity

This bit selects whether tim_etr_in or $\overline{\text{tim\_etr\_in}}$ is used for trigger operations

0: tim_etr_in is non-inverted, active at high level or rising edge

1: tim_etr_in is inverted, active at low level or falling edge

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the tim_etrf signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_etrf (SMS=111 and TS=00111).
It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tim_trgi must not be connected to tim_etrf in this case (TS bits must not be 00111).
If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is tim_etrf.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal tim_etrp frequency must be at most 1/4 of tim_ker_ck frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etr_in.

00: Prescaler OFF

01: tim_etrp frequency divided by 2

10: tim_etrp frequency divided by 4

11: tim_etrp frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample tim_etrp signal and the length of the digital filter applied to tim_etrp. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bit 7 **MSM**: Master/Slave mode

0: No action
1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]**: Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.

00000: Internal trigger 0 (tim_itr0)
00001: Internal trigger 1 (tim_itr1)
00010: Internal trigger 2 (tim_itr2)
00011: Internal trigger 3 (tim_itr3)
00100: tim_ti1 edge detector (tim_ti1f_ed)
00101: Filtered timer input 1 (tim_ti1fp1)
00110: Filtered timer input 2 (tim_ti2fp2)
00111: External trigger input (tim_etrf)
01000: Internal trigger 4 (tim_itr4)
01001: Internal trigger 5 (tim_itr5)
01010: Internal trigger 6 (tim_itr6)
01011: Internal trigger 7 (tim_itr7)
01100: Internal trigger 8 (tim_itr8)
01101: Internal trigger 9 (tim_itr9)
01110: Internal trigger 10 (tim_itr10)
01111: Internal trigger 11 (tim_itr11)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
Others: Reserved

See *Section 47.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation details.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS**: OCREF clear selection

This bit is used to select the OCREF clear source
0: tim_ocref_clr_int is connected to the tim_ocref_clr input
1: tim_ocref_clr_int is connected to tim_etrf

Note: If the OCREF clear selection feature is not supported, this bit is reserved and forced by hardware to '0'. Section 47.3: TIM2/TIM3/TIM4/TIM5 implementation.

Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

0000: Slave mode disabled - if CEN = ‘1 then the prescaler is clocked directly by the internal clock.

0001: Encoder mode 1 - Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

0010: Encoder mode 2 - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

0011: Encoder mode 3 - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

0100: Reset Mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

1010: Encoder mode: Clock plus direction, x2 mode.

1011: Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P.

1100: Encoder mode: Directional Clock, x2 mode.

1101: Encoder mode: Directional Clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

1110: Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

1111: Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS=00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

47.5.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 5)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

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

Bit 23 **TERRIE**: Transition error interrupt enable
0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 **IERRIE**: Index error interrupt enable
0: Index error interrupt disabled
1: Index error interrupt enabled

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction change interrupt disabled
1: Direction change interrupt enabled

Bit 20 **IDXIE**: Index interrupt enable
0: Index interrupt disabled
1: Index interrupt enabled

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled.
1: Trigger DMA request enabled.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable
0: CC4 DMA request disabled.
1: CC4 DMA request enabled.

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable
0: CC3 DMA request disabled.
1: CC3 DMA request enabled.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled.
1: CC2 DMA request enabled.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled.
1: CC1 DMA request enabled.

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled.
1: Trigger interrupt enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable
0: CC4 interrupt disabled.
1: CC4 interrupt enabled.

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable
0: CC3 interrupt disabled.
1: CC3 interrupt enabled.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled.
1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

47.5.5 TIMx status register (TIMx_SR)(x = 2 to 5)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	Res.	Res.
								rc_w0	rc_w0	rc_w0	rc_w0				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

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

Bit 23 **TERRF**: Transition error interrupt flag
This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to ‘0’.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 **IERRF**: Index error interrupt flag
This flag is set by hardware when an index error is detected. It is cleared by software by writing it to ‘0’.
0: No index error has been detected.
1: An index error has been detected

Bit 21 **DIRF**: Direction change interrupt flag
This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIMx_CR is changing). It is cleared by software by writing it to ‘0’.
0: No direction change
1: Direction change

Bit 20 **IDXF**: Index interrupt flag
This flag is set by hardware when an index event is detected. It is cleared by software by writing it to ‘0’.
0: No index event occurred.
1: An index event has occurred

Bits 19:13 Reserved, must be kept at reset value.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag
refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag
refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag
Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag
Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in downcounting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
At overflow or underflow and if UDIS=0 in the TIMx_CR1 register.
When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.
When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS=0 and UDIS=0 in the TIMx_CR1 register.

47.5.6 TIMx event generation register (TIMx_EGR)(x = 2 to 5)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

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

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4G**: Capture/compare 4 generation
Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation
Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

47.5.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2 to 5)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2.

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1.

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}

0001: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=2

0010: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=4

0011: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E=0 (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

47.5.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2 to 5)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode
refer to OC1M description on bits 6:4

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2
10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1
11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)
Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable
0: tim_oc1ref is not affected by the tim_ocref_clr_int input
1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas tim_oc1 active level depends on CC1P bit.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (tim_oc1ref=0) as long as TIMx_CNT>TIMx_CCR1 else active (tim_oc1ref=1).

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved.

1011: Reserved.

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Asymmetric PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

Note: In PWM mode, the tim_ocref_clr level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output.
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2.
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

47.5.9 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2 to 5)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

47.5.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2 to 5)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode
Refer to OC1M description (bits 6:4 in TIMx_CCMR1 register)

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bit 7 **OC3CE**: Output compare 3 clear enable

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode
Refer to OC1M description (bits 6:4 in TIMx_CCMR1 register)

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

47.5.11 TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 5)

Address offset: 0x020

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

Bit 15 **CC4NP**: *Capture/Compare 4 output Polarity.*
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: *Capture/Compare 4 output Polarity.*
Refer to CC1P description

Bit 12 **CC4E**: *Capture/Compare 4 output enable.*
refer to CC1E description

Bit 11 **CC3NP**: *Capture/Compare 3 output Polarity.*
Refer to CC1NP description

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC3P**: *Capture/Compare 3 output Polarity.*
Refer to CC1P description

Bit 8 **CC3E**: *Capture/Compare 3 output enable.*
Refer to CC1E description

Bit 7 **CC2NP**: *Capture/Compare 2 output Polarity.*
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: *Capture/Compare 2 output Polarity.*
refer to CC1P description

Bit 4 **CC2E**: *Capture/Compare 2 output enable.*
Refer to CC1E description

Bit 3 **CC1NP**: *Capture/Compare 1 output Polarity.*
CC1 channel configured as output: CC1NP must be kept cleared in this case.
CC1 channel configured as input: This bit is used in conjunction with CC1P to define tim_ti1fp1/tim_ti2fp1 polarity. refer to CC1P description.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: *Capture/Compare 1 output Polarity.*
0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)
When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.
CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).
CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).
CC1NP=1, CC1P=1: non-inverted/both edges. The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.
CC1NP=1, CC1P=0: this configuration is reserved, it must not be used.

Bit 0 **CC1E**: *Capture/Compare 1 output enable.*
0: Capture mode disabled / OC1 is not active
1: Capture mode enabled / OC1 signal is output on the corresponding output pin

Table 464. Output control bit for standard tim_ocx channels

CCxE bit	tim_ocx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

Note: The state of the external IO pins connected to the standard tim_ocx channels depends only on the GPIO registers when CCxE=0.

47.5.12 TIMx counter (TIMx_CNT)(x = 2 to 5)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIFCPY_CNT[31]	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_CNT[31]**: Value depends on IUFREMAP in TIMx_CR1.
If UIFREMAP = 0
CNT[31]: Most significant bit of counter value
If UIFREMAP = 1
UIFCPY: UIF Copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register

Bits 30:0 **CNT[30:0]**: Least significant part of counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register holds the non-dithered part in CNT[30:0]. The fractional part is not available.

47.5.13 TIMx prescaler (TIMx_PSC)(x = 2 to 5)

Address offset: 0x028

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency tim_cnt_ck is equal to $f_{tim\_psc\_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

47.5.14 TIMx auto-reload register (TIMx_ARR)(x = 2 to 5)

Address offset: 0x02C

Reset value: 0xFFFF FFFF

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

Bits 31:0 **ARR[31:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 47.4.3: Time-base unit on page 1781* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[31:4]. The ARR[3:0] bitfield contains the dithered part.

47.5.15 TIMx capture/compare register 1 (TIMx_CCR1)(x = 2 to 5)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR1[31:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[31:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[31:0]. The CCR1[3:0] bits are reset.

47.5.16 TIMx capture/compare register 2 (TIMx_CCR2)(x = 2 to 5)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR2[31:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[31:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[31:0]. The CCR2[3:0] bits are reset.

47.5.17 TIMx capture/compare register 3 (TIMx_CCR3)(x = 2 to 5)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR3[31:0]**: Capture/compare 3 value

If channel CC3 is configured as output:

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc3 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR3[31:4]. The CCR3[3:0] bitfield contains the dithered part.

If channel CC3 is configured as input:

CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIMx_CCR3 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR3[31:0]. The CCR3[3:0] bits are reset.

47.5.18 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2 to 5)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR4[31:0]**: Capture/compare 4 value

If channel CC4 is configured as output:

CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[31:4]. The CCR4[3:0] bitfield contains the dithered part.

If channel CC4 is configured as input:

CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIMx_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[31:0]. The CCR4[3:0] bits are reset.

47.5.19 TIMx timer encoder control register (TIMx_ECR)(x = 2 to 5)

Address offset: 0x058

Reset value: 0x0000 0000

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

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **PWPRSC[2:0]**: Pulse width prescaler

This bitfield sets the clock prescaler for the pulse generator, as following:

$t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim\_ker\_ck}$

Bits 23:16 **PW[7:0]**: Pulse width

This bitfield defines the pulse duration, as following:

$t_{PW} = PW[7:0] \times t_{PWG}$

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

Bits 7:6 **IPOS[1:0]**: Index positioning

In quadrature encoder mode (SMS[3:0] = 0001, 0010, 0011, 1110, 1111), this bit indicates in which AB input configuration the Index event resets the counter.

00: Index resets the counter when AB = 00
01: Index resets the counter when AB = 01
10: Index resets the counter when AB = 10
11: Index resets the counter when AB = 11

In directional clock mode or clock plus direction mode (SMS[3:0] = 1010, 1011, 1100, 1101), these bits indicates on which level the Index event resets the counter. In bidirectional clock mode, this applies for both clock inputs.

x0: Index resets the counter when clock is 0
x1: Index resets the counter when clock is 1

Note: IPOS[1] bit is not significant

Bit 5 **FIDX**: First index

This bit indicates if the first index only is taken into account

0: Index is always active
1: the first Index only resets the counter

Bits 4:3 **IBLK[1:0]**: Index blanking

This bit indicates if the Index event is conditioned by the tim_ti3 input

00: Index always active
01: Index disabled hen tim_ti3 input is active, as per CC3P bitfield
10: Index disabled when tim_ti4 input is active, as per CC4P bitfield
11: Reserved

Bits 2:1 **IDIR[1:0]**: Index direction

This bit indicates in which direction the Index event resets the counter.

00: Index resets the counter whatever the direction
01: Index resets the counter when up-counting only
10: Index resets the counter when down-counting only
11: Reserved

Note: The IDR[1:0] bitfield must be written when IE bit is reset (index disabled).

Bit 0 **IE**: Index enable

This bit indicates if the Index event resets the counter.

0: Index disabled
1: Index enabled

47.5.20 TIMx timer input selection register (TIMx_TISEL)(x = 2 to 5)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

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

Bits 27:24 **TI4SEL[3:0]**: Selects tim_ti4[15:0] input
0000: tim_ti4_in0: TIMx_CH4
0001: tim_ti4_in1
...
1111: tim_ti4_in15
Refer to *Section 47.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: Selects tim_ti3[15:0] input
0000: tim_ti3_in0: TIMx_CH3
0001: tim_ti3_in1
...
1111: tim_ti3_in15
Refer to *Section 47.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: Selects tim_ti2[15:0] input
0000: tim_ti2_in0: TIMx_CH2
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 47.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

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

Bits 3:0 **TI1SEL[3:0]**: Selects tim_ti1[15:0] input
0000: tim_ti1_in0: TIMx_CH1
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 47.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

47.5.21 TIMx alternate function register 1 (TIMx_AF1)(x = 2 to 5)

Address offset: 0x060

Reset value: 0x0000 0000

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

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: etr_in source selection
These bits select the etr_in input source.
0000: tim_etr0: TIMx_ETR input
0001: tim_etr1
...
1111: tim_etr15
Refer to *Section 47.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 13:0 Reserved, must be kept at reset value.

47.5.22 TIMx alternate function register 2 (TIMx_AF2)(x = 2 to 5)

Address offset: 0x064

Reset value: 0x0000 0000

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

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

Bits 18:16 **OCRSEL[2:0]**: ocref_clr source selection
These bits select the ocref_clr input source.
000: tim_ocref_clr0
001: tim_ocref_clr1
...
111: tim_ocref_clr7
Refer to *Section 47.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 15:0 Reserved, must be kept at reset value.

47.5.23 TIMx DMA control register (TIMx_DCR)(x = 2 to 5)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[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.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

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

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

0000: Reserved
0001: Update
0010: CC1
0011: CC2
0100: CC3
0101: CC4
0110: COM
0111: Trigger
Others: reserved

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
...
11010: 26 transfers

Example: Let us consider the following transfer: DBL = 7 bytes & DBA = TIM2_CR1.

–If DBL = 7 bytes and DBA = TIM2_CR1 represents the address of the byte to be transferred, the address of the transfer should be given by the following equation:

(TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data are copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

–If the DMA Data Size is configured in half-words, 16-bit data are transferred to each of the 7 registers.

–If the DMA Data Size is configured in bytes, the data are also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

47.5.24 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 5)

Address offset: 0x3E0

Reset value: 0x0000 0000

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

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

47.5.25 TIMx register map

TIMx registers are mapped as described in the table below.

Table 465. TIM2/TIM3/TIM4/TIM5 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																				0	0		0	0	0	0	0	0	0	0	0	0
0x004	**TIMx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
	Reset value							0																		0	0	0	0	0			
0x008	**TIMx_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value							0	0			0	0				0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x00C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.	Res.	Res.	TDE	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	0	0	0	0
0x010	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value								0	0	0	0									0	0	0	0			0		0	0	0	0	0
0x014	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																										0		0	0	0	0	0
0x018	**TIMx_CCMR1** Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	**TIMx_CCMR2** Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	**TIMx_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																	0		0	0	0		0	0	0		0	0	0		0	0

Table 465. TIM2/TIM3/TIM4/TIM5 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x024	**TIMx_CNT**	UIFCPY_CNT[31]	CNT[30:16] (CNT[31:16] on 32-bit timers only)															CNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	**TIMx_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	**TIMx_ARR** (x = 2 to 5)	ARR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x030	Reserved	Reserved																															
0x034	**TIMx_CCR1**	CCR1[31:20] (32-bit timers only)												CCR1[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	**TIMx_CCR2**	CCR2[31:20] (32-bit timers only)												CCR2[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	**TIMx_CCR3**	CCR3[31:20] (32-bit timers only)												CCR3[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	**TIMx_CCR4**	CCR4[31:20] (32-bit timers only)												CCR4[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044.. 0x054	Reserved	Res.																															
0x058	**TIMx_ECR**	Res.	Res.	Res.	Res.	Res.	PWPRSC [2:0]			PW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS [1:0]		FIDX	IBLK [1:0]		IDIR [1:0]		IE
	Reset value						0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x05C	**TIMx_TISEL**	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x060	**TIMx_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [3:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0	0	0														
0x064	**TIMx_AF2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0																
0x068.. 0x3D8	Reserved	Res.																															
0x3DC	**TIMx_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	**TIMx_DMAR**	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3 on page 115* for the register boundary addresses.

48 General purpose timers (TIM15/TIM16/TIM17)

48.1 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 48.4.26: Timer synchronization (TIM15 only)*.

48.2 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)

48.3 TIM16/TIM17 main features

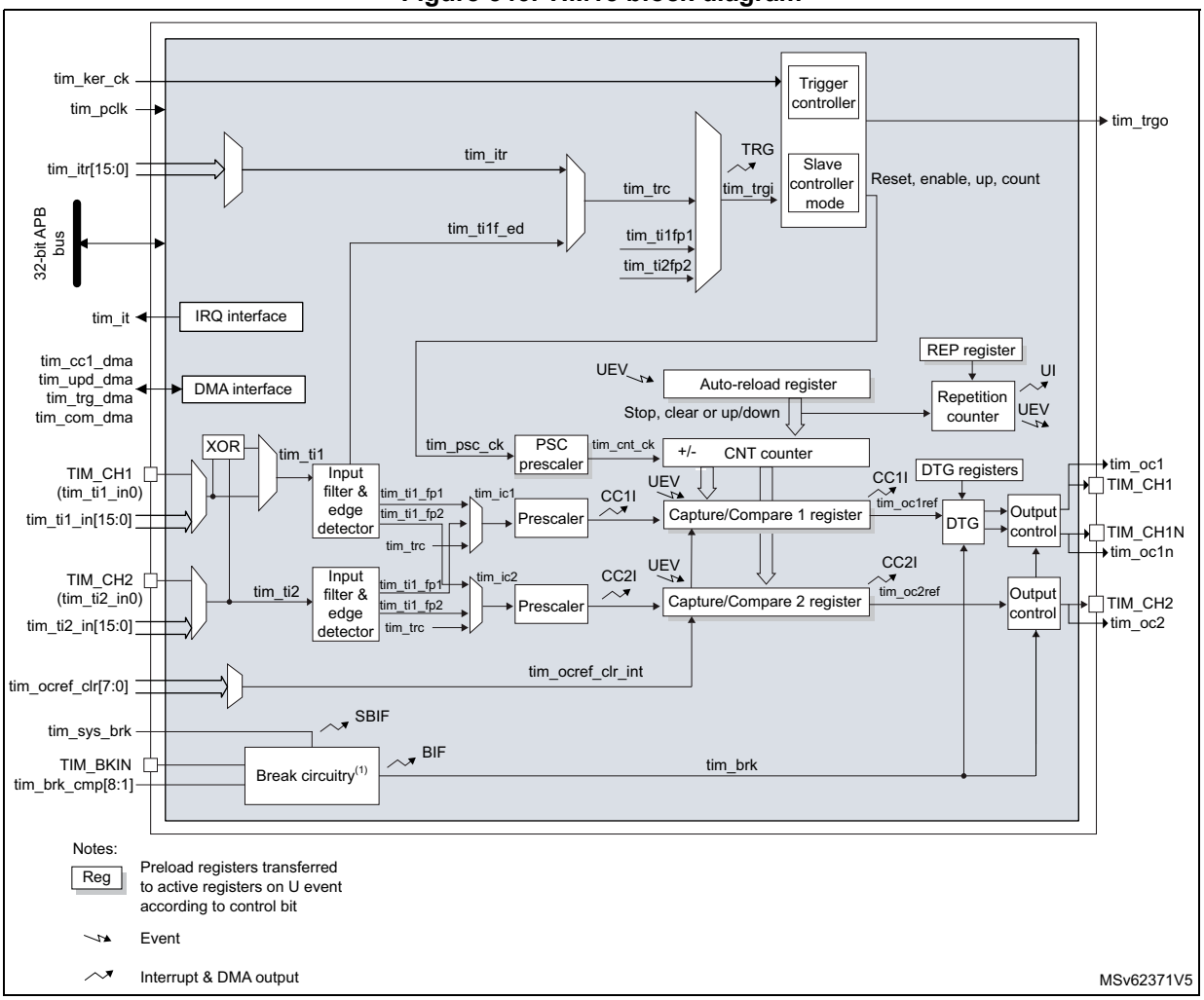
The TIM16/TIM17 timers include the following features:

- 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

48.4 TIM15/TIM16/TIM17 functional description

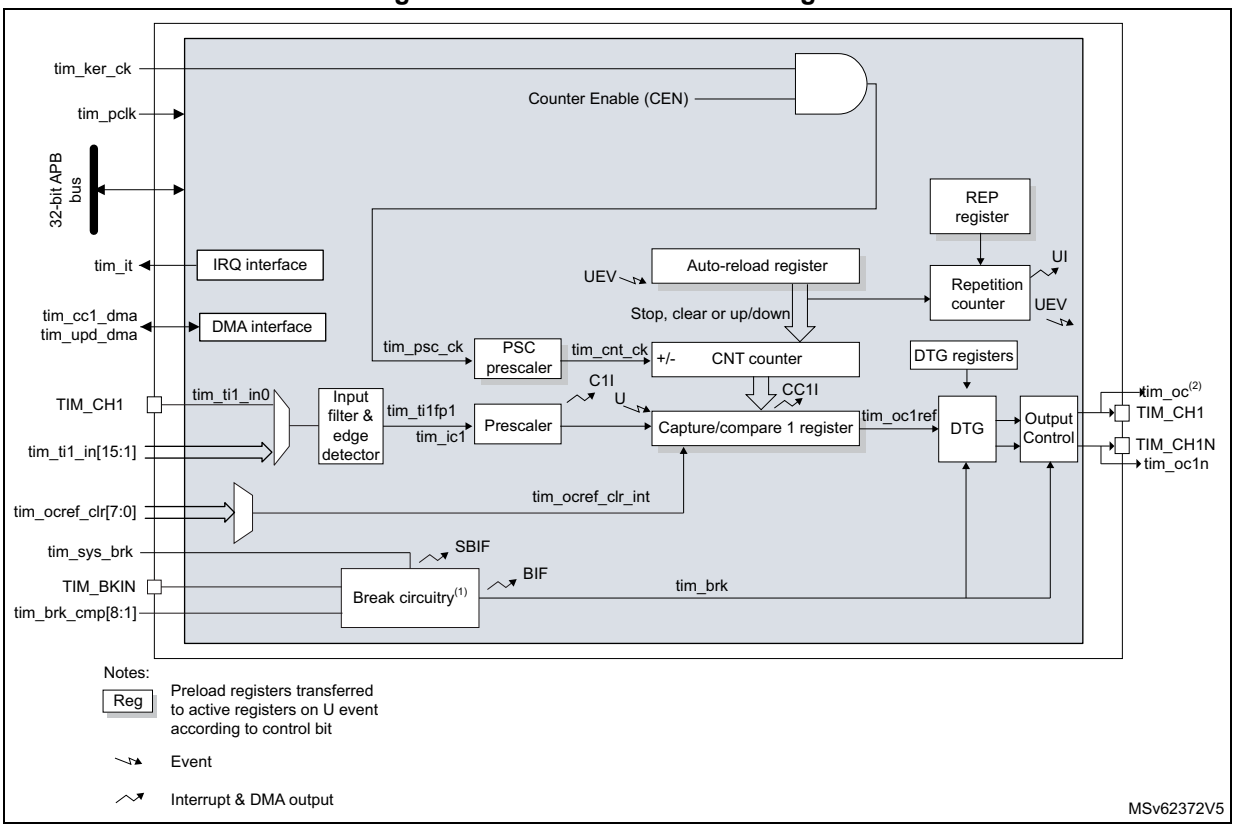
48.4.1 Block diagram

Figure 546. TIM15 block diagram



1. Refer to *Section 48.4.15: Using the break function* for details.

Figure 547. TIM16/TIM17 block diagram



1. Refer to *Section 48.4.15: Using the break function* for details.
2. This signal can be used as trigger for some slave timer (see internal trigger connection table in next section). See *Section 48.4.27: Using timer output as trigger for other timers (TIM16/TIM17 only)* for details.

48.4.2 TIM15/TIM16/TIM17 pins and internal signals

Table 466 and *Table 467* in this section summarize the TIM inputs and outputs.

Table 466. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2(1)	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) and external trigger inputs.
TIM_CH1N	Output	Timer complementary outputs, derived from TIM_CH1 output with the possibility to have deadtime insertion.
TIM_BKIN	Input / Output	Break input. This input can also be configured in bidirectional mode.

1. Available for TIM15 only.

Table 467. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0](1)	Input	Internal timer inputs bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock).
tim_itr[15:0](1)	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo(1)	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_ocref_clr[7:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.
tim_brk_cmp[8:1]	Input	Break input for internal signals
tim_sys_brk[n:0]	Input	System break input. This input gathers the MCU's system level errors.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 16 (maximum value)
tim_it	Output	Global Timer interrupt, gathering capture/compare, update, break trigger and commutation requests
tim_cc1_dma	Output	Timer capture / compare 1 dma request
tim_upd_dma	Output	Timer update dma request
tim_trg_dma	Output	Timer trigger dma request
tim_com_dma	Output	Timer commutation dma request

1. Available for TIM15 only.

The tables below list the sources connected to the tim_ti[2:1] input multiplexers.

Table 468. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources		
	TIM15	TIM16	TIM17
tim_ti1_in0	TIM15_CH1	TIM16_CH1	TIM17_CH1
tim_ti1_in1	LSE	Reserved	
tim_ti1_in2	comp1_out	MCO	
tim_ti1_in3	comp2_out	HSE / 32	
tim_ti1_in4	Reserved	rtc_wut_trg	
tim_ti1_in5		LSE	
tim_ti1_in6		LSI	

Table 468. Interconnect to the tim_ti1 input multiplexer (continued)

tim_ti1 inputs	Sources		
	TIM15	TIM16	TIM17
tim_ti1_in7	Reserved	MSIS / 1024	
tim_ti1_in8		MSIS / 4	
tim_ti1_in9		HSI / 256	
tim_ti1_in[15:10]	Reserved		

Table 469. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources
	TIM15
tim_ti2_in0	TIM15_CH2
tim_ti2_in1	comp2_out
tim_ti2_in[15:2]	Reserved

The table below lists the internal sources connected to the tim_itr input multiplexer.

Table 470. TIMx internal trigger connection

tim_itrx inputs	TIM15
tim_itr0	tim1_trgo
tim_itr1	tim2_trgo
tim_itr2	tim3_trgo
tim_itr3	tim4_trgo
tim_itr4	tim5_trgo
tim_itr5	tim8_trgo
tim_itr6	Reserved
tim_itr7	tim16_oc1
tim_itr8	tim17_oc1
tim_itr[15:9]	Reserved

The tables below list the sources connected to the tim_brk and tim_brk2inputs.

Table 471. Timer break interconnect

tim_brk inputs	TIM15	TIM16	TIM17
TIM_BKIN	TIM15_BKIN pin	TIM16_BKIN pin	TIM17_BKIN pin
tim_brk_cmp1	comp1_out	comp1_out	comp1_out

Table 471. Timer break interconnect (continued)

tim_brk inputs	TIM15	TIM16	TIM17
tim_brk_cmp2	comp2_out	comp2_out	comp2_out
tim_brk_cmp[8:3]	Reserved		

Table 472. System break interconnect

tim_sys_brk inputs	TIM15 / TIM16 / TIM17	Enable bit in SYSCFG_CFGR2 register
tim_sys_brk0	Cortex®-M33 LOCKUP	CLL
tim_sys_brk1	Programmable Voltage Detector (PVD)	PVDL
tim_sys_brk2	SRAM double ECC error	SPL
tim_sys_brk3	Flash double ECC error	ECCL
tim_sys_brk4	Clock Security System (CSS)	None (always enabled)

The table below lists the internal sources connected to the tim_ocref_clr input multiplexer.

Table 473. Interconnect to the ocref_clr input multiplexer

Timer OCREF clear signal	Timer OCREF clear signals assignment		
	TIM15	TIM16	TIM17
tim_ocref_clr0	comp1_out	comp1_out	comp1_out
tim_ocref_clr1	comp2_out	comp2_out	comp2_out
tim_ocref_clr[7:2]	Reserved		

48.4.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 548 and *Figure 549* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 548. Counter timing diagram with prescaler division change from 1 to 2

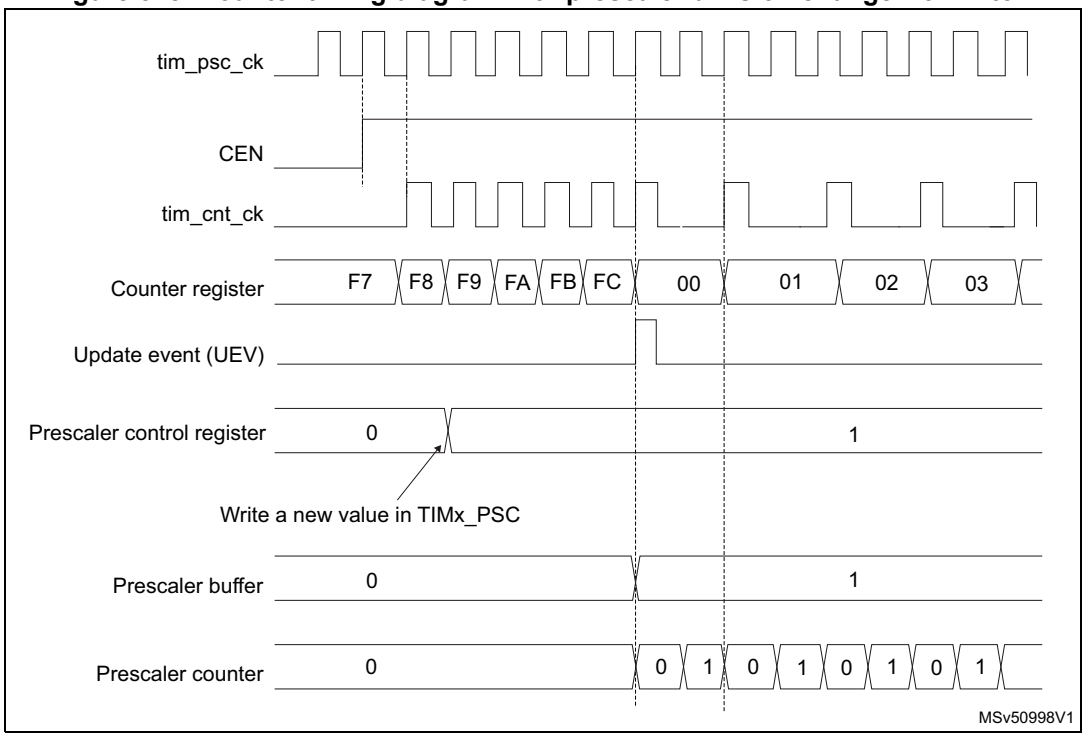
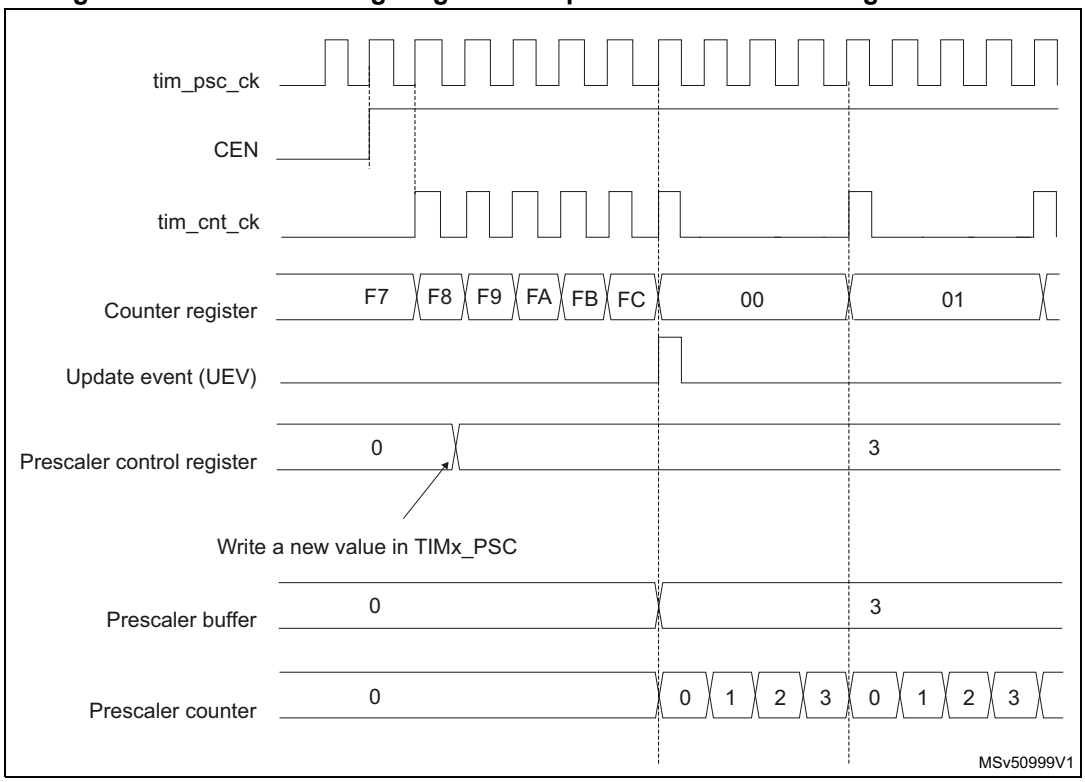


Figure 549. Counter timing diagram with prescaler division change from 1 to 4



48.4.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 550. Counter timing diagram, internal clock divided by 1

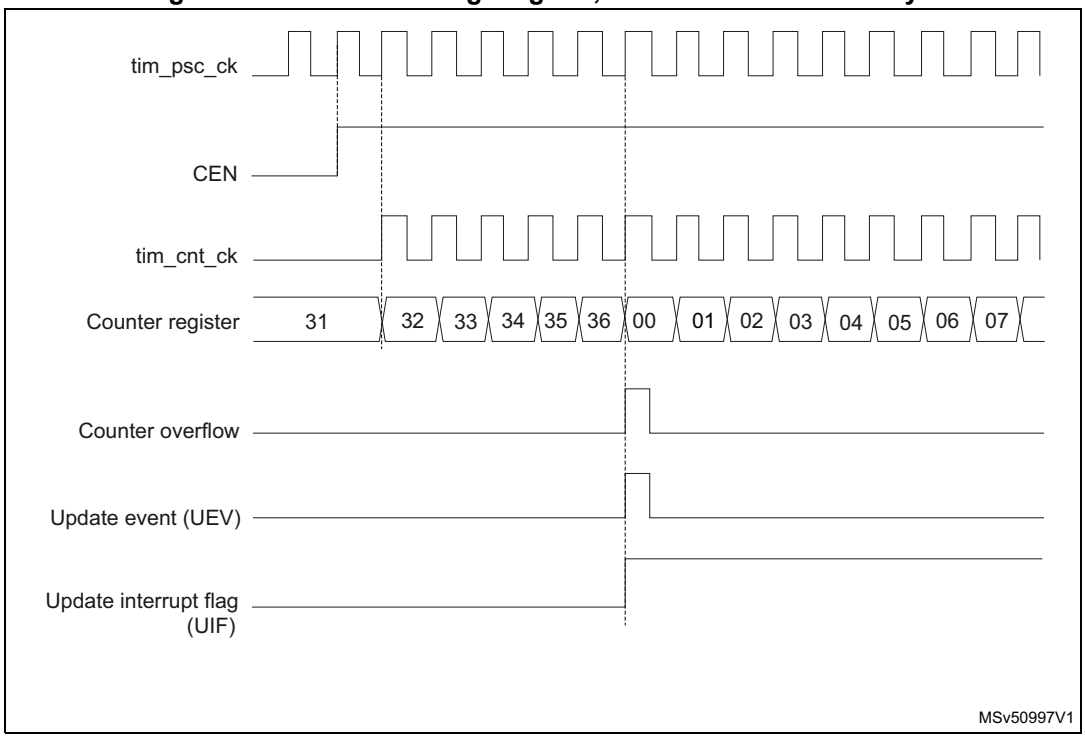


Figure 551. Counter timing diagram, internal clock divided by 2

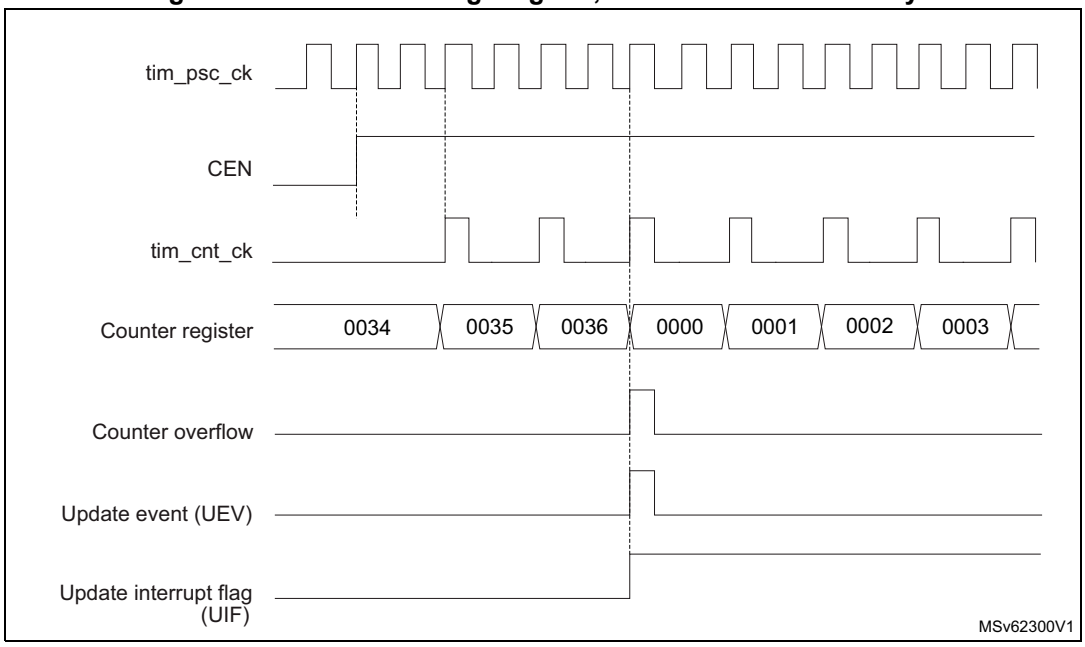


Figure 552. Counter timing diagram, internal clock divided by 4

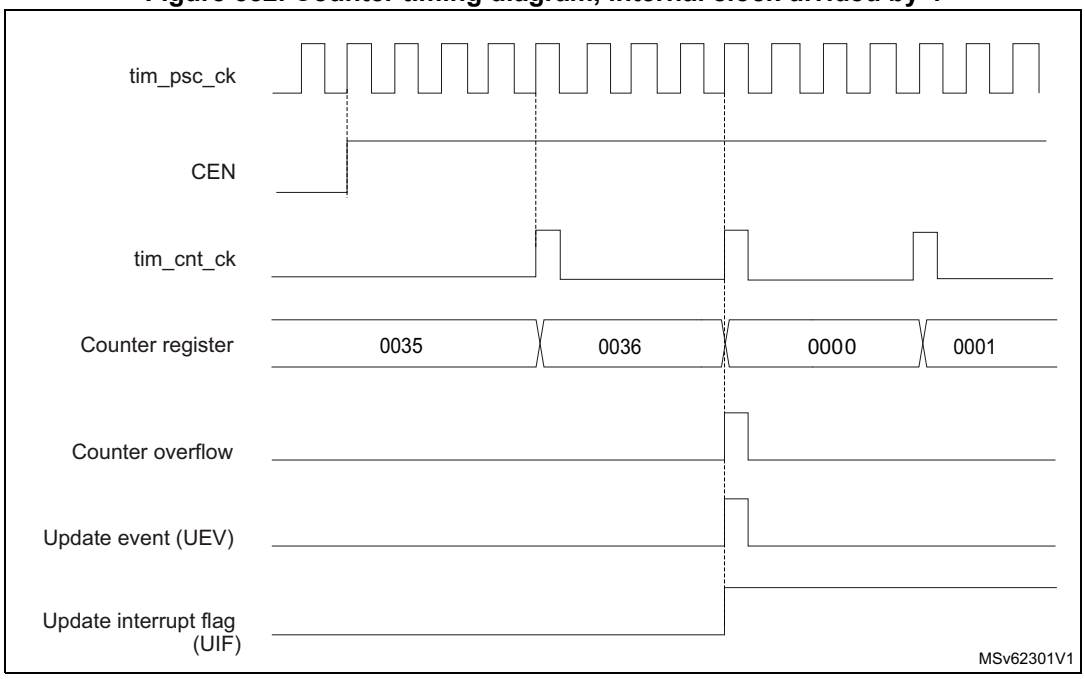


Figure 553. Counter timing diagram, internal clock divided by N

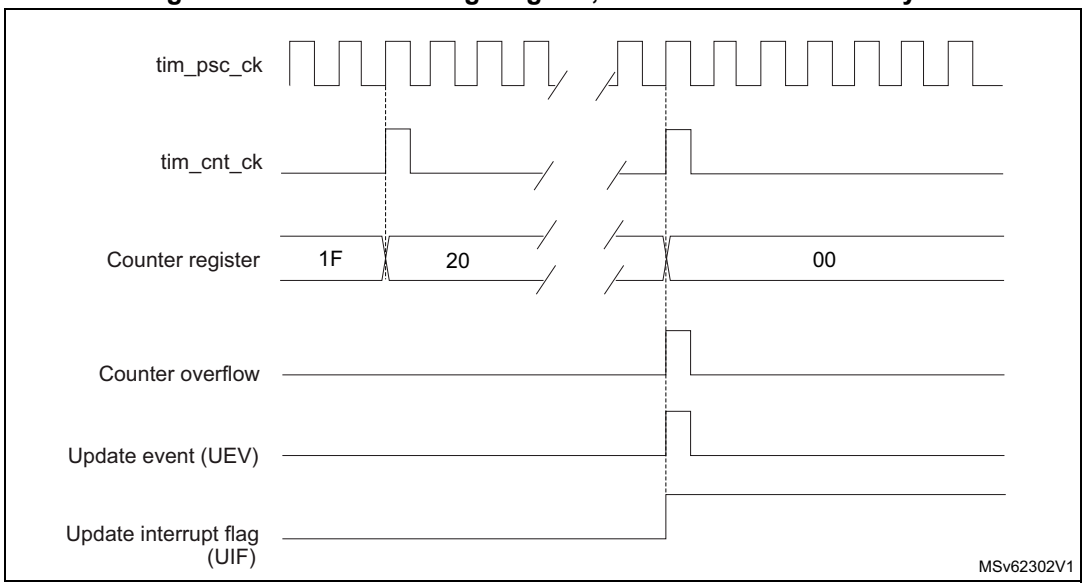


Figure 554. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

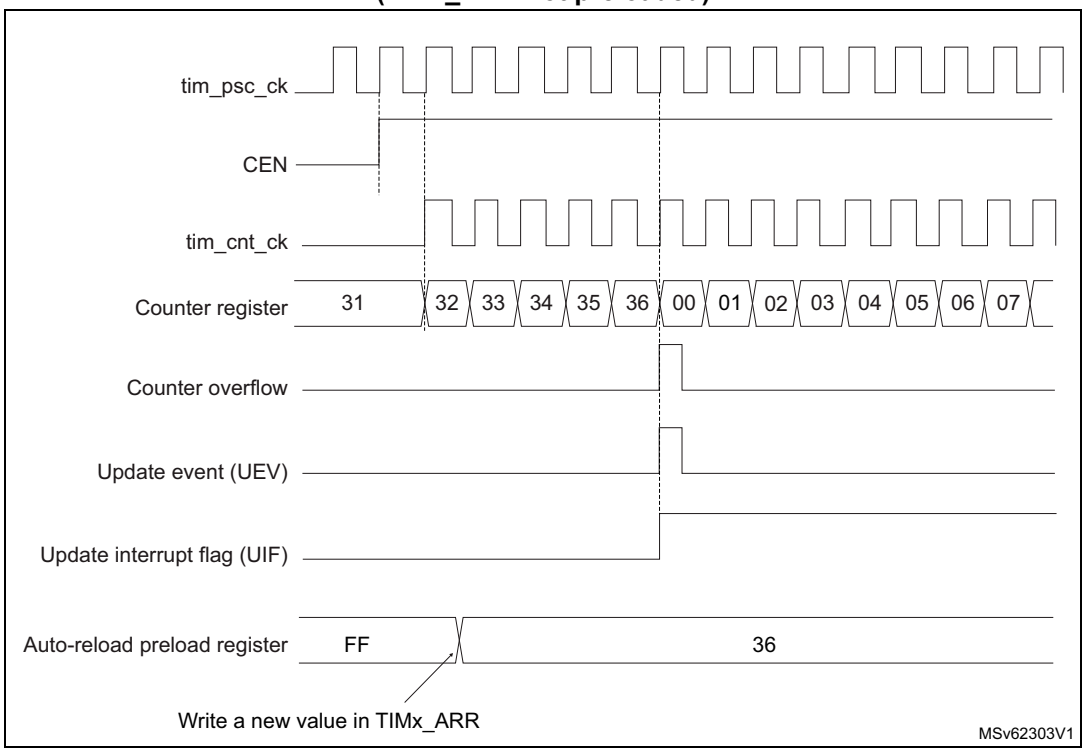
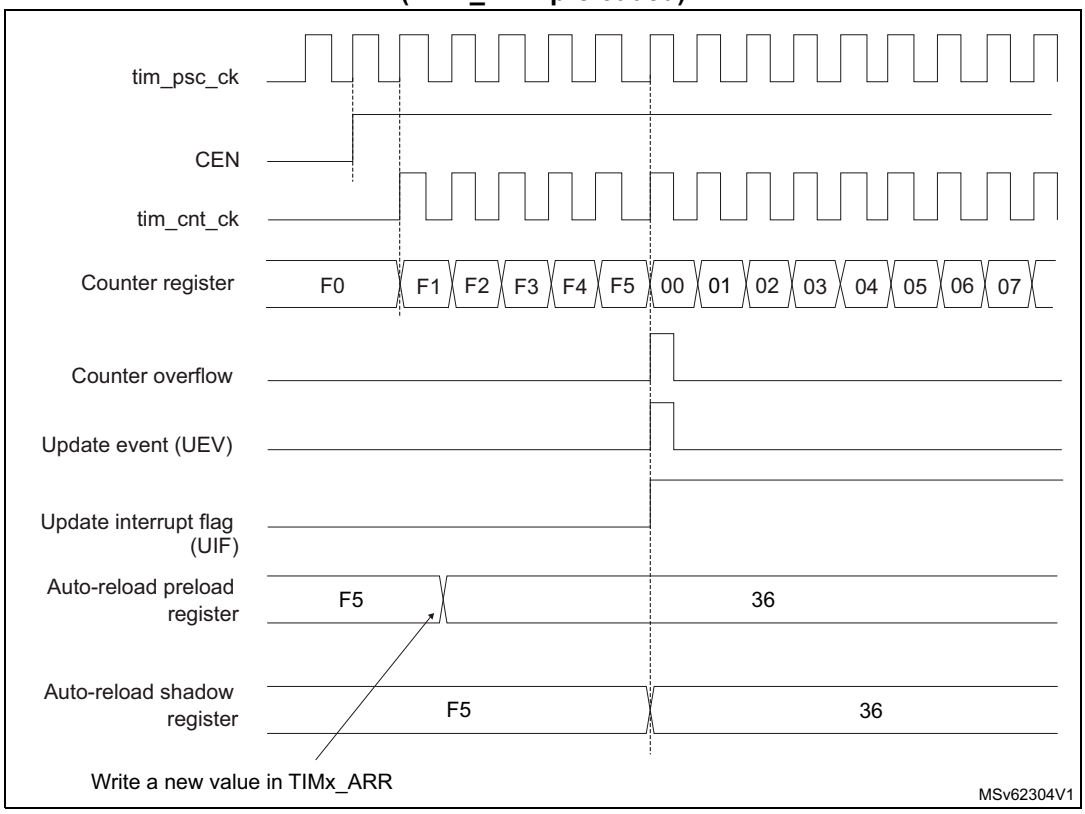


Figure 555. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



48.4.5 Repetition counter

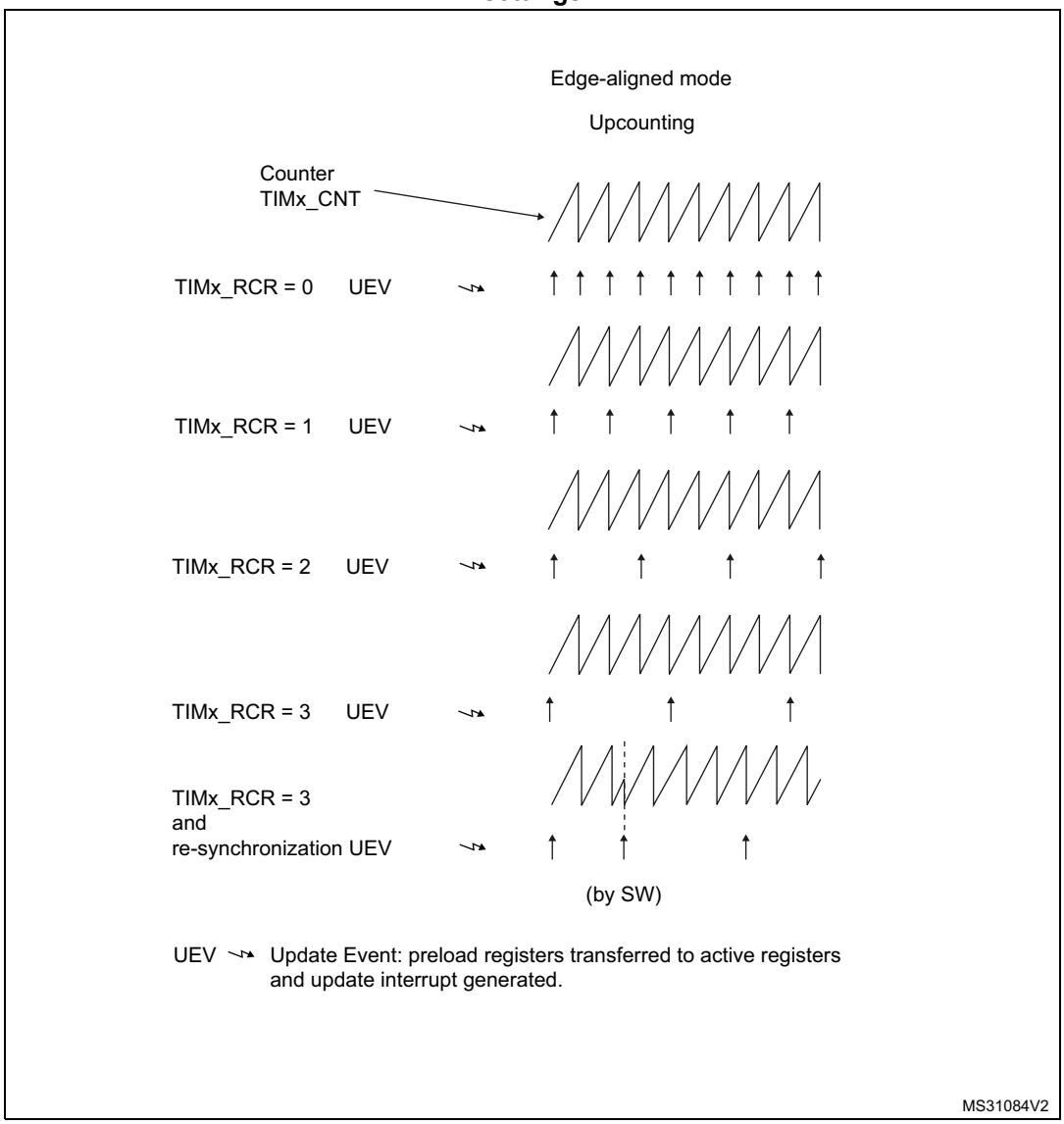
Section 48.4.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N counter overflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented at each counter overflow.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 556*). 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 556. Update rate examples depending on mode and TIMx_RCR register settings



48.4.6 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2, if available)
- Internal trigger inputs (tim_itrx) (only for TIM15): using one timer as the prescaler for another timer, for example, TIM1 can be configured to act as a prescaler for TIM15. Refer to *Using one timer to enable another timer* for more details.

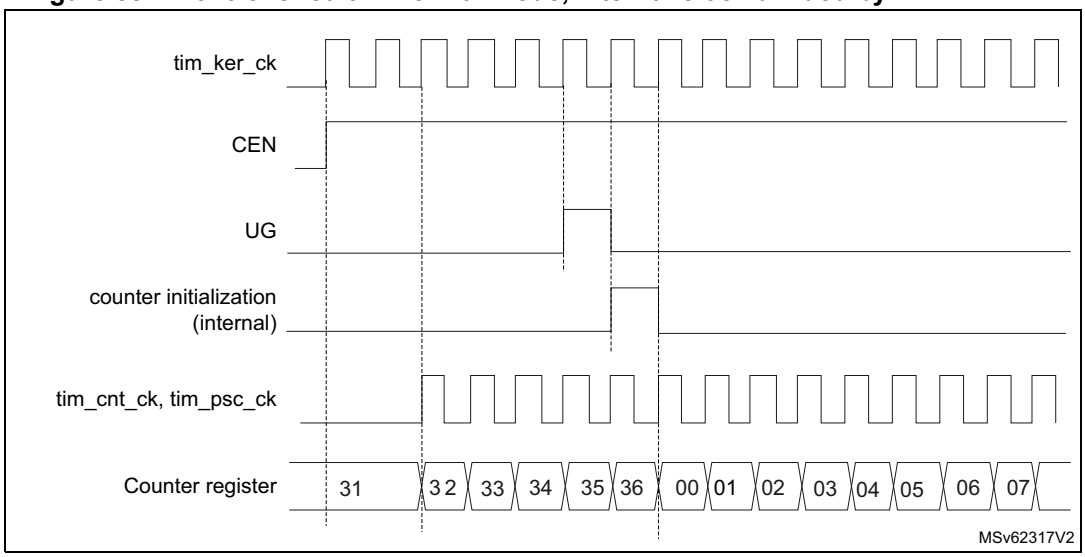
Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS=000), then the CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed

only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 557 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

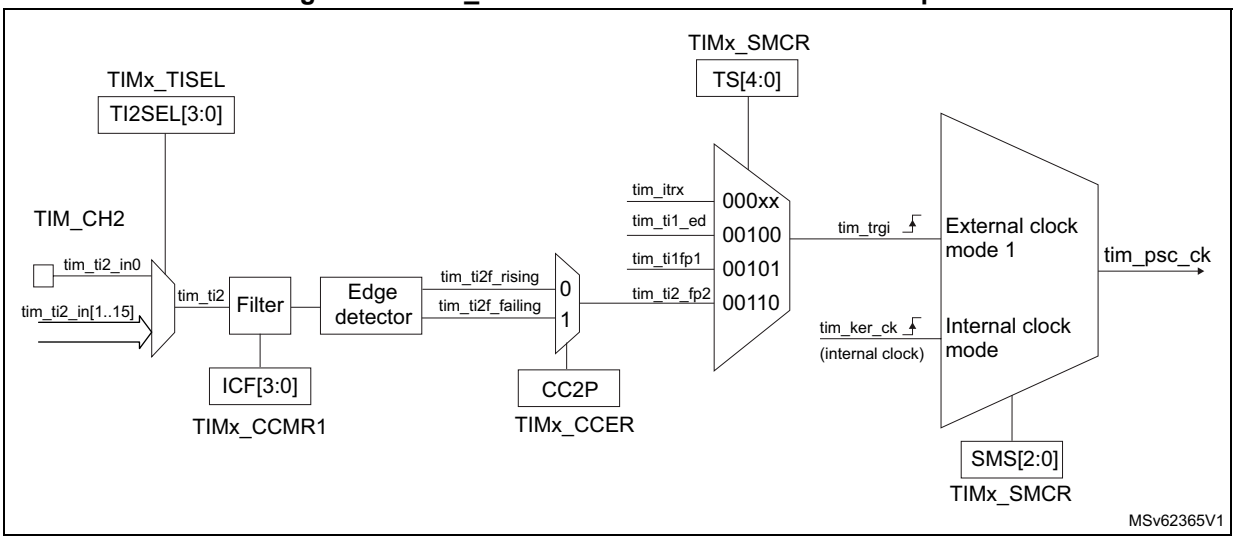
Figure 557. Control circuit in normal mode, internal clock divided by 1



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 558. tim_ti2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

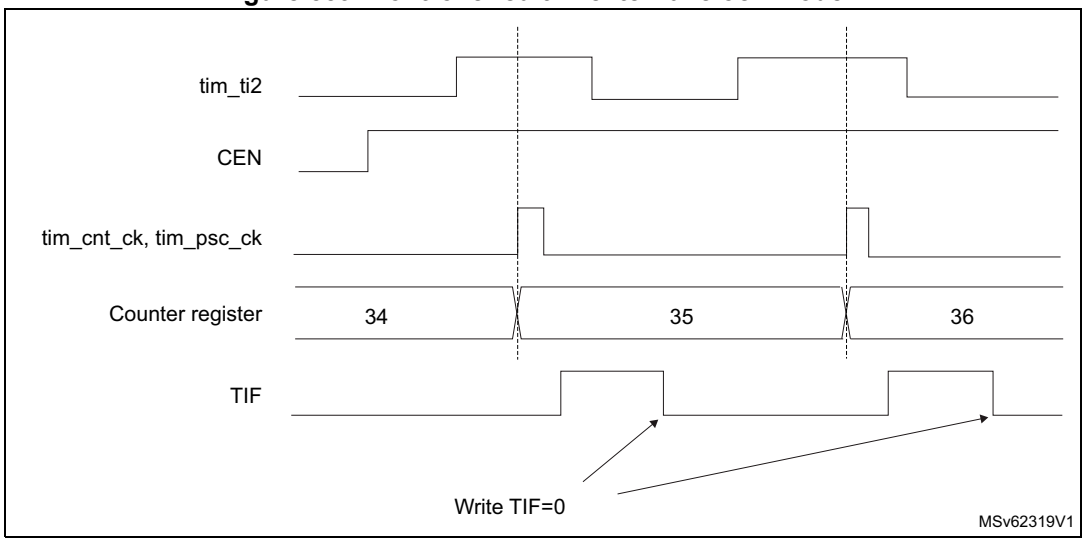
1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
4. Select rising edge polarity by writing CC2P=0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
6. Select tim_ti2 as the trigger input source by writing TS=00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 559. Control circuit in external clock mode 1



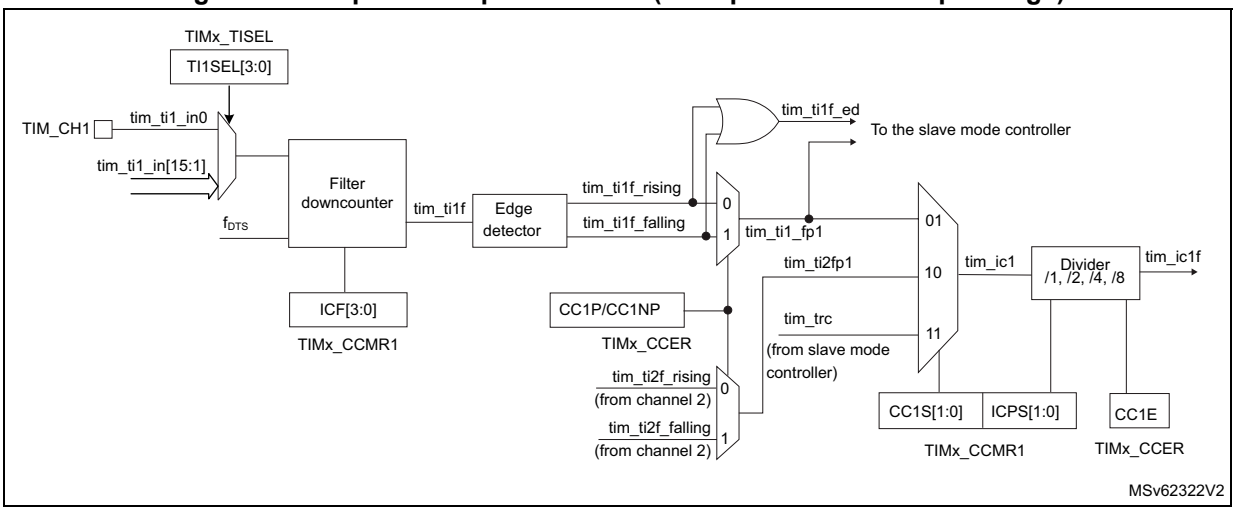
48.4.7 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 560 to *Figure 563* give an overview of one Capture/Compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 560. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 561. Capture/compare channel 1 main circuit

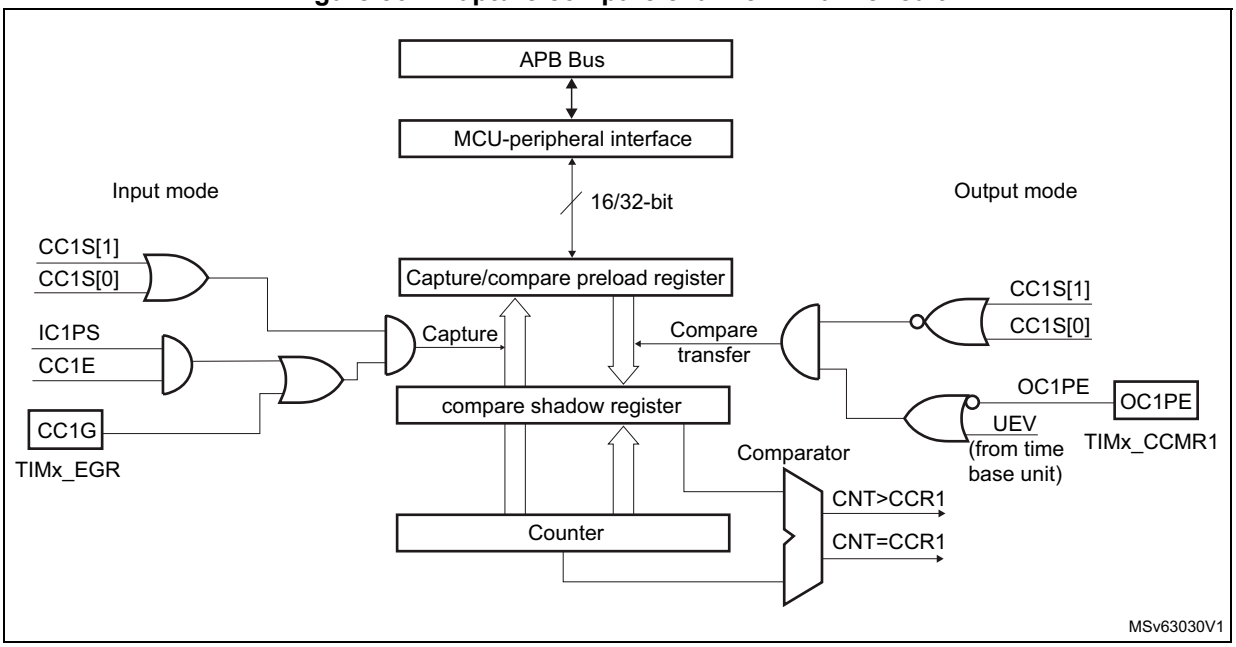


Figure 562. Output stage of capture/compare channel (channel 1)

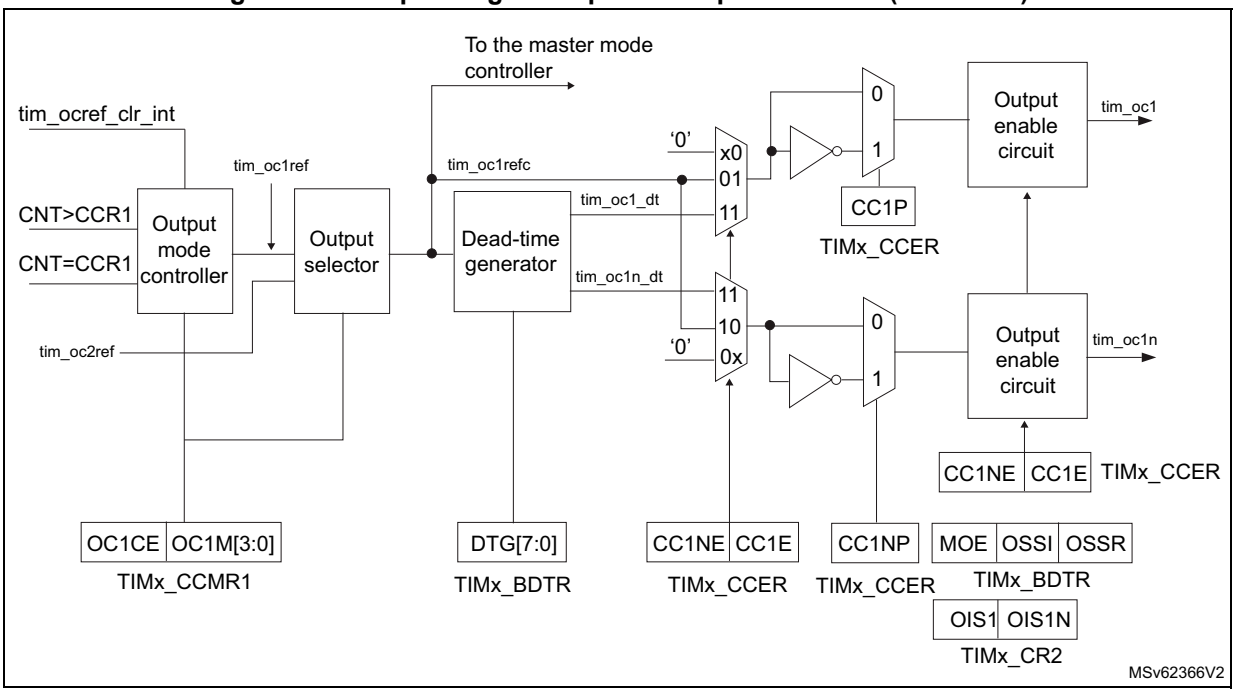
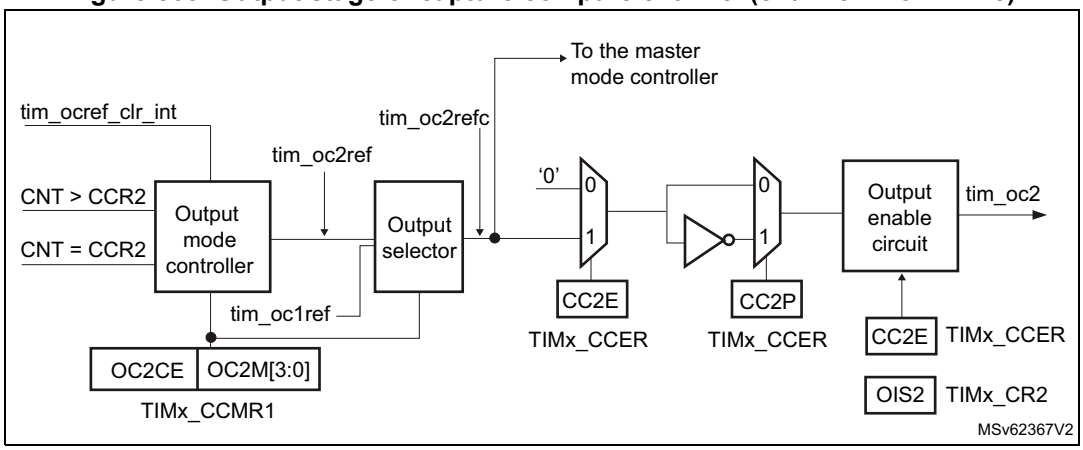


Figure 563. Output stage of capture/compare channel (channel 2 for TIM15)



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.

48.4.8 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding tim_icx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was

already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to ‘0’ or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_ti1_in[15:1] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let’s imagine that, when toggling, the input signal is not stable during at least 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on tim_ti1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by writing CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to ‘00’ in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

48.4.9 PWM input mode (only for TIM15)

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges)
- The TIMx_CCR2 register holds the pulsewidth (interval between two consecutive rising and falling edges

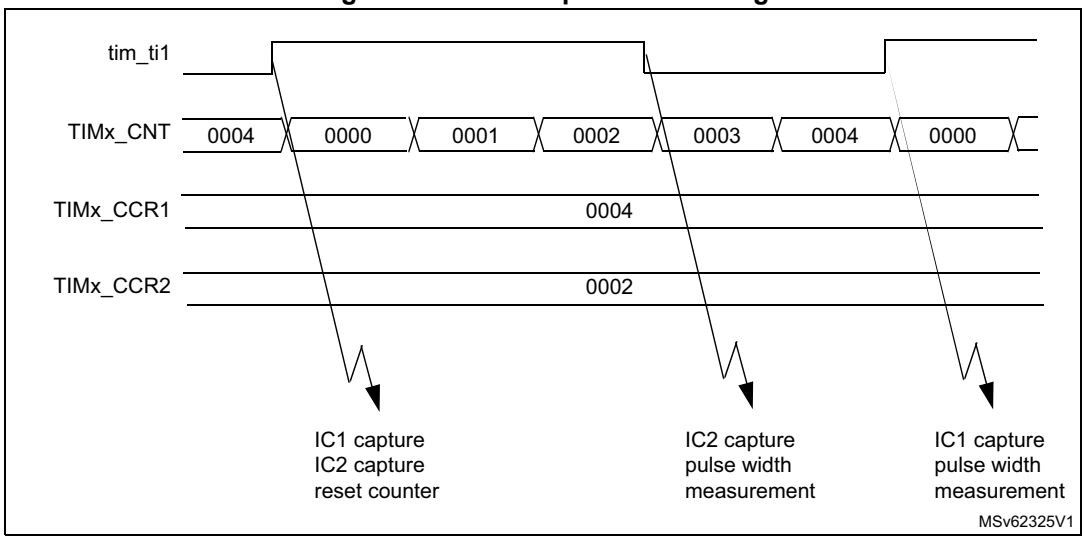
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two tim_icx signals are mapped on the same tim_tix input.
- These 2 tim_icx signals are active on edges with opposite polarity.
- One of the two tim_tixfpy signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on tim_ti1 using the following procedure (depending on tim_ker_ck frequency and prescaler value):

1. Select the proper tim_ti1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to ‘0’ (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to ‘10’ (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.

Figure 564. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only tim_ti1fp1 and tim_ti2fp2 are connected to the slave mode controller.

48.4.10 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_ocxn) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP=0 (tim_ocx active high) => tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

48.4.11 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP

bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

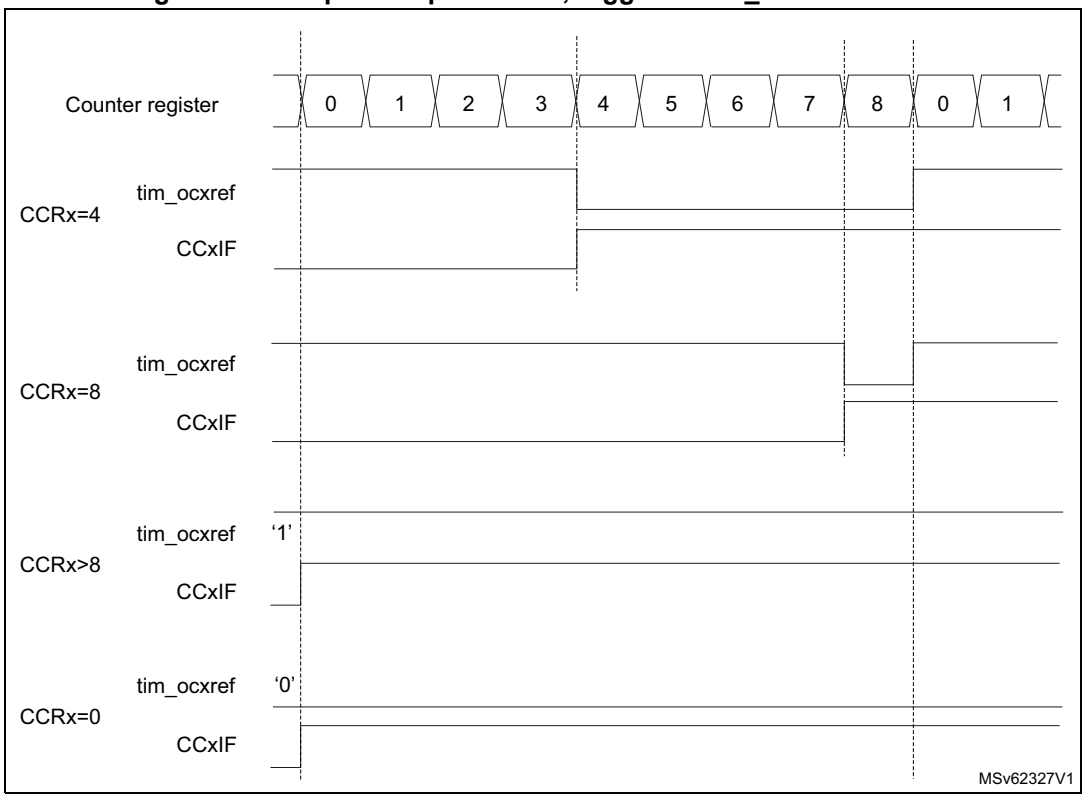
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 565*.

Figure 565. Output compare mode, toggle on tim_oc1



48.4.12 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing '110' (PWM mode 1) or '111' (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

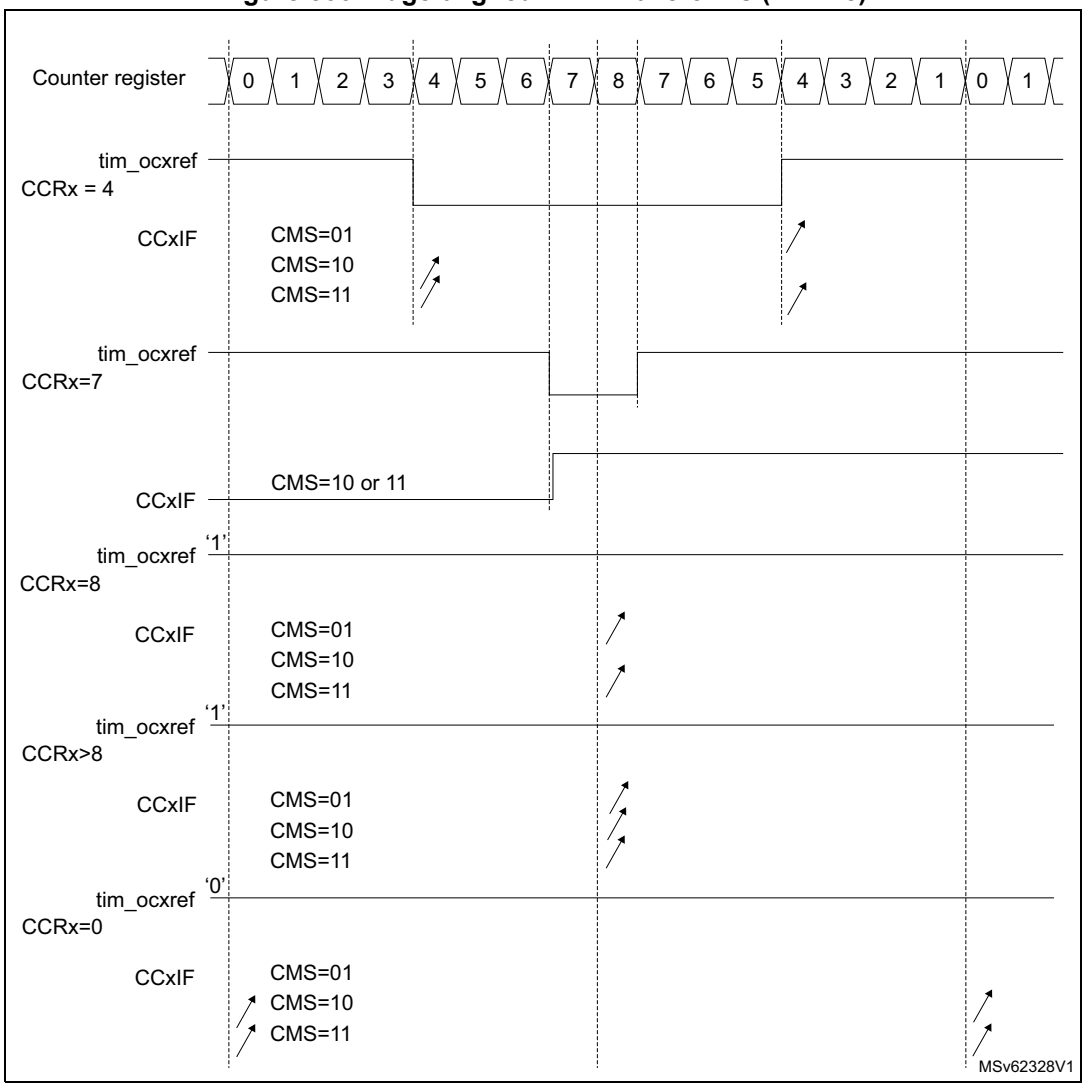
tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The TIM15/TIM16/TIM17 are capable of upcounting only. Refer to *Upcounting mode on page 1894*.

In the following example, we consider PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then tim_ocxref is held at ‘1’. If the compare value is 0 then tim_ocxref is held at ‘0’. *Figure 566* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 566. Edge-aligned PWM waveforms (ARR=8)

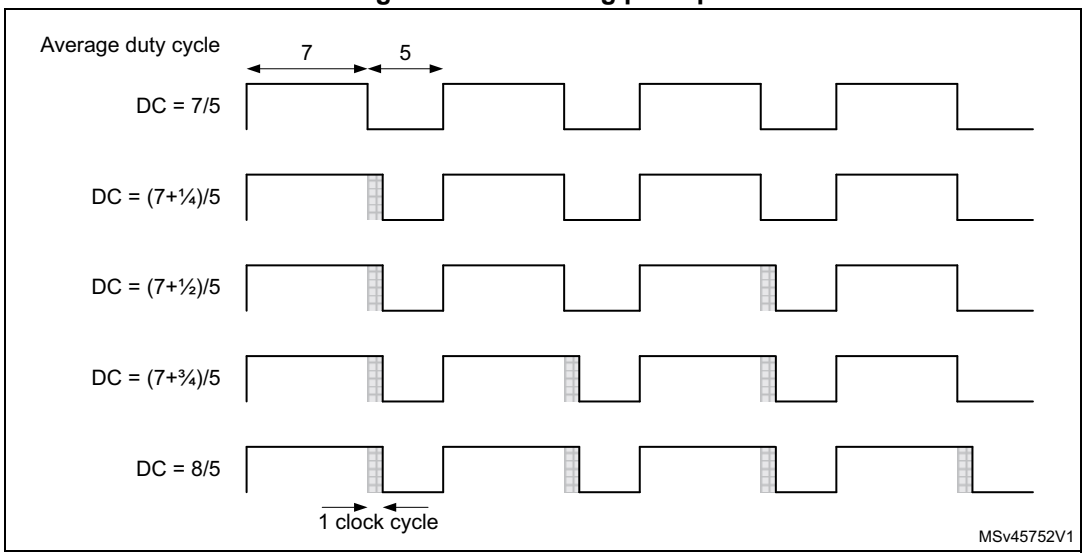


Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. The *Figure 567* below presents the dithering principle applied to 4 consecutive PWM cycles.

Figure 567. Dithering principle



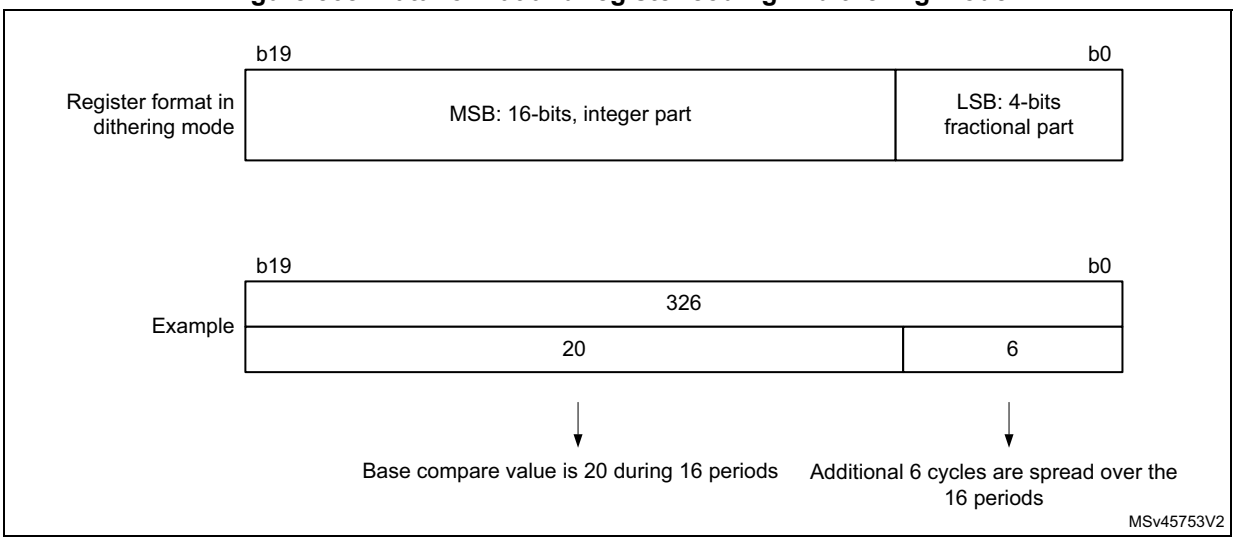
When the dithering mode is enabled, the register coding is changed as following (see *Figure 568* for example):

- the 4 LSBs are coding for the enhanced resolution part (fractional part)
- the MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:

1. *CEN and ARPE bits must be reset*
2. *The ARR[3:0] bits must be reset*
3. *The DITHEN bit must be reset*
4. *The CCIF flags must be cleared*
5. *The CEN bit can be set (eventually with ARPE = 1).*

Figure 568. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

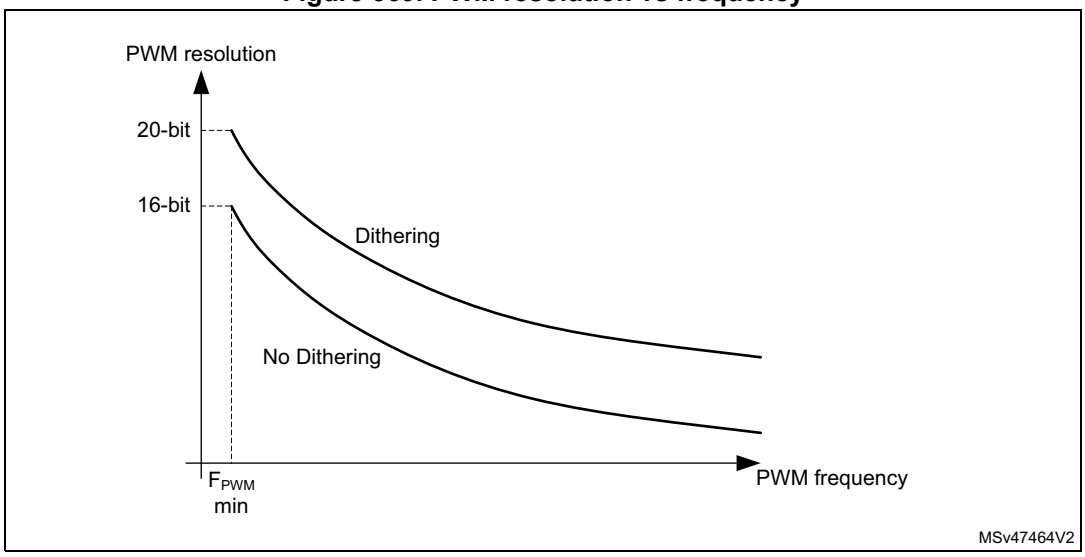
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

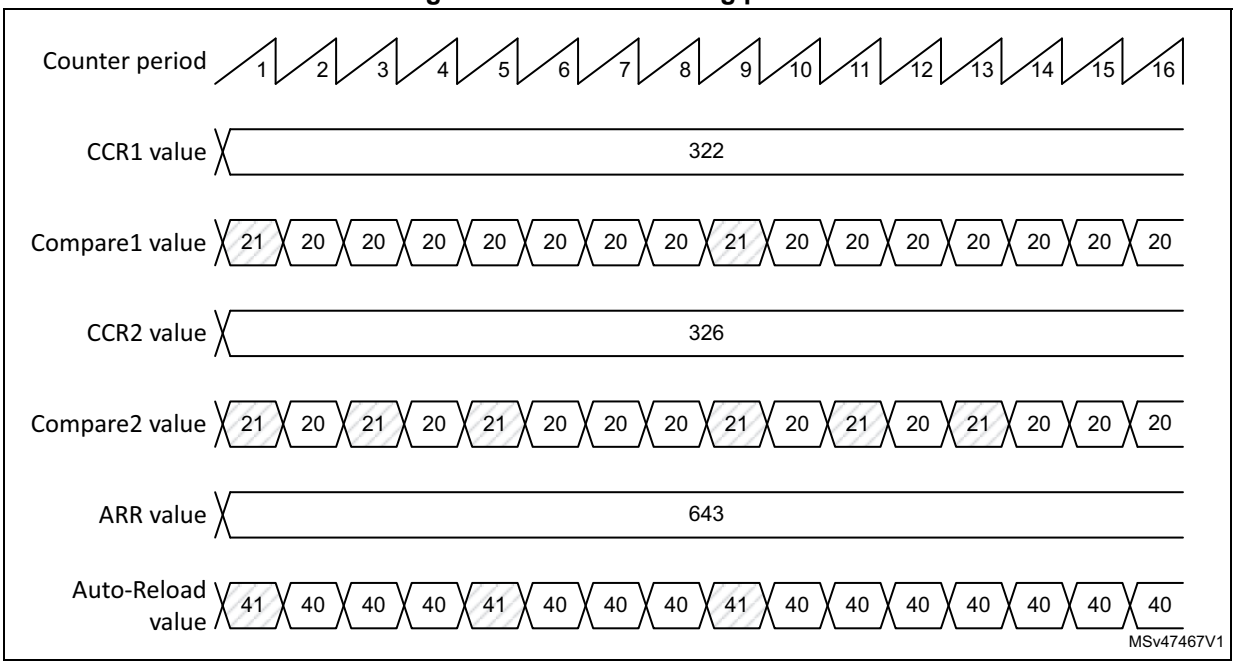
As shown on the *Figure 569* below, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 569. PWM resolution vs frequency



The duty cycle and / or period changes are spread over 16 consecutive periods, as described in the *Figure 570* below.

Figure 570. PWM dithering pattern



The auto-reload and compare values increments are spread following specific patterns described in the *Table 474* below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 474. CCR and ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-

Table 474. CCR and ARR register change dithering pattern (continued)

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

48.4.13 Combined PWM mode (TIM15 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by the TIMx_CCR1 and TIMx_CCR2 registers

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

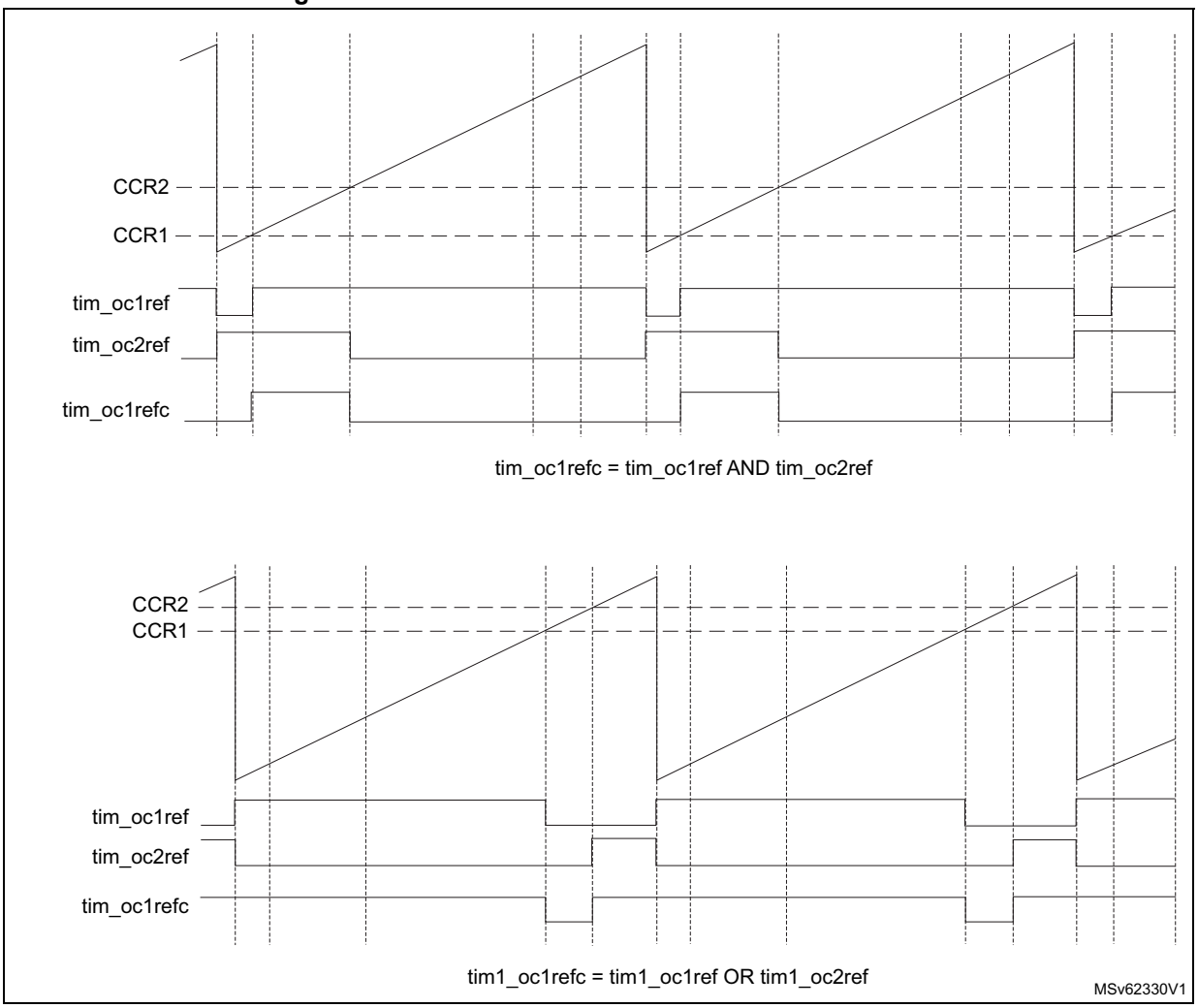
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 571 represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,

Figure 571. Combined PWM mode on channel 1 and 2



48.4.14 Complementary outputs and dead-time insertion

The TIM15/TIM16/TIM17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

The polarity of the outputs (main output tim_ocx or complementary tim_ocxn) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals tim_ocx and tim_ocxn are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to *Table 481: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17) on page 1973* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

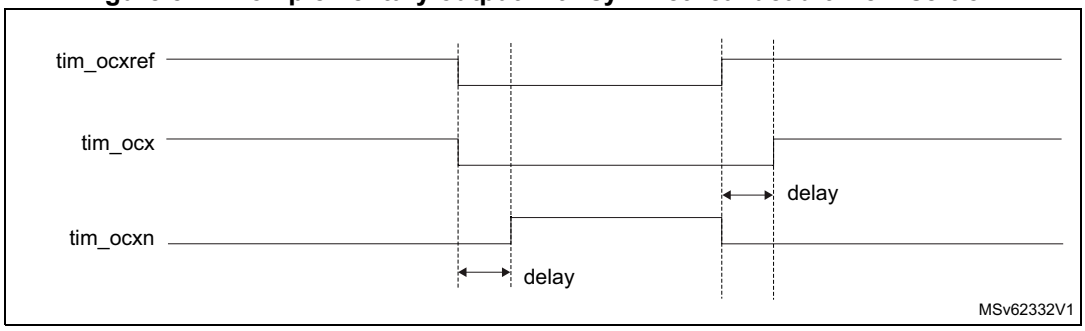
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform tim_ocxref, it generates 2 outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_ocx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The tim_ocxn output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (tim_ocx or tim_ocxn) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal tim_ocxref. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

Figure 572. Complementary output with symmetrical dead-time insertion.



The DTAE bit in the TIMx_DTR2 is used to differentiate the deadtime values for rising and falling edges of the reference signal, as shown on *Figure 573*.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bitfield in the TIMx_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bitfield in the TIMx_DTR2 register. The DTAE bit must be written before enabling the counter and must be not modified while CEN = 1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bitfield DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set, in the TIMX_DTR2 register. The preload value is loaded in the active register on the next update event.

Note: If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.

Figure 573. Asymmetrical deadtime

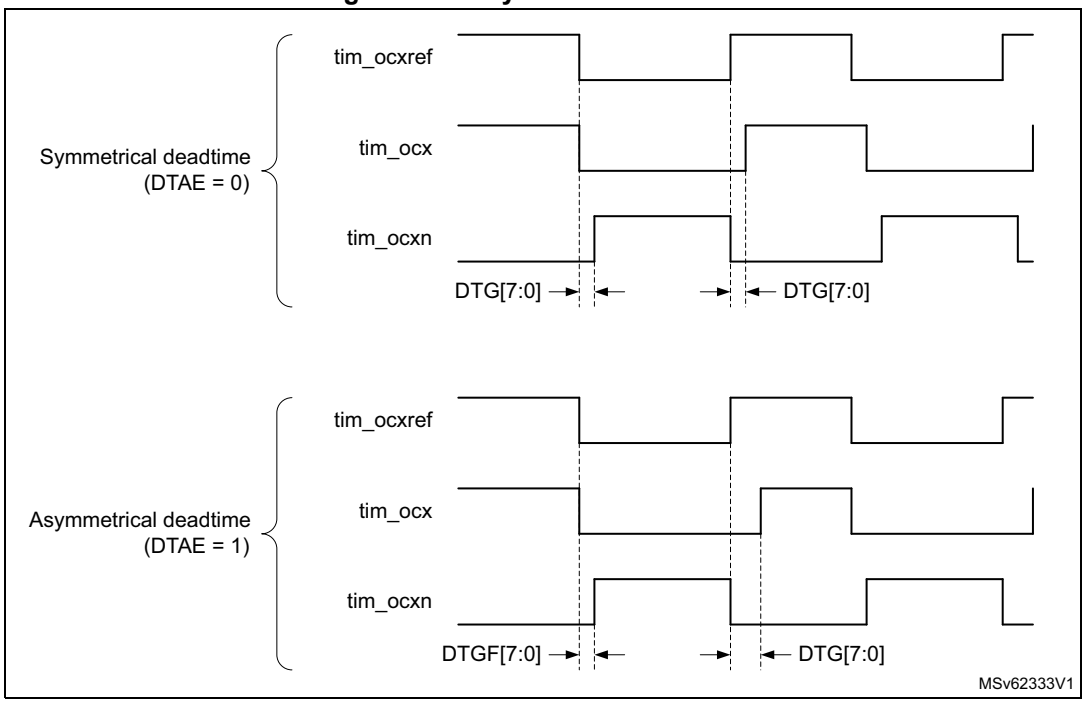


Figure 574. Dead-time waveforms with delay greater than the negative pulse.

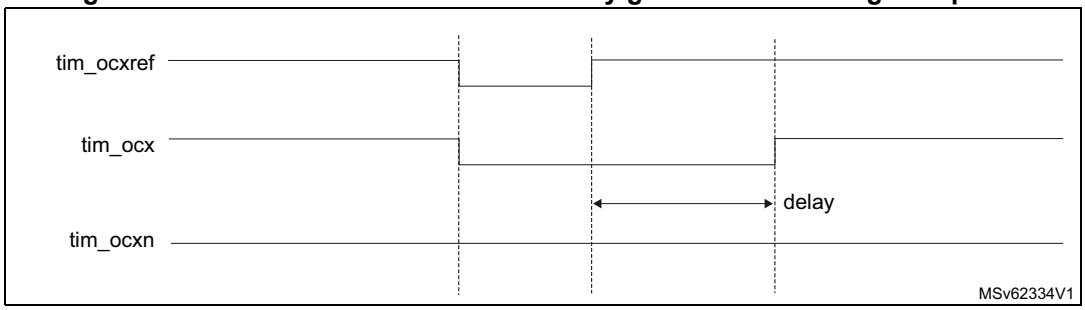
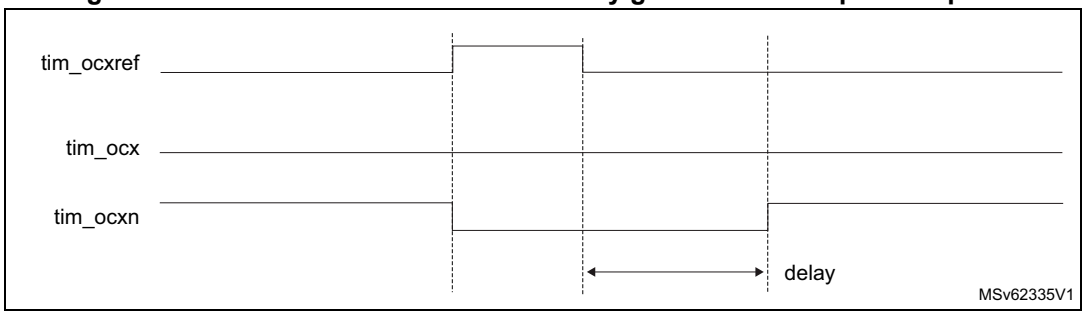


Figure 575. Dead-time waveforms with delay greater than the positive pulse.



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 48.8.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) on page 1977* for delay calculation.

Re-directing tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be re-directed to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This is used to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only tim_ocxn is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP=0 then tim_ocxn=tim_ocxref. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE=CCxNE=1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

48.4.15 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

The break channel gathers both system-level fault (clock failure, ECC / parity errors,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration.

The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx_BDTR register is used to enable /disable the outputs by software and is reset in case of break or break2 event.
- the OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- the OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 481: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17) on page 1973* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break function is enabled by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The break is generated by the tim_brk inputs which has:

- Programmable polarity (BKP bit in the TIMx_BDTR register)
- Programmable enable bit (BKE bit in the TIMx_BDTR register)
- Programmable filter (BKF[3:0] bits in the TIMx_BDTR register) to avoid spurious events.

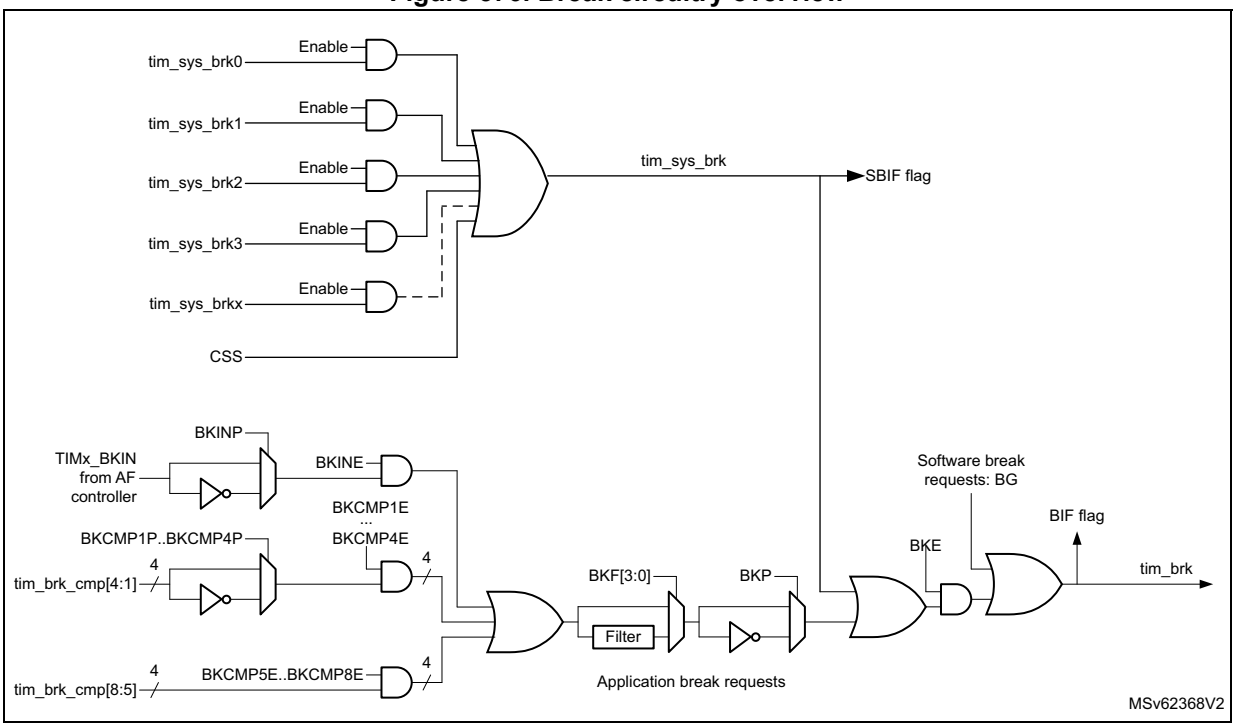
The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx_AF1 register.

The sources for break (tim_brk) channel are:

- External sources connected to one of the TIM_BKIN pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources:
  - coming from a tim_brk_cmpx input (refer to *Section 48.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation)
  - coming from a system break request on the tim_sys_brk inputs (refer to *Section 48.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation)

Break events can also be generated by software using BG bit in the TIMx_EGR register. All sources are ORed before entering the timer tim_brk inputs, as per *Figure 576* below.

Figure 576. Break circuitry overview



Caution: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example, using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the GPIO (selected by the OSSI bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE=0. If OSSI=0, the timer releases the output control (taken over by the GPIO) else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, tim_ocx and tim_ocxn cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 tim_ker_ck clock cycles).
  - If OSSI=0 then the timer releases the enable outputs (taken over by the GPIO which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written with 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value. If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.

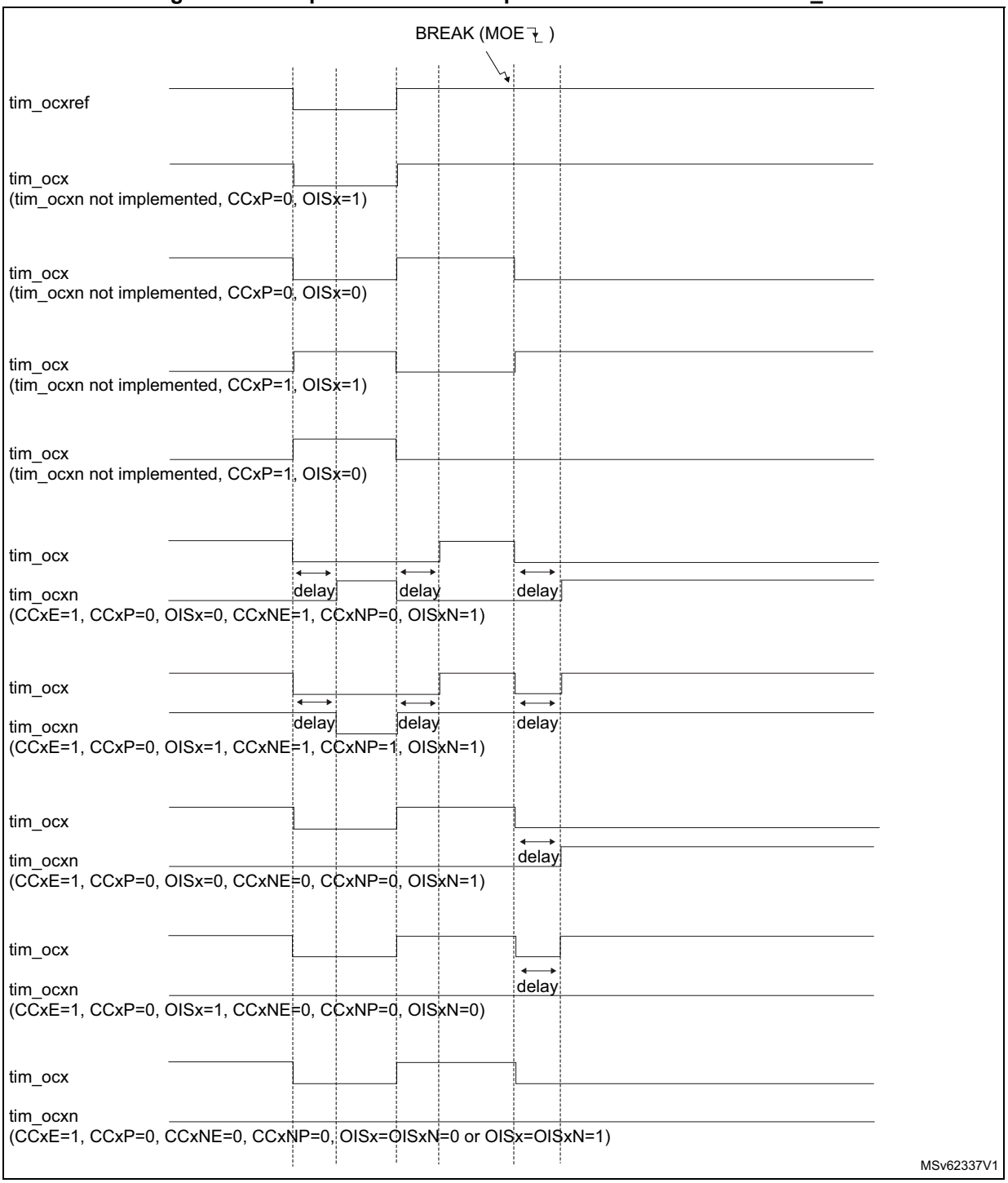
Note: The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the tim_brk input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR Register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It is used to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_ocxn polarities and state when disabled, OCxM configurations, break enable and polarity). The protection can be selected among 3 levels with the LOCK bits in the TIMx_BDTR register. Refer to *Section 48.8.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)*. The LOCK bits can be written only once after an MCU reset.

The *Figure 577* shows an example of behavior of the outputs in response to a break.

Figure 577. Output behavior in response to a break event on tim_brk



48.4.16 Bidirectional break input

The TIM15/TIM16/TIM17 are featuring bidirectional break I/Os, as represented on *Figure 578*.

They are used to have:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin
- Internal break sources and multiple external open drain sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged

The tim_brk input is configured in bidirectional mode using the BKBID bit in the TIMxBDTR register. The BKBID programming bit can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode requires the I/O to be configured in open-drain mode with active low polarity (using BKINP and BKP bits). Any break request coming either from system (e.g. CSS), from on-chip peripherals or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software event (triggered by setting the BG bit) also causes the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BKE = 1). When a software break event is generated with BKE = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the TIM_BKIN I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM bit is set and the open drain control is released. This prevents the PWM output to be re-started as long as the break condition is present.
- The BKDSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 475*)

Table 475. Break protection disarming conditions

MOE	BKBID	BKDSRM	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and re-arming break circuitry

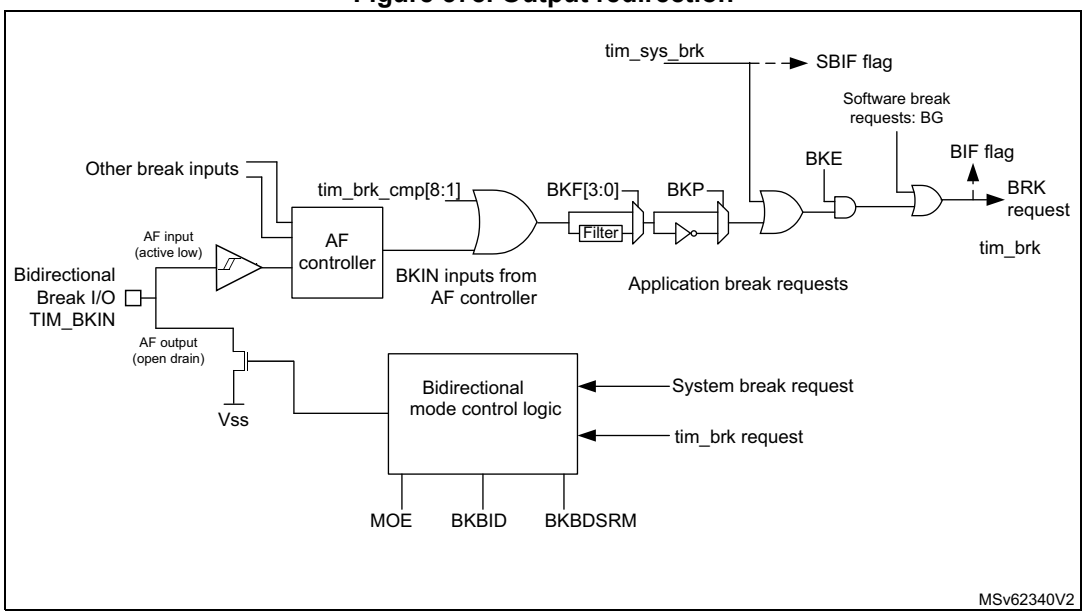
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break event:

- The BKDSRM bit must be set to release the output control
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before re-arming)
- The software must poll the BKDSRM bit until it is cleared by hardware (when the application break condition disappears)

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 578. Output redirection

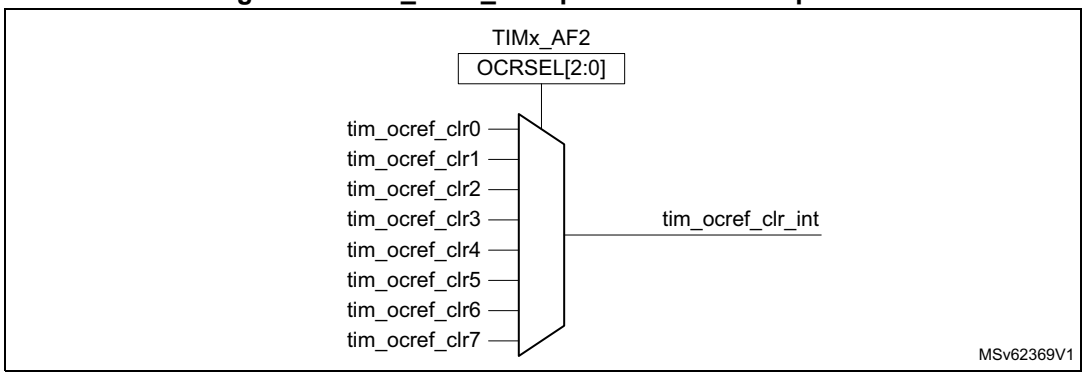


48.4.17 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next update event (UEV) occurs. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

The tim_ocref_clr_int input can be selected among several inputs, as shown on *Figure 579* below.

Figure 579. tim_ocref_clr input selection multiplexer



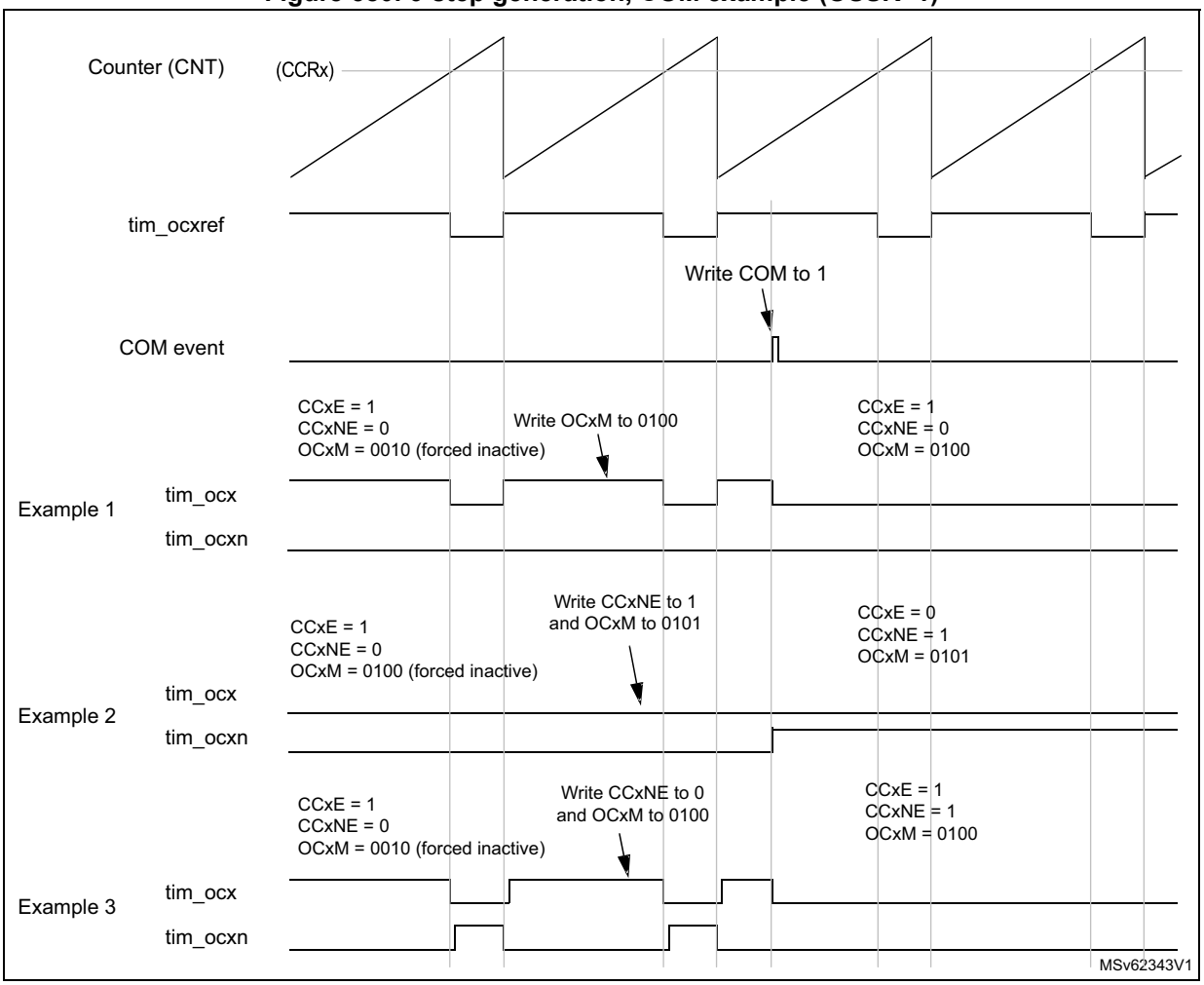
48.4.18 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The *Figure 580* describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 580. 6-step generation, COM example (OSSR=1)



48.4.19 One-pulse mode

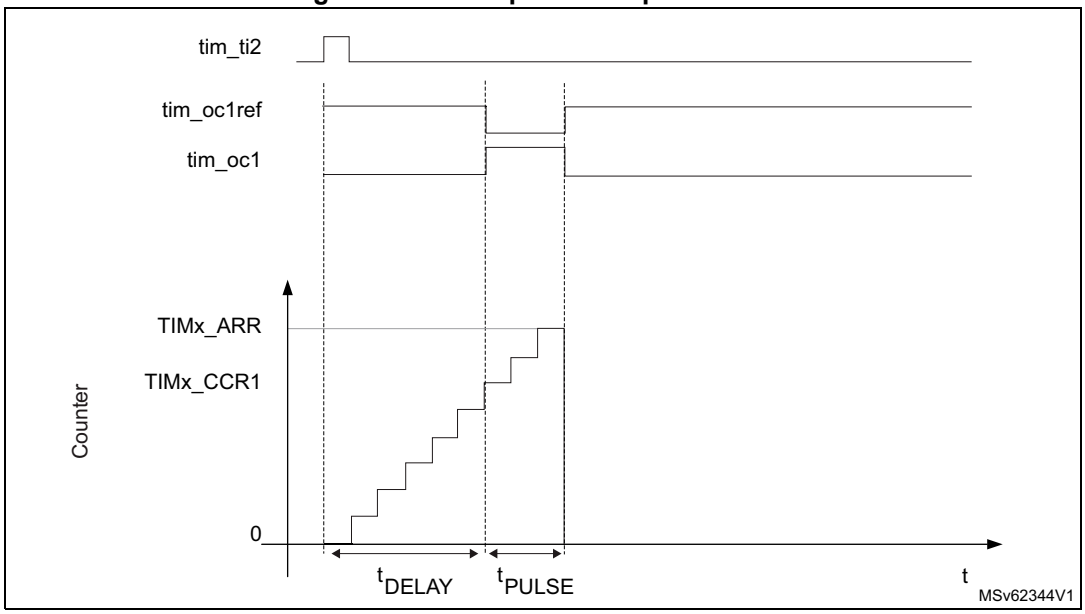
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

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 581. Example of one pulse mode.



For example one may want to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Let's use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:1] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map tim_ti2fp2 to tim_ti2 by writing CC2S='01' in the TIMx_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS='00110' in the TIMx_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to '0' in this example.

Since only 1 pulse is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0).

Particular case: tim_ocx fast enable

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

48.4.20 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 48.4.19*:

- 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

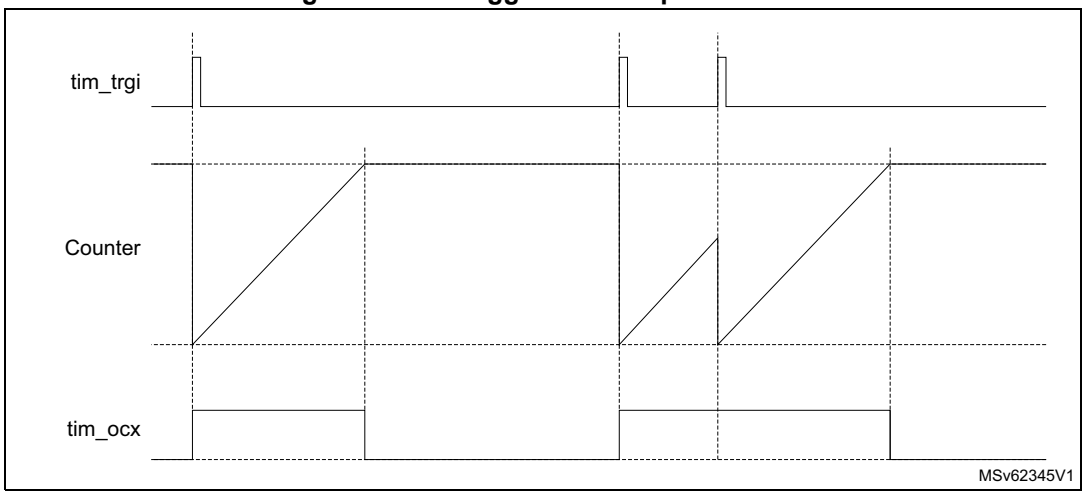
The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for Retrigerrable OPM mode 1 or 2.

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 582. Retriggerable one pulse mode



48.4.21 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 is used 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).

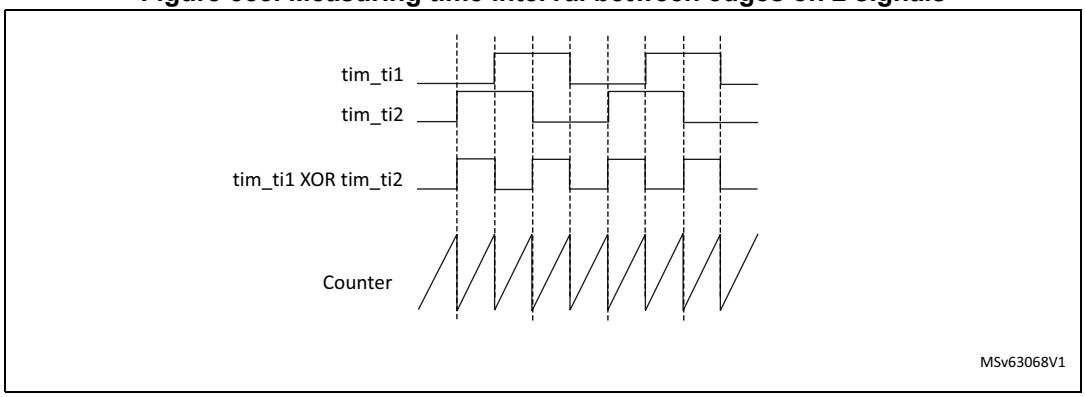
There is no latency between the assertions of the UIF and UIFCPY flags.

48.4.22 Timer input XOR function (TIM15 only)

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins tim_ti1 and tim_ti2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in *Figure 583*.

Figure 583. Measuring time interval between edges on 2 signals



48.4.23 External trigger synchronization (TIM15 only)

The TIM timers are linked together internally for timer synchronization or chaining.

The TIM15 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

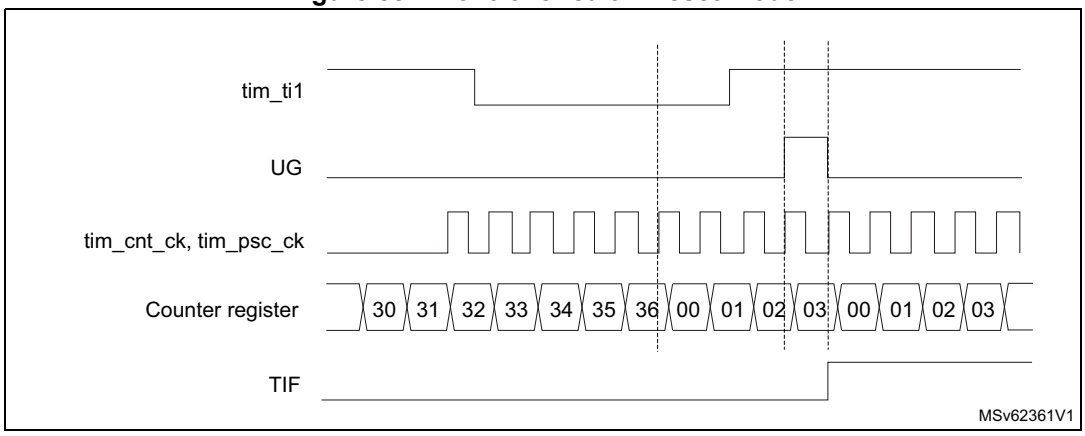
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P='0' and CC1NP='0' in the TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 584. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

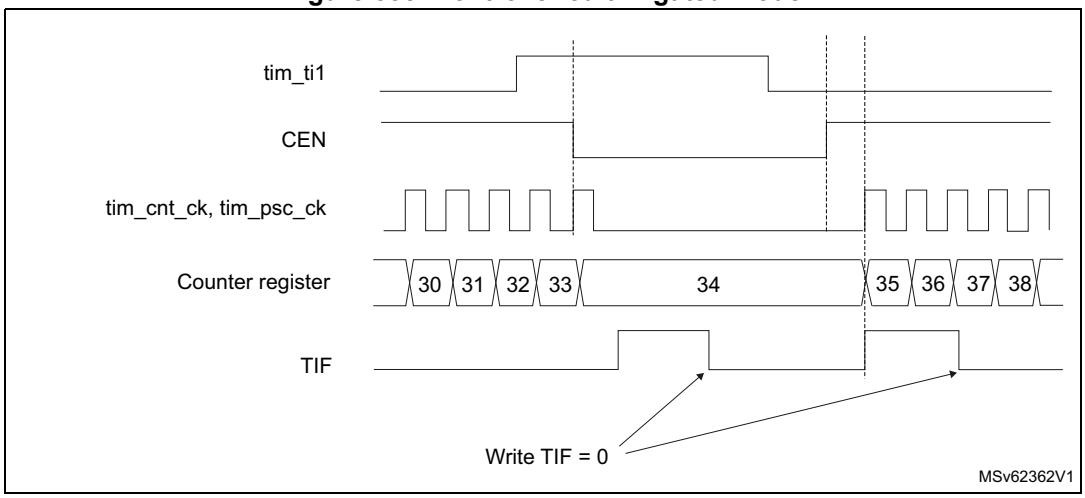
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP = '0' in the TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 585. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

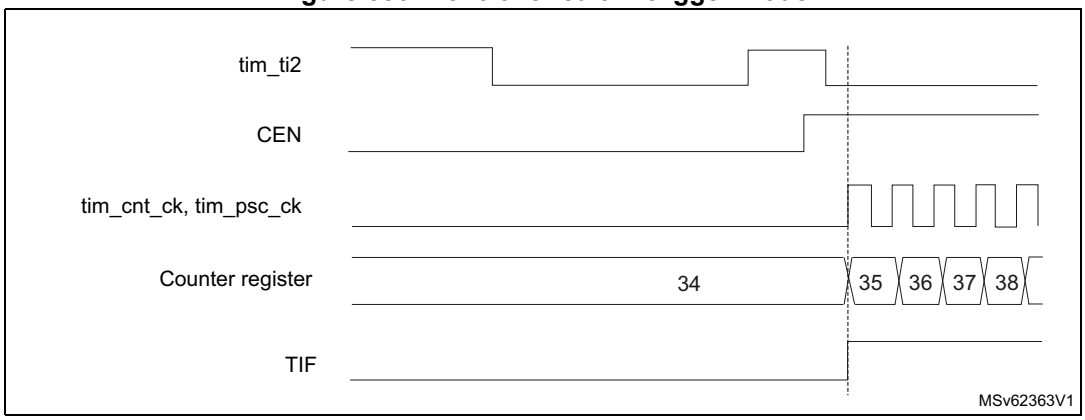
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P='1' and CC2NP='0' in the TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS=110 in the TIMx_SMCR register. Select tim_ti2 as the input source by writing TS=00110 in the TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 586. Control circuit in trigger mode



Slave mode selection preload for run-time encoder mode update

The SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value is the update event (UEV) occurring when the counter overflows.

48.4.24 Slave mode – combined reset + trigger mode (TIM15 only)

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

48.4.25 Slave mode – combined reset + gated mode (TIM15 only)

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

48.4.26 Timer synchronization (TIM15 only)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 47.4.23: Timer synchronization* for details.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

48.4.27 Using timer output as trigger for other timers (TIM16/TIM17 only)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any timer on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer detects the trigger.

For instance, if the destination's timer CK_INT clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

48.4.28 ADC triggers (TIM15 only)

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

48.4.29 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,

The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 48.8.20: TIMx DMA control register (TIMx_DCR)(x = 16 to 17)* for details).

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 and DBSS = 1.
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.

48.4.30 TIM15/TIM16/TIM17 DMA requests

The TIM15/TIM16/TIM17 can generate a DMA requests, as shown in *Table 476*.

Table 476. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_com_dma(1)	Commutation (COM)	COMDE
tim_trg_dma(1)	Trigger	TDE

1. Available for TIM15 only.

48.4.31 Debug mode

When the microcontroller enters debug mode (Cortex®-M33 core halted), the TIMx counter can either continue to work normally or stop.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

For more details, refer to the debug section.

48.5 TIM15/TIM16/TIM17 low-power modes

Table 477. Effect of low-power modes on TIM15/TIM16/TIM17

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

48.6 TIM15/TIM16/TIM17 interrupts

The TIM15/TIM16/TIM17 can generate multiple interrupts, as shown in *Table 478*.

Table 478. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM	Update	UIF	UIE	write 0 in UIF	Yes	No
	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
	Capture/compare 2(1)	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Commutation (COM)	COMIF	COMIE	write 0 in COMIF	Yes	No
	Trigger(1)	TIF	TIE	write 0 in TIF	Yes	No
	Break	BIF	BIE	write 0 in BIF	Yes	No

1. Available for TIM15 only.

48.7 TIM15 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

48.7.1 TIM15 control register 1 (TIM15_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 DITHEN: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIM15_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM15_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (tim_tix)
00: $t_{DTS} = t_{tim\_ker\_ck}$
01: $t_{DTS} = 2*t_{tim\_ker\_ck}$
10: $t_{DTS} = 4*t_{tim\_ker\_ck}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIM15_ARR register is not buffered
1: TIM15_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt if enabled. These events can be:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt if enabled

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

48.7.2 TIM15 control register 2 (TIM15_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
					rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

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

Bit 10 **OIS2:** Output idle state 2 (tim_oc2 output)

0: tim_oc2=0 when MOE=0

1: tim_oc2=1 when MOE=0

Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the TIM15_BKR register).

Bit 9 **OIS1N**: Output Idle state 1 (tim_oc1n output)

0: tim_oc1n=0 after a dead-time when MOE=0

1: tim_oc1n=1 after a dead-time when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (tim_oc1 output)

0: tim_oc1=0 after a dead-time when MOE=0

1: tim_oc1=1 after a dead-time when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BKR register).

Bit 7 **TI1S**: tim_ti1 selection

0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input

1: The tim_ti1_in[15:0] and tim_ti2_in[15:0] multiplexers output are connected to the tim_ti1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

000: **Reset** - the UG bit from the TIM15_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIM15_SMCR register).

010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo).

101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo).

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on tim_trgi.

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on tim_trgi, depending on the CCUS bit).

Note: This bit acts only on channels that have a complementary output.

48.7.3 TIM15 slave mode control register (TIM15_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
										rw	rw				rw

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

Bits 31:22 Reserved, must be kept at reset value.

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

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

Bit 7 **MSM:** Master/slave mode

0: No action

1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection

This bit field selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (tim_itr0)
00001: Internal Trigger 1 (tim_itr1)
00010: Internal Trigger 2 (tim_itr2)
00011: Internal Trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: Reserved
01000: Internal Trigger 4 (tim_itr4)
01001: Internal Trigger 5 (tim_itr5)
01010: Internal Trigger 6 (tim_itr6)
01011: Internal Trigger 7 (tim_itr7)
01100: Internal Trigger 8 (tim_itr8)
01101: Internal Trigger 9 (tim_itr9)
01110: Internal Trigger 10 (tim_itr10)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
Others: Reserved

See *Section 48.4.2: TIM15/TIM16/TIM17 pins and internal signals* for more details on tim_itrx meaning for each timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (see Input Control register and Control Register description.

0000: Slave mode disabled - if CEN = ‘1’ then the prescaler is clocked directly by the internal clock.

0001: Reserved

0010: Reserved

0011: Reserved

0100: Reset Mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

Others: Reserved.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS=’00100’). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

48.7.4 TIM15 DMA/interrupt enable register (TIM15_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	Res.	Res.	Res.	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	rw	rw				rw	rw	rw	rw	rw			rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable

0: Trigger DMA request disabled

1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable

0: COM DMA request disabled

1: COM DMA request enabled

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

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

48.7.5 TIM15 status register (TIM15_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
					rc_w0	rc_w0		rc_w0	rc_w0	rc_w0			rc_w0	rc_w0	rc_w0

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

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected
1: The counter value has been captured in TIM15_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred

1: An active level has been detected on the break input

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred

1: Trigger interrupt pending

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.

0: No COM event occurred

1: COM interrupt pending

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred

If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in downcounting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIM15_CR1 register.

– When CNT is reinitialized by software using the UG bit in TIM15_EGR register, if URS=0 and UDIS=0 in the TIM15_CR1 register.

– When CNT is reinitialized by a trigger event (refer to *Section 48.7.3: TIM15 slave mode control register (TIM15_SMCR)*), if URS=0 and UDIS=0 in the TIM15_CR1 register.

48.7.6 TIM15 event generation register (TIM15_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
								w	w	rw			w	w	w

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

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIM15_SR register. Related interrupt or DMA transfer can occur if enabled

Bit 5 **COMG:** Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/Compare 2 generation

Refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1 A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIM15_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

48.7.7 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2
10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1
11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIM15_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E='0' (TIM15_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIM15_CCER).

48.7.8 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2.

10: C2 channel is configured as input, tim_ic2 is mapped on tim_ti1.

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIM15_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIM15_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr_int input.

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input.

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIM15_CCR1 and the counter TIM15_CNT has no effect on the outputs.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIM15_CNT matches the capture/compare register 1 (TIM15_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIM15_CNT matches the capture/compare register 1 (TIM15_CCR1).

0011: Toggle - tim_oc1ref toggles when TIM15_CNT=TIM15_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIM15_CNT<TIM15_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIM15_CNT<TIM15_CCR1 else active.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved

1011: Reserved

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Reserved,

1111: Reserved,

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S='00' (the channel is configured in output).

In PWM mode, the tim_ocxref level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.

On channels that have a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIM15_CCR1 disabled. TIM15_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM15_CCR1 enabled. Read/Write operations access the preload register. TIM15_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S='00' (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, tim_ocx is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2.

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIM15_CCER).

48.7.9 TIM15 capture/compare enable register (TIM15_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
								rw		rw	rw	rw	rw	rw	rw

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

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: tim_oc1n active high

1: tim_oc1n active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1 and tim_ti2fp1. Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S="00" (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

CC1 channel configured as output:

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP=1, CC1P=0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 479* for details.

Table 479. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15)

Control bits					Output states(1)	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	tim_ocx output state	tim_ocxn output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) tim_ocx=0 tim_ocxn=0	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) tim_ocx=0	tim_ocxref + Polarity tim_ocxn=tim_ocxref XOR CCxNP
		0	1	0	tim_ocxref + Polarity tim_ocx=tim_ocxref XOR CCxP	Output Disabled (not driven by the timer: Hi-Z) tim_ocxn=0
		X	1	1	tim_ocxref + Polarity + dead-time	Complementary to tim_ocxref (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) tim_ocx=CCxP	tim_ocxref + Polarity tim_ocxn=tim_ocxref XOR CCxNP
		1	1	0	tim_ocxref + Polarity tim_ocx=tim_ocxref xor CCxP	Off-State (output enabled with inactive state) tim_ocxn=CCxNP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z)	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: tim_ocx=CCxP, tim_ocxn=CCxNP Then if the clock is present: tim_ocx=OISx and tim_ocxn=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to tim_ocx and tim_ocxn both in active state	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and GPIO control and alternate function selection registers.

48.7.10 TIM15 counter (TIM15_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIM15_ISR register.

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

Bits 15:0 **CNT[15:0]**: Counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

48.7.11 TIM15 prescaler (TIM15_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency ($f_{tim\_cnt\_ck}$) is equal to $f_{tim\_psc\_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIM15_EGR register or through trigger controller when configured in “reset mode”).

48.7.12 TIM15 auto-reload register (TIM15_ARR)

Address offset: 0x2C

Reset value: 0x0000 FFFF

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

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

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 48.4.3: Time-base unit on page 1892* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

48.7.13 TIM15 repetition counter register (TIM15_RCR)

Address offset: 0x30

Reset value: 0x0000

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

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

Bits 7:0 **REP[7:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the reptition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIM15_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode:

- the number of PWM periods in edge-aligned mode
- the number of half PWM period in center-aligned mode

48.7.14 TIM15 capture/compare register 1 (TIM15_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM15_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM15_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

48.7.15 TIM15 capture/compare register 2 (TIM15_CCR2)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM15_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM15_CNT and signalled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 1 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

48.7.16 TIM15 break and dead-time register (TIM15_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BK DSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
			rw		rw							rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIM15_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break bidirectional

0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break disarm

0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter

This bit-field defines the frequency used to sample the tim_brk input signal and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:
0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the tim_brk input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: tim_ocx and tim_ocxn outputs are disabled or forced to idle state depending on the OSSI bit.

1: tim_ocx and tim_ocxn outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIM15_CCER register)

See tim_ocx/tim_ocxn enable description for more details (*Section 48.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 1945*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low

1: Break input tim_brk is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (tim_brk and tim_sys_brk clock failure event) disabled

1; Break inputs (tim_brk and tim_sys_brk clock failure event) enabled

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See tim_ocx/tim_ocxn enable description for more details (*Section 48.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 1945*).

0: When inactive, tim_ocx/tim_ocxn outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)

1: When inactive, tim_ocx/tim_ocxn outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 on channels configured as outputs.

See tim_ocx/tim_ocxn enable description for more details (*Section 48.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 1945*).

0: When inactive, tim_ocx/tim_ocxn outputs are disabled (tim_ocx/tim_ocxn enable output signal=0)

1: When inactive, tim_ocx/tim_ocxn outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. tim_ocx/tim_ocxn enable output signal=1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM15_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIM15_BDTR register, OISx and OISxN bits in TIM15_CR2 register and BKBID/BKE/BKP/AOE bits in TIM15_BDTR register can no longer be written

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIM15_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: OCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIM15_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIM15_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5]=0xx => DT=DTG[7:0]x t_{dtg} with $t_{dtg}=t_{DTS}$

DTG[7:5]=10x => DT=(64+DTG[5:0])xt_{dtg} with $T_{dtg}=2xt_{DTS}$

DTG[7:5]=110 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=8xt_{DTS}$

DTG[7:5]=111 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=16xt_{DTS}$

Example if T_{DTS}=125ns (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

48.7.17 TIM15 timer deadtime register 2 (TIM15_DTR2)

Address offset: 0x054

Reset value: 0x0000 0000

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

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

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

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

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5]=0xx => DTF=DTGF[7:0]x t_{dtg} with $t_{dtg}=t_{DTS}$.

DTGF[7:5]=10x => DTF=(64+DTGF[5:0])xt_{dtg} with $T_{dtg}=2xt_{DTS}$.

DTGF[7:5]=110 => DTF=(32+DTGF[4:0])xt_{dtg} with $T_{dtg}=8xt_{DTS}$.

DTGF[7:5]=111 => DTF=(32+DTGF[4:0])xt_{dtg} with $T_{dtg}=16xt_{DTS}$.

Example if T_{DTS}=125ns (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 us to 31750 ns by 250 ns steps,

32 us to 63us by 1 us steps,

64 us to 126 us by 2 us steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

48.7.18 TIM15 input selection register (TIM15_TISEL)

Address offset: 0x5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects tim_ti2_in[15:0] input

0000: TIM15_CH2 input (tim_ti2_in0)

0001: tim_ti2_in1

...

1111: tim_ti2_in15

Refer to *Section 48.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

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

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input

0000: TIM15_CH1 input (tim_ti1_in0)

0001: tim_ti1_in1

...

1111: tim_ti1_in15

Refer to *Section 48.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

48.7.19 TIM15 alternate function register 1 (TIM15_AF1)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	BK CMP8E	BK CMP7E	BK CMP6E	BK CMP5E	BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Refer to *Section 48.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity

This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp4 input is active high

1: tim_brk_cmp4 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input is active high

1: tim_brk_cmp3 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input is active high

1: tim_brk_cmp2 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input is active high

1: tim_brk_cmp1 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input is active high

1: TIMx_BKIN input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. mdf_brkx output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. COMP7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

48.7.20 TIM15 alternate function register 2 (TIM15_AF2)

Address offset: 0x064

Reset value: 0x0000 0000

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

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

Bits 18:16 **OCRSEL[2:0]**: ocref_clr source selection

These bits select the ocref_clr input source.

000: tim_ocref_clr0

001: tim_ocref_clr1

010: tim_ocref_clr2

011: tim_ocref_clr3

100: tim_ocref_clr4

101: tim_ocref_clr5

110: tim_ocref_clr6

111: tim_ocref_clr7

Refer to *Section 48.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bits 15:0 Reserved, must be kept at reset value.

48.7.21 TIM15 DMA control register (TIM15_DCR)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[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.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

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

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

0000: Reserved

0001: Update

0010: CC1

0110: COM

0111: Trigger

Other: reserved

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 TIM15_DMAR address).

00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIM15_DMAR address). DBA is defined as an offset starting from the address of the TIM15_CR1 register.

Example:

00000: TIM15_CR1,
00001: TIM15_CR2,
00010: TIM15_SMCR,
...

48.7.22 TIM15 DMA address for full transfer (TIM15_DMAR)

Address offset: 0x3E0

Reset value: 0x0000 0000

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

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIM15_CR1 address) + (DBA + DMA index) x 4

where TIM15_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM15_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM15_DCR).

48.7.23 TIM15 register map

TIM15 registers are mapped as 16-bit addressable registers as described in the table below:

Table 480. TIM15 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIM15_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0
0x04	TIM15_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
	Reset value																						0	0	0	0	0	0	0	0	0		0
0x08	TIM15_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value											0	0				0									0	0	0	0		0	0	0
0x0C	TIM15_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	Res.	Res.	Res.	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	Reset value																		0	0				0	0	0	0	0			0	0	0
0x10	TIM15_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
	Reset value																						0	0		0	0	0			0	0	0
0x14	TIM15_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
	Reset value																									0	0	0			0	0	0

Table 480. 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
0x18	**TIM15_CCMR1** Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM15_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIM15_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value																									0		0	0	0	0	0	0
0x24	**TIM15_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIM15_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIM15_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	**TIM15_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	**TIM15_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIM15_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 - 0x40	Reserved	Res.																															
0x44	**TIM15_BDTR**	Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value				0		0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48 - 0x50	Reserved	Res.																															
0x54	**TIM15_DTR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x58	Reserved	Res.																															

Table 480. 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
0x5C	**TIM15_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																					0	0	0	0					0	0	0	0
0x60	**TIM15_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x64	**TIM15_AF2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCR SEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0																
0x68 - 0x3D8	Reserved	Res.																															
0x3DC	**TIM15_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	**TIM15_DMAR**	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3 on page 115* for the register boundary addresses.

48.8 TIM16/TIM17 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

48.8.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bit-field indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (tim_tix),
00: $t_{DTS}=t_{tim\_ker\_ck}$
01: $t_{DTS}=2*t_{tim\_ker\_ck}$
10: $t_{DTS}=4*t_{tim\_ker\_ck}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
1: nly counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

48.8.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
						rw	rw					rw	rw		rw

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

Bit 9 **OIS1N**: Output Idle state 1 (tim_oc1n output)

0: tim_oc1n=0 after a dead-time when MOE=0

1: tim_oc1n=1 after a dead-time when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (tim_oc1 output)

0: tim_oc1=0 after a dead-time when MOE=0

1: tim_oc1=1 after a dead-time when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

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

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when a rising edge occurs on tim_trgi (if available).

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.
Note: This bit acts only on channels that have a complementary output.

48.8.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17)

Address offset: 0x0C

Reset value: 0x0000

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

48.8.4 TIMx status register (TIMx_SR)(x = 16 to 17)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
						rc_w0		rc_w0		rc_w0				rc_w0	rc_w0

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

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.

0: No overcapture has been detected

1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the tim_brk input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred

1: An active level has been detected on the break input

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.

0: No COM event occurred

1: COM interrupt pending

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred

If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

48.8.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
								w		w				w	w

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

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMG:** Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

48.8.6 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

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

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (tim_ic1).
The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input.
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

48.8.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

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

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

Bit 7 **OC1CE**: Output Compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr input.

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr input.

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

In PWM mode 1 or 2, the tim_oc1ref level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, tim_ocx is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

48.8.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: tim_oc1n active high

1: tim_oc1n active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1. Refer to the description of CC1P.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=”00” (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP=1, CC1P=0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 481* for details.

Table 481. Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)

Control bits					Output states(1)	
MOE bit	OSSI bit	OSSR bit	CC1E bit	CC1NE bit	tim_oc1 output state	tim_oc1n output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) tim_oc1=0 tim_oc1n=0	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) tim_oc1=0	tim_oc1ref + Polarity tim_oc1n=tim_oc1ref XOR CC1NP
		0	1	0	tim_oc1ref + Polarity tim_oc1=tim_oc1ref XOR CC1P	Output Disabled (not driven by the timer: Hi-Z) tim_oc1n=0
		X	1	1	tim_oc1ref + Polarity + dead-time	Complementary to tim_oc1ref (not tim_oc1ref) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) tim_oc1=CC1P	tim_oc1ref + Polarity tim_oc1n=tim_oc1ref XOR CC1NP
		1	1	0	tim_oc1ref + Polarity tim_oc1=tim_oc1ref XOR CC1P	Off-State (output enabled with inactive state) tim_oc1n=CC1NP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z)	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: tim_oc1=CC1P, tim_oc1n=CC1NP Then if the clock is present: tim_oc1=OIS1 and tim_oc1n=OIS1N after a dead-time, assuming that OIS1 and OIS1N do not correspond to tim_oc1 and tim_oc1n both in active state	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OIS1, OIS1N, CC1P and CC1NP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_oc1 and tim_oc1n channels depends on the tim_oc1 and tim_oc1n channel state and GPIO control and alternate function selection registers.

48.8.9 TIMx counter (TIMx_CNT)(x = 16 to 17)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved.

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

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

48.8.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (tim_cnt_ck) is equal to $f_{tim\_psc\_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

48.8.11 TIMx auto-reload register (TIMx_ARR)(x = 16 to 17)

Address offset: 0x2C

Reset value: 0x0000 FFFF

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

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

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 48.4.3: Time-base unit on page 1892* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

48.8.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17)

Address offset: 0x30

Reset value: 0x0000

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

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

Bits 7:0 **REP[7:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode:

- the number of PWM periods in edge-aligned mode
- the number of half PWM period in center-aligned mode

48.8.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 19:0 **CCR1[19:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.
If channel CC1 is configured as input:
CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1).
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

48.8.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BK DSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
			rw		rw							rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break Bidirectional

0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break Disarm

0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter

This bit-field defines the frequency used to sample tim_brk input and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=2
0010: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=4
0011: $f_{SAMPLING}=f_{tim\_ker\_ck}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the tim_brk input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: tim_oc1 and tim_oc1n outputs are disabled or forced to idle state depending on the OSSI bit.

1: tim_oc1 and tim_oc1n outputs are enabled if their respective enable bits are set (CC1E, CC1NE in TIMx_CCER register)

See tim_oc1/tim_oc1n enable description for more details (*Section 48.8.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 1971*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the tim_brk input is not active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low
1: Break input tim_brk is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (tim_brk and tim_sys_brk event) disabled
1; Break inputs (tim_brk and tim_sys_brk event) enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See tim_oc1/tim_oc1n enable description for more details (*Section 48.8.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 1971*).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)

1: When inactive, tim_oc1/tim_oc1n outputs are enabled with their inactive level as soon as CC1E=1 or CC1NE=1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 on channels configured as outputs.

See tim_oc1/tim_oc1n enable description for more details (*Section 48.8.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 1971*).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (tim_oc1/tim_oc1n enable output signal=0)

1: When inactive, tim_oc1/tim_oc1n outputs are forced first with their idle level as soon as CC1E=1 or CC1NE=1. tim_oc1/tim_oc1n enable output signal=1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKBID/BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5]=0xx => DT=DTG[7:0]x t_{dtg} with $t_{dtg}=t_{DTS}$

DTG[7:5]=10x => DT=(64+DTG[5:0])xt_{dtg} with $T_{dtg}=2xt_{DTS}$

DTG[7:5]=110 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=8xt_{DTS}$

DTG[7:5]=111 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=16xt_{DTS}$

Example if T_{DTS}=125ns (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

48.8.15 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 16 to 17)

Address offset: 0x054

Reset value: 0x0000 0000

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

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

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

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5]=0xx => DTF=DTGF[7:0]x t_{dtg} with $t_{dtg}=t_{DTS}$.

DTGF[7:5]=10x => DTF=(64+DTGF[5:0])xt_{dtg} with $T_{dtg}=2xt_{DTS}$.

DTGF[7:5]=110 => DTF=(32+DTGF[4:0])xt_{dtg} with $T_{dtg}=8xt_{DTS}$.

DTGF[7:5]=111 => DTF=(32+DTGF[4:0])xt_{dtg} with $T_{dtg}=16xt_{DTS}$.

Example if T_{DTS}=125ns (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 us to 31750 ns by 250 ns steps,

32 us to 63us by 1 us steps,

64 us to 126 us by 2 us steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

48.8.16 TIMx input selection register (TIMx_TISEL)(x = 16 to 17)

Address offset: 0x5C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input
0000: TIMx_CH1 input (tim_ti1_in0)
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 48.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

48.8.17 TIMx alternate function register 1 (TIMx_AF1)(x = 16 to 17)

Address offset: 0x060

Reset value: 0x0000 0001

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	BK CMP8E	BK CMP7E	BK CMP6E	BK CMP5E	BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Refer to *Section 48.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity
This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp4 input is active high
1: tim_brk_cmp4 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input is active high

1: tim_brk_cmp3 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input is active high

1: tim_brk_cmp2 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input is active high

1: tim_brk_cmp1 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input is active high

1: TIMx_BKIN input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. mdf_brkx output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

48.8.18 TIMx alternate function register 2 (TIMx_AF2)(x = 16 to 17)

Address offset: 0x064

Reset value: 0x0000 0000

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

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

Bits 18:16 **OCRSEL[2:0]**: tim_ocref_clr source selection

These bits select the tim_ocref_clr input source.

000: tim_ocref_clr0

001: tim_ocref_clr1

010: tim_ocref_clr2

011: tim_ocref_clr3

100: tim_ocref_clr4

101: tim_ocref_clr5

110: tim_ocref_clr6

111: tim_ocref_clr7

Refer to *Section 48.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:0 Reserved, must be kept at reset value.

48.8.19 TIMx option register 1 (TIMx_OR1)(x = 16 to 17)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSE32 EN	Res.
														rw	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **HSE32EN**: HSE Divided by 32 enable

This bit enables the HSE divider by 32 for the tim_ti1_in3. See *Table 468: Interconnect to the tim_ti1 input multiplexer* for details.

0: HSE divided by 32 disabled

1: HSE divided by 32 enabled

Bit 0 Reserved, must be kept at reset value.

48.8.20 TIMx DMA control register (TIMx_DCR)(x = 16 to 17)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[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.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

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

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

0000: Reserved

0001: Update

0010: CC1

Other: reserved

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.

48.8.21 TIM16/TIM17 DMA address for full transfer (TIMx_DMAR)(x = 16 to 17)

Address offset: 0x3E0

Reset value: 0x0000 0000

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

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

48.8.22 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

Table 482. 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
0x08	Reserved	Res.																															
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** Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
	TIMx_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1CE	OC1M [2:0]				OC1PE	OC1FE	CC1 S [1:0]	
	Reset value																0									0	0	0	0		0	0	0	0
0x1C	Reserved	Res.																															
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

Table 482. 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
0x2C	**TIMx_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	**TIMx_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	**TIMx_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 - 0x40	Reserved	Res.																															
0x44	**TIMx_BDTR**	Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value				0		0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48 - 0x50	Reserved	Res.																															
0x54	**TIMx_DTR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x58	Reserved	Res.																															
0x5C	**TIMx_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																													0	0	0	0
0x60	**TIMx_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x64	**TIMx_AF2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCR SEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0																
0x68	**TIMx_OR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSE32EN	Res.
	Reset value																															0	
0x6C - 0x3D8	Reserved	Res.																															
0x3DC	**TIMx_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	**TIMx_DMAR**	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

49 Basic timers (TIM6/TIM7)

49.1 TIM6/TIM7 introduction

The basic timers TIM6/TIM7 consist in a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used as generic timers for time-base generation.

The basic timer can also be used for triggering the digital-to-analog converter. This is done with the trigger output of the timer.

The timers are completely independent, and do not share any resources.

49.2 TIM6/TIM7 main features

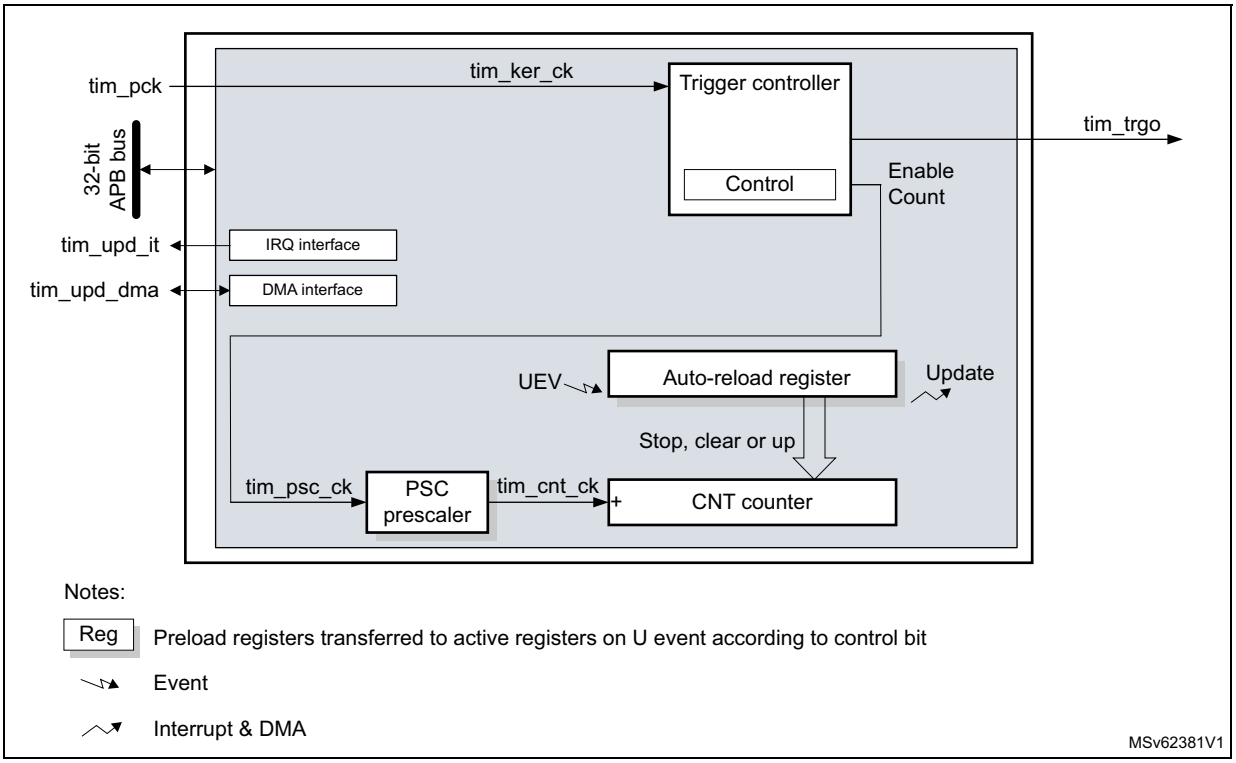
Basic timer (TIM6/TIM7) features include:

- 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

49.3 TIM6/TIM7 functional description

49.3.1 TIM6/TIM7 block diagram

Figure 587. Basic timer block diagram



49.3.2 TIM6/TIM7 internal signals

The table in this section summarizes the TIM inputs and outputs.

Table 483. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 16 (maximum value)
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals (DAC).
tim_upd_it	Output	Timer update event interrupt
tim_upd_dma	Output	Timer update dma request

49.3.3 TIM6/TIM7 clocks

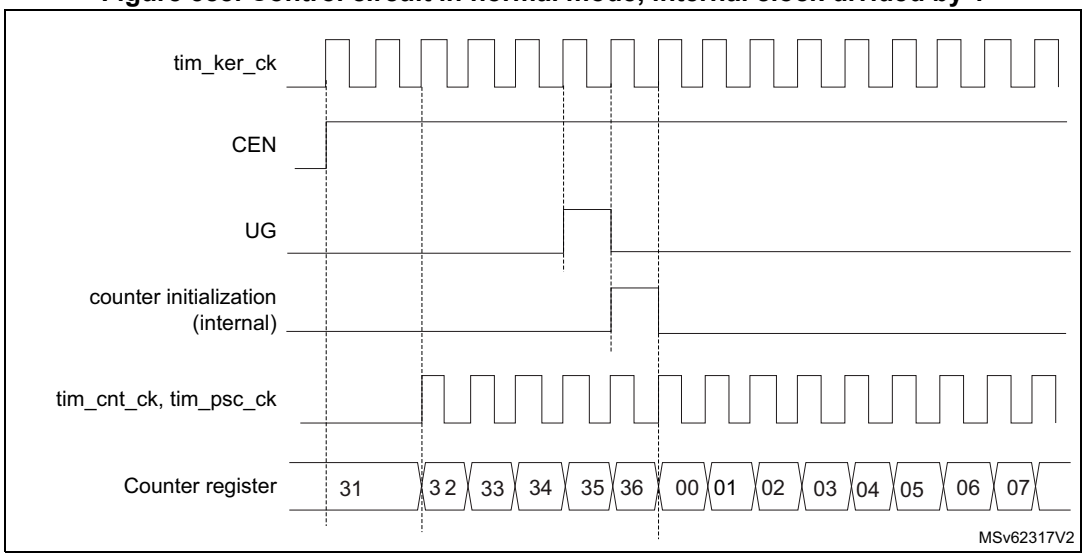
The timer bus interface is clocked by the tim_pclk APB clock.

The counter clock tim_ker_ck is connected to the tim_pclk input.

The CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 588 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 588. Control circuit in normal mode, internal clock divided by 1



49.3.4 Time-base unit

The main block of the programmable timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC)
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. The preload register is accessed each time an attempt is made to write or read the auto-reload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the auto-reload preload enable bit (ARPE) in the TIMx_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in the TIMx_CR1 register is set.

Note that the actual counter enable signal tim_cnt_en is set 1 clock cycle after CEN bit set.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as the TIMx_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 589 and *Figure 590* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 589. Counter timing diagram with prescaler division change from 1 to 2

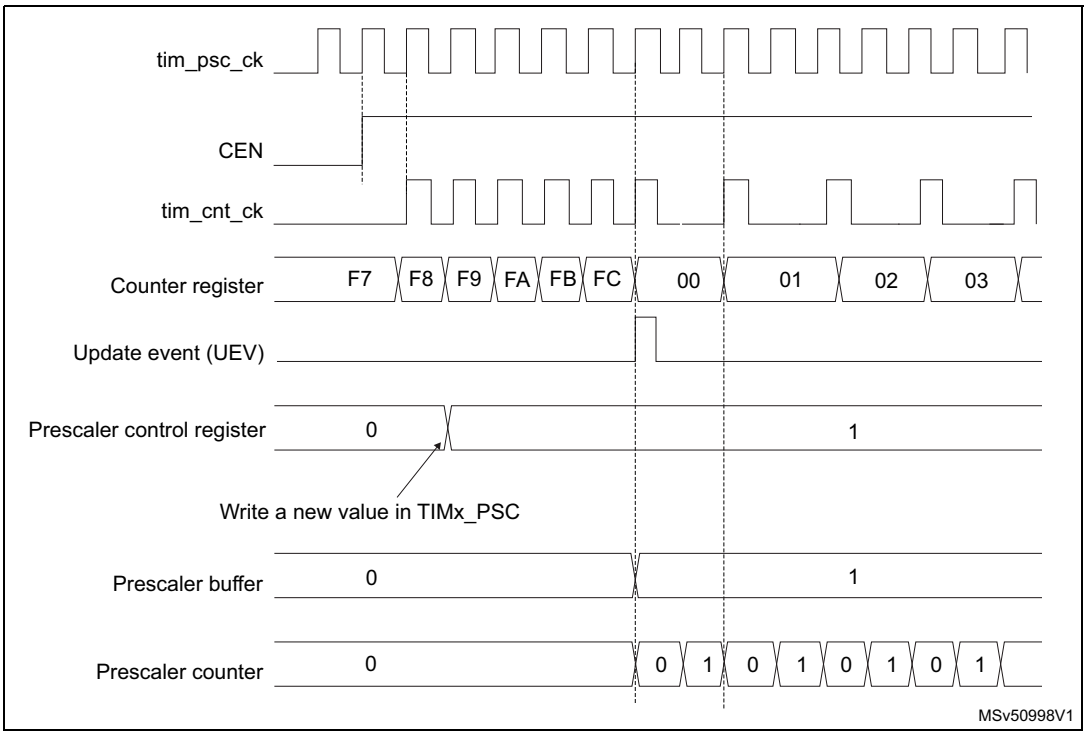
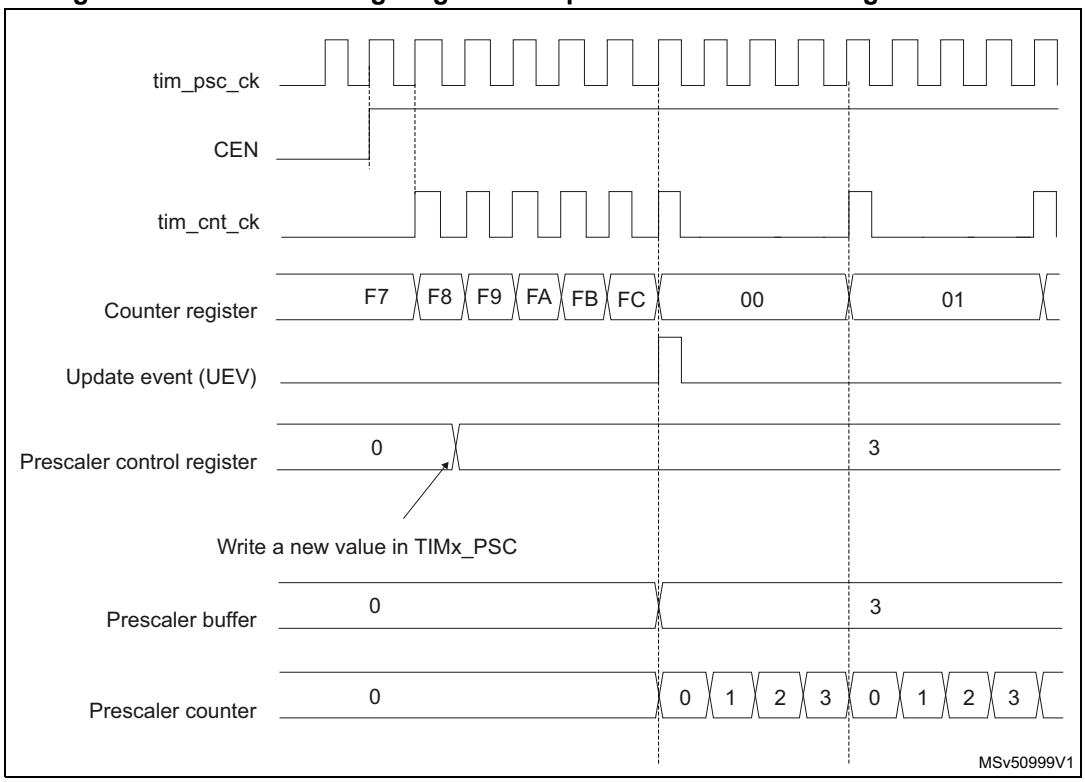


Figure 590. Counter timing diagram with prescaler division change from 1 to 4



49.3.5 Counting mode

The counter counts from 0 to the auto-reload value (contents of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generate at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been written to 0, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIMx_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents of the TIMx_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 591. Counter timing diagram, internal clock divided by 1

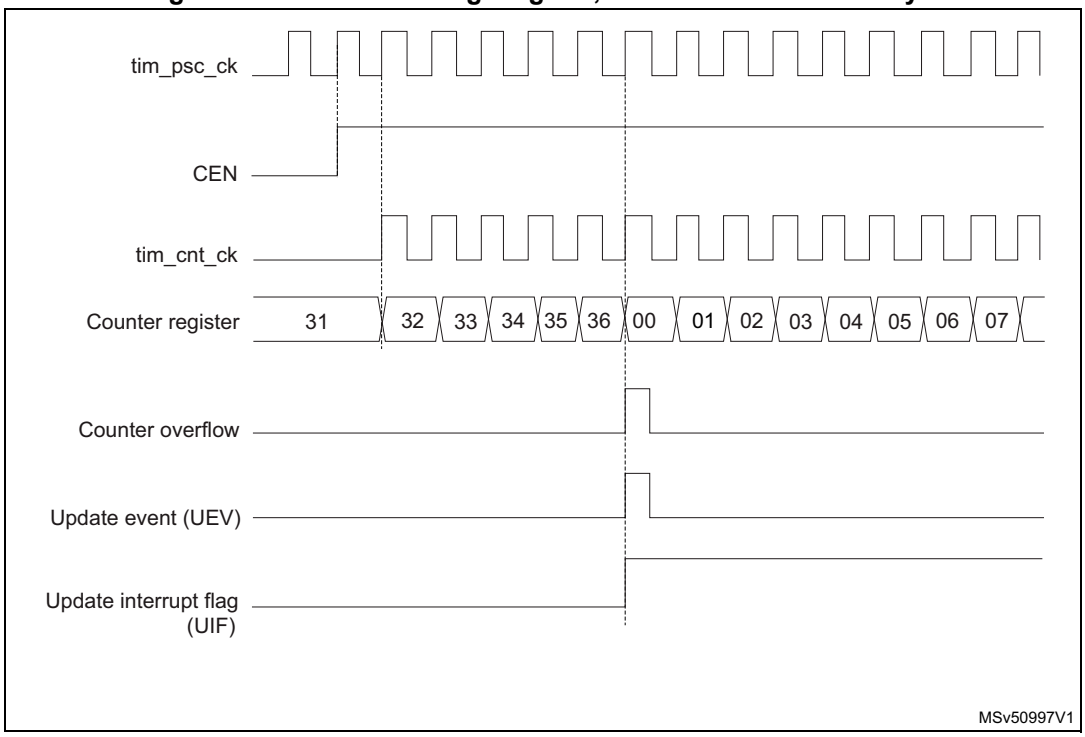


Figure 592. Counter timing diagram, internal clock divided by 2

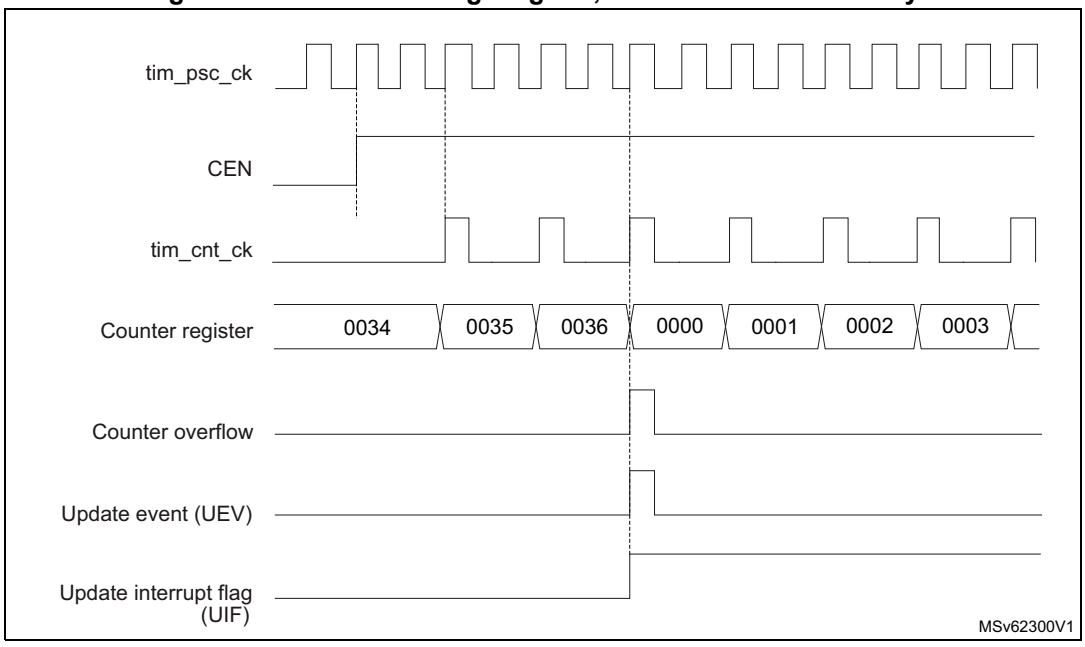


Figure 593. Counter timing diagram, internal clock divided by 4

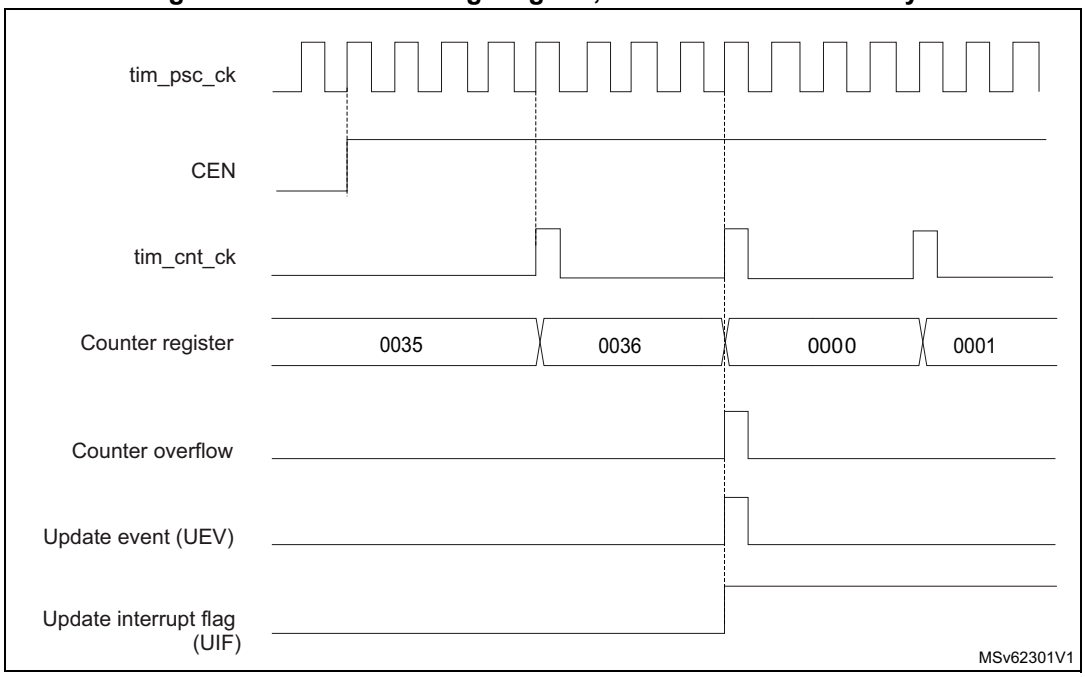


Figure 594. Counter timing diagram, internal clock divided by N

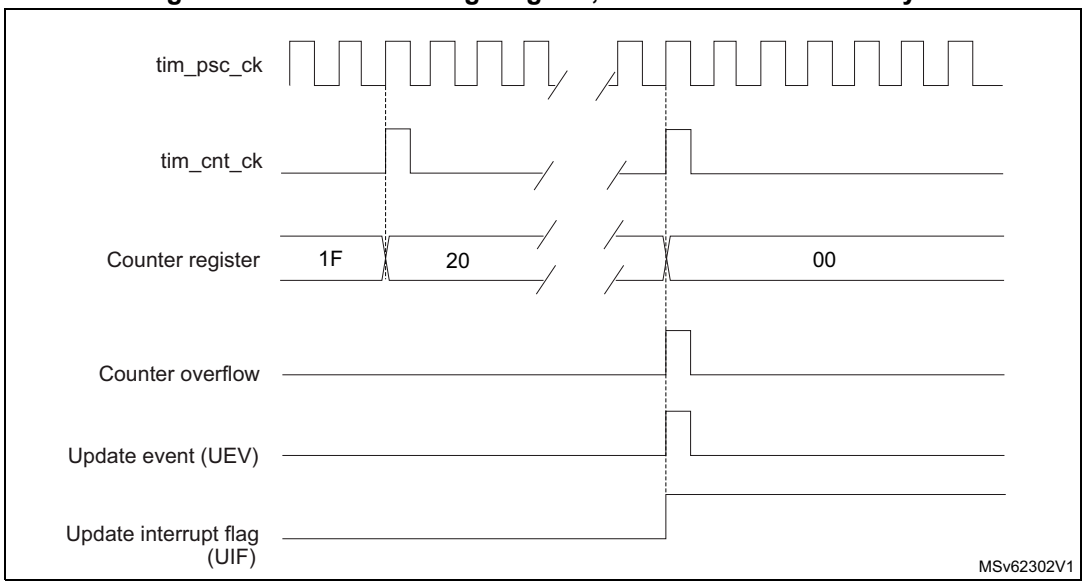


Figure 595. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

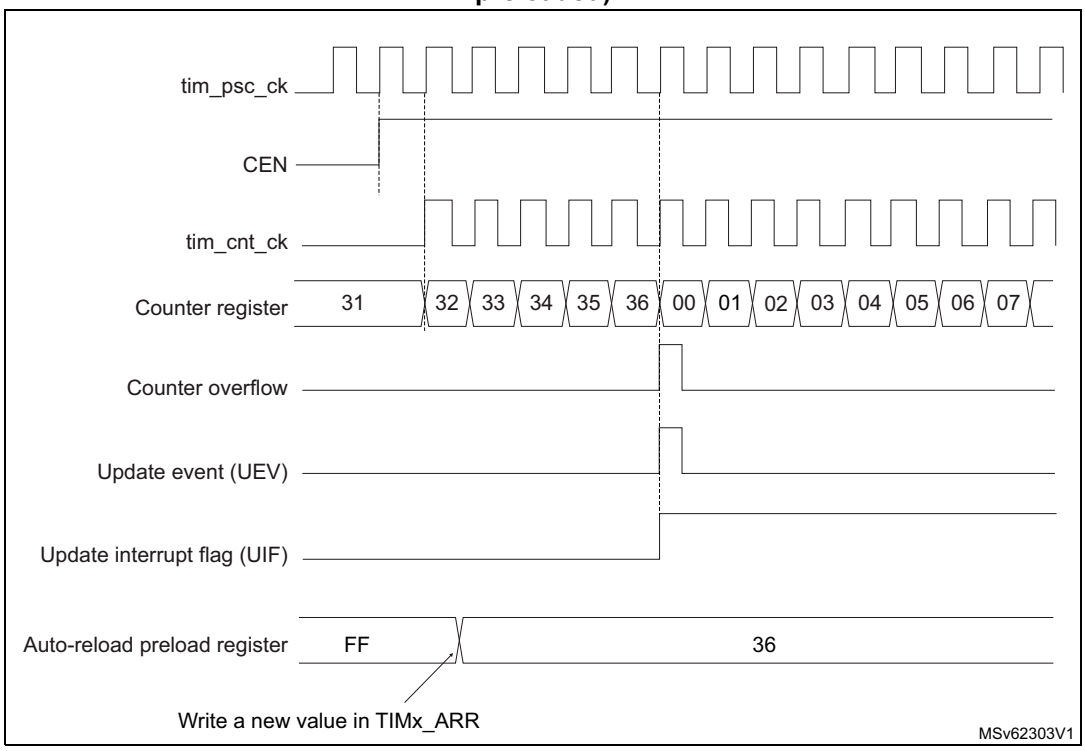
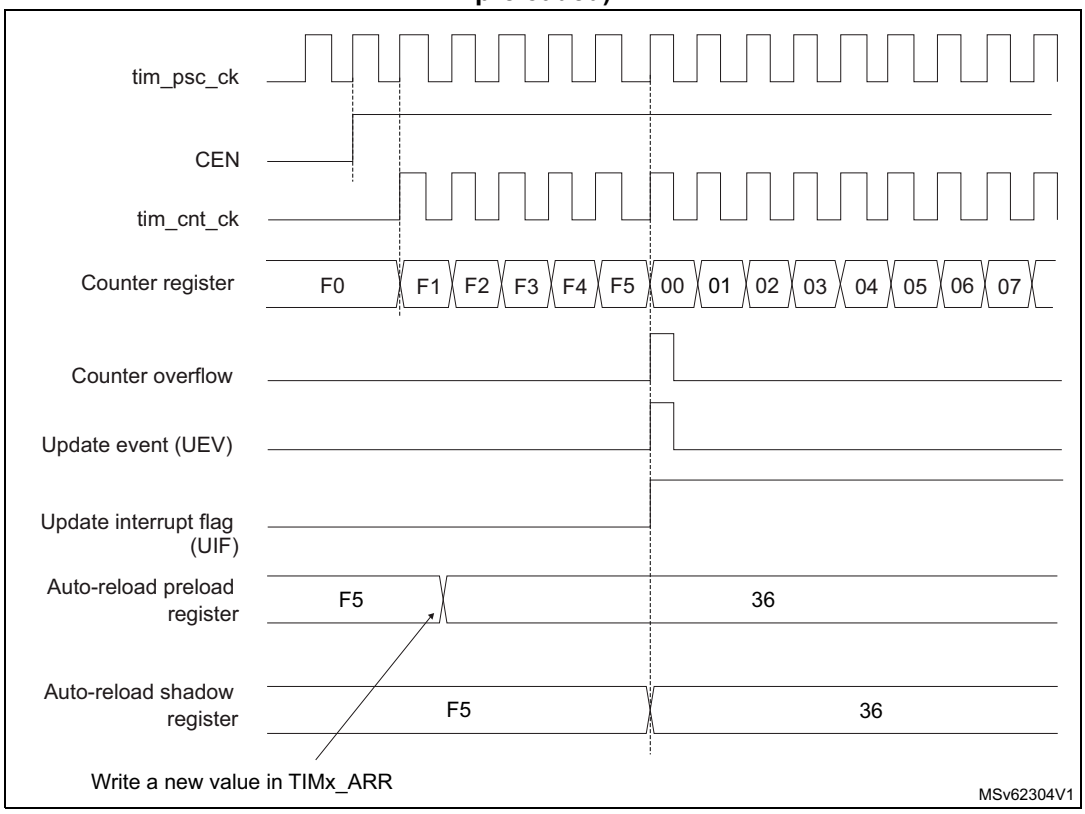


Figure 596. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



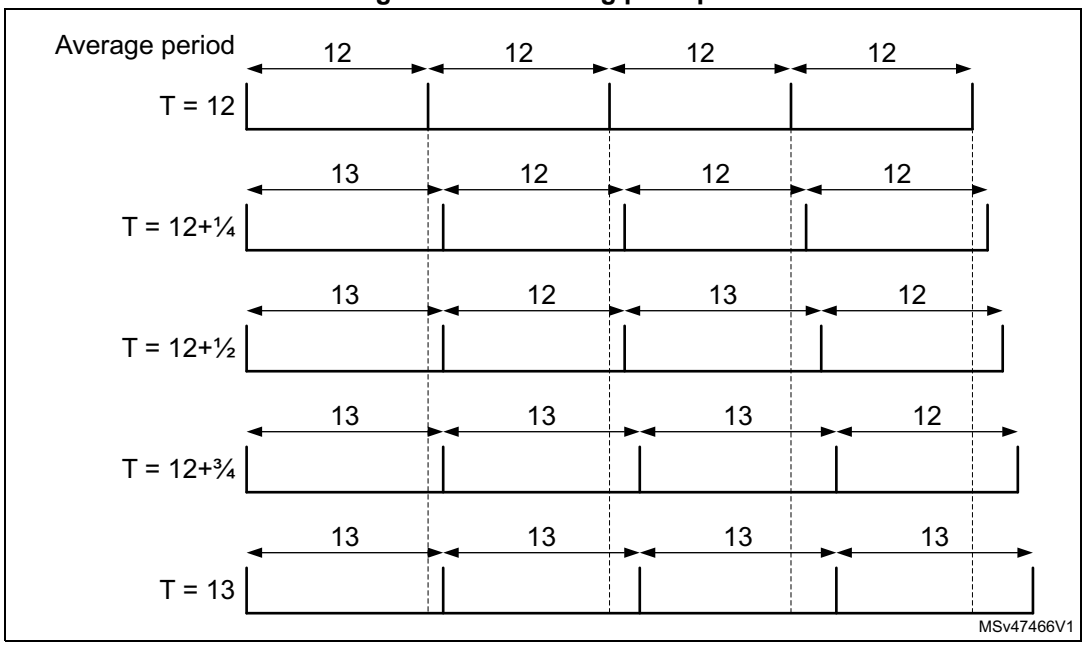
Dithering mode

The time base effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This affects the way the TIMx_ARR is behaving, and is useful for adjusting the average counter period when the timer is used as a trigger (typically for a DAC).

The operating principle is to have the actual ARR value slightly changed (adding or not one timer clock period) over 16 consecutive counting periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average counting period.

The *Figure 597* below presents the dithering principle applied to 4 consecutive counting periods.

Figure 597. Dithering principle

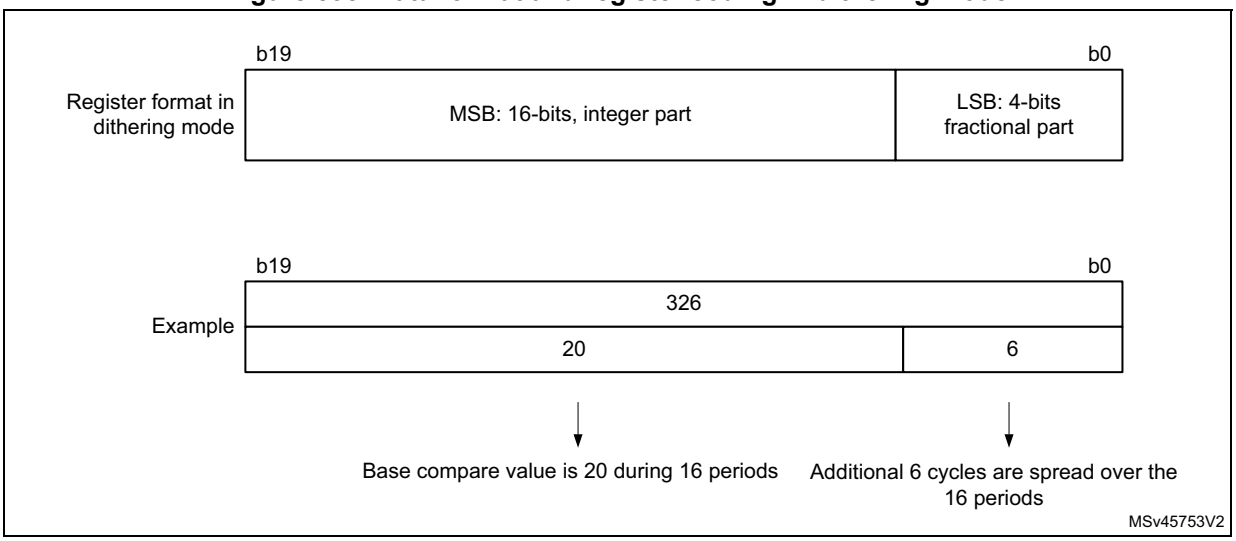


When the dithering mode is enabled, the register coding is changed as following (see *Figure 598* for example):

- the 4 LSBs are coding for the enhanced resolution part (fractional part)
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The ARR[3:0] bits must be reset
3. The DITHEN bit must be reset
4. The CEN bit can be set (eventually with ARPE = 1)

Figure 598. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

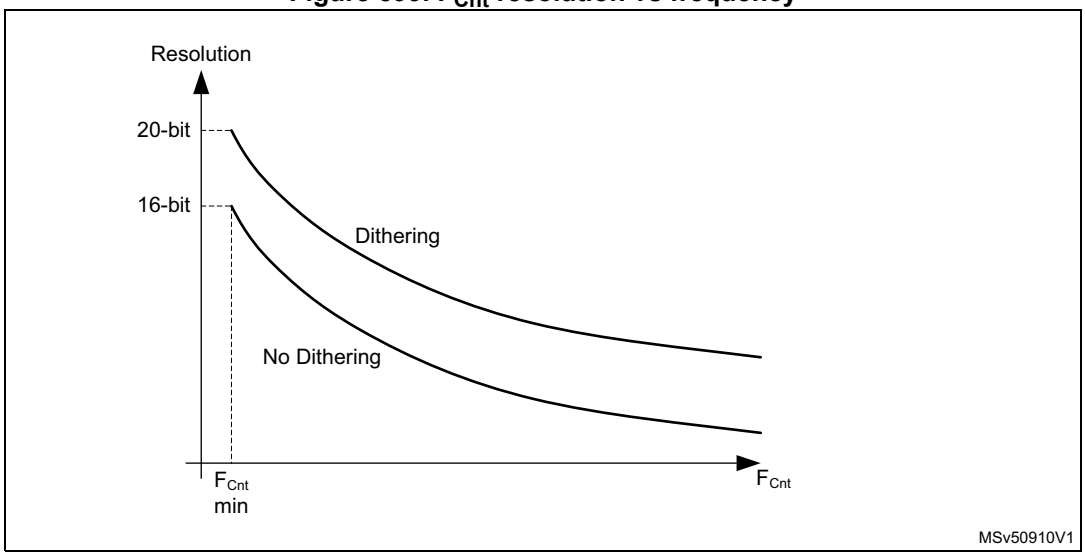
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR value is limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

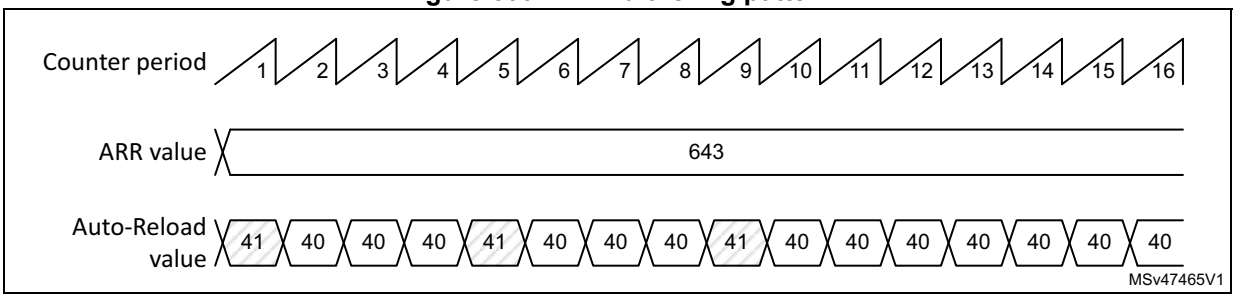
As shown on the *Figure 599* below, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 599. F_{Cnt} resolution vs frequency



The period changes are spread over 16 consecutive periods, as described in the *Figure 600* below.

Figure 600. PWM dithering pattern



The auto-reload and compare values increments are spread following specific patterns described in the *Table 484* below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 484. TIMx_ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

49.3.6 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This is used 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.

49.3.7 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receiving 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.

49.3.8 TIM6/TIM7 DMA requests

The TIM6/TIM7 can generate a single DMA request, as shown in *Table 485*.

Table 485. DMA request

DMA acronym	DMA request	Enable control bit
tim_upd_dma	Update	UDE

49.3.9 Debug mode

When the microcontroller enters debug mode (Cortex®-M4 with FPU core halted), the TIMx counter can either continue to work normally or be stopped.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

49.3.10 TIM6/TIM7 low-power modes

Table 486. Effect of low-power modes on TIM6/TIM7

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

49.3.11 TIM6/TIM7 interrupts

The TIM6/TIM7 can generate a single interrupt, as shown in *Table 487*.

Table 487. Interrupt request

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM6 TIM7	Update	UIF	UIE	write 0 in UIF	Yes	No

49.4 TIM6/TIM7 registers

Refer to *Section 1.2 on page 104* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

49.4.1 TIMx control register 1 (TIMx_CR1)(x = 6 to 7)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw				rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered.
1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generates an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

CEN is cleared automatically in one-pulse mode, when an update event occurs.

49.4.2 TIMx control register 2 (TIMx_CR2)(x = 6 to 7)

Address offset: 0x04

Reset value: 0x0000

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

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

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit from the TIMx_EGR register is used as a trigger output (tim_trgo).

001: **Enable** - the Counter enable signal, tim_cnt_en, is used as a trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated when the CEN control bit is written.

010: **Update** - The update event is selected as a trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

Note: The clock of the slave timer or he peripheral receiving the tim_trgo must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bits 3:0 Reserved, must be kept at reset value.

49.4.3 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.

49.4.4 TIMx status register (TIMx_SR)(x = 6 to 7)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
															rc_w0

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– On counter overflow if UDIS = 0 in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in the TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

49.4.5 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).

49.4.6 TIMx counter (TIMx_CNT)(x = 6 to 7)

Address offset: 0x24

Reset value: 0x0000 0000

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

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

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved and read as 0.

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

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

49.4.7 TIMx prescaler (TIMx_PSC)(x = 6 to 7)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency $f_{tim\_cnt\_ck}$ is equal to $f_{tim\_psc\_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded into the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIMx_EGR register.

49.4.8 TIMx auto-reload register (TIMx_ARR)(x = 6 to 7)

Address offset: 0x2C

Reset value: 0x0000 FFFF

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

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

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded into the actual auto-reload register.

Refer to *Section 49.3.4: Time-base unit on page 1991* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

49.4.9 TIMx register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

Table 488. TIMx register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																				0	0				0				0	0	0	0
0x04	**TIMx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS [2:0]			Res.	Res.	Res.	Res.
	Reset value																										0	0	0				
0x08	Reserved																																
0x0C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
	Reset value																								0								0
0x10	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
	Reset value																																0
0x14	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
	Reset value																																0
0x18-0x20	Reserved																																
0x24	**TIMx_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Section 2.3 on page 115* for the register boundary addresses.

50 Low-power timer (LPTIM)

50.1 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.

50.2 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 capture/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
- Up to 2 independent channels for:
  - Input capture
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on 10 events
- DMA request generation on the following events:
  - Update event
  - Input capture

50.3 LPTIM implementation

Table 489 describes LPTIM implementation on STM32U575/585 devices. The full set of features is implemented in LPTIM1, LPTIM2 and LPTIM3. LPTIM4 supports a smaller set of features.

Table 489. STM32U575/585 LPTIM features

LPTIM modes/features[1]	LPTIM1	LPTIM2	LPTIM3	LPTIM4
Encoder mode	X	X	-	-
PWM mode	X	X	X	X
Input Capture	X	X	X	-
Number of channels	2	2	2	-
Number of DMA requests	3	3	3	-
Wakeup from Stop mode	X[2]	X[3]	X[2]	X[2]
Autonomous mode	X	X	X	-

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.

50.4 LPTIM functional description

50.4.1 LPTIM block diagram

Figure 601. LPTIM1/2/3 timer block diagram

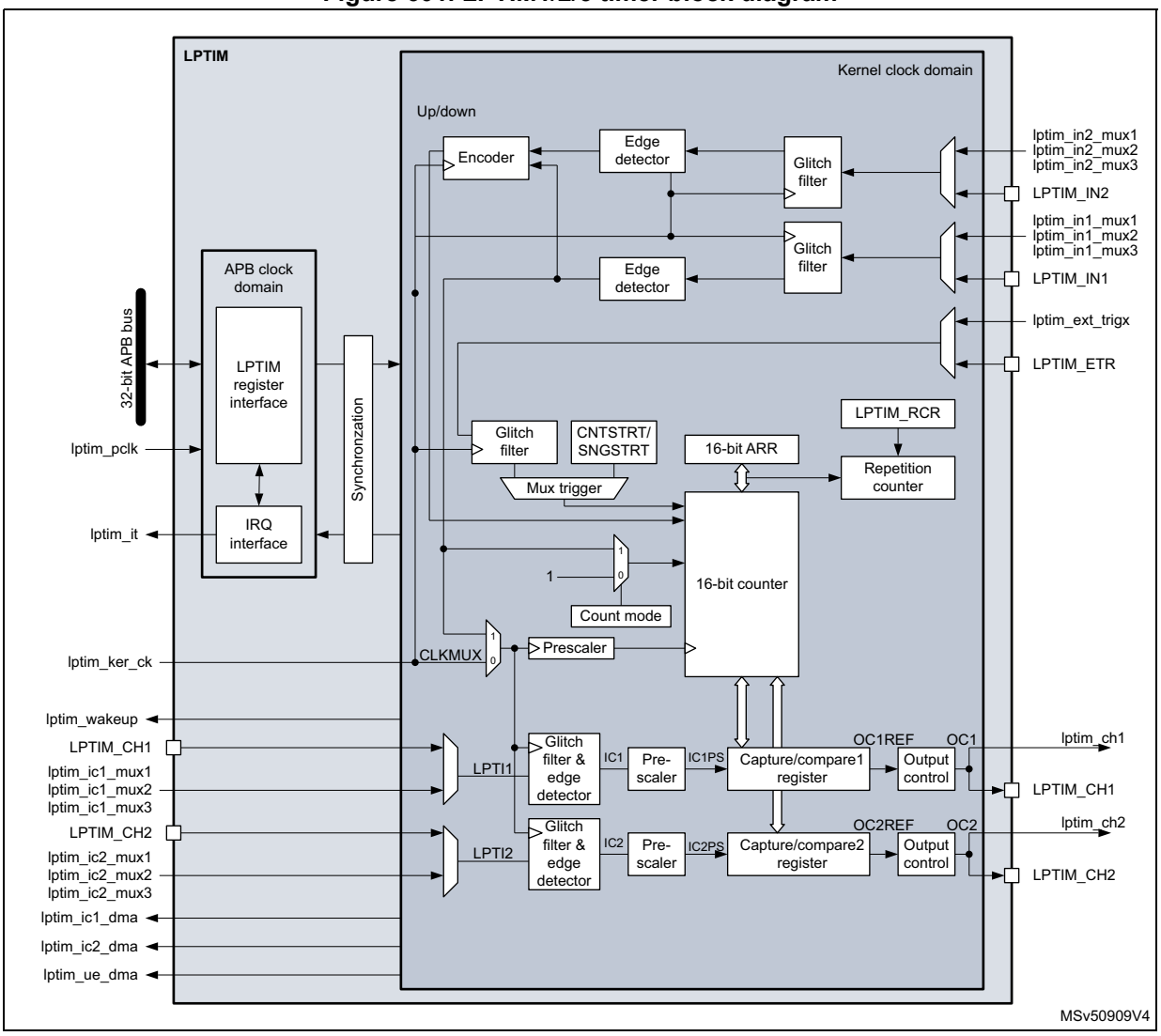
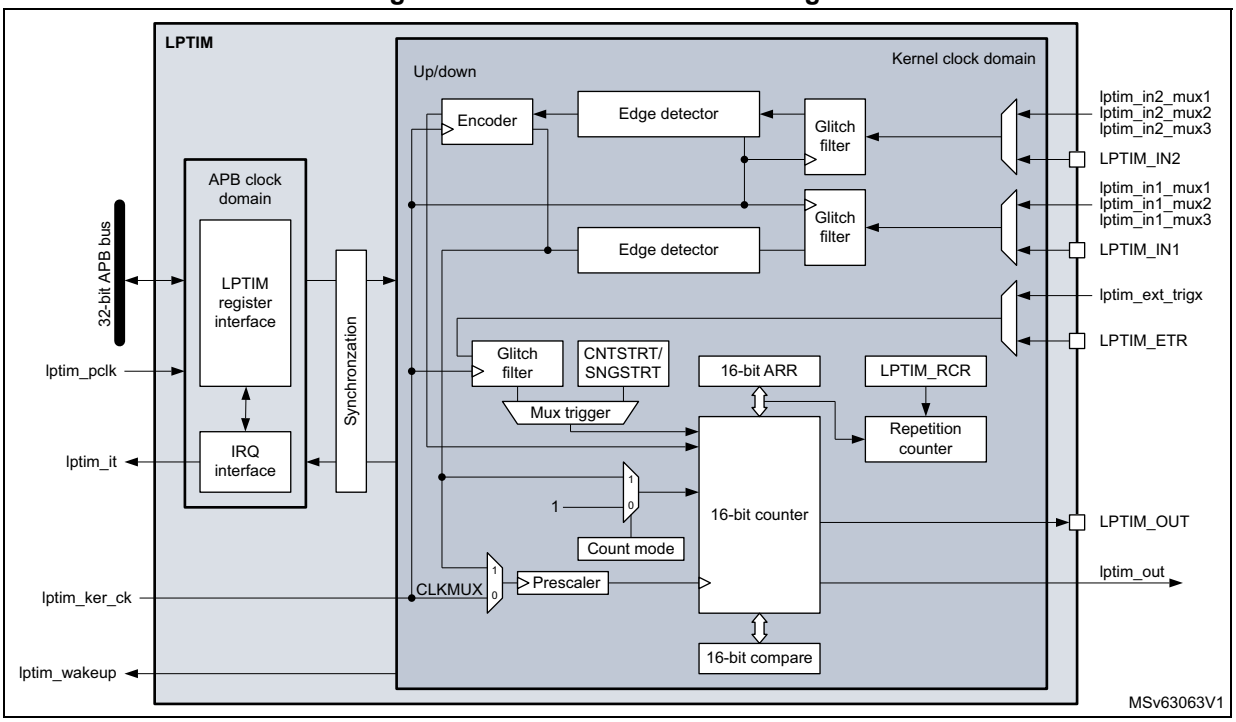


Figure 602. LPTIM4 timer block diagram



50.4.2 LPTIM pins and internal signals

The following tables provide the list of LPTIM pins and internal signals, respectively.

Table 490. LPTIM1/2/3 input/output pins

Names	Signal type	Description
LPTIM_IN1	Digital input	LPTIM input 1 from LPTIMx_IN1 pin on mux input 0
LPTIM_IN2(1)	Digital input	LPTIM input 2 from LPTIMx_IN2 pin on mux input 0
LPTIM_ETR	Digital input	LPTIM external trigger LPTIMx_ETR pin
LPTIM_CH1	Digital input/output	LPTIM channel 1 Input/Output LPTIMx_IN1 pin
LPTIM_CH2	Digital input/output	LPTIM channel 2 Input/Output LPTIMx_IN2 pin

1. LPTIM3 has only the input 1, no input 2.

Table 491. LPTIM4 input/output pins

Names	Signal type	Description
LPTIM_IN1	Digital input	LPTIM Input 1 from LPTIMx_IN1 pin on mux input 0
LPTIM_ETR	Digital input	LPTIM external trigger LPTIMx_ETR pin
LPTIM_OUT	Digital output	LPTIM Output LPTIMx_OUT pin

Table 492. LPTIM1/2/3 internal signals

Names	Signal type	Description
lptim_pclk	Digital input	LPTIM APB clock domain
lptim_ker_ck	Digital input	LPTIM kernel clock
lptim_in1_mux1	Digital input	Internal LPTIM input 1 connected to mux input 1
lptim_in1_mux2	Digital input	Internal LPTIM input 1 connected to mux input 2
lptim_in1_mux3	Digital input	Internal LPTIM input 1 connected to mux input 3
lptim_in2_mux1(1)	Digital input	Internal LPTIM input 2 connected to mux input 1
lptim_in2_mux2(1)	Digital input	Internal LPTIM input 2 connected to mux input 2
lptim_in2_mux3(1)	Digital input	Internal LPTIM input 2 connected to mux input 3
lptim_ic1_mux1	Digital input	Internal LPTIM input capture 1 connected to mux input 1
lptim_ic1_mux2	Digital input	Internal LPTIM input capture 1 connected to mux input 2
lptim_ic1_mux3	Digital input	Internal LPTIM input capture 1 connected to mux input 3
lptim_ic2_mux1	Digital input	Internal LPTIM input capture 2 connected to mux input 1
lptim_ic2_mux2	Digital input	Internal LPTIM input capture 2 connected to mux input 2
lptim_ic2_mux3	Digital input	Internal LPTIM input capture 2 connected to mux input 3
lptim_ext_trigx	Digital input	LPTIM external trigger input x
lptim_it	Digital output	LPTIM global interrupt
lptim_wakeup	Digital output	LPTIM wakeup event
lptim_ic1_dma	Digital output	LPTIM input capture 1 DMA request
lptim_ic2_dma	Digital output	LPTIM input capture 2 DMA request
lptim_ue_dma	Digital output	LPTIM update event DMA request

1. LPTIM3 has only the input 1, no input 2.

Table 493. LPTIM4 internal signals

Names	Signal type	Description
lptim_pclk	Digital input	LPTIM APB clock domain
lptim_ker_ck	Digital input	LPTIM kernel clock
lptim_in1_mux1	Digital input	Internal LPTIM input 1 connected to mux input 1
lptim_in1_mux2	Digital input	Internal LPTIM input 1 connected to mux input 2
lptim_in1_mux3	Digital input	Internal LPTIM input 1 connected to mux input 3
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

50.4.3 LPTIM input and trigger mapping

The LPTIM external trigger and input connections are detailed hereafter:

Table 494. LPTIM1/2/3/4 external trigger connection

TRIGSEL	External trigger			
	LPTIM1	LPTIM2	LPTIM3	LPTIM4
lptim_ext_trig0	GPIO			
lptim_ext_trig1	rtc_alra_trg			
lptim_ext_trig2	rtc_alrb_trg			
lptim_ext_trig3	tamp_trg1			lpdma_ch1_tc
lptim_ext_trig4	tamp_trg2	gpdma_ch0_tc	lpdma_ch1_tc	tamp_trg2
lptim_ext_trig5	lpdma_ch2_tc	gpdma_ch4_tc	tamp_trg3	tamp_trg3
lptim_ext_trig6	comp1_out			
lptim_ext_trig7	comp2_out			

Table 495. LPTIM1/2/3/4 input 1 connection

lptim_in1_mux	LPTIM1/2/3/4 input 1 connected to
lptim_in1_mux0	GPIO
lptim_in1_mux1	comp1_out
lptim_in1_mux2	-
lptim_in1_mux3	-

Table 496. LPTIM1/2 input 2 connection

lptim_in2_mux	LPTIM1/2 input 2 connected to
lptim_in2_mux0	GPIO
lptim_in2_mux1	comp2_out
lptim_in2_mux2	-
lptim_in2_mux3	-

Table 497. LPTIM1/2/3 input capture 1 connection

lptim_ic1_mux	LPTIM1/2/3 input capture 1 connected to
lptim_ic1_mux0	GPIO
lptim_ic1_mux1	comp1_out
lptim_ic1_mux2	comp2_out
lptim_ic1_mux3	-

Table 498. LPTIM1 input capture 2 connection

lptim_ic2_mux	LPTIM1 input capture 2 connected to
lptim_ic2_mux0	I/O
lptim_ic2_mux1	LSI
lptim_ic2_mux2	LSE
lptim_ic2_mux3	-

Table 499. LPTIM2 input capture 2 connection

lptim_ic2_mux	LPTIM2 input capture 2 connected to
lptim_ic2_mux0	I/O
lptim_ic2_mux1	HSI/256
lptim_ic2_mux2	MSIS/1024
lptim_ic2_mux3	MSIS/4

Table 500. LPTIM3 input capture 2 connection

lptim_ic2_mux	LPTIM3 input capture 2 connected to
lptim_ic2_mux0	I/O
lptim_ic2_mux1	-
lptim_ic2_mux2	-
lptim_ic2_mux3	-

50.4.4 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.

50.4.5 Glitch filter

The LPTIM inputs, either external (mapped to GPIOs) or internal (mapped on the chip-level to other embedded peripherals), are protected with digital filters that prevent any glitches and noise perturbations to propagate inside the LPTIM. This is in order to prevent spurious counts or triggers.

Before activating the digital filters, an internal clock source should first be provided to the LPTIM. This is necessary to guarantee the proper operation of the filters.

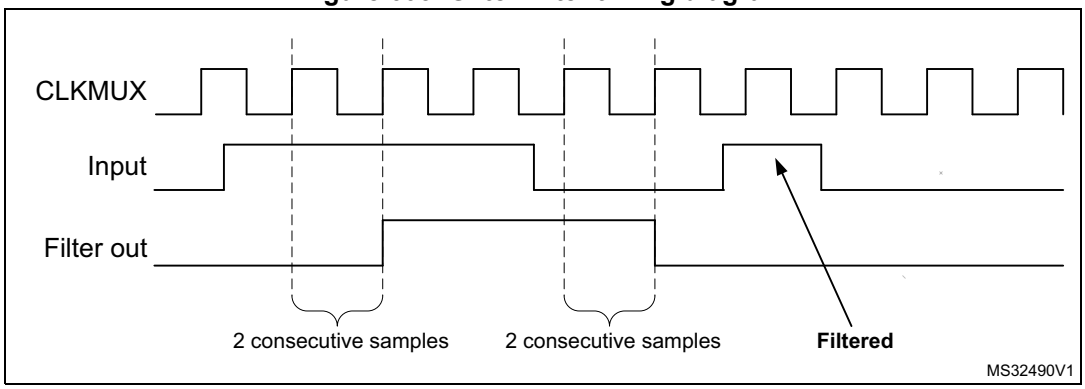
The digital filters are divided into three groups:

- The first group of digital filters protects the LPTIM internal or external inputs. The digital filters sensitivity is controlled by the CKFLT bits
- The second group of digital filters protects the LPTIM internal or external trigger inputs. The digital filters sensitivity is controlled by the TRGFLT bits.
- The third group of digital filters protects the LPTIM internal or external input captures. The digital filters sensitivity is controlled by the ICxF 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 603* shows an example of glitch filter behavior in case of a 2 consecutive samples programmed.

Figure 603. Glitch filter timing diagram



Note: In case no internal clock signal is provided, the digital filter must be deactivated by setting the CKFLT, ICxF and TRGFLT bits to '0'. In that case, an external analog filter may be used to protect the LPTIM external inputs against glitches.

50.4.6 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 501. Prescaler division ratios

programming	dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

50.4.7 Trigger multiplexer

The LPTIM counter may be started either by software or after the detection of an active edge on one of the 8 trigger inputs.

TRIGEN[1:0] is used to determine the LPTIM trigger source:

- When TRIGEN[1:0] equals ‘00’, The LPTIM counter is started as soon as one of the CNTSTRT or the SNGSTRT bits is set by software. The three remaining possible values for the TRIGEN[1:0] are used to configure the active edge used by the trigger inputs. The LPTIM counter starts as soon as an active edge is detected.
- When TRIGEN[1:0] is different than ‘00’, TRIGSEL[2:0] is used to select which of the 8 trigger inputs is used to start the counter.

The external triggers are considered asynchronous signals for the LPTIM. So after a trigger detection, a two-counter-clock period latency is needed before the timer starts running due to the synchronization.

If a new trigger event occurs when the timer is already started it is ignored (unless timeout function is enabled).

Note: The timer must be enabled before setting the SNGSTRT/CNTSTRT bits. Any write on these bits when the timer is disabled is discarded by hardware.

Note: When starting the counter by software (TRIGEN[1:0] = 00), there is a delay of 3 kernel clock cycles between the LPTIM_CR register update (set one of SNGSTRT or CNTSTRT bits) and the effective start of the counter.

50.4.8 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.

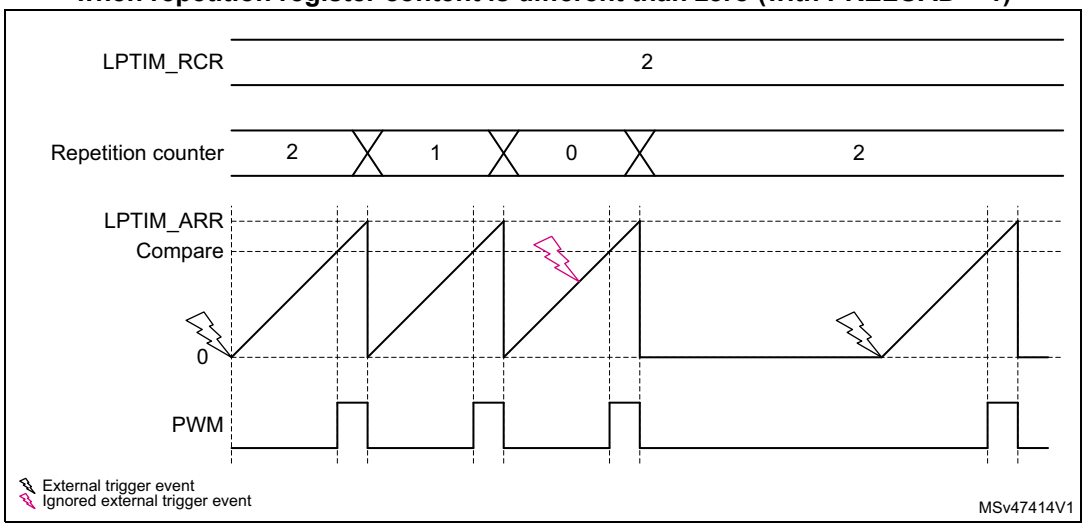
One-shot mode

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 604*.

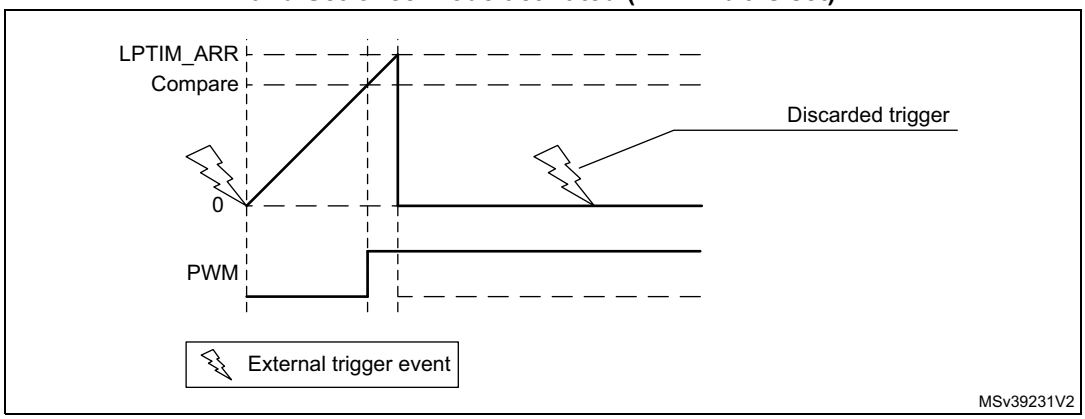
Figure 604. LPTIM output waveform, single counting mode configuration when repetition register content is different than zero (with PRELOAD = 1)



- 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 605*.

Figure 605. LPTIM output waveform, Single counting mode configuration and Set-once mode activated (WAVE bit is set)



In case of software start (TRIGEN[1:0] = ‘00’), the SNGSTRT setting starts the counter for one-shot counting.

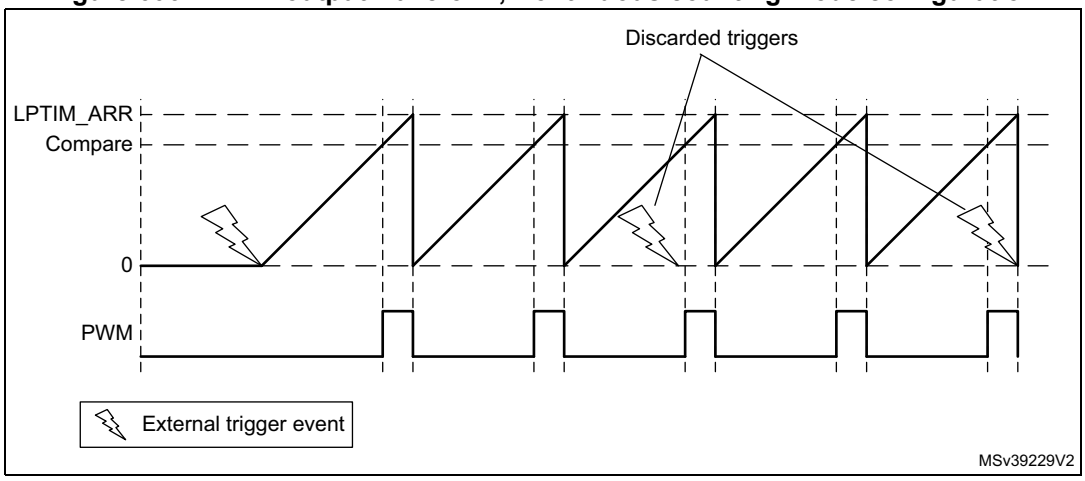
Continuous mode

To enable the continuous counting, the CNTSTRT bit must be set.

In case an external trigger is selected, an external trigger event arriving after CNTSTRT is set, starts the counter for continuous counting. Any subsequent external trigger event is discarded as shown in *Figure 606*.

In case of software start (TRIGEN[1:0] = ‘00’), setting CNTSTRT starts the counter for continuous counting.

Figure 606. LPTIM output waveform, Continuous counting mode configuration



SNGSTRT and CNTSTRT bits can only be set when the timer is enabled (The ENABLE bit is set to ‘1’). It is possible to change “on the fly” from One-shot mode to Continuous mode.

If the Continuous mode was previously selected, setting SNGSTRT switches the LPTIM to the One-shot mode. The counter (if active) stops as soon as an LPTIM update event is generated.

If the One-shot mode was previously selected, setting CNTSTRT switches the LPTIM to the Continuous mode. The counter (if active) restarts as soon as it reaches ARR.

50.4.9 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.

50.4.10 Waveform generation

Two 16-bit registers, the LPTIM_ARR (autoreload register) and LPTIM_CCRx (capture/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_CCRx. The LPTIM output is reset as soon as a match occurs between the LPTIM_ARR and the LPTIM_CNT register. For more details see *Section 50.4.19: PWM mode*.
- 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_CCRx register value.

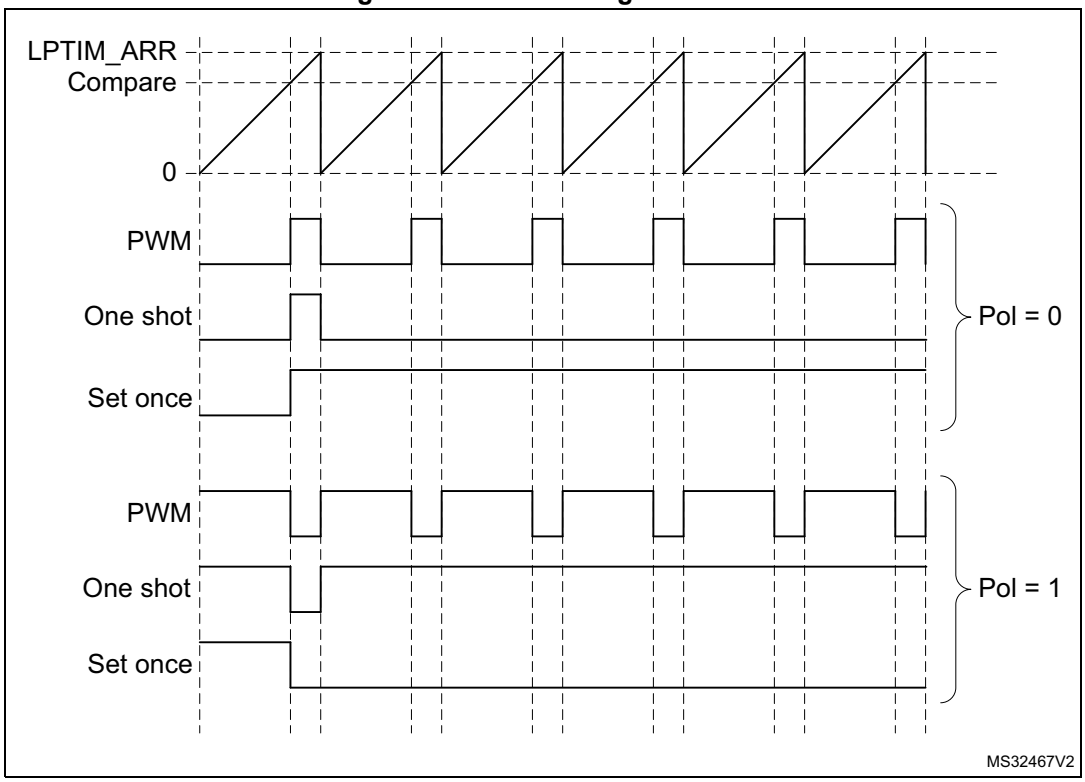
The LPTIM output waveform can be configured through the WAVE bit as follow:

- 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/CCxP 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 607* 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/CCxP bit.

Figure 607. Waveform generation



50.4.11 Register update

The LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx 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, the LPTIM_RCR and the LPTIM_CCRx registers are updated:

- When the PRELOAD bit is reset to ‘0’, the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are immediately updated after any write access.
- When the PRELOAD bit is set to ‘1’, the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx 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, the REPOK flag and the CMPxOK flag in the LPTIM_ISR register indicate when the write operation is completed to respectively the LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx register.

After a write to the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx 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, the REPOK flag or the CMPxOK flag be set, leads to unpredictable results.

50.4.12 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.

50.4.13 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 register must be modified only when the LPTIM is disabled.

50.4.14 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.

50.4.15 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.

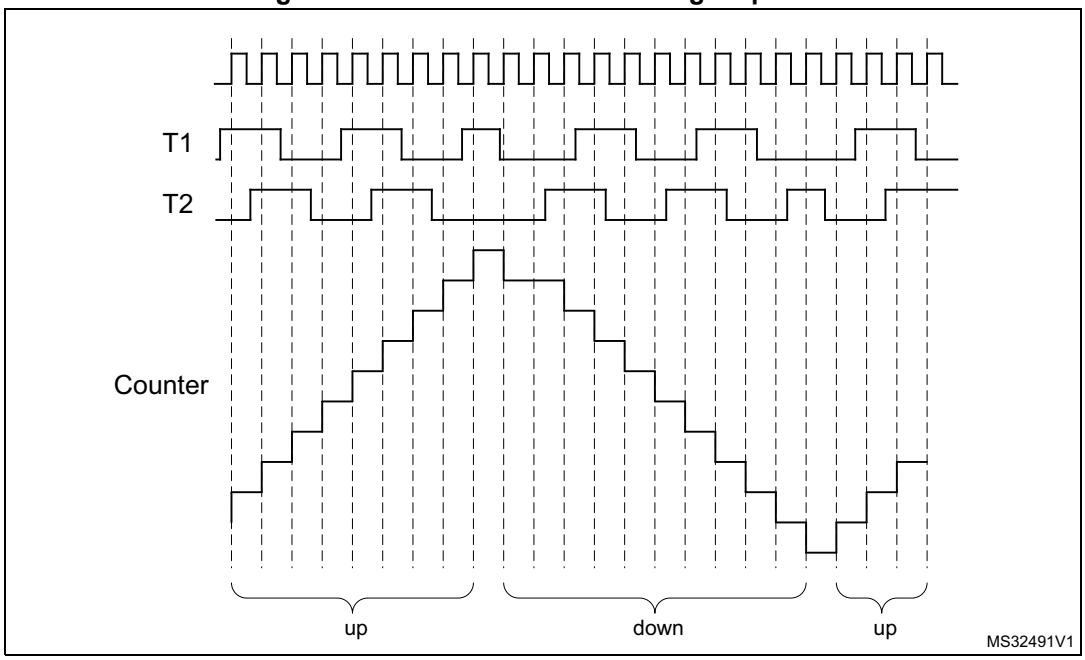
Table 502. Encoder counting scenarios

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 608. Encoder mode counting sequence



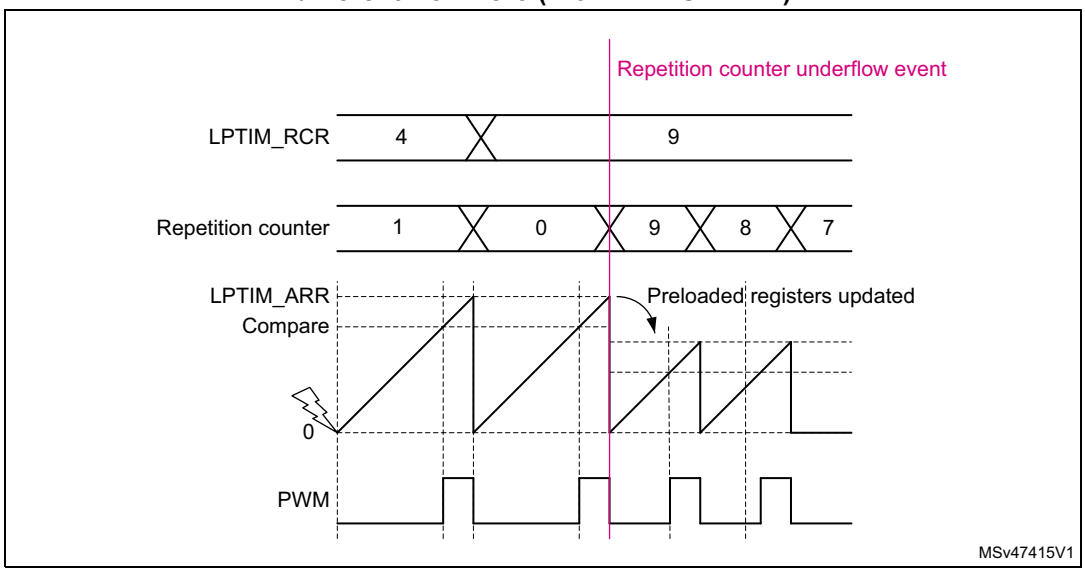
50.4.16 Repetition Counter

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 609*.

Figure 609. 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_DIER 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_DIER 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.

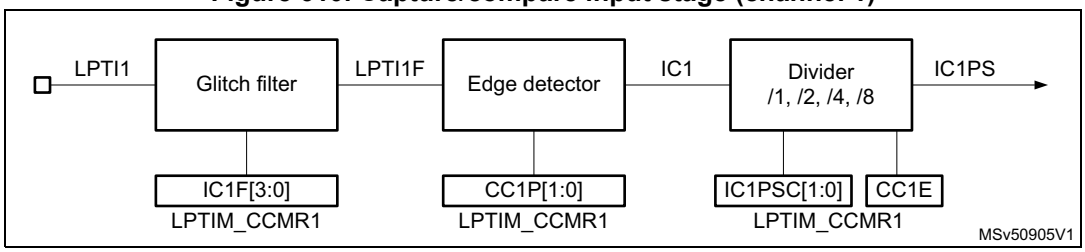
50.4.17 Capture/compare channels

Each capture/compare channel is built around a capture/compare register, an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control) for PWM.

Input stage

The input stage samples the corresponding LPTIx input to generate a filtered signal LPTIxF. Then, an edge detector with polarity selection generates ICx signal used as the capture command. It is prescaled to generate the capture command signal (ICxPS).

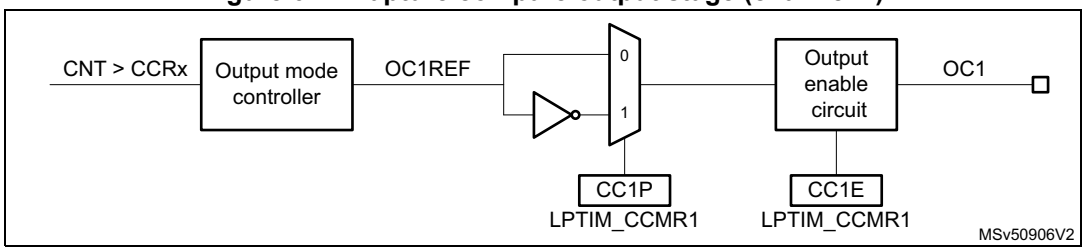
Figure 610. Capture/compare input stage (channel 1)



Output 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 611. Capture/compare output stage (channel 1)



50.4.18 Input capture mode

In Input capture mode, the capture/compare registers (LPTIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. Assuming input capture is enabled on a channel x (CCxE set) and when a capture occurs, the corresponding CCxIF flag (LPTIM_ISR 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 (LPTIM_ISR register) is set. CCxIF can be cleared by software by writing the CCxICF to 1 or by reading the captured data stored in the LPTIM_CCRx register. CCxOF is cleared by writing CCxOCF to 1.

Note: In DMA mode, the input capture channel have to be enabled (set CCxE bit) the last, after enabling the IC DMA request and after starting the counter. This is in order to prevent generating an input capture DMA request when the counter is not started yet.

Input capture Glitch filter latency

When a trigger event arrives on channel x input (LPTIx) and depending on the configured glitch filter (ICxF[1:0] field in CCMRx register) and on the kernel clock prescaler value

(PRESC[2:0] field in CFGR register), there is a variable latency that leads to a systematic offset (see *Table 503*) between the captured value stored in the CCRx register and the real value corresponding to the capture trigger.

This offset has no impact on pulse width measurement as it is systematic and compensated between two captures.

The real capture value corresponding to the input capture trigger can be calculated using the below formula:

Real capture value = captured(LPTIM_CCRx) - offset

The relevant offset should be used depending on the glitch filter and on the kernel clock prescaler value (PRESC field in CFGR register)

Example: determining the real capture value when PRESC[2:0] = 0x2 and ICxF = 0x3.

For this configuration (PRESC[2:0] = 0x2 and ICxF = 0x3) and according to the *Table 503*, the offset is 5.

Assuming that the captured value in CCRx is 9 (LPTIM_CNT = 9), this means that the capture trigger occurred when the LPTIM_CNT was equal to 9 - 5 = 4.

Table 503. Input capture Glitch filter latency (in counter step unit)

Prescaler PRESC[2:0]	ICxF[1:0]	Offset
0	0	2
	1	7
	2	9
	3	13
1	0	3
	1	5
	2	6
	3	8
2	0	2
	1	3
	2	4
	3	5
3	0	2
	1	2
	2	3
	3	3
4	0	2
	1	2
	2	2
	3	2

Table 503. Input capture Glitch filter latency (in counter step unit) (continued)

Prescaler PRESC[2:0]	ICxF[1:0]	Offset
5	0	2
	1	2
	2	2
	3	2
6	0	2
	1	2
	2	2
	3	2
7	0	2
	1	2
	2	2
	3	2

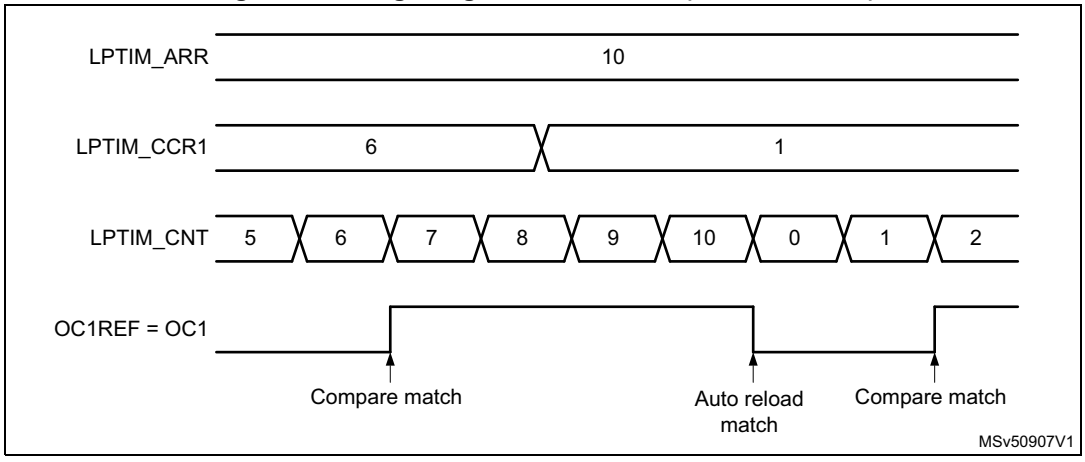
50.4.19 PWM mode

The PWM mode enables to generate a signal with a frequency determined by the value of the LPTIM_ARR register and a duty cycle determined by the value of the LPTIM_CCRx register. The LPTIM is able to generate PWM in edge-aligned mode.

OCx polarity is software programmable using the CCxP bit in the LPTIM_CCMRx register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the LPTIM_CCMRx register. Refer to the LPTIM_CCMRx register description for more details.

Figure 612 gives an example where the LPTIM channel 1 is configured in PWM mode with LPTIM_CCR1 = 6 then 1 and LPTIM_ARR=10.

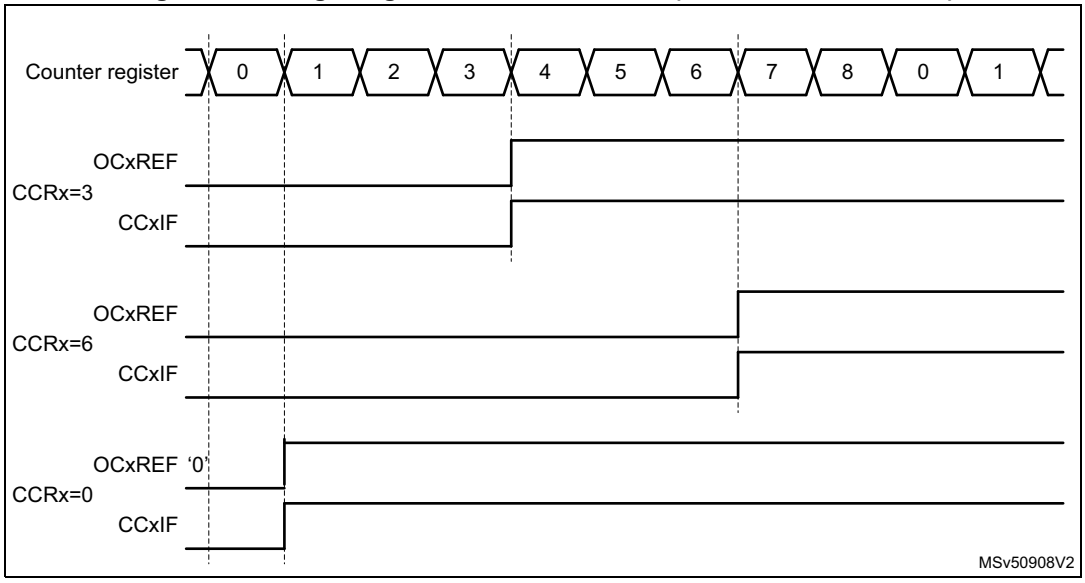
Figure 612. Edge-aligned PWM mode (PRELOAD = 1)



In the following example the reference PWM signal OCxREF is low as long as LPTIM_CNT ≤ LPTIM_CCRx else it becomes high.

Figure 613 shows some edge-aligned PWM waveforms in an example where LPTIM_ARR = 8.

Figure 613. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0)



PWM mode with immediate update PRELOAD = 0

The PWM mode with PRELOAD = 0 enables the early change of the output level within the current PWM cycle. Based on the immediate update (PRELOAD = 0) of the LPTIM_CCRx register and on the continuous comparison of LPTIM_CNT and LPTIM_CCRx registers, it permits to have a new duty cycle value applied as soon as possible within the current PWM cycle, without having to wait for the completion of the current PWM period.

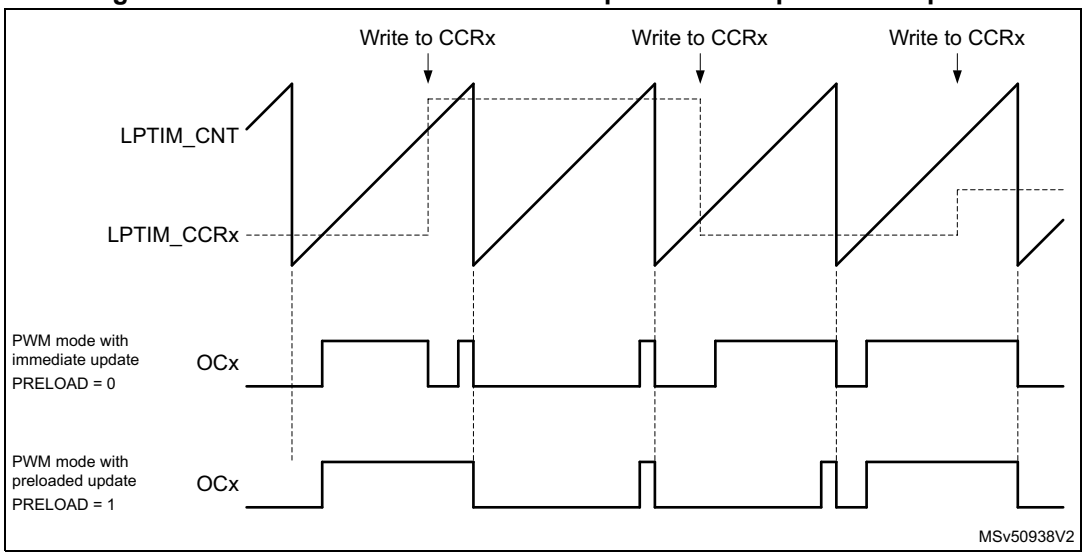
When the (PRELOAD = 0), the OCxREF signal level can be changed on-the-fly by software (or DMA) by updating the compare value in the LPTIM_CCRx register.

Depending on the written compare value and on the current counter and compare values, the OCxREF level is re-assigned as illustrated below:

- If the new compare value does not exceed the current counter value and the current compare value exceeds the counter, OCxREF level is re-assigned high as soon as the new compare value is written.
- If the new compare value exceeds the counter value and the current compare value does not exceed the counter, OCxREF level is re-assigned low as soon as the new compare value is written.

The output reference signal OCxREF level is left unchanged when none of the new compare value and the current compare value exceed the counter. *Figure 614* illustrates the behavior of the OCxREF signal level when PRELOAD = 0 and PRELOAD = 1.

Figure 614. PWM mode with immediate update versus preloaded update



Note: For both PWM modes, the compare match, auto-reload match and the update event flags are set one LPTIM counter cycle later after the corresponding event, the OCxREF level is also changed one LPTIM counter cycle later after the corresponding event. For instance when the LPTIM_CCRx is set to 3 the CCxIF is set when the LPTIM_CNT = 4. Figure 612 illustrates this behavior.

50.4.20 Autonomous mode

When clocked by oscillators available in this mode (refer to RCC), the LPTIM can operate in autonomous mode, permitting it to remain fully functional in Stop mode where the APB clock is stopped. The APB clock is requested by the peripheral each time a data must be transferred from or to the SRAM. Once the APB clock is received by the peripheral, either an interrupt or a DMA request is generated, depending on the LPTIM configuration.

In order to offload the CPU (in Run mode) or to avoid to wake it up when in Stop mode, it is possible to use LPTIM DMA requests to transfer the captured values when in input capture mode or to update LPTIM registers when in PWM mode.

When in Stop mode, the LPTIM counter can be automatically started after the detection of an active edge on one of its external input triggers.

Input capture mode

To operate autonomously in stop mode, the input capture DMA request should be enabled by setting the CCxDE bit in the LPTIM_DIER register.

Each time a counter value is captured and available in the LPTIM_CCRx register, the APB clock is requested by the peripheral and a DMA request is generated. The captured value is then transferred to the SRAM. The CCxIF flag is automatically cleared by hardware once the captured value is read by APB (can be any bus master like CPU or DMA).

PWM mode

The LPTIM can be configured to autonomously change, at each update event, the output waveform pulse width and/or the duty cycle without any CPU intervention. To enable this autonomous mode, the corresponding UEDE bit should be set in the LPTIM_DIER register.

At each update event, the APB clock is requested by the peripheral and a DMA request is generated. DMA direction should be configured as memory-to-peripheral which enables updating LPTIM registers, at each DMA request, with values stored in SRAM.

The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA. Thus, to enable automatic hardware clearing of UE flag, the application should configure the LPTIM_ARR register to be the last one to be written (at the end of list). For instance if LPTIM_CCR1 and LPTIM_RCR registers need to be updated in Stop mode by DMA, the update sequence must be: LPTIM_CCR1, LPTIM_RCR then LPTIM_ARR.

The UE flag can also be cleared over its corresponding clear bit UECF in the LPTIM_ICR register, this can be done by configuring the DMA to write the LPTIM_ICR register at the end of register update.

50.4.21 DMA requests

The LPTIM has the capability to generate two categories of DMA requests:

- DMA requests used to retrieve the input-capture counter values
- DMA update requests are used to re-program part of the LPTIMER, multiple times, at regular intervals, without software overhead.

Input capture DMA request

Each LPTIM channel has its dedicated input capture DMA request. A DMA request is generated (if CCxDE bit is set in LPTIM_DIER) and CCxIF is set each time a capture is ready in the CCRx register. The captured values in CCRx can then be transferred regularly by DMA to the desired memory destination. The CCxIF is automatically cleared by hardware when the captured value in CCRx register is read.

Note: The ICx DMA request signal lptim_icx_dma is reset in the following conditions:

- *if the corresponding DMA request is disabled (clear CCxDE bit in the LPTIM_DIER register)*
- *or if the channel x is disabled (clear CCxE bit)*
- *or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)*

Update event DMA request

A DMA request is generated (if UEDE is set in LPTIM_DIER) and the UE flag is set at each update event. DMA request can be used to regularly update the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx registers permitting to generate custom PWM waveforms.

The UE is automatically cleared by hardware upon any bus master (like CPU or DMA) write access to the LPTIM_ARR register.

Note: The UE DMA request signal lptim_ue_dma is reset in the following conditions:

- *if the corresponding DMA request is disabled (clear UEDE bit in the LPTIM_DIER register)*
- *or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)*

50.4.22 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.

50.5 LPTIM low-power modes

Table 504. Effect of low-power modes on the LPTIM

Mode	Description
Sleep	No effect. LPTIM interrupts cause the device to exit Sleep mode.
Stop(1)	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. The DMA requests are functional if the instance supports the autonomous mode (refer to *Section 50.3: LPTIM implementation*).
Standby	The LPTIM peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Only LPTIM1 and LPTIM3 support autonomous mode with wakeup capability in Stop 2 mode. LPTIM4 does not support autonomous mode but has wakeup capability in stop 2 mode. LPTIM2 must be disabled before entering Stop 2 mode.

50.6 LPTIM interrupts

The following events generate an interrupt/wake-up event, if they are enabled through the LPTIM_DIER 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
- Input capture occurred
- Over-capture occurred
- Interrupt enable register update OK

Note: If any bit in the LPTIM_DIER register is set after that its corresponding flag in the LPTIM_ISR register (Status Register) is set, the interrupt is not asserted.

Table 505. Interrupt events

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop mode(1)
LPTIMx	Compare match	CCxIF	CCxIE	Write 1 to CCxCF	Yes	Yes
	Input capture(2)	CCxIF	CCxIE	Write 1 to CCxCF	Yes	Yes
	Over-capture(2)	CCxOF	CCxOIE	Write 1 to CCxOCF	Yes	Yes
	Auto-reload match	ARRM	ARRMIE	Write 1 to ARRMCF	Yes	Yes
	External trigger event	EXTTRIG	EXTTRIGIE	Write 1 to EXTTRIGCF	Yes	Yes
	Auto-reload register update OK	ARROK	ARROKIE	Write 1 to ARROKCF	Yes	Yes
	Capture/compare register update OK	CMPxOK	CMPxOKIE	Write 1 to CMPxOKCF	Yes	Yes
	Direction change to up(3)	UP	UPIE	Write 1 to UPCF	Yes	Yes
	Direction change to down(3)	DOWN	DOWNIE	Write 1 to DOWNCF	Yes	Yes
	Update Event	UE	UEIE	Write 1 to UECF	Yes	Yes
	Repetition register update OK	REPOK	REPOKIE	Write 1 to REPOKCF	Yes	Yes

1. Each LPTIM event can wake up the device from Stop mode only if the LPTIM instance supports the Wakeup from Stop mode feature. Refer to *Section 50.3: LPTIM implementation*.
2. If LPTIM does not implement any channel this event does not exist. Refer to *Section 50.3: LPTIM implementation*.
3. If LPTIM does not support encoder mode feature, this event does not exist. Refer to *Section 50.3: LPTIM implementation*.

50.7 LPTIM registers

50.7.1 LPTIM4 interrupt and status register (LPTIM4_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.	DIER OK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							r								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP OK	UE	DOWN	UP	ARR OK	CMP1 OK	EXT TRIG	ARRM	CC1IF
							r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:9 Reserved, must be kept at reset value.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 50.3: LPTIM implementation.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 50.3: 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 **CMP1OK**: Compare register 1 update OK

CMP1OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR1 register has been successfully completed. CMP1OK flag can be cleared by writing 1 to the CMP1OKCF bit in the LPTIM_ICR register.

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 **CC1IF**: Compare 1 interrupt flag

The CC1IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. The CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR1 register's value

50.7.2 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 3)

This description of the register can only be used for output compare mode. See next section for input capture mode.

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.	DIEROK	Res.	Res.	Res.	Res.	CMP2OK	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.	CC2IF	REPOK	UE	DOWN	UP	ARROK	CMP1OK	EXTTRIG	ARRM	CC1IF
						r	r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OK**: Compare register 2 update OK

CMP2OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR2 register has been successfully completed. CMP2OK flag can be cleared by writing 1 to the CMP2OKCF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 50.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IF**: Compare 2 interrupt flag

If channel CC2 is configured as output:

The CC2IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR2 register's value

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 50.3.

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. The corresponding interrupt or DMA request is generated if enabled. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register. The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 50.3.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 50.3.

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 **CMP1OK**: Compare register 1 update OK

CMP1OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR1 register has been successfully completed. CMP1OK flag can be cleared by writing 1 to the CMP1OKCF bit in the LPTIM_ICR register.

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 **CC1IF**: Compare 1 interrupt flag

If channel CC1 is configured as output:

The CC1IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR1 register's value

50.7.3 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 3)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

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.	DIEROK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2OF	CC1OF	Res.	Res.	CC2IF	REPOK	UE	DOWN	UP	ARROK	Res.	EXTTRIG	ARRM	CC1IF
		r	r			r	r	r	r	r	r		r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OF**: Capture 2 over-capture 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 1 to the CC2OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.

1: The counter value has been captured in LPTIM_CCR2 register while CC2IF flag was already set.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 50.3.

Bit 12 **CC1OF**: Capture 1 over-capture 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 1 to the CC1OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.

1: The counter value has been captured in LPTIM_CCR1 register while CC1IF flag was already set.

Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 50.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IF**: Capture 2 interrupt flag

If channel CC2 is configured as input:

CC2IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR2 register. The corresponding interrupt or DMA request is generated if enabled. The CC2OF flag is set if the CC2IF flag was already high.

0: No input capture occurred

1: The counter value has been captured in the LPTIM_CCR2 register. (An edge has been detected on IC2 which matches the selected polarity). The CC2IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA). The CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 50.3.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 50.3.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 50.3.

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 Reserved, must be kept at reset value.

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 **CC1IF**: capture 1 interrupt flag

If channel CC1 is configured as input:

CC1IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR1 register. The corresponding interrupt or DMA request is generated if enabled. The CC1OF flag is set if the CC1IF flag was already high.

0:No input capture occurred

1:The counter value has been captured in the LPTIM_CCR1 register. (An edge has been detected on IC1 which matches the selected polarity). The CC1IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA).CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

50.7.4 LPTIM4 interrupt clear register (LPTIM4_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.	DIEROKCF	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.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKCF	UECF	DOWNCF	UPCF	ARROKCF	CMP1OKCF	EXTTRIGCF	ARRMCF	CC1CF
							w	w	w	w	w	w	w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag

Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:9 Reserved, must be kept at reset value.

Bit 8 **REPOKCF**: Repetition register update OK clear flag

Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag

Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag

Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 50.3.

Bit 5 **UPCF**: Direction change to UP clear flag

Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 50.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 **CMP1OKCF**: Compare register 1 update OK clear flag
Writing 1 to this bit clears the CMP1OK 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 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

50.7.5 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 3)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OKCF	Res.	Res.	Res.	Res.	CMP2 OKCF	Res.	Res.	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.	CC2CF	REPOK CF	UECF	DOWN CF	UPCF	ARR OKCF	CMP1 OKCF	EXT TRIG CF	ARRM CF	CC1CF
						w	w	w	w	w	w	w	w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag
Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OKCF**: Compare register 2 update OK clear flag
Writing 1 to this bit clears the CMP2OK flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 50.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2CF**: Capture/compare 2 clear flag
Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 50.3.

Bit 8 **REPOKCF**: Repetition register update OK clear flag
Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag
Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.
Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 50.3.

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.
Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 50.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 **CMP1OKCF**: Compare register 1 update OK clear flag
Writing 1 to this bit clears the CMP1OK 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 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

50.7.6 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 3)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	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.	Res.	CC2OCF	CC1OCF	Res.	Res.	CC2CF	REPOKCF	UECF	DOWNCF	UPCF	ARROKCF	Res.	EXTTRIGCF	ARRMCF	CC1CF
		w	w			w	w	w	w	w	w		w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag
Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OCF**: Capture/compare 2 over-capture clear flag
Writing 1 to this bit clears the CC2OF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 50.3.

Bit 12 **CC1OCF**: Capture/compare 1 over-capture clear flag
Writing 1 to this bit clears the CC1OF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 50.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2CF**: Capture/compare 2 clear flag
Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 50.3.

Bit 8 **REPOKCF**: Repetition register update OK clear flag
Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag
Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.
Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 50.3.

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.
Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 50.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 Reserved, must be kept at reset value.

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 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

50.7.7 LPTIM4 interrupt enable register (LPTIM4_DIER)

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	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 50.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 50.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 **CMP1OKIE**: Compare register 1 update OK interrupt enable
0: CMPOK register 1 interrupt disabled
1: CMPOK register 1 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 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

Caution: The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER 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 DIEROK flag is set, leads to unpredictable results.

50.7.8 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 3)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

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.	UEDE	Res.	Res.	Res.	CMP2OKIE	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.	CC2IE	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 23 **UEDE**: Update event DMA request enable
- 0: UE DMA request disabled. Writing '0' to the UEDE bit resets the associated ue_dma_req signal.
- 1: UE DMA request enabled

Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 50.3.

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OKIE**: Compare register 2 update OK interrupt enable
- 0: CMPOK register 2 interrupt disabled
- 1: CMPOK register 2 interrupt enabled

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 50.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IE**: Capture/compare 2 interrupt enable
- 0: Capture/compare 2 interrupt disabled
- 1: Capture/compare 2 interrupt enabled

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 50.3.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
- 0: Repetition register update OK interrupt disabled
- 1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
- 0: Update event interrupt disabled
- 1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 50.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 50.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 **CMP1OKIE**: Compare register 1 update OK interrupt enable
0: CMPOK register 1 interrupt disabled
1: CMPOK register 1 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 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

Caution: The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER 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 DIEROK flag is set, leads to unpredictable results.

50.7.9 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 3)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

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.	CC2DE	Res.	UEDE	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE
						rw		rw							rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2OIE	CC1OIE	Res.	Res.	CC2IE	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	Res.	EXTTRIGIE	ARRMIE	CC1IE
		rw	rw			rw	rw	rw	rw	rw	rw		rw	rw	rw

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

Bit 27 Reserved, must be kept at reset value.

Bit 26 Reserved, must be kept at reset value.

Bit 25 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled. Writing '0' to the CC2DE bit resets the associated ic2_dma_req signal.
1: CC2 DMA request enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 50.3.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **UEDE**: Update event DMA request enable
0: UE DMA request disabled. Writing '0' to the UEDE bit resets the associated ue_dma_req signal.
1: UE DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 50.3.

Bits 22:17 Reserved, must be kept at reset value.

Bit 16 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled. Writing '0' to the CC1DE bit resets the associated ic1_dma_req signal.
1: CC1 DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 50.3.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OIE**: Capture/compare 2 over-capture interrupt enable
0: CC2 over-capture interrupt disabled
1: CC2 over-capture interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 50.3.

Bit 12 **CC1OIE**: Capture/compare 1 over-capture interrupt enable
0: CC1 over-capture interrupt disabled
1: CC1 over-capture interrupt enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 50.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IE**: Capture/compare 2 interrupt enable
0: Capture/compare 2 interrupt disabled
1: Capture/compare 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 50.3.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 50.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 50.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 Reserved, must be kept at reset value.

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 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

Caution: The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER 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 DIEROK flag is set, leads to unpredictable results.

50.7.10 LPTIM configuration register (LPTIM_CFGR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC	COUNT MODE	PRE LOAD	WAV POL	WAVE	TIMOUT	TRIGEN[1:0]		Res.
							rw	rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGSEL[2:0]			Res.	PRESC[2:0]			Res.	TRGFLT[1:0]		Res.	CKFLT[1:0]		CKPOL[1:0]		CKSEL
rw	rw	rw		rw	rw	rw		rw	rw		rw	rw	rw	rw	rw

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

Bit 29 Reserved, must be kept at reset value.

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **ENC**: Encoder mode enable

The ENC bit controls the Encoder mode

0: Encoder mode disabled

1: Encoder mode enabled

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 50.3.

Bit 23 **COUNTMODE**: counter mode enabled

The COUNTMODE bit selects which clock source is used by the LPTIM to clock the counter:

0: the counter is incremented following each internal clock pulse

1: the counter is incremented following each valid clock pulse on the LPTIM external Input1

Bit 22 **PRELOAD**: Registers update mode

The PRELOAD bit controls the LPTIM_ARR, LPTIM_RCR and the LPTIM_CCRx registers update modality

0: Registers are updated after each APB bus write access

1: Registers are updated at the end of the current LPTIM period

Bit 21 **WAVPOL**: Waveform shape polarity

The WAVPOL bit controls the output polarity

0: The LPTIM output reflects the compare results between LPTIM_CNT and LPTIM_CCRx registers

1: The LPTIM output reflects the inverse of the compare results between LPTIM_CNT and LPTIM_CCRx registers

Note: If the LPTIM implements at least one capture/compare channel, this bit is reserved. Refer to Section 50.3.

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 50.4.3: LPTIM input and trigger mapping* for details.

Bit 12 Reserved, must be kept at reset value.

Bits 11:9 **PRESC[2:0]:** Clock prescaler

The PRESC bits configure the prescaler division factor. It can be one among the following division factors:

000: /1
001: /2
010: /4
011: /8
100: /16
101: /32
110: /64
111: /128

Bit 8 Reserved, must be kept at reset value.

Bits 7:6 **TRGFLT[1:0]:** Configurable digital filter for trigger

The TRGFLT value sets the number of consecutive equal samples that should be detected when a level change occurs on an internal trigger before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: any trigger active level change is considered as a valid trigger
01: trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger.
10: trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger.
11: trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **CKFLT[1:0]:** Configurable digital filter for external clock

The CKFLT value sets the number of consecutive equal samples that should be detected when a level change occurs on an external clock signal before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: any external clock signal level change is considered as a valid transition
01: external clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition.
10: external clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition.
11: external clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 2:1 **CKPOL[1:0]:** Clock Polarity

When the LPTIM is clocked by an external clock source, CKPOL bits is used to configure the active edge or edges used by the counter:

00: the rising edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 1 is active.

01: the falling edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 2 is active.

10: both edges are active edges. When both external clock signal edges are considered active ones, the LPTIM must also be clocked by an internal clock source with a frequency equal to at least four times the external clock frequency.
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 3 is active.

11: not allowed

Refer to *Section 50.4.15: 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').

50.7.11 LPTIM control register (LPTIM_CR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
											rw	rs	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RSTARE:** Reset after read enable

This bit is set and cleared by software. When RSTARE is set to '1', any read access to LPTIM_CNT register asynchronously resets LPTIM_CNT register content.
This bit can be set only when the LPTIM is enabled.

Bit 3 **COUNTRST:** Counter reset

This bit is set by software and cleared by hardware. When set to '1' this bit triggers a synchronous reset of the LPTIM_CNT counter register. Due to the synchronous nature of this reset, it only takes place after a synchronization delay of 3 LPTimer core clock cycles (LPTimer core clock may be different from APB clock).
This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Caution: COUNTRST must never be set to '1' by software before it is already cleared to '0' by hardware. Software should consequently check that COUNTRST bit is already cleared to '0' before attempting to set it to '1'.

Bit 2 **CNTSTRT:** Timer start in Continuous mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = ‘00’), setting this bit starts the LPTIM in Continuous mode.

If the software start is disabled (TRIGEN[1:0] different than ‘00’), setting this bit starts the timer in Continuous mode as soon as an external trigger is detected.

If this bit is set when a single pulse mode counting is ongoing, then the timer does not stop at the next match between the LPTIM_ARR and LPTIM_CNT registers and the LPTIM counter keeps counting in Continuous mode.

This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Bit 1 **SNGSTRT:** LPTIM start in Single mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = ‘00’), setting this bit starts the LPTIM in single pulse mode.

If the software start is disabled (TRIGEN[1:0] different than ‘00’), setting this bit starts the LPTIM in single pulse mode as soon as an external trigger is detected.

If this bit is set when the LPTIM is in continuous counting mode, then the LPTIM stops at the following match between LPTIM_ARR and LPTIM_CNT registers.

This bit can only be set when the LPTIM is enabled. It is automatically reset by hardware.

Bit 0 **ENABLE:** LPTIM enable

The ENABLE bit is set and cleared by software.

0: LPTIM is disabled. Writing '0' to the ENABLE bit resets all the DMA request signals (input capture and update event DMA requests).

1: LPTIM is enabled

50.7.12 LPTIM compare register 1 (LPTIM_CCR1)

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
CCR1[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 **CCR1[15:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the capture/compare 1 register.

Depending on the PRELOAD option, the CCR1 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PREOAD bit is reset.

The capture/compare register 1 contains the value to be compared to the counter LPTIM_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 becomes read-only, it contains the counter value transferred by the last input capture 1 event. The LPTIM_CCR1 register is read-only and cannot be programmed.

If LPTIM does not implement any channel:

The compare register 1 contains the value to be compared to the counter LPTIM_CNT and signaled on LPTIM output.

Caution: The LPTIM_CCR1 register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).

50.7.13 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 CCRx[15:0] value.

Caution: The LPTIM_ARR register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).

50.7.14 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.

50.7.15 LPTIM configuration register 2 (LPTIM_CFGR2)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2SEL[1:0]		Res.	Res.	IC1SEL[1:0]	
										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.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
										rw	rw			rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:20 **IC2SEL[1:0]**: LPTIM input capture 2 selection

The IC2SEL bits control the LPTIM Input capture 2 multiplexer, which connects LPTIM Input capture 2 to one of the available inputs.
00: lptim_ic2_mux0
01: lptim_ic2_mux1
10: lptim_ic2_mux2
11: lptim_ic2_mux3
For connection details refer to *Section 50.4.3: LPTIM input and trigger mapping*.

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **IC1SEL[1:0]**: LPTIM input capture 1 selection

The IC1SEL bits control the LPTIM Input capture 1 multiplexer, which connects LPTIM Input capture 1 to one of the available inputs.
00: lptim_ic1_mux0
01: lptim_ic1_mux1
10: lptim_ic1_mux2
11: lptim_ic1_mux3
For connection details refer to *Section 50.4.3: LPTIM input and trigger mapping*.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:4 **IN2SEL[1:0]**: LPTIM input 2 selection

The IN2SEL bits control the LPTIM input 2 multiplexer, which connects LPTIM input 2 to one of the available inputs.
00: lptim_in2_mux0
01: lptim_in2_mux1
10: lptim_in2_mux2
11: lptim_in2_mux3
For connection details refer to *Section 50.4.3: LPTIM input and trigger mapping*.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **IN1SEL[1:0]**: LPTIM input 1 selection

The IN1SEL bits control the LPTIM input 1 multiplexer, which connects LPTIM input 1 to one of the available inputs.
00: lptim_in1_mux0
01: lptim_in1_mux1
10: lptim_in1_mux2
11: lptim_in1_mux3
For connection details refer to *Section 50.4.3: LPTIM input and trigger mapping*.

50.7.16 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.

50.7.17 LPTIM capture/compare mode register 1 (LPTIM_CCMR1)

Address offset: 0x02C

Reset value: 0x0000 0000

The channels can be used in input (capture mode) or in output (PWM mode). The direction of a channel is defined by configuring the corresponding CCxSEL bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC2F[1:0]		Res.	Res.	IC2PSC[1:0]		Res.	Res.	Res.	Res.	CC2P[1:0]		CC2E	CC2 SEL
		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.	IC1F[1:0]		Res.	Res.	IC1PSC[1:0]		Res.	Res.	Res.	Res.	CC1P[1:0]		CC1E	CC1 SEL
		rw	rw			rw	rw					rw	rw	rw	rw

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

Bits 29:28 **IC2F[1:0]**: Input capture 2 filter

This bitfield defines the number of consecutive equal samples that should be detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

00: any external input capture signal level change is considered as a valid transition

01: external input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: external input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: external input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:24 **IC2PSC[1:0]**: Input capture 2 prescaler

This bitfield defines the ratio of the prescaler acting on the CC2 input (IC2).

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 23:20 Reserved, must be kept at reset value.

Bits 19:18 **CC2P[1:0]**: Capture/compare 2 output polarity.

Condition: CC2 as output

Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.

0: OC2 active high

1: OC2 active low

Condition: CC2 as input

This field is used to select the IC2 polarity for capture operations.

00: rising edge, circuit is sensitive to IC2 rising edge

01: falling edge, circuit is sensitive to IC2 falling edge

10: reserved, do not use this configuration.

11: both edges, circuit is sensitive to both IC2 rising and falling edges.

Bit 17 **CC2E**: Capture/compare 2 output enable.

Condition: CC2 as output

0: Off - OC2 is not active. Writing '0' to the CC2E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.

1: On - OC2 signal is output on the corresponding output pin

Condition: CC2 as input

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 2 (LPTIM_CCR2) or not.

0: Capture disabled. Writing '0' to the CC2E bit resets the associated ic2_dma_req signal.

1: Capture enabled.

Bit 16 **CC2SEL**: Capture/compare 2 selection

This bitfield defines the direction of the channel, input (capture) or output mode.

0: CC2 channel is configured in output PWM mode

1: CC2 channel is configured in input capture mode

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **IC1F[1:0]**: Input capture 1 filter

This bitfield defines the number of consecutive equal samples that should be detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

00: any external input capture signal level change is considered as a valid transition

01: external input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: external input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: external input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on the CC1 input (IC1).

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

Bits 3:2 **CC1P[1:0]**: Capture/compare 1 output polarity.

Condition: CC1 as output

Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.

0: OC1 active high, the LPTIM output reflects the compare results between LPTIM_ARR and LPTIM_CCRx registers

1: OC1 active low, the LPTIM output reflects the inverse of the compare results between LPTIM_ARR and LPTIM_CCRx registers

Condition: CC1 as input

This field is used to select the IC1 polarity for capture operations.

00: rising edge, circuit is sensitive to IC1 rising edge

01: falling edge, circuit is sensitive to IC1 falling edge

10: reserved, do not use this configuration.

11: both edges, circuit is sensitive to both IC1 rising and falling edges.

Bit 1 **CC1E**: Capture/compare 1 output enable.

Condition: CC1 as output

0: Off - OC1 is not active. Writing '0' to the CC1E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.

1: On - OC1 signal is output on the corresponding output pin

Condition: CC1 as input

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (LPTIM_CCR1) or not.

0: Capture disabled. Writing '0' to the CC1E bit resets the associated ic1_dma_req signal.

1: Capture enabled.

Bit 0 **CC1SEL**: Capture/compare 1 selection

This bitfield defines the direction of the channel input (capture) or output mode.

0: CC1 channel is configured in output PWM mode

1: CC1 channel is configured in input capture mode

Caution: After a write to the LPTIM_CCMRx register, a new write operation to the same register can only be performed after a delay that must be equal or greater than the value of (PRESC × 3)

kernel clock cycles, PRESC[2:0] being the clock decimal division factor (1, 2, 4,..128). Any successive write violating this delay, leads to unpredictable results.

Caution: The CCxSEL, ICxF[1:0], CCxP[1:0] and ICxPSC[1:0] fields must only be modified when the channel x is disabled (CCxE bit reset to 0).

If LPTIM does not implement any channel this register is reserved. Refer to Section 50.3.

50.7.18 LPTIM compare register 2 (LPTIM_CCR2)

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
CCR2[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 **CCR2[15:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the capture/compare 2 register.

Depending on the PRELOAD option, the CCR2 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PREOAD bit is reset.

The capture/compare register 2 contains the value to be compared to the counter LPTIM_CNT and signaled on OC2 output.

If channel CC2 is configured as input:

CCR2 becomes read-only, it contains the counter value transferred by the last input capture 2 event. The LPTIM_CCR2 register is read-only and cannot be programmed.

Caution: The LPTIM_CCR2 register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).

Note: If the LPTIM implements less than 2 channels this register is reserved. Refer to *Section 50.3: LPTIM implementation*.

50.7.19 LPTIM register map

The following table summarizes the LPTIM registers.

Table 506. 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	LPTIM4_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOK	UE	DOWN(2)	UP(2)	ARROK	CMP1OK	EXTTRIG	ARRM	CC1IF
	Reset value								0																0	0	0	0	0	0	0	0	0
0x000	LPTIMx_ISR (x = 1 to 3) Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	CMP2OK(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2IF(1)	REPOK	UE	DOWN(2)	UP(2)	ARROK	CMP1OK	EXTTRIG	ARRM	CC1IF
	Reset value								0					0										0	0	0	0	0	0	0	0	0	0
	LPTIMx_ISR (x = 1 to 3) Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF(1)	CC1OF	Res.	Res.	CC2IF(1)	REPOK	UE	DOWN(2)	UP(2)	ARROK	Res.	EXTTRIG	ARRM	CC1IF
	Reset value								0											0	0			0	0	0	0	0	0		0	0	0
0x004	LPTIM4_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKCF	UECF	DOWNCF(2)	UPCF(2)	ARROKCF	CMP1OKCF	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0																0	0	0	0	0	0	0	0	0
0x004	LPTIMx_ICR (x = 1 to 3) Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	CMP2OKCF(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2CF(1)	REPOKCF	UECF	DOWNCF(2)	UPCF(2)	ARROKCF	CMP1OKCF	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0					0										0	0	0	0	0	0	0	0	0	0
	LPTIMx_ICR (x = 1 to 3) Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OCF(1)	CC1OCF	Res.	Res.	CC2CF(1)	REPOKCF	UECF	DOWNCF(2)	UPCF(2)	ARROKCF	Res.	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0											0	0			0	0	0	0	0	0		0	0	0
0x008	LPTIM4_DIER	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(2)	UPIE(2)	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
	Reset value																								0	0	0	0	0	0	0	0	0

Table 506. 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
0x008	LPTIMx_DIER (x = 1 to 3) Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UEDE	Res.	Res.	Res.	CMP2OKIE(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2IE(1)	REPOKIE	UEIE	DOWNIE(2)	UPIE(2)	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
	Reset value									0				0										0	0	0	0	0	0	0	0	0	0
	LPTIMx_DIER (x = 1 to 3) Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	CC2DE(1)	Res.	UEDE	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	Res.	Res.	CC2OIE(1)	CC1OIE	Res.	Res.	CC2IE(1)	REPOKIE	UEIE	DOWNIE(2)	UPIE(2)	ARROKIE	Res.	EXTTRIGIE	ARRMIE	CC1IE
	Reset value							0		0							0			0	0			0	0	0	0	0	0		0	0	0
0x00C	LPTIM_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC(2)	COUNTMODE	PRELOAD	WAVPOL(3)	WAVE	TIMOUT	TRIGEN		Res.	TRIGSEL[2:0]			Res.	PRESC			Res.	TRGFLT		Res.	CKFLT		CKPOL		CKSEL
	Reset value								0	0	0	0	0	0	0	0		0	0	0		0	0	0		0	0		0	0	0	0	0	0
0x010	LPTIM_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
	Reset value																												0	0	0	0	0
0x014	LPTIM_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
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
0x024	LPTIM_ CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2SEL[1:0]		Res.	Res.	IC1SEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
	Reset value											0	0			0	0											0	0			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
0x02C	LPTIM_CCMR1 (4)	Res.	Res.	IC2F[1:0]		Res.	Res.	IC2PSC[1:0]		Res.	Res.	Res.	Res.	CC2P[1:0]		CC2E	CC2SEL	Res.	Res.	IC1F[1:0]		Res.	Res.	IC1PSC[1:0]		Res.	Res.	Res.	Res.	CC1P[1:0]		CC1E	CC1SEL
	Reset value			0	0			0	0					0	0	0	0			0	0			0	0					0	0	0	0

Table 506. 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
0x034	LPTIM_CCR2(5)	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

1. If LPTIM does not implement at least 2 channels this bit is reserved. Refer to *Section 50.3: LPTIM implementation*.
2. *If LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 50.3: LPTIM implementation.*
3. If the LPTIM implements at least one capture/compare channel, this bit is reserved. Refer to *Section 50.3: LPTIM implementation.*
4. If LPTIM does not implement any channel this register is reserved. Refer to *Section 50.3: LPTIM implementation*.
5. If the LPTIM implements less than 2 channels this register is reserved. Refer to *Section 50.3: LPTIM implementation*.

Refer to *Section 2.3 on page 115* for the register boundary addresses.

51 Infrared interface (IRTIM)

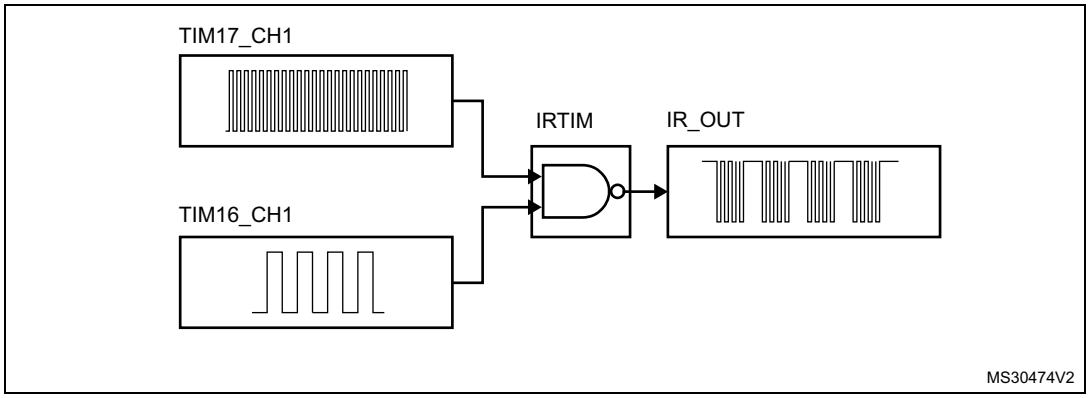
An infrared interface (IRTIM) for remote control is available on the device. It can be used with an infrared LED to perform remote control functions.

It uses internal connections withTIM16 and TIM17 as shown in *Figure 615*.

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 615. 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 PB9_FMP bit in the SYSCFG_CFGR1 register and used to sink the high current needed to directly control an infrared LED.

52 Independent watchdog (IWDG)

52.1 Introduction

The independent watchdog (IWDG) peripheral offers a high safety level, thanks to its capability to detect malfunctions due to software or hardware failures.

The IWDG is clocked by an independent clock, and stays active even if the main clock fails.

In addition, the watchdog function is performed in the V_{DD} voltage domain, allowing the IWDG to remain functional even in low-power modes. Refer to *Section 52.3* to check the capability of the IWDG in this product.

The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, making it very reliable to detect any unexpected behavior.

52.2 IWDG main features

- 12-bit down-counter
- Dual voltage domain, thus enabling operation in low-power modes
- Independent clock
- Early wake-up interrupt generation
- Reset generation
  - In case of timeout
  - In case of refresh outside the expected window

52.3 IWDG implementation

Table 507. STM32U575/585 IWDG features(1)

IWDG modes/features	IWDG
LSI used as IWDG kernel clock (iwdg_ker_ck)	X
Window function	X
Early wakeup interrupt generation	X
System reset generation(2)	X
Capability to work in system Stop	X
Capability to work in system Standby	-
Capability to generate an interrupt in system Stop(3)	X
Capability to generate an interrupt in system Standby	-
Capability to be frozen when the microcontroller enters in debug mode(4)	X
Option bytes to control the activity in Stop mode(5)	X
Option bytes to control the activity in Standby mode(6)	X
Option bytes to control the hardware mode(7)	X

1. ‘X’ = supported, ‘-’ = not supported.
2. Refer to the RCC section for additional information.

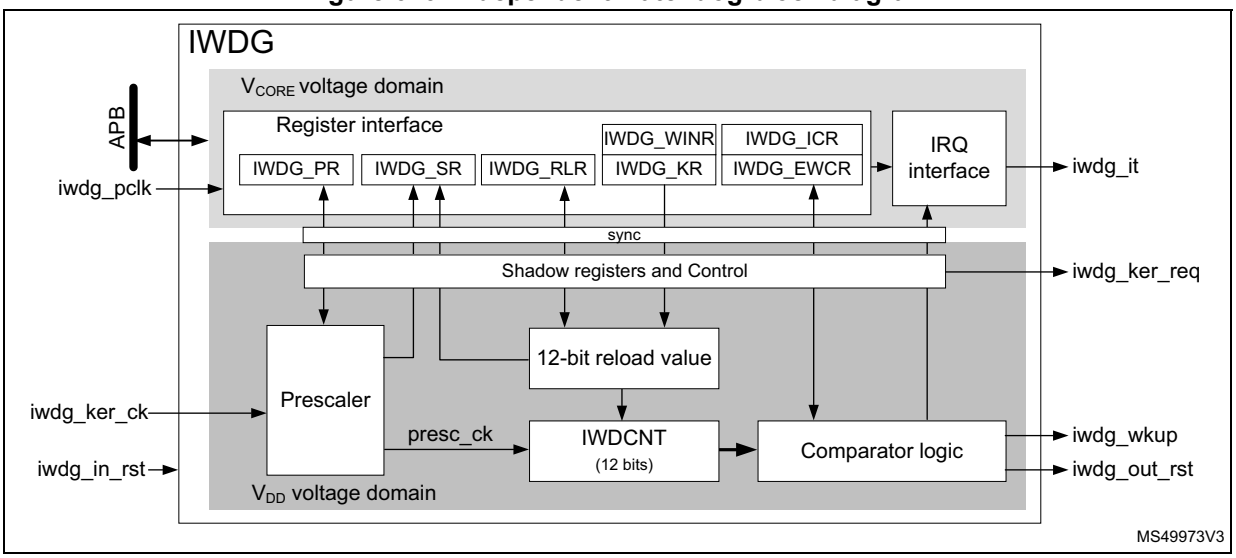
3. Wake-up from Stop with interrupt is supported only in Stop 0, Stop 1, and Stop 2 modes.
4. Controlled via DBG_IWDG_STOP in DBG section.
5. Controlled via the option byte IWDG_STOP in FLASH section.
6. Controlled via the option byte IWDG_STDBY in FLASH section.
7. Controlled via the option byte IWDG_SW in FLASH section.

52.4 IWDG functional description

52.4.1 IWDG block diagram

Figure 616 shows the functional blocks of the independent watchdog module.

Figure 616. Independent watchdog block diagram



The register and IRQ interfaces are located into the V_{CORE} voltage domain. The watchdog function itself is located into the V_{DD} voltage domain to remain functional in low-power modes. See *Section 52.3* for IWDG capabilities.

The register and IRQ interfaces are mainly clocked by the APB clock (iwdg_pclk), while the watchdog function is clocked by a dedicated kernel clock (iwdg_ker_ck). A synchronization mechanism makes the data exchange between the two domains possible. Note that most of the registers located in the register interface are shadowed into the V_{DD} voltage domain.

The IWDG down-counter (IWDCNT) is clocked by the prescaled clock (presc_ck). The prescaled clock is generated from the kernel clock iwdg_ker_ck divided by the prescaler, according to PR[3:0] bitfield.

52.4.2 IWDG internal signals

The list of IWDG internal signals is detailed in *Table 508*.

Table 508. IWDG internal input/output signals

Signal name	Signal type	Description
iwdg_ker_ck	Input	IWDG kernel clock
iwdg_ker_req	Input	IWDG kernel clock request
iwdg_pclk	Input	IWDG APB clock
iwdg_out_rst	Output	IWDG reset output
iwdg_in_rst	Input	IWDG reset input
iwdg_wkup	Output	IWDG wakeup event
iwdg_it	Output	IWDG early wakeup interrupt

52.4.3 Software and hardware watchdog modes

The watchdog modes allow the application to select the way the IWDG is enabled, either by software commands (Software watchdog mode), or automatically (Hardware watchdog mode). All other functions work similarly for both Software and Hardware modes.

The Software watchdog mode is the default working mode. The independent watchdog is started by writing the value 0x0000 CCCC into the *IWDG key register (IWDG_KR)*, and the IWDCNT starts counting down from the reset value (0xFFF).

In the Hardware watchdog mode the independent watchdog is started automatically at power-on, or every time it is reset (via iwdg_in_rst). The IWDCNT down-counter starts counting down from the reset value 0xFFF. The hardware watchdog mode feature is enabled through the device option bits, see *Section 52.3* for details.

When the IWDCNT reaches 0x000, a reset signal is generated (i.e. iwdg_out_rst asserted).

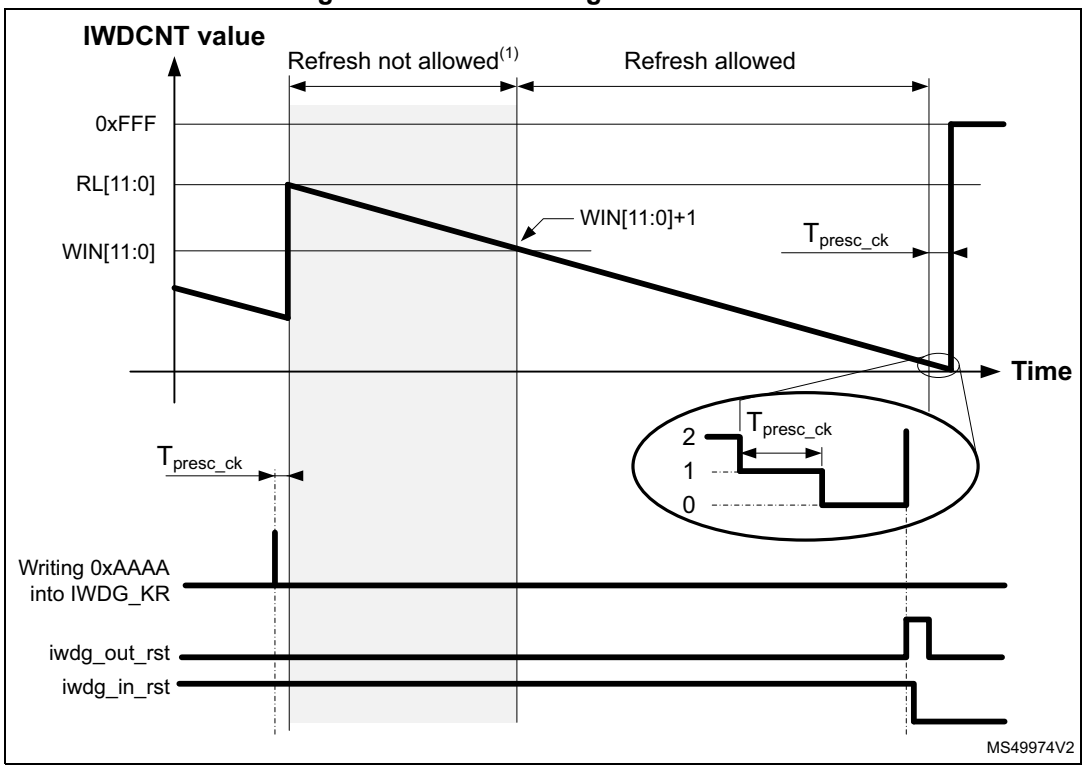
Whenever the key value 0x0000 AAAA is written in the *IWDG key register (IWDG_KR)*, the IWDG_RLR value is reloaded into the IWDCNT and the watchdog reset is prevented.

Due to re-synchronization delays, the IWDG must be refreshed before the IWDCNT down-counter reaches 1.

Once started, the IWDG can only be stopped when it is reset (i.e. iwdg_in_rst asserted).

As shown in *Figure 617*, when the refresh command is executed, one period of presc_ck later, the IWDCNT is reloaded with the content of RL[11:0].

Figure 617. Reset timing due to timeout



1. If window option activated.

If the IWDG is not refreshed before the IWDCNT reaches 1, then the IWDG generates a reset (i.e. iwdg_out_rst asserted). In return, the RCC resets the IWDG (assertion of iwdg_in_rst) to clear the reset source.

52.4.4 Window option

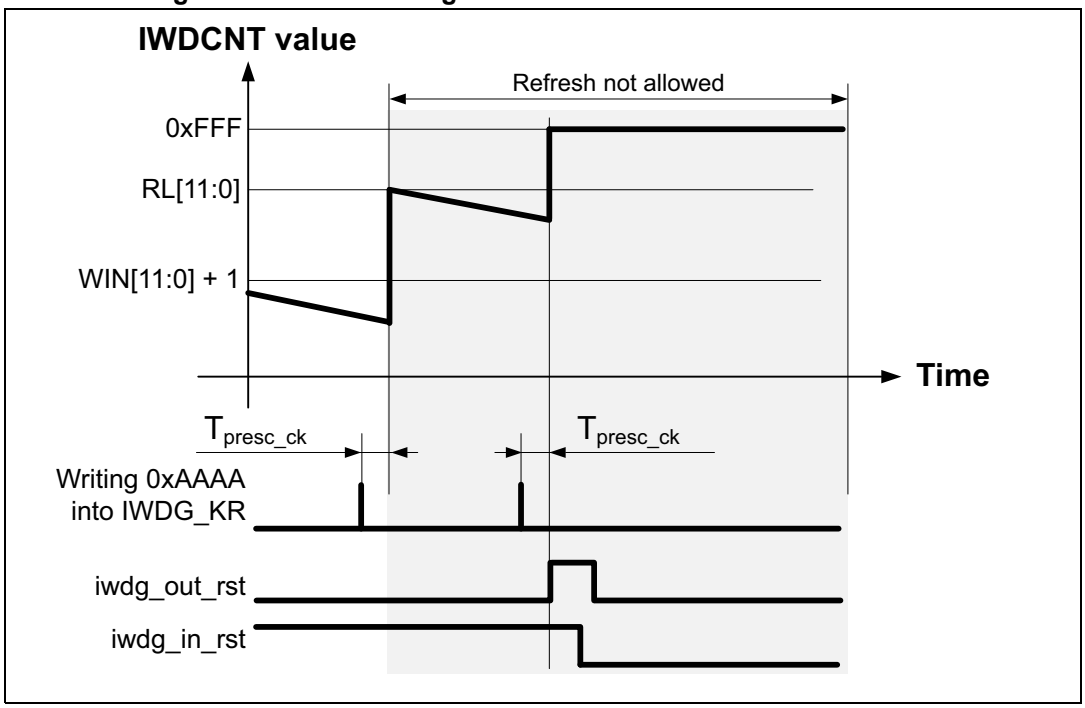
The IWDG can also work as a window watchdog by setting the appropriate window in the *IWDG window register (IWDG_WINR)*.

If the reload operation is performed while the counter is greater than WIN[11:0] + 1 a reset is generated. WIN[11:0] is located in the *IWDG window register (IWDG_WINR)*. As shown in *Figure 618*, the reset is generated one period of presc_ck after the unexpected refresh command.

The default value of the *IWDG window register (IWDG_WINR)* is 0x0000 0FFF, so, if not updated, the window option is disabled.

As soon as the window value changes, the down-counter (IWDCNT) is reloaded with the RL[11:0] value to ease the estimation for where the next refresh must take place.

Figure 618. Reset timing due to refresh in the not allowed area



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)*.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. If needed, enable the early wakeup interrupt, and program the early wakeup comparator, by writing the proper values into the *IWDG early wakeup interrupt register (IWDG_EWCR)*.
6. Write to the *IWDG window register (IWDG_WINR)*. This automatically reloads the IWDCNT down-counter with the RL[11:0] value.
7. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
8. Write 0x0000 0000 into *IWDG key register (IWDG_KR)* to write-protect registers.

Note: Step 7 can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

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)*.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. If needed, enable the early wakeup interrupt, and program the early wakeup comparator, by writing the proper values into the *IWDG early wakeup interrupt register (IWDG_EWCR)*.
6. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
7. Refresh the counter with RL[11:0] value, and write-protect registers by writing 0x0000 AAAA into *IWDG key register (IWDG_KR)*.

Updating the window comparator

It is possible to update the window comparator when the IWDG is already running. The IWDCNT is reloaded as well. The following sequence can be performed to update the window comparator:

1. Enable register access by writing 0x0000 5555 in the IWDG key register (IWDG_KR).
2. Write to the IWDG window register (IWDG_WINR). This automatically reloads the IWDCNT down-counter with RL[11:0] value.
3. Wait for WVU = 0
4. Lock registers by writing IWDG_KR to 0x0000 0000

Note that the step *3* can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

52.4.5 Debug

When the processor enters into Debug mode (core halted), the IWDCNT down-counter either continues to work normally or stops, depending on debug capability of the product. Refer to *Section 52.3* for details on the capabilities of this product.

52.4.6 Register access protection

Write accesses to *IWDG prescaler register (IWDG_PR)*, *IWDG reload register (IWDG_RLR)*, *IWDG early wakeup interrupt register (IWDG_EWCR)* and *IWDG window register (IWDG_WINR)* are protected. To modify them, first write 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 the down-counter reload value or the window value is ongoing.

52.5 IWDG low-power modes

Depending on option bytes configuration, the IWDG can continue counting or not during the low-power modes. Refer to *Section 52.3* for details.

Table 509. Effect of low-power modes on IWDG

Mode	Description
Sleep	No effect. IWDG interrupts cause the device to exit the Sleep mode.
Stop	The IWDG remains active or not, depending on option bytes configuration. Refer to *Section 52.3* for details. IWDG interrupts cause the device exit the Stop mode.
Standby	The IWDG remains active or not, depending on option bytes configuration. Refer to *Section 52.3* for details. IWDG interrupts do not make the device to exit from Standby mode.
Shutdown	The IWDG is not working.

52.6 IWDG interrupts

The IWDG offers the possibility to generate an early interrupt depending on the value of the down-counter. The early interrupt is enabled by setting the EWIE bit of the *IWDG early wakeup interrupt register (IWDG_EWCR)* to 1.

A comparator value (EWIT[11:0]) allows the application to define at which position the early interrupt must be generated.

When the IWDCNT down-counter reaches the value of EWIT[11:0] - 1, the iwdg_wkup is activated, making it possible for the system to exit from low-power mode if needed.

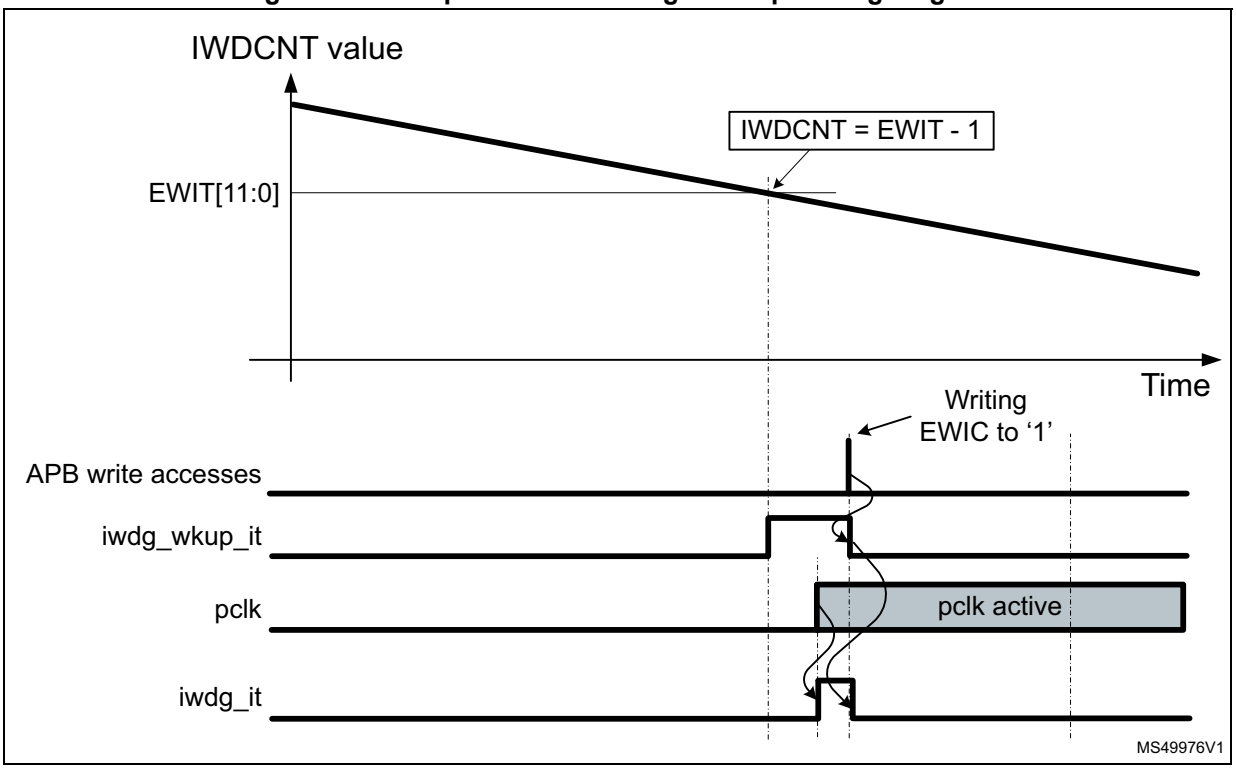
When the APB clock is available, the iwdg_it is activated as well.

In addition the flag EWIF of the *IWDG status register (IWDG_SR)* is set to 1.

The EWI interrupt is acknowledged by writing '1' to the EWIC bit in the *IWDG early wakeup interrupt register (IWDG_EWCR)*.

Writing into the IWDG_EWCR register also triggers a refresh of the down-counter (IWDCNT) with the reload value RL[11:0].

Figure 619. Independent watchdog interrupt timing diagram



The Early Wakeup Interrupt (EWI) can be used if specific safety operations or data logging must be performed before the watchdog reset is generated.

Changing the early wakeup comparator value

It is possible to change the early wakeup comparator value or to enable/disable the interrupt generation at any time, by performing the following sequence:

1. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
2. Enable or disable the early wakeup interrupt, and/or program the early wakeup comparator, by writing the proper values into the *IWDG early wakeup interrupt register (IWDG_EWCR)*.
3. Wait for EWU = ‘0’, EWU is located into the *IWDG status register (IWDG_SR)*.
4. Write-protect registers by writing 0x0000 0000 to *IWDG key register (IWDG_KR)*.

Step *3* can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

Table 510 summarizes the IWDG interrupt request.

Table 510. IWDG interrupt request

Interrupt event	Event flag	Interrupt clear method	Interrupt enable control bit	Activated interrupt	
				iwdg_it	iwdg_wkup_it
IWDCNT reaches EWIT value	EWIF	Writing EWIC to ‘1’	EWIE	Y(1)	Y(2)

1. Generated when a clock is present on iwdg_pclk input.

2. Generated when a clock is present on iwdg_ker_ck input.

52.7 IWDG registers

Refer to *Section 1.2 on page 104* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

Most of the registers located into the register interface are shadowed into the V_{DD} voltage domain. When the iwdg_in_rst is asserted, the watchdog logic and the shadow registers located into the V_{DD} voltage domain are reset.

When the application reads back a watchdog register, the hardware transfers the value of the corresponding shadow register to the register interface.

When the application writes a watchdog register, the hardware updates the corresponding shadow register.

52.7.1 IWDG key register (IWDG_KR)

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
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 can be used for several functions, depending upon the value written by the application:
- 0xAAAA: reloads the RL[11:0] value into the IWDCNT down-counter (watchdog refresh), and write-protects registers. This value must be written by software at regular intervals, otherwise the watchdog generates a reset when the counter reaches 0.
- 0x5555: enables write-accesses to the registers.
- 0xCCCC: enables the watchdog (except if the hardware watchdog option is selected) and write-protects registers.
- values different from 0X5555: write-protects registers.

Note that only IWDG_PR, IWDG_RLR, IWDG_EWCR and IWDG_WINR registers have a write-protection mechanism.

52.7.2 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.	PR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PR[3:0]:** Prescaler divider

These bits are write access protected, see *Section 52.4.6*. 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 to be able to change the prescaler divider.

0000: divider / 4
0001: divider / 8
0010: divider / 16
0011: divider / 32
0100: divider / 64
0101: divider / 128
0110: divider / 256
0111: divider / 512
Others: divider / 1024

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.

52.7.3 IWDG reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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 *Section 52.4.6*. 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 prescaler.clock. It is not recommended to set RL[11:0] to a value lower than 2.

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, hence the value read from this register is valid only when the RVU bit in the IWDG status register (IWDG_SR) is reset.

52.7.4 IWDG status register (IWDG_SR)

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.	EWIF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWU	WVU	RVU	PVU
	r											r	r	r	r

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

Bit 14 **EWIF**: Watchdog early interrupt flag

This bit is set to ‘1’ by hardware in order to indicate that an early interrupt is pending. This bit must be cleared by the software by writing the bit EWIC of IWDG_EWCR register to ‘1’.

Bits 13:4 Reserved, must be kept at reset value.

Bit 3 **EWU**: Watchdog interrupt comparator value update

This bit is set by hardware to indicate that an update of the interrupt comparator value (EWIT[11:0]) or an update of the EWIE is ongoing. It is reset by hardware when the update operation is completed in the V_{DD} voltage domain (takes up to three periods of the IWDG kernel clock iwdg_ker_ck).

The EWIT[11:0] and EWIE fields can be updated only when EWU bit is reset.

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 three periods of the IWDG kernel clock iwdg_ker_ck).

The window value can be updated only when WVU bit is reset.

This bit is generated only if generic “window” = 1.

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 three periods of the IWDG kernel clock iwdg_ker_ck).

The 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 three periods of the IWDG kernel clock iwdg_ker_ck).

The prescaler value can be updated only when PVU bit is reset.

Note: If several reload, prescaler, early interrupt position 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, to wait until WVU bit is reset before changing the window value, and to wait until EWU bit is reset before changing the early interrupt position value. After updating the prescaler and/or the reload/window/early interrupt value, it is not necessary to wait until RVU or PVU or WVU or EWU is reset before continuing code execution, except in case of low-power mode entry.

52.7.5 IWDG window register (IWDG_WINR)

Address offset: 0x10

Reset value: 0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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 52.4.6*.They contain the high limit of the window value to be compared with the downcounter.

To prevent a reset, the IWDCNT downcounter must be reloaded when its value is lower than WIN[11:0] + 1 and greater than 1.

The WVU 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 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.

52.7.6 IWDG early wakeup interrupt register (IWDG_EWCR)

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
EWIE	EWIC	Res.	Res.	EWIT[11:0]											
rw	w			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EWIE**: Watchdog early interrupt enable

Set and reset by software.

0: The early interrupt interface is disabled.

1: The early interrupt interface is enabled.

The EWU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the value of this bit.

Bit 14 **EWIC**: Watchdog early interrupt acknowledge

The software must write a 1 into this bit in order to acknowledge the early wakeup interrupt and to clear the EWIF flag. Writing 0 has not effect, reading this flag returns a 0.

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:0 **EWIT[11:0]:** Watchdog counter window value

These bits are write access protected (see *Section 52.4.6*). They are written by software to define at which position of the IWDCNT down-counter the early wakeup interrupt must be generated. The early interrupt is generated when the IWDCNT is lower or equal to EWIT[11:0] - 1.

EWIT[11:0] must be bigger than 1.

An interrupt is generated only if EWIE = 1.

The EWU 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 Early wakeup comparator value and the Interrupt enable bit from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the EWU bit in the IWDG status register (IWDG_SR) is reset.

52.7.7 IWDG register map

Table 511. 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.	PR[3:0]			
	Reset value																													0	0	0	0

Table 511. IWDG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08	**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.	EWIF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWU	WVU	RVU	PVU
	Reset value																		0											0	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
0x14	**IWDG_EWCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIE	EWIC	Res.	Res.	EWIT[11:0]											
	Reset value																	0	0			0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3 on page 115* for the register boundary addresses.

53 System window watchdog (WWDG)

53.1 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 a reset on expiry of a programmed time period, unless the program refreshes the contents of the down-counter before the T6 bit becomes cleared. A 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.

53.2 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 621*)
- Early wakeup interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40.

53.3 WWDG implementation

Table 512. STM32U575/585 WWDG features(1)

WWDG mode / feature	WWDG
Window function	X
Early wakeup interrupt generation	X
System reset generation(2)	X
Capability to work in system Stop	-
Capability to work in system Standby	-
Capability to be frozen when the microcontroller enters in Debug mode(3)	X
Option bytes to control the Hardware mode(4)	X

1. “X” = supported, “-” = not supported.
2. Refer to the RCC section for additional information.
3. Controlled via DBG_WWDG_STOP in DBG block.

4. Controlled via the option byte WWDG_SW. When WWDG_SW is set in HW mode the WWDG is running as soon as the CPU is in Run or Sleep modes.

53.4 WWDG functional description

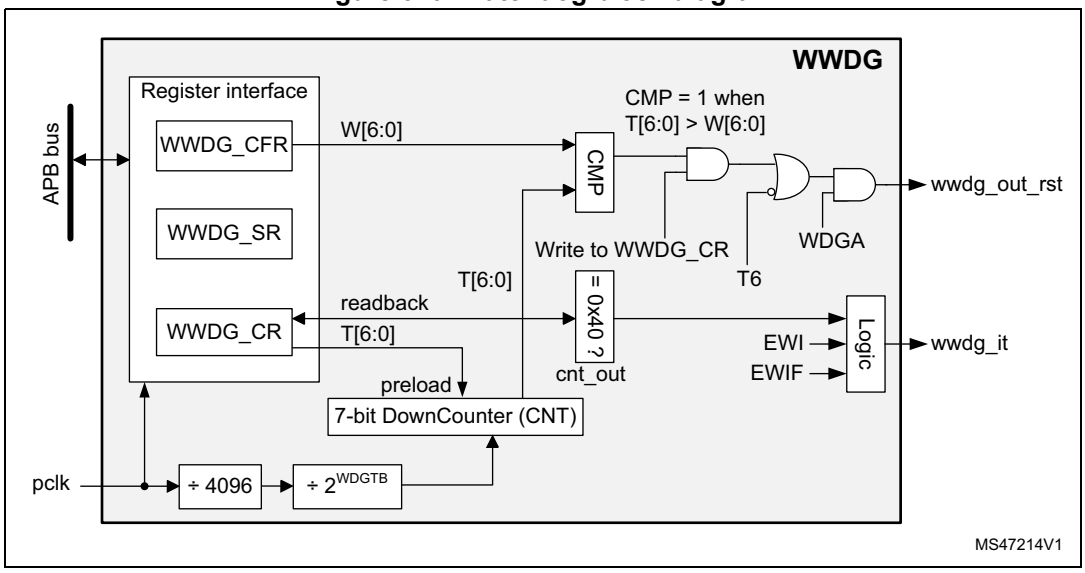
If the watchdog is activated (the WDGA bit is set in the WWDG_CR register) and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent . 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 620* for the WWDG block diagram.

53.4.1 WWDG block diagram

Figure 620. Watchdog block diagram



53.4.2 WWDG internal signals

Table 513 gives the list of WWDG internal signals.

Table 513. WWDG internal input/output signals

Signal name	Signal type	Description
pclk	Digital input	APB bus clock
wwdg_out_rst	Digital output	WWDG reset signal output
wwdg_it	Digital output	WWDG early interrupt output

53.4.3 Enabling the watchdog

When the user option WWDG_SW selects “Software window watchdog”, the watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again except by a reset.

When the user option WWDG_SW selects “Hardware window watchdog”, the watchdog is always enabled after a reset, it cannot be disabled.

53.4.4 Controlling the down-counter

This down-counter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments that represent the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value due to the unknown status of the prescaler when writing to the WWDG_CR register (see *Figure 621*). 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 621* describes the window watchdog process.

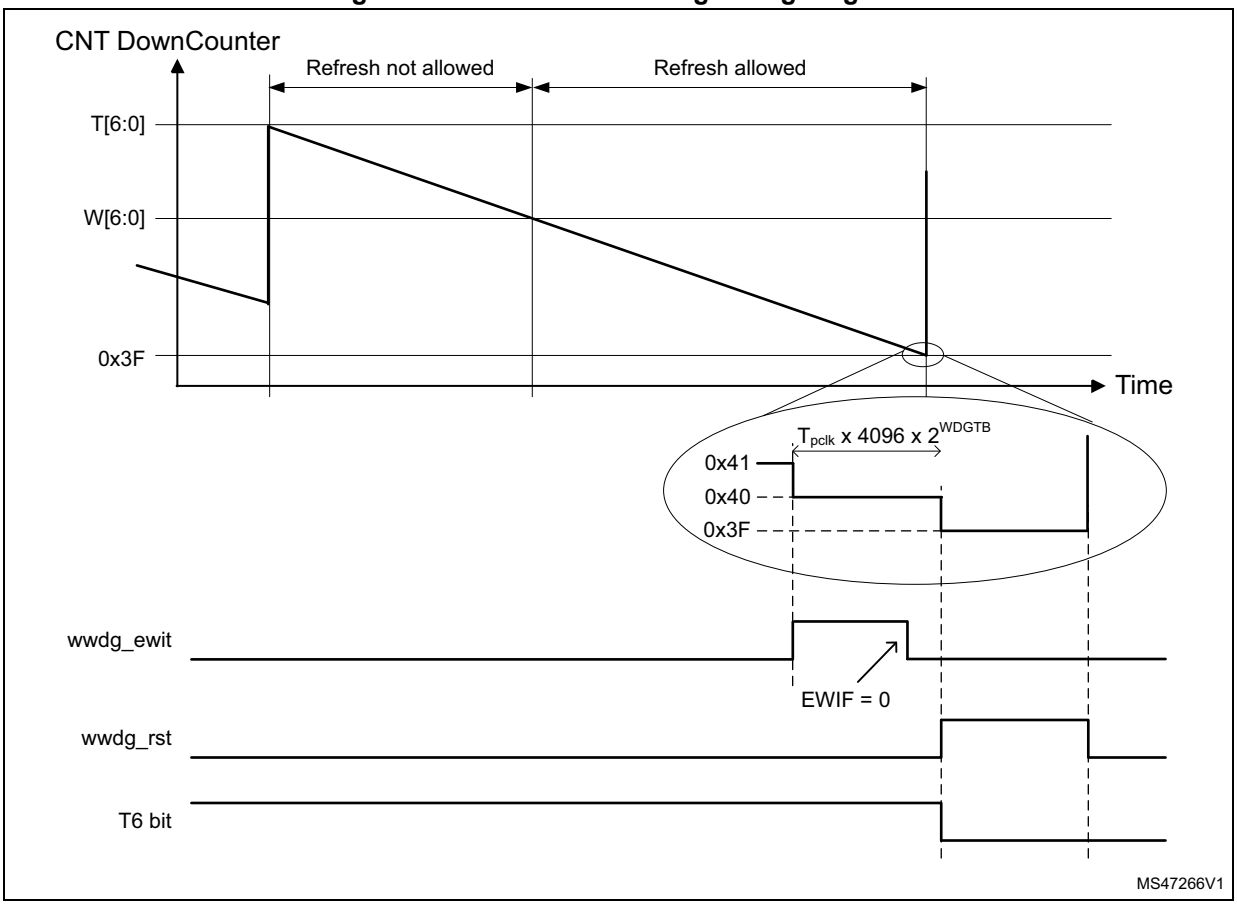
Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

53.4.5 How to program the watchdog timeout

Use the formula in *Figure 621* 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 621. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$t_{WWDG} = t_{PCLK} \times 4096 \times 2^{WDGTB[2:0]} \times (T[5:0] + 1) \quad (ms)$$

where:

t_{WWDG}: WWDG timeout

t_{PCLK}: APB clock period measured in ms

4096: value corresponding to internal divider

As an example, if APB frequency is 48 MHz, WDGTB[2:0] is set to 3 and T[5:0] is set to 63:

$$t_{WWDG} = (1/48000) \times 4096 \times 2^{3} \times (63 + 1) = 43.69ms$$

Refer to the datasheet for the minimum and maximum values of t_{WWDG}.

53.4.6 Debug mode

When the device enters debug mode (processor halted), the WWDG counter either continues to work normally or stops, depending on the configuration bit in DBG module. For more details refer to *Section 65: Debug support (DBG)*.

53.5 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.

Table 514. WWDG interrupt requests

Interrupt		Event flag	Enable control bit	Interrupt clearing method	Exit from mode		
Vector	Event				Sleep	Stop(1)	Standby(1)
WWDG(2)	Early wakeup interrupt	EWIF	EWI	Write EWIF flag to 0	Yes	No	No

1. The WWDG interrupt can have additional capabilities, refer to *Section 53.3* for details.
2. WWDG vector corresponds to the assertion of the wwdg_it signal.

53.6 WWDG registers

Refer to *Section 1.2 on page 104* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by halfwords (16-bit) or words (32-bit).

53.6.1 WWDG control register (WWDG_CR)

Address offset: 0x000

Reset value: 0x0000 007F

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

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

Bit 7 **WDGA:** Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Bits 6:0 **T[6:0]:** 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter, decremented every (4096 x $2^{WDGTB[2:0]}$) PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

53.6.2 WWDG configuration register (WWDG_CFR)

Address offset: 0x004

Reset value: 0x0000 007F

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

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

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:11 **WDGTB[2:0]:** Timer base

The timebase of the prescaler can be modified as follows:

000: CK counter clock (PCLK div 4096) div 1

001: CK counter clock (PCLK div 4096) div 2

010: CK counter clock (PCLK div 4096) div 4

011: CK counter clock (PCLK div 4096) div 8

100: CK counter clock (PCLK div 4096) div 16

101: CK counter clock (PCLK div 4096) div 32

110: CK counter clock (PCLK div 4096) div 64

111: CK counter clock (PCLK div 4096) div 128

Bit 10 Reserved, must be kept at reset value.

Bit 9 **EWI:** Early wakeup interrupt

When set, an interrupt occurs whenever the counter reaches the value 0x40. This interrupt is only cleared by hardware after a reset.

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:0 **W[6:0]:** 7-bit window value

These bits contain the window value to be compared with the down-counter.

53.6.3 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.

53.6.4 WWDG register map

The following table gives the WWDG register map and reset values.

Table 515. WWDG register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**WWDG_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
	Reset value																									0	1	1	1	1	1	1	1
0x004	**WWDG_CFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGTB [2:0]			Res.	EWI	Res.	Res.	W[6:0]						
	Reset value																			0	0	0		0			1	1	1	1	1	1	1
0x008	**WWDG_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
	Reset value																																0

Refer to *Section 2.3 on page 115* for the register boundary addresses.

54 Real-time clock (RTC)

54.1 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.

54.2 RTC main features

The RTC supports the following features (see *Figure 622: RTC block diagram*):

- Calendar with subseconds, seconds, minutes, hours (12 or 24 format), week day, date, month, year, in BCD (binary-coded decimal) format.
- Binary mode with 32-bit free-running counter.
- Automatic correction for 28, 29 (leap year), 30, and 31 days of the month.
- Two programmable alarms.
- On-the-fly correction from 1 to 32767 RTC clock pulses. This can be used to synchronize it with a master clock.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Digital calibration circuit with 0.95 ppm resolution, to compensate for quartz crystal inaccuracy.
- Timestamp feature which can be used to save the calendar content. This function can be triggered by an event on the timestamp pin, or by a tamper event, or by a switch to V_{BAT} mode.
- 17-bit auto-reload wakeup timer (WUT) for periodic events with programmable resolution and period.
- TrustZone support:
  - RTC fully securable
  - Alarm A, alarm B, wakeup Timer and timestamp individual secure or non-secure configuration
- Alarm A, alarm B, wakeup Timer and timestamp individual privilege protection

The RTC is supplied through a switch that takes power either from the V_{DD} supply when present or from the VBAT pin.

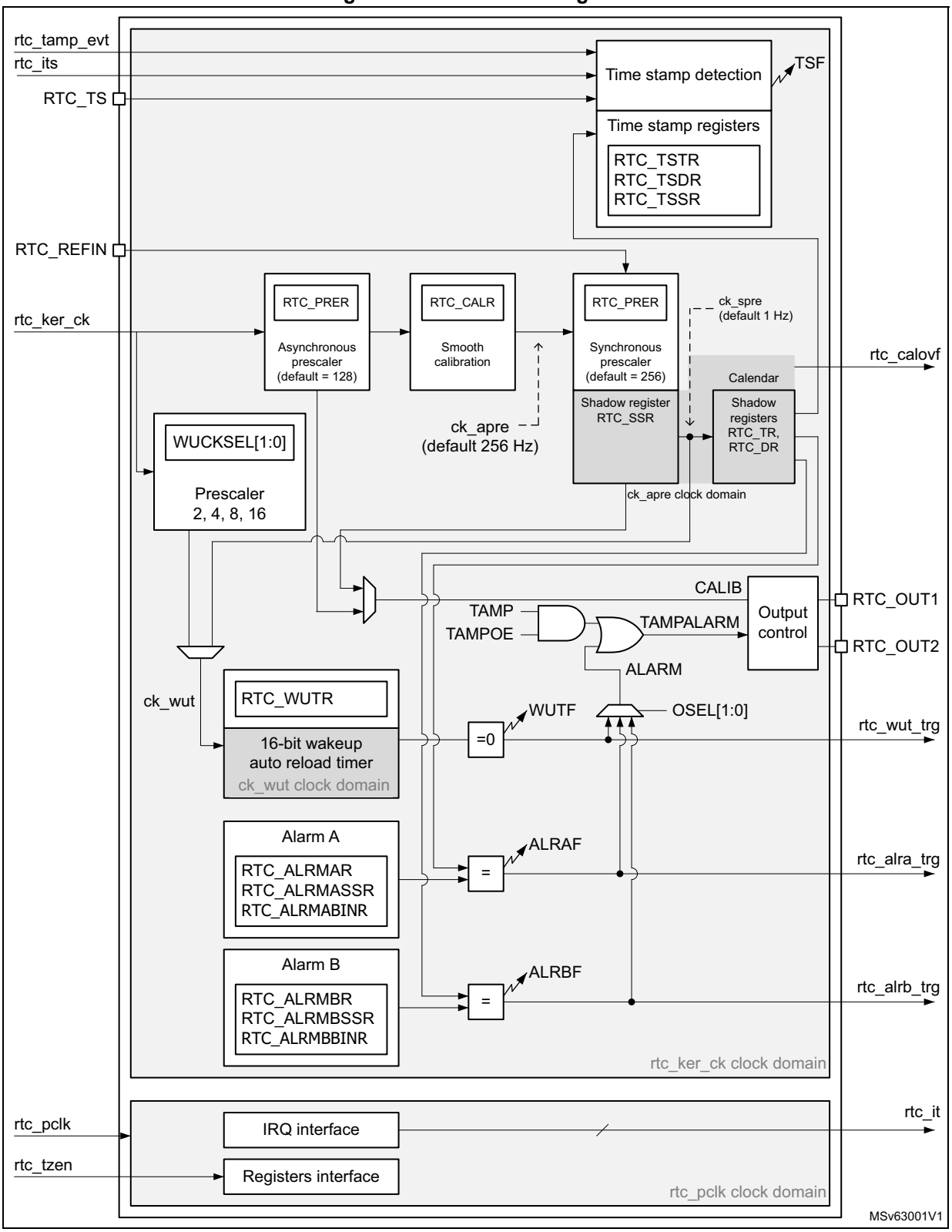
The RTC is functional in V_{BAT} mode and in all low-power modes when it is clocked by the LSE.

All RTC events (Alarm, WakeUp Timer, Timestamp) can generate an interrupt and wakeup the device from the low-power modes.

54.3 RTC functional description

54.3.1 RTC block diagram

Figure 622. RTC block diagram



54.3.2 RTC pins and internal signals

Table 516. RTC input/output pins

Pin name	Signal type	Description
RTC_TS	Input	RTC timestamp input
RTC_REFIN	Input	RTC 50 or 60 Hz reference clock input
RTC_OUT1	Output	RTC output 1
RTC_OUT2	Output	RTC output 2

RTC_OUT1 and RTC_OUT2 which 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 517. 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 54.5: 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 54.4: RTC low-power modes* for more details.

Table 518. 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_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.

54.3.3 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.

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 519*.

Table 519. 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 519. 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 519. 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. The different functions are mapped on RTC_OUT1 or on RTC_OUT2 depending on OSEL, COE and OUT2EN configuration, as shown in table *Table 520*.

Table 520. 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

54.3.4 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 configuration register (RTC_SECCFGR) which can be written in secure mode only. The RTC protection configuration is not affected by a system reset.

When the SEC bit is set in the RTC_SECCFGR register:

- Writing the RTC registers is possible only in secure mode.
- Reading RTC_SECCFGR, RTC_PRIVCFGR, 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 SEC bit is cleared, it is still possible to protect some of the registers by setting dedicated INITSEC, CALSEC, TSSEC, WUTSEC, ALRASEC or ALRBSEC 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 INITSEC is set:
  - RTC_TR, RTC_DR, RTC_PRER registers, plus INIT, BIN and BCDU in RTC_ICSR, FMT control bits in RTC_CR and INITPRIV in the RTC_PRIVCFGR can be written only in secure mode.
  - These registers and control bits can be read in secure and non-secure mode.
- When CALSEC is set:
  - RTC_SHIFTR and RTC_CALR registers, plus ADD1H, SUB1H and REFCKON control bits in the RTC_CR and CALPRIV in the RTC_PRIVCFGR can be written only in secure mode.
  - These registers and control bits can be read in secure and non-secure mode.
- When ALRASEC is set:
  - RTC_ALRMAR, RTC_ALRMASSR and RTC_ALRABINR registers, plus ALRAE, ALRAFCLR, ALRAIE and SSRUIE in the RTC_CR, CALRAF and CSSRUF in the RTC_SCR, ALRAF and SSRUF in RTC_SR, and ALRAMF and SSRUMF in RTC_SMISR can be read and written only in secure mode.
  - ALRAPRIV in the RTC_PRIVCFGR can be written only in secure mode
- When ALRBSEC is set:
  - RTC_ALRMBR, RTC_ALRMBSSR and RTC_ALRBBINR registers, plus ALRBE ALRBFCLR, 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_PRIVCFGR can be written only in secure mode.
- When WUTSEC is set:
  - 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_PRIVCFGR can be written only in secure mode
- When TSSEC 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_SMISR can be read and written only in secure mode.
  - TSPRIV in the RTC_PRIVCFGR 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.

54.3.5 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_PRIVCFGR) 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_PRIVCFGR register:

- Writing the RTC registers is possible only in privileged mode.
- Reading the RTC_SECCFGR, RTC_PRIVCFGR, 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, BIN and BCDU in RTC_ICSR and FMT control bits in RTC_CR, plus INITSEC in the RTC_SECCFGR 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 CALDSEC in the RTC_SECCFGR 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 ALRAFCLR, ALRAIE and SSRUIE in the RTC_CR, and CALRAF and CSSRUF in the RTC_SCR, ALRAF and SSRUF in RTC_SR, and ALRAMF and SSRUMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - ALRASEC in the RTC_SECCFGR can be written only in privilege mode.
- When ALRBPRIV is set:
  - RTC_ALRMBR, RTC_ALRMBSSR and RTC_ALRBBINR registers, plus ALRBE ALRBFCLR, 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.

  - ALRBSEC in the RTC_SECCFGR 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.
  - WUTSEC in the RTC_SECCFGR 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.
  - TSSEC in the RTC_SECCFGR 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.

54.3.6 Clock and prescalers

For more information on the RTC clock (RTCCLK) source configuration, refer to “Reset and clock control (RCC)”.

BCD mode (BIN=00)

A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into 2 programmable prescalers (see *Figure 622: 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 subsecond 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 54.3.10: Periodic auto-wakeup* for details).

Binary mode (BIN=01)

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are not functional.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.

PREDIV_S value is don't care.

Mixed mode (BIN=10 or 11)

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are also available.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register. The bits BCDU[2:0] are used to define when the calendar is incremented by 1 second, using the SSR least significant bits.

54.3.7 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 54.6.12: 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.

54.3.8 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 only instead of both RTCCLK and ck_apre. Consequently, some flags delays are longer, and the calibration window is longer (refer to *Section : RTC 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.

54.3.9 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.

When the binary mode is used, the subsecond field can be programmed in the alarm binary register RTC_ALRMABINR.

The alarm interrupt is enabled through the ALRAIE bit in the RTC_CR register.

In case the Alarm is used to generate a trigger event for another peripheral, the ALRAF can be automatically cleared by hardware by configuring the ALRAFCLR bit at 1 in the RTC_CR register. In this configuration there is no need for software intervention if the only purpose is clearing the ALRAF flag.

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.

54.3.10 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 permits the wakeup interrupt period to be configured from 122 µs to 32 s, with a resolution down to 61 µs.
- ck_spre (usually 1 Hz internal clock) in BCD mode, or the clock used to update the calendar as defined by BCDU in binary or mixed (BCD-binary) modes.

  When ck_spre frequency is 1 Hz, a wakeup time from 1 s to around 36 hours can be achieved 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 2097*), 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 54.3.1: 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.

54.3.11 RTC initialization and configuration

RTC Binary, BCD or Mixed mode

By default the RTC is in BCD mode (BIN = 00 in the RTC_ICSR register): the RTC_SSR register contains the subsecond field SS[15:0], clocked by ck_apre, allowing to generate a 1 Hz clock to update the calendar registers in BCD format (RTC_TR and RTC_DR).

When the RTC is configured in binary mode (BIN = 01 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar registers in BCD format (RTC_TR and RTC_DR) are not used.

When the RTC is configured in mixed mode (BIN = 10 or 11 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar is updated (1 second increment) each time the SSR[BCDU+7:0] reaches 0.

RTC register write protection

After system reset, the RTC registers are protected against parasitic write access by the DBP bit in the *PWR disable Backup domain register (PWR_DBPR)*. DBP bit must be set in order to enable RTC registers write access.

After Backup domain reset, some of the RTC registers are write-protected: RTC_TR, RTC_DR, RTC_PRER, RTC_CALR, RTC_SHIFTR, the bits INIT, BIN and BCDU in RTC_ICSR and the bits FMT, SUB1H, ADD1H, REFCKON in RTC_CR.

The following steps are required to unlock the write protection on the protected RTC registers.

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

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_PRIVCFGR, and/or SEC or INITSEC is set in the RTC_SECCFGR: 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, SEC, INITSEC configuration.

In case PRIV or CALPRIV is set in the RTC_PRIVCFGR, and/or SEC or CALSEC is set in the RTC_SECCFGR: 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, SEC, CALSEC 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, plus BIN and BCDU in the RTC_ICSR register.
4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded.
   - If LPCAL=0: the counting restarts after 4 RTCCLK clock cycles.
   - If LPCAL=1: the counting restarts after up to 2 RTCCLK + 1 ck_apre.

When the initialization sequence is complete, the calendar starts counting. The RTC_SSR content is initialized with:

- PREDIV_S in BCD mode (BIN=00)
- 0xFFFF FFFF in binary or mixed (BCD-binary) modes (BIN=01, 10 or 11).

In BCD mode, RTC_SSR contains the value of the synchronous prescaler counter. This enables 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).

Note: To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ICSR register.

Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

Programming the alarm

A similar procedure must be followed to program or update the programmable alarms. The procedure below is given for alarm A but can be translated in the same way for alarm B.

1. Clear ALRAE in RTC_CR to disable alarm A.
2. Program the alarm A registers (RTC_ALRMASSR/RTC_ALRMAR or RTC_ALRMABINR).
3. Set ALRAE in the RTC_CR register to enable alarm A again.

Note: Each change of the RTC_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.

Programming the wakeup timer

The following sequence is required to configure or change the wakeup timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wakeup timer.
2. Poll WUTWF until it is set in RTC_ICSR to make sure the access to wakeup auto-reload counter and to WUCKSEL[2:0] bits is allowed. This step must be skipped in calendar initialization mode.
   - If WUCKSEL[2] = 0: WUTWF is set around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is cleared.
   - If WUCKSEL[2] = 1: WUTWF is set up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is cleared.
3. Program the wakeup auto-reload value WUT[15:0], WUTOCLR[15:0] and the wakeup clock selection (WUCKSEL[2:0] bits in RTC_CR). Set WUTE in RTC_CR to enable the timer again. The wakeup timer restarts down-counting.
   - If WUCKSEL[2] = 0: WUTWF is cleared around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is set.
   - If WUCKSEL[2] = 1: WUTWF is cleared up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is set.

54.3.12 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 APB 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 APB 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 APB 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 2096*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to *Section 54.3.14: 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.

54.3.13 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), the alarm A and alarm B registers (RTC_ALRMASSR/RTC_ALRMAR/RTC_ALRABINR and RTC_ALRMBSSR/RTC_ALRMBR/RTC_ALRBBINR).

In addition, when clocked by LSE, the RTC keeps on running under system reset if the reset source is different from the Backup domain reset one (refer to RCC for details about RTC clock sources not affected by system reset). When a Backup domain reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

54.3.14 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the subsecond field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC_SHIFTR.

The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock with a resolution of 1 ck_apre period.

The shift operation consists in adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this delays the clock.

If at the same time the ADD1S bit is set in BCD or mixed mode, this results in adding one second and at the same time subtracting a fraction of second, so this advances the clock. ADD1S has no effect in binary mode.

As soon as a shift operation is initiated by a write to the RTC_SHIFTR register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.

Caution: In mixed mode (BIN=10 or 11), the SUBFS[14:BCDU+8] must be written with 0.

Caution: Before initiating a shift operation in BCD mode, the user must check that SS[15] = 0 in order to ensure that no overflow occurs. In mixed mode, the user must check that the bit SS[BCDU+8] = 0.

Caution: This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC_SHIFTR when REFCKON = 1.

54.3.15 RTC reference clock detection

This feature is available only in BCD mode (BIN=00).

The update of the RTC calendar can be synchronized to a reference clock, RTC_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC_REFIN reference clock should be higher than the 32.768 kHz LSE clock. When the RTC_REFIN detection is enabled (REFCKON bit of RTC_CR set to 1), the calendar is still clocked by the LSE, and RTC_REFIN is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest RTC_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck_apre) generated from the 32.768 kHz quartz. The detection is performed during a time window around each of the calendar updates (every 1 s). The window equals 7 ck_apre periods when detecting the first reference clock edge. A smaller window of 3 ck_apre periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the asynchronous prescaler which outputs the ck_spre clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 ck_apre window), the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a large 7 ck_apre period detection window centered on the ck_spre edge.

When the RTC_REFIN detection is enabled, PREDIV_A and PREDIV_S must be set to their default values:

- PREDIV_A = 0x007F
- PREVID_S = 0x00FF

Note: RTC_REFIN clock detection is not available in Standby mode.

54.3.16 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.

RTC ultra-low-power mode

The RTC 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 permits the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP to 1 effectively inserts an extra ck_cal pulse every 2^{11} ck_cal cycles, which means that 512 clocks are added during every calibration cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 ck_cal cycles can be added during the calibration cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (F_{CAL}) given the input frequency (F_{RTCCLK}) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)]$$

Caution: PREDIV_A must be greater or equal to 3.

Calibration when PREDIV_A < 3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV_A bits in RTC_PRER register) is less than 3. If CALP was already set to 1 and PREDIV_A bits are

set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

It is however possible to perform a calibration with PREDIV_A less than 3 in BCD mode, the synchronous prescaler value (PREDIV_S) should be reduced so that each second is accelerated by 8 ck_cal clock cycles, which is equivalent to adding 256 clock cycles every calibration cycle. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each calibration cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV_A equals 1 (division factor of 2), PREDIV_S should be set to 16379 rather than 16383 (4 less). The only other interesting case is when PREDIV_A equals 0, PREDIV_S should be set to 32759 rather than 32767 (8 less).

If PREDIV_S is reduced in this way, the formula given the effective frequency of the calibrated input clock is as follows:

$F_{CAL} = F_{RTCCLK} \times [1 + (256 - CALM) / (2^{20} + CALM - 256)]$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

Verifying the RTC calibration

It is recommended to verify the RTC calibration with LPCAL = 0, in order to have a 32-second calibration cycle.

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

  Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).
- CALW16 bit of the RTC_CALR register can be set to 1 to force a 16- second calibration cycle period.

  In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.
- CALW8 bit of the RTC_CALR register can be set to 1 to force a 8-second calibration cycle period.

  In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8 s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

Re-calibration on-the-fly

The calibration register (RTC_CALR) can be updated on-the-fly while RTC_ICSR/INITF = 0, by using the follow process:

1. Poll the RTC_ICSR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC_CALR, if necessary. RECALPF is then automatically set to 1
3. Within three ck_apre cycles after the write operation to RTC_CALR, the new calibration settings take effect.

54.3.17 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 an internal or external tamper event is detected.

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) and refer to *Section : Timestamp on tamper event*.

54.3.18 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 $f_{RTCCLK}/(256 * (PREDIV\_A+1))$. This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = Ox7F, PREDIV_S = 0xFF), with an RTCCLK frequency at 32.768 kHz.

Note: When 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.

54.3.19 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.*

In case the TAMPALRM is configured open-drain in the RTC, the RTC_OUT1 GPIO must be configured as input.

54.4 RTC low-power modes

Table 521. 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 522. RTC pins functionality over modes

Functions	Functional in all low-power modes except Stop 3, Standby and Shutdown modes	Functional in Stop 3, Standby and Shutdown modes	Functional in V_{BAT} mode
RTC_TS	Yes	Yes	Yes
RTC_REFIN	Yes	No	No
RTC_OUT1	Yes	Yes	Yes
RTC_OUT2	Yes	Yes	Yes

54.5 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 non-secure interrupt output or the secure interrupt output is also activated.

Table 523. Non-secure interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from low-power modes
RTC	Alarm A	ALRAF	ALRAIE and (ALRASEC=0 and SEC=0)	write 1 in CALRAF	Yes(3)
	Alarm B	ALRBF	ALRBIE and (ALRBSEC=0 and SEC=0)	write 1 in CALRBF	Yes(3)
	Timestamp	TSF	TSIE and (TSSEC=0 and SEC=0)	write 1 in CTSF	Yes(3)
	Wakeup timer	WUTF	WUTIE and (WUTSEC=0 and SEC=0)	write 1 in CWUTF	Yes(3)
	SSR underflow	SSRUF	SSRUIE and (ALRASEC=0 and SEC=0)	write 1 in CSSRUF	Yes(3)

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. When the RTC is clocked by an oscillator functional in the low-power mode.

Table 524. Secure interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from low-power modes
RTC_S	Alarm A	ALRAF	ALRAIE and (ALRASEC=1 or SEC=1)	write 1 in CALRAF	Yes(3)
	Alarm B	ALRBF	ALRBIE and (ALRBSEC=1 or SEC=1)	write 1 in CALRBF	Yes(3)
	Timestamp	TSF	TSIE and (TSSEC=1 or SEC=1)	write 1 in CTSF	Yes(3)
	Wakeup timer	WUTF	WUTIE and (WUTSEC=1 or SEC=1)	write 1 in CWUTF	Yes(3)
	SSR underflow	SSRUF	SSRUIE and (ALRASEC=1 or SEC=1)	write 1 in CSSRUF	Yes(3)

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. When the RTC is clocked by an oscillator functional in the low-power mode.

54.6 RTC registers

Refer to *Section 1.2* of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

54.6.1 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 2096* and *Reading the calendar on page 2098*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 2096*.

This register can be write-protected against non-secure access. Refer to *Section 54.3.4: RTC secure protection modes*.

This register can be write-protected against non-privileged access. Refer to *Section 54.3.5: RTC privilege protection modes*.

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

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

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

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

54.6.2 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 2096* and *Reading the calendar on page 2098*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 2096*.

This register can be write-protected against non-secure access. Refer to *Section 54.3.4: RTC secure protection modes*.

This register can be write-protected against non-privileged access. Refer to *Section 54.3.5: RTC privilege protection modes*.

Address offset: 0x04

Backup domain reset value: 0x0000 2101

System reset value: 0x0000 2101 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

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

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

Bits 23:20 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units
000: forbidden
001: Monday
...
111: Sunday

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

Note: The calendar is frozen when reaching the maximum value, and can't roll over.

54.6.3 RTC subsecond register (RTC_SSR)

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

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

Bits 31:0 **SS[31:0]**: Synchronous binary counter

SS[31:16]: Synchronous binary counter MSB values

When Binary or Mixed mode is selected (BIN = 01 or 10 or 11):

SS[31:16] are the 16 MSB of the SS[31:0] free-running down-counter.

When BCD mode is selected (BIN=00):

SS[31:16] are forced by hardware to 0x0000.

SS[15:0]: Subsecond value/synchronous binary counter LSB values

When Binary mode is selected (BIN = 01 or 10 or 11):

SS[15:0] are the 16 LSB of the SS[31:0] free-running down-counter.

When BCD mode is selected (BIN=00):

SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below:

Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)

SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.

54.6.4 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 2096*.

This register can be globally protected, or each bit of this register can be individually protected against non-secure access. Refer to *Section 54.3.4: 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 54.3.5: 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.	BCDU[2:0]			BIN[1:0]		INIT	INITF	RSF	INITS	SHPF	WUTWF	Res.	Res.
			rw	rw	rw	rw	rw	rw	r	rc_w0	r	r	r		

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

Bit 16 **RECALPF**: Recalibration pending Flag

The RECALPF status flag is automatically set to 1 when software writes to the RTC_CALR register, indicating that the RTC_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to 0. Refer to *Re-calibration on-the-fly*.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:10 **BCDU[2:0]**: BCD update (BIN = 10 or 11)

In mixed mode when both BCD calendar and binary extended counter are used (BIN = 10 or 11), the calendar second is incremented using the SSR Least Significant Bits.

0x0: 1s calendar increment is generated each time SS[7:0] = 0
0x1: 1s calendar increment is generated each time SS[8:0] = 0
0x2: 1s calendar increment is generated each time SS[9:0] = 0
0x3: 1s calendar increment is generated each time SS[10:0] = 0
0x4: 1s calendar increment is generated each time SS[11:0] = 0
0x5: 1s calendar increment is generated each time SS[12:0] = 0
0x6: 1s calendar increment is generated each time SS[13:0] = 0
0x7: 1s calendar increment is generated each time SS[14:0] = 0

Bits 9:8 **BIN[1:0]**: Binary mode

00: Free running BCD calendar mode (Binary mode disabled).
01: Free running Binary mode (BCD mode disabled)
10: Free running BCD calendar and Binary modes
11: Free running BCD calendar and Binary modes

Bit 7 **INIT**: Initialization mode

0: Free running mode

1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER), plus BIN and BCDU fields. Counters are stopped and start counting from the new value when INIT is reset.

Bit 6 **INITF**: Initialization flag

When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.

0: Calendar registers update is not allowed

1: Calendar registers update is allowed

Bit 5 **RSF**: Registers synchronization flag

This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSR, RTC_TR and RTC_DR). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF = 1), or when in bypass shadow register mode (BYPSHAD = 1). This bit can also be cleared by software.

It is cleared either by software or by hardware in initialization mode.

0: Calendar shadow registers not yet synchronized

1: Calendar shadow registers synchronized

Bit 4 **INITS**: Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (Backup domain reset state).

0: Calendar has not been initialized

1: Calendar has been initialized

Bit 3 **SHPF**: Shift operation pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

0: No shift operation is pending

1: A shift operation is pending

Bit 2 **WUTWF**: Wakeup timer write flag

This bit is set by hardware when WUT value can be changed, after the WUTE bit has been set to 0 in RTC_CR.

It is cleared by hardware in initialization mode.

0: Wakeup timer configuration update not allowed except in initialization mode

1: Wakeup timer configuration update allowed

Bits 1:0 Reserved, must be kept at reset value.

54.6.5 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 2096*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 2096*.

This register can be write-protected against non-secure access. Refer to *Section 54.3.4: RTC secure protection modes*.

This register can be write-protected against non-privileged access. Refer to *Section 54.3.5: 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)

54.6.6 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 54.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 54.3.5: 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.

54.6.7 RTC control register (RTC_CR)

This register is write protected. The write access procedure is described in RTC register write protection on page 2096.

This register can be globally protected, or each bit of this register can be individually protected against non-secure access. Refer to *Section 54.3.4: 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 54.3.5: 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	ALRBFCLR	ALRAFCLR	TAMPOE	TAMPTS	ITSE	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	SSRUIE	FMT	BYPSHAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **OUT2EN**: RTC_OUT2 output enable

With this bit set, the RTC outputs can 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

Bit 28 **ALRBFCLR**: Alarm B flag automatic clear

0: Alarm B event generates a trigger event and ALRBF must be cleared by software to allow next alarm event.
1: Alarm B event generates a trigger event. ALRBF is automatically cleared by hardware after 1 ck_apre cycle.

Bit 27 **ALRAFCLR**: Alarm A flag automatic clear

0: Alarm A event generates a trigger event and ALRAF must be cleared by software to allow next alarm event.
1: Alarm A event generates a trigger event. ALRAF is automatically cleared by hardware after 1 ck_apre cycle.

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 up to 3 ck_apre cycles after the tamper flags.

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 54.3.18: 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**: Subtract 1 hour (winter time change)
When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.
Setting this bit has no effect when current hour is 0.
0: No effect
1: Subtracts 1 hour to the current time. This can be used for winter time change.

Bit 16 **ADD1H**: Add 1 hour (summer time change)
When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.
0: No effect
1: Adds 1 hour to the current time. This can be used for summer time change

Bit 15 **TSIE**: Timestamp interrupt enable
0: Timestamp interrupt disable
1: Timestamp interrupt enable

Bit 14 **WUTIE**: Wakeup timer interrupt enable
0: Wakeup timer interrupt disabled
1: Wakeup timer interrupt enabled

Bit 13 **ALRBIE**: Alarm B interrupt enable
0: Alarm B interrupt disable
1: Alarm B interrupt enable

Bit 12 **ALRAIE**: Alarm A interrupt enable
0: Alarm A interrupt disabled
1: Alarm A interrupt enabled

Bit 11 **TSE**: timestamp enable
0: timestamp disable
1: timestamp enable

Bit 10 **WUTE**: Wakeup timer enable
0: Wakeup timer disabled
1: Wakeup timer enabled
Note: When the wakeup timer is disabled, wait for WUTWF = 1 before enabling it again.

Bit 9 **ALRBE**: Alarm B enable
0: Alarm B disabled
1: Alarm B enabled

Bit 8 **ALRAE**: Alarm A enable
0: Alarm A disabled
1: Alarm A enabled

Bit 7 **SSRUIE**: SSR underflow interrupt enable
0: SSR underflow interrupt disabled
1: SSR underflow interrupt enabled

Bit 6 **FMT**: Hour format
0: 24 hour/day format
1: AM/PM hour format

Bit 5 **BYPSHAD**: Bypass the shadow registers
0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.
1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters.
Note: If the frequency of the APB clock is less than seven times the frequency of RTCCLK, BYPSHAD must be set to 1.

Bit 4 **REFCKON**: RTC_REFIN reference clock detection enable (50 or 60 Hz)
0: RTC_REFIN detection disabled
1: RTC_REFIN detection enabled
Note: BIN must be 0x00 and PREDIV_S must be 0x00FF.

Bit 3 **TSEDGE**: Timestamp event active edge
0: RTC_TS input rising edge generates a timestamp event
1: RTC_TS input falling edge generates a timestamp event
TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bits 2:0 **WUCKSEL[2:0]**: ck_wut wakeup clock selection
000: RTC/16 clock is selected
001: RTC/8 clock is selected
010: RTC/4 clock is selected
011: RTC/2 clock is selected
10x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU.
11x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU. Furthermore, 2^{16} is added to the WUT counter value.

Note: Bits 6 and 4 of this register can be written in initialization mode only (RTC_ICSR/INITF = 1).

WUT = wakeup unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].

Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ICSR WUTWF bit = 1.

It is recommended not to change the hour during the calendar hour increment as it could mask the incrementation of the calendar hour.

ADD1H and SUB1H changes are effective in the next second.

54.6.8 RTC privilege mode control register (RTC_PRIVCFGR)

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_SECCFGR configuration (refer to *Section 54.3.5: 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 and SSR underflow privilege protection
0: RTC Alarm A and SSR underflow configuration and interrupt clear can be written when the APB access is privileged or non-privileged.
1: RTC Alarm A and SSR underflow configuration and interrupt clear can be written only when the APB access is privileged.

Note: Refer to Section 54.3.5: RTC privilege protection modes for details on the read protection.

54.6.9 RTC secure configuration register (RTC_SECCFGR)

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_PRIVCFGR configuration (refer to *Section 54.3.5: RTC privilege protection modes*).

Address offset: 0x20

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
SEC	INIT SEC	CAL SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS SEC	WUT SEC	ALRB SEC	ALRA SEC
rw	rw	rw										rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **SEC**: RTC global protection

0: All RTC registers can be written when the APB access is secure or non-secure, except the registers protected by other secure protection bits.

1: All RTC registers can be written only when the APB access is secure.

Bit 14 **INITSEC**: Initialization protection

0: RTC Initialization mode, calendar and prescalers registers can be written when the APB access is secure or non-secure.

1: RTC Initialization mode, calendar and prescalers registers can be written only when the APB access is secure.

Bit 13 **CALSEC**: Shift register, daylight saving, calibration and reference clock protection

0: Shift register, daylight saving, calibration and reference clock can be written when the APB access is secure or non-secure.

1: Shift register, daylight saving, calibration and reference clock can be written only when the APB access is secure.

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 **TSSEC**: Timestamp protection

0: RTC timestamp configuration and interrupt clear can be written when the APB access is secure or non-secure.

1: RTC timestamp configuration and interrupt clear can be written only when the APB access is secure.

Bit 2 **WUTSEC**: Wakeup timer protection

0: RTC wakeup timer configuration and interrupt clear can be written when the APB access is secure or non-secure.

1: RTC wakeup timer configuration and interrupt clear can be written only when the APB access is secure.

Bit 1 **ALRBSEC**: Alarm B protection

0: RTC alarm B configuration and interrupt clear can be written when the APB access is secure or non-secure.

1: RTC alarm B configuration and interrupt clear can be written only when the APB access is secure.

Bit 0 **ALRASEC**: Alarm A and SSR underflow protection

0: RTC alarm A and SSR underflow configuration and interrupt clear can be written when the APB access is secure or non-secure.

1: RTC alarm A and SSR underflow configuration and interrupt clear can be written only when the APB access is secure.

Note: Refer to Section 54.3.4: RTC secure protection modes for details on the read protection.

54.6.10 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.

54.6.11 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 2096*.

This register can be write-protected against non-secure access. Refer to *Section 54.3.4: RTC secure protection modes*.

This register can be write-protected against non-privileged access. Refer to *Section 54.3.5: 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	CALW 16	LPCAL	Res.	Res.	Res.	CALM[8:0]								
rw	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **CALP**: Increase frequency of RTC by 488.5 ppm
0: No RTCCLK pulses are added.
1: One RTCCLK pulse is effectively inserted every 2^{11} pulses (frequency increased by 488.5 ppm).
This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. if the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows:
(512 × CALP) - CALM.
Refer to *Section 54.3.16: 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 54.3.16: 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 54.3.16: RTC smooth digital calibration.

Bit 12 **LPCAL**: RTC low-power mode

0: Calibration window is 2^{20} RTCCLK, which is a high-consumption mode. This mode must 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 54.3.16: RTC smooth digital calibration on page 2101*.

54.6.12 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 2096*.

This register can be protected against non-secure access. Refer to *Section 54.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 54.3.5: 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))).

In mixed BCD-binary mode (BIN=10 or 11), the SUBFS[14:BCDU+8] must be written with 0.

Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF = 1 to be sure that the shadow registers have been updated with the shifted time.

54.6.13 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 54.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 54.3.5: 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.

54.6.14 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 54.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 54.3.5: 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

54.6.15 RTC timestamp subsecond 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 54.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 54.3.5: 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
SS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 SS[31:0]: Subsecond value/synchronous binary counter values

SS[31:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

54.6.16 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 54.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 54.3.5: 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.

54.6.17 RTC alarm A subsecond 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 54.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 54.3.5: 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
SSCLR	Res.	MASKSS[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw		rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bit 31 **SSCLR**: Clear synchronous counter on alarm (Binary mode only)

0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.

1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRMABINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one ck_apre cycle after reaching RTC_ALRMABINR.SS[31:0].

Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 **MASKSS[5:0]**: Mask the most-significant bits starting at this bit

0: No comparison on subseconds for Alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

1: SS[31:1] are don't care in Alarm A comparison. Only SS[0] is compared.

2: SS[31:2] are don't care in Alarm A comparison. Only SS[1:0] are compared.

...

31: SS[31] is don't care in Alarm A comparison. Only SS[30:0] are compared.

From 32 to 63: All 32 SS bits are compared and must match to activate alarm.

Note: In BCD mode (BIN=00) the overflow bits of the synchronous counter (bits 31:15) are never compared. These bits can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Subseconds value

This value is compared with the contents of the synchronous prescaler counter to determine if alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

This field is the mirror of SS[14:0] in the RTC_ALRMABINR, and so can also be read or written through RTC_ALRMABINR.

Note: SS[3:0] must be 0000 when SSCLR is set with ATCKSEL[3] = 1 in TAMP_ATCR1.

54.6.18 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 54.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 54.3.5: 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

54.6.19 RTC alarm B subsecond 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 54.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 54.3.5: 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
SSCLR	Res.	MASKSS[5:4]		MASKSS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw		rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bit 31 **SSCLR**: Clear synchronous counter on alarm (Binary mode only)
0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.
1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRMBBINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one ck_apre cycle after reaching RTC_ALRMBBINR.SS[31:0].
Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 **MASKSS[5:0]**: Mask the most-significant bits starting at this bit
0: No comparison on subseconds for Alarm B. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).
1: SS[31:1] are don't care in Alarm B comparison. Only SS[0] is compared.
2: SS[31:2] are don't care in Alarm B comparison. Only SS[1:0] are compared.
...
31: SS[31] is don't care in Alarm B comparison. Only SS[30:0] are compared.
From 32 to 63: All 32 SS bits are compared and must match to activate alarm.
Note: In BCD mode (BIN=00)The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Subseconds value
This value is compared with the contents of the synchronous prescaler counter to determine if alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.
This field is the mirror of SS[14:0] in the RTC_ALRMBBINR, and so can also be read or written through RTC_ALRMBBINR.
Note: SS[3:0] must be 0000 when SSCLR is set with ATCKSEL[3] = 1 in TAMP_ATCR1.

54.6.20 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 54.3.4: 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 54.3.5: 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.	SSRUF	ITSF	TSOVF	TSF	WUTF	ALRBF	ALRAF
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUF**: SSR underflow flag
This flag is set by hardware when the SSR rolls under 0. SSRUF is not set when SSCLR=1.

Bit 5 **ITSF**: Internal timestamp flag
This flag is set by hardware when a timestamp on the internal event occurs.

Bit 4 **TSOVF**: Timestamp overflow flag

This flag is set by hardware when a timestamp event occurs while TSF is already set.

It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSF**: Timestamp flag

This flag is set by hardware when a timestamp event occurs.

If ITSF flag is set, TSF must be cleared together with ITSF.

Note: TSF is not set if TAMPTS = 1 and the tamper flag is read during the 3 ck_apre cycles following tamper event. Refer to Timestamp on tamper event for more details.

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.

54.6.21 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 54.3.5: 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.	SSR UMF	ITS MF	TSOV MF	TS MF	WUT MF	ALRB MF	ALRA MF
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUMF**: SSR underflow non-secure masked flag
This flag is set by hardware when the SSR underflow non-secure interrupt occurs.

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.

54.6.22 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 54.3.4: 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 54.3.5: 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.	SSRUMF	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUMF**: SSR underflow secure masked flag
This flag is set by hardware when the SSR underflow secure interrupt occurs.

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.

54.6.23 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 54.3.4: 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 54.3.5: 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.	CSSRUF	CITSF	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
									w	w	w	w	w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CSSRUF**: Clear SSR underflow flag
Writing ‘1’ in this bit clears the SSRUF in the RTC_SR register.

Bit 5 **CITSF**: Clear internal timestamp flag
Writing 1 in this bit clears the ITSF bit in the RTC_SR register.

Bit 4 **CTSOVF**: Clear timestamp overflow flag
Writing 1 in this bit clears the TSOVF bit in the RTC_SR register.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **CTSF**: Clear timestamp flag
Writing 1 in this bit clears the TSF bit in the RTC_SR register.
If ITSF flag is set, TSF must be cleared together with ITSF by setting CRSF and CITSF.

Bit 2 **CWUTF**: Clear wakeup timer flag
Writing 1 in this bit clears the WUTF bit in the RTC_SR register.

Bit 1 **CALRBF**: Clear alarm B flag
Writing 1 in this bit clears the ALRBF bit in the RTC_SR register.

Bit 0 **CALRAF**: Clear alarm A flag
Writing 1 in this bit clears the ALRAF bit in the RTC_SR register.

54.6.24 RTC alarm A binary mode register (RTC_ALRABINR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register can be protected against non-secure access. Refer to *Section 54.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 54.3.5: RTC privilege protection modes*.

Address offset: 0x70

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 **SS[31:0]**: Synchronous counter alarm value in Binary mode

This value is compared with the contents of the synchronous counter to determine if Alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMASSRR, and so can also be read or written through RTC_ALRMASSR.

Note: SS[3:0] must be 0000 when SSCLR is set with ATCKSEL[3] = 1 in TAMP_ATCR1.

54.6.25 RTC alarm B binary mode register (RTC_ALRBBINR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register can be protected against non-secure access. Refer to *Section 54.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 54.3.5: RTC privilege protection modes*.

Address offset: 0x74

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 **SS[31:0]**: Synchronous counter alarm value in Binary mode

This value is compared with the contents of the synchronous counter to determine if Alarm Bis to be activated. Only bits 0 up MASKSS-1 are compared.

SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMBSSRR, and so can also be read or written through RTC_ALRMBSSR.

Note: SS[3:0] must be 0000 when SSCLR is set with ATCKSEL[3] = 1 in TAMP_ATCR1.

54.6.26 RTC register map

Table 525. RTC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**RTC_TR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT [1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x04	**RTC_DR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]				WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1			0	0	0	0	0	1
0x08	**RTC_SSR**	SS[31:16]																SS[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**RTC_ICSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF	Res.	Res.	Res.	BCDU [2:0]			BIN [1:0]		INIT	INITF	RSF	INITS	SHPF	WUT WF	Res.	Res.
	Reset value																0				0	0	0	0	0	0	0	0	0	0	1		
0x10	**RTC_PRER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]							PREDIV_S[14:0]															
	Reset value										1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x14	**RTC_WUTR**	WUTOCLR[15:0]																WUT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x18	**RTC_CR**	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	ALRBFCLR	ALRAFCLR	TAMPOE	TAMPTS	ITSE	COE	O SEL [1:0]		POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	SSRUIE	FMT	BYPSHAD	REFCKON	TSEDGE	WUCK SEL[2:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**RTC_ PRIVCFGR**	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_ SECCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC	INITSEC	CALSEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSSEC	WUTSEC	ALRBSEC	ALRASEC
	Reset value																	0	0	0										0	0	0	0
0x24	**RTC_WPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	**RTC_ CALR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALP	CALW8	CALW16	LPCAL	Res.	Res.	Res.	CALM[8:0]								
	Reset value																	0	0	0	0				0	0	0	0	0	0	0	0	0
0x2C	**RTC_SHIFTR**	ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBFS[14:0]														
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 525. 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**	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	**RTC_ALRMAR**	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**RTC_ ALRMASSR**	SSCLR	Res.	MASKSS [5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value	0		0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**RTC_ALRMBR**	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	**RTC_ ALRMBSSR**	SSCLR	Res.	MASKSS [5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value	0		0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	**RTC_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUF	ITSF	TSOVF	TSF	WUTF	ALRBF	ALRAF
	Reset value																										0	0	0	0	0	0	0
0x54	**RTC_MISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
	Reset value																										0	0	0	0	0	0	0
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.	SSRUMF	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
	Reset value																										0	0	0	0	0	0	0
0x5C	**RTC_SCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSRUF	CITSF	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
	Reset value																										0	0	0	0	0	0	0
0x70	**RTC_ ALRABINR**	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74	**RTC_ ALRBBINR**	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

55 Tamper and backup registers (TAMP)

55.1 Introduction

The anti-tamper detection circuit is used to protect sensitive data from external attacks. 32 32-bit backup registers are retained in all low-power modes and also in V_{BAT} mode. The backup registers, as well as other secrets in the device, are protected by this anti-tamper detection circuit with 8 tamper pins and 11 internal tampers. 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.

55.2 TAMP main features

- A tamper detection can erase the backup registers, backup SRAM, SRAM2, caches and cryptographic peripherals.
- 32 32-bit backup registers:
  - The backup registers (TAMP_BKPxR) are implemented in the backup domain that remains powered-on by V_{BAT} when the V_{DD} power is switched off.
- Up to 8 tamper pins for 8 external tamper detection events:
  - Active tamper mode: continuous comparison between tamper output and input to protect from physical open-short attacks.
  - Flexible active tamper I/O management: from 4 meshes (each input associated to its own exclusive output) to 7 meshes (single output shared for up to 7 tamper inputs)
  - Passive tampers: ultra-low power edge or level detection with internal pull-up hardware management.
  - Configurable digital filter.
- 11 internal tamper events to protect against transient or environmental perturbation attacks
- Each tamper can be configured in two modes:
  - Hardware mode: immediate erase of secrets on tamper detection, including backup registers erase
  - Software mode: the secrets erase following a tamper detection is launched by software
- Any tamper detection can generate a RTC timestamp event.
- 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.
  - Boot hardware key for secure AES, stored in backup registers, protected against read and write access.
- Tamper configuration and backup registers privilege protection
- Monotonic counter.

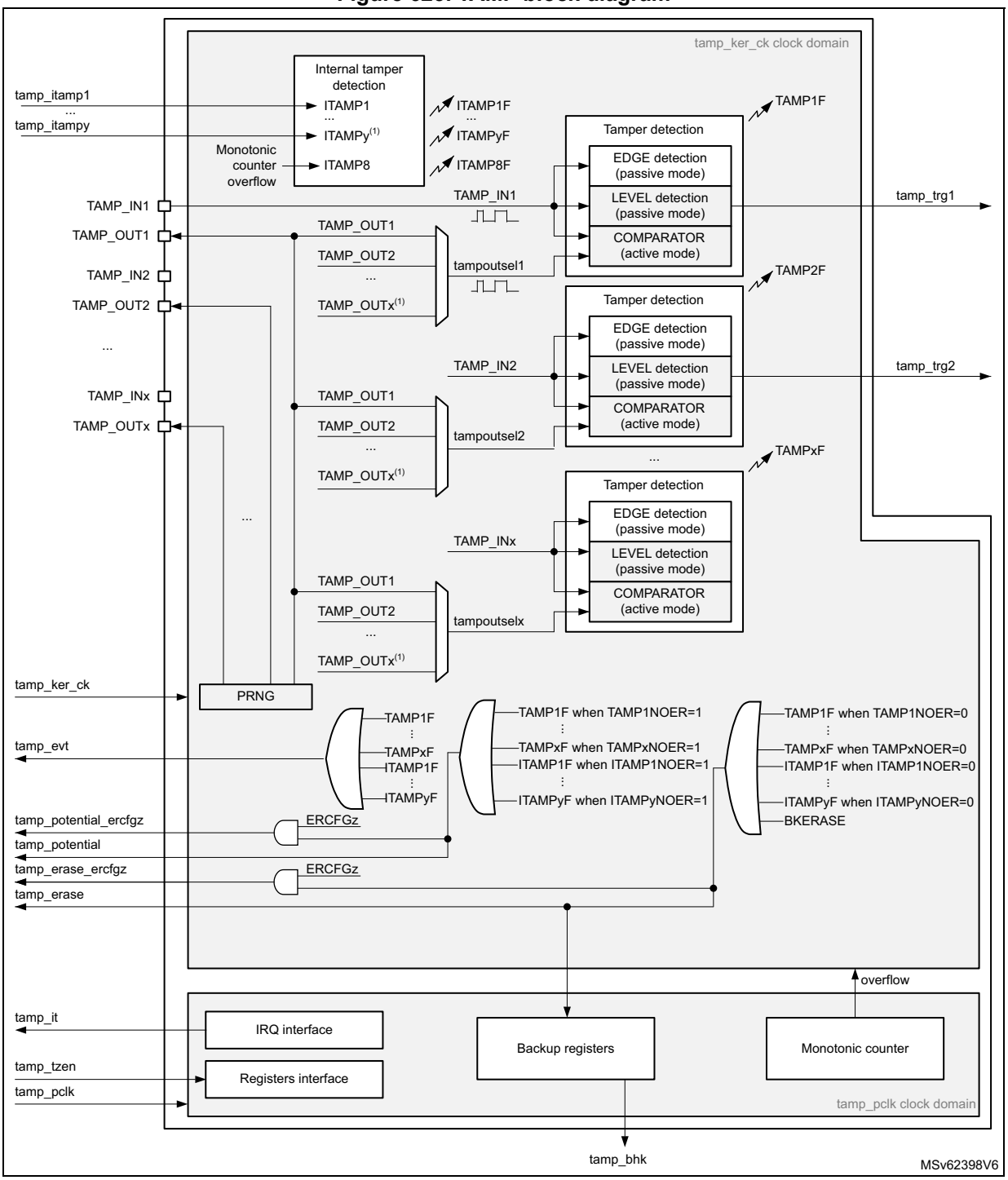
55.3 TAMP implementation

STM32U585/STM32U575 rev. X devices do not implement the active tamper prescaler extension. This feature is implemented in all other STM32U585/STM32U575 revisions.

55.4 TAMP functional description

55.4.1 TAMP block diagram

Figure 623. TAMP block diagram



1. The number of external and internal tampers depends on products.

55.4.2 TAMP pins and internal signals

Table 526. 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 (active mode only)

Table 527. 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 flag (internal or external tamper), whatever NOERASE configuration.
tamp_potential	Output	Potential tamper detection signal, used for device secrets(1) protection. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in NOERASE configuration: potential tamper detection. – or a software request is done by writing BKBLOCK to 1
tamp_erase	Output	Confirmed tamper detection signal, used for device secrets(1) protection. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in ERASE configuration. – or a software request is done by writing BKERASE to 1
tamp_potential_ercfgz (z = signal index)	Output	Potential tamper detection signal generated only when ERCFGz = 1. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in NOERASE configuration: potential tamper detection. – or a software request is done by writing BKBLOCK to 1

Table 527. TAMP internal input/output signals (continued)

Internal signal name	Signal type	Description
tamp_erase_ercfgz (z = signal index)	Output	Confirmed tamper detection signal generated only when ERCFGz = 1. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in ERASE configuration. – or a software request is done by writing BKERASE to 1
tamp_it	Output	TAMP interrupt (refer to *Section 55.6: TAMP interrupts* for details)
tamp_trg[x] (x = signal index)	Output	Tamper detection trigger
tamp_bhk	Output	Tamper boot hardware key bus

1. Refer to *Table 528: TAMP interconnection*.

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} depending on the selected clock (refer to *Section 55.5: TAMP low-power modes* for more details.

Table 528. 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_potential	The tamp_potential signal is used to block the read and write accesses to the device secrets listed hereafter: – backup registers – SRAM2 RHUK (root hardware unique key) in system Flash memory and BHK (boot hardware key) hardware buses to SAES are blocked. The tamp_potential signal is used to erase the device secrets listed hereafter: – ICACHE content – SAES, AES, HASH peripherals – PKA SRAM The device secrets access is blocked when erase is ongoing.

Table 528. TAMP interconnection (continued)

Signal name	Source/Destination
tamp_erase	The tamp_erase signal is used to erase the device secrets listed hereafter: – backup registers – SRAM2 – ICACHE/DCACHE content – OTFDEC keys and CRC registers – SAES, AES, HASH peripherals – PKA SRAM The device secrets access is blocked when erase is ongoing. RHUK in system Flash memory (root hardware unique key) hardware bus to SAES is blocked.
tamp_potential_ercfg0	When the bit ERCFG0 is set in the TAMP_ERCFGR, the tamp_potential_ercfg0 signal is used to block the read and write accesses to the device secrets listed hereafter: – Backup SRAM
tamp_erase_ercfg0	When the bit ERCFG0 is set in the TAMP_ERCFGR, the tamp_erase_ercfg0 signal is used to erase the device secrets listed hereafter: – Backup SRAM
tamp_itamp1	Backup domain voltage threshold monitoring(1)
tamp_itamp2	Temperature monitoring(1)
tamp_itamp3	LSE monitoring (CSS)(2)
tamp_itamp5	RTC calendar overflow (rtc_calovf)
tamp_itamp6	JTAG/SWD access when RDP > 0
tamp_itamp7	ADC4 (adc4_awd1) analog watchdog monitoring 1
tamp_itamp8(3)	Monotonic counter 1 overflow
tamp_itamp9	Cryptographic peripherals fault (SAES or AES or PKA or TRNG)
tamp_itamp11	IWDG reset when tamper flag is set (potential tamper timeout)
tamp_itamp12	ADC4 (adc4_awd2) analog watchdog monitoring 2
tamp_itamp13	ADC4 (adc4_awd3) analog watchdog monitoring 3
tamp_bhk	saes_bhk. This bus is used to load the boot hardware key in the secure AES co-processor.

1. This monitoring must be enabled by setting MONEN in *PWR disable Backup domain register (PWR_DBPR)*.
2. This monitoring must be enabled by setting LSECSSON in *RCC Backup domain control register (RCC_BDCR)*.
3. 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.

55.4.3 GPIOs controlled by the RTC and TAMP

Refer to *Section 54.3.3: GPIOs controlled by the RTC and TAMP*.

55.4.4 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.

55.4.5 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 configuration register (TAMP_SECCFGR) which can be written in secure mode only. The TAMP protection configuration is not affected by a system reset.

- When the TAMPSEC bit is set in the TAMP_SECCFGR register:
  - Writing the TAMP registers is possible only in secure mode, except for the backup registers which have their own protection setting.
  - Reading TAMP_SECCFGR, TAMP_PRIVCFGR 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 and monotonic counters which have their own protection setting.
- When the CNT1SEC bit is set in the TAMP_SECCFGR register: the TAMP_COUNT1R can be read and written only in secure mode.

A non-secure access to a secure-protected register is denied:

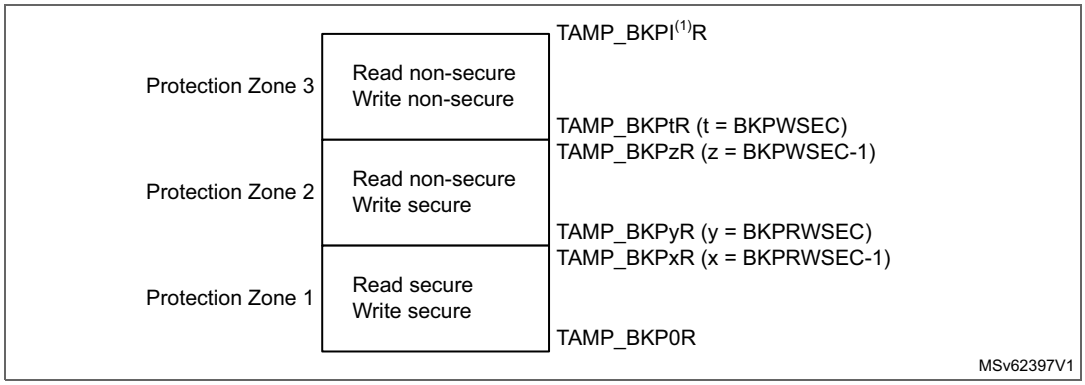
- There is no bus error generated.
- A notification is generated through a flag/interrupt in the TZIC (TrustZone illegal access controller).
- When write protected, the bits are not written.
- When read protected they are read as 0.

As soon as at least one function is configured to be secured, the TAMP reset and clock control is also secured in the RCC.

55.4.6 Backup registers protection zones

The backup registers protection is configured thanks to BKPRWSEC[7:0] and BKPWSEC[7:0] (refer to the figure below):

Figure 624. Backup registers protection zones



1. I = last backup register index

In case TZEN =1, the bits BKPWPRIV and BKPRWPRIV in the TAMP_PRIVCFGR can be written only in secure mode.

55.4.7 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 configuration register (TAMP_PRIVCFGR) 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_PRIVCFGR register:

- Writing the TAMP registers is possible only in privilege mode, except for the backup registers and the monotonic counters which have their own protection setting.
- When the CNT1PRIV bit is set in the TAMP_PRIVCFGR register: the TAMP_COUNT1R can be read and written only in privilege mode.
- Reading TAMP_SECCFGR, TAMP_PRIVCFGR 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 and the monotonic counters which have their own protection setting.

The backup registers protection is configured thanks to BKPRWSEC[7:0] and BKPRWPRIV for the protection zone 1, and thanks to BKPRWSEC[7:0], BKPWSEC[7:0] and BKPWPRIV for the protection zone 2 (refer to *Figure 624*). The BHKLOCK bit can be written only in privileged mode when the BKPRWPRIV bit is set.

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.

55.4.8 Boot hardware key

The eight first backup registers from TAMP_BKP0R to TAMP_BKP7R can be used to store a boot hardware key for the secure AES.

For this purpose, these registers must belong to the Protection Zone 1: BKPRWSEC must be greater or equal to 8.

Once the backup registers are written with the boot hardware key, the BHKLOCK bit must be set in the TAMP_SECCFGR register. Once BHKLOCK is set, the 8 backup registers cannot be accessed anymore by software: they are read as 0 and write to these registers is ignored. BHKLOCK cannot be cleared by software, and is cleared by hardware following a tamper event or when the readout protection (RDP) is disabled (in both cases the backup registers are also erased).

Refer to section secure AES co-processor (SAES) for details on procedure to download the boot hardware key in the SAES.

55.4.9 Tamper detection

The tamper detection main purpose is to protect the device secrets from device external attacks. The detection is made on events on TAMP_INx (x=1...8) I/Os, or on internal monitors detecting out-of-range device conditions.

The tamper detection can be configured for the following purposes:

- erase the backup registers and all secrets stored in SRAMs or peripherals listed in *Table 528: TAMP interconnection* (default configuration)
- generate an interrupt, capable to wakeup from low-power modes
- generate a hardware trigger for the low-power timers, or a RTC timestamp event

The external I/Os tamper detection supports 2 main configurations:

- Passive mode: TAMP_INx I/Os are monitored and a tamper is detected either on edge or on level.
- Active mode: TAMP_INx (x=1...8) is continuously compared with TAMP_OUTy (y=1...8) allowing open-short detection.

A digital filter can be applied on external tamper detection to avoid false detection. In addition, it is possible to configure each tamper source with NOERASE configuration, so that the secrets erase is not launched by hardware on tamper detection. The secrets erase can then be launched by software after software checks.

55.4.10 TAMP backup registers and other device secrets erase

The backup registers (TAMP_BKPxR) are not reset by system reset or when the device wakes up from Standby mode.

The backup registers and the other device secrets are not reset when the corresponding mask is set (TAMPxMSK=1 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.

ERASE configuration for confirmed tamper detection

The backup registers and the other device secrets erased by tamp_erase signal (refer to *Table 528: TAMP interconnection*) are reset by hardware on an external or an internal

tamper event when the ERASE configuration is selected for the tamper source (TAMPxNOER=0 in the TAMP_CR2 register, ITAMPxNOER=0 in the TAMP_CR3 register).

NOERASE configuration for potential tamper detection

In the NOERASE configuration (TAMPxNOER=1 in the TAMP_CR2 register, ITAMPxNOER=1 in the TAMP_CR3 register), the backup registers and other device secrets are not erased when the corresponding tamper event is detected. In addition, the read and write accesses to the backup registers and to other device secrets are blocked as soon as this potential tamper detection flag is set (tamp_potential signal, set if at least one TAMPxF or ITAMPxF is set), until this flag is cleared. Therefore the software can perform some checks to discriminate false from true tampers, and decide to launch secrets erase only in case of the potential tamper is confirmed to be a true tamper. The backup registers and other device secrets are erased by software by setting the BKERASE bit in the TAMP_CR2 register.

Note: Some other device secrets might be erased immediately also in this configuration. Refer to Table 528: TAMP interconnection for more details about tamp_potential effects in the device.

Potential tamper to confirmed tamper timeout

Some internal tampers generate a tamper event if the independent watchdog reset occurs when another tamper flag is set (refer to *Table 528: TAMP interconnection*). The IWDG tamper must be configured with ITAMPxNOER = 0. This permits the reset of the backup registers and other device secrets to be forced by hardware after a timeout, in case the previous tamper event was in NOERASE configuration. This is equivalent to change the “potential tamper” into “confirmed tamper” if a watchdog reset occurs before any software decision following the potential tamper event.

Device assets protection configuration

Some device assets can be configured in order to be added to the list of the device secrets protected by tamper detection.

By default (ERCFGy=0 in the TAMP_ERCFGR), the device assets associated to ERCFGy are not protected by the TAMP peripheral:

- They are not erased in case of confirmed tamper or by software BKERASE command.
- Their R/W accesses are not blocked in case of potential tamper nor by software BKBLOCK command.

When the ERCFGy bit is set to 1 in the TAMP_ERCFGR, the associated device assets are included in the device secrets protected by TAMP peripheral:

- They are erased in case of confirmed tamper or by software BKERASE command. Refer to tamp_erase_ercfg in *Table 526: TAMP input/output pins*.
- Their R/W accesses are blocked in case of potential tamper or by software BKBLOCK command. Refer to tamp_potential_ercfg in *Table 526: TAMP input/output pins*.

Device secrets access blocked by software

By default, the device secrets can be accessed by the application, except if a tamper event flag is detected: the device secrets access is not possible as long as a tamper flag is set.

It is possible to block the access to the device secrets by software, by setting the BKBLOCK bit of the TAMP_CR2 register. The device secrets access is possible only when BKBLOCK = 0 and no tamper flag is set.

55.4.11 Tamper detection configuration and 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.

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 permits the low-power timers to be triggered automatically in Stop mode, without requiring the system wakeup to perform the TAMPxF clearing. In this case, the backup registers are not cleared.

This feature is available only when the tamper is configured in the *Level detection with filtering on tamper inputs (passive mode)* mode (TAMPFLT ≠ 00 and active mode is not selected).

Timestamp on tamper event

With TAMPTS set to 1 in the RTC_CR, any internal or external 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.

Note: TSF is set up to 3 ck_apre cycles after TAMPxF flags. TSF is not set if RTCCLK is stopped (it is set when RTCCLK restarts).

Note: If TAMPxF is cleared before the expected rise of TSF, TSF is not set. Consequently, in case TAMPTS = 1, the software should either wait for timestamp flag before clearing the tamper flag, or should read the RTC counters values in the TAMP interrupt routine.

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 or a falling edge 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 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} × CK_ATPRE), 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.

Table 529. Active tamper output change period

ATCKSEL[3:0]	CK_ATPRE frequency	ATPER[2:0]	Tamper output change (CK_ATPER) frequency	Tamper output change period(1) (ms)
0x0	f_{RTCCLK}	0x0	f_{RTCCLK}	0.030
		0x1	$f_{RTCCLK}/2$	0.061
		0x2	$f_{RTCCLK}/4$	0.122
		0x3	$f_{RTCCLK}/8$	0.244
		0x4	$f_{RTCCLK}/16$	0.488
		0x5	$f_{RTCCLK}/32$	0.977
		0x6	$f_{RTCCLK}/64$	1.953
		0x7	$f_{RTCCLK}/128$	3.906

Table 529. Active tamper output change period (continued)

ATCKSEL[3:0]	CK_ATPRE frequency	ATPER[2:0]	Tamper output change (CK_ATPER) frequency	Tamper output change period(1) (ms)
…	…	…	…	…
0x7	f_{RTCCLK}/128	0x0	f_{RTCCLK}/128	3.906
		0x1	f_{RTCCLK}/256	7.8125
		0x2	f_{RTCCLK}/512	15.625
		0x3	f_{RTCCLK}/1024	31.250
		0x4	f_{RTCCLK}/2048	62.5
		0x5	f_{RTCCLK}/4096	125
		0x6	f_{RTCCLK}/8192	250
		0x7	f_{RTCCLK}/16384	500
0xB(2)	f_{RTCCLK}/2048(3)	0x0	f_{RTCCLK}/2048	62.5
		0x1	f_{RTCCLK}/4096	125
		0x2	f_{RTCCLK}/8192	250
		0x3	f_{RTCCLK}/16384	500
		0x4	f_{RTCCLK}/32768	1000
		0x5	f_{RTCCLK}/65536	2000
		0x6	f_{RTCCLK}/131072	4000
		0x7	f_{RTCCLK}/262144	8000

1. Assuming f_{RTCCLK} = 32768 Hz.
2. These values are supported only when the active tamper prescaler extension is supported. Refer to *Section 55.3: TAMP implementation*.
3. This setting requires that (PREDIV_A+1) = 128 and (PREDIV_S+1) is a multiple of 16.

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 530. Minimum ATPER value

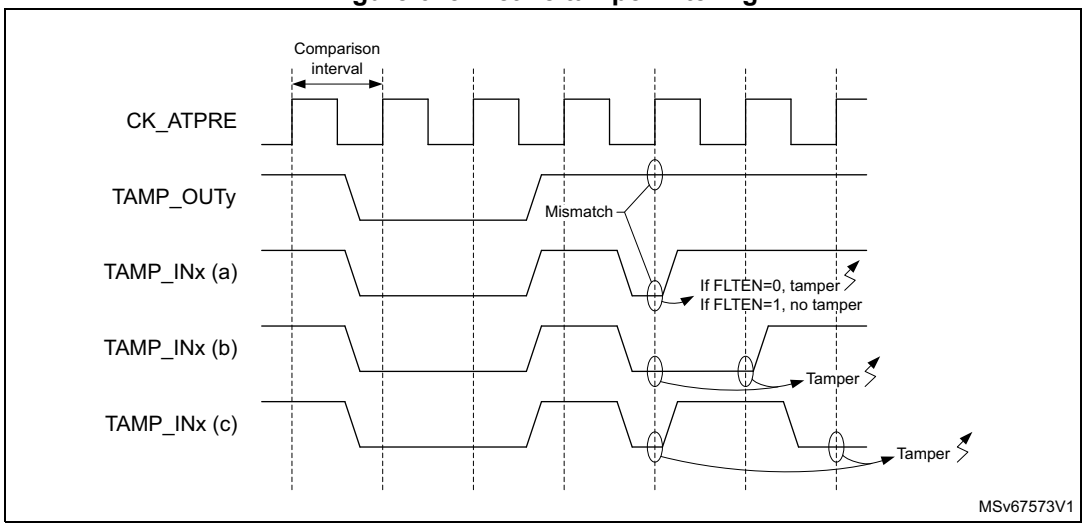
Number of selected outputs	Minimum ATPER
1	0
2	1
3 or 4	2
5 or more	3

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, every CK_ATPRE cycle. 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.

Figure 625. Active tamper filtering



As illustrated in *Figure 625*, if FLTEN = 0, any mismatch between the TAMP_OUTy output and the associated TAMP_INx input when the latter is sampled generates a tamper. This is the case in all three examples (a), (b) and (c).

If FLTEN = 1, example (a) does not generate a tamper, since only one mismatch is detected in four consecutive comparisons. In example (b), a tamper is generated since two successive mismatches are detected. Example (c) also generates a tamper, since two mismatches occur in four consecutive comparisons, even though the mismatches do not occur on successive samples.

Setting FLTEN = 1 avoids unwanted detection of tampers due to glitches, bounce or transitory states on the TAMP_INx inputs, by ignoring single pulses which are shorter than one period of CK_ATPRE, programmed in the ATCKSEL field of the TAMP_ATCR1 register. The minimum filtered pulse width is listed in *Table 531* for each possible setting of ATCKSEL, assuming f_{RTCCLK} = 32.768 kHz.

Table 531. Active tamper filtered pulse duration

ATCKSEL[3:0]	CK_ATPRE frequency	Minimum filtered pulse width (ms)
0x0	f_{RTCCLK}	0.030
0x1	$f_{RTCCLK}/2$	0.061
0x2	$f_{RTCCLK}/4$	0.122
0x3	$f_{RTCCLK}/8$	0.244
0x4	$f_{RTCCLK}/16$	0.488
0x5	$f_{RTCCLK}/32$	0.977
0x6	$f_{RTCCLK}/64$	1.953
0x7	$f_{RTCCLK}/128$	3.906
0xB(1)	$f_{RTCCLK}/2048$	62.500(2)

1. These values are supported only when the active tamper prescaler extension is supported. Refer to *Section 55.3: TAMP implementation*.
2. This setting requires that (PREDIV_A+1) = 128 and (PREDIV_S+1) is a multiple of 16.

Note: Multiple pulses which are shorter than one CK_ATPRE period may nevertheless cause a tamper if they result in two mismatches in four consecutive comparisons.

Caution: Entering RTC initialization mode stops CK_ATPRE and CK_ATPER clocks when ATCKSEL[3] = 1. Therefore, TAMP_OUTy pin stops toggling until INIT mode exit.

Refer to section *Section : Calendar initialization and configuration*.

Refer also to *RTC alarm A subsecond register (RTC_ALRMASSR)*, *RTC alarm B subsecond register (RTC_ALRMBSSR)*, *RTC alarm A binary mode register (RTC_ALRABINR)* and *RTC alarm B binary mode register (RTC_ALRBBINR)* in case RTC binary mode is used in conjunction with ATCKSEL[3] = 1.

Caution: Caution: The active tamper detection is no more functional in case of calendar overflow when ATCKSEL[3] = 1. It is mandatory to enable the internal tamper 5 on calendar overflow to ensure tamper protection.

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 TAMP_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 TAMP 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 TAMP_ATOR. Backup registers are then protected by active tamper.

55.5 TAMP low-power modes

Table 532. 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 533. TAMP pins functionality over modes

Pin name	Functional in all low-power modes	Functional in V_{BAT} mode
TAMP_IN[8:1]	Yes	Yes
TAMP_OUT[8:1]	Yes	Yes

55.6 TAMP 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 (TAMPSEC).

Table 534. Interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from low-power modes
TAMP	Tamper x(3)	TAMPxF	TAMPxIE	Write 1 in CTAMPxF	Yes(4)
	Internal tamper y(3)	ITAMPyF	ITAMPyIE	Write 1 in CITAMPyF	Yes(4)

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. Refer to *Table 532: Effect of low-power modes on TAMP* for more details about available features in the low-power modes.

55.7 TAMP registers

Refer to *Section 1.2* of the reference manual for a list of abbreviations used in register descriptions. The peripheral registers can be accessed by words (32-bit).

55.7.1 TAMP control register 1 (TAMP_CR1)

This register can be protected against non-secure access. Refer to *Section 55.4.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 55.4.7: TAMP privilege protection modes*.

Address offset: 0x00

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.	ITAMP13E	ITAMP12E	ITAMP11E	Res.	ITAMP9E	ITAMP8E	ITAMP7E	ITAMP6E	ITAMP5E	Res.	ITAMP3E	ITAMP2E	ITAMP1E
			rw	rw	rw		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.	TAMP8E	TAMP7E	TAMP6E	TAMP5E	TAMP4E	TAMP3E	TAMP2E	TAMP1E
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **ITAMP13E**: Internal tamper 13 enable
0: Internal tamper 13 disabled.
1: Internal tamper 13 enabled.

Bit 27 **ITAMP12E**: Internal tamper 12 enable
0: Internal tamper 12 disabled.
1: Internal tamper 12 enabled.

Bit 26 **ITAMP11E**: Internal tamper 11 enable
0: Internal tamper 11 disabled.
1: Internal tamper 11 enabled.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9E**: Internal tamper 9 enable
0: Internal tamper 9 disabled.
1: Internal tamper 9 enabled.

Bit 23 **ITAMP8E**: Internal tamper 8 enable
0: Internal tamper 8 disabled.
1: Internal tamper 8 enabled.

Bit 22 **ITAMP7E**: Internal tamper 7 enable
0: Internal tamper 7 disabled.
1: Internal tamper 7 enabled

Bit 21 **ITAMP6E**: Internal tamper 6 enable
0: Internal tamper 6 disabled.
1: Internal tamper 6 enabled.

Bit 20 **ITAMP5E**: Internal tamper 5 enable
0: Internal tamper 5 disabled.
1: Internal tamper 5 enabled.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **ITAMP3E**: Internal tamper 3 enable
0: Internal tamper 3 disabled.
1: Internal tamper 3 enabled.

Bit 17 **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(1)
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(1)
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(1)
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(1)
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(1)
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(1)
0: Tamper detection on TAMP_IN3 is disabled.
1: Tamper detection on TAMP_IN3 is enabled.

Bit 1 **TAMP2E**: Tamper detection on TAMP_IN2 enable(1)
0: Tamper detection on TAMP_IN2 is disabled.
1: Tamper detection on TAMP_IN2 is enabled.

Bit 0 **TAMP1E**: Tamper detection on TAMP_IN1 enable(1)
0: Tamper detection on TAMP_IN1 is disabled.
1: Tamper detection on TAMP_IN1 is enabled.

1. Tamper detection mode (selected with TAMP_FLTCR, TAMP_ATCR1, TAMP_ATCR2 registers and TAMPxTRG bits in TAMP_CR2), must be configured before enabling the tamper detection.

55.7.2 TAMP control register 2 (TAMP_CR2)

This register can be protected against non-secure access. Refer to *Section 55.4.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 55.4.7: 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	BK BLOCK	Res.	Res.	Res.	TAMP3 MSK	TAMP2 MSK	TAMP1 MSK
rw	rw	rw	rw	rw	rw	rw	rw	w	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.	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 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 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 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 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 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 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 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 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 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 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 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 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 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 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 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 triggers a tamper detection event.

Bit 23 **BKERASE**: Backup registers and device secrets(1) erase
Writing ‘1’ to this bit reset the backup registers and device secrets(1). Writing 0 has no effect. This bit is always read as 0.

Bit 22 **BKBLOCK**: Backup registers and device secrets(1) access blocked
0: backup registers and device secrets(1) can be accessed if no tamper flag is set
1: backup registers and device secrets(1) cannot be accessed

Bits 21:19 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 and device secrets(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 and device secrets(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 and device secrets(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 and all device secrets(1).

1: Tamper 8 event does not erase the backup registers and device secrets(2).

Bit 6 **TAMP7NOER**: Tamper 7 no erase

0: Tamper 7 event erases the backup registers and all device secrets(1).

1: Tamper 7 event does not erase the backup registers and device secrets(2).

Bit 5 **TAMP6NOER**: Tamper 6 no erase

0: Tamper 6 event erases the backup registers and all device secrets(1).

1: Tamper 6 event does not erase the backup registers and device secrets(2).

Bit 4 **TAMP5NOER**: Tamper 5 no erase

0: Tamper 5 event erases the backup registers and all device secrets(1).

1: Tamper 5 event does not erase the backup registers and device secrets(2).

Bit 3 **TAMP4NOER**: Tamper 4 no erase

0: Tamper 4 event erases the backup registers and all device secrets(1).

1: Tamper 4 event does not erase the backup registers and device secrets(2).

Bit 2 **TAMP3NOER**: Tamper 3 no erase

0: Tamper 3 event erases the backup registers and all device secrets(1).

1: Tamper 3 event does not erase the backup registers and device secrets(2).

Bit 1 **TAMP2NOER**: Tamper 2 no erase

0: Tamper 2 event erases the backup registers and all device secrets(1).

1: Tamper 2 event does not erase the backup registers and device secrets(2).

Bit 0 **TAMP1NOER**: Tamper 1 no erase

0: Tamper 1 event erases the backup registers and all device secrets(1).

1: Tamper 1 event does not erase the backup registers and device secrets(2).

1. Refer to the list of device secrets erased by tamp_erase and tamp_erase_ercfg signals described in *Table 528: TAMP interconnection*.
2. Refer to the effects of tamp_potential and tamp_potential_ercfg signals described in *Table 528: TAMP interconnection*.

55.7.3 TAMP control register 3 (TAMP_CR3)

This register can be protected against non-secure access. Refer to *Section 55.4.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 55.4.7: 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.	ITAMP13NOER	ITAMP12NOER	ITAMP11NOER	Res.	ITAMP9NOER	ITAMP8NOER	ITAMP7NOER	ITAMP6NOER	ITAMP5NOER	Res.	ITAMP3NOER	ITAMP2NOER	ITAMP1NOER
			rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

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 **ITAMP13NOER**: Internal Tamper 13 no erase
0: Internal Tamper 13 event erases the backup registers and all device secrets(1).
1: Internal Tamper 13 event does not erase the backup registers and device secrets(2).

Bit 11 **ITAMP12NOER**: Internal Tamper 12 no erase
0: Internal Tamper 12 event erases the backup registers and all device secrets(1).
1: Internal Tamper 12 event does not erase the backup registers and device secrets(2).

Bit 10 **ITAMP11NOER**: Internal Tamper 11 no erase
0: Internal Tamper 11 event erases the backup registers and all device secrets(1).
1: Internal Tamper 11 event does not erase the backup registers and device secrets(2).

Bit 9 Reserved, must be kept at reset value.

Bit 8 **ITAMP9NOER**: Internal Tamper 9 no erase
0: Internal Tamper 9 event erases the backup registers and all device secrets(1).
1: Internal Tamper 9 event does not erase the backup registers and device secrets(2).

Bit 7 **ITAMP8NOER**: Internal Tamper 8 no erase
0: Internal Tamper 8 event erases the backup registers and all device secrets(1).
1: Internal Tamper 8 event does not erase the backup registers and device secrets(2).

Bit 6 **ITAMP7NOER**: Internal Tamper 7 no erase
0: Internal Tamper 7 event erases the backup registers and all device secrets(1).
1: Internal Tamper 7 event does not erase the backup registers and device secrets(2).

Bit 5 **ITAMP6NOER**: Internal Tamper 6 no erase
0: Internal Tamper 6 event erases the backup registers and all device secrets(1).
1: Internal Tamper 6 event does not erase the backup registers and device secrets(2).

Bit 4 **ITAMP5NOER**: Internal Tamper 5 no erase
0: Internal Tamper 5 event erases the backup registers and all device secrets(1).
1: Internal Tamper 5 event does not erase the backup registers and device secrets(2).

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 and all device secrets(1).
1: Internal Tamper 3 event does not erase the backup registers and device secrets(2).

Bit 1 **ITAMP2NOER**: Internal Tamper 2 no erase
0: Internal Tamper 2 event erases the backup registers and all device secrets(1).
1: Internal Tamper 2 event does not erase the backup registers and device secrets(2).

Bit 0 **ITAMP1NOER**: Internal Tamper 1 no erase
0: Internal Tamper 1 event erases the backup registers and all device secrets(1).
1: Internal Tamper 1 event does not erase the backup registers and device secrets(2).

1. Refer to the list of device secrets erased by tamp_erase and tamp_erase_ercfg signals described in *Table 528: TAMP interconnection*
2. Refer to the effects of tamp_potential and tamp_potential_ercfg signals described in *Table 528: TAMP interconnection*.

55.7.4 TAMP filter control register (TAMP_FLTCR)

This register can be protected against non-secure access. Refer to *Section 55.4.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 55.4.7: 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.

55.7.5 TAMP active tamper control register 1 (TAMP_ATCR1)

This register can be protected against non-secure access. Refer to *Section 55.4.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 55.4.7: 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.	ATCKSEL[3:0]			
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
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 530: Minimum ATPER value*.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **ATCKSEL[3:0]**: Active tamper RTC asynchronous prescaler clock selection

These bits selects the RTC asynchronous prescaler stage output. The selected clock is CK_ATPRE. ATCKSEL[3] is reserved, read only, and tied to 0 when the active tamper prescaler extension is not implemented.

0000: RTCCLK is selected
0001: RTCCLK/2 is selected
0010: RTCCLK/4 is selected
0011: RTCCLK/8 is selected
0100: RTCCLK/16 is selected
0101: RTCCLK/32 is selected
0110: RTCCLK/64 is selected
0111: RTCCLK/128 is selected
1011: RTCCLK/2048 is selected when (PREDIV_A+1) = 128 and (PREDIV_S+1) is a multiple of 16. This value is supported only when the active tamper prescaler extension is supported. Refer to *Section 55.3: TAMP implementation*.
Others: Reserved

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.

55.7.6 TAMP active tamper seed register (TAMP_ATSEEDR)

This register can be protected against non-secure access. Refer to *Section 55.4.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 55.4.7: 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.

55.7.7 TAMP active tamper output register (TAMP_ATOR)

This register can be protected against non-secure access. Refer to *Section 55.4.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 55.4.7: 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.

55.7.8 TAMP active tamper control register 2 (TAMP_ATCR2)

This register can be protected against non-secure access. Refer to *Section 55.4.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 55.4.7: 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.

55.7.9 TAMP secure configuration register (TAMP_SECCFGR)

If TZEN = 1, this register can be written only when the APB access is secure. If TZEN = 0, BKPRWSEC[7:0], BKPWSEC[7:0] and BHKLOCK can be written with non-secure APB access, and TAMPSEC, CNT1SEC 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_PRIVCFGR configuration (refer to *Section 55.4.7: TAMP privilege protection modes*).

Address offset: 0x20

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
TAMPSEC	BHKLOCK	Res.	Res.	Res.	Res.	Res.	Res.	BKPWSEC[7:0]							
rw	rs							rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT1SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPRWSEC[7:0]							
rw								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TAMPSEC**: Tamper protection (excluding monotonic counters and backup registers)

0: Tamper configuration and interrupt can be written when the APB access is secure or non-secure.

1: Tamper configuration and interrupt can be written only when the APB access is secure.

Note: Refer to Section 55.4.5: TAMP secure protection modes for details on the read protection.

Bit 30 **BHKLOCK**: Boot hardware key lock

This bit can be read and can only be written to 1 by software. It is cleared by hardware together with the backup registers following a tamper detection event or when the readout protection (RDP) is disabled.

0: The Backup registers from TAMP_BKP0R to TAMP_BKP7R can be accessed according to the Protection zone they belong to.

1: The backup registers from TAMP_BKP0R to TAMP_BKP7R cannot be accessed neither in read nor in write (they are read as 0 and write ignore).

Bits 29:24 Reserved, must be kept at reset value.

Bits 23:16 **BKPWSEC[7:0]**: Backup registers write protection offset

BKPWSEC value must be from 0 to 32.

Protection zone 2 is defined for backup registers from TAMP_BKPyR (y = BKPRWSEC) to TAMP_BKPzR (z = BKPWSEC-1, with BKPWSEC > BKPRWSEC):

- if TZEN=1, these backup registers can be written only with secure access.
- They can be read with secure or non-secure access.

If BKPWSEC = 0 or if BKPWSEC ≤ BKPRWSEC: there is no protection zone 2.

Protection zone 3 is defined for backup registers from TAMP_BKPtR (t = BKPWSEC if BKPWSEC ≥ BKPRWSEC, else t = BKPRWSEC).

- They can be read or written with secure or non-secure access.

If BKPWSEC = 32: there is no protection zone 3.

Refer to *Figure 624: Backup registers protection zones*.

Note: If TZEN=0: the protection zone 2 can be read and written with non-secure access.

Note: If BKPWPRIV is set, BKPRWSEC[7:0] can be written only in privileged mode.

Bit 15 **CNT1SEC**: Monotonic counter 1 secure protection
0: Monotonic counter 1 (TAMP_COUNT1R) can be read and written when the APB access is secure or non-secure.
1: Monotonic counter 1 (TAMP_COUNT1R) can be read and written only when the APB access is secure.

Bits 14:8 Reserved, must be kept at reset value.

Bits 7:0 **BKPRWSEC[7:0]**: Backup registers read/write protection offset
BKPRWSEC value must be from 0 to 32.

Protection zone 1 is defined for backup registers from TAMP_BKP0R to TAMP_BKPxR (x = BKPRWSEC-1, with BKPRWSEC ≥ 1).
– if TZEN=1, these backup registers can be read and written only with secure access.
If BKPRWSEC = 0: there is no protection zone 1.

Refer to *Figure 624: Backup registers protection zones*.

Note: If TZEN=0: the protection zone 1 can be read and written with non-secure access.
Note: If BKPRWPRIV is set, BKPRWSEC[7:0] can be written only in privileged mode.

55.7.10 TAMP privilege configuration register (TAMP_PRIVCFGR)

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_SECCFGR configuration (refer to *Section 55.4.5: 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
CNT1 PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

Bit 31 **TAMPPRIV**: Tamper privilege protection (excluding backup registers)
0: Tamper configuration and interrupt can be written with privileged or unprivileged access.
1: Tamper configuration and interrupt can be written only with privileged access.
Note: Refer to Section 55.4.7: 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 with privileged or unprivileged access.
1: Backup registers zone 2 can be written only with privileged access.

Bit 29 **BKPRWPRIV**: Backup registers zone 1 privilege protection
0: Backup registers zone 1 can be read and written with privileged or unprivileged access.
1: Backup registers zone 1 can be read and written only with privileged access

Bits 28:16 Reserved, must be kept at reset value.

Bit 15 **CNT1PRIV**: Monotonic counter 1 privilege protection
0: Monotonic counter 1 (TAMP_COUNT1R) can be read and written when the APB access is privileged or non-privileged.
1: Monotonic counter 1 (TAMP_COUNT1R) can be read and written only when the APB access is privileged.

Bits 14:0 Reserved, must be kept at reset value.

55.7.11 TAMP interrupt enable register (TAMP_IER)

This register can be protected against non-secure access. Refer to *Section 55.4.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 55.4.7: 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.	ITAMP13IE	ITAMP12IE	ITAMP11IE	Res.	ITAMP9IE	ITAMP8IE	ITAMP7IE	ITAMP6IE	ITAMP5IE	Res.	ITAMP3IE	ITAMP2IE	ITAMP1IE
			rw	rw	rw		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.	TAMP8IE	TAMP7IE	TAMP6IE	TAMP5IE	TAMP4IE	TAMP3IE	TAMP2IE	TAMP1IE
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **ITAMP13IE**: Internal tamper 13 interrupt enable
0: Internal tamper 13 interrupt disabled.
1: Internal tamper 13 interrupt enabled.

Bit 27 **ITAMP12IE**: Internal tamper 12 interrupt enable
0: Internal tamper 12 interrupt disabled.
1: Internal tamper 12 interrupt enabled.

Bit 26 **ITAMP11IE**: Internal tamper 11 interrupt enable
0: Internal tamper 11 interrupt disabled.
1: Internal tamper 11 interrupt enabled.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9IE**: Internal tamper 9 interrupt enable
0: Internal tamper 9 interrupt disabled.
1: Internal tamper 9 interrupt enabled.

Bit 23 **ITAMP8IE**: Internal tamper 8 interrupt enable
0: Internal tamper 8 interrupt disabled.
1: Internal tamper 8 interrupt enabled.

Bit 22 **ITAMP7IE**: Internal tamper 7 interrupt enable
0: Internal tamper 7 interrupt disabled.
1: Internal tamper 7 interrupt enabled.

Bit 21 **ITAMP6IE**: Internal tamper 6 interrupt enable
0: Internal tamper 6 interrupt disabled.
1: Internal tamper 6 interrupt enabled.

Bit 20 **ITAMP5IE**: Internal tamper 5 interrupt enable
0: Internal tamper 5 interrupt disabled.
1: Internal tamper 5 interrupt enabled.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **ITAMP3IE**: Internal tamper 3 interrupt enable
0: Internal tamper 3 interrupt disabled.
1: Internal tamper 3 interrupt enabled.

Bit 17 **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.

55.7.12 TAMP status register (TAMP_SR)

This register can be protected against non-secure access. Refer to *Section 55.4.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 55.4.7: 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.	ITAMP13F	ITAMP12F	ITAMP11F	Res.	ITAMP9F	ITAMP8F	ITAMP7F	ITAMP6F	ITAMP5F	Res.	ITAMP3F	ITAMP2F	ITAMP1F
			r	r	r		r	r	r	r	r		r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP8F	TAMP7F	TAMP6F	TAMP5F	TAMP4F	TAMP3F	TAMP2F	TAMP1F
								r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **ITAMP13F**: Internal tamper 13 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 13.

Bit 27 **ITAMP12F**: Internal tamper 12 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 12.

Bit 26 **ITAMP11F**: Internal tamper 11 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 11.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9F**: Internal tamper 9 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 9.

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 **ITAMP7F**: Internal tamper 7 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 7.

Bit 21 **ITAMP6F**: Internal tamper 6 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 6.

Bit 20 **ITAMP5F**: Internal tamper 5 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 5.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **ITAMP3F**: Internal tamper 3 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 3.

Bit 17 **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.

55.7.13 TAMP non-secure masked interrupt status register (TAMP_MISR)

This register can be protected against non-privileged access. Refer to *Section 55.4.7: 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.	ITAMP13MF	ITAMP12MF	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	Res.	ITAMP3MF	ITAMP2MF	ITAMP1MF
			r	r	r		r	r	r	r	r		r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP8MF	TAMP7MF	TAMP6MF	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
								r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **ITAMP13MF**: internal tamper 13 non-secure interrupt masked flag
This flag is set by hardware when the internal tamper 13 non-secure interrupt is raised.

Bit 27 **ITAMP12MF**: internal tamper 12 non-secure interrupt masked flag
This flag is set by hardware when the internal tamper 12 non-secure interrupt is raised.

Bit 26 **ITAMP11MF**: internal tamper 11 non-secure interrupt masked flag
This flag is set by hardware when the internal tamper 11 non-secure interrupt is raised.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9MF**: internal tamper 9 non-secure interrupt masked flag
This flag is set by hardware when the internal tamper 9 non-secure interrupt is raised.

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 **ITAMP7MF**: Internal tamper 7 tamper non-secure interrupt masked flag
This flag is set by hardware when the internal tamper 7 non-secure interrupt is raised.

Bit 21 **ITAMP6MF**: Internal tamper 6 non-secure interrupt masked flag
This flag is set by hardware when the internal tamper 6 non-secure interrupt is raised.

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.

55.7.14 TAMP secure masked interrupt status register (TAMP_SMISR)

This register can be protected against non-secure access. Refer to *Section 55.4.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 55.4.7: 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.	ITAMP13MF	ITAMP12MF	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	Res.	ITAMP3MF	ITAMP2MF	ITAMP1MF
			r	r	r		r	r	r	r	r		r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP8MF	TAMP7MF	TAMP6MF	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
								r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **ITAMP13MF**: internal tamper 13 secure interrupt masked flag
This flag is set by hardware when the internal tamper 13 secure interrupt is raised.

Bit 27 **ITAMP12MF**: internal tamper 12 secure interrupt masked flag
This flag is set by hardware when the internal tamper 12 secure interrupt is raised.

Bit 26 **ITAMP11MF**: internal tamper 11 secure interrupt masked flag
This flag is set by hardware when the internal tamper 11 secure interrupt is raised.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9MF**: internal tamper 9 secure interrupt masked flag
This flag is set by hardware when the internal tamper 9 secure interrupt is raised.

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 **ITAMP7MF**: Internal tamper 7 secure interrupt masked flag
This flag is set by hardware when the internal tamper 7 secure interrupt is raised.

Bit 21 **ITAMP6MF**: Internal tamper 6 secure interrupt masked flag
This flag is set by hardware when the internal tamper 6 secure interrupt is raised.

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.

55.7.15 TAMP status clear register (TAMP_SCR)

This register can be protected against non-secure access. Refer to *Section 55.4.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 55.4.7: 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.	CITAMP13F	CITAMP12F	CITAMP11F	Res.	CITAMP9F	CITAMP8F	CITAMP7F	CITAMP6F	CITAMP5F	Res.	CITAMP3F	CITAMP2F	CITAMP1F
			w	w	w		w	w	w	w	w		w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP8F	CTAMP7F	CTAMP6F	CTAMP5F	CTAMP4F	CTAMP3F	CTAMP2F	CTAMP1F
								w	w	w	w	w	w	w	w

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **CITAMP13F**: Clear ITAMP13 detection flag
Writing 1 in this bit clears the ITAMP13F bit in the TAMP_SR register.

Bit 27 **CITAMP12F**: Clear ITAMP12 detection flag
Writing 1 in this bit clears the ITAMP12F bit in the TAMP_SR register.

Bit 26 **CITAMP11F**: Clear ITAMP11 detection flag
Writing 1 in this bit clears the ITAMP11F bit in the TAMP_SR register.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **CITAMP9F**: Clear ITAMP9 detection flag
Writing 1 in this bit clears the ITAMP9F bit in the TAMP_SR register.

Bit 23 **CITAMP8F**: Clear ITAMP8 detection flag
Writing 1 in this bit clears the ITAMP8F bit in the TAMP_SR register.

Bit 22 **CITAMP7F**: Clear ITAMP7 detection flag
Writing 1 in this bit clears the ITAMP7F bit in the TAMP_SR register.

Bit 21 **CITAMP6F**: Clear ITAMP6 detection flag
Writing 1 in this bit clears the ITAMP6F bit in the TAMP_SR register.

Bit 20 **CITAMP5F**: Clear ITAMP5 detection flag
Writing 1 in this bit clears the ITAMP5F bit in the TAMP_SR register.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **CITAMP3F**: Clear ITAMP3 detection flag
Writing 1 in this bit clears the ITAMP3F bit in the TAMP_SR register.

Bit 17 **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.

55.7.16 TAMP monotonic counter 1 register (TAMP_COUNT1R)

This register can be protected against non-secure access. Refer to *Section 55.4.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 55.4.7: 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.

55.7.17 TAMP erase configuration register (TAMP_ERCFGR)

This register can be protected against non-secure access. Refer to *Section 55.4.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 55.4.7: TAMP 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.	Res.	Res.	Res.	Res.	Res.	ERCFG0
															rw

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ERCFG0**: Configurable device secrets(1) configuration

0: Configurable device secrets are not included in the device secrets protected by TAMP peripheral

1: Configurable device secrets are is included in the device secrets protected by TAMP peripheral

1. Refer to tamp_erase_ercfg0 and tamp_potential_ercfg0 signals in *Table 526: TAMP input/output pins* and *Table 528: TAMP interconnection*.

55.7.18 TAMP backup x register (TAMP_BKPxR)

This register can be protected against non-secure access. Refer to *Section 55.4.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 55.4.7: 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.
In the default (ERASE) 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.

55.7.19 TAMP register map

Table 535. 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.	ITAMP13E	ITAMP12E	ITAMP11E	Res.	ITAMP9E	ITAMP8E	ITAMP7E	ITAMP6E	ITAMP5E	Res.	ITAMP3E	ITAMP2E	ITAMP1E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP8E	TAMP7E	TAMP6E	TAMP5E	TAMP4E	TAMP3E	TAMP2E	TAMP1E
	Reset value				0	0	0		0	0	0	0	0		0	0	0									0	0	0	0	0	0	0	0
0x04	**TAMP_CR2**	TAMP8TRG	TAMP7TRG	TAMP6TRG	TAMP5TRG	TAMP4TRG	TAMP3TRG	TAMP2TRG	TAMP1TRG	BKERASE	BKBLOCK	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	0
0x08	**TAMP_CR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP13NOER	ITAMP12NOER	ITAMP11NOER	Res.	ITAMP9NOER	ITAMP8NOER	ITAMP7NOER	ITAMP6NOER	ITAMP5NOER	Res.	ITAMP3NOER	ITAMP2NOER	ITAMP1NOER
	Reset value																				0	0	0		0	0	0	0	0		0	0	0
0x0C	**TAMP_FLTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]		
	Reset value																									0	0	0	0	0	0	0	0
0x10	**TAMP_ATCR1**	FLTEN	ATOSHARE	Res.	Res.	Res.	AT PER[2:0]			Res.	Res.	Res.	Res.	ATCK SEL[3:0]				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					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]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x20	**TAMP_SEC CFGR**	TAMPSEC	BHKLOCK	Res.	Res.	Res.	Res.	Res.	Res.	BKPWSEC[7:0]								CNT1SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPRWSEC[7:0]							
	Reset value	0	0							0	0	0	0	0	0	0	0	0								0	0	0	0	0	0	0	0
0x240	**TAMP_ PRIVCFGR**	TAMPPRIV	BKPWPRIV	BKPRWPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT1PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0														0															

Table 535. 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.	ITAMP13IE	ITAMP12IE	ITAMP11IE	Res.	ITAMP9IE	ITAMP8IE	ITAMP7IF	ITAMP6IE	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	0	0	0	0	0	0
0x30	**TAMP_SR**	Res.	Res.	Res.	ITAMP13F	ITAMP12F	ITAMP11F	Res.	ITAMP9F	ITAMP8F	ITAMP7F	ITAMP6F	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	0	0	0	0	0	0
0x34	**TAMP_MISR**	Res.	Res.	Res.	ITAMP13MF	ITAMP12MF	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	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	0	0	0	0	0	0
0x38	**TAMP_SMISR**	Res.	Res.	Res.	ITAMP13F	ITAMP12F	ITAMP11F	Res.	ITAMP9F	ITAMP8F	ITAMP7F	ITAMP6MF	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	0	0	0	0	0	0
0x3C	**TAMP_SCR**	Res.	Res.	Res.	CITAMP13F	CITAMP12F	CITAMP11F	Res.	CITAMP9F	CITAMP8F	CITAMP7F	CITAMP6F	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	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
0x54	**TAMP_ ERCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERCFG0
	Reset value																																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.

56 Inter-integrated circuit (I2C) interface

56.1 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.

56.2 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 56.3: 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 i2c_pclk reprogramming

- Wakeup from stop mode.
- Autonomous functionality in Stop mode.

56.3 I2C implementation

Table 536. STM32U575/585 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 20 mA output drive I/Os (up to 1 Mbit/s)	X	X	X	X
Independent clock	X	X	X	X
Autonomous mode	X	X	X	X
Wakeup from Stop mode	X[2]	X[2]	X[3]	X[2]
SMBus/PMBus	X	X	X	X

1. X = supported.
2. Wakeup supported from Stop 0 and Stop 1 modes.
3. Wakeup supported from Stop 0, Stop 1 and Stop 2 modes.

56.4 I2C functional description

In addition to receiving and transmitting data, this interface converts them from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the I^2C bus by a data pin (SDA) and by a clock pin (SCL). It can be connected with a standard (up to 100 kHz), Fast-mode (up to 400 kHz) or Fast-mode Plus (up to 1 MHz) I^2C bus.

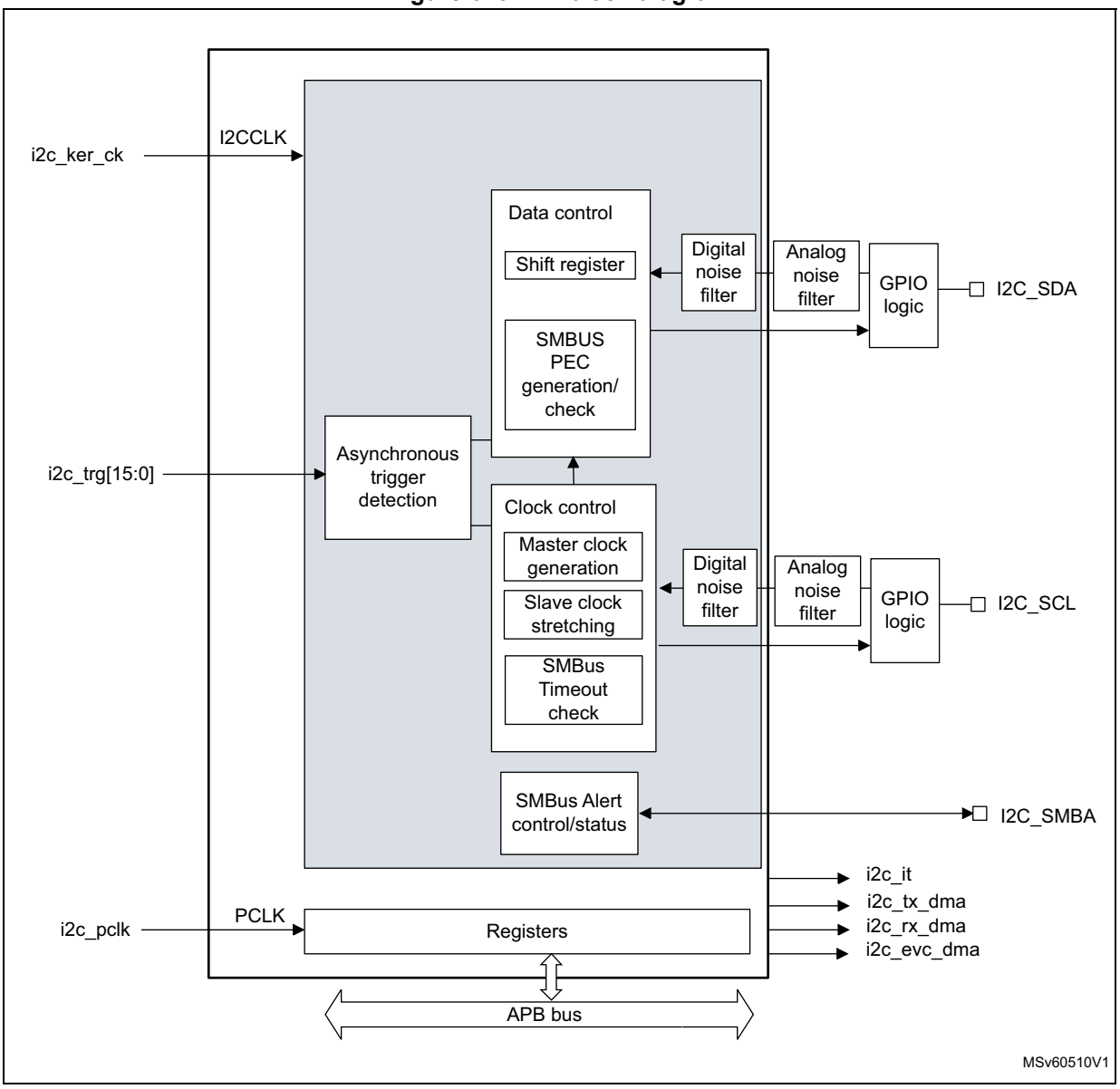
This interface can also be connected to an SMBus with the data pin (SDA) and clock pin (SCL).

If SMBus feature is supported, the additional optional SMBus Alert pin (SMBA) is also available.

56.4.1 I2C block diagram

The block diagram of the I2C interface is shown in *Figure 626*.

Figure 626. I2C block diagram



The I2C is clocked by an independent clock source, which allows the I2C to operate independently from the i2c_pclk frequency.

For I2C I/Os supporting 20 mA output current drive for Fast-mode Plus operation, the driving capability is enabled through control bits in the system configuration controller (SYSCFG). *Refer to Section 56.3: I2C implementation.*

56.4.2 I2C pins and internal signals

Table 537. 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 538. 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_trg[15:0]	Input	I2C triggers
i2c_it	Output	I2C interrupts, refer to *Table 553* 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)
i2c_evc_dma	Output	I2C event control DMA request (I2C_EVC)

Table 539. I2C1, I2C2, I2C4 interconnection

Signal name	Source/destination
i2c_trg0	gpdma1_ch0_tc
i2c_trg1	gpdma1_ch1_tc
i2c_trg2	gpdma1_ch2_tc
i2c_trg3	gpdma1_ch3_tc
i2c_trg4	exti5
i2c_trg5	exti9
i2c_trg6	lptim1_ch1
i2c_trg7	lptim2_ch1
i2c_trg8	comp1_out
i2c_trg9	comp2_out
i2c_trg10	rtc_alra_trg
i2c_trg11	rtc_wut_trg
i2c_trg12	-
i2c_trg13	-
i2c_trg14	-
i2c_trg15	-

Table 540. I2C3 interconnection

Signal name	Source/destination
i2c_trg0	lpdma1_ch0_tc
i2c_trg1	lpdma1_ch1_tc
i2c_trg2	lpdma1_ch2_tc
i2c_trg3	lpdma1_ch3_tc
i2c_trg4	exti5
i2c_trg5	exti8
i2c_trg6	lptim1_ch1
i2c_trg7	lptim3_ch1
i2c_trg8	comp1_out
i2c_trg9	comp2_out
i2c_trg10	rtc_alra_trg
i2c_trg11	rtc_wut_trg
i2c_trg12	-
i2c_trg13	-
i2c_trg14	-
i2c_trg15	-

56.4.3 I2C clock requirements

The I2C kernel is clocked by i2c_ker_ck.

The i2c_ker_ck 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 i2c_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 i2c_pclk, this clock must respect the conditions for t_{I2CCLK}.

56.4.4 Mode selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

By default, it operates in slave mode. The interface automatically switches from slave to master when it generates a START condition, and from master to slave if an arbitration loss or a STOP generation occurs, allowing multimaster capability.

Communication flow

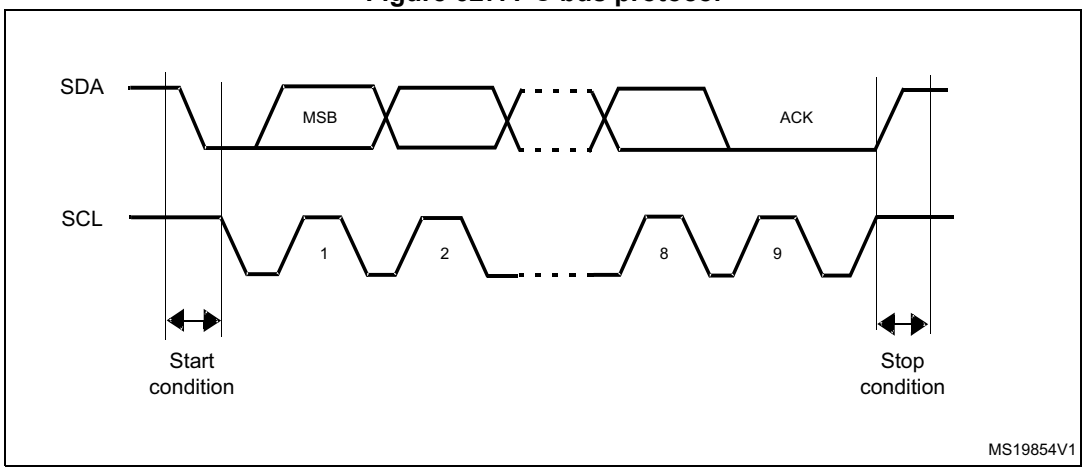
In Master mode, the I2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a START condition and ends with a STOP condition. Both START and STOP conditions are generated in master mode by software.

In Slave mode, the interface is capable of recognizing its own addresses (7 or 10-bit), and the general call address. The general call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the START condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.

A ninth clock pulse follows the eight clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter (see *Figure 627*).

Figure 627. I^2C bus protocol



Acknowledge can be enabled or disabled by software. The I2C interface addresses can be selected by software.

56.4.5 I2C initialization

Enabling and disabling the peripheral

The I2C peripheral clock must be configured and enabled in the clock controller, then the I2C can be enabled by setting the PE bit in the I2C_CR1 register.

When the I2C is disabled (PE = 0), the I^{2}C performs a software reset. Refer to *Section 56.4.6* for more details.

Noise filters

Before enabling the I2C peripheral by setting the PE bit in I2C_CR1 register, the user must configure the noise filters, if needed. By default, an analog noise filter is present on the SDA and SCL inputs. This analog filter is compliant with the I^{2}C specification, which requires the suppression of spikes with a pulse width up to 50 ns in Fast-mode and Fast-mode Plus. The user can disable this analog filter by setting the ANFOFF bit, and/or select a digital filter by configuring the DNF[3:0] bit in the I2C_CR1 register.

When the digital filter is enabled, the level of the SCL or the SDA line is internally changed only if it remains stable for more than DNF x i2c_ker_ck periods. This allows spikes with a programmable length of 1 to 15 i2c_ker_ck periods to be suppressed.

Table 541. 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	Functionality in Stop mode is not supported when digital filter is enabled.

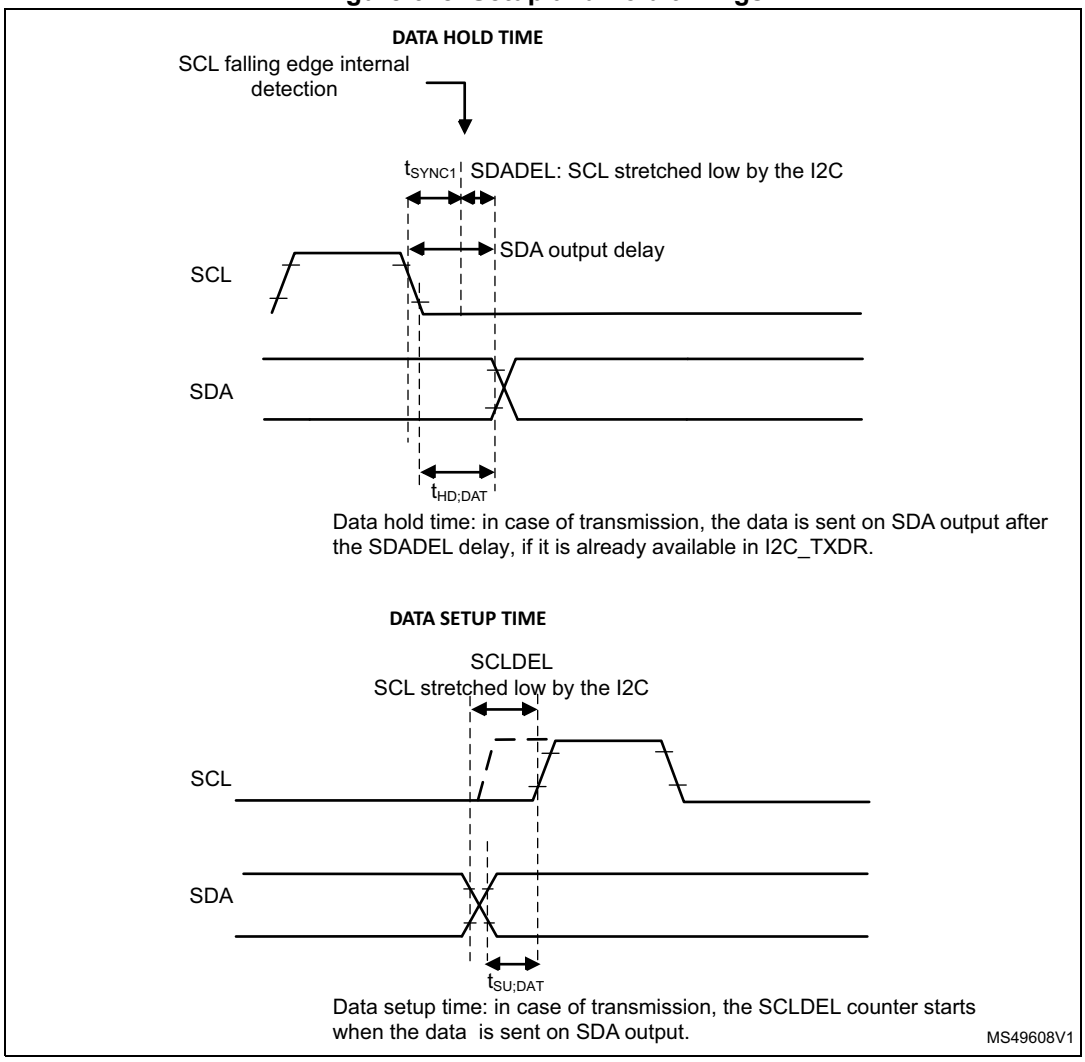
Caution: Changing the filter configuration is not allowed when the I2C is enabled.

I2C timings

The timings must be configured in order to guarantee correct data hold and setup times, used in master and slave modes. This is done by programming the PRESC[3:0], SCLDEL[3:0] and SDADEL[3:0] bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C configuration window.

Figure 628. 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 i2c_ker_ck clock (2 to 3 i2c_ker_ck 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 542* for t_f, t_r, $t_{HD;DAT}$ and $t_{VD;DAT}$ standard values.

- After t_{SDADEL} delay, or after sending SDA output in case the slave had to stretch the clock because the data was not yet written in I2C_TXDR register, SCL line is kept at low level during the setup time. This setup time is t_{SCLDEL} = (SCLDEL+1) x t_{PRESC} where t_{PRESC} = (PRESC+1) x t_{I2CCLK}.

  t_{SCLDEL} impacts the setup time $t_{SU;DAT}$.

In order to bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program SCLDEL in such a way that:

$\{[t_{r\ (max)} + t_{SU;DAT\ (min)}] / [(PRESC+1)] \times t_{I2CCLK}]\} - 1 \leq SCLDEL$

Refer to *Table 542* for t_r and $t_{SU;DAT}$ standard values.

The SDA and SCL transition time values to be used are the ones in the application. Using the maximum values from the standard increases the constraints for the SDADEL and SCLDEL calculation, but ensures the feature whatever the application.

Note: *At every clock pulse, after SCL falling edge detection, the I2C master or slave stretches SCL low during at least [(SDADEL + SCLDEL + 1) x (PRESC + 1) + 1] x t_{I2CCLK}, in both*

transmission and reception modes. In transmission mode, if the data is not yet written in I2C_TXDR when SDADEL counter is finished, the I2C keeps on stretching SCL low until the next data is written. Then new data MSB is sent on SDA output, and SCLDEL counter starts, continuing stretching SCL low to guarantee the data setup time.

If NOSTRETCH = 1 in slave mode, the SCL is not stretched. Consequently the SDADEL must be programmed in such a way to guarantee also a sufficient setup time.

Table 542. I^2C-SMBus specification data setup and hold times

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min.	Max	Min.	Max	Min.	Max	Min.	Max	
$t_{HD;DAT}$	Data hold time	0	-	0	-	0	-	0.3	-	µs
$t_{VD;DAT}$	Data valid time	-	3.45	-	0.9	-	0.45	-	-	
$t_{SU;DAT}$	Data setup time	250	-	100	-	50	-	250	-	ns
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Additionally, in master mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0] and SCLL[7:0] bits in the I2C_TIMINGR register.

- When the SCL falling edge is internally detected, a delay is inserted before releasing the SCL output.
  This delay is $t_{SCLL} = (SCLL + 1) \times t_{PRESC}$ where $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$.
  t_{SCLL} impacts the SCL low time t_{LOW}.
- When the SCL rising edge is internally detected, a delay is inserted before forcing the SCL output to low level. This delay is $t_{SCLH} = (SCLH + 1) \times t_{PRESC}$, where $t_{PRESC} = (PRESC+1) \times t_{I2CCLK}$. t_{SCLH} impacts the SCL high time t_{HIGH}.

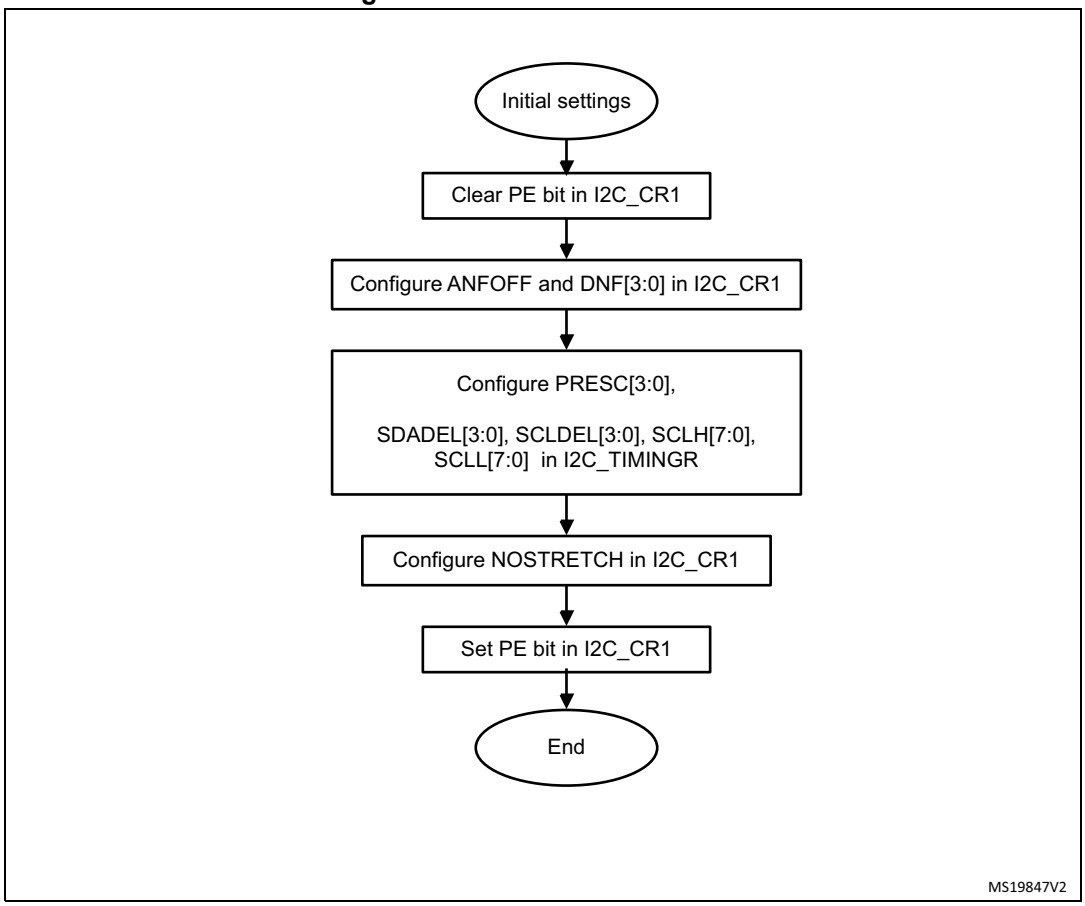
Refer to *I2C master initialization* for more details.

Caution: Changing the timing configuration is not allowed when the I2C is enabled.

The I2C slave NOSTRETCH mode must also be configured before enabling the peripheral. Refer to *I2C slave initialization* for more details.

Caution: Changing the NOSTRETCH configuration is not allowed when the I2C is enabled.

Figure 629. I2C initialization flow



56.4.6 Software reset

A software reset can be performed by clearing the PE bit in the I2C_CR1 register. In that case I2C lines SCL and SDA are released. Internal states machines are reset and communication control bits, as well as status bits come back to their reset value. The configuration registers are not impacted.

Here is the list of impacted register bits:

1. I2C_CR2 register: START, STOP, NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, OVR

and in addition when the SMBus feature is supported:

1. I2C_CR2 register: PECBYTE
2. I2C_ISR register: PECERR, TIMEOUT, ALERT

PE must be kept low during at least three APB clock cycles in order to perform the software reset. This is ensured by writing the following software sequence:

1. Write PE = 0
2. Check PE = 0
3. Write PE = 1.

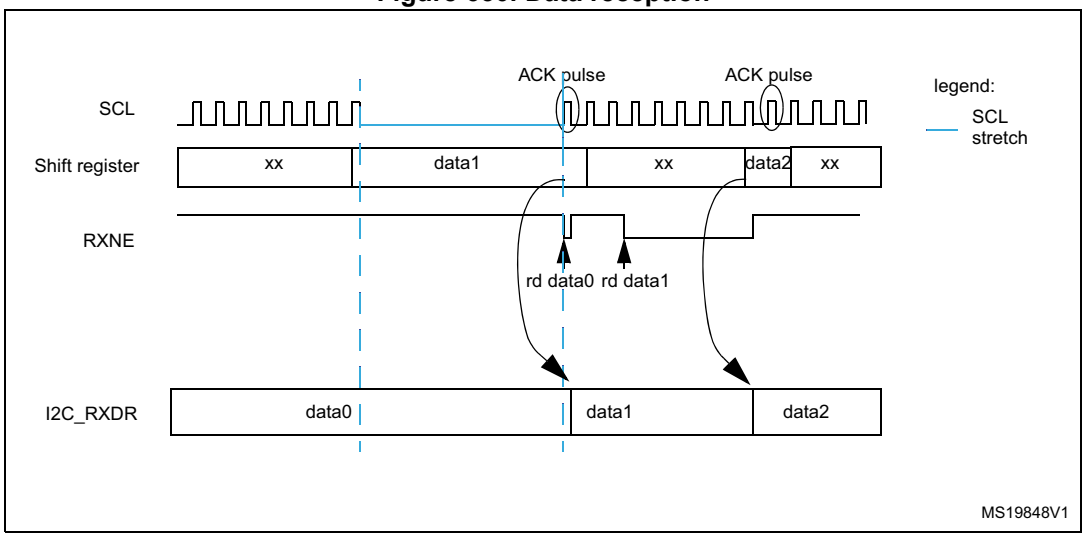
56.4.7 Data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

Reception

The SDA input fills the shift register. After the eighth SCL pulse (when the complete data byte is received), the shift register is copied into I2C_RXDR register if it is empty (RXNE = 0). If RXNE = 1, meaning that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch is inserted between the eighth and ninth SCL pulse (before the acknowledge pulse).

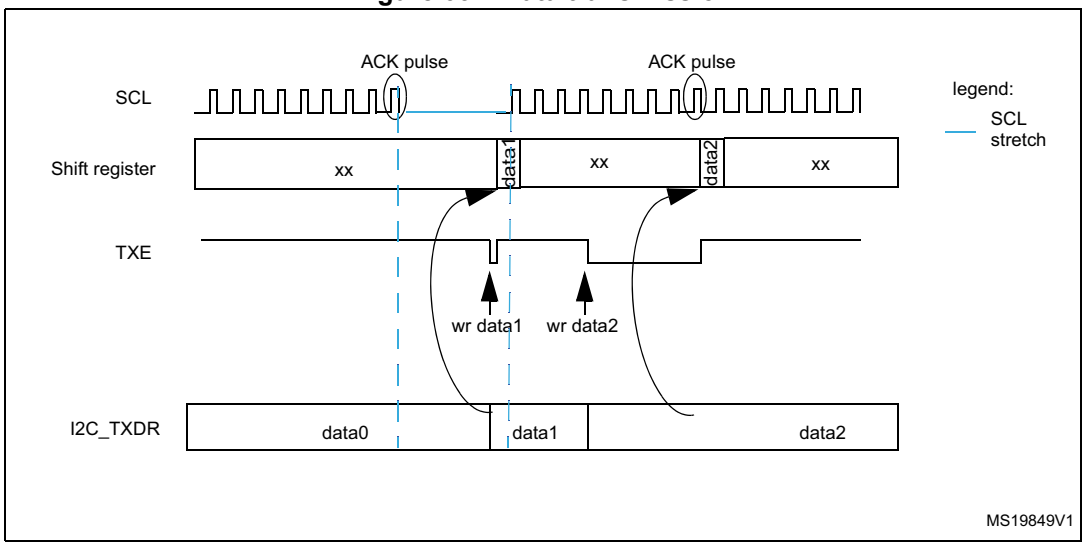
Figure 630. Data reception



Transmission

If the I2C_TXDR register is not empty (TXE=0), its content is copied into the shift register after the ninth SCL pulse (the Acknowledge pulse). Then the shift register content is shifted out on SDA line. If TXE = 1, meaning that no data is written yet in I2C_TXDR, SCL line is stretched low until I2C_TXDR is written. The stretch is done after the ninth SCL pulse.

Figure 631. Data transmission



Hardware transfer management

The I2C has a byte counter embedded in hardware in order to manage byte transfer and to close the communication in various modes such as:

- NACK, STOP and ReSTART generation in master mode
- ACK control in slave receiver mode
- PEC generation/checking when SMBus feature is supported

The byte counter is always used in master mode. By default it is disabled in slave mode, but it can be enabled by software by setting the SBC (slave byte control) bit in the I2C_CR1 register.

The number of bytes to be transferred is programmed in the NBYTES[7:0] bit field in the I2C_CR2 register. If the number of bytes to be transferred (NBYTES) is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this mode, the TCR flag is set when the number of bytes programmed in NBYTES is transferred, and an interrupt is generated if TCIE is set. SCL is stretched as long as TCR flag is set. TCR is cleared by software when NBYTES is written to a non-zero value.

When the NBYTES counter is reloaded with the last number of bytes, RELOAD bit must be cleared.

When RELOAD=0 in master mode, the counter can be used in 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 543. 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

56.4.8 I2C slave mode

I2C slave initialization

In order to work in slave mode, the user must enable at least one slave address. Two registers I2C_OAR1 and I2C_OAR2 are available in order to program the slave own addresses OA1 and OA2.

- OA1 can be configured either in 7-bit mode (by default) or in 10-bit addressing mode by setting the OA1MODE bit in the I2C_OAR1 register.

  OA1 is enabled by setting the OA1EN bit in the I2C_OAR1 register.
- If additional slave addresses are required, the second slave address OA2 can be configured. Up to 7 OA2 LSB can be masked by configuring the OA2MSK[2:0] bits in the I2C_OAR2 register. Therefore for OA2MSK configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6] or OA2[7] are compared with the received address. As soon as OA2MSK is not equal to 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX), which are not acknowledged. If OA2MSK=7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

  These reserved addresses can be acknowledged if they are enabled by the specific enable bit, if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK=0.

  OA2 is enabled by setting the OA2EN bit in the I2C_OAR2 register.
- The general call address is enabled by setting the GCEN bit in the I2C_CR1 register.

When the I2C is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the slave uses its clock stretching capability, which means that it stretches the SCL signal at low level when needed, in order to perform software actions. If the master does not support clock stretching, the I2C must be configured with NOSTRETCH = 1 in the I2C_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled the user must read the ADDCODE[6:0] bits in the I2C_ISR register in order to check which address matched. DIR flag must also be checked in order to know the transfer direction.

Slave clock stretching (NOSTRETCH = 0)

In default mode, the I2C slave stretches the SCL clock in the following situations:

- When the ADDR flag is set: the received address matches with one of the enabled slave addresses. This stretch is released when the ADDR flag is cleared by software setting the ADDRCF bit.
- In transmission, if the previous data transmission is completed and no new data is written in I2C_TXDR register, or if the first data byte is not written when the ADDR flag is cleared (TXE = 1). This stretch is released when the data is written to the I2C_TXDR register.
- In reception when the I2C_RXDR register is not read yet and a new data reception is completed. This stretch is released when I2C_RXDR is read.
- When TCR = 1 in Slave Byte Control mode, reload mode (SBC=1 and RELOAD=1), meaning that the last data byte has been transferred. This stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] field.
- After SCL falling edge detection, the I2C stretches SCL low during [(SDADEL+SCLDEL+1) x (PRESC+1) + 1] x t_{I2CCLK}.

Slave without clock stretching (NOSTRETCH = 1)

When NOSTRETCH = 1 in the I2C_CR1 register, the I2C slave does not stretch the SCL signal.

- The SCL clock is not stretched while the ADDR flag is set.
- In transmission, the data must be written in the I2C_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if the user clears the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, he ensures that the OVR status is provided, even for the first data to be transmitted.
- In reception, the data must be read from the I2C_RXDR register before the ninth SCL pulse (ACK pulse) of the next data byte occurs. If not an overrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Slave byte control mode

In order to allow byte ACK control in slave reception mode, The Slave byte control mode must be enabled by setting the SBC bit in the I2C_CR1 register. This is required to be compliant with SMBus standards.

The Reload mode must be selected in order to allow byte ACK control in slave reception mode (RELOAD = 1). To get control of each byte, NBYTES must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the eighth and ninth SCL pulses. The user can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit in the I2C_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and next byte can be received.

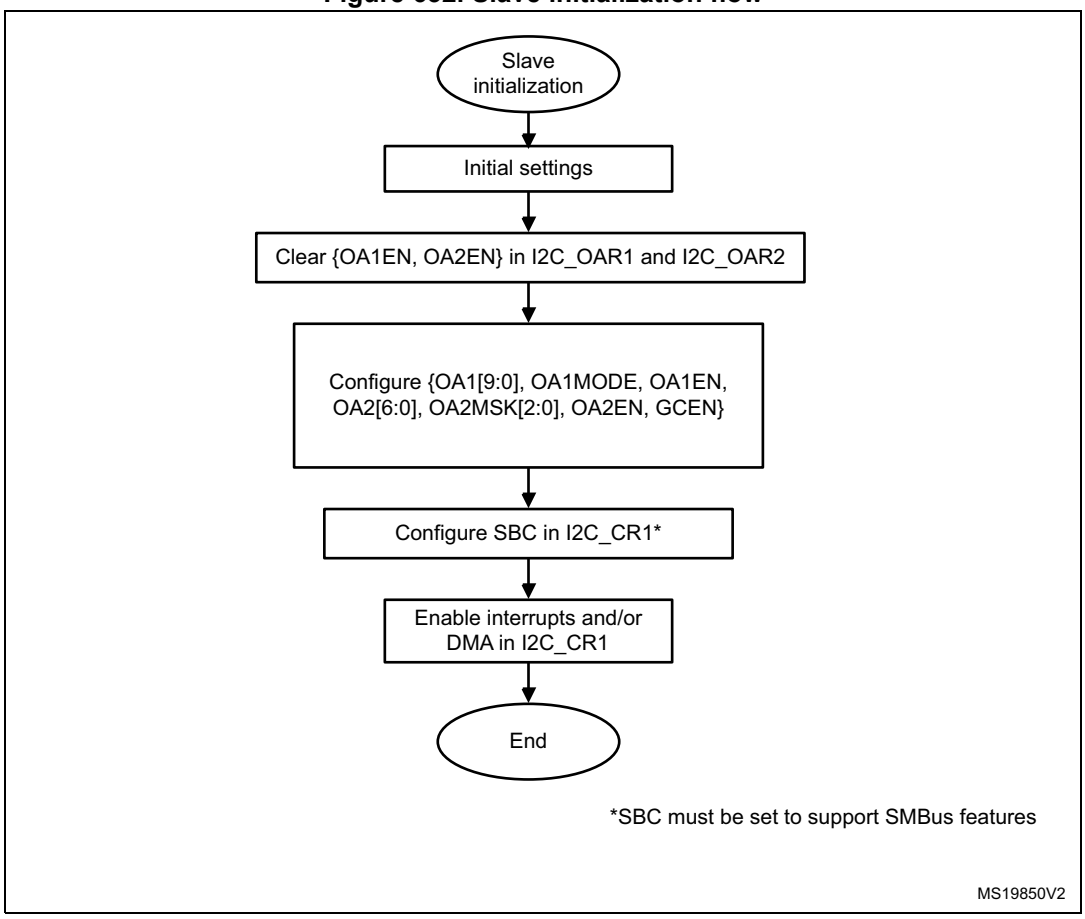
NBYTES can be loaded with a value greater than 0x1, and in this case, the reception flow is continuous during NBYTES data reception.

Note: The SBC bit must be configured when the I2C is disabled, or when the slave is not addressed, or when ADDR = 1.

The RELOAD bit value can be changed when ADDR = 1, or when TCR = 1.

Caution: The Slave byte control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH = 1 is not allowed.

Figure 632. Slave initialization flow



Slave transmitter

A transmit interrupt status (TXIS) is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit is set in the I2C_CR1 register.

The TXIS bit is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

When a NACK is received, the NACKF bit is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit is set in the I2C_CR1 register. The slave automatically releases the SCL and SDA lines in order to let the master perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When a STOP is received and the STOPIE bit is set in the I2C_CR1 register, the STOPF flag is set in the I2C_ISR register and an interrupt is generated. In most applications, the SBC bit is usually programmed to ‘0’. In this case, If TXE = 0 when the slave address is received (ADDR = 1), the user can choose either to send the content of the I2C_TXDR register as the first data byte, or to flush the I2C_TXDR register by setting the TXE bit in order to program a new data byte.

In Slave byte control mode (SBC = 1), the number of bytes to be transmitted must be programmed in NBYTES in the address match interrupt subroutine (ADDR = 1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES.

Caution: When NOSTRETCH = 1, the SCL clock is not stretched while the ADDR flag is set, so the user cannot flush the I2C_TXDR register content in the ADDR subroutine, in order to program the first data byte. The first data byte to be sent must be previously programmed in the I2C_TXDR register:

- This data can be the data written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to be sent, the I2C_TXDR register can be flushed by setting the TXE bit in order to program a new data byte. The STOPF bit must be cleared only after these actions, in order to guarantee that they are executed before the first data transmission starts, following the address acknowledge.

  If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVR flag is set).

  If a TXIS event is needed, (transmit interrupt or transmit DMA request), the user must set the TXIS bit in addition to the TXE bit, in order to generate a TXIS event.

Figure 633. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 0

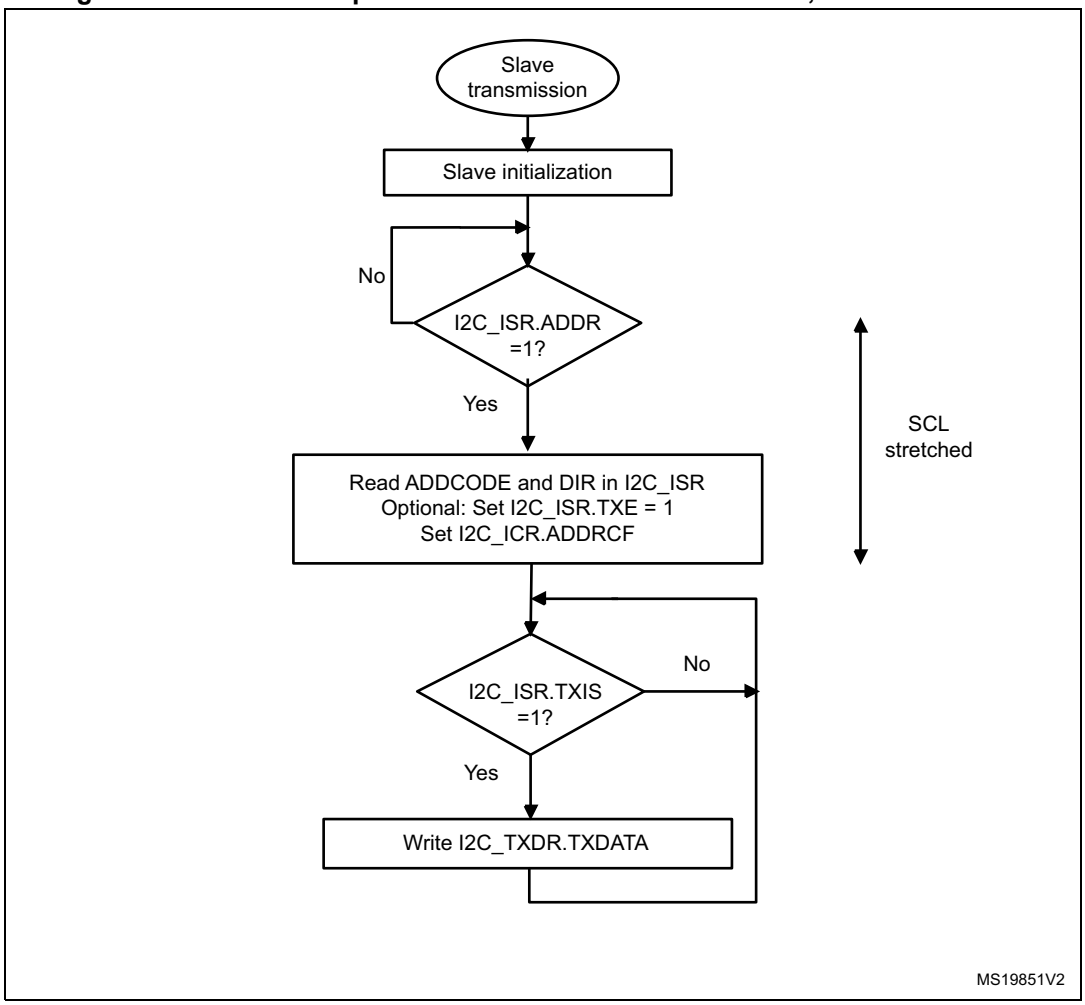


Figure 634. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 1

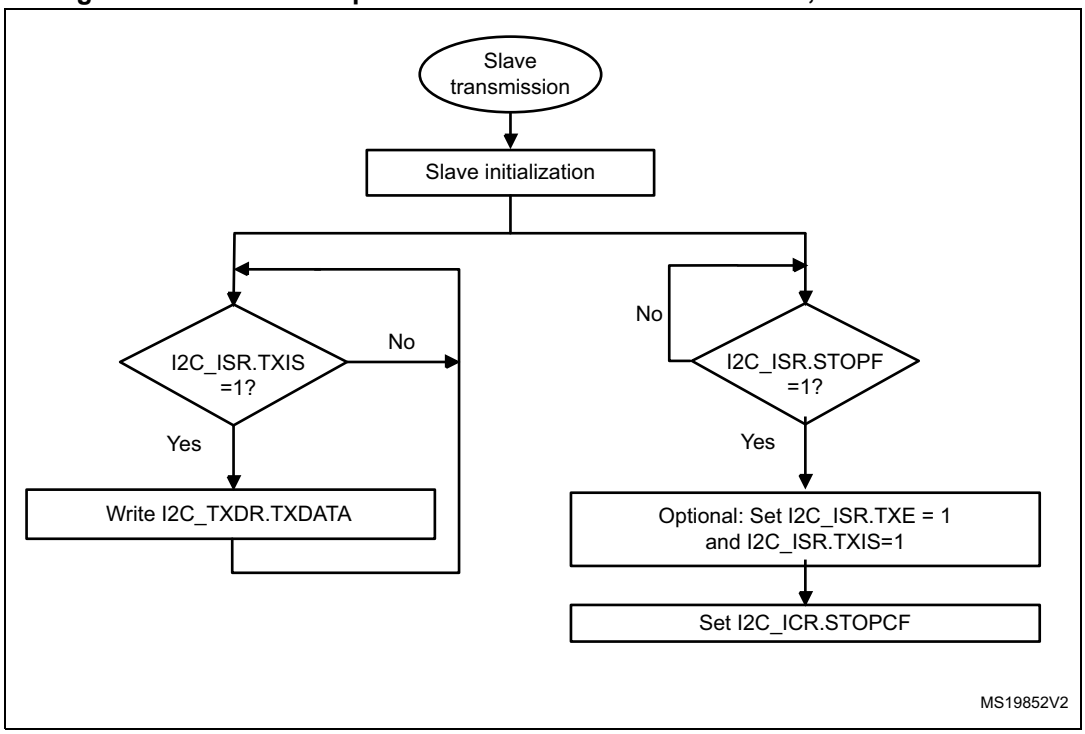
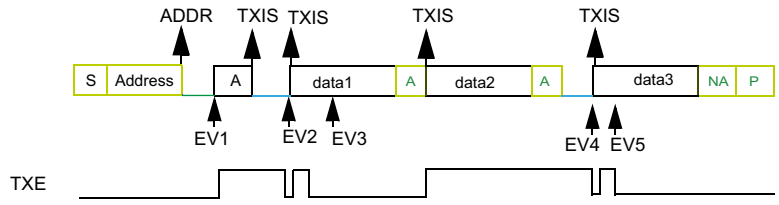


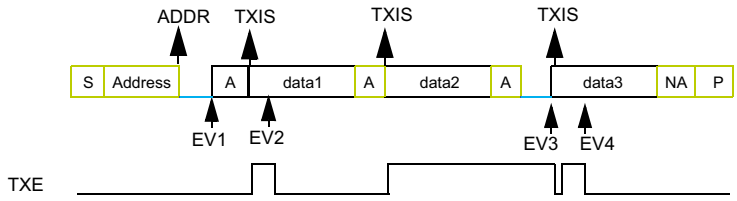
Figure 635. Transfer bus diagrams for I2C slave transmitter

Example I2C slave transmitter 3 bytes with 1st data flushed, NOSTRETCH=0:



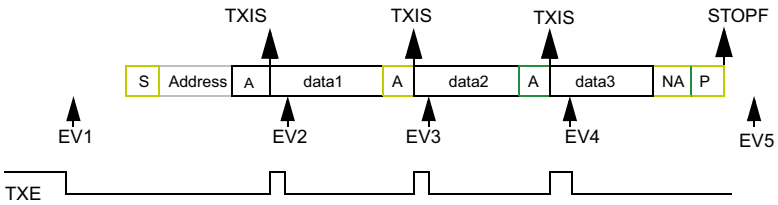
EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF
EV2: TXIS ISR: wr data1
EV3: TXIS ISR: wr data2
EV4: TXIS ISR: wr data3
EV5: TXIS ISR: wr data4 (not sent)

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 636. Transfer sequence flow for slave receiver with NOSTRETCH = 0

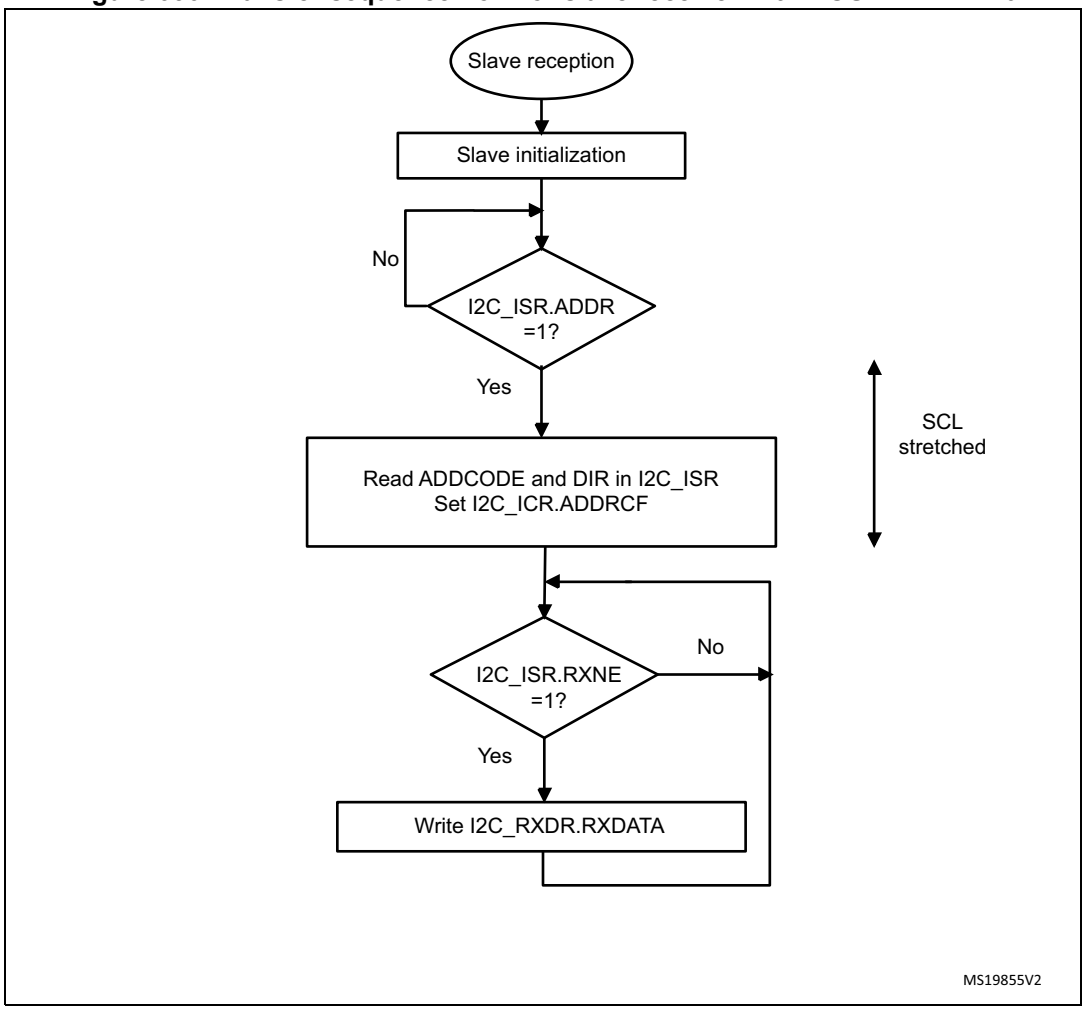


Figure 637. Transfer sequence flow for slave receiver with NOSTRETCH = 1

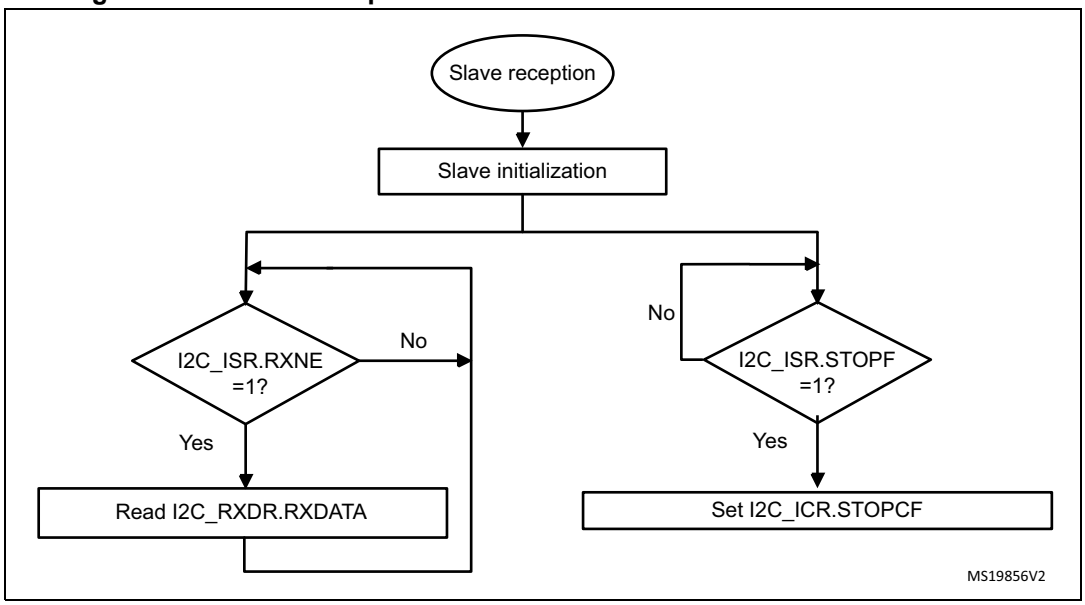
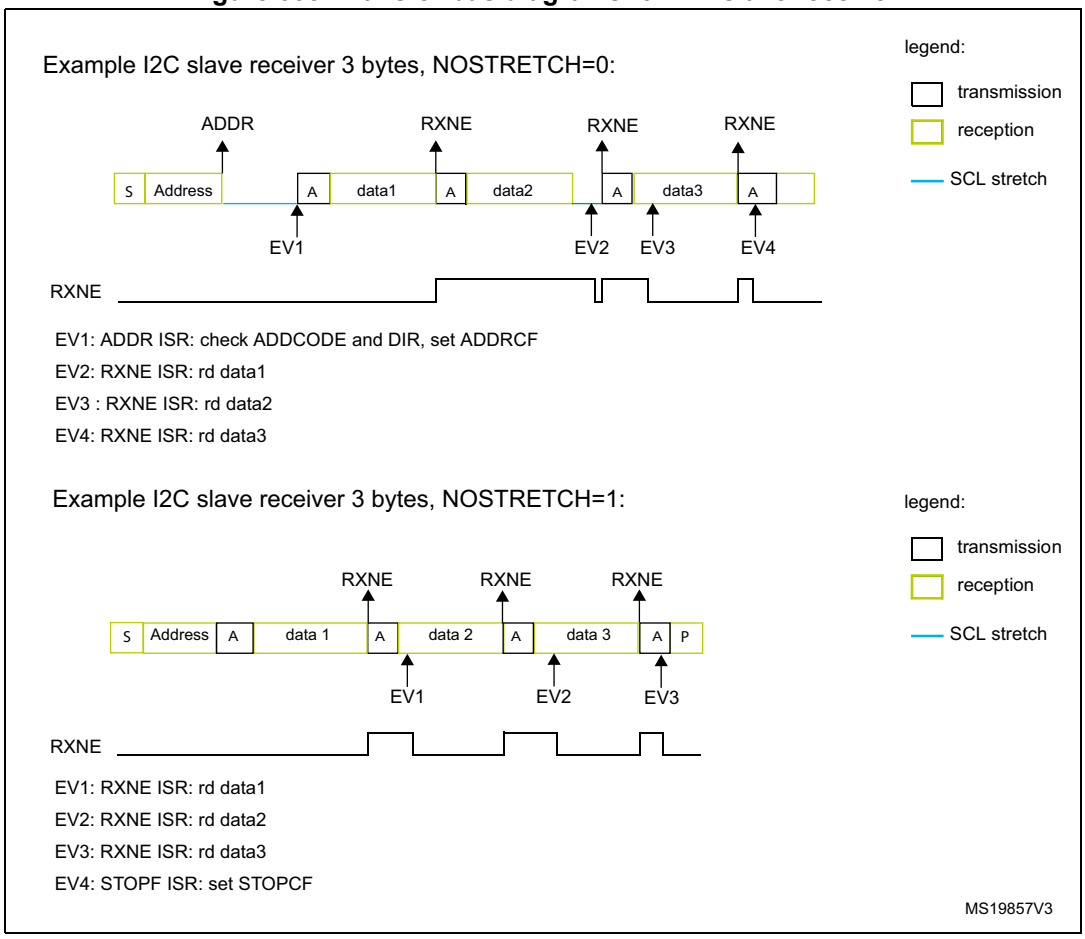


Figure 638. Transfer bus diagrams for I2C slave receiver



56.4.9 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:

$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+1) + (SCLL+1)] \times (PRESC+1) \times t_{I2CCLK}\}$

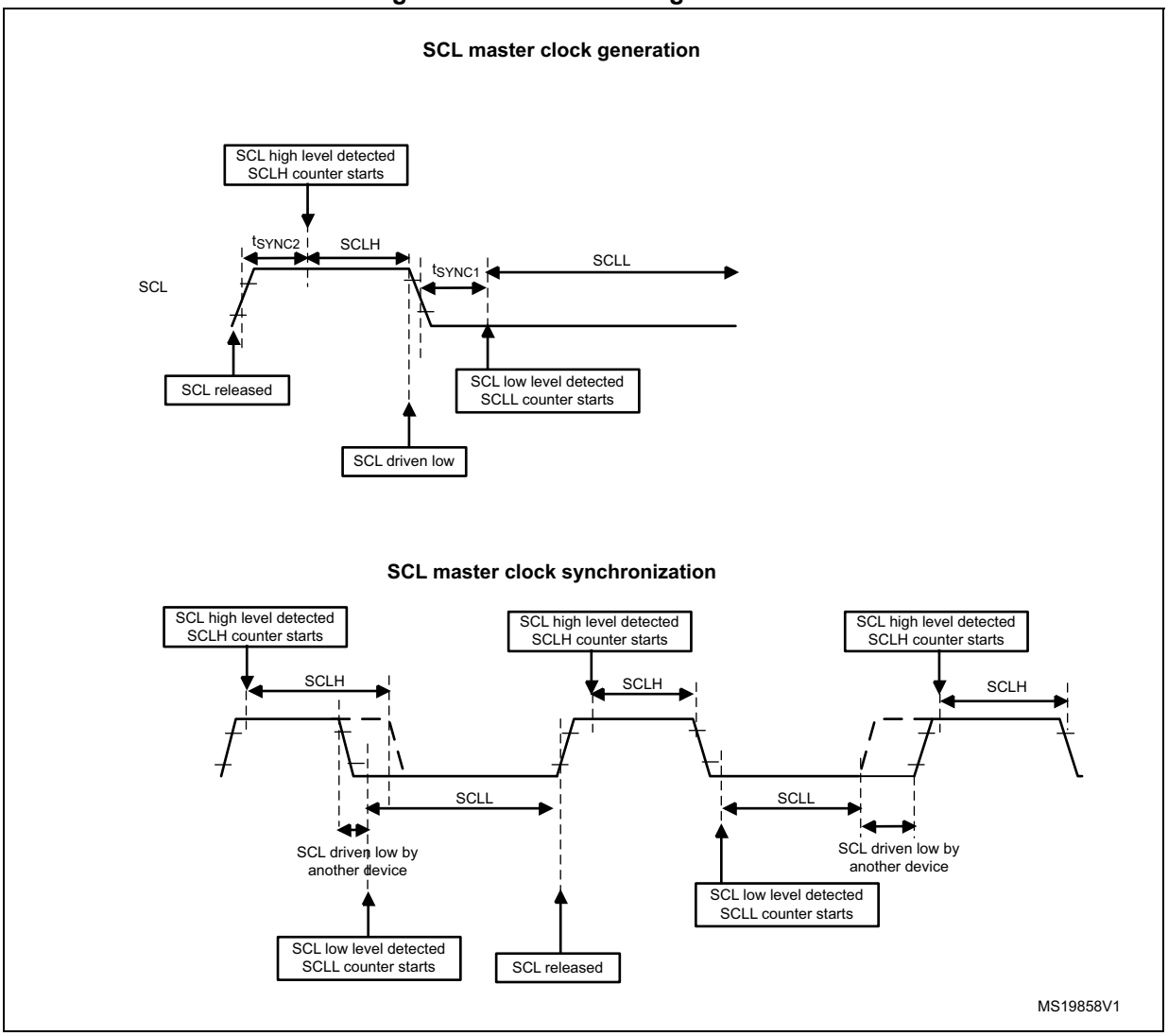
The duration of t_{SYNC1} depends on these parameters:

- 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 i2c_ker_ck clock (2 to 3 i2c_ker_ck 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 i2c_ker_ck clock (2 to 3 i2c_ker_ck periods)

Figure 639. 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 544. 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 56.4.10: I2C_TIMINGR register configuration examples* for examples of I2C_TIMINGR settings vs. i2c_ker_ck frequency.

Master communication initialization (address phase)

In order to initiate the communication, the user must program the following parameters for the addressed slave in the I2C_CR2 register:

- Addressing mode (7-bit or 10-bit): ADD10
- Slave address to be sent: SADD[9:0]
- Transfer direction: RD_WRN
- In case of 10-bit address read: HEAD10R bit. HEAD10R must be configure to indicate if the complete address sequence must be sent, or only the header in case of a direction change.
- The number of bytes to be transferred: NBYTES[7:0]. If the number of bytes is equal to or greater than 255 bytes, NBYTES[7:0] must initially be filled with 0xFF.

The user must then set the START bit in I2C_CR2 register. Changing all the above bits is not allowed when START bit is set.

Then the master automatically sends the START condition followed by the slave address as soon as it detects that the bus is free (BUSY = 0) and after a delay of t_{BUF}.

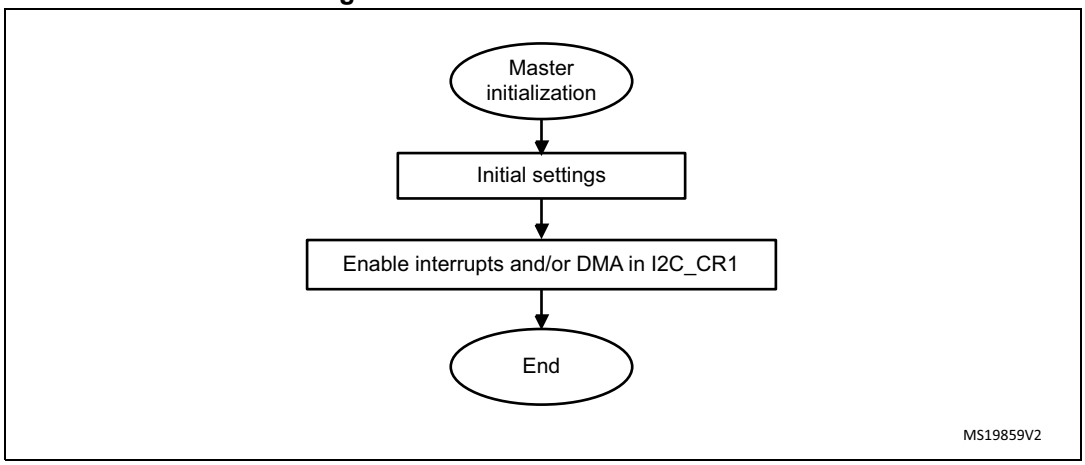
In case of an arbitration loss, the master automatically switches back to slave mode and can acknowledge its own address if it is addressed as a slave.

Note: The START bit is reset by hardware when the slave address has been sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware if an arbitration loss occurs.
In 10-bit addressing mode, when the Slave Address first 7 bits are NACKed by the slave,

the master re-launches automatically the slave address transmission until ACK is received. In this case ADDRCF must be set if a NACK is received from the slave, in order to stop sending the slave address.
If the I2C is addressed as a slave (ADDR = 1) while the START bit is set, the I2C switches to slave mode and the START bit is cleared.

Note: The same procedure is applied for a Repeated Start condition. In this case BUSY = 1.

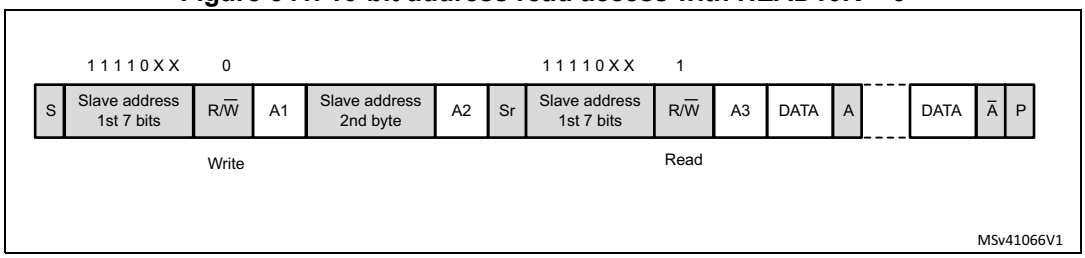
Figure 640. Master initialization flow



Initialization of a master receiver addressing a 10-bit address slave

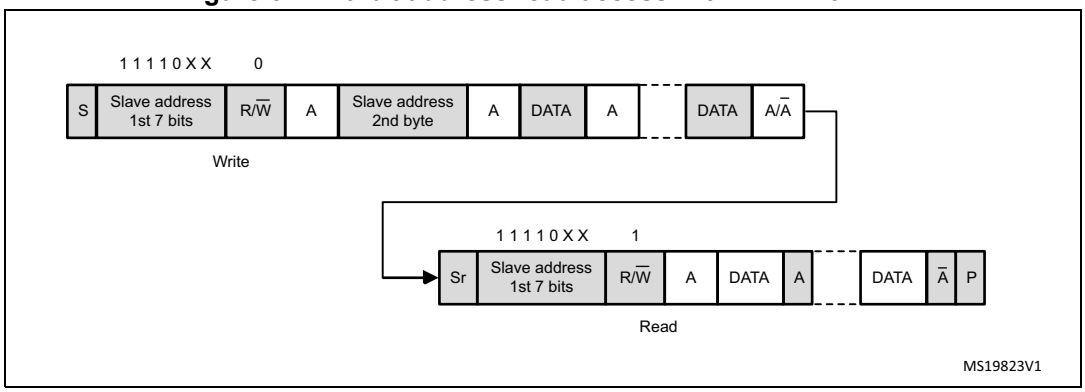
- If the slave address is in 10-bit format, the user can choose to send the complete read sequence by clearing the HEAD10R bit in the I2C_CR2 register. In this case the master automatically sends the following complete sequence after the START bit is set: (Re)Start + Slave address 10-bit header Write + Slave address second byte + REStart + Slave address 10-bit header Read

Figure 641. 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 642. 10-bit address read access with HEAD10R = 1



Master transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the ninth SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit is set in the I2C_CR1 register. The flag is cleared when the I2C_TXDR register is written with the next data byte to be transmitted.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to be sent is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when NBYTES data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

The TXIS flag is not set when a NACK is received.

- When RELOAD=0 and NBYTES data have been transferred:
  - In automatic end mode (AUTOEND=1), a STOP is automatically sent.
  - In software end mode (AUTOEND=0), the TC flag is set and the SCL line is stretched low in order to perform software actions:

    A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition is sent on the bus.

    A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.
- If a NACK is received: the TXIS flag is not set, and a STOP condition is automatically sent after the NACK reception. the NACKF flag is set in the I2C_ISR register, and an interrupt is generated if the NACKIE bit is set.

Figure 643. Transfer sequence flow for I2C master transmitter for N≤255 bytes

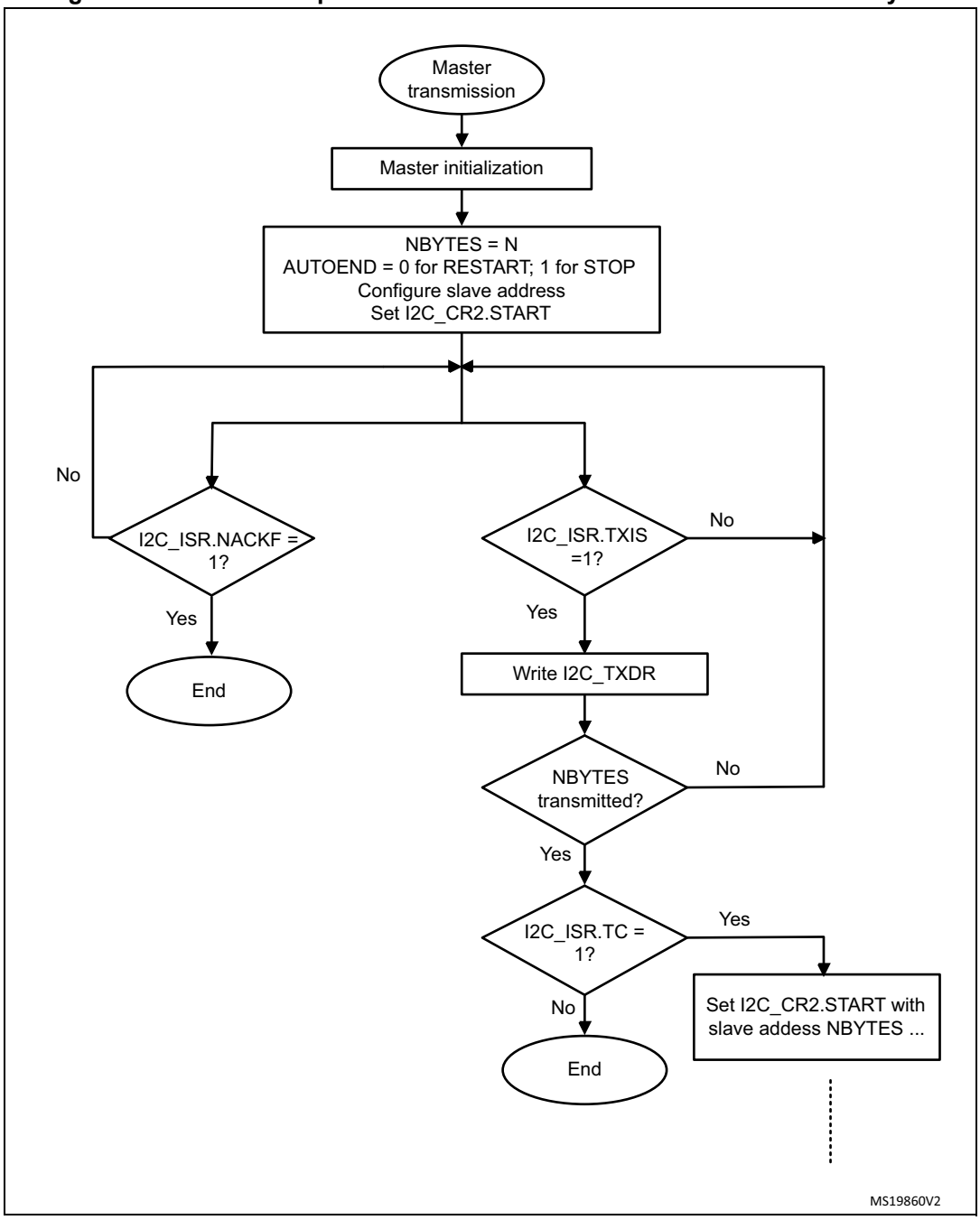


Figure 644. Transfer sequence flow for I2C master transmitter for N>255 bytes

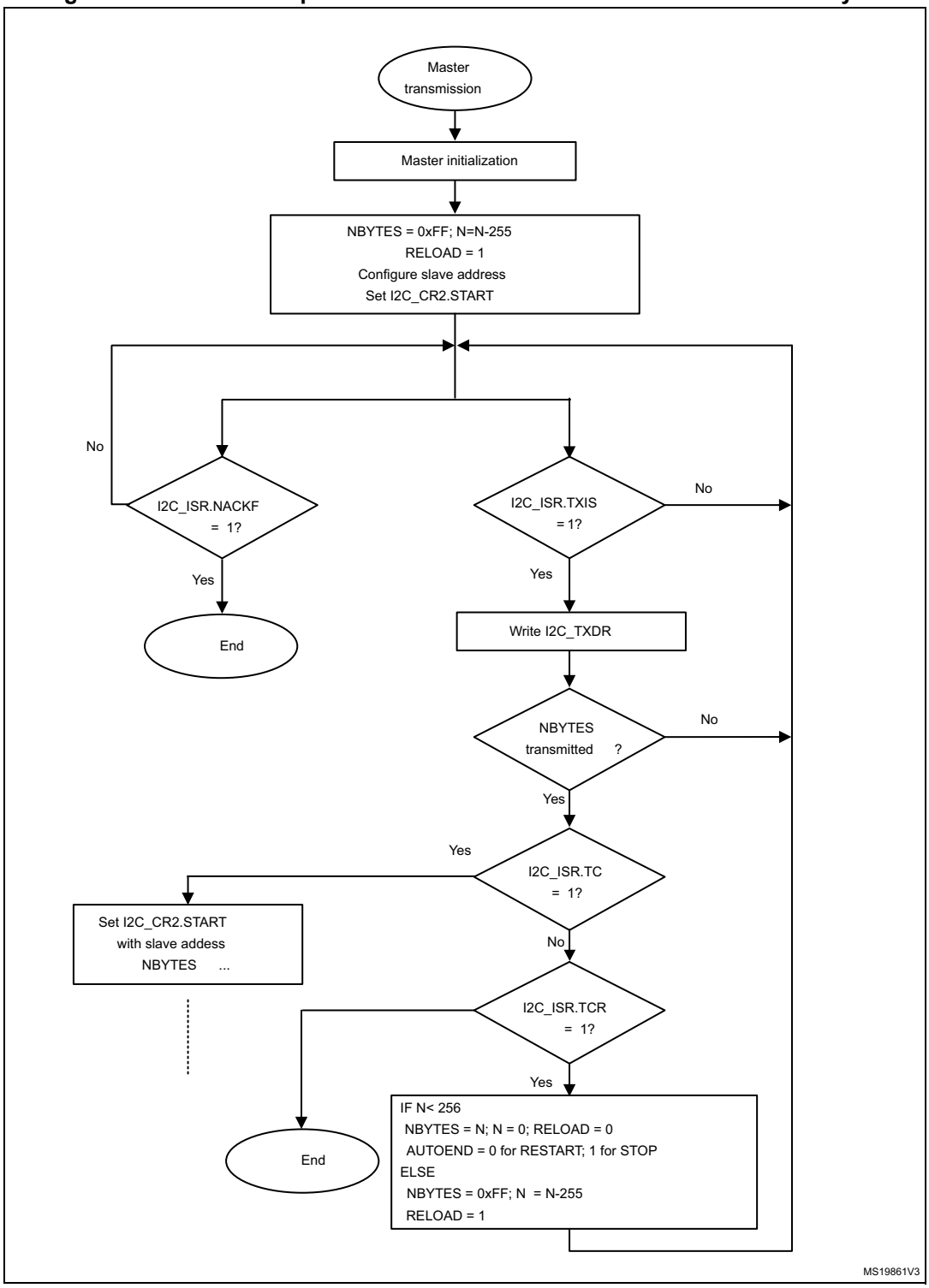
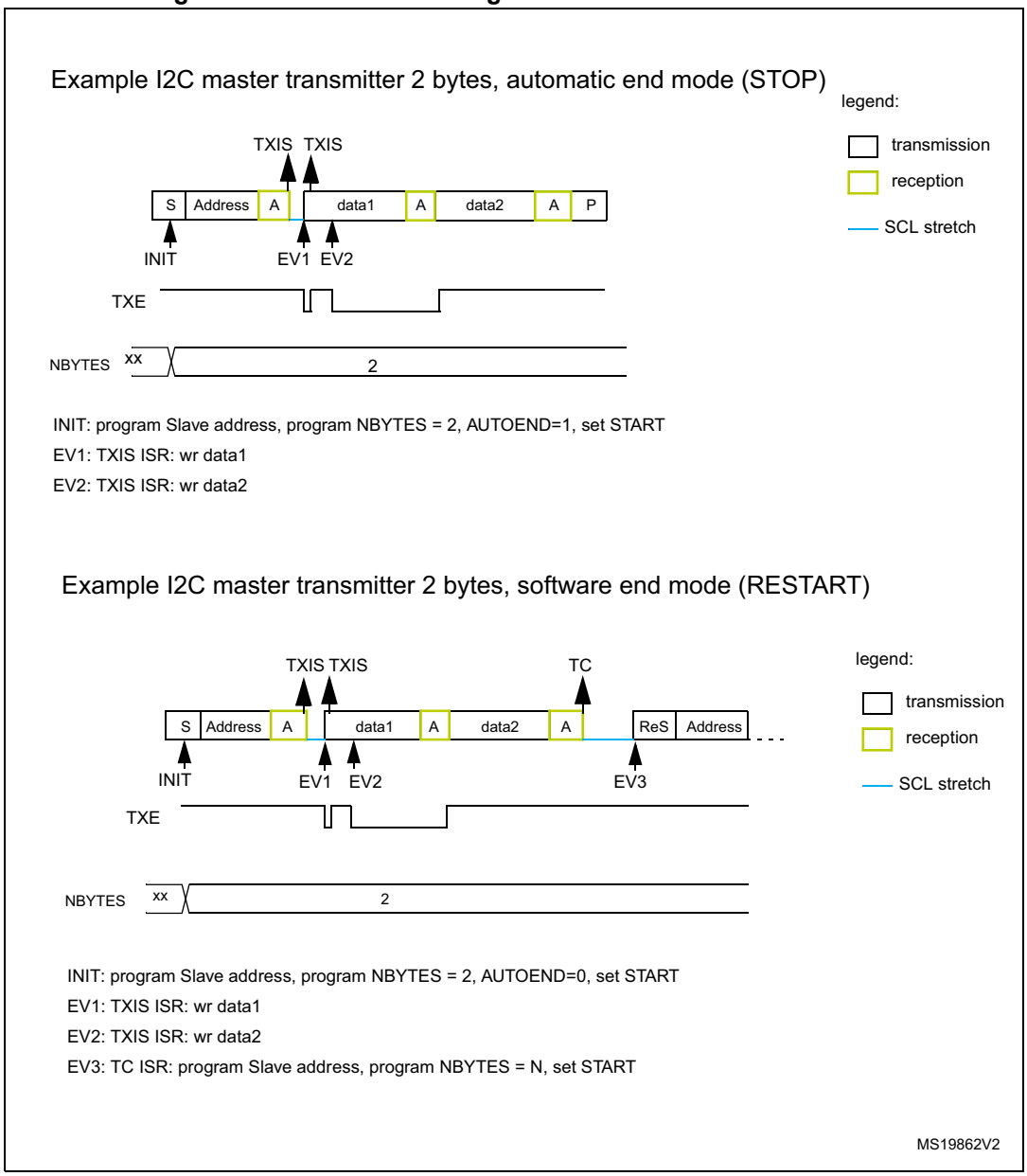


Figure 645. Transfer bus diagrams for I2C master transmitter



Master receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the eighth SCL pulse. An RXNE event generates an interrupt if the RXIE bit is set in the I2C_CR1 register. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to be received is greater than 255, reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when NBYTES[7:0] data have been transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written to a non-zero value.

- When RELOAD=0 and NBYTES[7:0] data have been transferred:
  - In automatic end mode (AUTOEND=1), a NACK and a STOP are automatically sent after the last received byte.
  - In software end mode (AUTOEND=0), a NACK is automatically sent after the last received byte, the TC flag is set and the SCL line is stretched low in order to allow software actions:

    A RESTART condition can be requested by setting the START bit in the I2C_CR2 register with the proper slave address configuration, and number of bytes to be transferred. Setting the START bit clears the TC flag and the START condition, followed by slave address, are sent on the bus.

    A STOP condition can be requested by setting the STOP bit in the I2C_CR2 register. Setting the STOP bit clears the TC flag and the STOP condition is sent on the bus.

Figure 646. Transfer sequence flow for I2C master receiver for N≤255 bytes

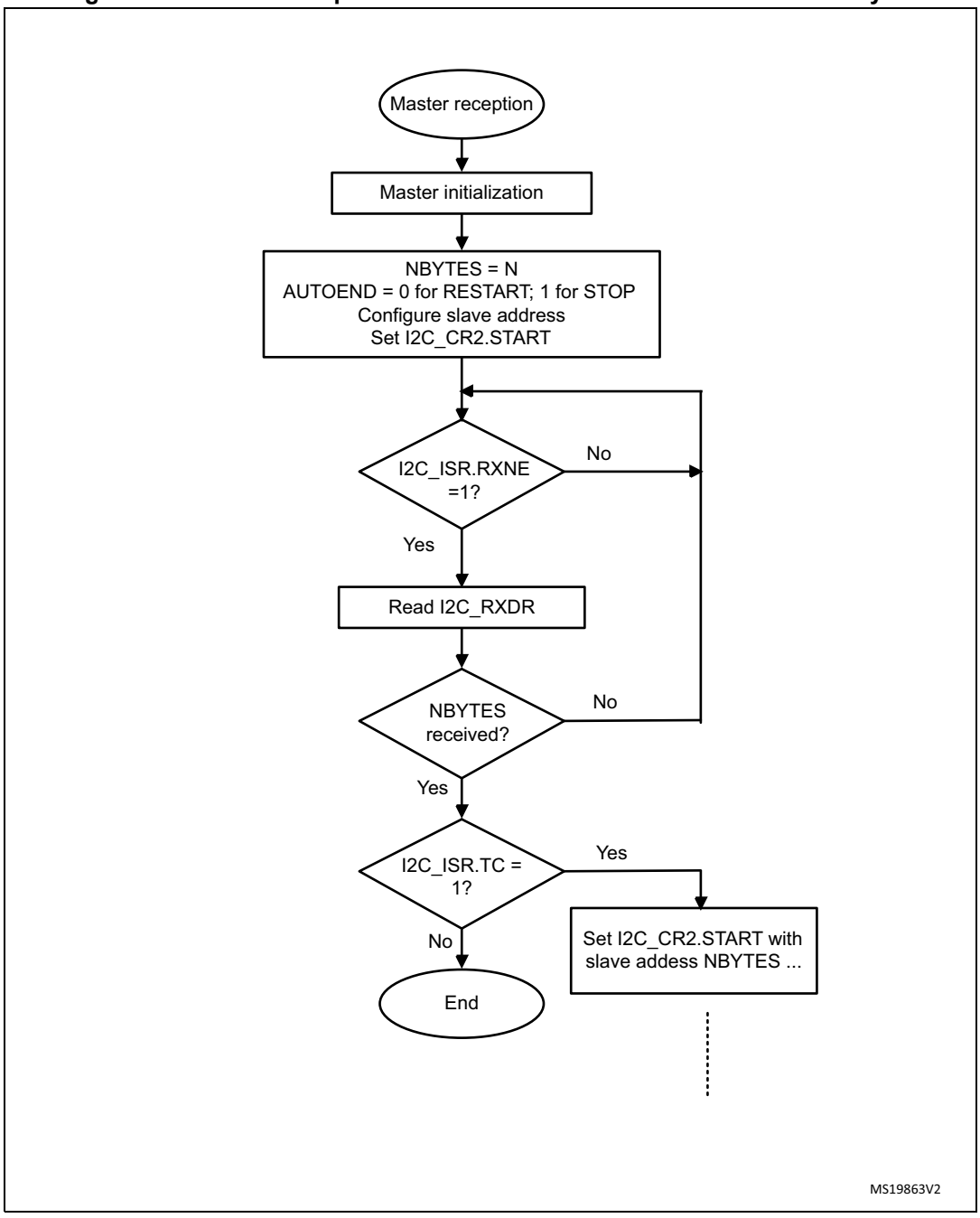


Figure 647. Transfer sequence flow for I2C master receiver for N >255 bytes

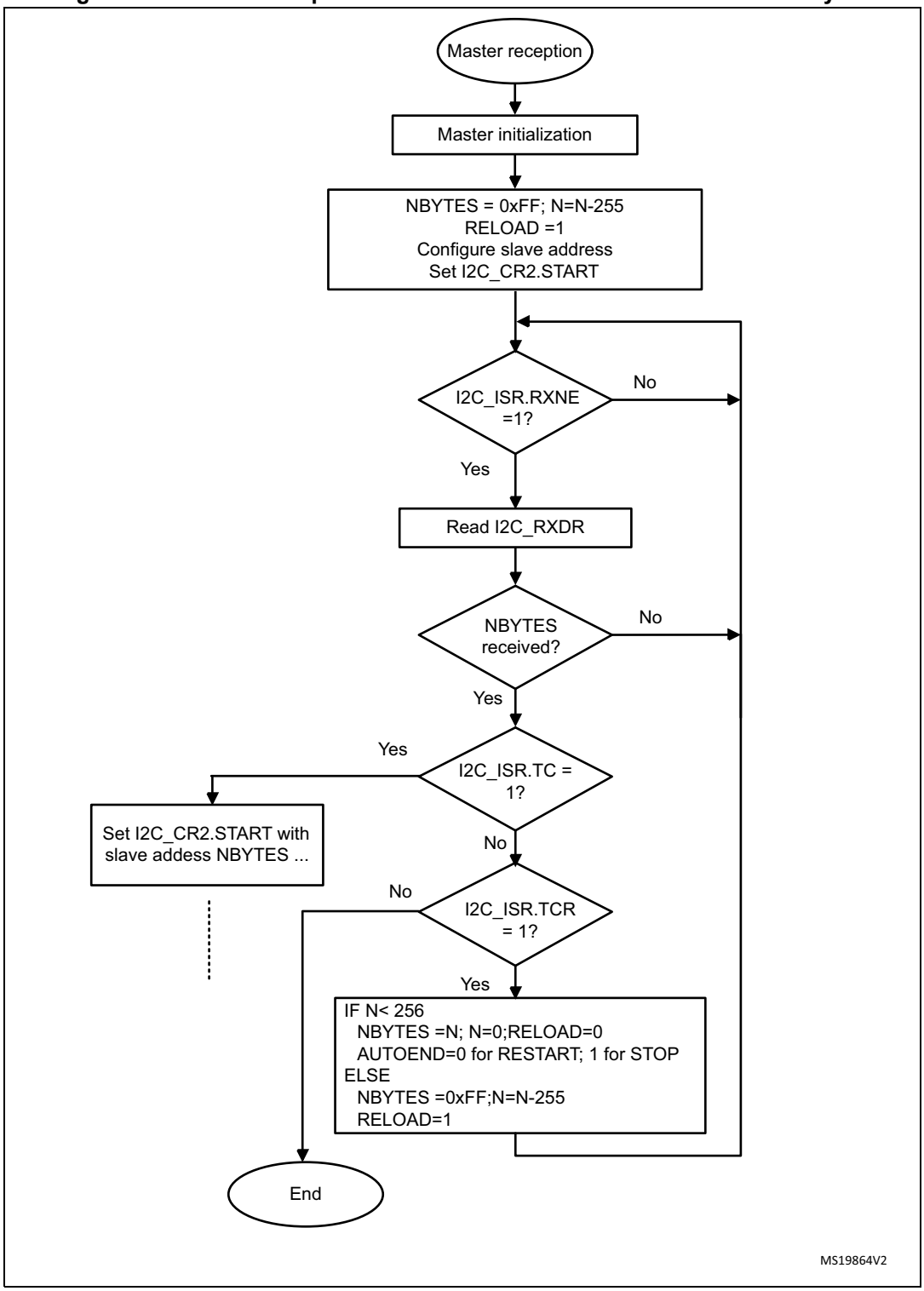
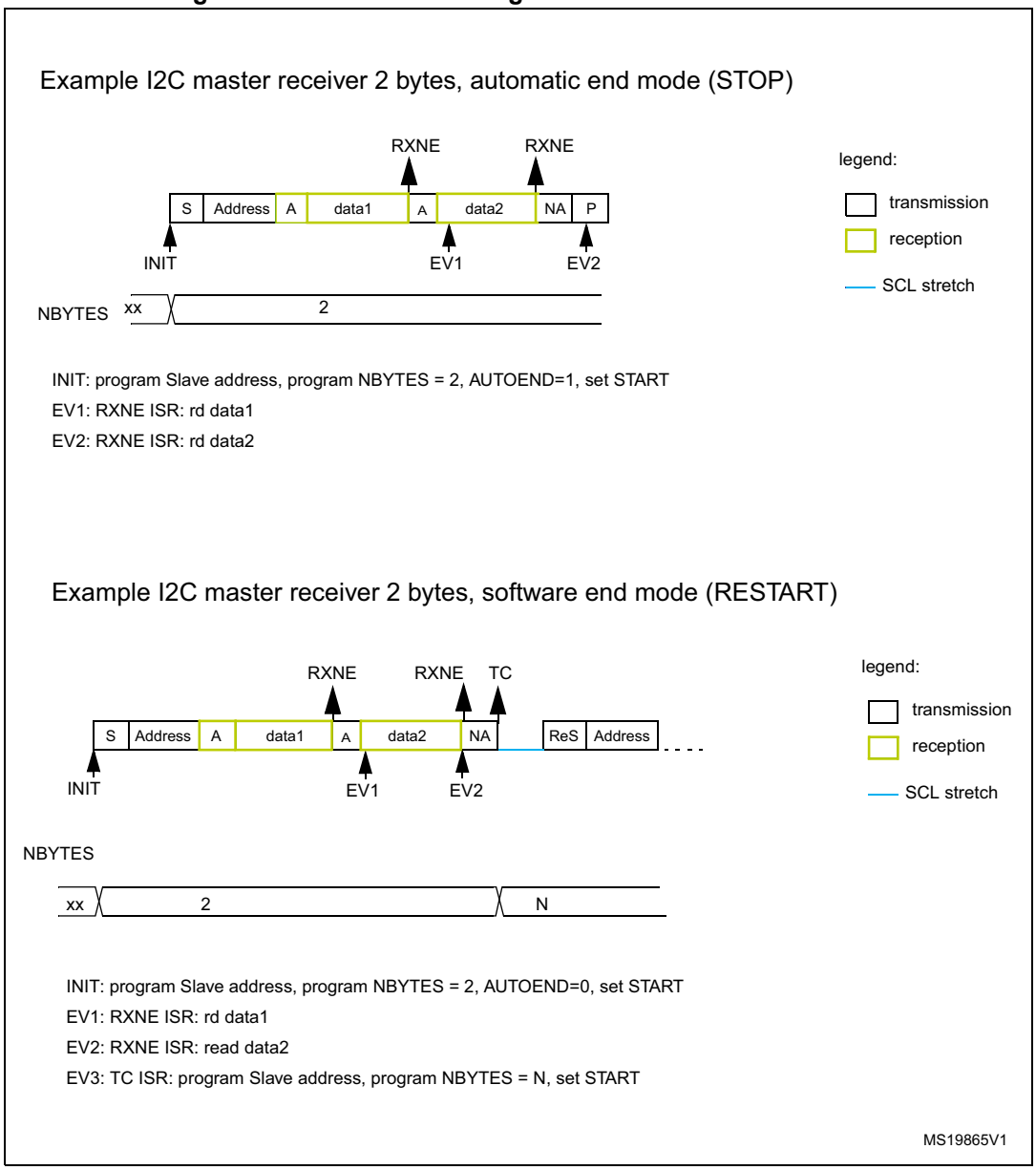


Figure 648. Transfer bus diagrams for I2C master receiver



56.4.10 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 545. 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 546. 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.

56.4.11 SMBus specific features

This section is relevant only when SMBus feature is supported. Refer to *Section 56.3: I2C implementation*.

Introduction

The system management bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on I^2C principles of operation. The SMBus provides a control bus for system and power management related tasks.

This peripheral is compatible with the SMBus specification (http://smbus.org).

The System Management Bus Specification refers to three types of devices.

- A slave is a device that receives or responds to a command.
- A master is a device that issues commands, generates the clocks and terminates the transfer.
- A host is a specialized master that provides the main interface to the system's CPU. A host must be a master-slave and must support the SMBus host notify protocol. Only one host is allowed in a system.

This peripheral can be configured as master or slave device, and also as a host.

Bus protocols

There are eleven possible command protocols for any given device. A device may use any or all of the eleven protocols to communicate. The protocols are Quick Command, Send Byte, Receive Byte, Write Byte, Write Word, Read Byte, Read Word, Process Call, Block Read, Block Write and Block Write-Block Read Process Call. These protocols should be implemented by the user software.

For more details of these protocols, refer to SMBus specification (http://smbus.org).

Address resolution protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. In order to provide a mechanism to isolate each device for the purpose of address assignment each device must implement a unique device identifier (UDID). This 128-bit number is implemented by software.

This peripheral supports the Address Resolution Protocol (ARP). The SMBus Device Default Address (0b1100 001) is enabled by setting SMBDEN bit in I2C_CR1 register. The ARP commands should be implemented by the user software.

Arbitration is also performed in slave mode for ARP support.

For more details of the SMBus address resolution protocol, refer to SMBus specification (http://smbus.org).

Received command and data acknowledge control

A SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in slave mode, the Slave Byte Control mode must be enabled by setting SBC bit in I2C_CR1 register. Refer to *Slave byte control mode* for more details.

Host notify protocol

This peripheral supports the host notify protocol by setting the SMBHEN bit in the I2C_CR1 register. In this case the host acknowledges the SMBus host address (0b0001 000).

When this protocol is used, the device acts as a master and the host as a slave.

SMBus alert

The SMBus ALERT optional signal is supported. A slave-only device can signal the host through the SMBALERT# pin that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the alert response address (0b0001 100). Only the device(s) which pulled SMBALERT# low acknowledges the alert response address.

When configured as a slave device(SMBHEN=0), the SMBA pin is pulled low by setting the ALERTEN bit in the I2C_CR1 register. The Alert Response Address is enabled at the same time.

When configured as a host (SMBHEN=1), the ALERT flag is set in the I2C_ISR register when a falling edge is detected on the SMBA pin and ALERTEN=1. An interrupt is generated if the ERRIE bit is set in the I2C_CR1 register. When ALERTEN=0, the ALERT line is considered high even if the external SMBA pin is low.

If the SMBus ALERT pin is not needed, the SMBA pin can be used as a standard GPIO if ALERTEN=0*.*

Packet error checking

A packet error checking mechanism has been introduced in the SMBus specification to improve reliability and communication robustness. The packet error checking is implemented by appending a packet error code (PEC) at the end of each message transfer. The PEC is calculated by using the $C(x) = x_8 + x^2 + x + 1$ CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The peripheral embeds a hardware PEC calculator and allows a not acknowledge to be sent automatically when the received byte does not match with the hardware calculated PEC.

Timeouts

This peripheral embeds hardware timers in order to be compliant with the 3 timeouts defined in SMBus specification.

Table 547. SMBus timeout specifications

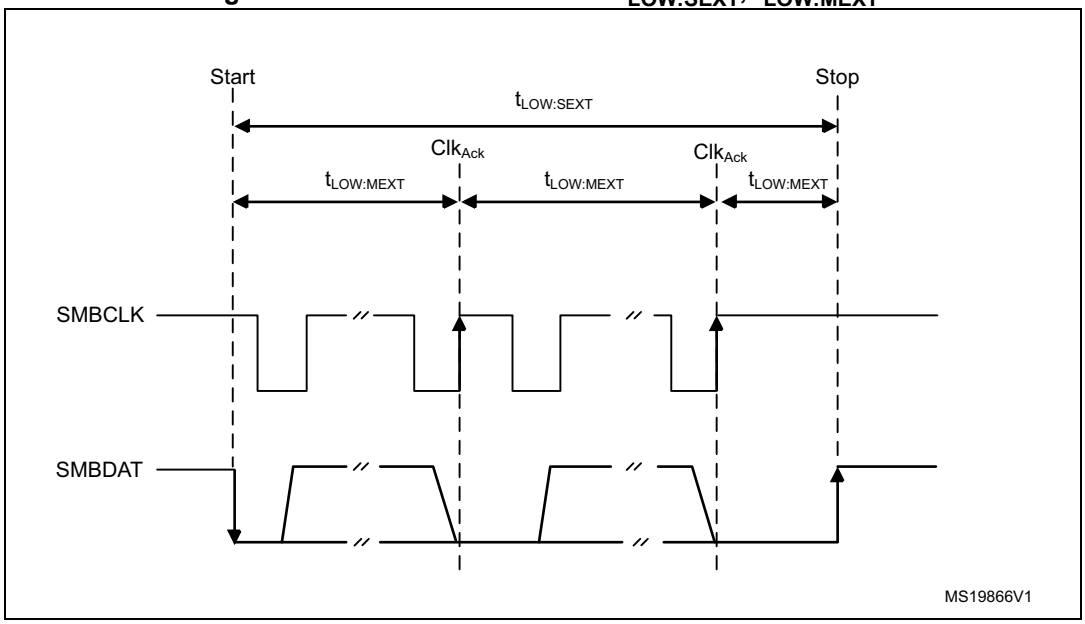
Symbol	Parameter	Limits		Unit
		Min	Max	
$t_{TIMEOUT}$	Detect clock low timeout	25	35	ms

Table 547. SMBus timeout specifications (continued)

Symbol	Parameter	Limits		Unit
		Min	Max	
$t_{LOW:SEXT}$(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 649. 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 542*)

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.

56.4.12 SMBus initialization

This section is relevant only when SMBus feature is supported. Refer to *Section 56.3: 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 2199* for more details.

Specific address (Slave mode)

The specific SMBus addresses must be enabled if needed. Refer to *Bus idle detection on page 2221* 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 548. 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 549*.

Caution: Changing the TIMEOUTA[11:0] bits and TIDLE bit configuration is not allowed when the TIMEOUTEN bit is set.

- $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ check

  Depending on if the peripheral is configured as a master or as a slave, The 12-bit TIMEOUTB timer must be configured in order to check $t_{LOW:SEXT}$ for a slave and $t_{LOW:MEXT}$ for a master. As the standard specifies only a maximum, the user can choose the same value for the both.

  Then the timer is enabled by setting the TEXTEN bit in the I2C_TIMEOUTR register.

  If the SMBus peripheral performs a cumulative SCL stretch for a time greater than (TIMEOUTB+1) x 2048 x t_{I2CCLK}, and in the timeout interval described in *Bus idle detection* section, the TIMEOUT flag is set in the I2C_ISR register.

  Refer to *Table 550*

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 551*.

Caution: Changing the TIMEOUTA and TIDLE configuration is not allowed when the TIMEOUTEN is set.

56.4.13 SMBus: I2C_TIMEOUTR register configuration examples

This section is relevant only when SMBus feature is supported. Refer to *Section 56.3: I2C implementation*.

- Configuring the maximum duration of $t_{TIMEOUT}$ to 25 ms:

Table 549. Examples of TIMEOUTA settings for various i2c_ker_ck **frequencies (max $t_{TIMEOUT}$ = 25 ms)**

f_{I2CCLK}	TIMEOUTA[11:0] bits	TIDLE bit	TIMEOUTEN bit	$t_{TIMEOUT}$
8 MHz	0x61	0	1	98 x 2048 x 125 ns = 25 ms
16 MHz	0xC3	0	1	196 x 2048 x 62.5 ns = 25 ms

- Configuring the maximum duration of $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ to 8 ms:

Table 550. Examples of TIMEOUTB settings for various i2c_ker_ck frequencies

f_{I2CCLK}	TIMEOUTB[11:0] bits	TEXTEN bit	$t_{LOW:EXT}$
8 MHz	0x1F	1	32 x 2048 x 125 ns = 8 ms
16 MHz	0x3F	1	64 x 2048 x 62.5 ns = 8 ms

- Configuring the maximum duration of t_{IDLE} to 50 µs

Table 551. Examples of TIMEOUTA settings for various i2c_ker_ck 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

56.4.14 SMBus slave mode

This section is relevant only when the SMBus feature is supported. Refer to *Section 56.3: I2C implementation*.

In addition to I2C slave transfer management (refer to *Section 56.4.8: I2C slave mode*) some additional software flows are provided to support the SMBus.

SMBus slave transmitter

When the IP is used in SMBus, SBC must be programmed to ‘1’ in order to allow the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case the total number of TXIS interrupts is NBYTES - 1 and the content of the I2C_PECR register is automatically transmitted if the master requests an extra byte after the NBYTES - 1 data transfer.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 650. Transfer sequence flow for SMBus slave transmitter N bytes + PEC

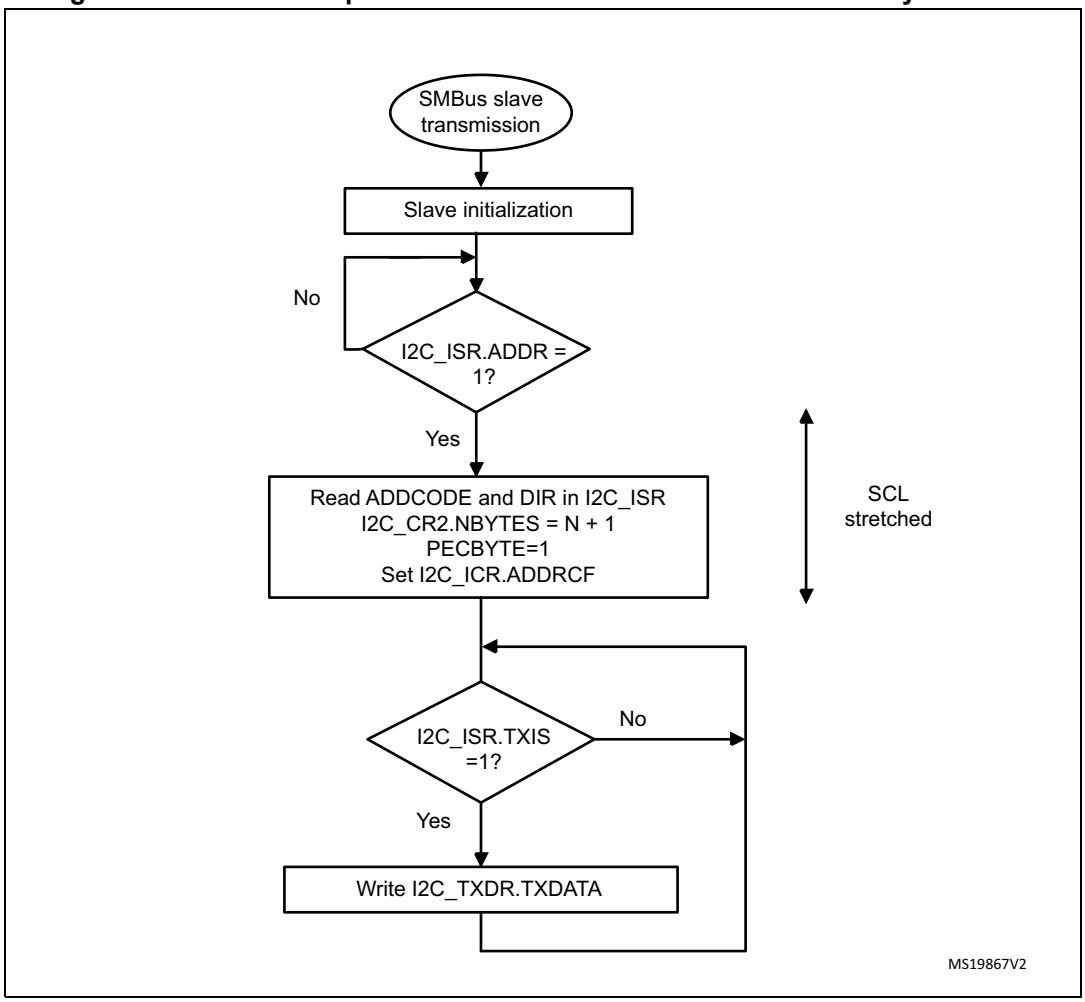
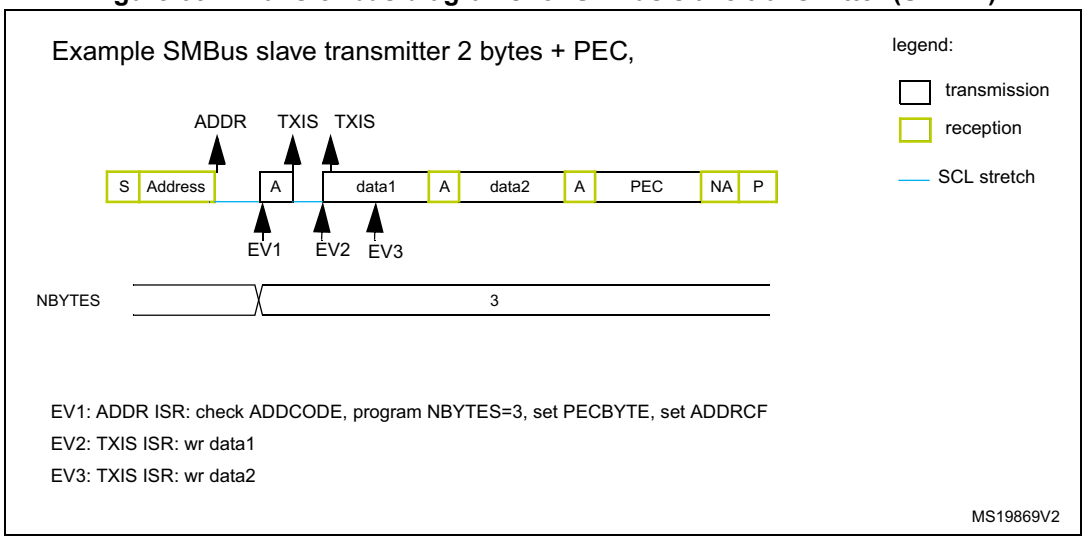


Figure 651. Transfer bus diagrams for SMBus slave transmitter (SBC=1)



SMBus Slave receiver

When the I2C is used in SMBus mode, SBC must be programmed to ‘1’ in order to allow the PEC checking at the end of the programmed number of data bytes. In order to allow the ACK control of each byte, the reload mode must be selected (RELOAD=1). Refer to *Slave byte control mode* for more details.

In order to check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after NBYTES - 1 data have been received, the next received byte is compared with the internal I2C_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C_RXDR register like any other data, and the RXNE flag is set.

In the case of a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

If no ACK software control is needed, the user can program PECBYTE=1 and, in the same write operation, program NBYTES with the number of bytes to be received in a continuous flow. After NBYTES - 1 are received, the next received byte is checked as being the PEC.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 652. Transfer sequence flow for SMBus slave receiver N Bytes + PEC

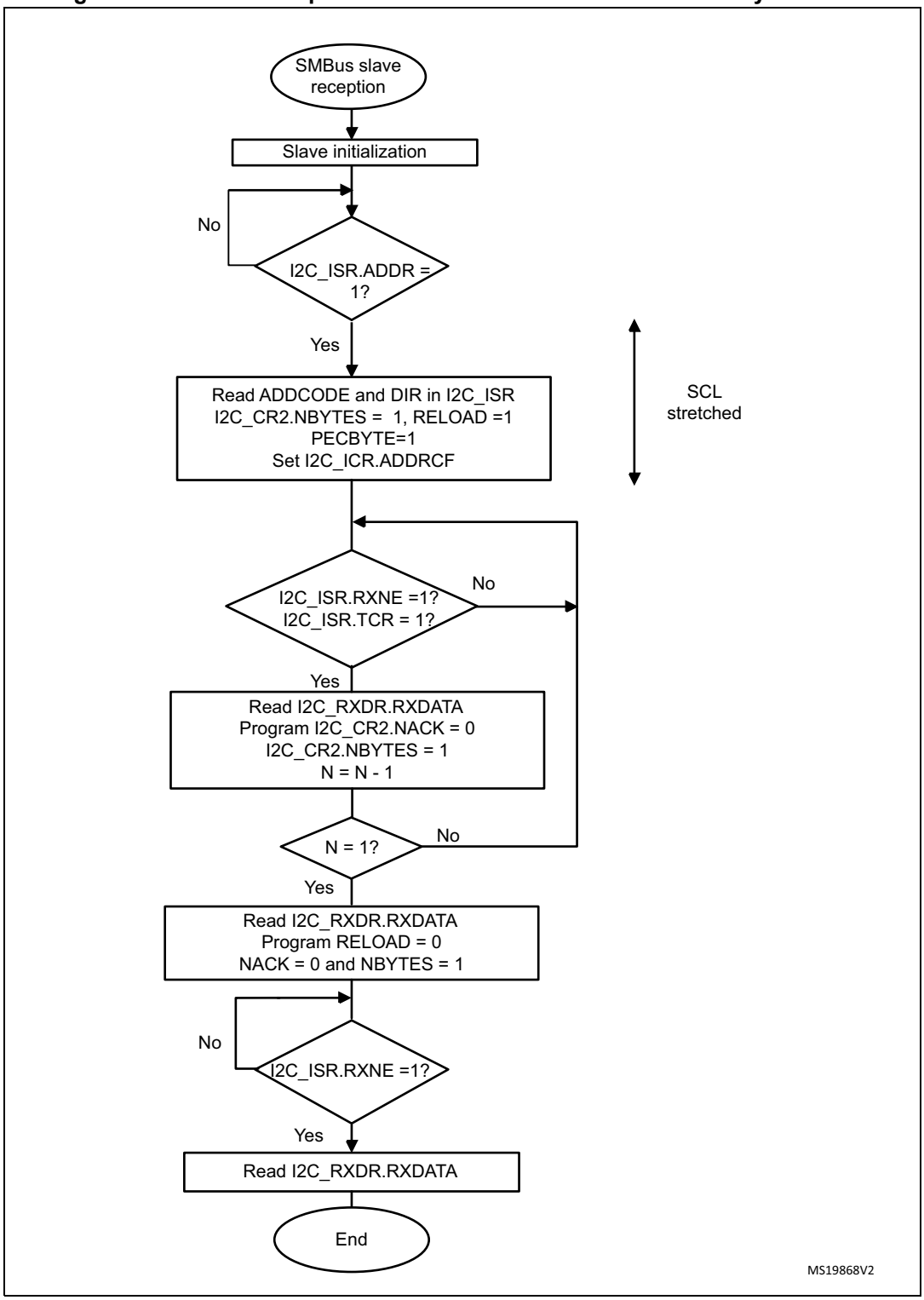
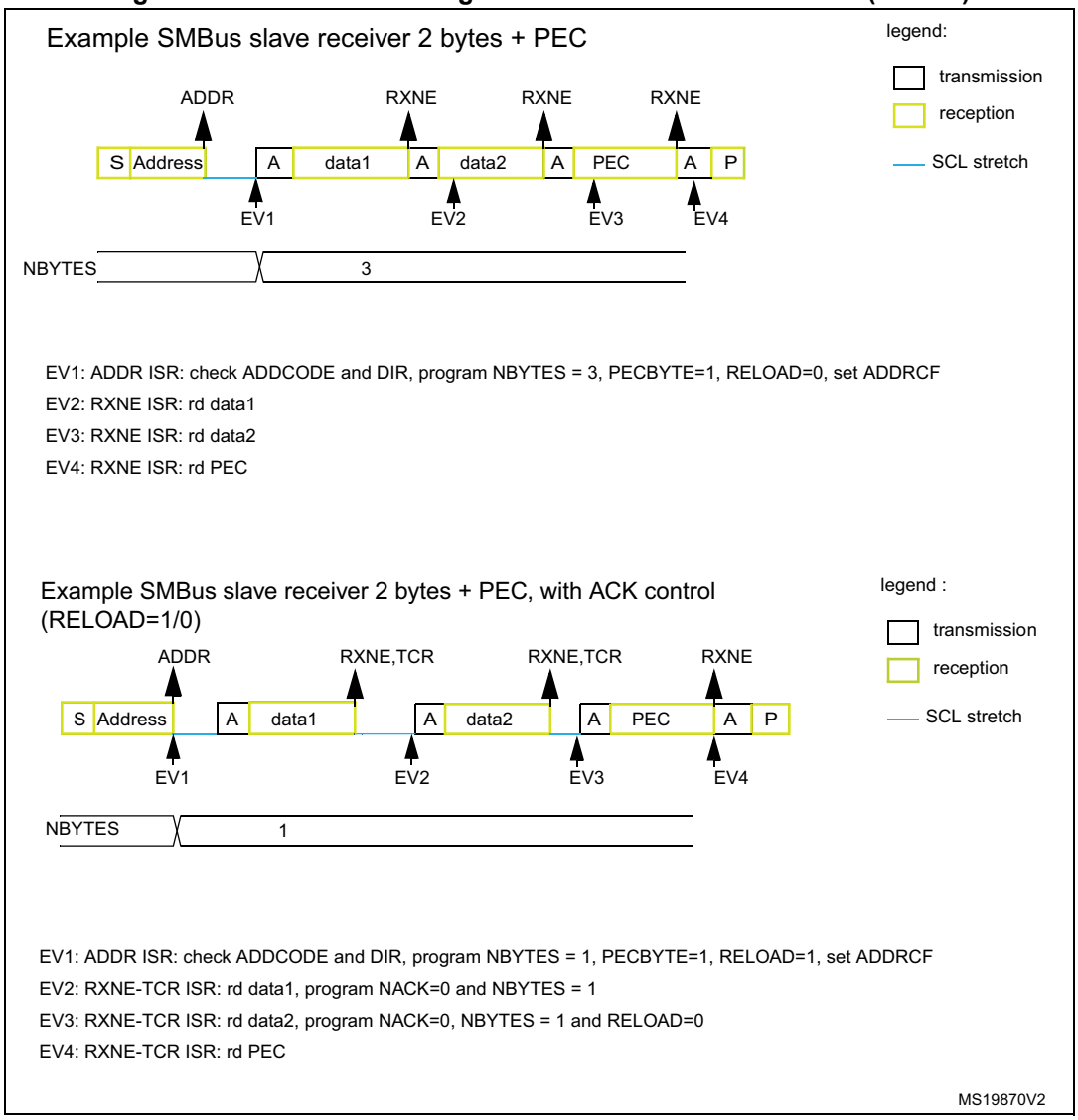


Figure 653. Bus transfer diagrams for SMBus slave receiver (SBC=1)



This section is relevant only when the SMBus feature is supported. Refer to *Section 56.3: I2C implementation*.

In addition to I2C master transfer management (refer to *Section 56.4.9: I2C master mode*), some additional software flows are provided to support the SMBus.

SMBus master transmitter

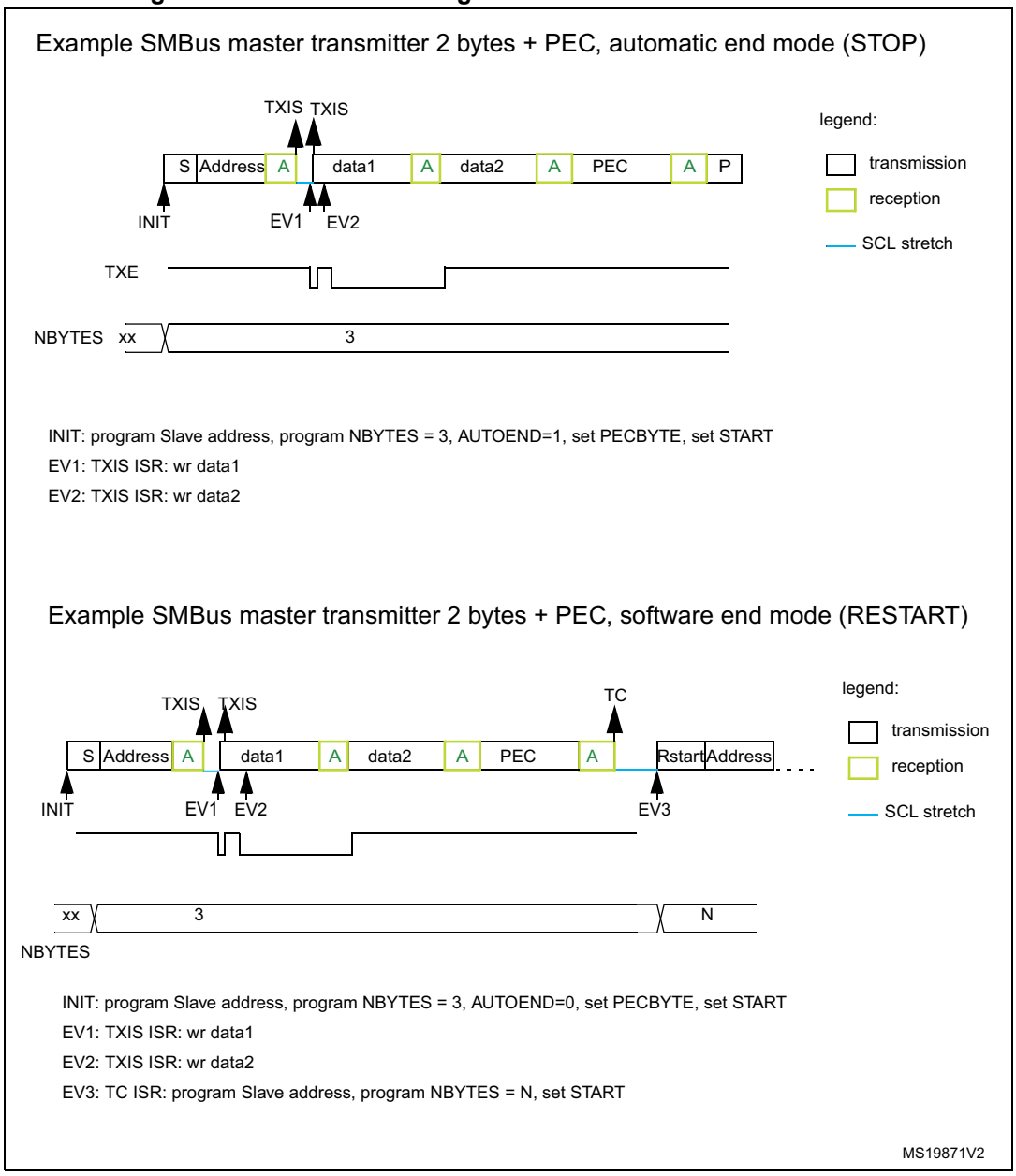
When the SMBus master wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be programmed in the NBYTES[7:0] field, before setting the START bit. In this case the total number of TXIS interrupts is NBYTES - 1. So if the PECBYTE bit is set when NBYTES = 0x1, the content of the I2C_PECR register is automatically transmitted.

If the SMBus master wants to send a STOP condition after the PEC, automatic end mode must be selected (AUTOEND = 1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus master wants to send a RESTART condition after the PEC, software mode must be selected (AUTOEND=0). In this case, once NBYTES - 1 have been transmitted, the I2C_PECR register content is transmitted and the TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 654. Bus transfer diagrams for SMBus master transmitter



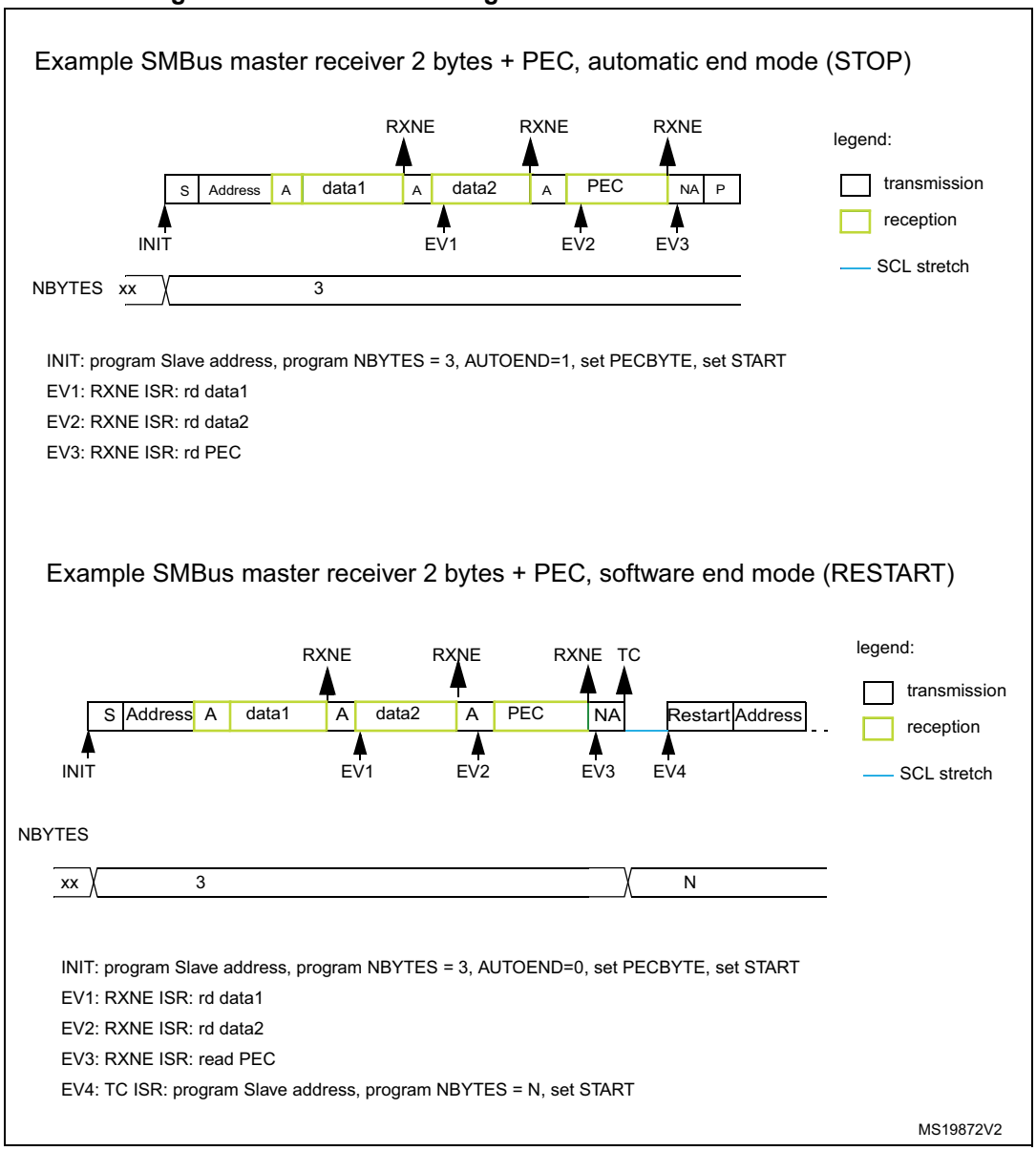
SMBus master receiver

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 655. Bus transfer diagrams for SMBus master receiver



56.4.15 Autonomous mode

The I2C peripheral can be functional in Stop mode thanks to its autonomous mode. The autonomous mode is supported in master mode and in slave mode with NOSTRETCH = 0. It is not supported when NOSTRETCH = 1. It is possible to use the autonomous mode in Run, Sleep or Stop mode.

The APB clock is requested by the peripheral each time the I2C status needs to be updated. Once the APB clock is received by the peripheral, either an interrupt or a DMA request is generated, depending on the I2C configuration.

In case an interrupt is generated, the device wakes up from Stop mode.

In case there is no interrupt, the device remains in Stop mode, but the kernel and AHB/APB clocks are available for the I2C, and all the autonomous peripherals enabled in the RCC. If

the DMA requests are enabled, the data are directly transferred to or from the SRAM thanks to the DMA, while the product remains in Stop mode.

Slave mode

In slave mode, the autonomous mode is enabled in Stop mode if WUPEN = 1 in the I2C_CR1. This mode is supported only when NOSTRETCH = 0.

The kernel clock is requested by the peripheral on Start detection during all the transfer until the Stop condition occurs, when the slave is addressed. If the slave is not addressed, the kernel clock request is released after the address phase.

To optimize the functionality in Slave mode by using only DMA transfers, it is possible to set the ADDRACLR and STOPFACLR bits in the I2C_CR1 in order to avoid to serve the Address match (ADDR) and Stop detection (STOPF) events.

Note: If the I2C clock is the system clock, or if WUPEN = 0, the internal oscillator is not switched on after a START is received in Slave mode.

Master mode

In master mode, a transfer can be automatically launched when an asynchronous trigger is detected in Run, Sleep or Stop mode. The trigger, selected by TRIGSEL in the I2C_AUTOCR register, generates a kernel clock request to allow the transfer, and launches a START condition and the I2C transfer as defined in the I2C_CR2 register. The kernel clock is requested until the Stop condition occurs.

In order to avoid to wake up the CPU too often, it is possible to replace some interrupts by DMA requests in order to make some control registers write actions. In master mode, Transfer Complete (TC) and Transfer Complete Reload (TCR) events can generate a DMA request when the corresponding TCDMAEN or TCRDMAEN are set in the I2C_AUTOCR register. Consequently the I2C_CR2 can be written thanks to a DMA transfer, in order to program a new transfer (in TC event) or to reload the number of bytes (in TCR event).

In case a trigger is enabled, but the application needs to take back control in order to start a different transfer, the following steps are required before writing in the I2C_CR2 register to launch the new transfer:

1. Wait for BUSY = 0 in the I2C_ISR
2. Disable the TRIGEN bit in I2C_AUTOCR register
3. Wait for a delay greater than the t_{BUF} timing (Bus free time between a STOP and a START condition).
4. Wait for BUSY = 0 in the I2C_ISR

Caution: When the product is in Stop mode, the I2C receives its kernel clock only when it is implicated in the transfer. Consequently some features are not reliable in Stop mode, such as the timeouts and bus idle detection.

Caution: The digital filter is not compatible with the functionality in Stop mode. If the DNF bit is not equal to 0, setting the WUPEN bit has no effect.

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).

56.4.16 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 56.3: 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 56.3: 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 56.3: 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.

56.4.17 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) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

- In master mode: the initialization, the slave address, direction, number of bytes and START bit are programmed by software (the transmitted slave address cannot be transferred with DMA). When all data are transferred using DMA, the DMA must be

initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to *Master transmitter*.

- In slave mode:
  - With NOSTRETCH = 0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
  - With NOSTRETCH = 1, the DMA must be initialized before the address match event.
- For instances supporting SMBus: the PEC transfer is managed with NBYTES counter. Refer to *SMBus slave transmitter* and *SMBus master transmitter*.

Note: If DMA is used for transmission, the TXIE bit does not need to be enabled.

Reception using DMA

DMA (direct memory access) can be enabled for reception by setting the RXDMAEN bit in the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured using the DMA peripheral (refer to) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

- In Master mode, the initialization, the slave address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.
- In Slave mode with NOSTRETCH = 0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.
- If SMBus is supported (see *Section 56.3: I2C implementation*): the PEC transfer is managed with the NBYTES counter. Refer to *SMBus Slave receiver* and *SMBus master receiver*.

Note: If DMA is used for reception, the RXIE bit does not need to be enabled.

Master event control using DMA

In master mode, the transfer can be automatically managed while the product is in Run, Sleep or Stop mode thanks to TC and TCR DMA requests.

- If TCDMAEN is set in the I2C_AUTOCR register, the I2C_EVC (I2C Control Event) DMA request is generated when TC is set in the I2C_ISR register. The DMA must be programmed to write the next command in the I2C_CR2 register. If both STOP and START bit are set in the new I2C_CR2 command, a STOP condition followed by a START condition is sent, followed by the address, direction and number of bytes defined in I2C_CR2. If only START bit is set in the new I2C_CR2 command, a ReSTART condition is sent, followed by the address, direction and number of bytes defined in I2C_CR2.
- If TCRDMAEN is set in the I2C_AUTOCR register, the I2C_EVC (I2C Control Event) DMA request is generated when TCR is set in the I2C_ISR register. The DMA must be programmed to write the remaining number of bytes to be transferred in the I2C_CR2 register.

56.4.18 Debug mode

When the microcontroller enters debug mode (core halted), the SMBus timeout either continues to work normally or stops, depending on the DBG_I2Cx_ configuration bits in the DBG module.

56.5 I2C low-power modes

Table 552. Effect of low-power modes on the I2C

Mode	Description
Sleep	No effect. I2C interrupts cause the device to exit the Sleep mode.
Stop(1)	The I2C registers content is kept. If the autonomous mode is enabled and I2C is clocked by an internal oscillator available in Stop mode: transfers in master and in slave modes are functional. DMA requests are functional, and the interrupts cause 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 I2C implementation table 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.

56.6 I2C interrupts

The table below gives the list of I2C interrupt requests.

Table 553. I2C Interrupt requests

Interrupt acronym		Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit the Sleep mode	Exit the Stop mode	Exit the Stop3, Standby, Shutdown mode
I2C	I2C_EV	Receive buffer not empty	RXNE	RXIE	Read I2C_RXDR register	Yes	Yes(1)	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			
		NACK reception	NACKF	NACKIE	Write NACKCF=1			
		Address matched	ADDR	ADDRIE	Write ADDRCF=1			
		NACK reception	NACKF	NACKIE	Write NACKCF=1			
	I2C_ER	Bus error	BERR	ERRIE	Write BERRCF=1	Yes	Yes(1)	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. Refer to I2C implementation table for information about wakeup from Stop mode support per instance.

56.7 I2C registers

Refer to *Section 1.2 on page 104* for a list of abbreviations used in register descriptions.

The peripheral registers are accessed by words (32-bit).

56.7.1 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 i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STOPFACLR	ADDRACLR	Res.	Res.	Res.	Res.	Res.	FMP	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC
rw	rw						rw	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

Bit 31 **STOPFACLR**: STOP detection flag (STOPF) automatic clear

0: STOPF flag is set by hardware and cleared by software by setting STOPCF bit.

1: STOPF flag remains cleared by hardware. This mode can be used in NOSTRETCH slave mode, to avoid the overrun error if the STOPF flag is not cleared before next data transmission. This allows a slave data management by DMA only, without any interrupt from peripheral.

Bit 30 **ADDRACLR**: Address match flag (ADDR) automatic clear

0: ADDR flag is set by hardware and cleared by software by setting ADDRCF bit.

1: ADDR flag remains cleared by hardware. This mode can be used in slave mode, to avoid the ADDR clock stretching if the I2C enables only one slave address. This allows a slave data management by DMA only, without any interrupt from peripheral.

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

Bit 24 **FMP**: Fast-mode Plus 20 mA drive enable

0: 20 mA I/O drive disabled

1: 20 mA I/O drive enabled

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 56.3: 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 56.3: 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 56.3: 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 56.3: 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 56.3: 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.

56.7.2 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 i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PEC BYTE	AUTO END	RE LOAD	NBYTES[7:0]							
					rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD10R	ADD10	RD_ WRN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **PECBYTE**: Packet error checking byte
This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when PE = 0.
0: No PEC transfer.
1: PEC transmission/reception is requested
Note: Writing '0' to this bit has no effect.
This bit has no effect when RELOAD is set.
This bit has no effect is slave mode when SBC=0.
If the SMBus feature is not supported, this bit is reserved and forced by hardware to '0'. Refer to Section 56.3: I2C implementation.

Bit 25 **AUTOEND**: Automatic end mode (master mode)
This bit is set and cleared by software.
0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.
1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.
Note: This bit has no effect in slave mode or when the RELOAD bit is set.

Bit 24 **RELOAD**: NBYTES reload mode
This bit is set and cleared by software.
0: The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).
1: The transfer is not completed after the NBYTES data transfer (NBYTES is reloaded). TCR flag is set when NBYTES data are transferred, stretching SCL low.

Bits 23:16 **NBYTES[7:0]**: Number of bytes

The number of bytes to be transmitted/received is programmed there. This field is don't care in slave mode with SBC=0.

Note: Changing these bits when the START bit is set is not allowed.

Bit 15 **NACK**: NACK generation (slave mode)

The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE = 0.

0: an ACK is sent after current received byte.

1: a NACK is sent after current received byte.

Note: Writing '0' to this bit has no effect.

This bit is used in slave mode only: in master receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.

When an overrun occurs in slave receiver NOSTRETCH mode, a NACK is automatically generated whatever the NACK bit value.

When hardware PEC checking is enabled (PECBYTE=1), the PEC acknowledge value does not depend on the NACK value.

Bit 14 **STOP**: Stop generation (master mode)

The bit is set by software, cleared by hardware when a STOP condition is detected, or when PE = 0.

In Master Mode:

0: No Stop generation.

1: Stop generation after current byte transfer.

Note: Writing '0' to this bit has no effect.

Bit 13 **START**: Start generation

This bit is set by software, and cleared by hardware after the Start followed by the address sequence is sent, by an arbitration loss, by an address matched in slave mode, by a timeout error detection, or when PE = 0.

0: No Start generation.

1: Restart/Start generation:

If the I2C is already in master mode with AUTOEND = 0, setting this bit generates a Repeated Start condition when RELOAD=0, after the end of the NBYTES transfer.

Otherwise setting this bit generates a START condition once the bus is free.

Note: Writing '0' to this bit has no effect.

The START bit can be set even if the bus is BUSY or I2C is in slave mode.

This bit has no effect when RELOAD is set.

Bit 12 **HEAD10R**: 10-bit address header only read direction (master receiver mode)

0: The master sends the complete 10 bit slave address read sequence: Start + 2 bytes 10bit address in write direction + Restart + 1st 7 bits of the 10 bit address in read direction.

1: The master only sends the 1st 7 bits of the 10 bit address, followed by Read direction.

Note: Changing this bit when the START bit is set is not allowed.

Bit 11 **ADD10**: 10-bit addressing mode (master mode)

0: The master operates in 7-bit addressing mode,

1: The master operates in 10-bit addressing mode

Note: Changing this bit when the START bit is set is not allowed.

Bit 10 **RD_WRN**: Transfer direction (master mode)
0: Master requests a write transfer.
1: Master requests a read transfer.
Note: Changing this bit when the START bit is set is not allowed.

Bits 9:0 **SADD[9:0]**: Slave address (master mode)
In 7-bit addressing mode (ADD10 = 0):
SADD[7:1] should be written with the 7-bit slave address to be sent. The bits SADD[9], SADD[8] and SADD[0] are don't care.
In 10-bit addressing mode (ADD10 = 1):
SADD[9:0] should be written with the 10-bit slave address to be sent.
Note: Changing these bits when the START bit is set is not allowed.

56.7.3 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 i2c_pclk + 6 x i2c_ker_ck.

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

56.7.4 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 i2c_pclk + 6 x i2c_ker_ck.

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

56.7.5 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 i2c_ker_ck in order to generate the clock period t_{PRESC} used for data setup and hold counters (refer to *I2C timings on page 2191*) and for SCL high and low level counters (refer to *I2C master initialization on page 2206*).

t_{PRESC} = (PRESC+1) x t_{I2CCLK}

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **SCLDEL[3:0]**: Data setup time

This field is used to generate a delay t_{SCLDEL} between SDA edge and SCL rising edge. In master mode and in slave mode with NOSTRETCH = 0, the SCL line is stretched low during t_{SCLDEL}.

t_{SCLDEL} = (SCLDEL+1) x t_{PRESC}

Note: t_{SCLDEL} is used to generate $t_{SU:DAT}$ timing.

Bits 19:16 **SDADEL[3:0]**: Data hold time

This field is used to generate the delay t_{SDADEL} between SCL falling edge and SDA edge. In master mode and in slave mode with NOSTRETCH = 0, the SCL line is stretched low during t_{SDADEL}.

t_{SDADEL}= SDADEL x t_{PRESC}

Note: SDADEL is used to generate $t_{HD:DAT}$ timing.

Bits 15:8 **SCLH[7:0]**: SCL high period (master mode)

This field is used to generate the SCL high period in master mode.

t_{SCLH} = (SCLH+1) x t_{PRESC}

Note: SCLH is also used to generate $t_{SU:STO}$ and $t_{HD:STA}$ timing.

Bits 7:0 **SCLL[7:0]**: SCL low period (master mode)

This field is used to generate the SCL low period in master mode.

t_{SCLL} = (SCLL+1) x t_{PRESC}

Note: SCLL is also used to generate t_{BUF} and $t_{SU:STA}$ timings.

Note: This register must be configured when the I2C is disabled (PE = 0).

Note: The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.

56.7.6 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 i2c_pclk + 6 x i2c_ker_ck.

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 56.3: I2C implementation.

56.7.7 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 56.3: 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 56.3: 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 56.3: 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.

56.7.8 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 56.3: 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 56.3: 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 56.3: 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.

56.7.9 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 56.3: I2C implementation.

56.7.10 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

56.7.11 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.

56.7.12 I2C Autonomous mode control register (I2C_AUTOCR)

Address offset: 0x2C

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.	TRIGEN	TRIGPOL	TRIGSEL[3:0]			
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCRDMAEN	TCDMAEN	Res.	Res.	Res.	Res.	Res.	Res.
								rw	rw						

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **TRIGEN:** Trigger enable

0: Trigger disabled

1: Trigger enabled

When a trigger is detected, a START condition is sent and the transfer is launched as defined in I2C_CR2.

Bit 20 **TRIGPOL:** Trigger polarity

0: Trigger active on rising edge

1: Trigger active on falling edge

Note: This bit can be written only when PE = 0

Bits 19:16 **TRIGSEL[3:0]:** Trigger selection (refer to *Section 56.4.2: I2C pins and internal signals* I2C interconnections tables).

0000: i2c_trg0 selected

0001: i2c_trg1 selected

...

1111: i2c_trg15 selected

Note: This bit can be written only when PE = 0

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

Bit 7 **TCRDMAEN**: DMA request enable on Transfer Complete Reload event

0: DMA request not generated on Transfer Complete Reload event

1: DMA request generated on Transfer Complete Reload event

Bit 6 **TCDMAEN**: DMA request enable on Transfer Complete event

0: DMA request not generated on Transfer Complete event

1: DMA request generated on Transfer Complete event

Bits 5:0 Reserved, must be kept at reset value.

56.7.13 I2C register map

The table below provides the I2C register map and reset values.

Table 554. 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	STOPFACLR	ADDRACLR	Res.	Res.	Res.	Res.	Res.	FMP	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	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 554. 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
0x2C	I2C_AUTOCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGEN	TRIGPOL	TRIGSEL [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCRDMAEN	TCDMAEN	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0	0	0	0	0	0									0	0						

Refer to *Section 2.3 on page 115* for the register boundary addresses.

57 Universal synchronous/asynchronous receiver transmitter (USART/UART)

This section describes the universal synchronous asynchronous receiver transmitter (USART/UART).

57.1 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.

57.2 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
- Autonomous functionality in Stop mode

57.3 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

57.4 USART implementation

The tables below describe USART implementation on STM32U575/585 devices. It also includes LPUART for comparison.

Table 555. STM32U575/585 features

USART modes/features	STM32U575/585
USART1	FULL
USART2	FULL
USART3	FULL
UART4	BASIC
UART5	BASIC
LPUART1	LP

Table 556. USART/LPUART features

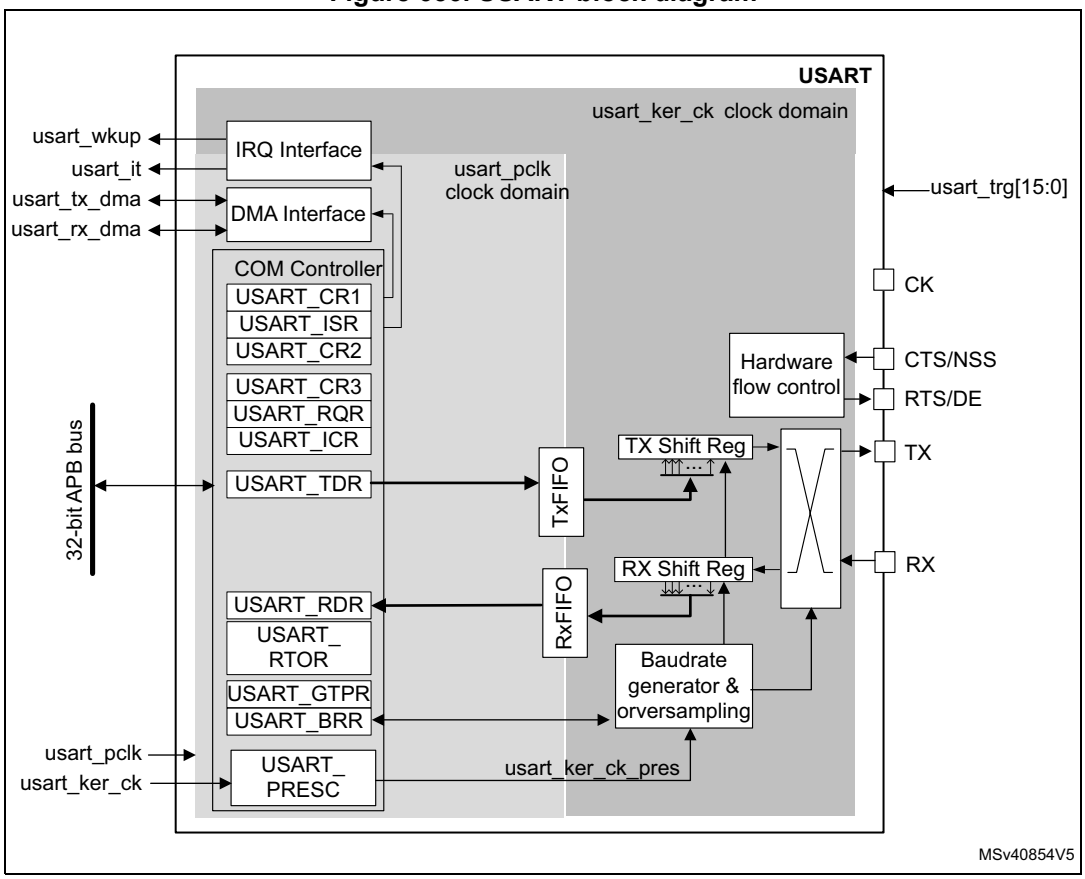
USART modes/features(1)	Full feature set	Basic feature set	Low-power feature set
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (Master/Slave)	X	-	-
Smartcard mode	X	-	-
Single-wire Half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	-
LIN mode	X	X	-
Dual clock domain	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		
Wakeup from low-power mode	X(2)	X(2)	X(3)
Autonomous mode	X	X	X

1. X = supported.
2. Wakeup supported from Stop 0 and Stop 1 modes.
3. Wakeup supported from Stop 0, Stop 1 and Stop 2 modes.

57.5 USART functional description

57.5.1 USART block diagram

Figure 656. USART block diagram



57.5.2 USART pins and internal signals

Description USART input/output pins

- 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 **DE** (Driver Enable) pin is required in RS485 Hardware control mode. This signal activates the Transmission mode of the external transceiver.

- Synchronous Master/Slave mode and Smartcard mode

  The following pins are 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.

Refer to *Table 557* and *Table 558* for the list of USART input/output pins and internal signals.

Table 557. USART input/output pins

Pin name	Signal type	Description
USART_RX	Input	Serial data receive input.
USART_TX	Output	Transmit data output.
USART_CTS	Input	Clear to send
USART_RTS	Output	Request to send
USART_DE(1)	Output	Driver enable
USART_CK	Output	Clock output in Synchronous master and Smartcard modes.
USART_NSS(2)	Input	Slave select input in Synchronous slave mode.

1. USART_DE and USART_RTS share the same pin.
2. USART_NSS and USART_CTS share the same pin.

Description of USART input/output signals

Table 558. USART internal input/output signals

Pin name	Signal type	Description
usart_pclk	Input	APB clock
usart_ker_ck	Input	USART kernel clock
usart_wkup	Output	USART provides a wakeup interrupt
usart_it	Output	USART global interrupt
usart_tx_dma	Input/output	USART transmit DMA request
usart_rx_dma	Input/output	USART receive DMA request
usart_trg[15:0]	Input	USART triggers.

Description of USART interconnections

Table 559. USART interconnection (USART1/2/3 and UART4/5)

Signal name	Source
usart_trg0	gpdma1_ch0_tc
usart_trg1	gpdma1_ch1_tc
usart_trg2	gpdma1_ch2_tc
usart_trg3	gpdma1_ch3_tc
usart_trg4	exti6
usart_trg5	exti9
usart_trg6	lptim1_ch1
usart_trg7	lptim2_ch1
usart_trg8	comp1_out
usart_trg9	comp2_out
usart_trg10	rtc_alra_trg
usart_trg11	rtc_wut_trg
usart_trg12	-
usart_trg13	-
usart_trg14	-
usart_trg15	-

57.5.3 USART clocks

The simplified block diagram given in *Figure 656* 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 not supported, the **usart_ker_ck** clock is the same as the **usart_pclk** clock.

There is no constraint between **usart_pclk** and **usart_ker_ck**: **usart_ker_ck** can be faster or slower than **usart_pclk**. The only limitation is the software ability to manage the communication fast enough.

When the USART operates in SPI slave mode, it handles data flow using the serial interface clock derived from the external CK signal provided by the external master SPI device. The **usart_ker_ck** clock must be at least 3 times faster than the clock on the CK input.

57.5.4 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 657*):

- 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.

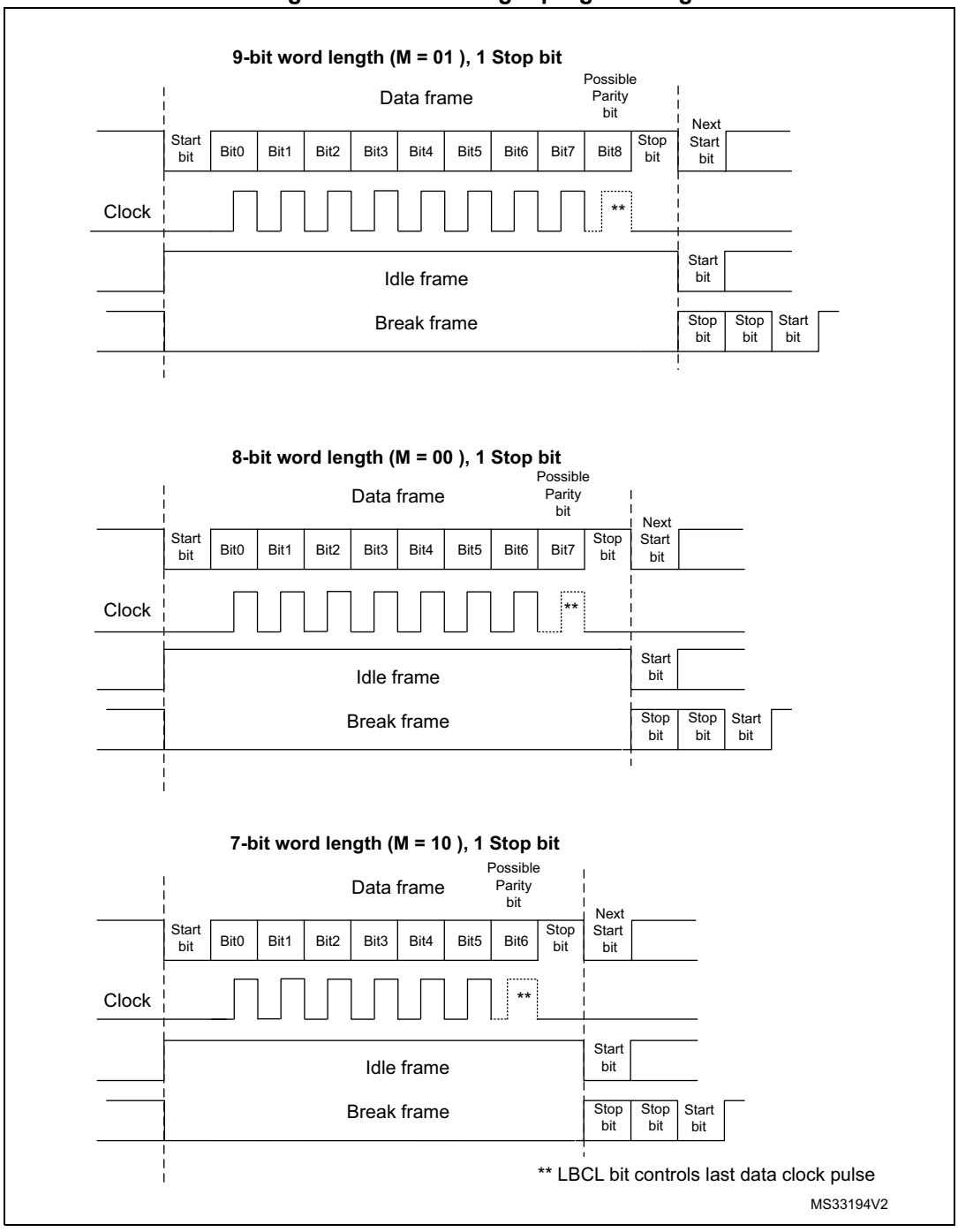
An ***Idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

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 657. Word length programming



57.5.5 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 Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bits fields.

  In this case, the RXFT flag is set in the USART_ISR register. This means that RXFTCFG data have been received: 1 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 does not set the overrun flag.
- The Tx interrupt is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bits fields.

57.5.6 USART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin while the corresponding clock pulses are output on the CK pin.

Character transmission

During an USART transmission, data shifts out the least significant bit first (default configuration) on the TX pin. In this mode, the USART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register.

When FIFO mode is enabled, the data written to the transmit data register (USART_TDR) are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be configured to 0.5, 1, 1.5 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the USART_TDR.

The TE bit should not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters get frozen. The current data being transmitted are then lost.

An idle frame is sent when the TE bit is enabled.

Configurable stop bits

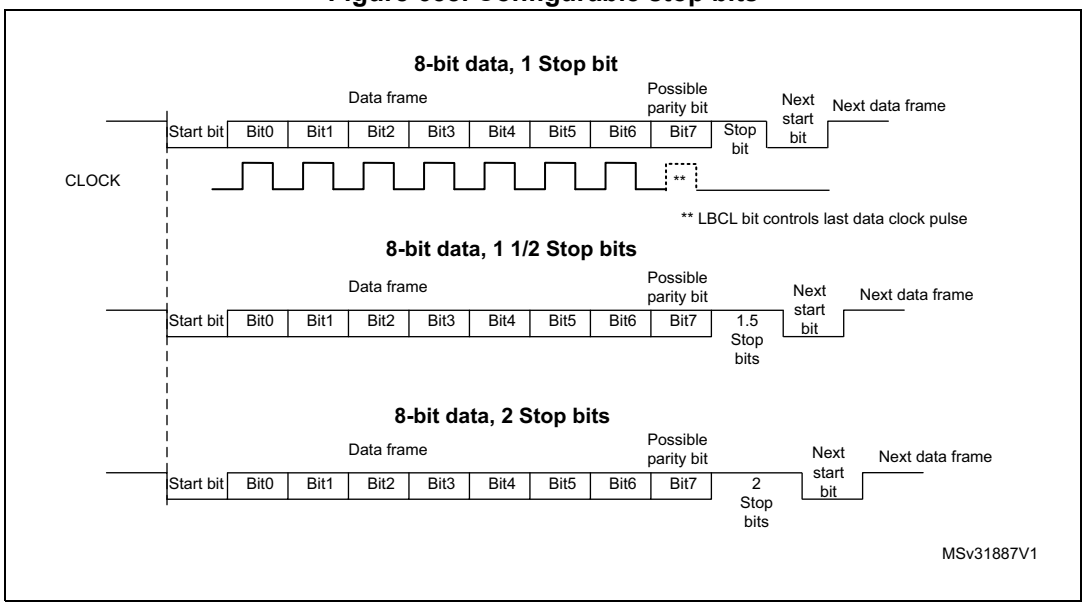
The number of stop bits to be transmitted with every character can be programmed in USART_CR2, bits 13,12.

- ***1 stop bit***: This is the default value of number of stop bits.
- ***2 stop bits***: This is supported by normal USART, Single-wire and Modem modes.
- ***1.5 stop bits***: To be used in Smartcard mode.

An idle frame transmission includes the stop bits.

A break transmission 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 (see *Figure 658*). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 658. 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 57.5.20: Continuous communication using USART and DMA*.
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.

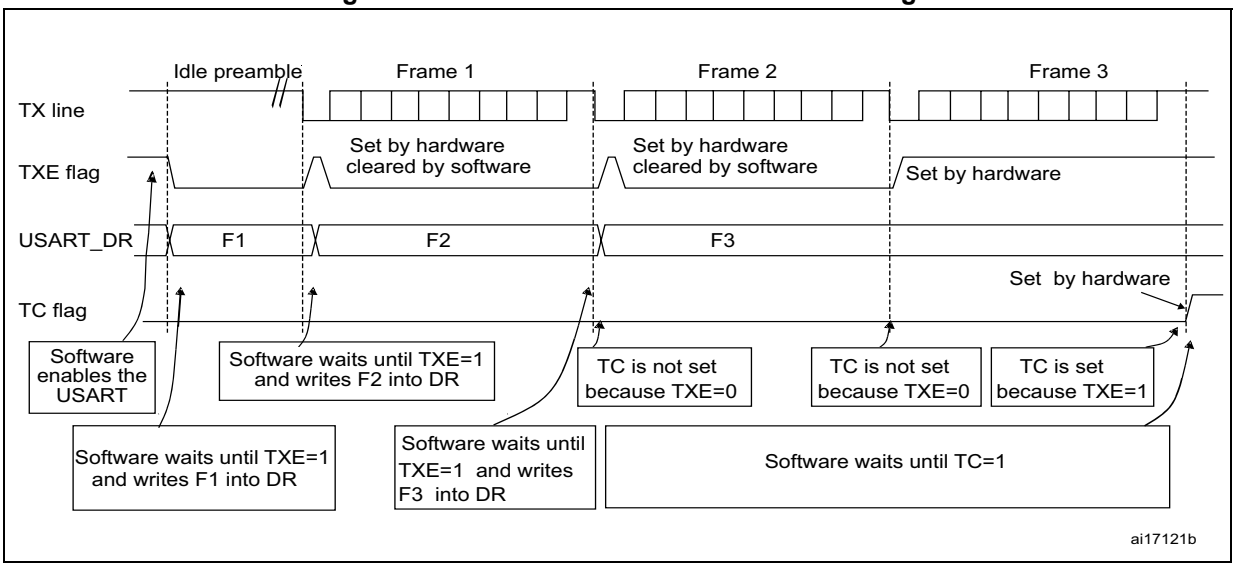
  When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write operation to USART_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.

  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 microcontroller to enter the low-power mode (see *Figure 659: TC/TXE behavior when transmitting*).

Figure 659. 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 657*).

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.

57.5.7 USART receiver

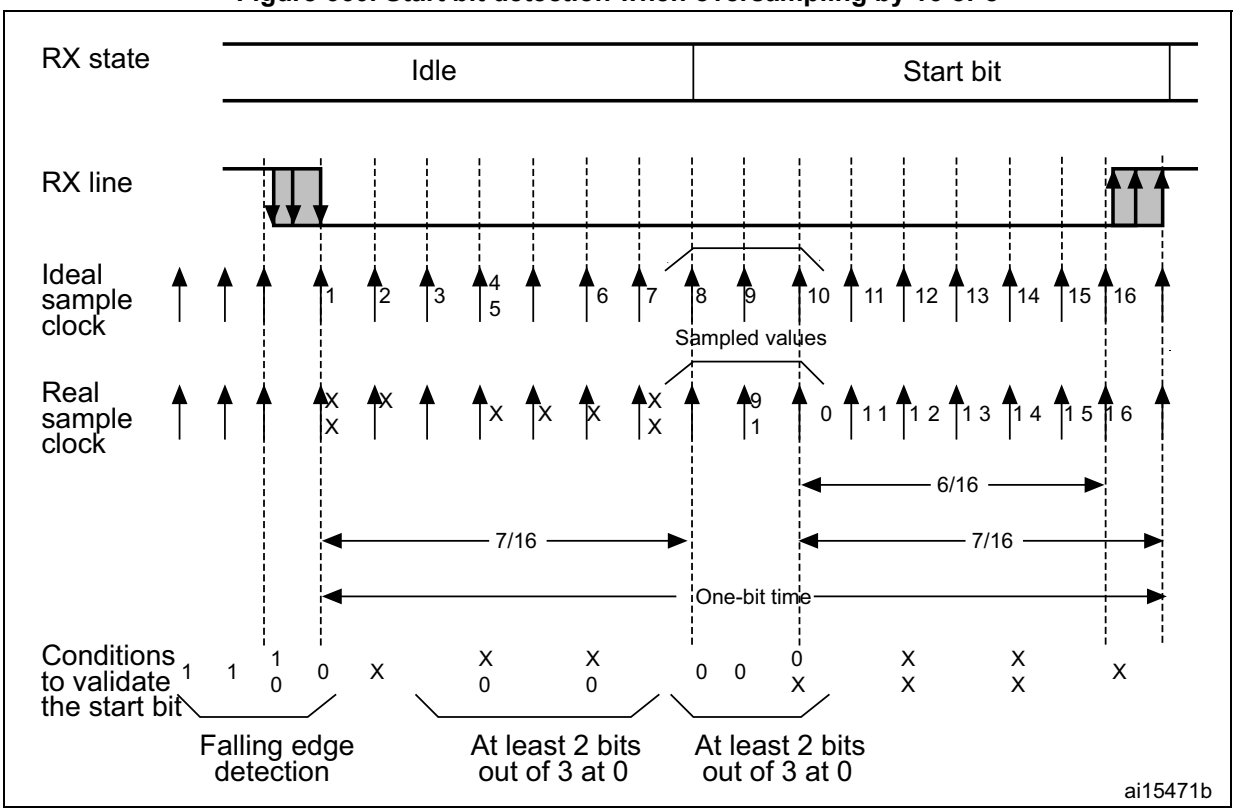
The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

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 660. 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 57.5.20: Continuous communication using USART and DMA*.
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:

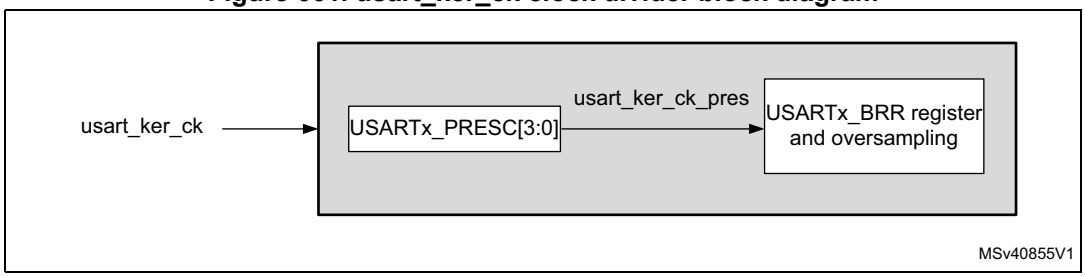
- Possible use of the USART in low-power mode
- Communication speed.

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 661. 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 662* and *Figure 663*).

Depending on the 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 57.5.9: Tolerance of the USART receiver to clock deviation on page 2275*)
- 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 the 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 *Table 560*) 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 57.5.9: Tolerance of the USART receiver to clock deviation on page 2275*). 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 NFCF bit in 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 662. Data sampling when oversampling by 16

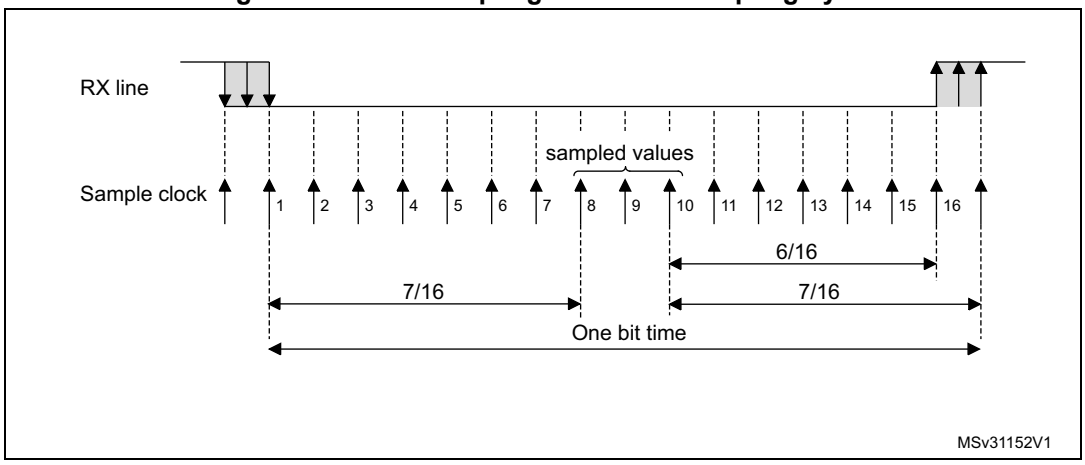


Figure 663. Data sampling when oversampling by 8

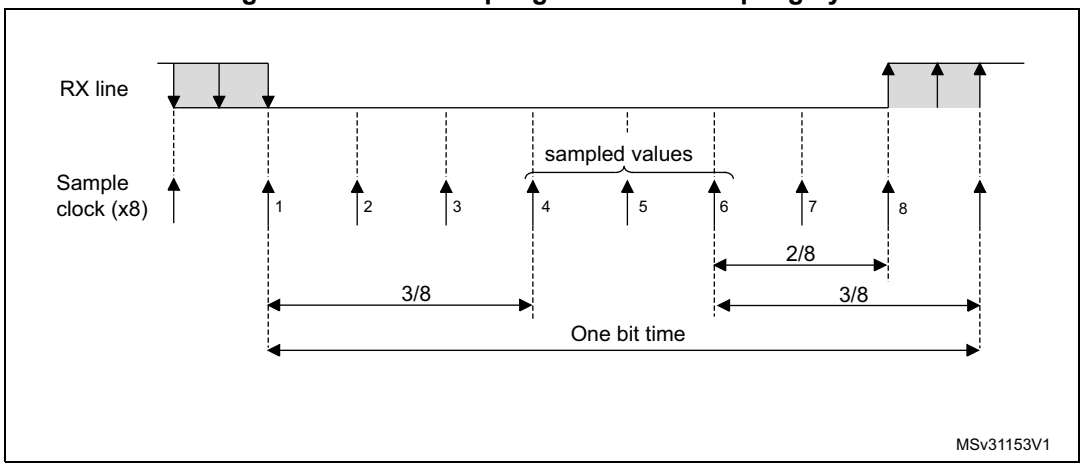


Table 560. Noise detection from sampled data

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 57.5.17: USART receiver timeout on page 2288* 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.

57.5.8 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:

$$\text{Tx/Rx baud} = \frac{\text{usart\_ker\_ck\_pres}}{\text{USARTDIV}}$$

In case of oversampling by 8, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{2 \times \text{usart\_ker\_ck\_pres}}{\text{USARTDIV}}$$

Equation 2: baud rate in Smartcard, LIN and IrDA modes (OVER8 = 0)

The baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart\_ker\_ck\_pres}}{\text{USARTDIV}}$$

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 bauds 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 (0x1667 = 0x683)
  BRR[3:0] = 0x3 >>1 = 0x1
  BRR = 0x681

Example 2

To obtain 921.6 kbauds with usart_ker_ck_pres = 48 MHz:

- In case of oversampling by 16:
  USARTDIV = 48 000 000/921 600
  BRR = USARTDIV = 0x52 = 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

57.5.9 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 561*, *Table 562*, 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 561. 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 562. 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 561 and Table 562 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).

57.5.10 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.

57.5.11 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:

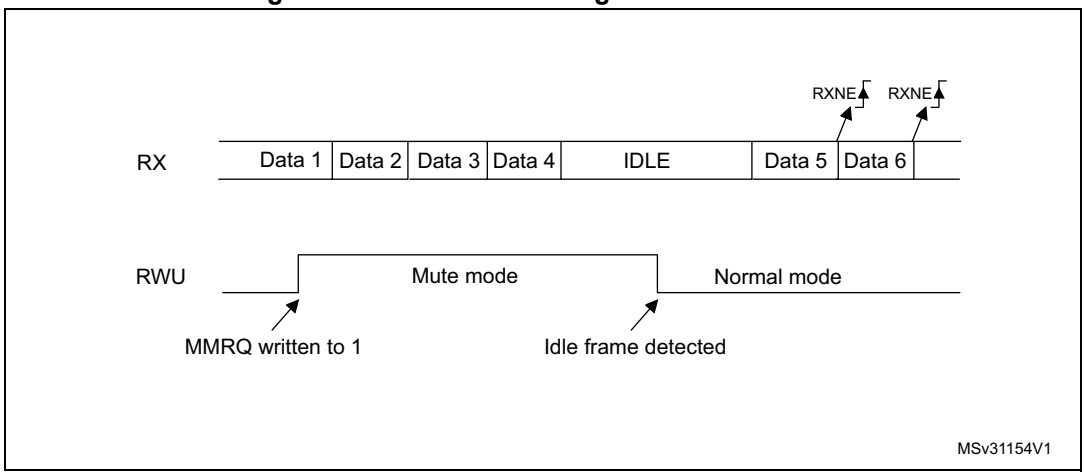
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

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 664*.

Figure 664. 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.

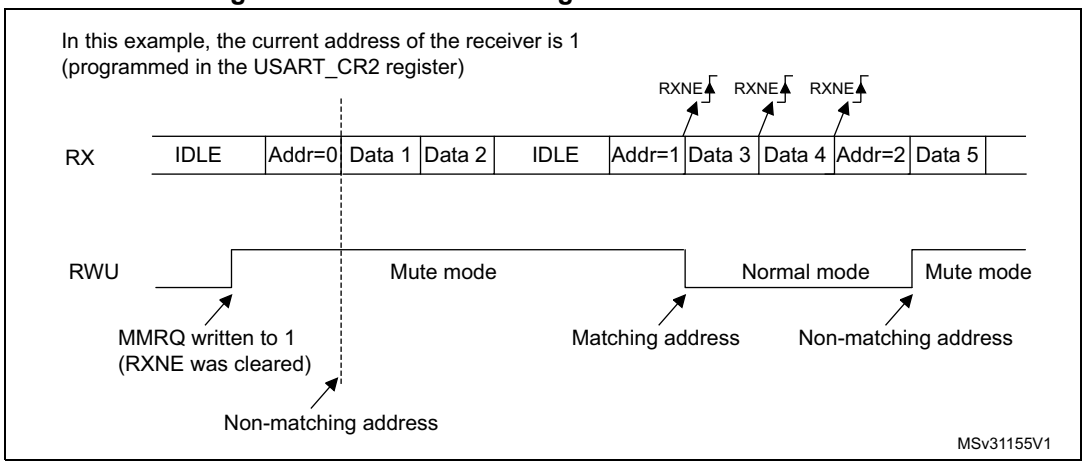
The USART also enters Mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

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 665*.

Figure 665. Mute mode using address mark detection



57.5.12 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.

57.5.13 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 563*.

Table 563. 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, then 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)).

57.5.14 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to *Section 57.4: USART implementation on page 2256*.

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 57.5.5* 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 zero bits as a break character. Then 2 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 666: Break detection in LIN mode (11-bit break length - LBDL bit is set) on page 2283*.

Examples of break frames are given on *Figure 667: Break detection in LIN mode vs. Framing error detection on page 2284*.

Figure 666. Break detection in LIN mode (11-bit break length - LBDL bit is set)

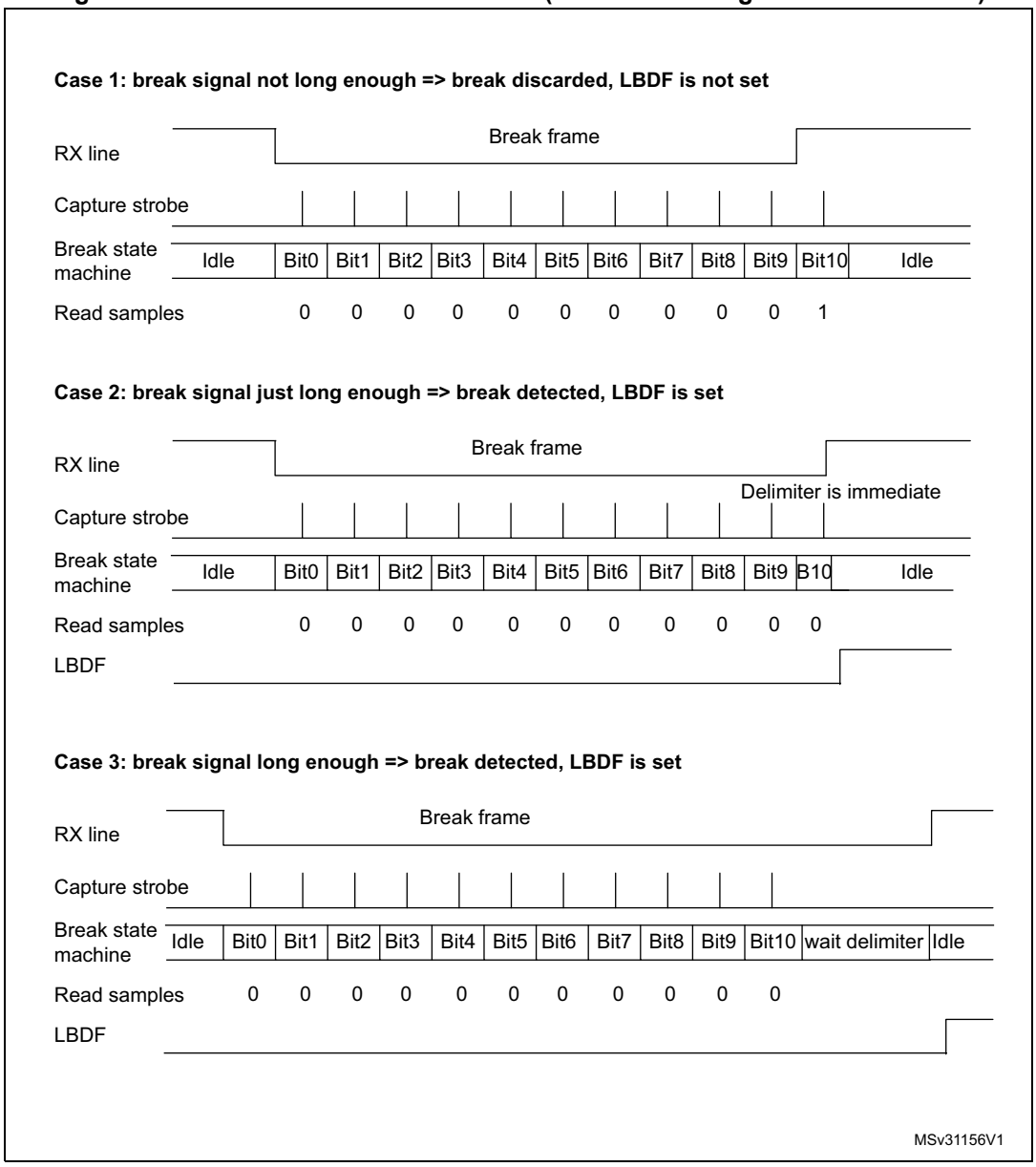
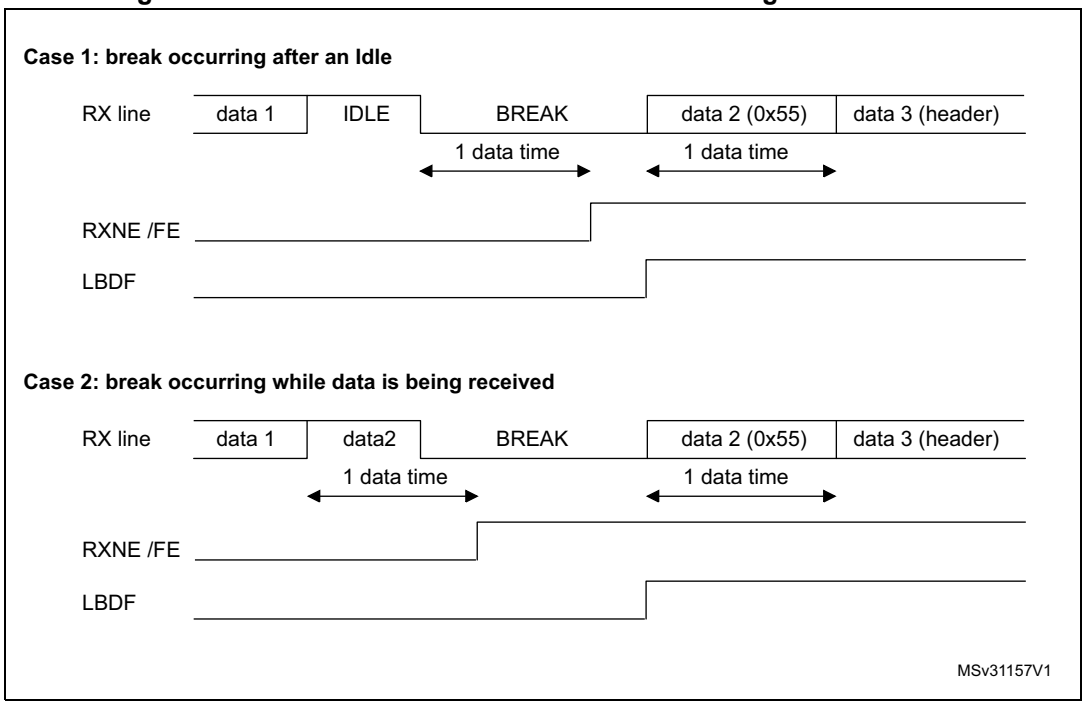


Figure 667. Break detection in LIN mode vs. Framing error detection



57.5.15 USART synchronous mode

Master mode

The Synchronous master mode is selected by programming the CLKEN bit in the USART_CR2 register to 1. In Synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in Master mode. The CK pin is the output of the USART transmitter clock. No clock pulses are sent to the CK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register is used to select the clock polarity, and the CPHA bit in the USART_CR2 register is used to select the phase of the external clock (see *Figure 668*, *Figure 669* and *Figure 670*).

During the Idle state, preamble and send break, the external CK clock is not activated.

In Synchronous master mode, the USART transmitter operates exactly like in Asynchronous mode. However, since CK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In Synchronous master mode, the USART receiver operates in a different way compared to Asynchronous mode. If RE is set to 1, the data are sampled on CK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A given setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).

Note: In Master mode, the CK pin operates in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE=1) and data are being transmitted (USART_TDR data register written). This means that it is not possible to receive synchronous data without transmitting data.

Figure 668. USART example of synchronous master transmission

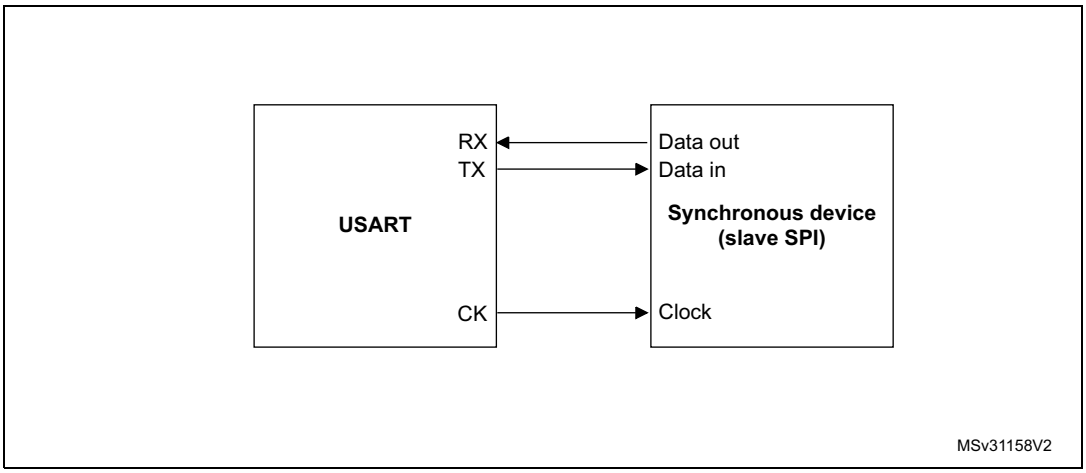


Figure 669. USART data clock timing diagram in Synchronous master mode (M bits =00)

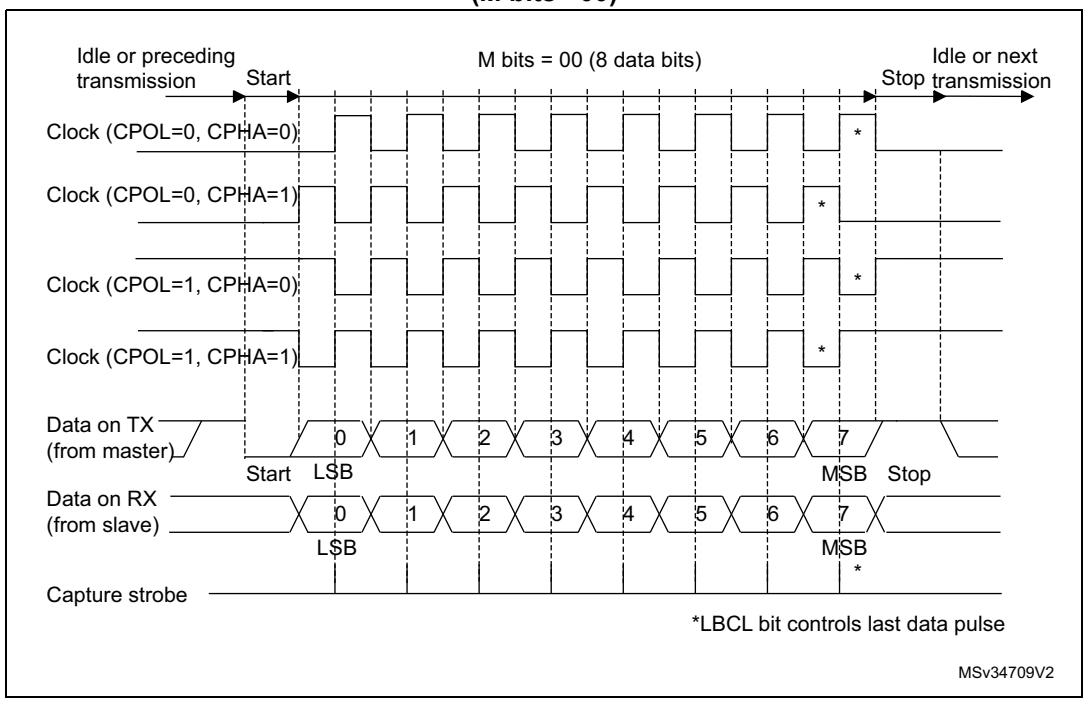
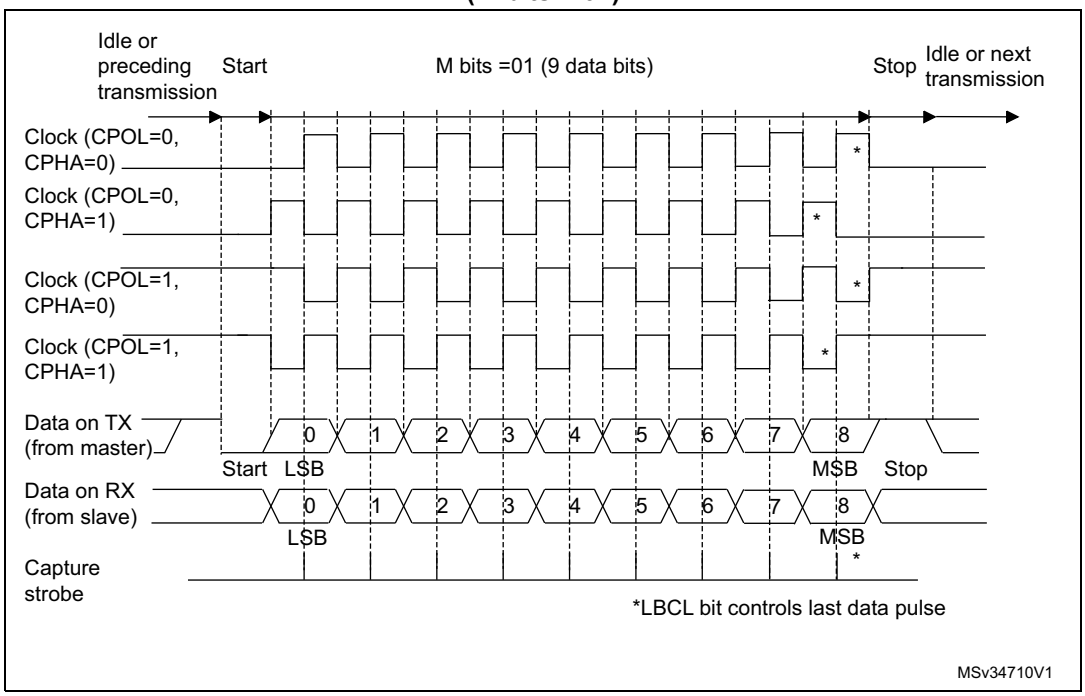


Figure 670. USART data clock timing diagram in Synchronous master mode (M bits = 01)



Slave mode

The Synchronous slave mode is selected by programming the SLVEN bit in the USART_CR2 register to 1. In Synchronous slave mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in Slave mode. The CK pin is the input of the USART in Slave mode.

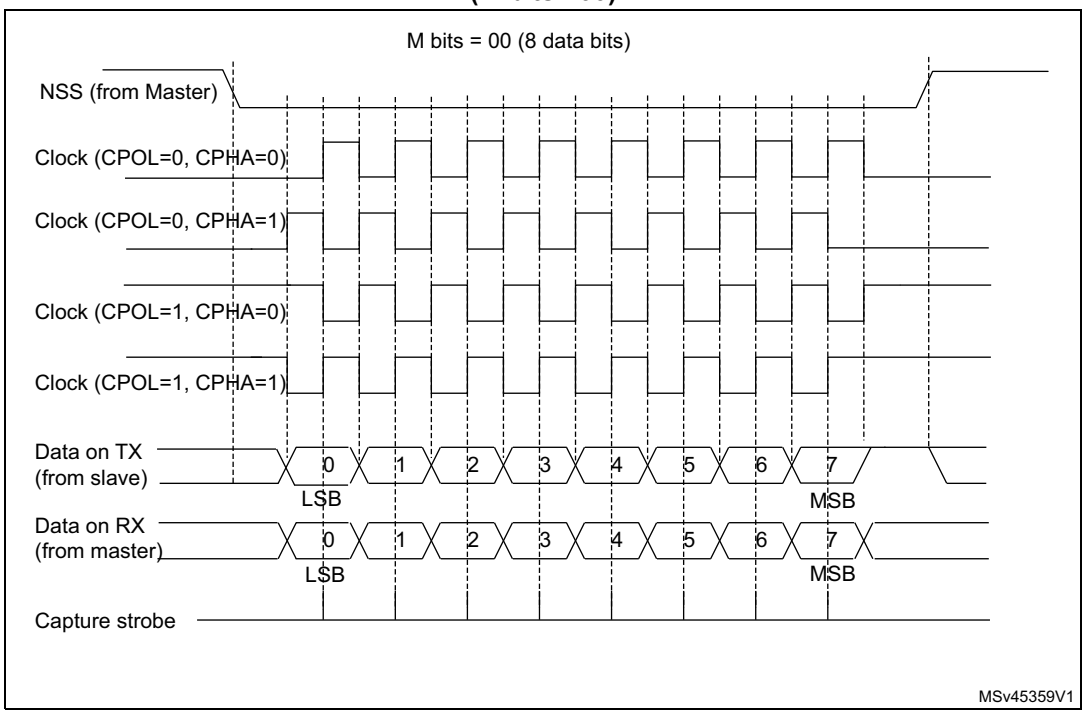
*Note: When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (*usart_ker_ck_pres*) must be greater than 3 times the CK input frequency.*

The CPOL bit and the CPHA bit in the USART_CR2 register are used to select the clock polarity and the phase of the external clock, respectively (see *Figure 671*).

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 671. 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)

  SPI slave is always selected and NSS input pin is ignored.

  The external NSS pin remains free for other application uses.
- Hardware NSS management (DIS_NSS = 0)

  The SPI slave selection depends on NSS input pin. The slave is selected when NSS is low and deselected when NSS is high.

Note: The LBCL (used only on SPI master mode), CPOL and CPHA bits have to be selected when the USART is disabled (UE=0) to ensure that the clock pulses function correctly.

In SPI slave mode, the USART must be enabled before starting the master communications (or between frames while the clock is stable). Otherwise, if the USART slave is enabled while the master is in the middle of a frame, it becomes desynchronized with the master. The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication, otherwise the SPI slave transmits zeros.

SPI Slave underrun error

When an underrun error occurs, the UDR flag is set in the USART_ISR register, and the SPI slave goes on sending the last data until the underrrun error flag is cleared by software.

The underrun flag is set at the beginning of the frame. An underrun error interrupt is triggered if EIE bit is set in the USART_CR3 register.

The underrun error flag is cleared by setting bit UDRCF in the USART_ICR register.

In case of underrun error, it is still possible to write to the TDR register. Clearing the underrun error enables sending new data.

If an underrun error occurred and there is no new data written in TDR, then the TC flag is set at the end of the frame.

Note: An underrun error may occur if the moment the data is written to the USART_TDR is too close to the first CK transmission edge. To avoid this underrun error, the USART_TDR should be written 3 usart_ker_ck *cycles before the first CK edge.*

57.5.16 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.

57.5.17 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.

57.5.18 USART Smartcard mode

This section is relevant only when Smartcard mode is supported. Refer to *Section 57.4: USART implementation on page 2256*.

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.

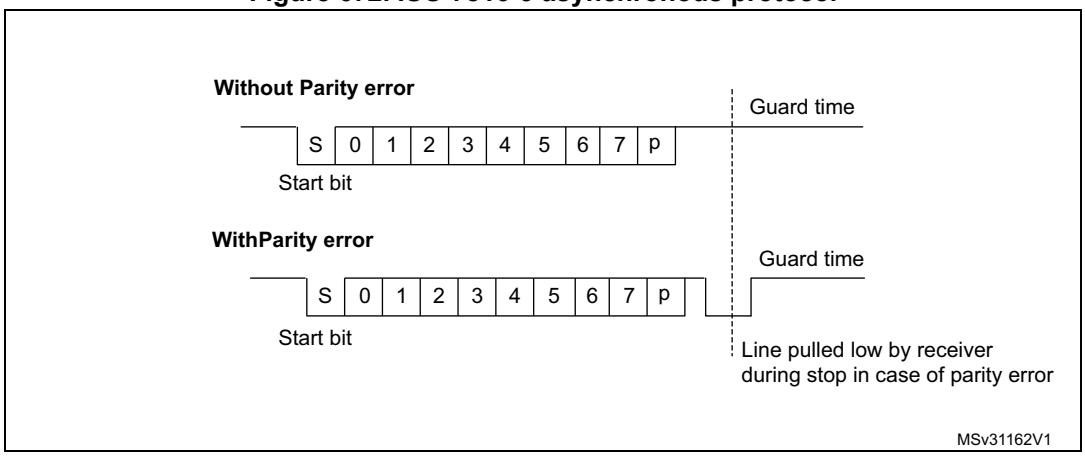
The USART should be configured as:

- 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 672 shows examples of what can be seen on the data line with and without parity error.

Figure 672. 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.

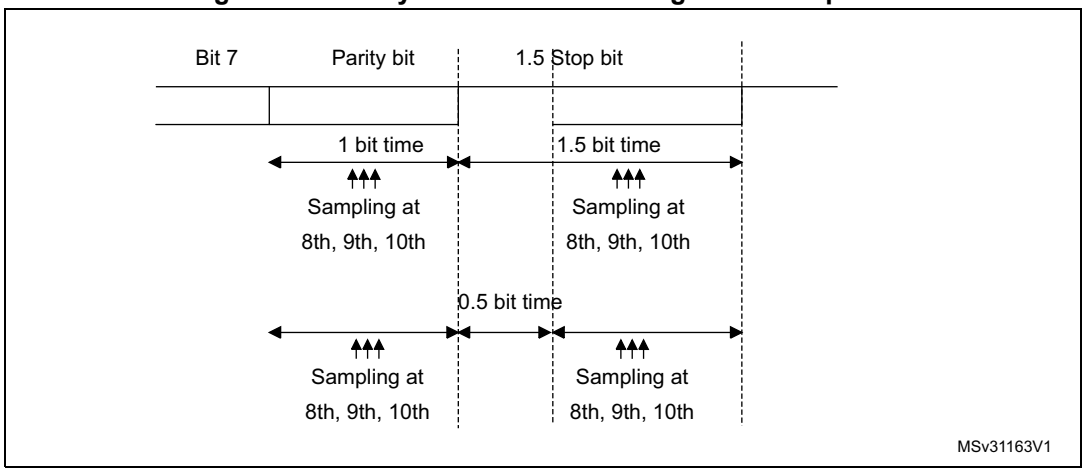
- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guardtime). If the software wants to repeat it 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 de-assertion 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 673 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 673. Parity error detection using the 1.5 stop bits



The USART can provide a clock to the Smartcard through the CK output. In Smartcard mode, CK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the USART_GTPR register. CK frequency can be programmed from usart_ker_ck_pres/2 to usart_ker_ck_pres/62, where usart_ker_ck_pres is the peripheral input clock divided by a programmed prescaler.

Block mode (T=1)

In T=1 (block) mode, the parity error transmission can be deactivated by clearing the NACK bit in the UART_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, supposing that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH => the USART received character is equal to 03 and the parity is odd.

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.

57.5.19 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to *Section 57.4: USART implementation on page 2256*.

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 674*).

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 kbauds 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 675*).
- 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 2 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 674. IrDA SIR ENDEC block diagram

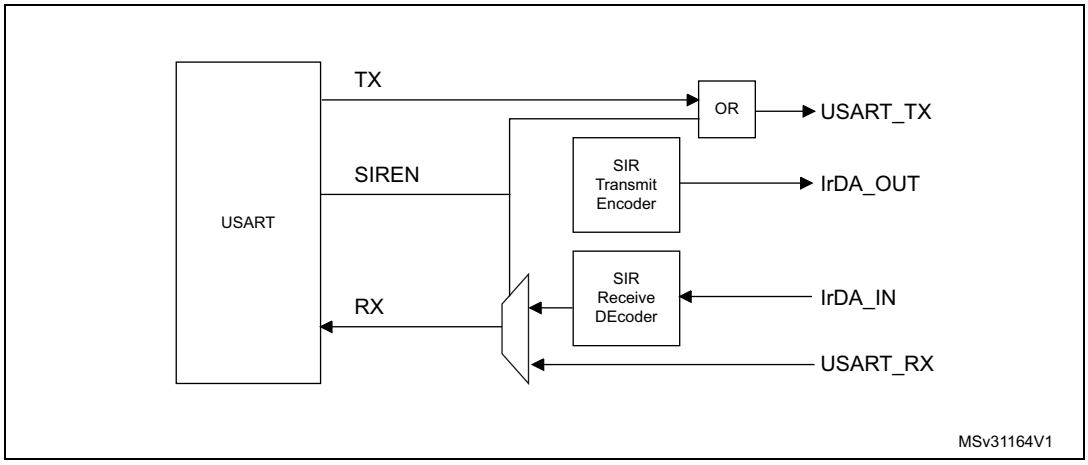
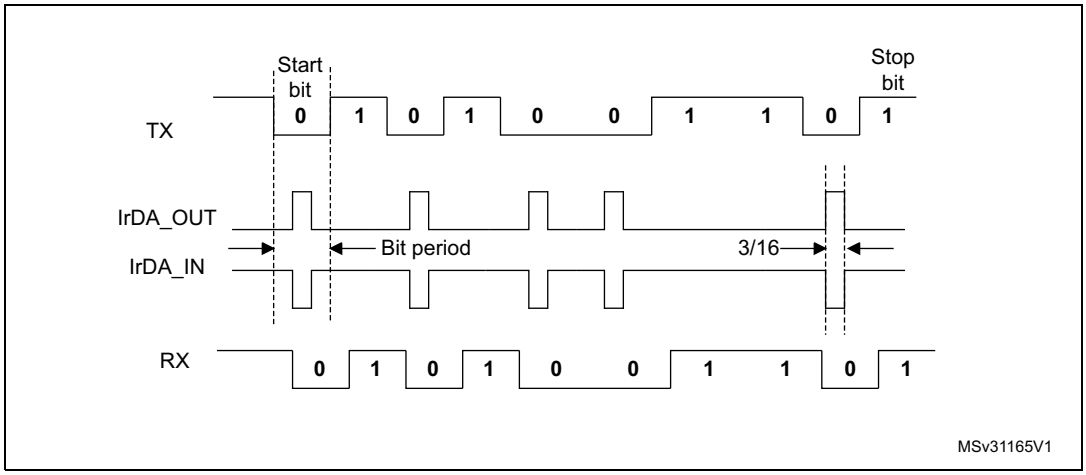


Figure 675. IrDA data modulation (3/16) - Normal mode



57.5.20 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 57.4: USART implementation on page 2256 to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in Section 57.5.7. 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 *section Direct memory access controller (DMA)*) 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):

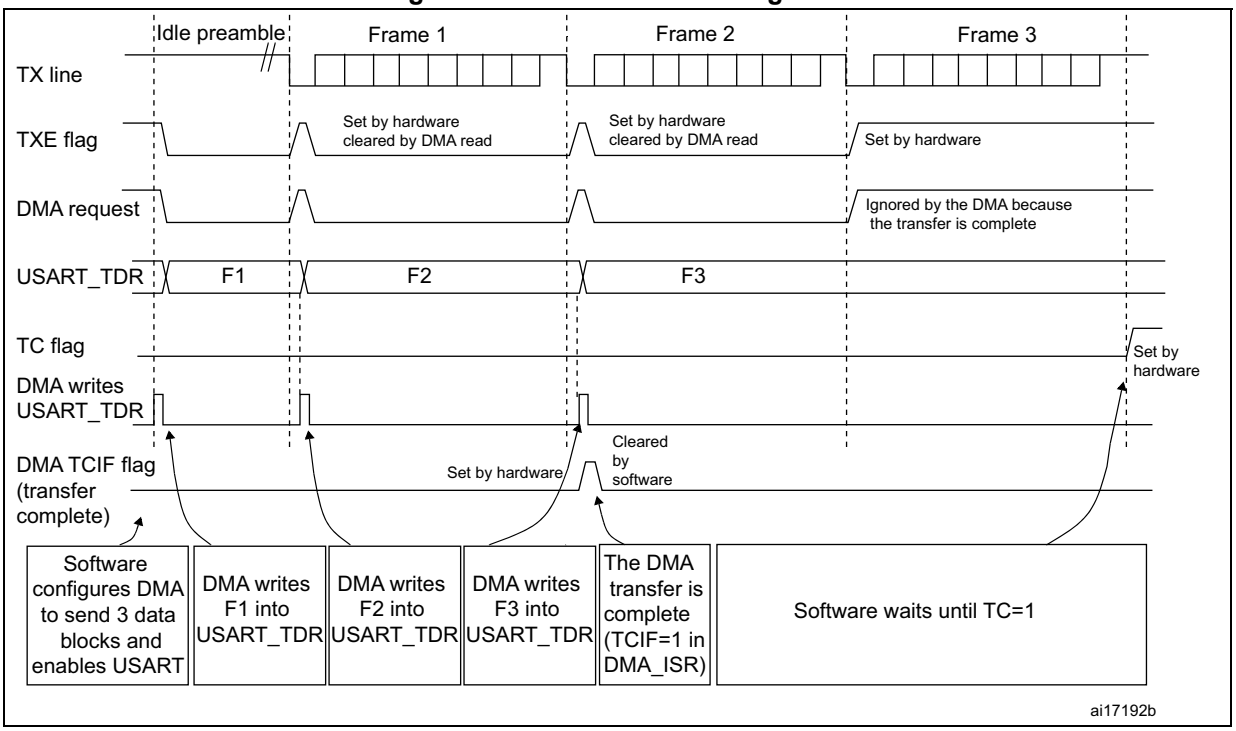
1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the 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.

Note: The DMAT bit must not be cleared before the DMA end of transfer.

Figure 676. 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

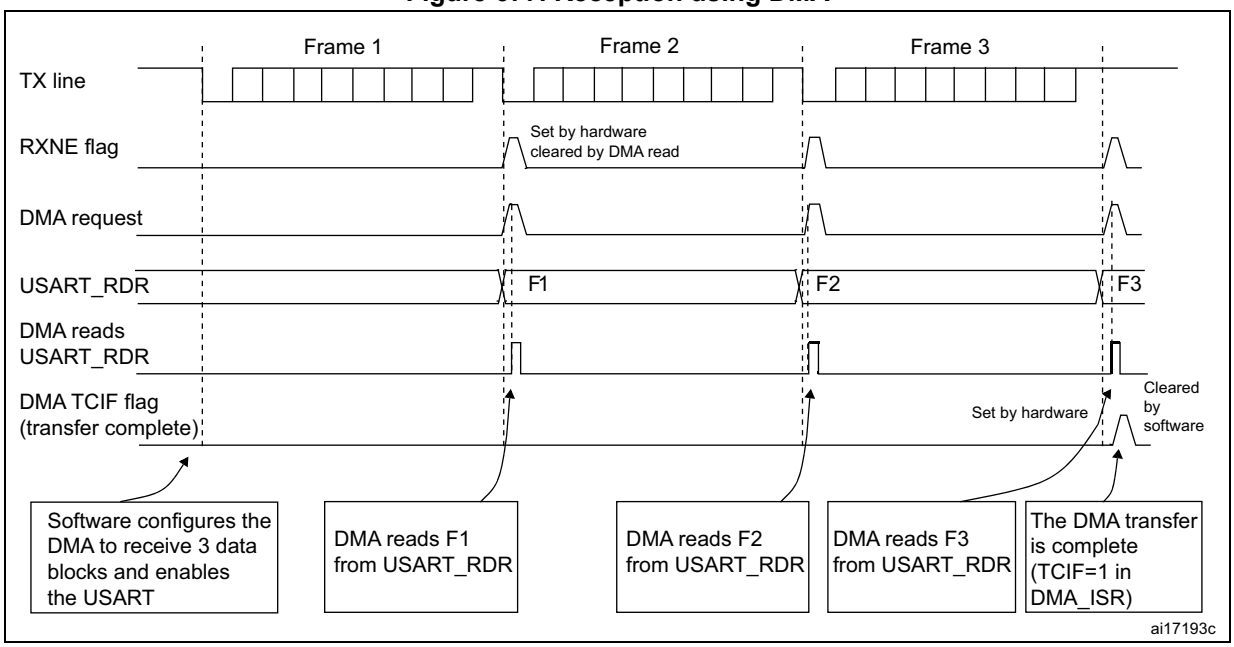
DMA mode can be enabled for reception by setting the DMAR bit in USART_CR3 register. Data are loaded from the USART_RDR register to an SRAM area configured using the DMA peripheral (refer to *section Direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for USART 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.

Note: The DMAR bit must not be cleared before the DMA end of transfer.

Figure 677. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

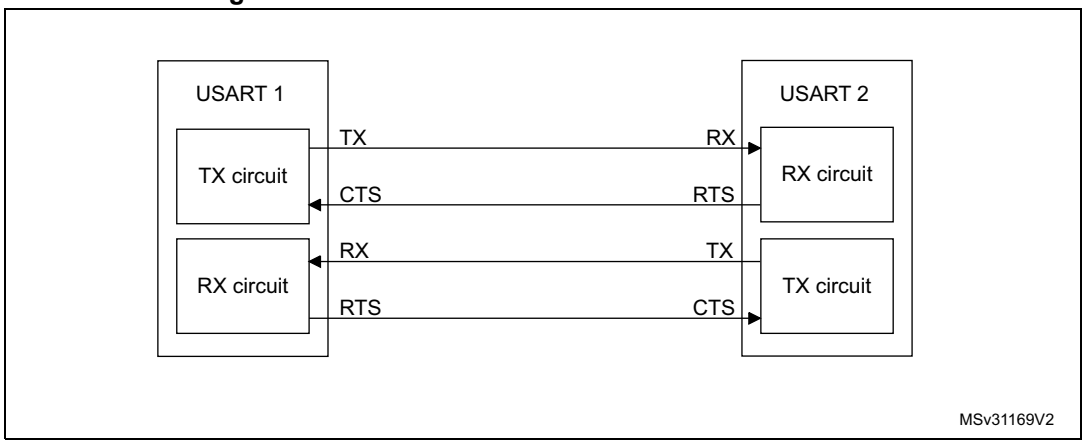
Error flagging and interrupt generation in multibuffer communication

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.

57.5.21 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the CTS input and the RTS output. The *Figure 678* shows how to connect 2 devices in this mode:

Figure 678. Hardware flow control between 2 USARTs

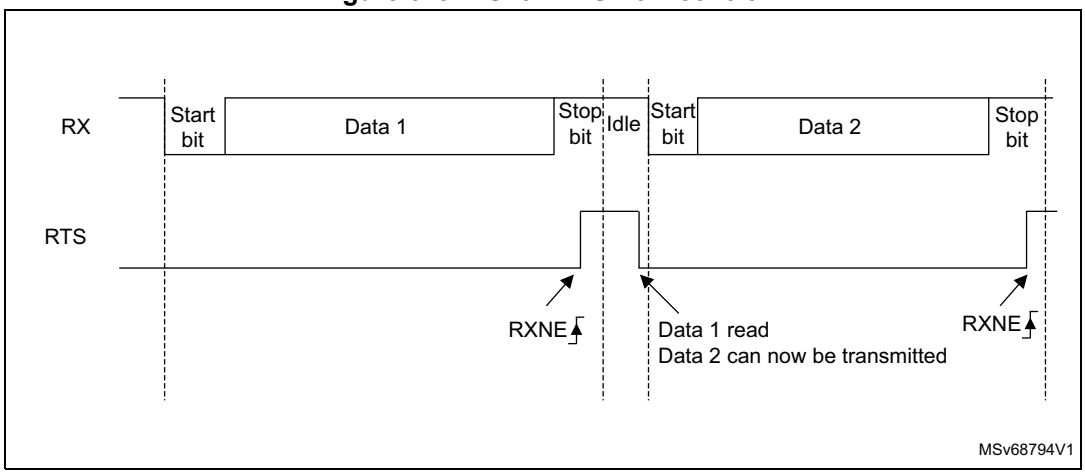


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to 1 in the USART_CR3 register.

RS232 RTS flow control

If the RTS flow control is enabled (RTSE=1), then RTS is deasserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 679* shows an example of communication with RTS flow control enabled.

Figure 679. RS232 RTS flow control



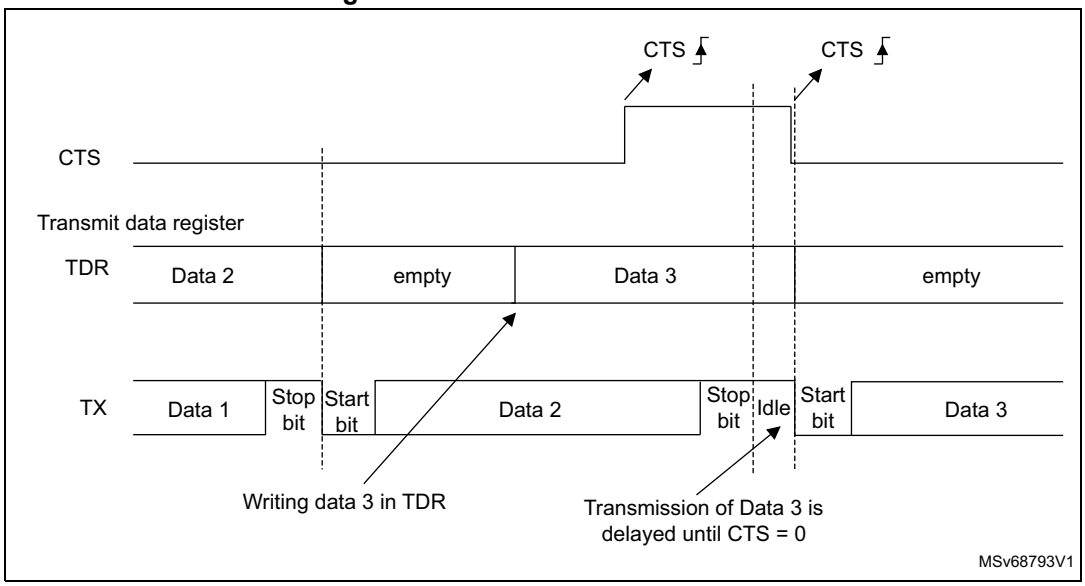
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE=0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. *Figure 680* shows an example of communication with CTS flow control enabled.

Figure 680. RS232 CTS flow control



Note: For correct behavior, CTS must be deasserted at least 3 USART clock source periods before the end of the current character. In addition it should be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the USART_CR3 control register. This enables the user to activate the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the USART_CR1 control register. The de-assertion 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).

57.5.22 USART Autonomous mode

The USART peripheral can be functional in Stop mode thanks to the Autonomous mode. This mode can also be used in Run and Sleep mode. The UESM bit must be set prior to entering low-power mode.

The APB clock is requested by the peripheral each time the USART status needs to be updated. Once the USART receives the APB clock, it generates either an interrupt or a DMA request, depending on the peripheral configuration.

If an interrupt is generated, the device wakes up from Stop mode. If no interrupt is generated, the device remains in Stop mode but the kernel and APB clocks are still available for the USART and all the autonomous peripherals enabled in the reset and clock controller (RCC). If DMA requests are enabled, the data are directly transferred to/from the SRAM thanks to the DMA while the product remains in Stop mode.

Transmission mode

In transmission, the APB clock is requested only when the TE bit is set and in the following cases:

- If the FIFO mode is enabled, the APB clock is requested when
  - the TxFIFO is empty (TXFE = 1) and the corresponding interrupt is enabled (TXFEIE = 1)
  - the TxFIFO threshold is reached (TXFT = 1) and the corresponding interrupt is enabled (TXFTIE = 1)
  - the TxFIFO is not full (TXFNF = 1) and the corresponding interrupt or DMA is enabled (TXFNFIE = 1 or DMAT = 1)
- If the FIFO mode is disabled, the APB clock is requested as soon as data are transferred to the shift register. The DMA or associated interrupt must be enabled.

The TE bit is set by hardware if an asynchronous trigger is detected.

A transmission is automatically launched when an asynchronous trigger is detected in Run, Sleep or Stop mode. The trigger is selected through the TRIGSEL bit in the USART_AUTOCR register. It sets the TE bit in the USART_CR1 register and generates an APB clock request to enable the transfer. The APB clock is requested until the transmission completes and the TE bit is cleared by hardware when the programmed number of data to be transmitted (TDN bits field in the USART_AUTOCR register) is reached. In this case, the TC flag is set when the number of data to be transmitted is reached and the last byte is transmitted.

Reception mode

- If the FIFO mode is enabled, the APB clock is requested when
  - the RxFIFIO is full (RXFF = 1) and the corresponding interrupt is enabled (RXFFIE = 1)
  - the RxFIFO threshold is reached (RXFT = 1) and the corresponding interrupt is enabled (RXFTIE = 1)
  - the RxFIFO is not empty (RXFNE = 1) and the corresponding interrupt or DMA is enabled (RXFNEIE= 1 or DMAR =1)
- If the FIFO mode is disabled, the APB clock is requested when the USART finishes sampling data and it is ready to be written in the USART_RDR. The DMA or the associated interrupt must be enabled.

Note: The APB clock is requested in Reception mode when an overrun error occurs (ORE = 1). The EIE bit must be set to enable the generation of an interrupt and waking up the MCU, and the OVRDIS bit must remain cleared. The APB clock request is kept until the interrupt flag is cleared.

The APB clock is also requested in Reception mode when a Parity/Noise/Framing error occurs and the DMA is used for reception. The APB clock request is kept until the interrupt flag is cleared.

Only UART and SPI master modes support the Autonomous mode.

Determining the maximum USART baud rate that enables to correctly wake up the microcontroller from low-power mode

The maximum baud rate that enables to correctly wake up the microcontroller from low-power mode depends on the wakeup time parameter (refer to the device datasheet) and on the USART receiver tolerance (see *Section 57.5.9: 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 561: 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

$D_{WUmax} = t_{WUUSART} / (11 \times T_{bitmin})$

$T_{bitmin} = t_{WUUSART} / (11 \times D_{WUmax})$

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 fact, we need to consider at least the usart_ker_ck inaccuracy (DREC).

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} = USART receiver tolerance – DREC = 3.41% - 1% = 2.41%

T_{bitmin} = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate enables to wakeup correctly from low-power mode is: 1/11.32 µs = 88.36 kbauds.

57.6 USART in low-power modes

Table 564. 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. If the USART is clocked by an oscillator available in Stop mode, transfers in Asynchronous and SPI master modes are functional. DMA requests are functional, and the interrupts cause the device to exit Stop mode.
Standby	The USART peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 57.4: 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.

57.7 USART interrupts

Refer to *Table 565* for a detailed description of all USART interrupt requests.

Table 565. 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	Yes	No
	Transmit FIFO Not Full	TXFNF	TXFNFIE	TXFIFO full		Yes	
	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		Yes	
	Transmission Complete Before Guard Time	TCBGT	TCBGTIE	Write TDR or write 1 in TCBGT		No	

Table 565. USART interrupt requests (continued)

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	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	RX-NEIE/RXFNEIE	Write 1 in ORECF		Yes	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		Yes(3)	
	LIN break	LBDF	LBDIE	Write 1 in LBDCF		No	
	Noise error in multibuffer communication.	NE	EIE	Write 1 in NFCF		Yes(3)	
	Overrun error in multibuffer communication.	ORE(4)		Write 1 in ORECF		Yes	
	Framing Error in multibuffer communication.	FE		Write 1 in FECF		Yes(3)	
	Character match	CMF	CMIE	Write 1 in CMCF		Yes(5)	
	Receiver timeout	RTOF	RTOFIE	Write 1 in RTOCCF		No	
	End of Block	EOBF	EOBIE	Write 1 in EOBCF		No	
	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 57.4: 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. Parity/Noise/Framing error interrupts enable waking up from Stop modes when the DMA is used.
4. When OVRDIS = 0.
5. The DMA must be used when the FIFO mode is enabled.

57.8 USART registers

Refer to *Section 1.2 on page 104* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

57.8.1 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 57.4: USART implementation on page 2256.

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 57.4: USART implementation on page 2256.

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 57.4: USART implementation on page 2256.

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 57.4: USART implementation on page 2256.

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. When the Autonomous mode is not used, TE bit is set and cleared by software. When the Autonomous mode is used, TE bit becomes a status bit, which is set and cleared by hardware.

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 request its kernel clock and is not functional in 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 functional in low-power mode.

1: USART functional in low-power mode.

Note: The UESM bit must be set at the initialization phase.

If the USART does not support the wakeup from low-power mode, this bit is reserved and must be kept at reset value. Refer to Section 57.4: USART implementation on page 2256.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In Smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.

57.8.2 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 57.4: USART implementation on page 2256.

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 57.4: USART implementation on page 2256.

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 57.4: USART implementation on page 2256.

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 57.4: USART implementation on page 2256.

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. When the Autonomous mode is not used, TE bit is set and cleared by software. When the Autonomous mode is used, TE bit becomes a status bit, which is set and cleared by hardware.

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 request its kernel clock and is not functional in 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 functional in low-power mode.

1: USART functional in low-power mode.

Note: The UESM bit must be set at the initialization phase.

If the USART does not support the wakeup from low-power mode, this bit is reserved and must be kept at reset value. Refer to Section 57.4: USART implementation on page 2256.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In Smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.

57.8.3 USART control register 2 (USART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw			rw

Bits 31:24 **ADD[7:0]**: Address of the USART node

These bits give the address of the USART node in Mute mode or a character code to be recognized in low-power or Run mode:

- In Mute mode: they are used in multiprocessor communication to wakeup from Mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When a character, received during low-power mode, corresponds to the character programmed through ADD[7:0] bitfield, the CMF flag is set and wakes up the device from low-power mode if the corresponding interrupt is enabled by setting CMIE bit.
- In Run mode with Mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the reception is disabled (RE = 0) or when the USART is disabled (UE = 0).

Bit 23 **RTOEN**: Receiver timeout enable

This bit is set and cleared by software.

0: Receiver timeout feature disabled.

1: Receiver timeout feature enabled.

When this feature is enabled, the RTOF flag in the USART_ISR register is set if the RX line is idle (no reception) for the duration programmed in the RTOR (receiver timeout register).

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 57.4: USART implementation on page 2256.

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 -> 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 57.4: USART implementation on page 2256.

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 57.4: USART implementation on page 2256.

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 57.4: USART implementation on page 2256.

Bits 13:12 **STOP[1:0]**: stop bits

These bits are used for programming the stop bits.

00: 1 stop bit

01: 0.5 stop bit.

10: 2 stop bits

11: 1.5 stop bits

This bitfield can only be written when the USART is disabled (UE=0).

Bit 11 **CLKEN**: Clock enable

This bit enables the user to enable the CK pin.

0: CK pin disabled

1: CK pin enabled

This bit can only be written when the USART is disabled (UE=0).

Note: If neither Synchronous mode nor Smartcard mode is supported, this bit is reserved and must be kept at reset value. Refer to Section 57.4: USART implementation on page 2256.

In Smartcard mode, in order to provide correctly the CK clock to the smartcard, the steps below must be respected:
UE = 0
SCEN = 1
GTPR configuration
CLKEN= 1
UE = 1

Bit 10 **CPOL**: Clock polarity

This bit enables the user to select the polarity of the clock output on the CK pin in Synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on CK pin outside transmission window

1: Steady high value on CK pin outside transmission window

This bit can only be written when the USART is disabled (UE=0).

Note: If Synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 57.4: USART implementation on page 2256.

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the CK pin in Synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see *Figure 662* and *Figure 663*)

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 57.4: USART implementation on page 2256.

Bit 8 **LBCL**: Last bit clock pulse

This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the CK pin in Synchronous mode.

0: The clock pulse of the last data bit is not output to the CK pin

1: The clock pulse of the last data bit is output to the CK pin

Caution: The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bit in the USART_CR1 register.

This bit can only be written when the USART is disabled (UE=0).

Note: If Synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 57.4: USART implementation on page 2256.

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 57.4: USART implementation on page 2256.

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 57.4: USART implementation on page 2256.

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 57.4: USART implementation on page 2256.

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 57.4: USART implementation on page 2256.

Note: The CPOL, CPHA and LBCL bits should not be written while the transmitter is enabled.

57.8.4 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	Res.	Res.	Res.	SCARCNT[2:0]			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
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 57.4: USART implementation on page 2256.

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.

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

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 Transmission 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 57.4: USART implementation on page 2256.

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 57.4: USART implementation on page 2256.

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 57.4: USART implementation on page 2256.

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 are 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 57.4: USART implementation on page 2256.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission is completed before stopping.If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the USART is disabled (UE=0)

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 57.4: USART implementation on page 2256.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE=0).

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 57.4: USART implementation on page 2256.

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 57.4: USART implementation on page 2256.

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 57.4: USART implementation on page 2256.

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 57.4: USART implementation on page 2256.

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 57.4: USART implementation on page 2256.

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=1or 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.

57.8.5 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] correspond to 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.

57.8.6 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 57.4: USART implementation on page 2256.

Bits 7:0 **PSC[7:0]**: Prescaler value

Condition: IrDA low-power and normal IrDA mode

PSC[7:0] = IrDA Normal and Low-power baud rate

This bitfield is used for programming 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):

00000000: Reserved - do not program this value

00000001: divides the source clock by 1

00000010: divides the source clock by 2

...

Condition: Smartcard mode

PSC[4:0]: Prescaler value

This bitfield is used for programming 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 2

00010: divides the source clock by 4

00011: divides the source clock by 6

...

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 57.4: USART implementation on page 2256.

57.8.7 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 bit duration.

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 57.4: USART implementation on page 2256**.*

57.8.8 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 57.4: USART implementation on page 2256.

57.8.9 USART interrupt and status register [alternate] (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 28 if FIFO/Smartcard mode enabled

XX = 08 if FIFO enabled and Smartcard mode 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	REACK	TEACK	Res.	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
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 57.4: USART implementation on page 2256.

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 57.4: USART implementation on page 2256.

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 Reserved, must be kept at reset value.

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 57.4: USART implementation on page 2256.

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: No break character transmitted

1: Break character transmitted

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 57.4: USART implementation on page 2256.

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 EOBIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to 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 57.4: USART implementation on page 2256.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE=1 in the USART_CR2 register.

In Smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE=1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 57.4: USART implementation on page 2256.

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 is 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. The TC flag behaves as follows:

- When TDN = 0, the TC flag is set when the transmission of a frame containing data is complete and when TXE/TXFE is set.
- When TDN is equal to the number of data in the TXFIFO, the TC flag is set when TXFIFO is empty and TDN is reached.
- When TDN is greater than the number of data in the TXFIFO, TC remains cleared until the TXFIFO is filled again to reach the programmed number of data to be transferred.
- When TDN is less than the number of data in the TXFIFO, TC is set when TDN is reached even if the TXFIFO is not empty.

An interrupt is generated if TCIE=1 in the USART_CR1 register.

TC bit is cleared by software by writing 1 to the TCCF in the USART_ICR register or by writing to the USART_TDR register.

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, meaning that data can be read from the USART_RDR register. Every read operation from the USART_RDR frees a location in the RXFIFO.

RXFNE is cleared when the RXFIFO is empty. The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXFNEIE=1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE=1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (i.e. a new idle line occurs).

If Mute mode is enabled (MME=1), IDLE is set if the USART is not mute (RWU=0), whatever the Mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXFNEIE=1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF 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 57.5.9: Tolerance of the USART receiver to clock deviation on page 2275).

This error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.

When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: This error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in Reception 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.

57.8.10 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	Res.	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
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 57.4: USART implementation on page 2256.

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 57.4: USART implementation on page 2256.

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 Reserved, must be kept at reset value.

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 57.4: USART implementation on page 2256.

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: No break character transmitted

1: Break character transmitted

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 57.4: USART implementation on page 2256.

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 EOBIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to 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 57.4: USART implementation on page 2256.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE=1 in the USART_CR2 register.

In Smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE=1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 57.4: USART implementation on page 2256.

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 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. The TC flag is set 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 cleared by software by writing 1 to the TCCF in the USART_ICR register or by writing to the USART_TDR register.

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 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

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 NFCF 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 57.5.9: Tolerance of the USART receiver to clock deviation on page 2275).

This error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.

When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: This error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in Reception 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.

57.8.11 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.	Res.	Res.	Res.	CMCF	Res.
														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 31:18 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 57.4: USART implementation on page 2256

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 57.4: USART implementation on page 2256.

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 57.4: USART implementation on page 2256.

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 57.4: USART implementation on page 2256.

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 57.4: USART implementation on page 2256.

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.

57.8.12 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 656*).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

57.8.13 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 656*).

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.

57.8.14 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 equal to 1011 i.e. input clock divided by 256.

57.8.15 USART Autonomous mode control register (USART_AUTOCR)

Address offset: 0x30

Reset value: 0x8000 0000

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

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

Bits 22:19 **TRIGSEL[3:0]**: Trigger selection bits

Refer to *Description of USART interconnections*.

This bitfield can be written only when the UE bit is cleared in USART_CR1 register.

0000: usart_trg0 selected

0001: usart_trg1 selected

...

1111: usart_trg15 selected

Note: This bitfield can be written only when the UE bit of USART_CR1 register is cleared.

Bit 18 **IDLEDIS**: Idle frame transmission disable bit after enabling the transmitter

0: Idle frame sent after enabling the transmitter (TE = 1 in USART_CR1)

1: Idle frame not sent after enabling the transmitter

Note: This bitfield can be written only when the UE bit of USART_CR1 register is cleared.

Bit 17 **TRIGEN**: Trigger enable bit

0: Trigger disabled

1: Trigger enabled

Note: This bitfield can be written only when the UE bit of USART_CR1 register is cleared.

When a trigger is detected, TE is set to 1 in USART_CR1 and the data transfer is launched.

Bit 16 **TRIGPOL**: Trigger polarity bit

This bitfield can be written only when the UE bit is cleared in USART_CR1 register.

0: Trigger active on rising edge

1: Trigger active on falling edge

Bits 15:0 **TDN[15:0]**: TDN transmission data number

This bitfield enables the programming of the number of data to be transmitted. It can be written only when UE is cleared in USART_CR1.

57.8.16 USART register map

Table 566. 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 mode 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 mode 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

Table 566. 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
0x08	USART_CR3	TXFTCFG[2:0]			RXFTIE[2:0]	RXFTCFG			TCBGTIE	TXFTIE	Res.	Res.	Res.	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
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	Res.	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
	Reset value					0	0	X	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x1C	USART_ISR FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	REACK	TEACK	Res.	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	1	0	0	0	0	0	0	0
0x20	USART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
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
0x30	USART_ AUTOCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGSEL [3:0]				IDLEDIS	TRIGEN	TRIGPOL	TDN[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

Refer to *Section 2.3 on page 115* for the register boundary addresses.

58 Low-power universal asynchronous receiver transmitter (LPUART)

This section describes the low-power universal asynchronous receiver transmitted (LPUART).

58.1 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 bauds. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Even when the microcontroller 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.

58.2 LPUART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Programmable baud rate
- From 300 bauds to 9600 bauds 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
- Wakeup from Stop mode
- Autonomous functionality in Stop mode

58.3 LPUART implementation

The tables below describe LPUART implementation on STM32U575/585 devices. It also includes USARTs and UARTs for comparison.

Table 567. STM32U575/585 features

USART modes/features	STM32U575/585
USART1	FULL
USART2	FULL
USART3	FULL
UART4	BASIC
UART5	BASIC
LPUART1	LP

Table 568. USART/LPUART features

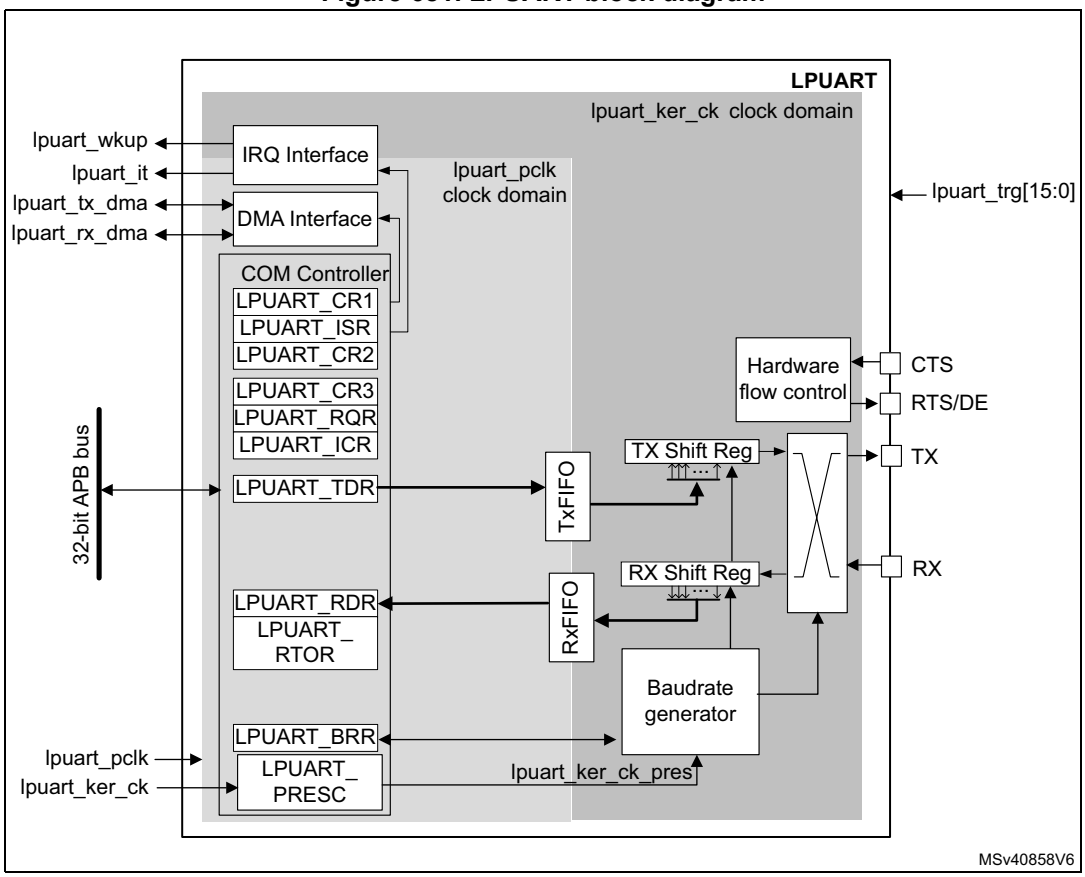
USART modes/features(1)	Full feature set	Basic feature set	Low-power feature set
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (Master/Slave)	X	-	-
Smartcard mode	X	-	-
Single-wire Half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	-
LIN mode	X	X	-
Dual clock domain	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		
Wakeup from low-power mode	X(2)	X(2)	X(3)
Autonomous mode	X	X	X

1. X = supported.
2. Wakeup supported from Stop 0 and Stop 1 modes.
3. Wakeup supported from Stop 0, Stop 1 and Stop 2 modes.

58.4 LPUART functional description

58.4.1 LPUART block diagram

Figure 681. LPUART block diagram



58.4.2 LPUART pins and internal signals

Description LPUART input/output pins

- LPUART bidirectional communications

  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 **DE** (Driver Enable) pin is required in RS485 Hardware control mode. This signal activates the transmission mode of the external transceiver.

Refer to *Table 569* and *Table 570* for the list of LPUART input/output pins and internal signals.

Table 569. LPUART input/output pins

Pin name	Signal type	Description
LPUART_RX	Input	Serial data receive input.
LPUART_TX	Output	Transmit data output.
LPUART_CTS	Input	Clear to send
LPUART_RTS	Output	Request to send
LPUART_DE(1)	Output	Driver enable

1. LPUART_DE and LPUART_RTS share the same pin.

Description LPUART input/output signals

Table 570. LPUART internal input/output signals

Pin name	Signal type	Description
usart_pclk	Input	APB clock
lpuart_ker_ck	Input	LPUART kernel clock
lpuart_wkup	Output	LPUART provides a wakeup interrupt

Table 570. LPUART internal input/output signals

Pin name	Signal type	Description
lpuart_it	Output	LPUART global interrupt
lpuart_tx_dma	Input/output	LPUART transmit DMA request
lpuart_rx_dma	Input/output	LPUART receive DMA request
lpuart_trg[15:0]	Input	LPUART triggers.

Description LPUART interconnections

Table 571. LPUART interconnections (LPUART1)

Signal name	Source
lpuart_trg0	lpdma1_ch0_tc
lpuart_trg1	lpdma1_ch1_tc
lpuart_trg2	lpdma1_ch2_tc
lpuart_trg3	lpdma1_ch3_tc
lpuart_trg4	exti6
lpuart_trg5	exti8
lpuart_trg6	lptim1_out
lpuart_trg7	lptim3_out
lpuart_trg8	comp1_out
lpuart_trg9	comp2_out
lpuart_trg10	rtc_alra_trg
lpuart_trg11	rtc_wut_trg
lpuart_trg12	-
lpuart_trg13	-
lpuart_trg14	-
lpuart_trg15	-

58.4.3 LPUART clocks

The simplified block diagram given in *Figure 681* 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 not supported, 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.

58.4.4 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 657*).

- 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.

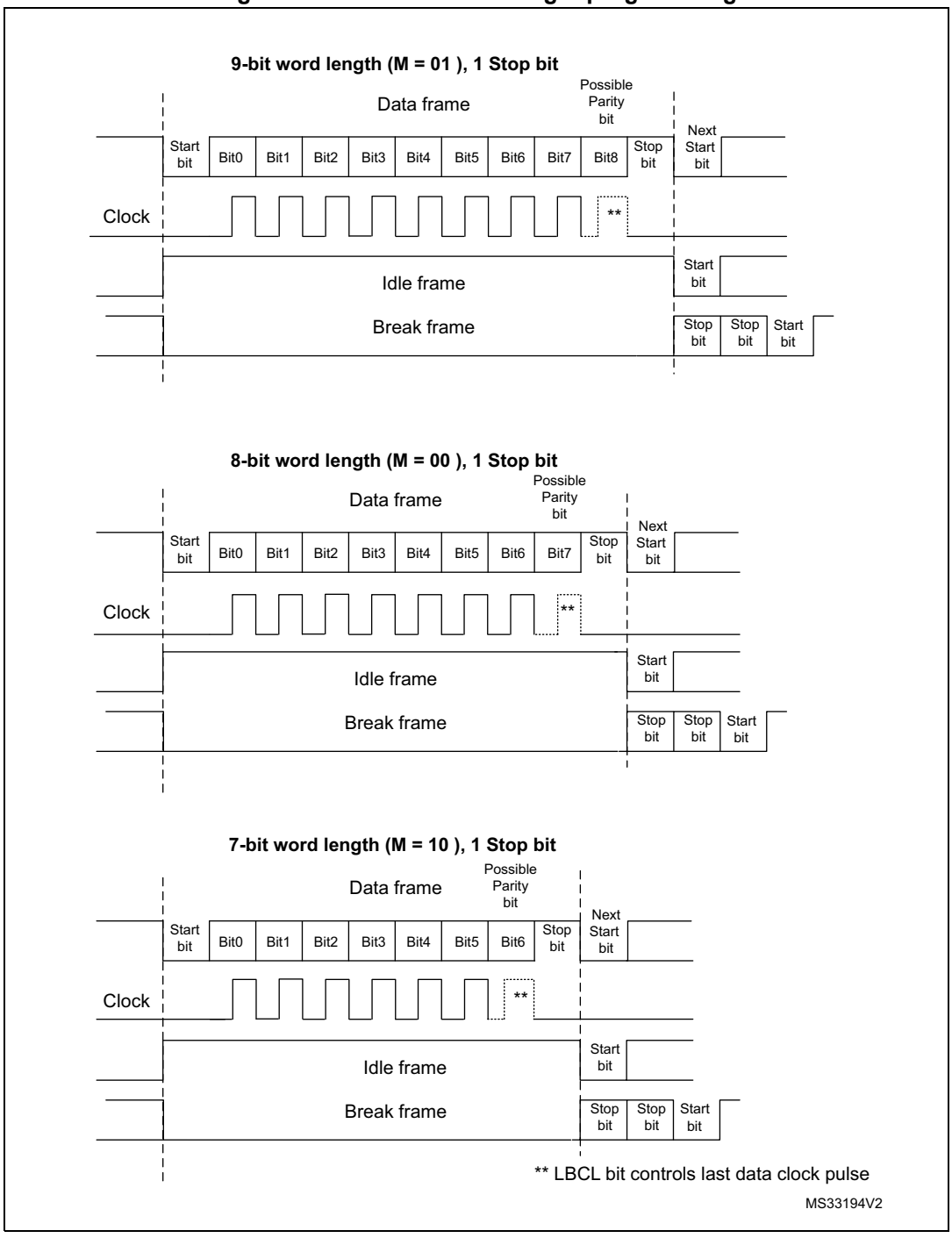
An ***Idle character*** is interpreted as an entire frame of “1”s. (The number of “1” ‘s includes the number of stop bits).

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 682. LPUART word length programming



58.4.5 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 Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bitfields.

  In this case, the RXFT flag is set in the LPUART_ISR register. This means that RXFTCFG data have been received: 1 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 Tx interrupt is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bitfields.

58.4.6 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 681*).

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

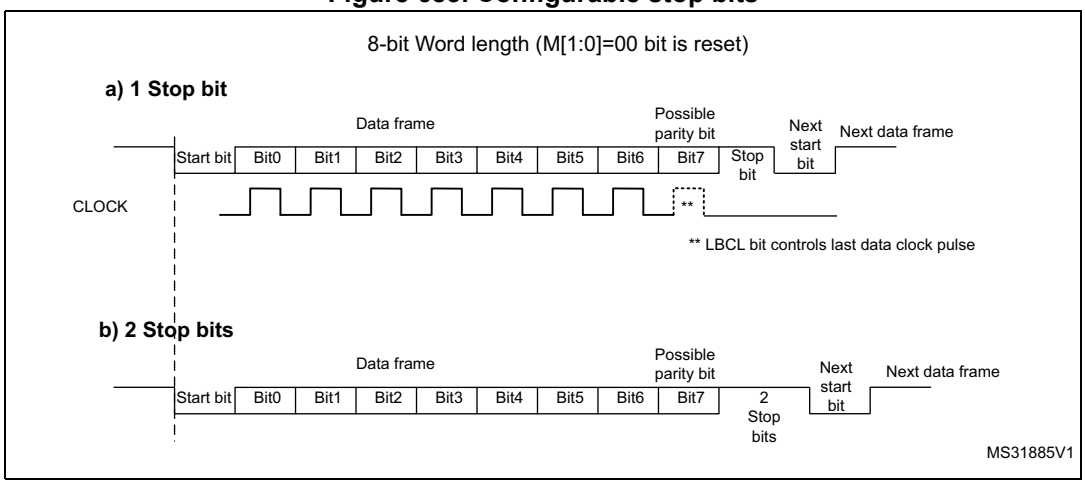
The number of stop bits to be transmitted with every character can be programmed in LPUART_CR2 (bits 13,12).

- ***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 683. 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 58.4.13: Continuous communication using DMA and LPUART*.
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.

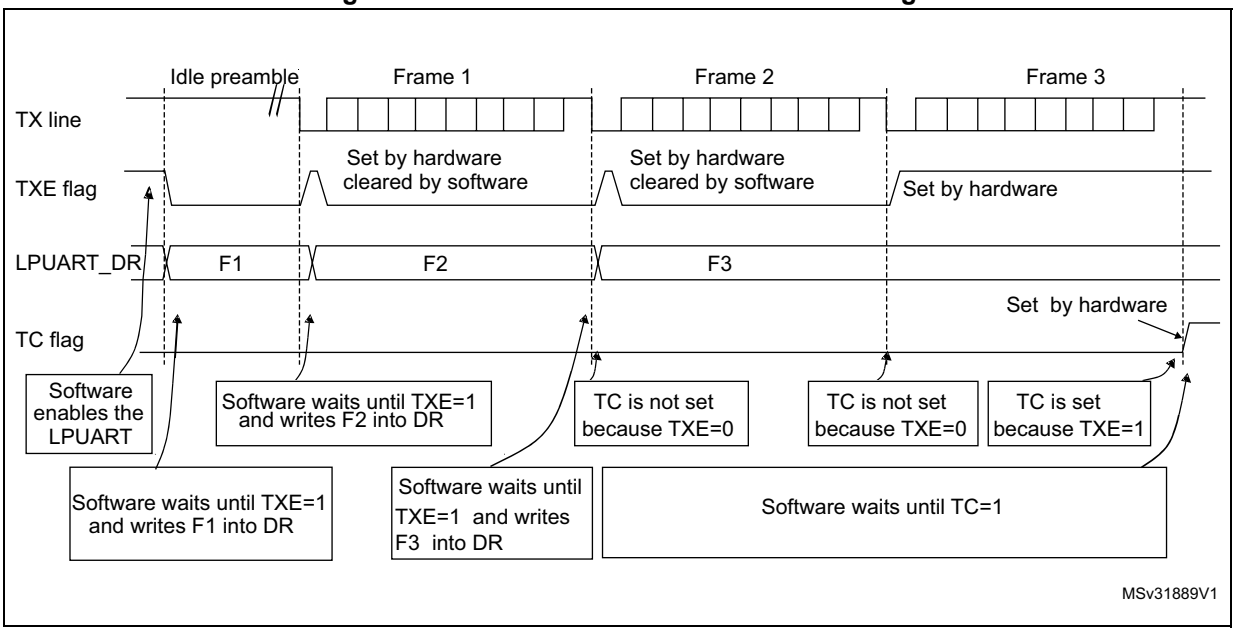
  When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write in LPUART_TDR. It is cleared when the TXFIFO is full. This flag generates an interrupt if TXFNEIE bit is set.

  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 microcontroller to enter the low-power mode (see *Figure 684: TC/TXE behavior when transmitting*).

Figure 684. 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 682*).

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.

58.4.7 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 58.4.13: Continuous communication using DMA and LPUART*.
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:

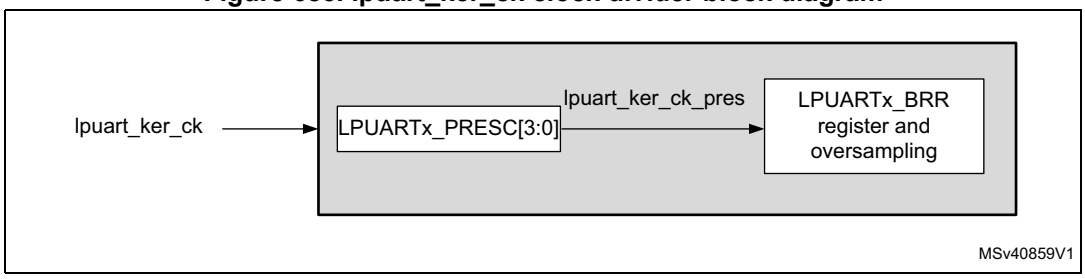
- Possible use of the LPUART in low-power mode
- Communication speed.

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 685. lpuart_ker_ck clock divider block diagram



Some lpuart_ker_ck sources enable the LPUART to receive data while the MCU is in low-power mode. Depending on the received data and Wakeup mode selection, the LPUART wakes up the MCU, when needed, in order to transfer the received data by software reading the LPUART_RDR register or by DMA.

For the other clock sources, the system must be active to enable LPUART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver samples each incoming bit as close as possible to the middle of the bit-period. Only a single sample is taken of each of the incoming bits.

Note: There is no noise detection for data.

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- the FE bit is set by hardware;
- the invalid data is transferred from the Shift register to the LPUART_RDR register.
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of

multibuffer communication, an interrupt is issued if the EIE bit is set in the LPUART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the LPUART_ICR register.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of LPUART_CR2: it can be either 1 or 2 in Normal mode.

- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- **2 stop bits**: sampling for the 2 stop bits is done in the middle of the second stop bit. The RXNE and FE flags are set just after this sample i.e. during the second stop bit. The first stop bit is not checked for framing error.

58.4.8 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.

$$\text{Tx/Rx baud} = \frac{256 \times \text{lpuart\_ker\_ck\_pres}}{\text{LPUARTDIV}}$$

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 bauds. 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 Mbauds.

Table 572. 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 kbaud	0.3 kbaud	0x6D3A	0
2	0.6 kbaud	0.6 kbaud	0x369D	0
3	1200 bauds	1200.087 bauds	0x1B4E	0.007
4	2400 bauds	2400.17 bauds	0xDA7	0.007
5	4800 bauds	4801.72 bauds	0x6D3	0.035
6	9600 kbauds	9608.94 bauds	0x369	0.093

Table 573. 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 bauds	38400,04 bauds	A2C2A	0,0001
2	57600 bauds	57600,06 bauds	6C81C	0,0001
3	115200 bauds	115200,12 bauds	3640E	0,0001
4	230400 bauds	230400,23 bauds	1B207	0,0001
5	460800 bauds	460804,61 bauds	D903	0,001
6	921600 bauds	921625,81 bauds	6C81	0,0028
7	4000 kbauds	4000000,00 bauds	1900	0
8	10000 kbauds	10000000,00 bauds	A00	0
9	20000 kbauds	20000000,00 bauds	500	0
10	30000 kbauds	33032258,06 bauds	307	0,1

58.4.9 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)

$$\text{DTRA} + \text{DQUANT} + \text{DREC} + \text{DTCL} + \text{DWU} < \text{LPUART receiver tolerance}$$

where

DWU is the error due to sampling point deviation when the wakeup from low-power mode is used.

when M[1:0] = 01:

$$\text{DWU} = \frac{t_{\text{WULPUART}}}{11 \times \text{Tbit}}$$

when M[1:0] = 00:

$$\text{DWU} = \frac{t_{\text{WULPUART}}}{10 \times \text{Tbit}}$$

when M[1:0] = 10:

$$\text{DWU} = \frac{t_{\text{WULPUART}}}{9 \times \text{Tbit}}$$

t_{WULPUART} 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 LPUART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 574*:

- Number of Stop bits defined through STOP[1:0] bits in the LPUART_CR2 register
- LPUART_BRR register value.

Table 574. 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 574 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).

58.4.10 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:

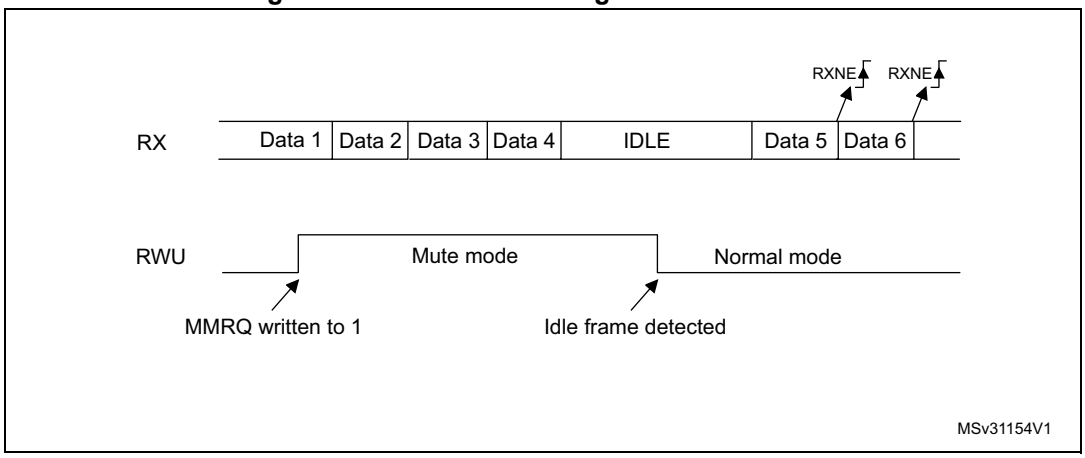
- Idle Line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

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 686*.

Figure 686. 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 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.

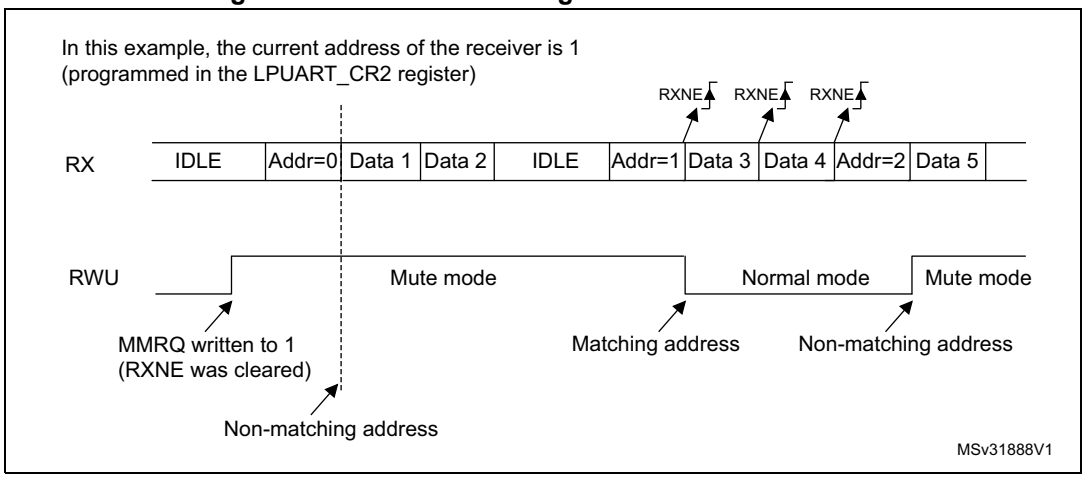
The LPUART also enters Mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

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 687*.

Figure 687. Mute mode using address mark detection



58.4.11 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 575*.

Table 575: 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=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=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)).

58.4.12 LPUART single-wire Half-duplex communication

Single-wire Half-duplex mode is selected by setting the HDSEL bit 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.

58.4.13 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 57.4: USART implementation on page 2256 to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in Section 57.5.7. To perform continuous communication. When FIFO is disabled, 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 Section *Direct memory access controller*) 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):

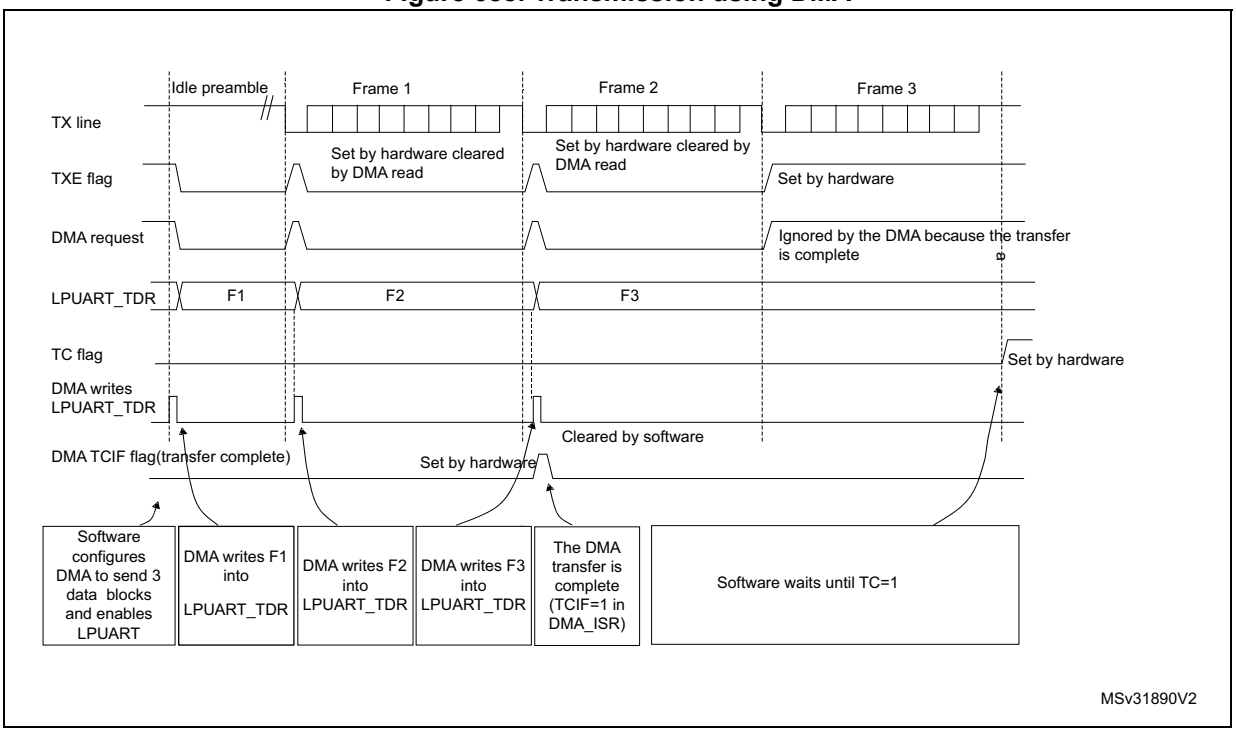
1. Write the LPUART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the 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.

Note: The DMAT bit should not be cleared before the DMA end of transfer.

Figure 688. 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

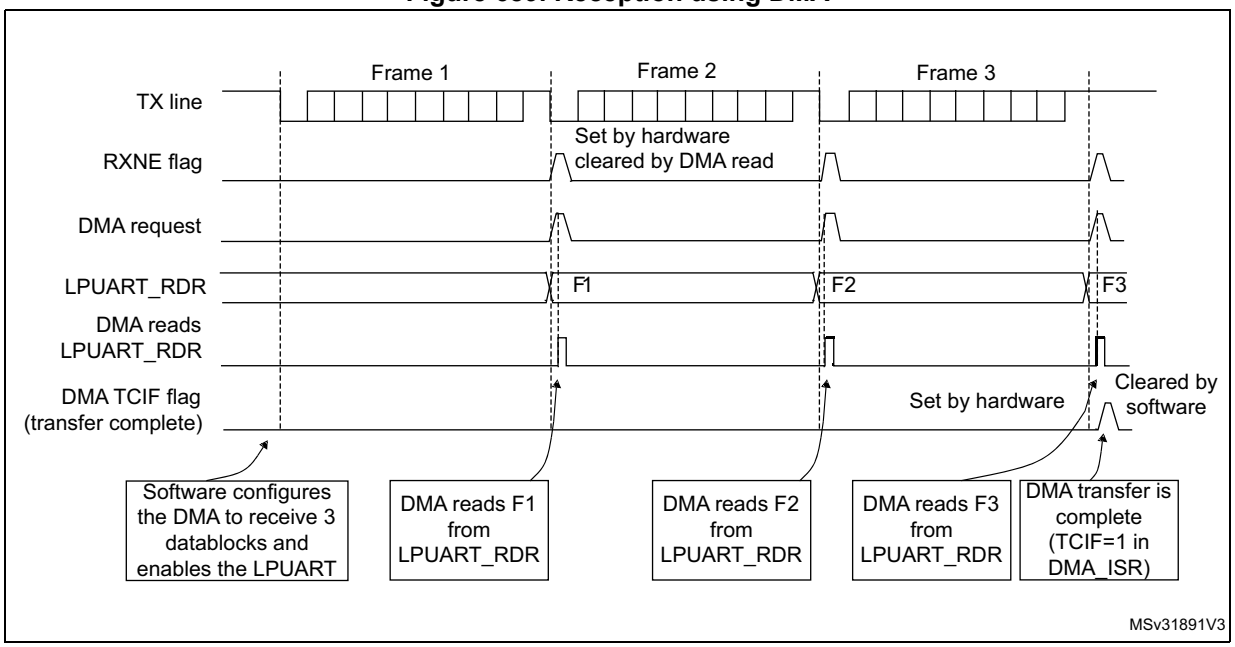
DMA mode can be enabled for reception by setting the DMAR bit in LPUART_CR3 register. Data are loaded from the LPUART_RDR register to a SRAM area configured using the DMA peripheral (refer to section *Direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for LPUART 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.

Note: The DMAR bit should not be cleared before the DMA end of transfer.

Figure 689. 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

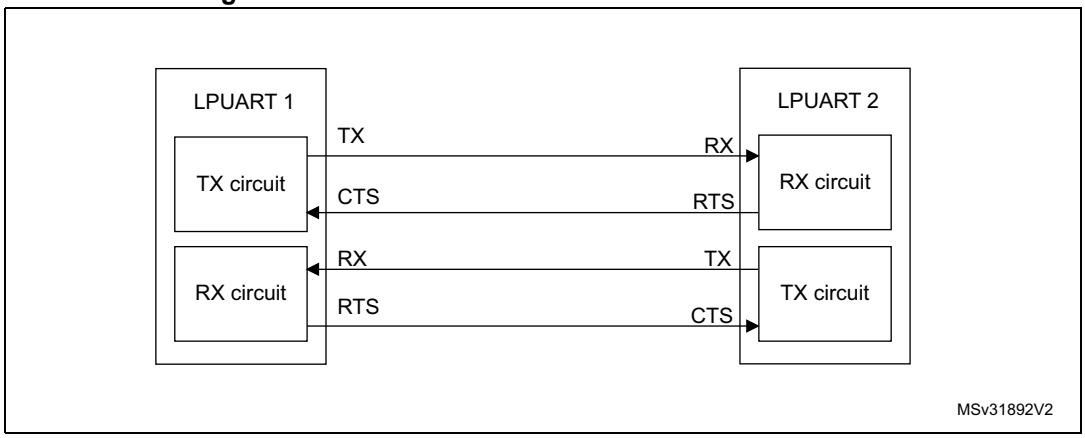
If any error occurs during a transaction In Multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) 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.

58.4.14 RS232 Hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the CTS input and the RTS output. The *Figure 690* shows how to connect 2 devices in this mode:

Figure 690. Hardware flow control between 2 LPUARTs

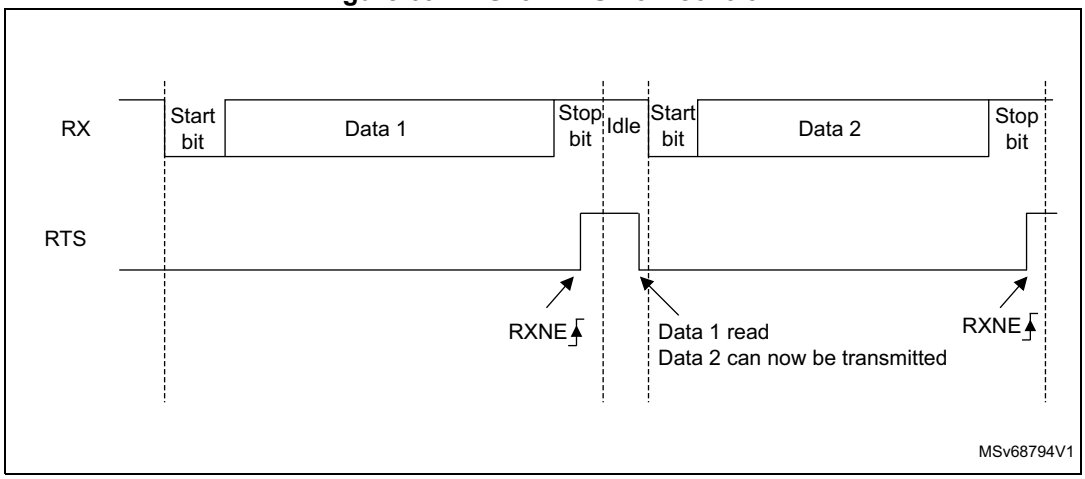


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART_CR3 register).

RS232 RTS flow control

If the RTS flow control is enabled (RTSE=1), then RTS is deasserted (tied low) as long as the LPUART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 691* shows an example of communication with RTS flow control enabled.

Figure 691. RS232 RTS flow control



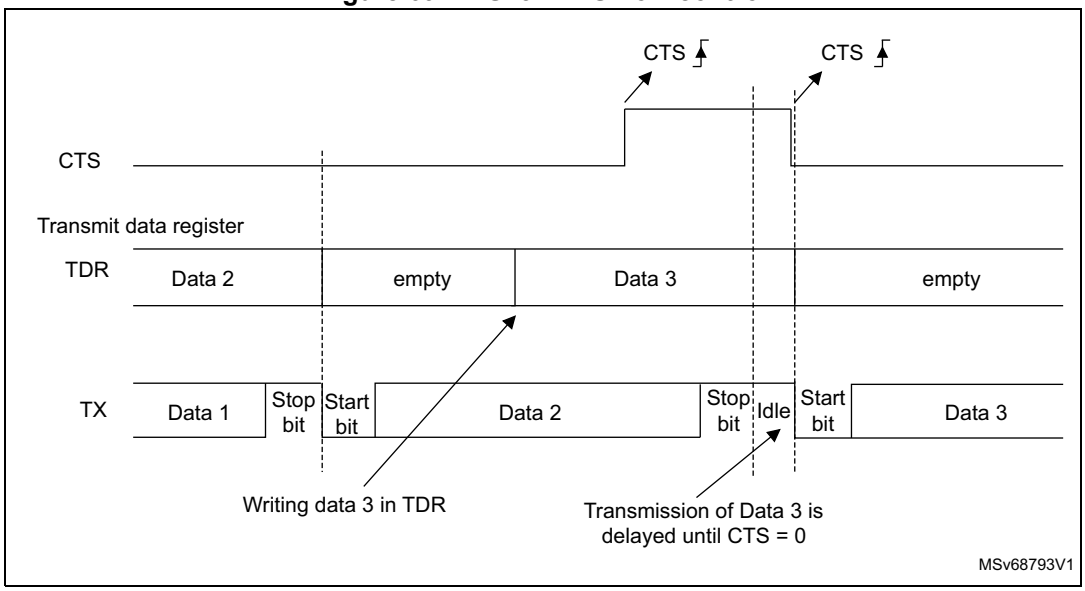
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE=0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the LPUART_CR3 register is set. *Figure 692* shows an example of communication with CTS flow control enabled.

Figure 692. RS232 CTS flow control



Note: For correct behavior, CTS must be deasserted at least 3 LPUART clock source periods before the end of the current character. In addition it should be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the LPUART_CR3 control register. This enables activating the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the LPUART_CR1 control register. The de-assertion 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 de-assertion 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]

58.4.15 LPUART Autonomous mode

The LPUART peripheral can be functional in Stop mode thanks to the Autonomous mode. This mode can also be used in Run and Sleep mode. The UESM bit must be set prior to entering low-power mode.

The APB clock is requested by the peripheral each time the LPUART status needs to be updated. Once the LPUART receives the kernel and APB clocks, it generates either an interrupt or a DMA request, depending on the peripheral configuration.

If an interrupt is generated, the device wakes up from Stop mode. If no interrupt is generated, the device remains in Stop mode but the APB clock is still available for the LPUART and all the autonomous peripherals enabled in the reset and clock controller (RCC). If DMA requests are enabled, the data are directly transferred to/from the SRAM thanks to the DMA while the product remains in Stop mode.

Transmission mode

In transmission, the APB clock is requested only when the TE bit is set and in the following cases:

- If the FIFO mode is enabled, the APB clock is requested when
  - the TxFIFO is empty (TXFE = 1) and the corresponding interrupt is enabled (TXFEIE = 1)
  - the TxFIFO threshold is reached (TXFT = 1) and the corresponding interrupt is enabled (TXFTIE = 1)
  - the TxFIFO is not full (TXFNF = 1) and the corresponding interrupt or DMA is enabled (TXFNFIE = 1 or DMAT = 1)
- If the FIFO mode is disabled, the APB clock is requested as soon as data are transferred to the shift register. The DMA or associated interrupt must be enabled.

The TE bit is set by hardware if an asynchronous trigger is detected.

A transmission is automatically launched when an asynchronous trigger is detected in Run, Sleep or Stop mode. The trigger is selected through the TRIGSEL bit in the LPUART_AUTOCR register. It sets the TE bit in the LPUART_CR1 register and generates an APB clock request to enable the transfer. The APB clock is requested until the transmission completes and the TE bit is cleared by hardware when the programmed number of data to be transmitted (TDN bits field in the LPUART_AUTOCR register) is reached. In this case, the TC flag is set when the number of data to be transmitted is reached and the last byte is transmitted.

Reception mode

- If the FIFO mode is enabled, the APB clock is requested when
  - the RxFIFIO is full (RXFF = 1) and the corresponding interrupt is enabled (RXFFIE = 1)
  - the RxFIFO threshold is reached (RXFT = 1) and the corresponding interrupt is enabled (RXFTIE = 1)
  - the RxFIFO is not empty (RXFNE = 1) and the corresponding interrupt or DMA is enabled (RXFNEIE= 1)
- If the FIFO mode is disabled, the APB clock is requested when the LPUART finishes sampling data and it is ready to be written in the LPUART_RDR. The DMA or the associated interrupt must be enabled.

Note: The APB clock is requested in Reception mode when an overrun error occurs (ORE = 1). The EIE bit must be set to enable the generation an interrupt and waking up the MCU, and the OVRDIS bit must remain cleared. The APB clock request is kept until the interrupt flag is cleared.

In Reception mode, the APB clock is requested when a Parity/Noise/Framing error occurs and the DMA is used for reception. The APB clock request is kept until the interrupt flag is cleared.

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 58.4.9: 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 574: 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 kbauds.

58.5 LPUART in low-power modes

Table 576. 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. If the LPUART is clocked by an oscillator available in Stop mode, transfers in Asynchronous mode are functional. DMA requests are functional, and the interrupts cause the device to exit Stop mode.
Standby	The LPUART peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 58.3: 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.

58.6 LPUART interrupts

Refer to *Table 577* for a detailed description of all LPUART interrupt requests.

Table 577. 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	Yes	No
	Transmit FIFO Not Full	TXFNF	TXFNFIE	TXFIFO full		Yes	
	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		Yes	
	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		Yes	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		Yes(3)	
	Noise error in multibuffer communication.	NE	EIE	Write 1 in NFCF		Yes(3)	
	Overrun error in multibuffer communication.	ORE(4)		Write 1 in ORECF		Yes	
	Framing Error in multibuffer communication.	FE		Write 1 in FECF		Yes(3)	
	Character match	CMF	CMIE	Write 1 in CMCF		Yes(5)	

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 58.3: 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. Parity/Noise/Framing error interrupts enable waking up from Stop modes when the DMA is used.
4. When OVRDIS = 0.
5. The DMA must be used when the FIFO mode is enabled.

58.7 LPUART registers

Refer to *Section 1.2 on page 104* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

58.7.1 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 57.5.21: 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 58.4.14: 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 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 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. When the Autonomous mode is not used, TE bit is set and cleared by software. When the Autonomous mode is used, TE bit becomes a status bit, which is set and cleared by hardware.

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**: LPUART enable in low-power mode

When this bit is cleared, the LPUART cannot request its kernel clock and is not functional in low-power mode.

When this bit is set, the LPUART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: LPUART not functional in low-power mode.

1: LPUART functional in low-power mode.

Note: The UESM bit must be set at the initialization phase.

If the LPUART *does not support the Wakeup from low-power mode, this bit is reserved and must be kept at reset value. Refer to Section 58.3: LPUART implementation on page 2343.*

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.

58.7.2 LPUART control register 1 [alternate] (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	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 57.5.21: 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 58.4.14: 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 =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=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. When the Autonomous mode is disabled, TE bit is set and cleared by software. When the Autonomous mode is enabled, TE bit becomes a status bit, which is set and cleared by hardware.

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 LPUART_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**: LPUART enable in low-power mode

When this bit is cleared, the LPUART cannot request its kernel clock and is not functional in low-power mode.

When this bit is set, the LPUART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: LPUART not functional in low-power mode.

1: LPUART functional in low-power mode.

Note: The UESM bit must be set at the initialization phase.

If the LPUART does not support the Wakeup from low-power mode, this bit is reserved and must be kept at reset value. Refer to Section 58.3: LPUART implementation on page 2343.

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.

58.7.3 LPUART control register 2 (LPUART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	Res.	STOP[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
rw		rw	rw								rw				

Bits 31:24 **ADD[7:0]**: Address of the LPUART node

These bits give the address of the LPUART node in Mute mode or a character code to be recognized in low-power or Run mode:

- In Mute mode: they are used in multiprocessor communication to wakeup from Mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When a character, received during low-power mode, corresponds to the character programmed through ADD[7:0] bitfield, the CMF flag is set and wakes up the device from low-power mode if the corresponding interrupt is enabled by setting CMIE bit.
- In Run mode with Mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the reception is disabled (RE = 0) or when the USART is disabled (UE = 0).

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.

1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)

1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd=0/mark)

1: TX pin signal values are inverted. (V_{DD} =0/mark, Gnd=1/idle).

This enables the use of an external inverter on the TX line.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 16 **RXINV:** RX pin active level inversion
This bit is set and cleared by software.
0: RX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd=0/mark)
1: RX pin signal values are inverted. ((V_{DD} =0/mark, Gnd=1/idle).
This enables the use of an external inverter on the RX line.
This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 15 **SWAP:** Swap TX/RX pins
This bit is set and cleared by software.
0: TX/RX pins are used as defined in standard pinout
1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.
This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **STOP[1:0]**: STOP bits
These bits are used for programming the stop bits.
00: 1 stop bit
01: Reserved.
10: 2 stop bits
11: Reserved
This bitfield can only be written when the LPUART is disabled (UE=0).

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 **ADDM7**:7-bit Address Detection/4-bit Address Detection
This bit is for selection between 4-bit address detection or 7-bit address detection.
0: 4-bit address detection
1: 7-bit address detection (in 8-bit data mode)
This bit can only be written when the LPUART is disabled (UE=0)
Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bits 3:0 Reserved, must be kept at reset value.

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

Bits 22:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE=0).

Bit 14 **DEM**: Driver enable mode
This bit enables the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE=0).

Bit 13 **DDRE**: DMA Disable on Reception Error
0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.
1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.
This bit can only be written when the LPUART is disabled (UE=0).
Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable
This bit is used to disable the receive overrun detection.
0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.
This bit can only be written when the LPUART is disabled (UE=0).
Note: This control bit enables *checking the communication flow w/o reading the data.*

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable
0: Interrupt is inhibited
1: An interrupt is generated whenever CTSIF=1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable
0: CTS hardware flow control disabled
1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.
This bit can only be written when the LPUART is disabled (UE=0)

Bit 8 **RTSE**: RTS enable
0: RTS hardware flow control disabled
1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.
This bit can only be written when the LPUART is disabled (UE=0).

Bit 7 **DMAT**: DMA enable transmitter
This bit is set/reset by software
1: DMA mode is enabled for transmission
0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver
This bit is set/reset by software
1: DMA mode is enabled for reception
0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection
Selection of Single-wire Half-duplex mode
0: Half-duplex mode is not selected
1: Half-duplex mode is selected
This bit can only be written when the LPUART is disabled (UE=0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE=1 or ORE=1 or NE=1 in the LPUART_ISR register).
0: Interrupt is inhibited
1: An interrupt is generated when FE=1 or ORE=1 or NE=1 in the LPUART_ISR register.

58.7.5 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 division (LPUARTDIV)

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].

58.7.6 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.

58.7.7 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	Res.	RWU	SBKF	CMF	BUSY
				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 Reserved, must be 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: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.

An interrupt is generated if CMIE=1in the LPUART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: LPUART is idle (no reception)

1: Reception on going

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

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE=1 in the LPUART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXFNF:** TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full, and so data can be written in the LPUART_TDR. Every write in the LPUART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the LPUART_TDR.

The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF should be checked prior to writing in TXFIFO (TXFNF and TXFE are set at the same time).

An interrupt is generated if the TXFNFIE bit =1 in the LPUART_CR1 register.

0: Data register is full/Transmit FIFO is full.

1: Data register/Transmit FIFO is not full.

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the LPUART_TDR has been transmitted out of the shift register. The TC flag behaves as follows:

- When TDN = 0, the TC flag is set when the transmission of a frame containing data is complete and when TXFE is set.
- When TDN is equal to the number of data in the TXFIFO, the TC flag is set when TXFIFO is empty and TDN is reached.
- When TDN is greater than the number of data in the TXFIFO, TC remains cleared until the TXFIFO is filled again to reach the programmed number of data to be transferred.
- When TDN is less than the number of data in the TXFIFO, TC is set when TDN is reached even if the TXFIFO is not empty.

An interrupt is generated if TCIE=1 in the LPUART_CR1 register.

TC bit is cleared by software by writing 1 to the TCCF in the LPUART_ICR register or by writing to the LPUART_TDR register.

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 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.

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 NFCF bit in the LPUART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

This error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.

When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the LPUART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: This error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in Reception 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.

58.7.8 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	Res.	RWU	SBKF	CMF	BUSY
									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 Reserved, must be 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: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.

An interrupt is generated if CMIE=1in the LPUART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: LPUART is idle (no reception)

1: Reception on going

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

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE=1 in the LPUART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXE:** Transmit data register empty

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 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 indicates that the last data written in the USART_TDR has been transmitted out of the shift register. The TC flag is set when the transmission of a frame containing data is complete and when TXE is set.

An interrupt is generated if TCIE=1 in the LPUART_CR1 register.

TC bit is cleared by software by writing 1 to the TCCF in the USART_ICR register or by writing to the USART_TDR register.

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 a read to the LPUART_RDR register. The

RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

The RXFNE 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 (RXFF = 1 in case FIFO mode is enabled). It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXNEIE=1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.

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 NFCF 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/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.

In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.

When transmitting data in Smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the LPUART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: In FIFO mode, 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 Reception 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: In FIFO mode, this error is associated with the character in the LPUART_RDR.

58.7.9 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.	Res.	Res.	Res.	CMCF	Res.
														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 31:18 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.

Bit 8 Reserved, must be kept at reset value.

Bit 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.

58.7.10 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 681*).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

58.7.11 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 681*).

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.

58.7.12 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 equal to 1011 i.e. input clock divided by 256.

58.7.13 LPUART Autonomous mode control register (LPUART_AUTOCR)

Address offset: 0x30

Reset value: 0x8000 0000

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

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

Bits 22:19 **TRIGSEL[3:0]**: Trigger selection bits

Refer to *Section : Description LPUART interconnections*.
This bitfield can be written only when the UE bit is cleared in LPUART_CR1 register.
0000: lpuart_trg0 selected
0001: lpuart_trg1 selected
...
1111: lpuart_trg15 selected

Note: This bitfield can be written only when the UE bit of LPUART_CR1 register is cleared.

Bit 18 **IDLEDIS**: Idle frame transmission disable bit after enabling the transmitter

0: Idle frame sent after enabling the transmitter (TE = 1 in LPUART_CR1)
1: Idle frame not sent after enabling the transmitter

Note: This bitfield can be written only when the UE bit of LPUART_CR1 register is cleared.

Bit 17 **TRIGEN**: Trigger enable bit

0: Trigger disabled

1: Trigger enabled

Note: This bitfield can be written only when the UE bit of LPUART_CR1 register is cleared. When a trigger is detected, TE is set to 1 in LPUART_CR1 and the data transfer is launched.

Bit 16 **TRIGPOL**: Trigger polarity bit

This bitfield can be written only when the UE bit is cleared in LPUART_CR1 register.

0: Trigger active on rising edge

1: Trigger active on falling edge

Bits 15:0 **TDN[15:0]**: TDC transmission data number

This bitfield enables the programming of the number of data to be transmitted. It can be written only when UE is cleared in LPUART_CR1.

58.7.14 LPUART register map

Table 578. 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	Res.	Res.	Res.	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
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.	Res.	Res.	Res.	RXFF	TXFF	REACK	TEACK	Res.	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
	Reset value								0	1	0	0		0	0	0	0						0	0		1	0	0	0	0	0	0	0

Table 578. 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
0x1C	LPUART_ISR FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	Res.	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		1	0	0	0	0	0	0	0
0x20	LPUART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
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
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
0x30	LPUART_ AUTOCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGSEL[3:0]			IDLEDIS	TRIGEN	TRIGPOL	TDN[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

Refer to *Section 2.3* for the register boundary addresses.

59 Serial peripheral interface (SPI)

59.1 Introduction

The serial peripheral interface (SPI) can be used to communicate with external devices while using the specific synchronous protocol. The SPI protocol supports half-duplex, full-duplex and simplex synchronous, serial communication with external devices. The interface can be configured as master or slave and is capable of operating in multi slave or multi master configurations. The device configured as master provides communication clock (SCK) to the slave device. The Slave select (SS) and ready (RDY) signals can be applied optionally just to setup communication with concrete slave and to assure it handles the data flow properly. The Motorola data format is used by default, but some other specific modes are supported as well.

59.2 SPI main features

- 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)
- From 4- up to 32-bit data size selection or fixed to 8-bit multiples
- Multi master or multi slave mode capability
- Dual clock domain, the peripheral kernel clock is independent from the APB bus clock
- Baud rate prescaler up to kernel frequency/2 or bypass from RCC in Master mode
- Protection of configuration and setting
- Hardware or software management of SS for both master and slave
- Adjustable minimum delays between data and between SS and data flow
- Configurable SS signal polarity and timing, MISO x MOSI swap capability
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Programmable number of data within a transaction to control SS and CRC
- Dedicated transmission and reception flags with interrupt capability
- SPI Motorola and TI formats support
- Hardware CRC feature can verify integrity of the communication at the end of transaction by:
  - Adding CRC value in Tx mode
  - Automatic CRC error checking for Rx mode
- Error detection with interrupt capability in case of data overrun, CRC error, data underrun, the mode fault and frame error, depending upon the operating mode
- Two multiples of 8-bit embedded Rx and Tx FIFOs (FIFO size depends on instance)
- Configurable FIFO thresholds (data packing)
- Capability to handle data streams by system DMA controller
- Configurable behavior for slave underrun condition (support of cascaded circular buffers)

- Autonomous functionality in Stop modes (handling of the transaction flow and required clock distribution) with wakeup from Stop capability
- Optional status pin RDY signalizing the slave device ready to handle the data flow

59.3 SPI implementation

Table 579 describes the SPI implementation. The instances are applied either with a full set or a limited set of features.

Table 579. SPI features

SPI feature	SPI1, SPI2 (full feature set instances)	SPI3 (limited feature set instance)
Data size	Configurable from 4 to 32 bits	8/16 bits
CRC computation	CRC polynomial length configurable from 5 to 33 bits	CRC polynomial length 9/17 bits
Size of FIFOs	16 x 8 bits	8 x 8 bits
Number of data control	Up to 65536	Up to 1024, no remaining data counter (CTSIZE)
Autonomous in Stop mode with wakeup capability	Yes	Yes
Autonomous in Standby mode with wakeup capability	No	No

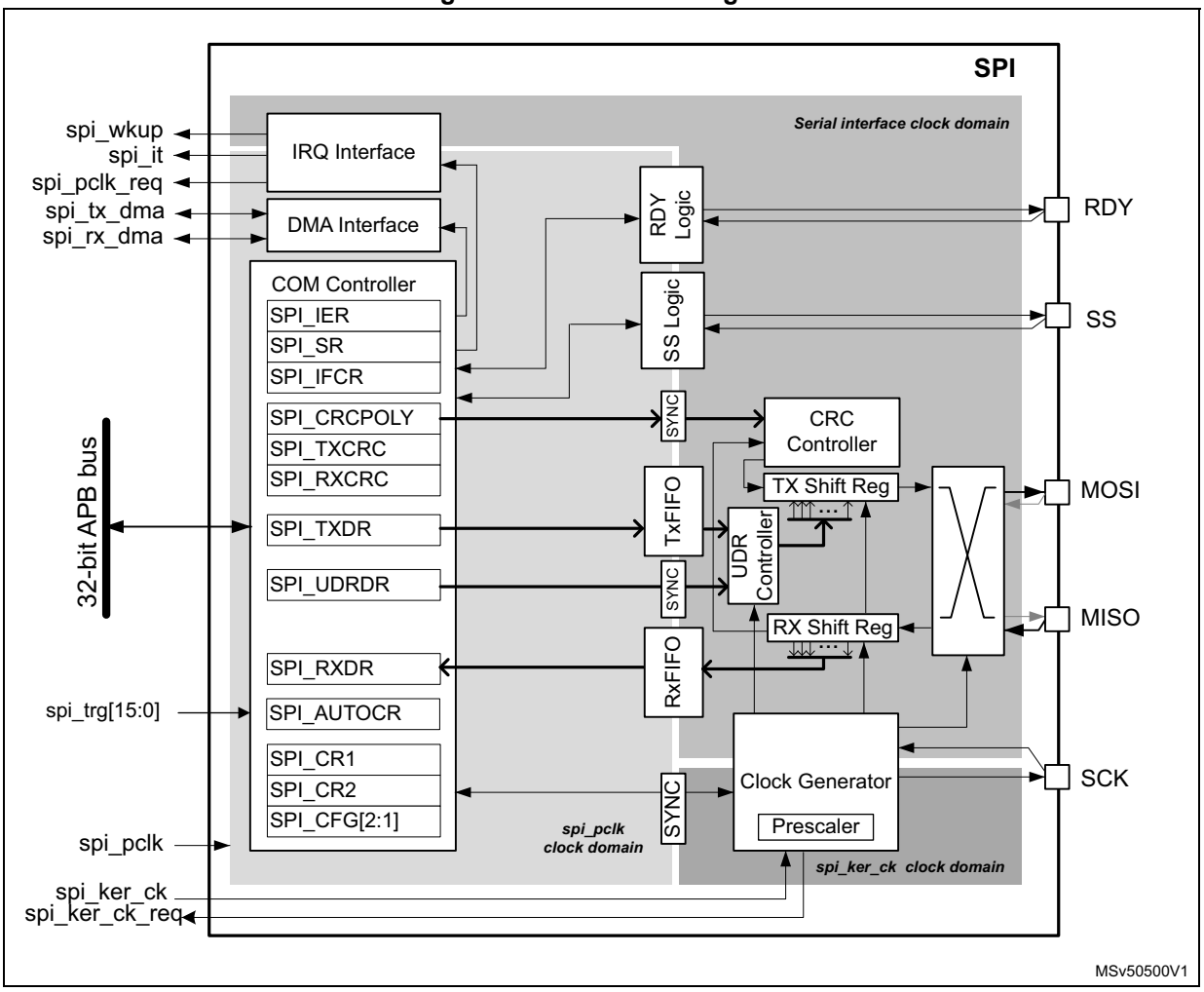
Note: For detailed information about instances capabilities to exit from Stop and Standby modes refer to Table 585: SPI wakeup and interrupt requests and Table 89: Functionalities depending on the working mode.

59.4 SPI functional description

59.4.1 SPI block diagram

The SPI enables a synchronous, serial communication between the MCU and external devices. The application software can manage the communication by polling the status flag or using a dedicated SPI interrupt. The main SPI elements and their interactions are shown in *Figure 693*.

Figure 693. SPI block diagram



The simplified scheme of *Figure 693* shows three fully independent clock domains:

- the **spi_pclk** clock domain
- the **spi_ker_ck** kernel clock domain
- the serial interface clock domain

All the control and status signals between these domains are strictly synchronized. There is no specific constraint concerning the frequency ratio between these clock signals. The user has to consider a ratio compatible with the data flow speed to avoid data underrun or overrun events.

The **spi_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the SPI registers are required.

The SPI working in Slave mode handles a data flow using the serial interface clock derived from the external SCK signal provided by the external master SPI device. This is why the SPI slave is able to receive and send data even when the **spi_pclk** and **spi_ker_ck** clock signals are inactive. As a consequence, a specific slave logic working within the serial interface clock domain needs some additional traffic to be setup correctly (e.g. when underrun or overrun is evaluated, see *Section 59.5.2* for details). This cannot be done when the bus becomes idle. In some cases the slave even requires the clock generator working (see *Section 59.5.1*).

When the SPI works as master, it needs **spi_ker_ck** kernel clock coming from RCC active during communication to feed the serial interface clock via the clock generator where it can be divided by prescaler or bypassed optionally. The signal is then provided to slaves via the SCK pin and internally to the serial interface domain of the master.

59.4.2 SPI pins and internal signals

Up to five 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.
- **SS:** slave select pin. Depending on the SPI and SS 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 59.4.7* for details.
- **RDY:** optional status pin signaling slave FIFO occupancies and so the slave availability to continue the transaction without any risk of data flow corruption. It can be checked by master to control temporal suspension of the ongoing communication.

The SPI bus enables 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 are optional and can be added depending on the data exchange between SPI nodes and their communication control management.

Refer to *Table 580* and *Table 581* for the list of SPI input / output pins and internal signals.

Table 580. SPI/ input/output pins

Pin name	I/O type	Description
MISO(1)	Input/output	Master data input / slave data output
MOSI(1)	Input/output	Master data output / slave data input
SCK	Input/output	Master clock output / slave clock input
SS	Input/output	Master output / slave selection input
RDY	Input/output	SPI master input / slave FIFOs status occupancy output

1. Functionality of MOSI and MISO pins can be swapped. Their directions may vary in SPI bidirectional half duplex mode.

Description of SPI input/output signals

Table 581. SPI internal input/output signals

Signal name	Signal type	Description
spi_pclk	Input	SPI clock signal feeds the peripheral bus interface
spi_ker_ck	Input	SPI kernel clock
spi_ker_ck_req	Output	SPI kernel clock request
spi_pclk_req	Output	SPI clock request
spi_wkup	Output	SPI provides a wakeup interrupt
spi_it	Output	SPI global interrupt
spi_tx_dma	Input/output	SPI transmit DMA request
spi_rx_dma	Input/output	SPI receive DMA request
spi_trg[15:0]	Input	SPI triggers

Description of SPI interconnections

Table 582. SPI interconnection (SPI1 and SPI2)

Signal name	Trigger source
spi_trg0	gpdma1_ch0_tc
spi_trg1	gpdma1_ch1_tc
spi_trg2	gpdma1_ch2_tc
spi_trg3	gpdma1_ch3_tc
spi_trg4	exti4
spi_trg5	exti9
spi_trg6	lptim1_ch1
spi_trg7	lptim2_ch1
spi_trg8	comp1_out
spi_trg9	comp2_out
spi_trg10	rtc_alra_trg
spi_trg11	rtc_wut_trg
spi_trg12	-
spi_trg13	-
spi_trg14	-
spi_trg15	-

Table 583. SPI interconnection (SPI3)

Signal name	Trigger source
spi_trg0	lpdma1_ch0_tc
spi_trg1	lpdma1_ch1_tc
spi_trg2	lpdma1_ch2_tc
spi_trg3	lpdma1_ch3_tc
spi_trg4	exti4
spi_trg5	exti8
spi_trg6	lptim1_ch1
spi_trg7	lptim3_ch1
spi_trg8	comp1_out
spi_trg9	comp2_out
spi_trg10	rtc_alra_trg
spi_trg11	rtc_wut_trg
spi_trg12	-
spi_trg13	-
spi_trg14	-
spi_trg15	-

59.4.3 SPI communication general aspects

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 SS management) or 3 or 4 wires (with hardware SS management). The communication is always initiated and controlled by the master. The master provides a clock signal on the SCK line and selects or synchronizes slave(s) for communication by SS line when it is managed by hardware. The data between the master and the slave flow on the MOSI and/or MISO lines.

59.4.4 Communications between one master and one slave

The communication flow may use one of 3 possible modes: the full-duplex (3 wires) mode, half-duplex (2 wires) mode or the simplex (2 wires) mode. The SS signal is optional in single master-slave configuration and is often not connected between the two communication nodes. Nevertheless, the SS signal can be helpful in this configuration to synchronize the data flow and it is used by default for some specific SPI modes (for example the TI mode).

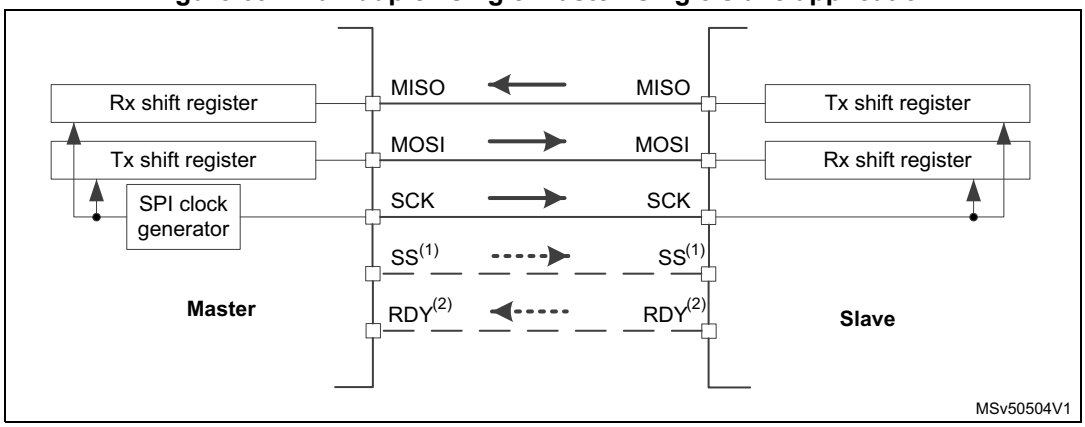
The next optional RDY signal can help to assure correct management of all the transacted data at slave side.

Full-duplex communication

By default, the SPI is configured for full-duplex communication (bits COMM[1:0] = 00 in the SPI_CFG2 register). 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 the SPI

communication, the data are 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 simultaneously. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

Figure 694. Full-duplex single master/ single slave application

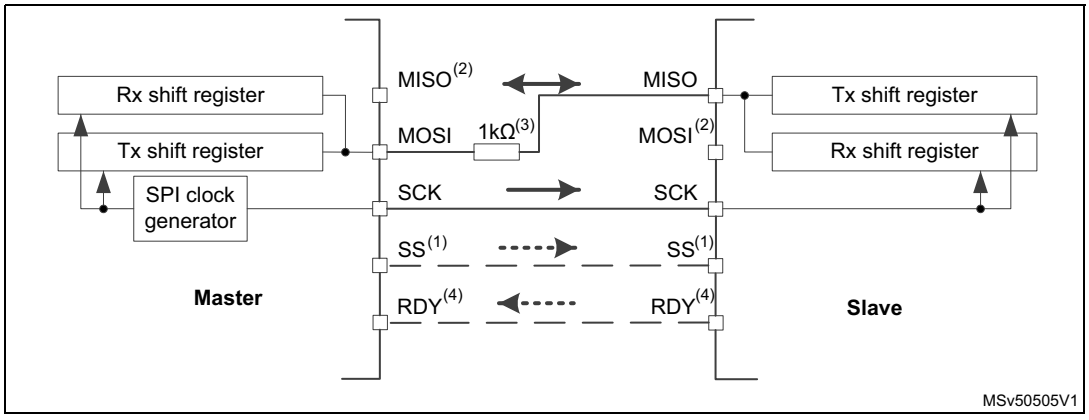


1. To apply SS pins interconnection is not mandatory to make the SPI interface working (see *Section 59.4.7* for details).
2. RDY signal provided by slave can be read by master optionally.

Half-duplex communication

The SPI can communicate in half-duplex mode by setting COMM[1:0] = 11 in the SPI_CFG2 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 are synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the HDDIR bit in their SPI_CR1 registers. Note that the SPI must be disabled when changing direction of the communication. In this configuration, the MISO pin at master and the MOSI pin at slave are free for other application uses and act as GPIOs.

Figure 695. Half-duplex single master/ single slave application



1. To apply SS pins interconnection is not mandatory to make the SPI interface working (see *Section 59.4.7* for details).
2. In this configuration, the MISO pin at master and MOSI pin at slave can be used as GPIOs
3. A critical situation can happen when the communication direction is not changed synchronously between

two nodes working in bidirectional mode. The new transmitter accesses the common data line while the former transmitter still keeps an opposite value on the line (the value depends on the SPI configuration and communicated data). The nodes can conflict temporary with opposite output levels on the line until the former transmitter changes its data direction setting. It is suggested to insert a serial resistance between MISO and MOSI pins in this mode to protect the conflicting outputs and limit the current flow between them.

4. RDY signal provided by slave can be read by master optionally.

Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the COMM[1:0] field in the SPI_CFG2 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO or MOSI pins pair is not used for communication and can be used as standard GPIOs.

- **Transmit-only mode: COMM[1:0] = 01**

  The master in transmit-only mode generates the clock as long as there are data available in the TxFIFO and the master transfer is ongoing.

  The slave in transmit-only mode sends data as long as it receives a clock on the SCK pin and the SS pin (or software managed internal signal) is active (see *Section 59.4.7*).

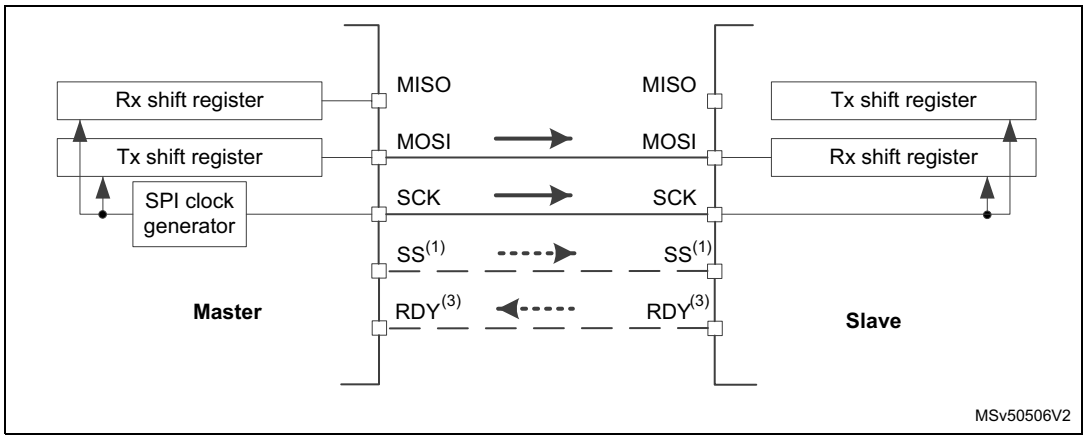
- **Receive-only mode: COMM[1:0] = 10**

  In Master mode, the MOSI output is disabled and may be used as GPIO. The clock signal is generated continuously as long as the SPI is enabled and the CSTART bit in the SPI_CR1 register is set. The clock is stopped either by software explicitly requesting this by setting the CSUSP bit in the SPI_CR1 register or automatically when the RxFIFO is full, when the MASRX bit in the SPI_CR1 is set.

  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 *Section 59.4.7*).

Note: In whatever Master and Slave modes, the data pin dedicated for transmission can be replaced by the data pin dedicated for reception and vice versa by changing the IOSWP bit value in the SPI_CFG2 register (This bit may only be modified when the SPI is disabled). Any simplex communication can be replaced by a variant of the half duplex communication with a constant setting of the transaction direction (bidirectional mode is enabled, while the HDDIR bit is never changed) or by full duplex control when unused data line and corresponding data flow is ignored.

Figure 696. Simplex single master / single slave application (master in transmit-only / slave in receive-only mode)

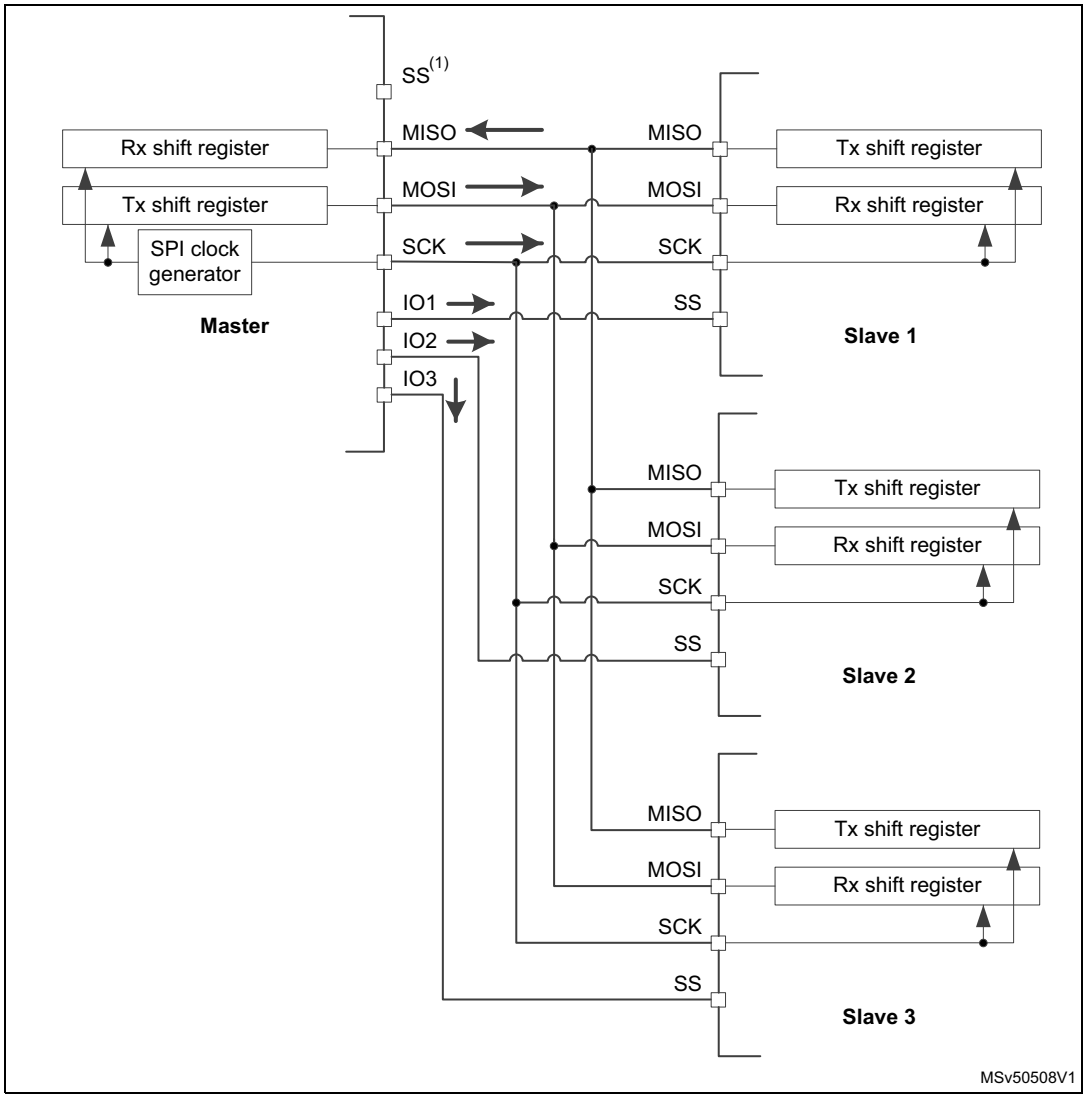


1. SS pins interconnection is not mandatory to make the SPI interface working (see *Section 59.4.7*).
2. In this configuration, both the MISO pins can be used as GPIOs.
3. RDY signal provided by slave can be read by master optionally.

59.4.5 Standard multi-slave communication

In a configuration with two or more independent slaves, the master uses a star topology with dedicated GPIO pins to manage the chip select lines for each slave separately (see *Figure 697.*). The master must select one of the slaves individually by pulling low the GPIO connected to the slave SS input (only one slave can control data on common MISO line at a given time). When this is done, a communication between the master and the selected slave is established. Except the simplicity, the advantage of this topology is that a specific SPI configuration can be applied for each slave as all the communication sessions are performed separately just within single master-slave pair. Optionally, when there is no need to read any information from slaves, the master can transmit the same information to the multiple slaves.

Figure 697. Master and three independent slaves connected in star topology



1. Master single SS pin hardware output functionality cannot support this topology (to be replaced by set of GPIOs under SW control) and user should avoid SPI AF setting at the pin (see *Section 59.4.7* for details).
2. If the application cannot assure that no more than a single SS active signal is provided by the master at a given time, it is better to configure MISO pins into open-drain configuration with an external pull up on the MISO line to prevent conflicts between interconnected outputs of the slaves. Else, a push-pull configuration can be applied without extra resistor (see I/O alternate function input/output (GPIO) section).
3. RDY signals can be read by master from the slaves optionally.

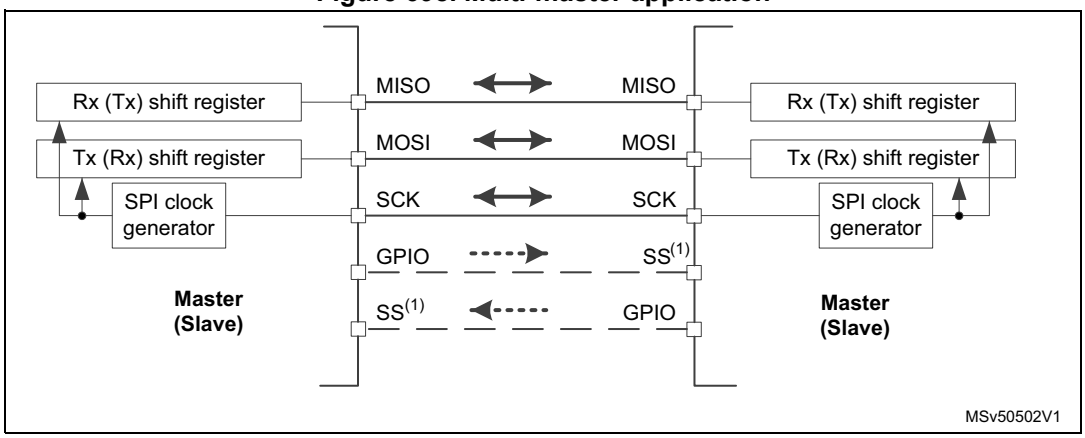
59.4.6 Multi-master communication

Unless the SPI bus is not designed primarily for a multi-master capability, it is possible to use a built-in feature that detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, the SS pin is used configured in hardware input mode. The connection of more than two SPI nodes working in this mode is impossible, as only one node can apply its output on a common data line at a given time.

When the nodes are not active, both stay in Slave mode by default. When a node wants to take control on the bus, it switches itself into Master mode and applies active level on the slave select input of the other node via the 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 both nodes raise their mastering request at the same time, a bus conflict event appears (see mode fault MODF event). The user can apply some simple arbitration process (e.g. postpone next attempt by predefined different time-outs applied in both nodes).

Figure 698. Multi-master application



1. The SS pin is configured at hardware input mode at both nodes. Its active level enables the MISO line output control as passive node is configured as a slave.
2. The RDY signal is not used in this communication.

59.4.7 Slave select (SS) pin management

In Slave mode, the SS works as a standard ‘chip select’ input and lets the slave communicate with the master. In Master mode, the SS can be used either as an output or an input. As an input it can prevent a multi master bus collision, and as an output it can drive a slave select signal of a single slave. The SS signal can be managed internally (software management of the SS input) or externally when both the SS input and output are associated with the SS pin (hardware SS management). The user can configure which level of this input/output external signal (present on the SS pin) is considered as active one by the SSIOP bit setting. SS level is considered as active if it is equal to SSIOP.

The hardware or software slave select management can be set using the SSM bit in the SPI_CFG2 register:

- **Software SS management (SSM = 1)**: in this configuration, slave select information is driven internally by the SSI bit value in the register SPI_CR1. The external SS pin is free for other application uses (as GPIO or other alternate function).
- **Hardware SS management (SSM = 0)**: in this case, there are two possible configurations. The configuration used depends on the SS output configuration (SSOE bit in register SPI_CFG2).
  - **SS output enable (SSOE = 1)**: this configuration is only used when the MCU is set as master. The SS pin is managed by the hardware. The functionality is tied to CSTART and EOT control. As a consequence, the master must apply proper TSIZE>0 setting to control the SS output correctly. Even if SPI AF is not applied at the SS pin (it can be used as a standard GPIO then), keep anyway SSOE = 1 to assure default SS input level and prevent any mode fault evaluation at input of the master SS internal logic applicable at a multi-master topology exclusively.

  a) When SSOM = 0 and SP = 000, the SS signal is driven to the active level as soon as the master transfer starts (CSTART = 1) and it is kept active until its EOT flag is set or the transmission is suspended.

  b) When SP = 001, a pulse is generated as defined by the TI mode.

  c) When SSOM = 1, SP = 000 and MIDI > 1 the SS is pulsed inactive between data frames, and kept inactive for a number of SPI clock periods defined by the MIDI value decremented by one (1 to 14).

  d) SS input is forced to non active state internally at master to prevent its any mode fault.

  – **SS output disable (SSM = 0, SSOE = 0)**:

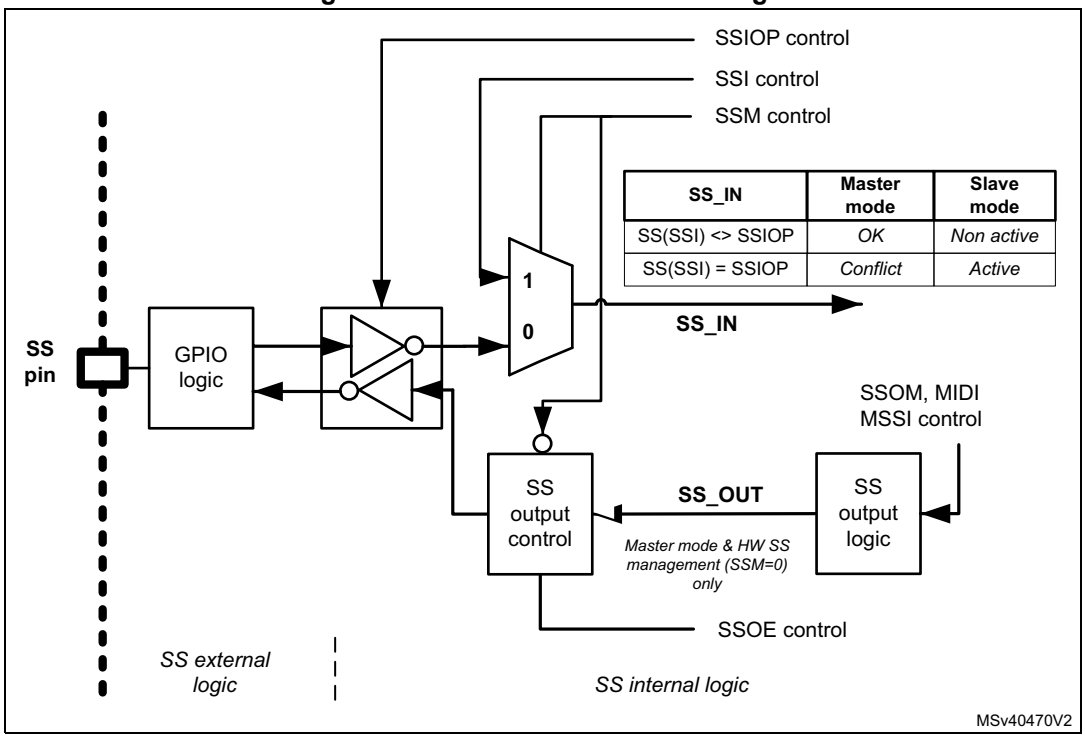
  a) if the micro-controller is acting as the master on the bus, this configuration allows multi master capability. If the SS pin is pulled into an active level in this mode, the SPI enters Master mode fault state and the SPI device is automatically reconfigured in Slave mode (MASTER = 0).

  b) In Slave mode, the SS pin works as a standard ‘chip select’ input and the slave is selected while the SS line is at its active level.

Note: The purpose of automatic switching into Slave mode at mode fault condition is to avoid the possible conflicts on data and clock line. As the SPE is automatically reset in this condition, both Rx and Tx FIFOs are flushed and current data is lost.

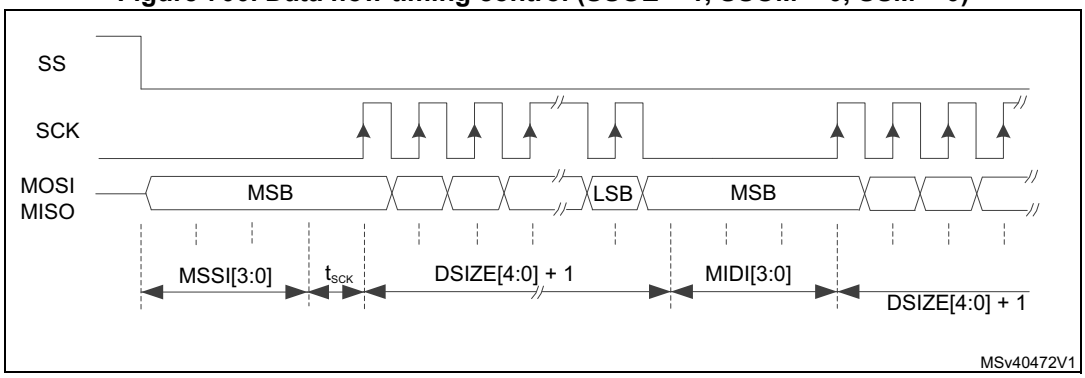
When the SPI slave is enabled in the hardware SS management mode, all the traffic is ignored even in case of the SS is found at active level. They are ignored until the slave detects a start of the SS signal (transition from non-active to active level) just synchronizing the slave with the master. This is because the hardware management mode cannot be used when the external SS pin is fixed. There is no such protection in the SS software management. Then the SSI bit must be changed when there is no traffic on the bus and the SCK signal is at idle state level between transfers exclusively in this case.

Figure 699. Scheme of SS control logic



When the hardware output SS control is applied (SSM = 0, SSOE = 1), by configuration of the MIDI[3:0] and MSSI[3:0] bit fields, the user can control the timing of the SS signal between data frames and can insert an extra delay at the beginning of every transaction (to separate the SS and clock starts). This can be useful when the slave needs to slow down the flow to obtain sufficient room for correct data handling (see *Figure 700*).

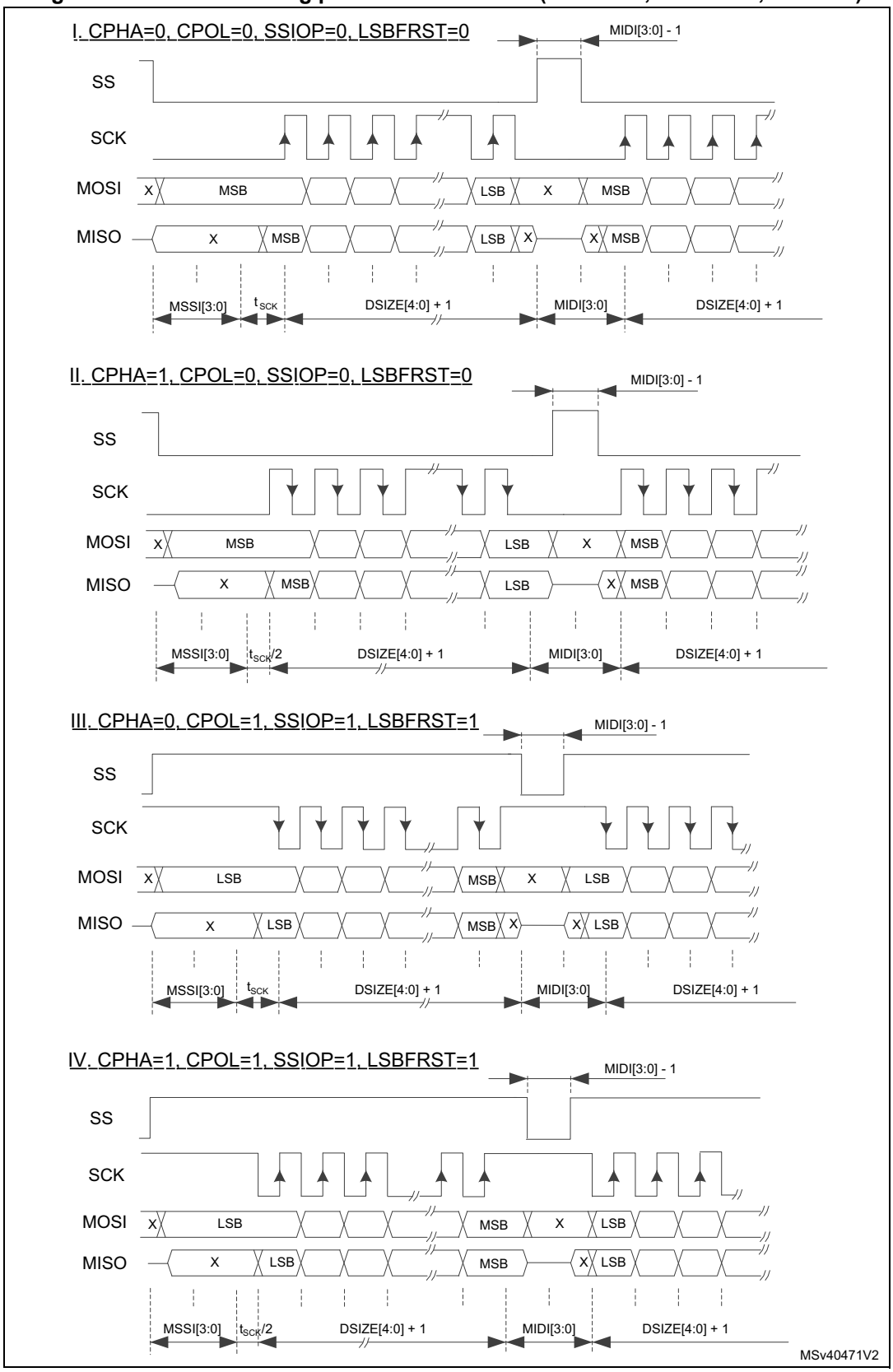
Figure 700. Data flow timing control (SSOE = 1, SSOM = 0, SSM = 0)



1. MSSI[3:0] = 0011, MIDI[3:0] = 0011 (SCK flow is continuous when MIDI[3:0] = 0).
2. CPHA = 0, CPOL = 0, SSIOP = 0, LSBFRST = 0.

Additionally, bit SSOM = 1 setting invokes specific mode which interleaves pulses between data frames if there is a sufficient space to provide them (MIDI[3:0] must be set greater then one SPI period). Some configuration examples are shown in *Figure 701*.

Figure 701. SS interleaving pulses between data (SSOE = 1, SSOM = 1, SSM = 0)



1. MSSI[3:0] = 0010, MIDI[3:0] = 0010.
2. SS interleaves between data when MIDI[3:0] > 1 wide of the interleaving pulse is always one SCK period less then gap provided between frames (defined by MIDI parameter). If MIDI is set to 1 the frames are separated by single SCK period but no interleaving pulse appears on SS.

59.4.8 Ready pin (RDY) management

The status of the slave capability to handle data, can be checked on the RDY pin . By default, a low level indicates that the slave is not ready for transaction. The reason can be that slave's TxFIFO is empty, RxFIFO full or the SPI is disabled. An active level of the signal can be selected by the RDIOP bit. If the master continues or starts to communicate with the slave when it indicates a not ready status, the transaction fails great probably.

The logic to control the RDY output is rather complex, tied closely with TSIZE and DSIZE settings. The RDY reaction is more pessimistic and sensitive to TxFIFO becoming nearly empty and/or RxFIFO nearly full during a frame transaction. This pessimistic logic is suppressed at the end of a transaction only when RDY stays active, despite TxFIFO becomes fully empty and/or RxFIFO becomes fully occupied. The target is to prevent any data corruption and inform master in time that it is necessary to suspend the transaction temporarily till the next transacted data can be processed safely again. When RDY signal input is enabled at master side, master suspends the communication once the slave indicates not ready status. This prevents the master to complete transaction of an ongoing frame which just empties slave's TxFIFO or full fills its RxFIFO till a next data is written and/or read there (despite the frame still could be completed without any constraint). It could make a problem if TSIZE = 0 configuration is applied at slave because slave then never evaluates end of transaction (which suppresses the not ready status just when the last data is sent). Then the user has to release the RxFIFO and/or write additional (even dummy) data to TxFIFO by software at slave side to release the not RDY signal, unblock ST master and so enable it to continue at the communication suspended at middle of a frame occasionally.

When RDY is not used by the master, it must be disabled (RDIOM = 0). Then an internal logic of the master simulates the slave status always ready. In this case, the RDIOP bit setting has no meaning.

Due to synchronization between clock domains and evaluation of the RDY logic on both master and slave sides, the RDY pin feature is not reliable and cannot be used when the size of data frames is configured shorter than 8-bit.

59.4.9 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 slave devices must follow the same communication format and be synchronized correctly.

Clock phase and polarity controls

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPI_CFG2 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data are 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.

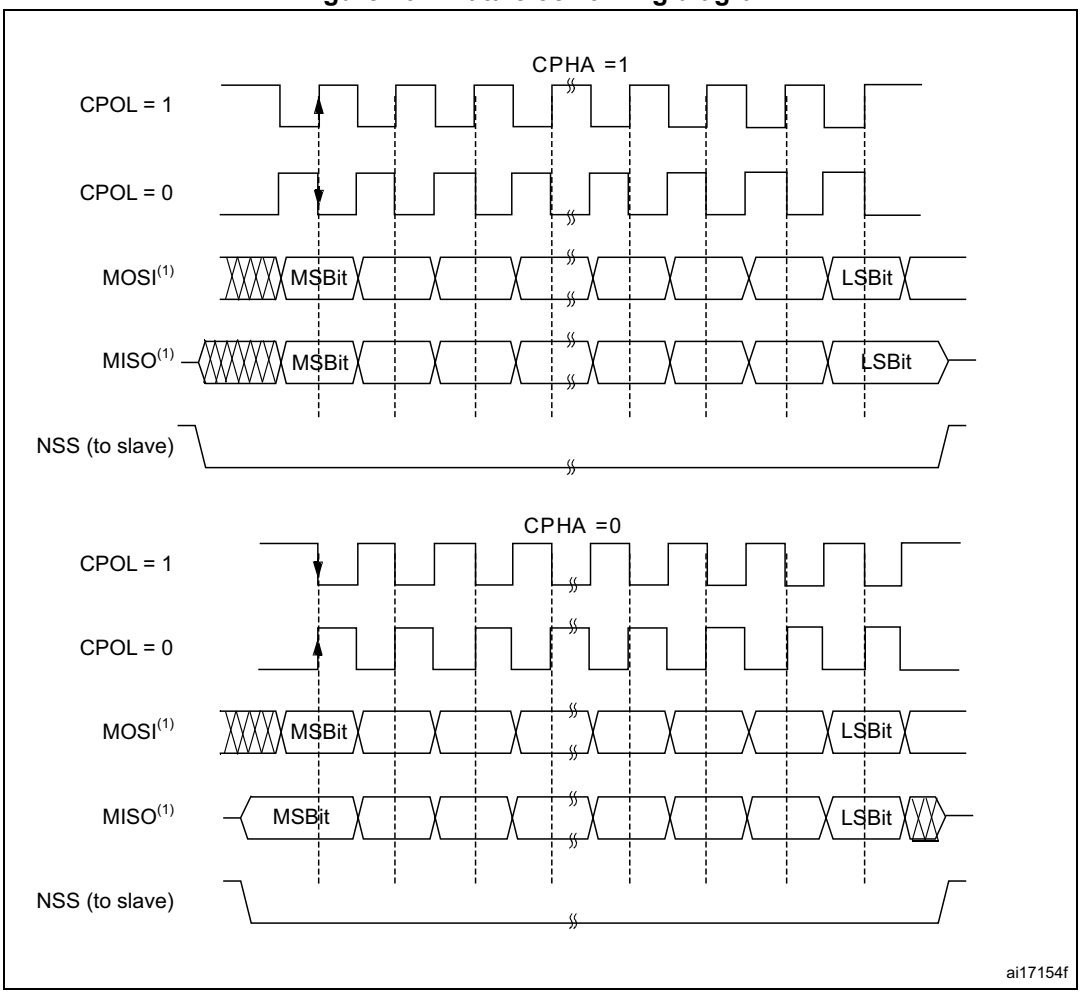
The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edges (dotted lines in *Figure 702*).

Figure 702, 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 SPI_CFG2 register (by pulling the SCK pin up if CPOL = 1 or pulling it down if CPOL = 0).

Figure 702. Data clock timing diagram



1. The order of data bits depends on LSBFRST bit setting.

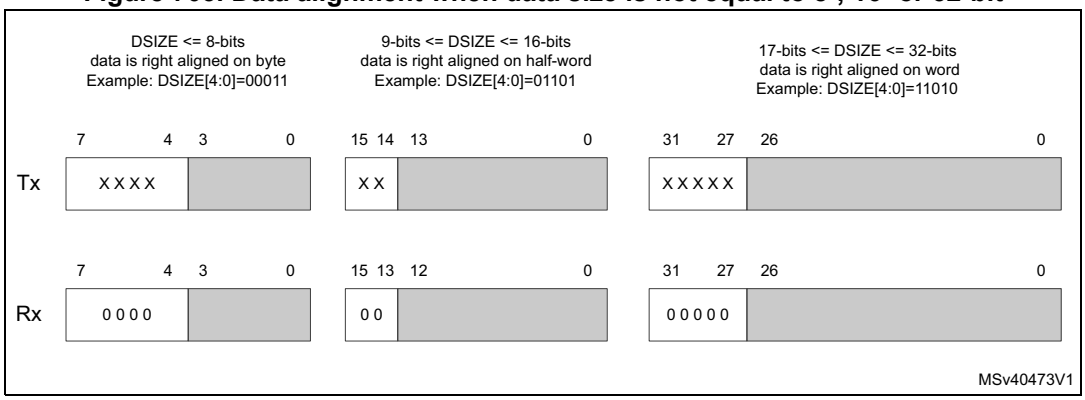
Data frame format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBFRST bit in SPI_CFG2 register.

At instance with full feature set, the data frame size is chosen by using the DSIZE[4:0] bits at SPI_CFG1 register. It can be set from 4-bit up to 32-bit length and the setting applies for both transmission and reception. When the SPI_TXDR/SPI_RXDR registers are accessed, data frames are always right-aligned into either a byte (if the data fit into a byte), a half-word or a word (see *Figure 703*).

If the access is a multiple of the configured data size, data packing is applied automatically. During communication, only bits within the data frame are clocked and transferred.

Figure 703. Data alignment when data size is not equal to 8-, 16- or 32-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 4-bit data frame size.

At the instance with limited set of features, data size is fixed to multiply 8-bit up to maximum data length (depends on instance) in according to DSIZE[4:3] bits value. If the SPI_TXDR or SPI_RXDR are accessed by wider access (a multiply of the configured the data size), data packing is applied automatically.

59.4.10 Configuring the SPI

The configuration procedure is almost the same for the master and the slave. For specific mode setups, follow the dedicated chapters. When a standard communication must be initialized, perform these steps:

1. Write the proper GPIO registers: configure GPIO alternate functions at MOSI, MISO, SCK, SS and RDY pins if applied.
2. Write into the SPI_CFG1 and SPI_CFG2 registers and set up proper values of all ‘not reserved’ bits and bit fields, prior SPI is enabled, with the following exceptions:
   a) The SSOM, MASRX, SSOE, RDIOM, MBR[2:0], BPASS, MIDI[3:0], MSSI[3:0] bits are taken into account in Master mode only, the MSSI[3:0] bits take effect when the SSOE bit is set, the RDIOP bit takes no effect when the RDIOM bit is not set in Master mode. When slave is configured at TI mode, MBR[2:0] setting is considered, too.
   b) UDRCFG is taken into account in Slave mode only.
   c) CRCSIZE[4:0] is required if CRCEN is set.

d) CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP, SSM, RDIOP, RDIOM, MSSI and MIDI are not required in TI mode.

e) Once the AFCNTR bit is set in the SPI_CFG2 register, all the SPI outputs start to be propagated onto the associated GPIO pins regardless the peripheral enable. So any later configuration changes of the SPI_CFG1 and SPI_CFG2 registers can affect the level of signals on these pins.

3. Write to the SPI_CR2 register to select the length of the transfer, if it is not known TSIZE must be programmed to zero.
4. Write to SPI_CRCPOLY and into TCRCINI, RCRCINI and CRC33_17 bits at the SPI_CR1 register to configure the CRC polynomial and CRC calculation if needed.
5. Configure DMA streams dedicated for the SPI Tx and Rx in DMA registers if the DMA streams are used (see chapter Communication using DMA).
6. Configure SSI, HDDIR and MASRX at SPI_CR1 register if required.
7. Program the IOLOCK bit in the SPI_CFG1 register if the configuration protection is required (for safety).

59.4.11 Enabling the SPI

It is recommended to configure and enable the SPI slave before the master sends the clock. But there is no impact if the configuration and enabling procedure is done while a traffic is ongoing on the bus, assuming that the SS signal is managed by hardware at slave or kept inactive by slave's software when the software management of the SS signal is applied (see *Section 59.4.7*). The data register of the slave transmitter should contain data to be sent before the master starts its clocking.The SCK signal must be settled to the idle state level corresponding to the selected polarity, before the SPI slave is selected by SS, else the following transaction may be desynchronized.

When the SPI slave is enabled at the hardware SS management mode, all the traffics are ignored even in case of the SS is found at active level. They are ignored until the slave detects a start of the SS signal (its transition from non-active to active level) just synchronizing the slave with the master. This is why the hardware management mode cannot be used when external SS pin is fixed. There is no such protection at the SS software management. In this case, the SSI bit must be changed when there is no traffic on the bus and the SCK signal is at idle state level between transfers exclusively in this case.

The master in full duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled, the CSTART bit is set and the TxFIFO is not empty, or with the next write to TxFIFO.

In any master receive-only mode, the master starts to communicate and the clock starts running after the SPI is enabled and the CSTART bit is set.

For handling DMA, see *Section 59.4.15*.

59.4.12 SPI data transmission and reception procedures

The setting of data communication format follows the basic principle that sure number of data with a flexible size must be transferred within a session (transaction) while, optionally, the data handling can be cumulated effectively into a single access of the SPI data registers (data packing) or even grouped into a sequence of such services if data is collected at consistent bigger data packets. The data handling services are based upon FIFO packet occupancy events. This is why the complete data packet must be serviced exclusively upon a dedicated packet flag.

To understand better the next detailed content of this section, the user should capture the configuration impact and meaning of the following items at first:

Data size (DSIZE) - defines data frame (sets the number of bits at single data frame).

FIFO threshold (FTHLV) - defines data packet, sets the number of data frames at single data packet and so the occurrence of the packet occupancy events to handle SPI data registers either by software or by DMA.

Data access – a way how to handle the SPI data register content when the transfer data between the application and the SPI FIFOs upon a packet event. It depends on the packet size configuration. Optionally, multiply data can be handled effectively by a single access of the register (by data packing) or by sequence of such accesses (when servicing a bigger data packet).

FIFO size – capacity or space to absorb available data. It depends on the data size and the internal hardware efficiency how the data is compressed and organized within this space. The FTHLV setting must respect the FIFO capacity to store two data packets at least.

Transaction size (TSIZE) – defines total number of data frames involved at a transaction session overall possibly covered by several data packet services. There is no need to align this number with the packet size (handling of a last not aligned data packet is supported if TSIZE is programmed properly).

Data handling via RxFIFO and TxFIFO

All SPI data transactions pass through the embedded FIFOs organized by bytes (N x 8-bit). The size of the FIFOs (N) is dependent on the product and the peripheral instance. This enables the SPI to work in a continuous flow, and prevents overruns when the data frame size is short or the interrupt/DMA latency is too long. Each direction has its own FIFO called TxFIFO and RxFIFO, respectively.

The handling of the FIFOs content is based on servicing data packet events exclusively raised by dedicated FIFO packet occupancy flags (TXP, RXP or DXP). The flags occurrence depends on the data exchange mode (duplex, simplex), the data frame size (number of bits in the frame) and how data are organized at data packets. The frequency of the packet events can be decreased significantly when data are organized into packets via defining the FIFOs threshold. Several data frames grouped at packet can be then handled effectively based on a single FIFO occupancy packet event either by a single SPI data register access or their sequence what consumes less system performance. The user can control the access type by casting the data register address to force a concrete CPU instruction applied for the register read or write. The access then can be 8-bit, 16-bit or 32-bit but single data frame must be always accessed at least. It is crucial to keep the setting of the packet size (FTHVL) and the data size (DSIZE) always balanced with the applied data registers access (no matter if a single access or their sequence is applied) just to apply and complete service of a single data packet upon its event. This principle, occurrence and clearing capabilities of the FIFO occupancy flags are common no matter if DMA, interrupt, or polling is applied.

A read access to the SPI_RXDR register returns the oldest value stored in the RxFIFO that has not been read yet. A write access to the SPI_TXDR stores the written data in the TxFIFO at the end of a send queue.

A read access to the SPI_RXDR register must be managed by the RXP event. This flag is set by hardware when at least one complete data packet (defined as receiver threshold by FTHLV[3:0] bits at the SPI_CFG1 register) is available at the reception FIFO while reception is active. The RXP is cleared as soon as less data than complete single packet is available in the RxFIFO, when reading SPI_RXDR by software or by DMA.

The RXP triggers an interrupt if the RXPIE bit is set and/or a DMA request if the RXDMAEN bit is set.

Upon setting of the RXP flag, the application performs the due number of SPI data register reads to download the content of one data packet. Once a complete data packet is downloaded, the application software or DMA checks the RXP value to see if other packets are pending into the receive FIFO and, if so, downloads them packet by packet until the RXP reads 0. RxFIFO can store up to N data frames (for frame size ≤ 8-bit), N/2 data frames (for 8-bit < frame ≤ 16-bit), N/3 data frames (for 16-bit < frame ≤ 24-bit) or N/4 data frames (if data frame > 24-bit) where N is the size of the FIFO in bytes.

At the end of a reception, it may happen that some data are still available in the RxFIFO, without reaching the FTHLV level, thus the RXP is not set. In this case, the number of remaining RX data frames in the FIFO is indicated by RXWNE and RXPLVL fields in the SPI_SR register. It happens when the number of the last data received in a transfer cannot fully accomplish the configured packet size; in case the transfer size and the packet size are not aligned. Nevertheless the application software can still perform the standard number of reads from the RxFIFO used for the previous complete data packets without drawbacks: only the consistent data (completed data frames) are popped from the RxFIFO while redundant reads (or any uncompleted data) are reading 0. Thanks to that, the application software can treat all the data in a transfer in the same way, and is off-loaded to foresee the reception of the last data in a transfer and to calculate the due number of reads to be popped from RxFIFO.

In a similar way, the write access of a data frame to be transmitted is managed by the TXP event. This flag is set by hardware when there is enough space for the application to push at least one complete data packet (defined at FTHLV[3:0] bits at SPI_CFG1 register) into the transmission FIFO while transmission is active. The TXP is cleared as soon as the TxFIFO is filled by software and/or by the DMA. The space currently available for any next complete data packet is lost. This can lead to oscillations of the TXP signal when data are released out from the TxFIFO while a new packet is stored frame by frame. Any write to the TxFIFO is ignored when there is no sufficient room to store at least a single data frame (TXP event is not respected), when TXTF is set or when the SPI is disabled.

The TXP triggers an interrupt if the TXPIE bit is set and/or with a DMA request if the TXDMAEN bit is set. The TXPIE mask is cleared by hardware when the TXTF flag is set.

Upon setting of the TXP flag, the application performs the due number of SPI data register writes to upload the content of one entire data packet. Once new complete data packet is uploaded, the application software or DMA checks the TXP value to see if other packets can be pushed into the TxFIFO and, if so, uploads them packet by packet until TXP reads 0.

The number of last data in a transfer can be shorter than the configured packet size in the case when the transfer size and the packet size are not aligned. Nevertheless the application software can still perform the standard number of data register writes used for the previous packets without drawbacks: only the consistent data are pushed into the TxFIFO while redundant writes are discarded. Thanks to that, the application software can treat all the data in a transfer in the same way and is off-loaded to foresee the transmission of the last data in a transfer and from calculating the due number of writes to push the last data into TxFIFO. Just for the last data case, the TXP event is asserted by SPI once there is enough space into TxFIFO to store remaining data to complete current transfer.

Both TXP and RXP events can be polled or handled by interrupts. The DXP bit can be monitored as a common TXP and RXP event at full-duplex mode.

Upon setting of the DXP flag the application performs the due number of writes to the SPI data register to upload the content of one entire data packet for transmission, followed by the same number of reads from the SPI data register to download the content of one data packet. Once one data packet is uploaded and one is downloaded, the application software or DMA checks the DXP value to see if other packets can be pushed and popped in sequence and, if so, uploads/downloads them packet by packet until DXP reads 0.

The DXP triggers an interrupt if the DXPIE bit is set. The DXPIE mask is cleared by hardware when the TXTF flag is set.

The DXP is useful in full-duplex communication in order to optimize performance in data uploading/downloading, and reducing the number of interrupts or DMA sequences required to support an SPI transfer thus minimizing the request for CPU bandwidth and system power especially when SPI is operated in Stop mode.

When relay on the DXP interrupt exclusively, the user must consider the drawback of such a simplification when TXP and RXP events are serviced by common procedures because the TXP services are delayed by purpose in this case. This is due to fact that the TXP events precedes the reception RXP ones normally to allow the TXP servicing prior transaction of the last frame fully emptying the TxFIFO else master cannot provide a continuous SCK clock flow and the slave can even face an underrun condition. The possible solution is to prefill the TxFIFO by few data packets ahead prior the session starts and to handle all the data received after the TXTF event by EOT exclusively at the end of the transaction (as TXTF suppresses the DXP interrupts at the end of the transaction). In case of CRC computation is enabled, the user must calculate with additional space to accommodate the CRC frame at RxFIFO when relying on EOT exclusively at the end of transaction.

Another way to manage the data exchange is to use DMA (see *Section 59.4.15*).

If the next data is received when the RxFIFO is full, an overrun event occurs (see description of OVR flag in *Section 59.5.2*). An overrun event can be polled or handled by an interrupt.

This may happen in Slave mode or in a master receive mode when MASRX = 0. If MASRX bit is set at a master receiver, the generated clock stops automatically when the RxFIFO is full, therefore overrun is prevented.

Both RxFIFO and TxFIFO content is kept flushed and cannot be accessed when SPI is disabled (SPE = 0).

Transaction handling

A few data frames can be passed at single sequence to complete a message. The user can handle a number of data within a message thanks to the value stored into TSIZE. In principle, the transaction of a message starts when the SPI is enabled by setting CSTART bit and finishes when the total number of required data is transacted. The end of transaction controls the CRC and the hardware SS management when applied. To restart the internal state machine properly, SPI is strongly suggested to be disabled and re-enabled before next transaction starts despite its setting is not changed.

If TSIZE is kept at zero while CSTART is set, an endless transaction is initialized (no control of transfer size is applied). During an endless transaction, the number of transacted data aligned with FIFOs threshold is supported exclusively. If the number of data (or its grouping into packets) is unpredictable, the user must keep the FIFO threshold setting (packet size) at single data (FTHLV = 0) to assure that each data frame raises its own packet event to be serviced by the application or DMA. The transaction can be suspended at any time thanks

to CSUSP which clears the CSTART bit. SPI must be always disabled after such software suspension and re-enabled before the next transaction starts.

When the transmission is enabled, a sequence begins and continues while any data is present in the TxFIFO of the master. The clock signal is provided permanently by the master until TxFIFO becomes empty, then it stops, waiting for additional data.

In receive-only modes, half-duplex (COMM[1:0] = 11, HDDIR = 0) or simplex (COMM[1:0] = 10) modes, the master starts the sequence when the SPI is enabled and the transaction is released by setting the CSTART bit. The clock signal is provided by the master and it does not stop until either SPI or receive-only mode is disabled/suspended by the master. The master receives data frames permanently up to this moment. The reception can be suspended either by software control, writing 1 to the CSUSP bit in the SPI_CR1 register, or automatically when MASRX = 1 and RxFIFO becomes full or upon the RDY status if this signal is applied (see *Section 59.4.8*). The reception is automatically stopped also when the number of frames programmed in TSIZE has been completed.

In order to disable the master receive-only mode, the SPI must first be suspended. When the SPI is suspended, the current frame is completed, before changing the configuration.

Caution: If the SPE bit is written to 0 in Master mode, while the reception is ongoing without any suspending, the clock is stopped without completing the current frame, and the RxFIFO is flushed.

While the master can provide all the transactions in continuous mode (SCK signal is continuous) it must respect the slave capability to handle data flow and its content at anytime. If the slave features the RDY signal option, the master can monitor the RDY signal issued by the slave, to control the communication flow. If the RDY pin is not used, the slave is considered always ready for communication with the master.

When necessary, the master must slow down the communication and provide either a slower clock or separate frames or data sessions with sufficient delays by MIDI[3:0] bits setting or provide an initial delay by setting MSSI[1:0], which postpones any transaction start to give slave sufficient room for preparing data. Be aware data from the slave are 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 short, FIFO is accessed by bytes and the SPI bus rate is high.

In order to add some software control on the SPI communication flow from a slave transmitter node, a specific value written in the SPI_UDRDR (SPI Underrun Data Register) may be used. On slave side, when TxFIFO becomes empty, this value is sent out automatically as next data and may be interpreted by software on the master receiver side (either simply dropped or interpreted as a XOFF like command, in order to suspend the master receiver by software).

In the multi-slave star topology, only a single slave only can be enabled for output data at a given time. The slave just selected for the communication with the master needs to detect a change of its SS input into active level before communication with the master starts. In a single slave system it is not necessary to control the slave with SS, but it is often better to provide the pulse here too, to synchronize the slave with the beginning of each data sequence. The SS can be managed by both software and hardware (*Section 59.4.7*).

59.4.13 Disabling the SPI

To disable the SPI, it is mandatory to follow the disable procedures described in this paragraph.

In the Master mode, it is important to do this before the system enters a low-power mode when the peripheral clock is stopped, otherwise, ongoing transactions may be corrupted.

In Slave mode, the SPI communication can continue when the **spi_pclk** and **spi_ker_ck** clocks are stopped, without interruption, until any end of communication or data service request condition is reached. The **spi_pclk** can generally be stopped by setting the system into Stop mode. Refer to the RCC section for further information.

The 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. TXC flag can be polled (or interrupt enabled with EOTIE = 1) in order to wait for the last data frame to be sent.

When the master is in any receive-only mode, in order to stop the peripheral, the SPI communication must first be suspended, by setting the CSUSP bit to 1.

The data received but not read remain stored in RxFIFO when the SPI is suspended.

After such a software suspension, SPI must be always disabled to restart the internal state machine properly.

When SPI is disabled, RxFIFO is flushed. To prevent losing unread data, the user must ensure that RxFIFO is empty when disabling the SPI, by reading all remaining data (as indicated by the RXP, RXWNE and RXPLVL fields in the SPI_SR register).

The standard disable procedure is based on polling EOT and/or TXC status 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 the master handles SS signal by a GPIO not related to SPI (for example at case of multi-slave star topology) and it has to provide proper end of SS pulse for slave, or
- When transaction streams from DMA or FIFO are completed while the last data frame or CRC frame transaction is still ongoing in the peripheral bus.

When TSIZE>0, EOT and TXC signals are equal so polling of EOT is reliable at whatever SPI communication mode to check end of the bus activity. When TSIZE = 0, the user has to check TXC, SUSP or FIFO occupancy flags in according with the applied SPI mode and the way of the data flow termination.

The correct disable procedure in Master mode, except when receive-only mode is used, is:

1. Wait until TXC = 1 and/or EOT = 1 (no more data to transmit and last data frame sent). When CRC is used, it is sent automatically after the last data in the block is processed. TXC/EOT is set when CRC frame is completed in this case. When a transmission is suspended the software has to wait till CSTART bit is cleared.
2. Read all RxFIFO data (until RXWNE = 0 and RXPLVL = 00).
3. Disable the SPI (SPE = 0).

The correct disable procedure for master receive-only modes is:

1. Wait on EOT or break the receive flow by suspending SPI (CSUSP = 1).
2. Wait until SUSP = 1 (the last data frame is processed) if receive flow is suspended.
3. Read all RxFIFO data (until RXWNE = 0 and RXPLVL = 00).
4. Disable the SPI (SPE = 0).

In Slave mode, any on going data are lost when disabling the SPI.

Controlling the I/Os

As soon as the SPI is disabled, the associated and enabled AF outputs can still be driven by the device depending on the AFCNTR setting. When active output control is applied (AFCNTR = 1) and SPI is just been disabled (SPE = 0), the enabled outputs associated with SPI control signals (like NSS and SCK at master and RDY at slave) can toggle immediately to inactive level (according to SSIOP and CPOL settings at master and RDIOP at slave respectively). The data line output (MOSI at master and MISO at slave) can instead change its level immediately at dependency on the actual TxFIFO content with the effect of potentially making invalid and no more guaranteed the value of the latest transacted bit on the bus. If necessary, the user has to take care about proper data hold time at the data line and avoid any eventual fast SPI disable just after the last data transaction is completed.

Note: Despite stability of the latest bit is guaranteed by design during the sampling edge of the clock, some devices can require even extension of this data bit stability interval during the sampling. It can be done for example by inserting small software delay between EOT event occurrence and SPI disable action.

59.4.14 Data packing

From user point of view there are two ways of data packing which can overlay each other:

- Type of access when data are written to TxFIFO or read from RxFIFO
  Multiple data can be pushed or fetched effectively by single access if data size is multiplied less than the access performed upon SPI_TXDR or SPI_RXDR registers.
- Number of data to be handled during the single software service
  It is convenient to group data into packets and cumulate the FIFO services overall the data packet content exclusively instead of handling data frame by frame separately. The user can define packets by FIFO threshold settings. Then all the FIFO occupancy events are related to that threshold level while required services are signalized by proper flags with interrupt and/or wake up capabilities.

When the data frame size fits into one byte (less than or equal to 8 bits), the data packing is used automatically when any read or write 16-bit or 32-bit access is performed on the SPI_RXDR/SPI_TXDR register. The multiple 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 data stored in the MSB.

Figure 704 provides an example of data packing mode sequence handling at full feature set instance. While DSIZE[4:0] is configured to 4-bit there, two or four data frames are written in the TxFIFO after the single 16-bit or 32-bit access the SPI_TXDR register of the transmitter. When the data frame size is between 9-bit and 16-bit, data packing is used automatically when a 32-bit access is done. Least significant half-word is used first. (regardless of the LSBFRST value)

This sequence can generate two or four RXP events in the receiver if the RxFIFO threshold is set to 1 frame (and data is read on a frame basis, unpacked), or it can generate a single RXP event if the FTHLV[3:0] field in the SPI_CFG1 register is programmed to a multiple of the frames to be read in a packed mode (16-bit or 32-bit read access).

The data are aligned in accordance with *Figure 703*. The valid bits are performed on the bus exclusively. Unused bits are not cared at transmitter while padded by zeros at receiver.

When short data frames (< 8-bit or < 16-bit) are used together with a larger data access mode (16-bit or 32-bit), the FTHLV value must be programmed as a multiple of the number

of frames/data access (multiple of 4 if 32-bit access is used to up to 8-bit frames or multiple of 2 if 16-bit access is used to up to 8-bit frames or 32-bit access to up to 16-bit frames.).

The RxFIFO threshold setting must always be higher or equal at least than the following read access size, as spurious extra data would be read otherwise.

The FIFO data access less than the configured data size is forbidden. One complete data frame must be always accessed at minimum.

A specific problem appears if an incomplete data packet is available at FIFO: less than threshold set at FTHLV bits.

There are two ways of dealing with this problem:

A. without using TSIZE field

On transmitter side, writing the last data frame of any odd sequence with an 8-bit/16-bit access to SPI_TXDR is enough.

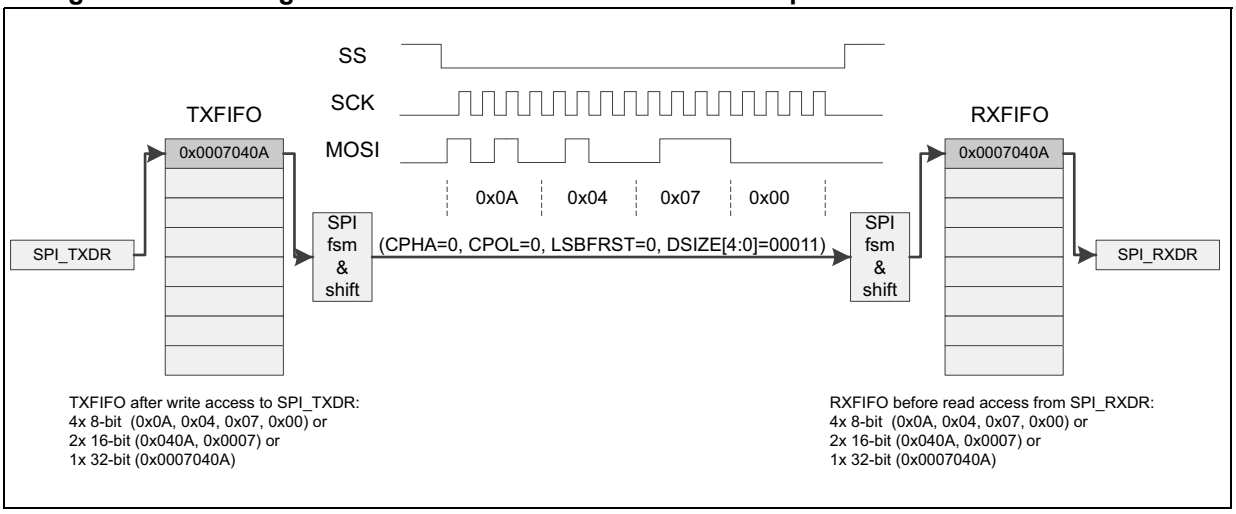
On receiver side, the remaining data may be read by any access. Any extra data read are padded with zeros. Polling the RXWNE and RXPLVL may be used to detect when the RX data are available in the RxFIFO. (A time out may be used at system level in order to detect the polling)

B. using the TSIZE field

On transmitter side, the transaction is stopped by the master when it faces EOT event.

In reception, the RXP flag is not set when EOT is set. In the case when the number of data to be received (TSIZE) is not a multiple of packet size, the number of remaining data is indicated by the RXWNE and RXPLVL fields in the SPI_SR register. The remaining data can be read by any access. Any extra read is padded by zeros.

Figure 704. Packing data in FIFO for transmission and reception at full feature set instance



1. DSIZE[4:0] is configured to 4-bit, data is right aligned, valid bits are performed only on the bus, their order depends on LSBFRST, if it is set, the order is reversed at all the data frames.

59.4.15 Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXDMAEN or RXDMAEN enable bits in the SPI_CFG1 register are set. Separate requests must be issued to the Tx and Rx buffers to fulfill service of the defined packet.

- In transmission, a series of DMA requests is triggered each time TXP is set to 1. The DMA then performs series of writes to the SPI_TXDR register.
- In reception, a series of DMA requests is triggered each time RXP is set to 1. The DMA then performs series of reads from the SPI_RXDR register. When EOT is set at the end of transaction and last data packet is incomplete then DMA request is activated automatically in according with RXWNE and RXPLVL[1:0] setting to read rest of data.

If the SPI is programmed in receive-only mode, UDR is never set.

If the SPI is programmed in a transmit mode, TXP and UDR can be eventually set at slave side, because transmit data may not be available. In this case, some data are sent on the TX line according with the UDR management selection.

When the SPI is used at a simplex mode, the user must enable the adequate DMA channel only while keeping the complementary unused channel disabled.

If the SPI is programmed in transmit-only mode, RXP and OVR are never set.

If the SPI is programmed in full-duplex mode, RXP and OVR are eventually set, because received data are not read.

In transmission mode, when the DMA or the user has written all the data to be transmitted (the TXTF flag is set at SPI_SR register), the EOT (or TXC at case TISEZE = 0) 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 before disabling the **spi_pclk** in Master mode. The software must first wait until EOT = 1 and/or TXC = 1.

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_CFG1 register, if DMA Rx is used.
2. Enable DMA requests for Tx and Rx in DMA registers, if the DMA is used.
3. Enable DMA Tx buffer in the TXDMAEN bit in the SPI_CFG1 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 request for Tx and Rx in the DMA registers, if the DMA issued.
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_CFG1 register, if DMA Tx and/or DMA Rx are used.

Data packing with DMA

If the transfers are managed by DMA (TXDMAEN and RXDMAEN set in the SPI_CFG1 register) the 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 the SPI data size is less than or equal to 8-bit, then the packing mode is enabled. Similarly, If the DMA channel PSIZE value is equal to 32-bit and the SPI data size is less than or equal to 16-bit, then the packing mode is

enabled. The DMA then automatically manages the write operations to the SPI_TXDR register.

Regardless data packing mode is used and the number of data to transfer is not a multiple of the DMA data size (16-bit or 32-bit) while the frame size is smaller, DMA completes the transfer automatically in according with the TSIZE field setting.

Alternatively, last data frames may be written by software, in the single/unpacked mode.

To configure any DMA data access less than the configured data size is forbidden. One complete data frame must be always accessed at minimum.

59.4.16 Autonomous mode

The SPI is capable to handle and initialize transactions autonomously requiring no specific system execution interaction till the ongoing transaction ends.
Such autonomous transactions can be handled not only in Run or Sleep modes but even in Stop mode when the SPI logic is able to provide temporary clock requests addressed to the reset and clock controller (RCC) to ensure clocking of those SPI domains just necessary for handling the data flow between the memory and the peripheral interface at dependency in the SPI mode.

In Stop mode, the APB clock is requested by the peripheral each time the SPI registers need to be updated based on specific traffic events (mainly TXP and RXP). The required clock is provided by RCC if SPI autonomous mode is enabled at the RCC configuration and the SPI is clocked by an internal oscillator available in Stop mode.

Interrupts or DMA requests are then generated, depending on the SPI configuration. If no interrupt is enabled, the device remains in Stop mode. If DMA requests are enabled, the data are directly transferred to/from the SRAM thanks to the DMA while the device remains in Stop mode. If an enabled interrupt occurs, the device wakes up from Stop mode.

Note: The peripheral clock request stays pending till flag with enabled interrupt stays set. This is why it is important to service these pending requests and clear their flag as soon as possible at system sensitive to the low-power consumption especially and the application must acknowledge all pending interrupts events before switching the SPI to low-power mode.

Slave mode

When the SPI is configured as a standard slave and device is at Stop mode, the SPI kernel clock and the SPI APB clock are not provided permanently. All the data flow between the SPI interface and associated FIFOs is handled by an external SCK clock provided by outer master device within the serial interface clock domain. APB clock temporal requests are then based upon specific traffic events at dependency on the SPI configuration. As slave never initializes a transaction, there is no need to synchronize any transaction start in this mode.

Note: The peripheral clock request stays pending till flag with enabled interrupt stays set. This is why it is important to service these pending requests and clear their flag as soon as possible at system sensitive to the low-power consumption especially and the application must acknowledge all pending interrupts events before switching the SPI to low-power mode.

Master mode

The SPI operating in Master mode provides the SCK signal for outer slaves until the transaction is completed. The SCK signal is derived from the SPI clock generator running within the kernel clock domain fed from RCC upon kernel clock request provided by the SPI

when the device is in Stop mode. Temporal requests for APB clock are then based upon specific traffic events, depending upon the SPI configuration. The SPI master always initializes a transaction.

To minimize consumption in Stop mode, it is suggested to combine communication starts triggered by hardware (TRIGEN bit set in the SPI_AUTOCR register) and transfers of predefined data size (TSIZE > 0 in the SPI_CR2 register). This ensures that any APB clock request is suppressed between EOT handling and the next trigger event.

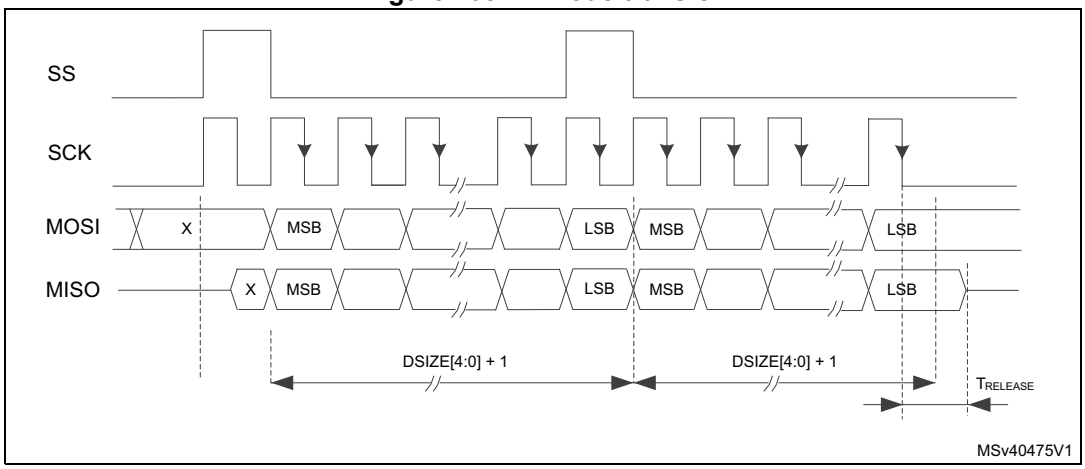
A transaction starts once the CSTART bit is set. In case of master transmitter, the TxFIFO must be filled by data, too. The CSTART bit can be written either by software or by hardware when a synchronous trigger is detected at Run, Sleep or Stop mode. The trigger source is selected by the TRIGSEL bits and enabled by the TRIGEN bit in the SPI_AUTOCR register. When the enabled trigger is detected, the transfer starts and continues by handling data till the EOT event or the transaction suspension. When the TRIGEN bit is changed, the user must prevent any trigger event occurrence. If the user cannot prevent that, the TRIGEN bit must be written while the SPI is disabled otherwise the peripheral behavior is not guaranteed.

59.5 SPI specific modes and control

59.5.1 TI mode

With a specific SP[2:0] bit field setting of the SPI_CFG2 register, the SPI can be configured compliant with the TI protocol. The SCK and SS signals polarity, phase and flow as well as the bits order are fixed so the setting of CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP, SSM, RDIOP, RDIOM, MSSI and MIDI is not required when the SPI is in TI mode configuration. The SS signal synchronizes the protocol by pulses over the LSB data bit as it is shown in *Figure 705*.

Figure 705. TI mode transfer



In Slave mode, the clock generator is used to define time when the slave output at MISO pin becomes to HiZ when the current transaction finishes. The master baud rate setting (MBR[2:0] at SPI_CFG1) is applied and any baud rate can be used to determine this moment with optimal flexibility. The delay for the MISO signal to become HiZ ($T_{RELEASE}$) depends on internal re-synchronization, too, which takes next additional 2-4 periods of the clock signal feeding the generator. It is given by formula:

$$\frac{Tbaud}{2} + 2 \times Tspi\_ker\_ck \leq Trelease \leq \frac{Tbaud}{2} + 4 \times Tspi\_ker\_ck$$

If the slave detects misplaced SS pulse during data transaction the TIFRE flag is set.

59.5.2 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and interrupt is enabled by setting the corresponding Interrupt Enable bit.

Overrun flag (OVR)

An overrun condition occurs when data are received by a master or slave and the RxFIFO has not enough space to store these 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).

When an overrun condition occurs, the OVR flag is set and the newly received value does not overwrite the previous one in the RxFIFO. The newly received value is discarded and all data transmitted subsequently are lost. OVR flag triggers an interrupt if OVRIE bit is set. Clearing the OVR bit is done by a writing 1 to the OVRC bit in the SPI_IFCR. To prevent any next overrun event the clearing should be done after RxFIFO is emptied by software reads. It is suggested to release the RxFIFO space as much as possible, this means to read out all the available data packets based on RXP flag indication.

In Master mode, the user can prevent the RxFIFO overrun by automatic communication suspend (MASRX bit).

Underrun flag (UDR)

In a slave-transmitting mode, the underrun condition is captured internally by hardware if no data is available for transmission in the slave TxFIFO commonly. The UDR flag setting is then propagated into the status register by hardware (see note below). UDR triggers an interrupt if the UDRIE bit is set.

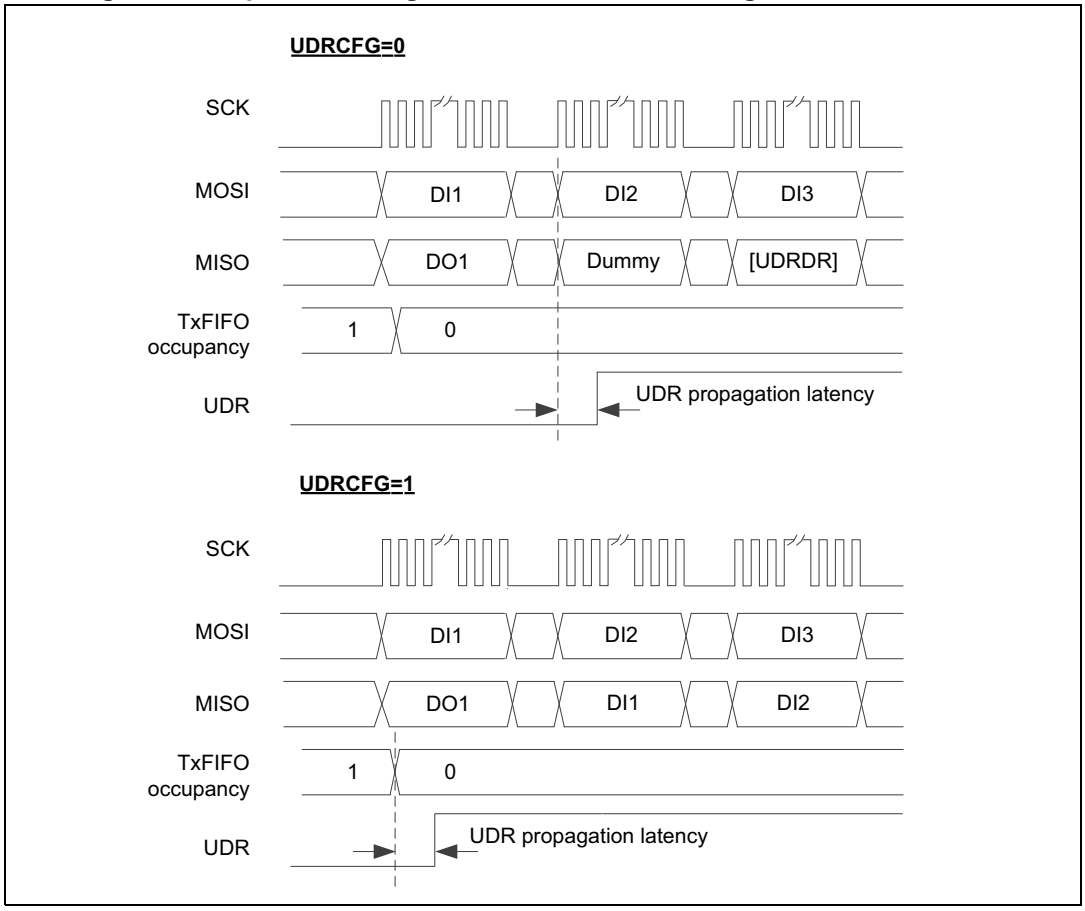
Underrun detection logic and system behavior depends on the UDRCFG bit. When an underrun is detected by slave, it can provide out either a constant pattern stored by the user at the UDRDR register or the data received previously from the master. When the first configuration (UDRCFG = 0) is applied, the underrun condition is evaluated whenever master starts to communicate a new data frame while TxFIFO is empty. Then single additional dummy (accidental) data is always inserted between last valid data and constant pattern defined at the UDRDR register (see *Figure 706*). Assuming that TxFIFO is not empty when master starts the communication, the underrun condition is evaluated just once the FIFO becomes empty during the next data flow. Valid data from TxFIFO is then upended by the lastly received data immediately.

The standard transmission is re-enabled once the software clears the UDR flag and this clearing is propagated into SPI logic by hardware. The user should write some data into TxFIFO prior clearing the UDR flag to prevent any next underrun condition occurrence capture.

The data transacted by slave is unpredictable especially when the transaction starts or continues while TxFIFO is empty and underrun condition is either not yet captured or just cleared. Typically, this is the case when SPI is just enabled or when a transaction with a defined size just starts. First bits can be corrupted in this case, as well, when slave software

writes first data into the empty TxFIFO too close prior the data transaction starts (propagation of the data into TxFIFO takes few APB clock cycles).

Figure 706. Optional configurations of slave detecting underrun condition



Note: The hardware propagation of an UDR event needs additional traffic on the bus. It always takes few extra SPI clock cycles after the event happens (both underrun captured by hardware and cleared by software). If clearing of the UDR flag by software is applied close to the end of data frame transaction or when SCK line is at idle in between the frames, next extra underrun pattern is sent initially by slave prior the valid data from TxFIFO becomes transacted again. The user can prevent this by SPI disable/enable action between sessions to restart the underrun logic and so initiate the next session by the valid data.

Mode fault (MODF)

Mode fault occurs when the master device has its internal SS signal (SS pin in SS hardware mode, or SSI bit in SS software mode) pulled low. This automatically affects the SPI interface in the following ways:

- The MODF bit is set and the SPI interrupt is triggered if the MODFIE bit is set.
- The SPE bit is forced to zero till MODF bit is set. This disables SPI and blocks all the peripheral outputs except the MODF interrupt request if enabled.
- The MASTER bit is cleared, thus forcing the device into Slave mode.

MODF is cleared by writing 1 to the MODFC bit in the SPI_IFCR.

To avoid any multiple slave conflicts in a system comprising several MCUs, the SS pin must be pulled to its non-active level before re-enabling the SPI, by setting the SPE bit.

As a security, hardware does not allow the SPE bit 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 multi master conflict.

A correct software procedure when master overtakes the bus at multi master system should be the following one:

- Switch into Master mode while SSOE = 0 (potential conflict can appear when another master occupies the bus. MODF is raised in this case, preventing any next node switching into Master mode)
- Put GPIO pin dedicated for another master SS control into active level
- Perform data transaction
- Put GPIO pin dedicated for another master SS control into non active level
- Switch back to Slave mode

CRC error (CRCE)

This flag is used to verify the validity of the value received when the CRCEN bit in the SPI_CFG1 register is set. The CRCE flag in the SPI_SR register is set if the value received in the shift register does not match the receiver SPI_RXCRC value, after the last data is received (as defined by TSIZE). The CRCE flag triggers an interrupt if CRCEIE bit is set. Clearing the bit CRCE is done by a writing 1 to the CRCEC bit in the SPI_IFCR.

TI mode frame format error (TIFRE)

A TI mode frame format error is detected when an SS 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 TIFRE flag is set in the SPI_SR register. The SPI is not disabled when an error occurs, the SS pulse is ignored, and the SPI waits for the next SS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of few data frames.

The TIFRE flag is cleared by writing 1 to the TIFREC bit in the SPI_IFCR. If the TIFREIE bit is set, an interrupt is generated on the SS error detection. As data consistency is no longer guaranteed, communication should be re-initiated by software between master and slave.

59.5.3 CRC computation

Two separate 33-bit or two separate 17-bit CRC calculators are implemented to check the reliability of transmitted and received data. For instances with full feature set, the SPI offers CRC polynomial length from 5 to 33 bits when maximum data size is 32-bit and from 9 to 17 bits for the peripheral instances with data size limited to 16 bits. For instances with limited set of features, the CRC polynomial length can be set either to 9 or 17 only when data size is limited to 16 bit and optionally to 33 when data size is extended to 32-bit. The length of the polynomial is defined by the most significant bit of the value stored at the CRCPOLY register. It must be set greater than data frame length defined at DSIZE field. When

maximum data size is applied, the CRC33_17 bit must be set additionally to define the most significant bit of the polynomial string while keep its size always greater than data. The CRCSIZE field in the SPI_CFG1 then defines how many the most significant bits from CRC calculation registers are transacted and compared as CRC frame. It is defined independently from the data frame length, but it must be either equal or an integer multiple of the data frame size while its size cannot exceed the maximum data size of the instance.

To fully benefit from the CRC calculation capability, the polynomial length setting must correspond to the CRC pattern size, else the bits unused at the calculation are transacted and expected all zero at the end of the CRC pattern if its size is set greater then the polynomial length.

CRC principle

The CRC calculation is enabled by setting the CRCEN bit in the SPI_CFG1 register before the SPI is enabled (SPE = 1). The CRC value is then calculated using the CRC polynomial defined by the CRCPOLY register and CRC33_17 bit. When SPI is enabled, the CRC polynomial can be changed but only in case when there is no traffic on the bus.

The CRC computation is done, bit by bit, on the sampling clock edge defined by the CPHA and CPOL bits in the SPI_CR1 register. The calculated CRC value is checked automatically at the end of the data block defined by the SPI_CR2 register exclusively.

When a mismatch is detected between the CRC calculated internally on the received data and the CRC received from the transmitter, a CRCE flag is set to indicate a data corruption error. The right procedure for handling the CRC depends on the SPI configuration and the chosen transfer management.

CRC transfer management

Communication starts and continues normally until the last data frame must be sent or received in the SPI_DR register.

The length of the transfer must be defined by TSIZE. When the desired number of data is transacted, the TXCRC is transmitted and the data received on the line are compared to the RXCRC value.

No matter what is the CRCSIZE configuration, TSIZE cannot be set neither to 0xFFFF at full feature set instance nor to 0x3FF value at limited feature one if CRC is enabled.

In transmission, the CRC computation is frozen during CRC transaction and the TXCRC are transmitted, in a frame of length equal to the CRCSIZE field value.

In reception, the RXCRC is also frozen when desired number of data is transacted. Information to be compared with the RXCRC register content is then received in a frame of length equal to the CRCSIZE value.

Once the CRC frame is completed, an automatic check is performed comparing the received CRC value and the value calculated in the SPI_RXCRC register. The software has to check the CRCE flag in the SPI_SR register to determine if the data transfers were corrupted or not. Software clears the CRCE flag by writing 1 to the CRCEC.

The user takes no care about any flushing redundant CRC information, it is done automatically.

Resetting the SPI_TXCRC and SPI_RXCRC values

The SPI_TXCRC and SPI_RXCRC values are initialized automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode in order to transfer data without any interruption (several data blocks covered by intermediate CRC checking phases). Initialization patterns for receiver and transmitter can be configured either to zero or to all ones in dependency on setting bits TCRCINI and RCRCINI at SPI_CR1 register.

The CRC values are reset when the SPI is disabled.

59.6 SPI low-power modes

The SPI supports autonomous operation down to Stop mode, refer to *Section 59.4.16: Autonomous mode*.

Table 584. Effect of low-power modes on the SPI

Mode	Description
Sleep	No effect. SPI interrupts cause the device to exit Sleep mode.
Stop(1)	The SPI registers content is kept. If the autonomous mode is enabled at RCC configuration and SPI is clocked by an internal oscillator available in Stop mode, transfers are functional. The DMA requests are functional and the interrupts in these modes cause the device to exit Stop mode.
Standby(1)	The SPI instance not functional when this mode is powered down and must be reinitialized after exiting Standby mode. The SPI instance capable to work in this mode features DMA requests and wakeup capabilities.

1. Refer to *Section 59.3: SPI implementation* for information about wakeup from Stop mode support per instance as well as Standby mode availability. If an instance is not functional in a Stop mode, it must be disabled before entering this Stop mode.

59.7 SPI interrupts

Table 585 gives an overview of the SPI events capable to generate interrupts if enabled. Some of them feature wake up from Low-power mode capability, additionally. Most of them can be enabled and disabled independently while using specific interrupt enable control bits. The flags associated with the events are cleared by specific methods. Refer to description of SPI registers for more details about the event flags. When SPI is disabled, all the pending interrupt requests are blocked to prevent their propagation into the interrupt services, except the MODF interrupt request.

Table 585. SPI wakeup and interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Event clear method	Exit from Stop and Standby modes capability(1)(2)
SPI	TxFIFO ready to be loaded (space available for one data packet - FIFO threshold)	TXP	TXPIE	TXP cleared by hardware when TxFIFO contains less than FTHLV empty locations	Yes
	Data received in RxFIFO (one data packet available - FIFO threshold)	RXP	RXPIE	RXP cleared by hardware when RxFIFO contains less than FTHLV samples	Yes
	Both TXP and RXP active	DXP	DXPIE	When TXP or RXP are cleared	Yes
	Transmission Transfer Filled	TXTF	TXTFIE	Writing TXTFC to 1	No
	Underrun	UDR	UDRIE	Writing UDRC to 1	Yes
	Overrun	OVR	OVRIE	Writing OVRC to 1	Yes
	CRC Error	CRCE	CRCEIE	Writing CRCEC to 1	Yes
	TI Frame Format Error	TIFRE	TIFREIE	Writing TIFREC to 1	No
	Mode Fault	MODF	MODFIE	Writing MODFC to 1	No
	End Of Transfer (full transfer sequence completed - based on TSIZE value)	EOT	EOTIE	Writing EOTC to 1	Yes
	Master mode suspended	SUSP		Writing SUSPC to 1	Yes
	TxFIFO transmission complete (TxFIFO empty)	TXC(3)		TXC cleared by hardware when a transmission activity starts on the bus	No

1. All the interrupt events are capable to wakeup system from Sleep mode at each instance. For detailed information about instances capabilities to exit from concrete Stop and Standby mode refer to ‘functionalities depending on the working mode’ table.
2. Refer to *Section 59.3: SPI implementation* for information about Standby mode availability.
3. The TXC flag behavior depends on the TSIZE setting. When TSIZE>0, the flag fully follows the EOT one including its clearing by EOTC.

59.8 SPI registers

59.8.1 SPI control register 1 (SPI_CR1)

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.	IOLOCK
															rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCRCINI	RCRCINI	CRC33_17	SSI	HDDIR	CSUSP	CSTART	MASRX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPE
rw	rw	rw	rw	rw	w	rs	rw								rw

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

Bit 16 **IOLOCK**: locking the AF configuration of associated I/Os

This bit is set by software and cleared by hardware whenever the SPE bit is changed from 1 to 0.

0: AF configuration is not locked

1: AF configuration is locked

When this bit is set, SPI_CFG2 register content cannot be modified. This bit can be set when SPI is disabled only else it is write protected. It is cleared and cannot be set when MODF bit is set.

Bit 15 **TCRCINI**: CRC calculation initialization pattern control for transmitter

0: all zero pattern is applied

1: all ones pattern is applied

Bit 14 **RCRCINI**: CRC calculation initialization pattern control for receiver

0: All zero pattern is applied

1: All ones pattern is applied

Bit 13 **CRC33_17**: 32-bit CRC polynomial configuration

0: Full size (33-bit or 17-bit) CRC polynomial is not used

1: Full size (33-bit or 17-bit) CRC polynomial is used

Bit 12 **SSI:** internal SS signal input level

This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the peripheral SS input internally and the I/O value of the SS pin is ignored.

Bit 11 **HDDIR:** Rx/Tx direction at Half-duplex mode

In Half-Duplex configuration the HDDIR bit establishes the Rx/Tx direction of the data transfer. This bit is ignored in Full-Duplex or any Simplex configuration.

0: SPI is receiver

1: SPI is transmitter

Bit 10 **CSUSP:** master suspend request

This bit reads as zero.

In Master mode, when this bit is set by software, the CSTART bit is reset at the end of the current frame and communication is suspended. The user has to check SUSP flag to check end of the frame transaction.

The Master mode communication must be suspended (using this bit or keeping TXDR empty) before going to Low-power mode. Can be used in SPI or I2S mode.

After software suspension, SUSP flag must be cleared and SPI disabled and re-enabled before the next transaction starts.

Bit 9 **CSTART:** master transfer start

This bit can be set by software if SPI is enabled only to start an SPI communication. it is cleared by hardware when end of transfer (EOT) flag is set or when a transaction suspend request is accepted.

0: master transfer is at idle

1: master transfer is ongoing or temporary suspended by automatic suspend

In SPI mode, the bit is taken into account at master mode only. If transmission is enabled, communication starts or continues only if any data is available in the transmission FIFO.

Bit 8 **MASRX:** master automatic suspension in Receive mode

This bit is set and cleared by software to control continuous SPI transfer in master receiver mode and automatic management in order to avoid overrun condition.

0: SPI flow/clock generation is continuous, regardless of overrun condition. (data are lost)

1: SPI flow is suspended temporary on RxFIFO full condition, before reaching overrun condition. The SUSP flag is set when the SPI communication is suspended.

When SPI communication is suspended by hardware automatically, it could happen that few bits of next frame are already clocked out due to internal synchronization delay.

This is why, the automatic suspension is not quite reliable when size of data drops below 8 bits. In this case, a safe suspension can be achieved by combination with delay inserted between data frames applied when MIDI parameter keeps a non zero value; sum of data size and the interleaved SPI cycles should always produce interval at length of 8 SPI clock periods at minimum. After software clearing of the SUSP bit, the communication resumes and continues by subsequent bits transaction without any next constraint. Prior the SUSP bit is cleared, the user must release the RxFIFO space as much as possible by reading out all the data packets available at RxFIFO based on the RXP flag indication to prevent any subsequent suspension.

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

Bit 0 **SPE:** serial peripheral enable

This bit is set by and cleared by software.

0: Serial peripheral disabled.

1: Serial peripheral enabled

When SPE = 1, SPI data transfer is enabled, SPI_CFG1 and SPI_CFG2 configuration registers, CRCPOLY, UDRDR, part of SPI_AUTOCR register and IOLOCK bit in the SPI_CR1 register are write protected. They can be changed only when SPE = 0.

When SPE = 0 any SPI operation is stopped and disabled, all the pending requests of the events with enabled interrupt are blocked except the MODF interrupt request (but their pending still propagates the request of the spi_plck clock), the SS output is deactivated at master, the RDY signal keeps not ready status at slave, the internal state machine is reseted, all the FIFOs content is flushed, CRC calculation initialized, receive data register is read zero.

SPE is cleared and cannot be set when MODF error flag is active.

59.8.2 SPI control register 2 (SPI_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIZE[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 **TSIZE[15:0]**: number of data at current transfer

When these bits are changed by software, the SPI must be disabled.

Endless transaction is initialized when CSTART is set while zero value is stored at TSIZE.

TSIZE cannot be set to 0xFFFF respective 0x3FFF value when CRC is enabled.

Note: TSIZE[15:10] bits are reserved at limited feature set instances and must be kept at reset value.

59.8.3 SPI configuration register 1 (SPI_CFG1)

Address offset: 0x08

Reset value: 0x0007 0007

Content of this register is write protected when SPI is enabled, except TXDMAEN and RXDMAEN bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BPASS	MBR[2:0]			Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	CRCSIZE[4:0]				
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
TXDMAEN	RXDMAEN	Res.	Res.	Res.	Res.	UDRCFG	FTHLV[3:0]				DSIZE[4:0]				
rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BPASS:** bypass of the prescaler at master baud rate clock generator
0: bypass is disabled
1: bypass is enabled

Bits 30:28 **MBR[2:0]:** master baud rate prescaler setting
000: SPI master clock/2
001: SPI master clock/4
010: SPI master clock/8
011: SPI master clock/16
100: SPI master clock/32
101: SPI master clock/64
110: SPI master clock/128
111: SPI master clock/256
Note: MBR setting is considered at slave working at TI mode, too (see Section 59.5.1: TI mode).

Bits 27:23 Reserved, must be kept at reset value.

Bit 22 **CRCEN:** hardware CRC computation enable
0: CRC calculation disabled
1: CRC calculation enabled

Bit 21 Reserved, must be kept at reset value.

Bits 20:16 **CRCSIZE[4:0]:** length of CRC frame to be transacted and compared
Most significant bits are taken into account from polynomial calculation when CRC result is transacted or compared. The length of the polynomial is not affected by this setting.
00000: reserved
00001: reserved
00010: reserved
00011: 4-bits
00100: 5-bits
00101: 6-bits
00110: 7-bits
00111: 8-bits
.....
11101: 30-bits
11110: 31-bits
11111: 32-bits
The value must be set equal or multiply of data size (DSIZE[4:0]). Its maximum size corresponds to DSIZE maximum at the instance.
Note: The most significant bit at CRCSIZE bit field is reserved at the peripheral instances where data size is limited to 16-bit.

Bit 15 **TXDMAEN:** Tx DMA stream enable
0: Tx DMA disabled
1: Tx DMA enabled

Bit 14 **RXDMAEN:** Rx DMA stream enable
0: Rx-DMA disabled
1: Rx-DMA enabled

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **UDRCFG:** behavior of slave transmitter at underrun condition

0: slave sends a constant pattern defined by the user at the SPI_UDRDR register

1: Slave repeats lastly received data from master. When slave is configured at transmit only mode (COMM[1:0] = 01), all zeros pattern is repeated.

For more details see *Figure 706: Optional configurations of slave detecting underrun condition*.

Bits 8:5 **FTHLV[3:0]:** FIFO threshold level

Defines number of data frames at single data packet. Size of the packet should not exceed 1/2 of FIFO space.

0000: 1-data
0001: 2-data
0010: 3-data
0011: 4-data
0100: 5-data
0101: 6-data
0110: 7-data
0111: 8-data
1000: 9-data
1001: 10-data
1010: 11-data
1011: 12-data
1100: 13-data
1101: 14-data
1110: 15-data
1111: 16-data

SPI interface is more efficient if configured packet sizes are aligned with data register access parallelism:

– If SPI data register is accessed as a 16-bit register and DSIZE ≤ 8 bit, better to select FTHLV = 2, 4, 6.
– If SPI data register is accessed as a 32-bit register and DSIZE> 8 bit, better to select FTHLV = 2, 4, 6, while if DSIZE ≤ 8bit, better to select FTHLV = 4, 8, 12.

Note: FTHLV[3:2] bits are reserved at instances with limited set of features

Bits 4:0 **DSIZE[4:0]:** number of bits in at single SPI data frame

00000: not used
00001: not used
00010: not used
00011: 4-bits
00100: 5-bits
00101: 6-bits
00110: 7-bits
00111: 8-bits
.....
11101: 30-bits
11110: 31-bits
11111: 32-bits

Note: Maximum data size can be limited up to 16-bits at some instances. At instances with limited set of features, DSIZE2:0] bits are reserved and must be kept at reset state. DSIZE[4:3] bits then control next settings of data size:
00xxx: 8-bits
01xxx: 16-bits
10xxx: 24-bits
11xxx: 32-bits.

59.8.4 SPI configuration register 2 (SPI_CFG2)

Address offset: 0x0C

Reset value: 0x0000 0000

Content of this register is write protected when SPI is enabled or IOLOCK bit is set at SPI_CR1 register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFCNTR	SSOM	SSOE	SSIOP	Res.	SSM	CPOL	CPHA	LSBFRST	MASTER	SP[2:0]			COMM[1:0]		Res.
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
IOSWP	RDIOP	RDIOM	Res.	Res.	Res.	Res.	Res.	MIDI[3:0]				MSSI[3:0]			
rw	rw	rw						rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **AFCNTR:** alternate function GPIOs control
This bit is taken into account when SPE = 0 only
0: The peripheral takes no control of GPIOs while it is disabled
1: The peripheral keeps always control of all associated GPIOs
When SPI must be disabled temporary for a specific configuration reason (e.g. CRC reset, CPHA or HDDIR change) setting this bit prevents any glitches on the associated outputs configured at alternate function mode by keeping them forced at state corresponding the current SPI configuration.

Bit 30 **SSOM**: SS output management in Master mode
This bit is taken into account in Master mode when SSOE is enabled. It allows the SS output to be configured between two consecutive data transfers.
0: SS is kept at active level till data transfer is completed, it becomes inactive with EOT flag
1: SPI data frames are interleaved with SS non active pulses when MIDI[3:0]>1

Bit 29 **SSOE:** SS output enable
This bit is taken into account in Master mode only
0: SS output is disabled and the SPI can work in multi-master configuration
1: SS output is enabled. The SPI cannot work in a multi-master environment. It forces the SS pin at inactive level after the transfer is completed or SPI is disabled with respect to SSOM, MIDI, MSSI, SSIOP bits setting

Bit 28 **SSIOP**: SS input/output polarity
0: low level is active for SS signal
1: high level is active for SS signal

Bit 27 Reserved, must be kept at reset value.

Bit 26 **SSM:** software management of SS signal input
0: SS input value is determined by the SS PAD
1: SS input value is determined by the SSI bit
When master uses hardware SS output (SSM = 0 and SSOE = 1) the SS signal input is forced to not active state internally to prevent master mode fault error.

Bit 25 **CPOL:** clock polarity
0: SCK signal is at 0 when idle
1: SCK signal is at 1 when idle

Bit 24 **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

Bit 23 **LSBFRST:** data frame format
0: MSB transmitted first
Note: 1: LSB transmitted first

Bit 22 **MASTER:** SPI Master
0: SPI Slave
1: SPI Master

Bits 21:19 **SP[2:0]:** serial protocol
000: SPI Motorola
001: SPI TI
others: reserved, must not be used

Bits 18:17 **COMM[1:0]:** SPI Communication Mode
00: full-duplex
01: simplex transmitter
10: simplex receiver
11: half-duplex

Bit 16 Reserved, must be kept at reset value.

Bit 15 **IOSWP:** swap functionality of MISO and MOSI pins
0: no swap
1: MOSI and MISO are swapped
When this bit is set, the function of MISO and MOSI pins alternate functions are inverted. Original MISO pin becomes MOSI and original MOSI pin becomes MISO.
Note:

Bit 14 **RDIOP:** RDY signal input/output polarity
0: high level of the signal means the slave is ready for communication
1: low level of the signal means the slave is ready for communication

Bit 13 **RDIOM:** RDY signal input/output management
0: RDY signal is defined internally fixed as permanently active (RDIOP setting has no effect)
1: RDY signal is overtaken from alternate function input (at master case) or output (at slave case) of the dedicated pin (RDIOP setting takes effect)
Note: When DSIZE at the SPI_CFG1 register is configured shorter than 8-bit, the RDIOM bit must be kept at zero.

Bits 12:8 Reserved, must be kept at reset value.

Bits 7:4 **MIDI[3:0]**: master Inter-Data Idleness
Specifies minimum time delay (expressed in SPI clock cycles periods) inserted between two consecutive data frames in Master mode.
0000: no delay
0001: 1 clock cycle period delay
...
1111: 15 clock cycle periods delay
Note: This feature is not supported in TI mode.

Bits 3:0 **MSSI[3:0]**: Master SS Idleness
Specifies an extra delay, expressed in number of SPI clock cycle periods, inserted additionally between active edge of SS opening a session and the beginning of the first data frame of the session in Master mode when SSOE is enabled.
0000: no extra delay
0001: 1 clock cycle period delay added
...
1111: 15 clock cycle periods delay added
Note: This feature is not supported in TI mode.
To include the delay, the SPI must be disabled and re-enabled between sessions.

59.8.5 SPI interrupt enable register (SPI_IER)

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.	MODFIE	TIFREIE	CRCEIE	OVRIE	UDRIE	TXTFIE	EOTIE	DXPIE	TXPIE	RXPIE
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **MODFIE**: mode Fault interrupt enable
0: MODF interrupt disabled
1: MODF interrupt enabled

Bit 8 **TIFREIE**: TIFRE interrupt enable
0: TIFRE interrupt disabled
1: TIFRE interrupt enabled

Bit 7 **CRCEIE:** CRC error interrupt enable
0: CRC interrupt disabled
1: CRC interrupt enabled

Bit 6 **OVRIE:** OVR interrupt enable
0: OVR interrupt disabled
1: OVR interrupt enabled

Bit 5 **UDRIE:** UDR interrupt enable
0: UDR interrupt disabled
1: UDR interrupt enabled

Bit 4 **TXTFIE:** TXTFIE interrupt enable
0: TXTF interrupt disabled
1: TXTF interrupt enabled

Bit 3 **EOTIE:** EOT, SUSP and TXC interrupt enable
0: EOT/SUSP/TXC interrupt disabled
1: EOT/SUSP/TXC interrupt enabled

Bit 2 **DXPIE**: DXP interrupt enabled
DXPIE is set by software and cleared by TXTF flag set event.
0: DXP interrupt disabled
1: DXP interrupt enabled

Bit 1 **TXPIE:** TXP interrupt enable
TXPIE is set by software and cleared by TXTF flag set event.
0: TXP interrupt disabled
1: TXP interrupt enabled

Bit 0 **RXPIE:** RXP interrupt enable
0: RXP interrupt disabled
1: RXP interrupt enabled

59.8.6 SPI status register (SPI_SR)

Address offset: 0x14

Reset value: 0x0000 1002

All the flags of this register are not cleared automatically when the SPI is re-enabled. They require specific clearing access exclusively via the flag clearing register as noted in the bits descriptions below.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTSIZE[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
RXWNE	RXPLVL[1:0]		TXC	SUSP	Res.	MODF	TIFRE	CRCE	OVR	UDR	TXTF	EOT	DXP	TXP	RXP
r	r	r	r	r		r	r	r	r	r	r	r	r	r	r

Bits 31:16 **CTSIZE[15:0]**: number of data frames remaining in current TSIZE session

The value is not quite reliable when traffic is ongoing on bus or during autonomous operation in low-power mode.

Note: CTSIZE[15:0] bits are not available in instances with limited set of features.

Bit 15 **RXWNE:** RxFIFO word not empty

0: less than four bytes of RxFIFO space is occupied by data

1: at least four bytes of RxFIFO space is occupied by data

Note: This bit value does not depend on DSIZE setting and keeps together with RXPLVL[1:0] information about RxFIFO occupancy by residual data.

Bits 14:13 **RXPLVL[1:0]:** RxFIFO packing level

When RXWNE = 0 and data size is set up to 16-bit, the value gives number of remaining data frames persisting at RxFIFO.

00: no next frame is available at RxFIFO

01: 1 frame is available

10: 2 frames are available*

11: 3 frames are available*

Note: (*): Optional value when data size is set up to 8-bit only.

When data size is greater than 16-bit, these bits are always read as 00. In that consequence, the single data frame received at the FIFO cannot be detected neither by RWNE nor by RXPLVL bits if data size is set from 17 to 24 bits. The user then must apply other methods like TSIZE > 0 or FTHLV = 0.

Bit 12 **TXC:** TxFIFO transmission complete

The flag behavior depends on TSIZE setting.

When TSIZE = 0 the TXC is changed by hardware exclusively and it raises each time the TxFIFO becomes empty and there is no activity on the bus.

If TSIZE ≠ 0 there is no specific reason to monitor TXC as it just copies the EOT flag value including its software clearing. The TXC generates an interrupt when EOTIE is set.

0: current data transaction is still ongoing, data is available in TxFIFO or last frame transmission is on going.

1: last TxFIFO frame transmission complete

Bit 11 **SUSP:** suspension status

In Master mode, SUSP is set by hardware either as soon as the current frame is completed after CSUSP request is done or at master automatic suspend receive mode (MASRX bit is set at SPI_CR1 register) on RxFIFO full condition.

SUSP generates an interrupt when EOTIE is set.

This bit must be cleared prior SPI is disabled and this is done by writing 1 to SUSPC bit of SPI_IFCR exclusively.

0: SPI not suspended (Master mode active or other mode).

1: Master mode is suspended (current frame completed).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **MODF**: mode fault

0: no mode fault

1: mode fault detected. When MODF is set, SPE and IOLOCK bits of SPI_CR1 register are reset and their setting is blocked.

This bit is cleared by writing 1 to MODFC bit of SPI_IFCR exclusively.

Bit 8 **TIFRE**: TI frame format error

0: no TI Frame Error

1: TI frame error detected

This bit is cleared by writing 1 to TIFREC bit of SPI_IFCR exclusively.

Bit 7 **CRCE:** CRC error

0: no CRC error

1: CRC error detected

This bit is cleared when SPI is re-enabled or by writing 1 to CRCEC bit of SPI_IFCR optionally.

Bit 6 **OVR:** overrun

0: no overrun

1: overrun detected

This bit is cleared when SPI is re-enabled or by writing 1 to OVRC bit of SPI_IFCR optionally.

Bit 5 **UDR:** underrun

0: no underrun

1: underrun detected

This bit is cleared when SPI is re-enabled or by writing 1 to UDRC bit of SPI_IFCR optionally.

Note: In SPI mode, the UDR flag applies to Slave mode only. In I2S/PCM mode, (when available) this flag applies to Master and Slave mode

Bit 4 **TXTF:** transmission transfer filled

0: upload of TxFIFO is ongoing or not started

1: TxFIFO upload is finished

TXTF is set by hardware as soon as all of the data packets in a transfer have been submitted for transmission by application software or DMA, that is when TSIZE number of data have been pushed into the TxFIFO.

This bit is cleared by software write 1 to TXTFC bit of SPI_IFCR exclusively.

TXTF flag triggers an interrupt if TXTFIE bit is set.

TXTF setting clears the TXPIE and DXPIE masks so to off-load application software from calculating when to disable TXP and DXP interrupts.

Bit 3 **EOT:** end of transfer

EOT is set by hardware as soon as a full transfer is complete, that is when SPI is re-enabled or when TSIZE number of data have been transmitted and/or received on the SPI. EOT is cleared when SPI is re-enabled or by writing 1 to EOTC bit of SPI_IFCR optionally.

EOT flag triggers an interrupt if EOTIE bit is set.

If DXP flag is used until TXTF flag is set and DXPIE is cleared, EOT can be used to download the last packets contained into RxFIFO in one-shot.

0: transfer is ongoing or not started

1: transfer complete

In master, EOT event terminates the data transaction and handles SS output optionally. When CRC is applied, the EOT event is extended over the CRC frame transaction.

To restart the internal state machine properly, SPI is strongly suggested to be disabled and re-enabled before next transaction starts despite its setting is not changed.

Bit 2 **DXP**: duplex packet

0: TxFIFO is Full and/or RxFIFO is Empty

1: both TxFIFO has space for write and RxFIFO contains for read a single packet at least

DXP flag is set whenever both TXP and RXP flags are set regardless SPI mode.

Bit 1 **TXP:** Tx-Packet space available

0: In SPI mode, it must be interpreted as follow: there is not enough space to locate next data packet at TxFIFO

In I2S mode, it must be interpreted as follow: there is less than FTHLV free locations in the TxFIFO

1: In SPI mode, it must be interpreted as follow: TxFIFO has enough free location to host 1 data packet

In I2S mode, it must be interpreted as follow: there is FTHLV or more than FTHLV free locations in the TxFIFO

TXP flag is changed by hardware. It monitors overall space currently available at TxFIFO no matter if SPI is enabled or not. It must be checked once a complete data packet is stored at TxFIFO.

Bit 0 **RXP:** Rx-Packet available

0: In SPI mode, it must be interpreted as follow: RxFIFO is empty or a not complete data packet is received

In I2S mode, it must be interpreted as follow: RxFIFO level is lower than FTHLV

1: In SPI mode, it must be interpreted as follow: RxFIFO contains at least 1 data packet

In I2S mode, it must be interpreted as follow: RxFIFO level is higher or equal to FTHLV

RXP flag is changed by hardware. It monitors number of overall data currently available at RxFIFO if SPI is enabled. It must be checked once a data packet is completely read out from RxFIFO.

59.8.7 SPI interrupt/status flags clear register (SPI_IFCR)

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.	SUSPC	Res.	MODFC	TIFREC	CRCEC	OVRC	UDRC	TXTFC	EOTC	Res.	Res.	Res.
				w		w	w	w	w	w	w	w			

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SUSPC:** Suspend flag clear
Writing a 1 into this bit clears SUSP flag in the SPI_SR register

Bit 10 Reserved, must be kept at reset value.

Bit 9 **MODFC**: mode fault flag clear
Writing a 1 into this bit clears MODF flag in the SPI_SR register

Bit 8 **TIFREC**: TI frame format error flag clear
Writing a 1 into this bit clears TIFRE flag in the SPI_SR register

Bit 7 **CRCEC:** CRC error flag clear
Writing a 1 into this bit clears CRCE flag in the SPI_SR register

Bit 6 **OVRC:** overrun flag clear
Writing a 1 into this bit clears OVR flag in the SPI_SR register

Bit 5 **UDRC:** underrun flag clear
Writing a 1 into this bit clears UDR flag in the SPI_SR register

Bit 4 **TXTFC:** transmission transfer filled flag clear
Writing a 1 into this bit clears TXTF flag in the SPI_SR register

Bit 3 **EOTC:** end of transfer flag clear
Writing a 1 into this bit clears EOT flag in the SPI_SR register

Bits 2:0 Reserved, must be kept at reset value.

59.8.8 SPI autonomous mode control register (SPI_AUTOCR)

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

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **TRIGEN:** HW control of CSTART triggering enable
0: HW control disabled
1: HW control enabled
Note: if user cannot prevent trigger event during write, the TRIGEN must be changed when SPI is disabled

Bit 20 **TRIGPOL:** trigger polarity
0: trigger is active on raising edge
1: trigger is active on falling edge
Note: This bit can be written only when SPE = 0.

Bits 19:16 **TRIGSEL[3:0]**: trigger selection (refer *Section : Description of SPI interconnections*).
0000: spi_trg0 is selected
0001: spi_trg1 is selected
...
1111: spi_trg15 is selected
Note: these bits can be written only when SPE = 0.

Bits 15:0 Reserved, must be kept at reset value.

59.8.9 SPI transmit data register (SPI_TXDR)

Address offset: 0x20

Reset value: 0x0000 0000

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

Bits 31:0 **TXDR[31:0]:** transmit data register
The register serves as an interface with TxFIFO. A write to it accesses TxFIFO.
Note: data is always right-aligned. Unused bits are ignored when writing to the register, and read as zero when the register is read.
Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is written by single access.
halfword-wise (16 bit access) in this case 2 data-bytes or 1 halfword-data can be written by single access.
word-wise (32 bit access). In this case 4 data-bytes or 2 halfword-data or word-data can be written by single access.
Write access of this register less than the configured data size is forbidden.

59.8.10 SPI receive data register (SPI_RXDR)

Address offset: 0x30

Reset value: 0x0000 0000

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

Bits 31:0 **RXDR[31:0]:** receive data register

The register serves as an interface with RxFIFO. When it is read, RxFIFO is accessed.

Note: data is always right-aligned. Unused bits are read as zero when the register is read. Writing to the register is ignored.

Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is read by single access

halfword-wise (16 bit access) in this case 2 data-bytes or 1 halfword-data can be read by single access

word-wise (32 bit access). In this case 4 data-bytes or 2 halfword-data or word-data can be read by single access.

Read access of this register less than the configured data size is forbidden.

59.8.11 SPI polynomial register (SPI_CRCPOLY)

Address offset: 0x40

Reset value: 0x0000 0107

The content of this register is write protected when SPI is enabled.

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

Bits 31:0 **CRCPOLY[31:0]:** CRC polynomial register

This register contains the polynomial for the CRC calculation.

The default 9-bit polynomial setting 0x107 corresponds to default 8-bit setting of DSIZE. It is compatible with setting 0x07 used in other ST products with fixed length of the polynomial string, where the most significant bit of the string is always kept hidden.

Length of the polynomial is given by the most significant bit of the value stored in this register. It must be set greater than DSIZE. CRC33_17 bit must be set additionally with CRCPOLY register when DSIZE is configured to maximum 32-bit or 16-bit size and CRC is enabled (to keep polynomial length grater than data size).

Note: CRCPOLY[31:16] bits are reserved at instances with data size limited to 16-bit. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

59.8.12 SPI transmitter CRC register (SPI_TXCRC)

Address offset: 0x44

Reset value: 0x0000 0000

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

Bits 31:0 **TXCRC[31:0]:** CRC register for transmitter

When CRC calculation is enabled, the TXCRC[31:0] bits contain the computed CRC value of the subsequently transmitted bytes. CRC calculation is initialized when the CRCEN bit of SPI_CR1 is written to 1 or when a data block is transacted completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.

The number of bits considered at calculation depends on SPI_CRCPOLY register and CRCSIZE bits settings at SPI_CFG1 register.

Note: a read to this register when the communication is ongoing could return an incorrect value.

Note: TXCRC[31-16] bits are reserved at instances with data size limited to 16-bit. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

Note: The configuration of CRCSIZE bit field is not taken into account when the content of this register is read by software. No masking is applied for unused bits in this case.

59.8.13 SPI receiver CRC register (SPI_RXCRC)

Address offset: 0x48

Reset value: 0x0000 0000

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

Bits 31:0 **RXCRC[31:0]:** CRC register for receiver

When CRC calculation is enabled, the RXCRC[31:0] bits contain the computed CRC value of the subsequently received bytes. CRC calculation is initialized when the CRCEN bit of SPI_CR1 is written to 1 or when a data block is transacted completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register. The number of bits considered at calculation depends on SPI_CRCPOLY register and CRCSIZE bits settings at SPI_CFG1 register.

Note: a read to this register when the communication is ongoing could return an incorrect value.

RXCRC[31-16] bits are reserved at the peripheral instances with data size limited to 16-bit. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

Note: The configuration of CRCSIZE bit field is not taken into account when the content of this register is read by software. No masking is applied for unused bits in this case.

59.8.14 SPI underrun data register (SPI_UDRDR)

Address offset: 0x4C

Reset value: 0x0000 0000

Content of this register is write protected when SPI is enabled.

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

Bits 31:0 **UDRDR[31:0]:** data at slave underrun condition

The register is taken into account in Slave mode and at underrun condition only. The number of bits considered depends on DSIZE bit settings of the SPI_CFG1 register. Underrun condition handling depends on setting UDRCFG bit at SPI_CFG1 register.

Note: UDRDR[31-16] bits are reserved at the peripheral instances with data size limited to 16-bit. There is no constraint when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

59.8.15 SPI register map

Table 586. SPI 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	SPI_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOLOCK	TCRCINI	RCRCINI	CRC33_17	SSI	HDDIR	CSUSP	CSTART	MASRX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPE
	Reset value																0	0	0	0	0	0	0	0	0								0

Table 586. SPI 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
0x04	**SPI_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSIZE[15:0]															
	Reset value																	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	0	0	0	0	0	0	0	0	0	0
0x08	**SPI_CFG1**	BPASS	MBR[2:0]			Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	CRCSIZE[4:0]					TXDMAEN	RXDMAEN	Res.	Res.	Res.	Res.	UDRCFG	FTHLV[3:0]				DSIZE[4:0]				
	Reset value	0	0	0	0						0		1	1	1	1	1	0	0					0	0(1)	0(1)	0	0	0	0(1)	1(1)	1(1)	1(1)
0x0C	**SPI_CFG2**	AFCNTR	SSOM	SSOE	SSIOP	Res.	SSM	CPOL	CPHA	LSBFRST	MASTER	SP[2:0]			COMM [1:0]		Res.	IOSWP	RDIOP	RDIOM	Res.	Res.	Res.	Res.	Res.	MIDI[3:0]				MSSI[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
0x10	**SPI_IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODFIE	TIFREIE	CRCEIE	OVRIE	UDRIE	TXTFIE	EOTIE	DXPIE	TXPIE	RXPIE
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x14	**SPI_SR**	CTSIZE[15:0](1)																RXWNE	RXPLVL [1:0]		TXC	SUSP	Res.	MODF	TIFRE	CRCE	OVR	UDR	TXTF	EOT	DPXP	TXP	RXP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0		0	0	0	0	0	0	0	0	1	0
0x18	**SPI_IFCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSPC	Res.	MODFC	TIFREC	CRCEC	OVRC	UDRC	TXTFC	EOTC	Res.	Res.	Res.
	Reset value																					0		0	0	0	0	0	0	0			
0x1C	**SPI_AUTOCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGEN	TRIGPOL	TRIGSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0	0	0	0	0	0																
0x20	**SPI_TXDR**	TXDR[31:16]																TXDR[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
0x24-0x2C	Reserved	Reserved																															
0x30	**SPI_RXDR**	RXDR[31:16]																RXDR[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 - 0x3C	Reserved	Reserved																															
0x40	**SPI_CRCPOLY**	CRCPOLY[31:16](2)																CRCPOLY[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	1	0	0	0	0	0	1	1	1
0x44	**SPI_TXCRC**	TXCRC[31:16](2)																TXCRC[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
0x48	**SPI_RXCRC**	RXCRC[31:16](2)																RXCRC[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
0x4C	**SPI_UDRDR**	UDRDR[31:16](2)																UDRDR[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

1. The bit field is reserved for instances with limited set of features and it must be kept at reset value. For more details, refer to the concrete register description in *Section 59.8: SPI registers*.
2. The bits 31-16 are reserved for the peripheral instances with data size limited to 16-bit. There is no constrain when the 32-bit access is applied at these addresses. The bits 31-16, when reserved, are always read to zero while any write to them is ignored.

Refer to *Section 2.3* for the register boundary addresses.

60 Serial audio interface (SAI)

60.1 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.

60.2 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.

60.3 SAI implementation

Table 587. STM32U575/585 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)	-

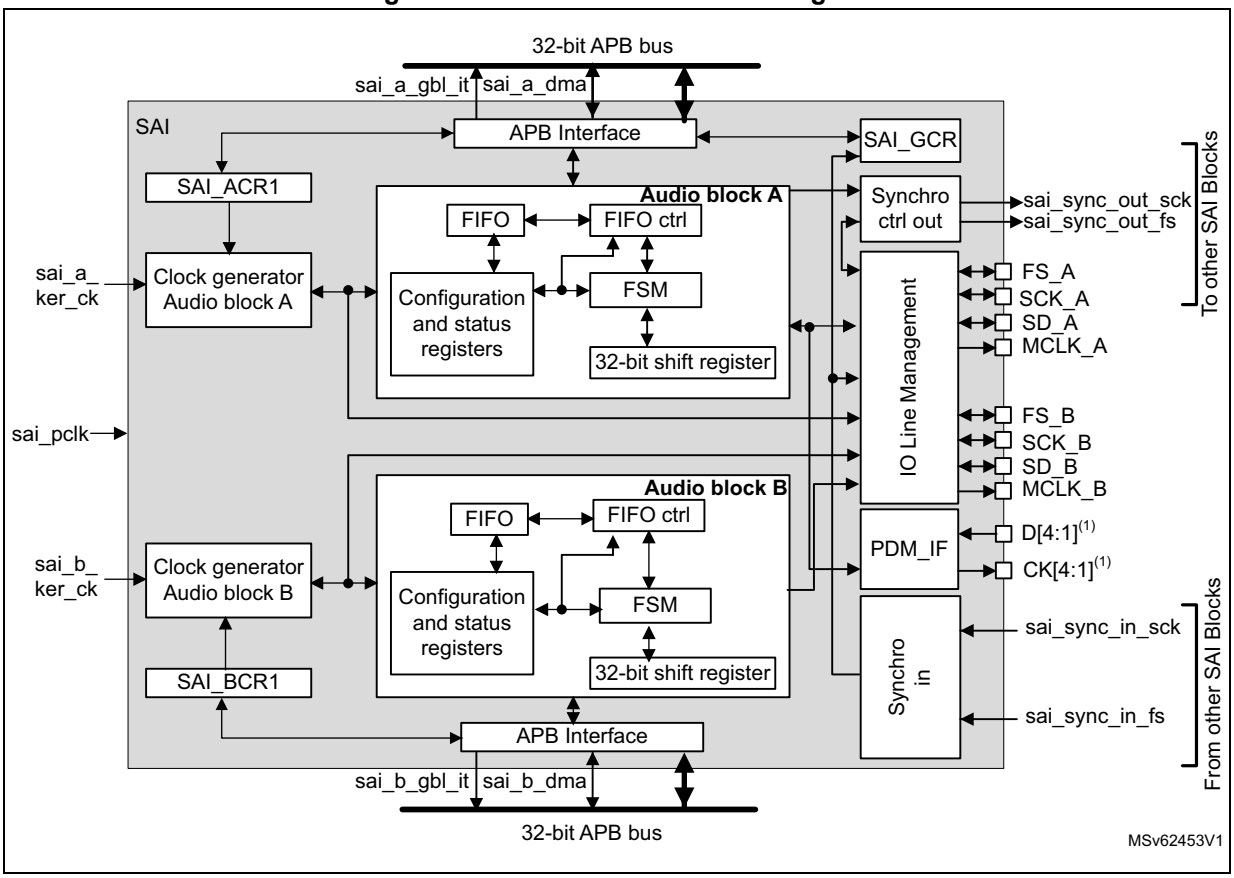
1. 'X' = supported, '-' = not supported.
2. Only signals D[3:1], and CK[2:1] are available.

60.4 SAI functional description

60.4.1 SAI block diagram

Figure 707 shows the SAI block diagram while *Table 588* and *Table 589* list SAI internal and external signals.

Figure 707. SAI functional block diagram



1. These signals might not be available for all SAI instances. Refer to *Section 60.3: 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 is an SAI output even if the SAI (link controller) is set-up to consume the SCK clock (and so to be in Slave mode).

Note: For ease of reading of this section, the notation SAI_x refers to SAI_A or SAI_B, where 'x' represents the SAI A or B subblock.

60.4.2 SAI pins and internal signals

Table 588. 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 589. 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. Refer to *Section 60.3: SAI implementation* for details.

60.4.3 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.

60.4.4 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 590. External synchronization selection

Block instance	SYNCIN = 1	SYNCIN = 0
SAI1	SAI2 sync.	Reserved
SAI2	Reserved	SAI1 sync.

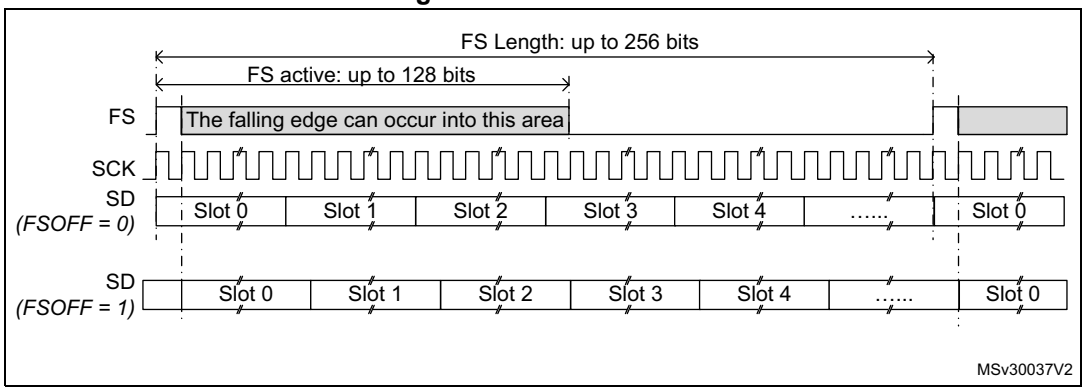
60.4.5 Audio data size

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.

60.4.6 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 708* illustrates this flexibility.

Figure 708. 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 60.4.8: 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 ongoing audio frame. In this case an error is generated. For more details refer to *Section 60.4.14: 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 60.4.14: 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 60.4.14: 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 enable the configuration of 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 enables to choose one of the two configurations.

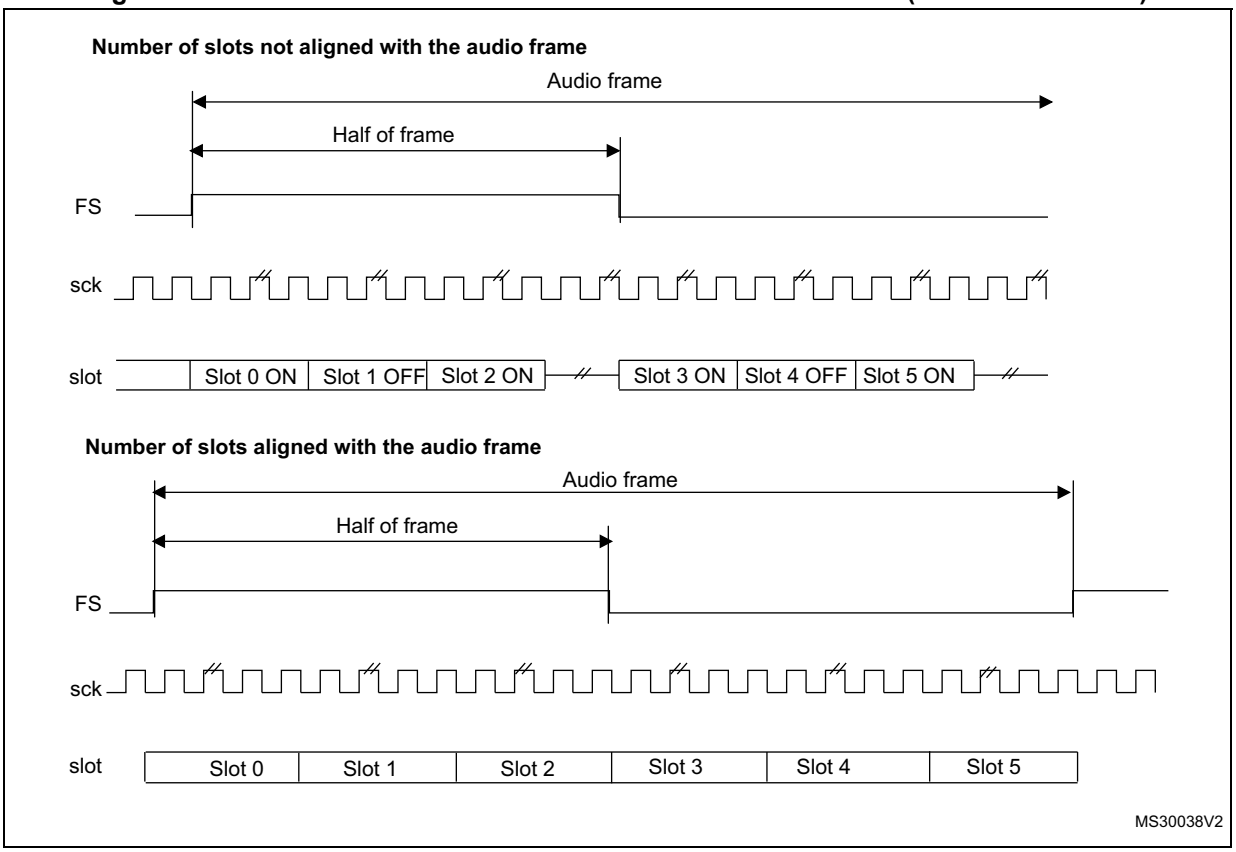
FS signal role

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 709. FS role is start of frame + channel side identification (FSDEF = TRIS = 1)

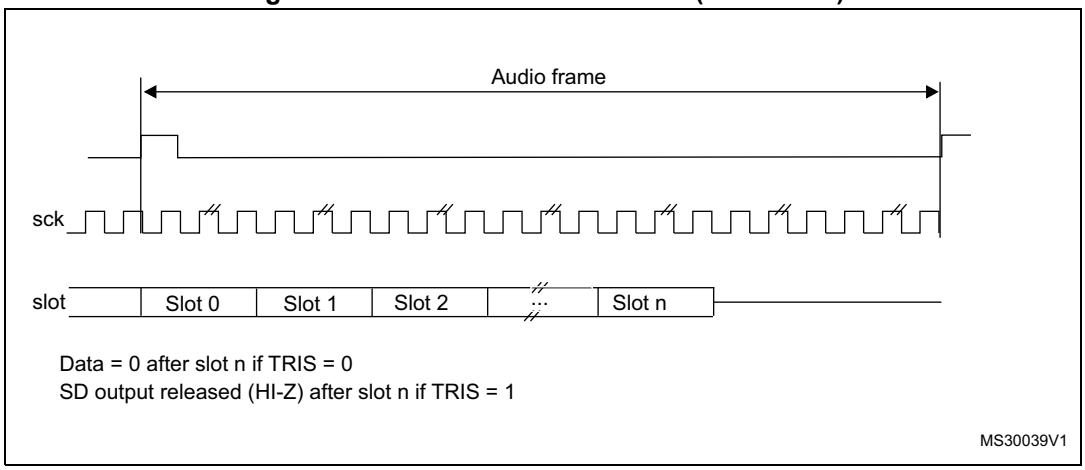


1. The frame length should be even.

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 710. 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.

60.4.7 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.

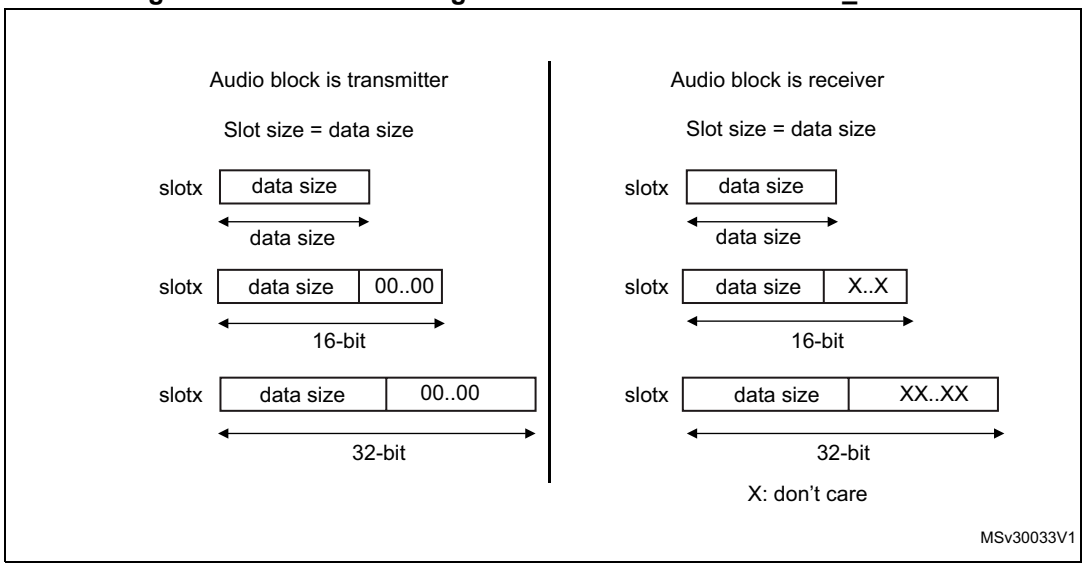
For AC'97 protocol or SPDIF (when bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the number of slots is automatically set to target the protocol specification, and the value of NBSLOT[3:0] is ignored.

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.

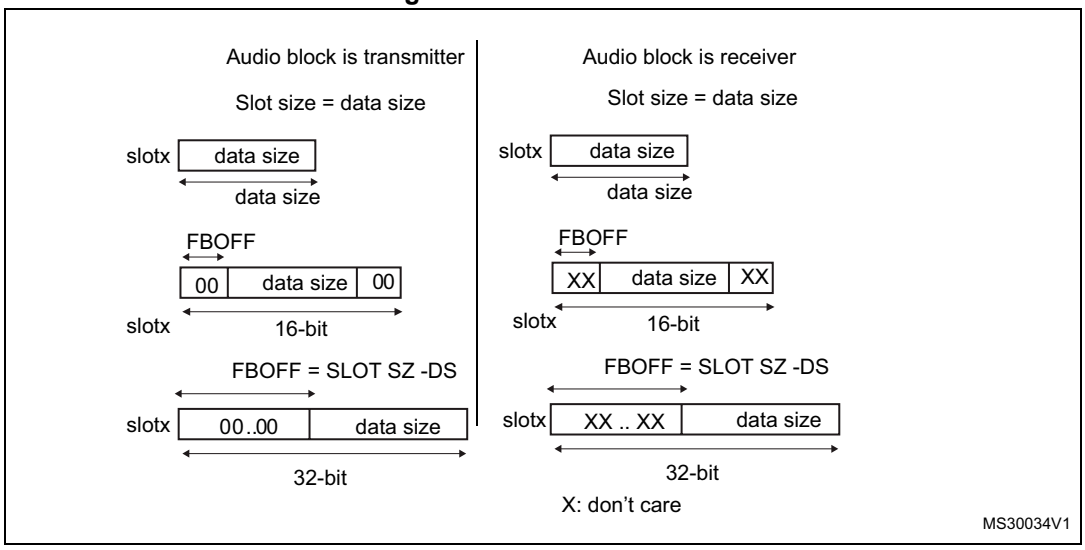
The slot size is also configurable as shown in *Figure 711*. The size of the slots is selected by setting SLOTSZ[1:0] bits in the SAI_xSLOTR register. The size is applied identically for each slot in an audio frame.

Figure 711. 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 712. 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 60.4.11: AC'97 link controller*.

60.4.8 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 enables the generation of 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 591 shows MCLK_x activation conditions.

Table 591. 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)

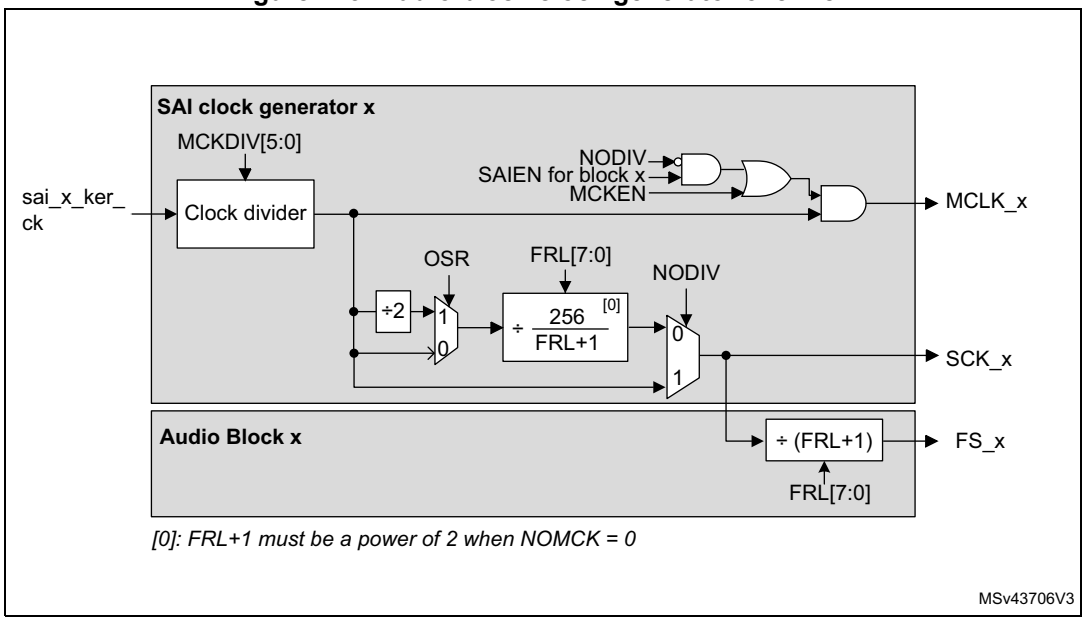
The clock generator provides the bit clock (SCK_x) when the audio block is defined as Master. The frame synchronization (FS_x) is also derived from the signals provided by the clock generator.

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 713 illustrates the architecture of the audio block clock generator.

Figure 713. 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_x frequency 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 592 gives programming examples for 48, 96 and 192 kHz.

Table 592. 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.

60.4.9 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.

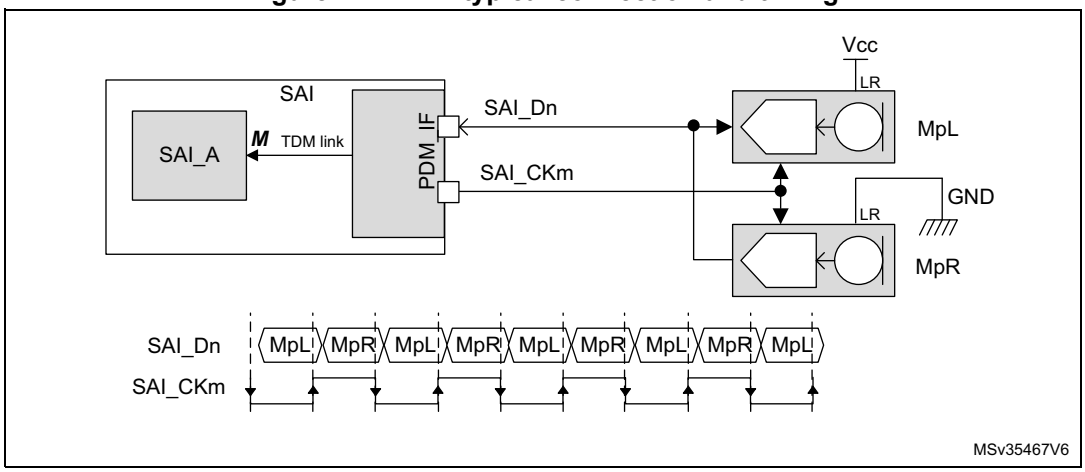
The FIFO pointers can be reinitialized when the SAI is disabled by setting bit FFLUSH in the SAI_xCR2 register. If FFLUSH is set when the SAI is enabled the data present in the FIFO are lost automatically.

60.4.10 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. Depending on product implementation, less microphones can be supported (refer to *Section 60.3: SAI implementation*).

Figure 714 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 714. 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]).

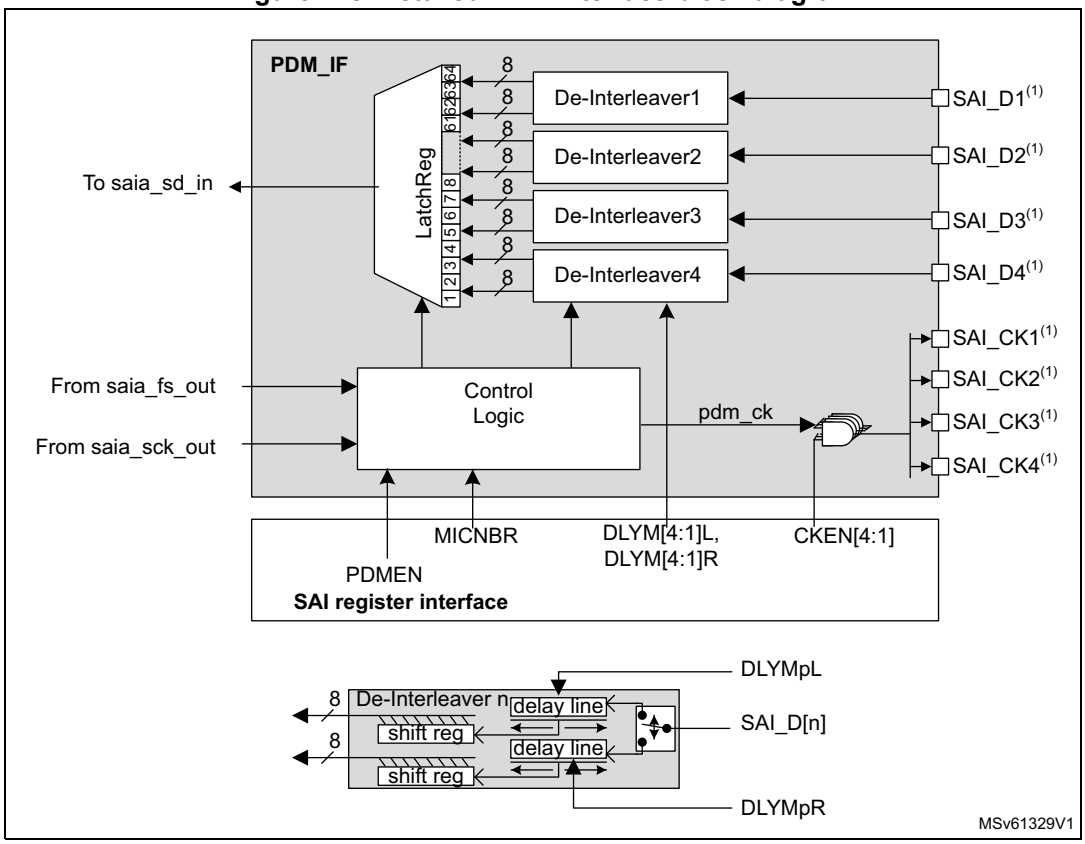
The data processing sequence into the PDM is the following:

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 715 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 715. Detailed PDM interface block diagram



1. These signals might not be available in all SAI instances. Refer to *Section 60.3: SAI implementation* for details.
2. **n** refers to the number of data lines and **p** to the number of microphone pairs.

The PDM interface can be enabled through the PDMEN bit in SAI_PDMCR register. However the PDM interface must be enabled prior to enabling SAI_A block.

To reduce the memory footprint, the user can select the amount of microphones the application needs. This can be done through MICNBR[1:0] bits. It is possible to select

between 2,4,6 or 8 microphones. For example, if the application is using 3 microphones, the user has to select 4.

Enabling the PDM interface

To enable the PDM interface, follow the sequence below:

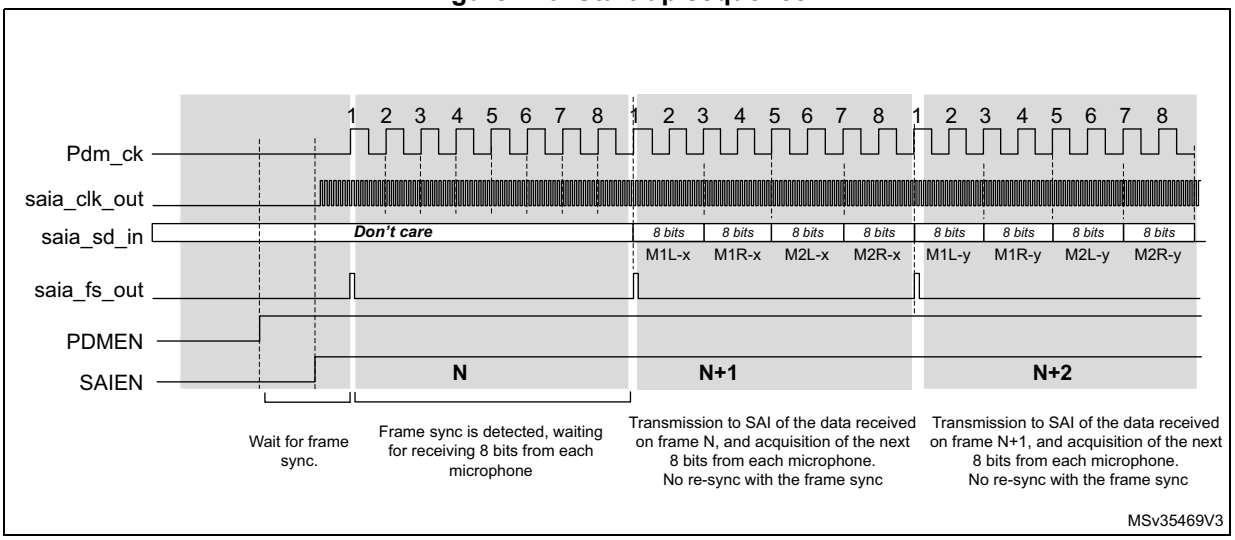
1. Configure SAI_A in TDM master mode (see *Table 593*).
2. Configure the PDM interface as follows:
   a) Define the number of digital microphones via MICNBR.
   b) Enable the bitstream clock needed in the application by setting the corresponding bits on CKEN to 1.
3. Enable the PDM interface, via PDMEN 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 716 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 716. 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.

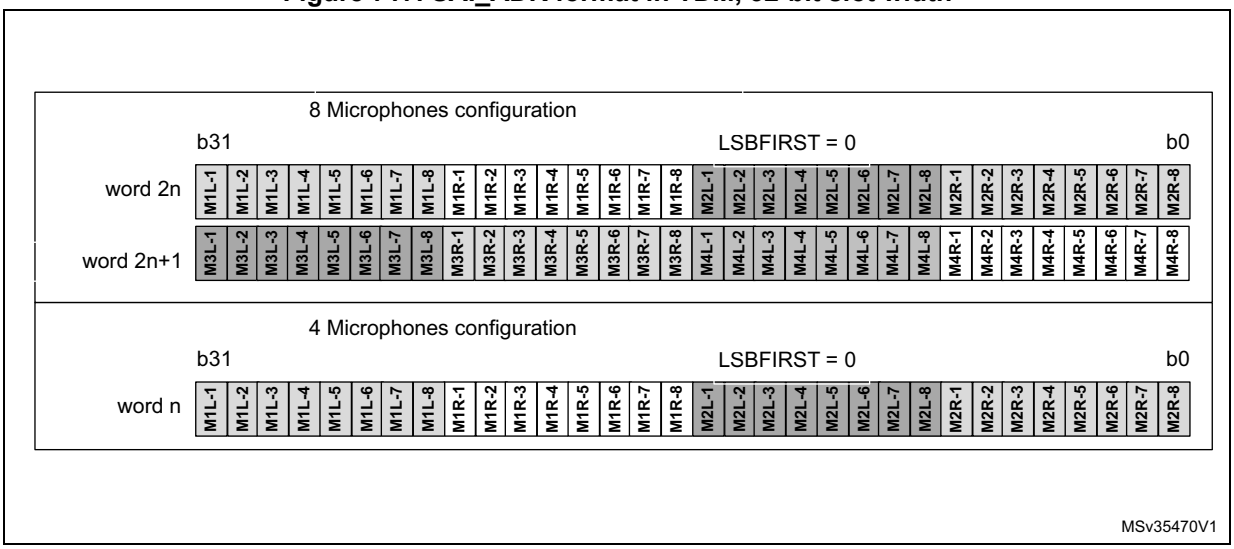
SAI_ADR data format example

- **32-bit slot width** (DS = 0b111 and SLOTSZ = 0). Refer to *Figure 717*.

  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 717. SAI_ADR format in TDM, 32-bit slot width

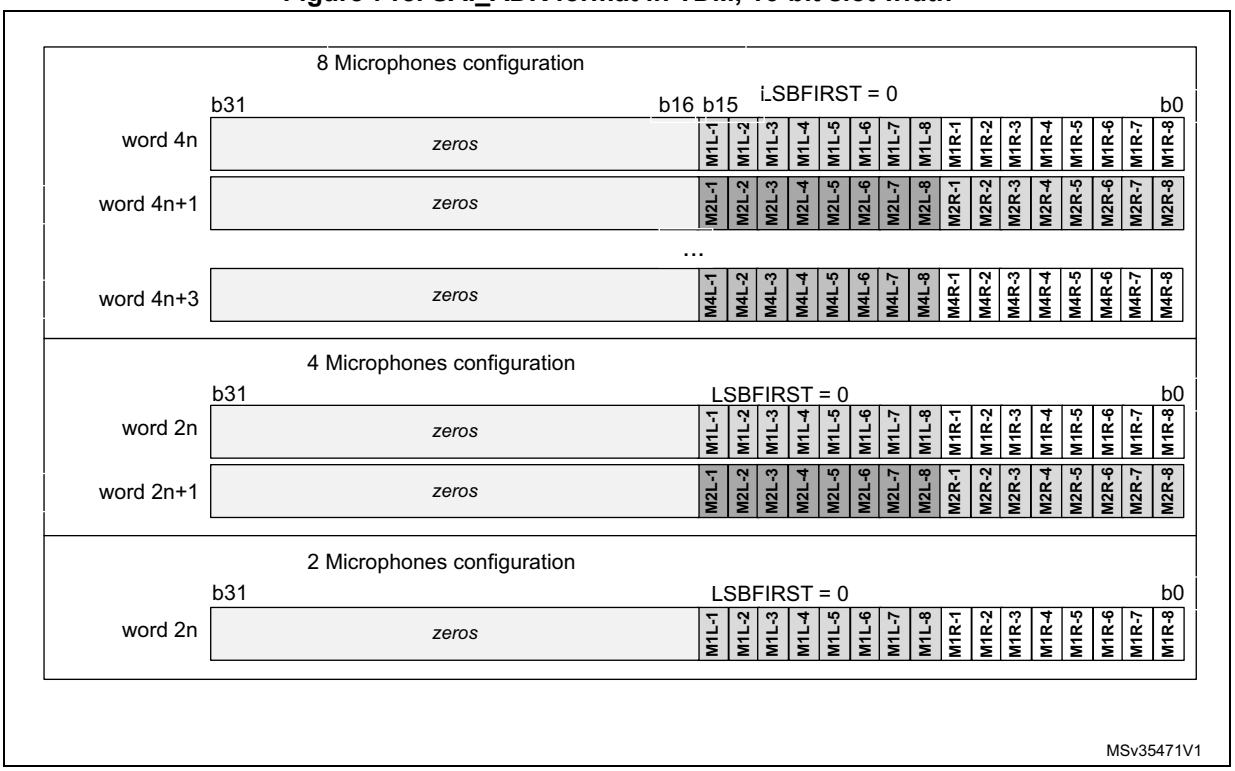


- **16-bit slot width** (DS = 0b100 and SLOTSZ = 0). Refer to *Figure 718*.

  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 718. SAI_ADR format in TDM, 16-bit slot width

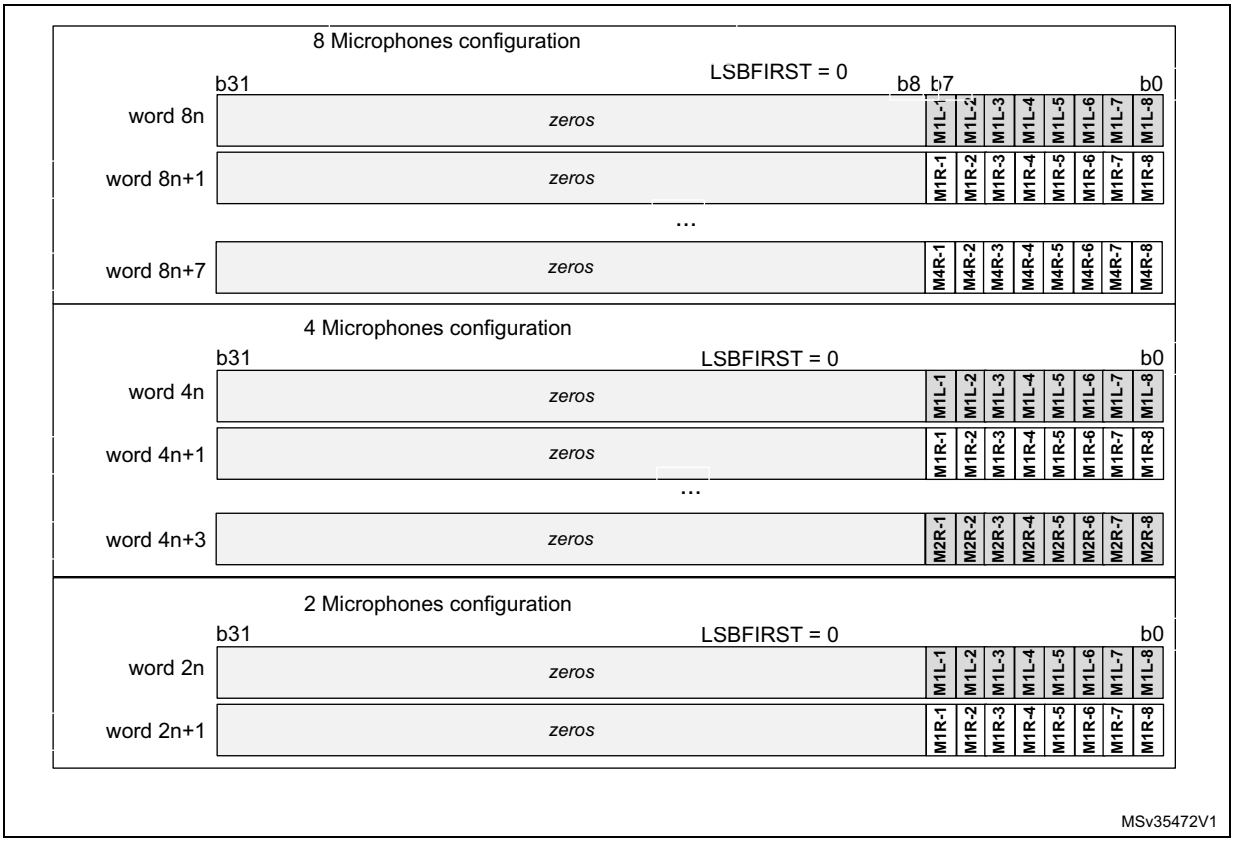


- **Using a 8-bit slot width** (DS = 0b010 and SLOTSZ = 0). Refer to *Figure 719*.

  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 719. 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 593* to ensure a good connection with the PDM interface.

Table 593. 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 594*.
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 594*.
FSALL	0	Pulse width is one bit clock cycle
FSDEF	0	FS signal is a start of frame

Table 593. 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 594*.
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 594*.

Adjusting the bitstream clock rate

To properly program the SAI TDM interface, the user application must take into account the settings given in *Table 593*, 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 60.6.18*)

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 594. TDM frame configuration examples(1)(2)

Microphone sampling rate	Nber of microphones	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 593: TDM settings* for additional information on TDM configuration. The sai_a_ker_ck clock frequency provided to the SAI must be a multiple of the SCK_A frequency, and MCKDIV should be programmed accordingly.
2. The above sai_a_ker_ck frequencies are given as examples only. Refer to section *Reset and clock controller (RCC) to check if they can be generated on the device.*
3. 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.

60.4.11 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.

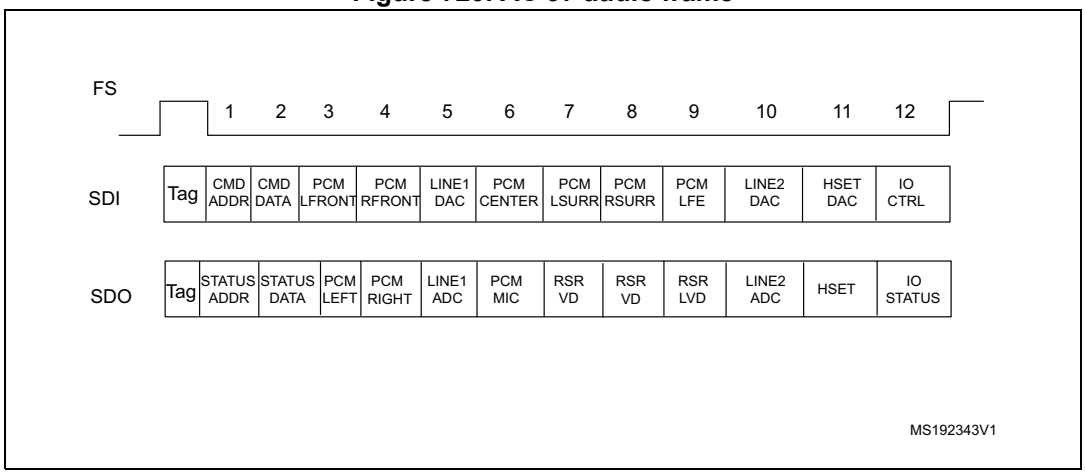
To select this protocol, set PRTCFG[1:0] bits in the SAI_xCR1 register to 10. When AC’97 mode is selected, only data sizes of 16 or 20 bits can be used, otherwise the SAI behavior is not guaranteed.

- 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 720 shows an AC’97 audio frame structure.

Figure 720. 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.

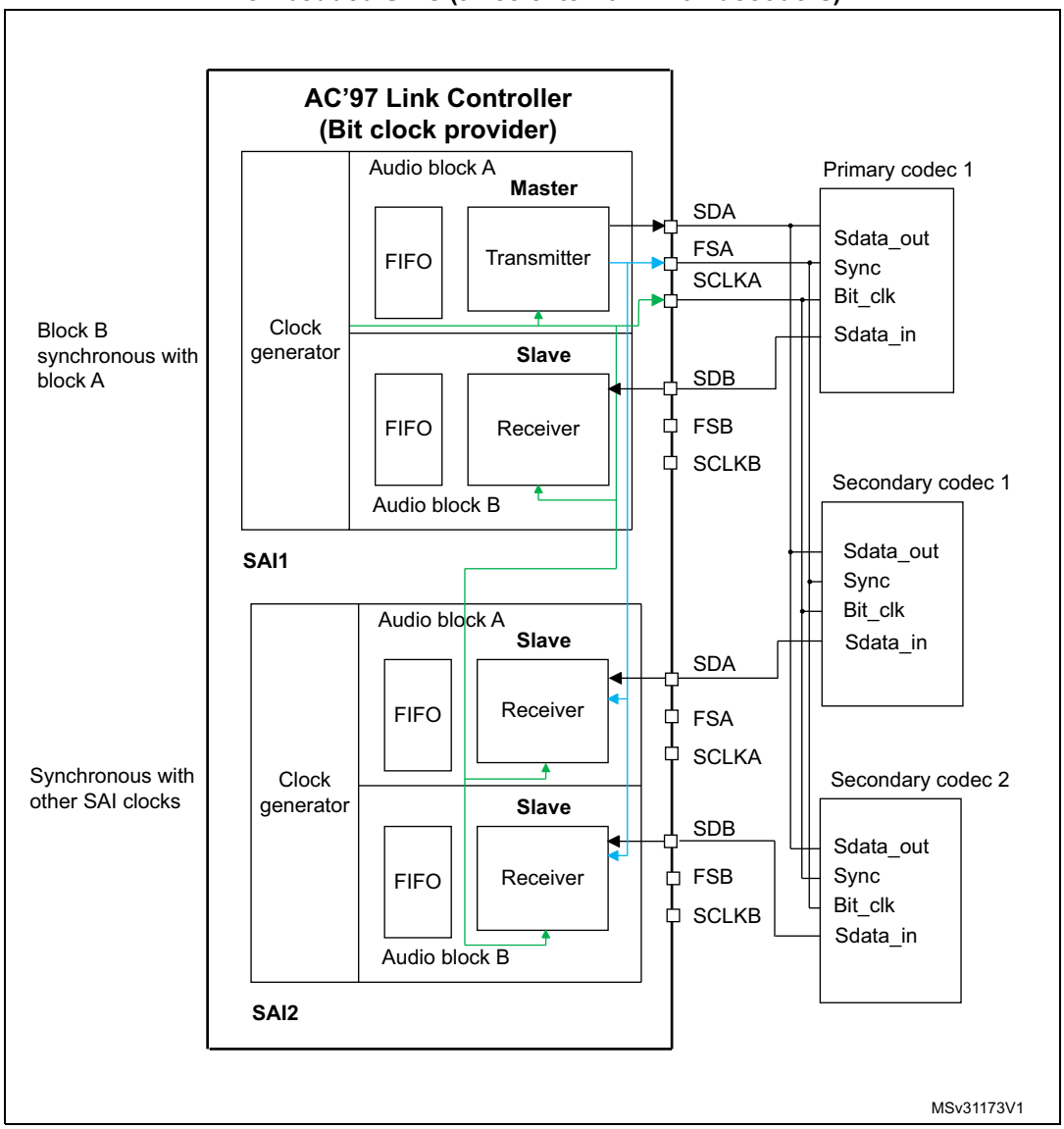
One SAI can be used to target an AC’97 point-to-point communication.

Using two SAIs (for devices featuring two embedded SAIs) enables the control of three external AC’97 decoders as illustrated in *Figure 721*.

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 721. 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 60.4.8: SAI clock generator* must be used with FRL = 255, in order to generate the proper frame rate (F_{FS_x}).

60.4.12 SPDIF output

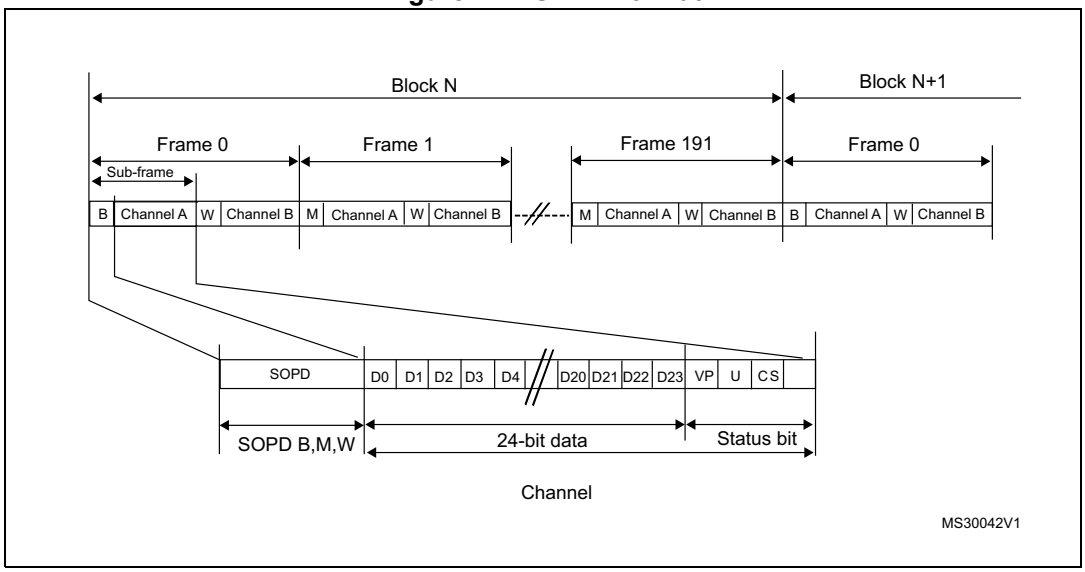
The SPDIF interface is available in transmitter mode only. It supports the audio IEC60958.

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 722*.

Figure 722. 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 595*). The next 28 bits of channel information are composed of 24 bits data + 4 status bits.

Table 595. 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:

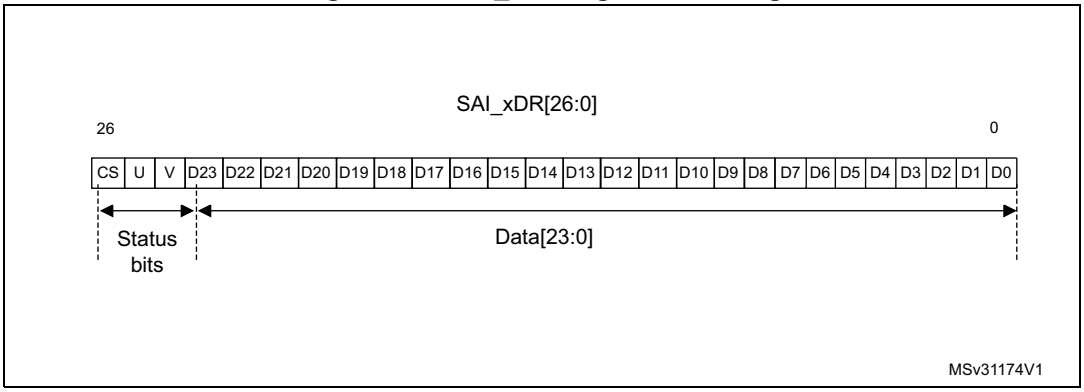
- SAI_xDR[26:24] contain the Channel status, User and Validity bits.
- SAI_xDR[23:0] contain the 24-bit data for the considered channel.

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 723. 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 596*.

Table 596. 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 597. 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.

60.4.13 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 724*. 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).

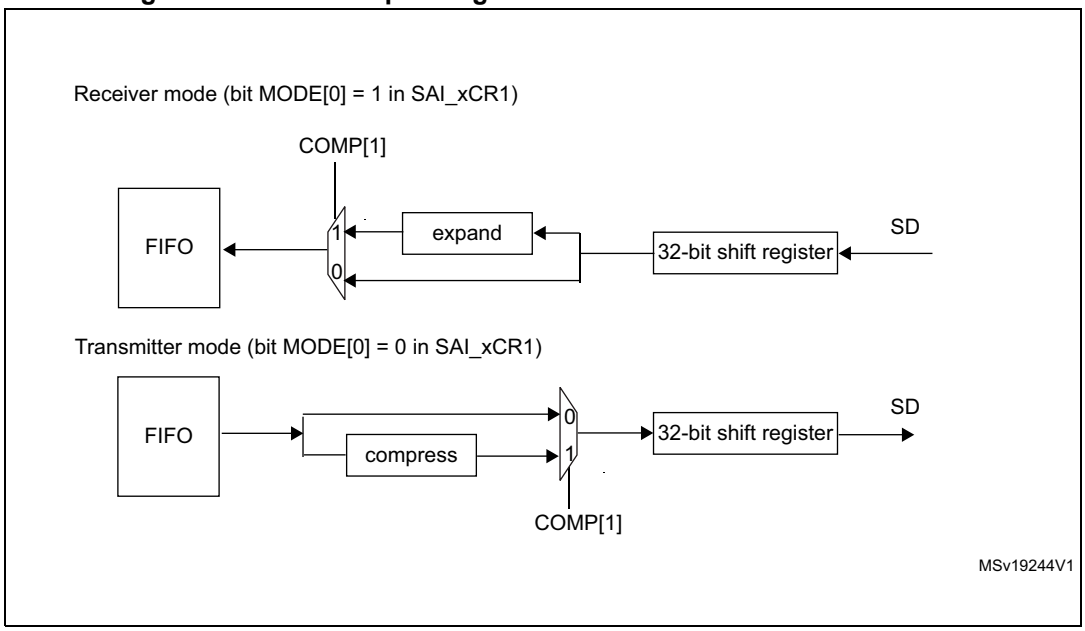
The European companding standard is A-Law and supports 13 bits of dynamic range (COMP[1:0] = 11 in the SAI_xCR2 register).

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 724. 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).

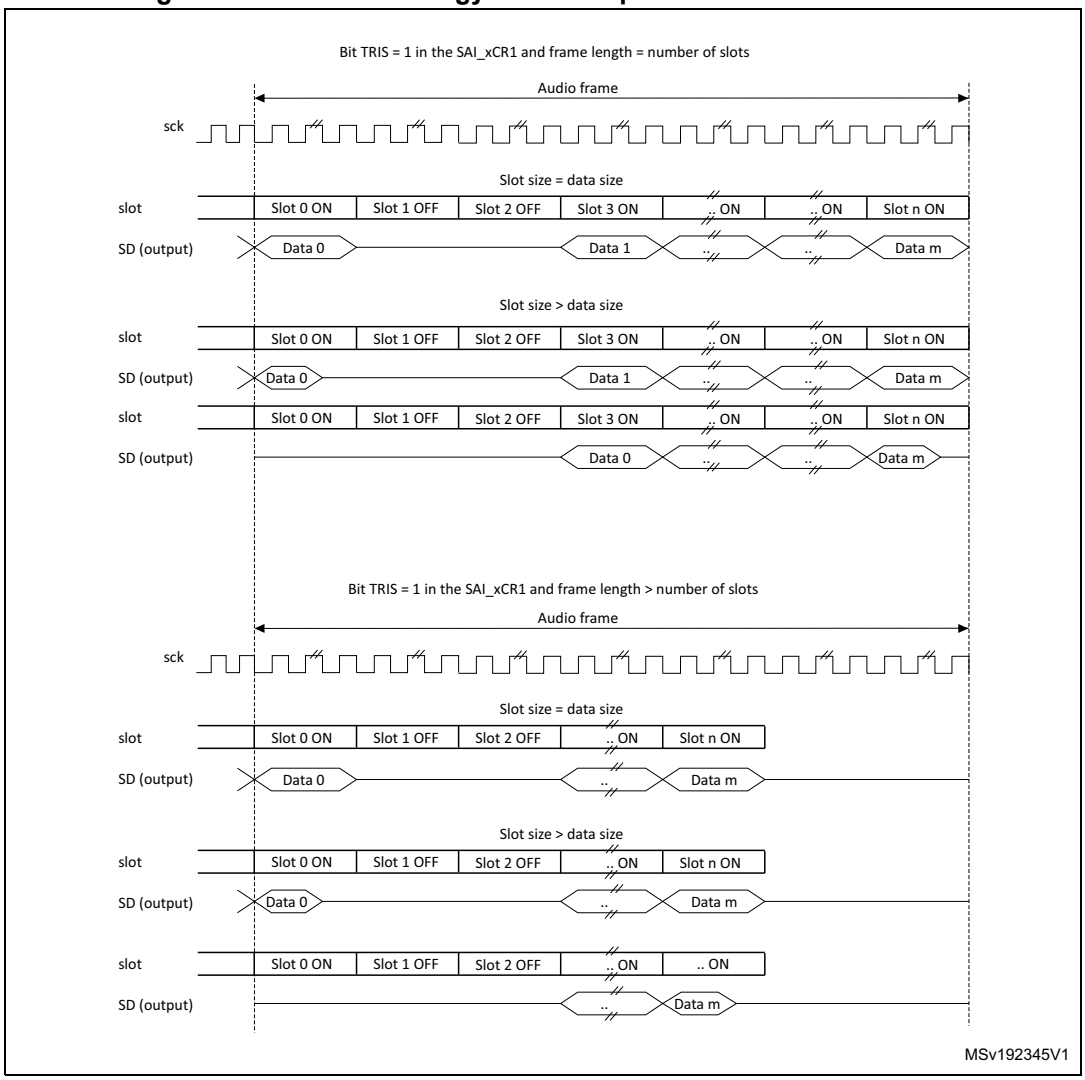
- Either the SAI forces 0 on the SD output line when an inactive slot is transmitted, or
- The line is released in HI-z state at the end of the last bit of data transferred, to release the line for other transmitters connected to this node.

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.

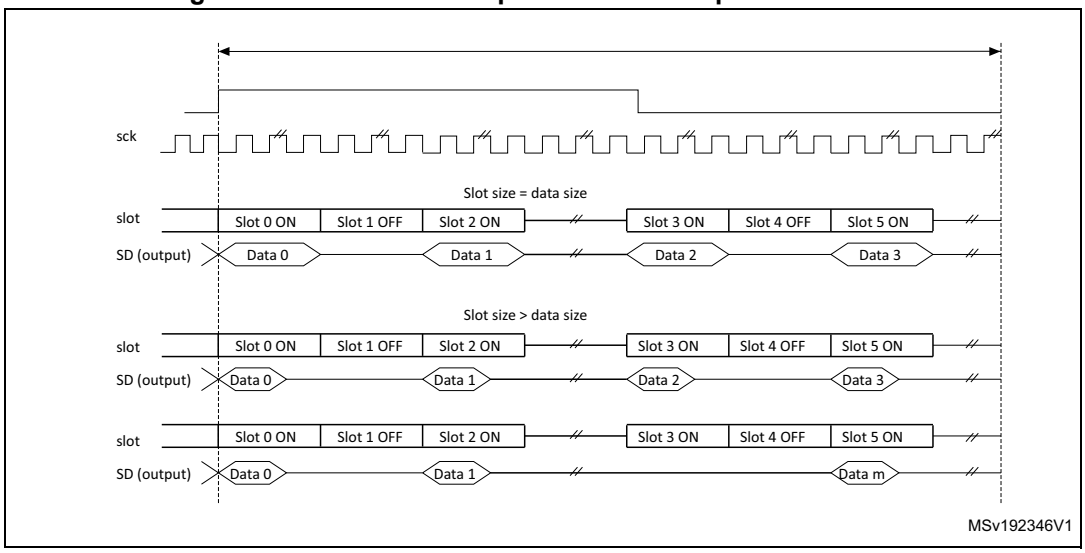
Figure 725 illustrates these behaviors.

Figure 725. 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 726* (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 726. 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 725* and *Figure 726* are replaced by a drive with a value of 0.

60.4.14 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.

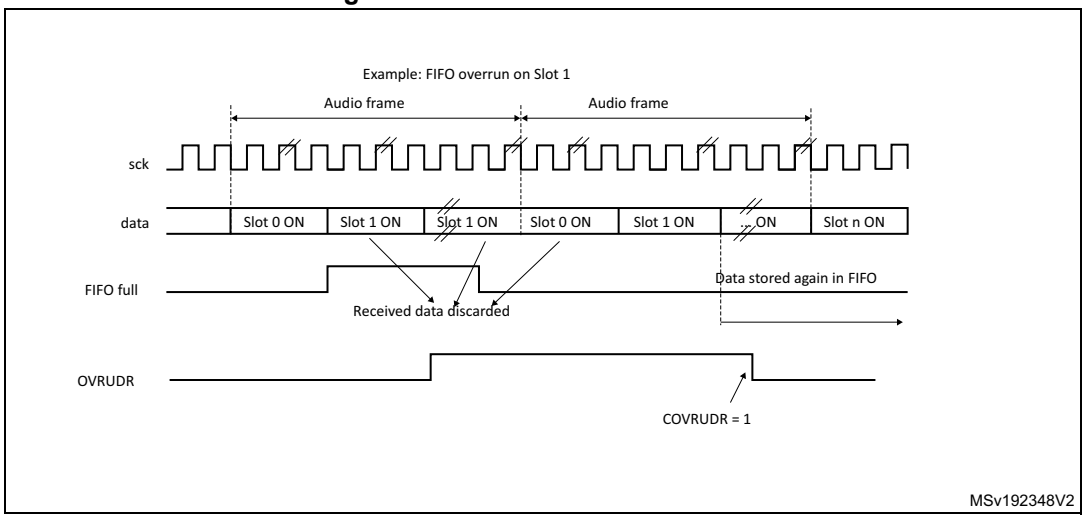
The overrun or underrun errors share the same bit since an audio block can be either receiver or transmitter and each audio block in a given SAI has its own SAI_xSR register.

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 727*).

The OVRUDR flag is cleared when COVRUDR bit is set in the SAI_xCLRFR register.

Figure 727. Overrun detection error



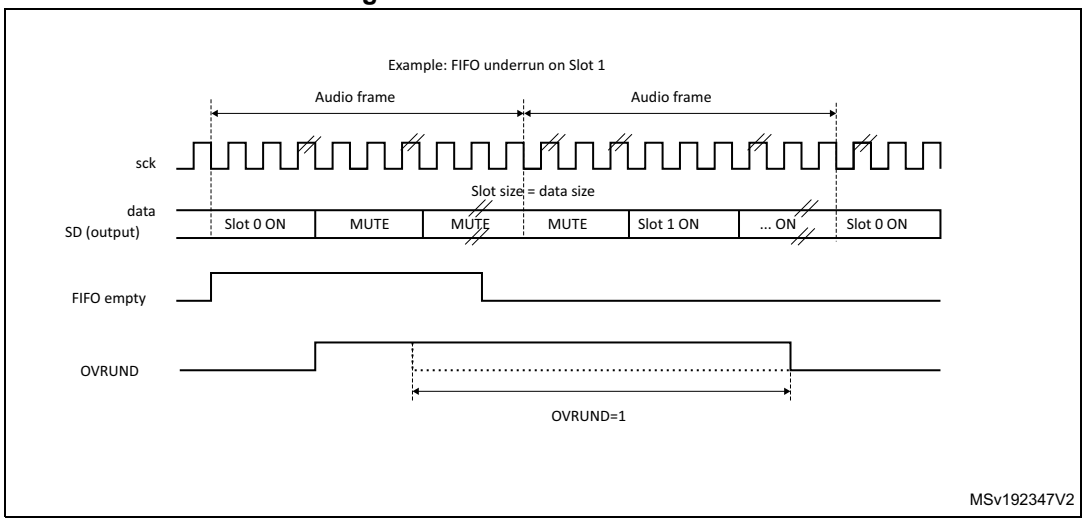
Underrun

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 728*). 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 728. 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.

60.4.15 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.

60.4.16 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 60.4.9: 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.

60.5 SAI interrupts

The SAI supports 7 interrupt sources as shown in *Table 598*.

Table 598. 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 60.4.9: 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.

60.6 SAI registers

The peripheral registers have to be accessed by words (32 bits).

60.6.1 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 590: 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).

60.6.2 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 60.4.8: 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 enables to control 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 enables 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).

60.6.3 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 60.4.8: 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 enables to control 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 enables 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.

60.6.4 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

60.6.5 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

60.6.6 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.

60.6.7 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.

60.6.8 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.

60.6.9 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.

60.6.10 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.

60.6.11 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.

60.6.12 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 60.4.6: 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.

60.6.13 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 60.4.6: 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.

60.6.14 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.

60.6.15 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.

60.6.16 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.

60.6.17 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.

60.6.18 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.	CKEN4	CKEN3	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]		Res.	Res.	Res.	PDMEN
				rw	rw	rw	rw			rw	rw				rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **CKEN4**: Clock enable of bitstream clock number 4
This bit is set and cleared by software.
0: SAI_CK4 clock disabled
1: SAI_CK4 clock enabled

Note: It is not recommended to configure this bit when PDMEN = 1.
SAI_CK4 might not be available for all SAI instances. Refer to Section 60.3: SAI implementation for details.

Bit 10 **CKEN3**: Clock enable of bitstream clock number 3
This bit is set and cleared by software.
0: SAI_CK3 clock disabled
1: SAI_CK3 clock enabled
Note: It is not recommended to configure this bit when PDMEN = 1.
SAI_CK3 might not be available for all SAI instances. Refer to Section 60.3: SAI implementation for details.

Bit 9 **CKEN2**: Clock enable of bitstream clock number 2
This bit is set and cleared by software.
0: SAI_CK2 clock disabled
1: SAI_CK2 clock enabled
Note: It is not recommended to configure this bit when PDMEN = 1.
SAI_CK2 might not be available for all SAI instances. Refer to Section 60.3: 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 60.3: 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 60.3: 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 enables 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

60.6.19 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.

Note: This field can be used only if D4 line is available.Refer to Section 60.3: SAI implementation to check if it is available.

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.

Note: This field can be used only if D4 line is available.Refer to Section 60.3: SAI implementation to check if it is available.

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.

Note: This field can be used only if D3 line is available.Refer to Section 60.3: SAI implementation to check if it is available.

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.

Note: This field can be used only if D3 line is available.Refer to Section 60.3: SAI implementation to check if it is available.

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.

Note: This field can be used only if D2 line is available.Refer to Section 60.3: SAI implementation to check if it is available.

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.

Note: This field can be used only if D2 line is available.Refer to Section 60.3: SAI implementation to check if it is available.

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.

Note: This field can be used only if D1 line is available.Refer to Section 60.3: SAI implementation to check if it is available.

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.

Note: This field can be used only if D1 line is available.Refer to Section 60.3: SAI implementation to check if it is available.

60.6.20 SAI register map

Table 599. 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	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

Table 599. 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
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
0x0044	SAI_PDMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKEN4	CKEN3	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]		Res.	Res.	Res.	PDMEN
	Reset value																					0	0	0	0			0	0				0
0x0048	SAI_PDMDLY	Res.	DLYM4R[2:0]			Res.	DLYM4L[2:0]			Res.	DLYM3R[2:0]			Res.	DLYM3L[2:0]			Res.	DLYM2R[2:0]			Res.	DLYM2L[2:0]			Res.	DLYM1R[2:0]			Res.	DLYM1L[2:0]		
	Reset value		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0

Refer to *Section 2.3 on page 115* for the register boundary addresses.

61 Secure digital input/output MultiMediaCard interface (SDMMC)

61.1 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 5.1*. Card support for three different databus modes: 1-bit (default), 4-bit and 8-bit. (HS200 SDMMC_CK speed limited to maximum allowed I/O speed)(HS400 is not supported).
- Full compatibility with previous versions of MultiMediaCards (backward compatibility).
- Full compliance with *SD memory card specifications version 6.0*. (SDR104 SDMMC_CK speed limited to maximum allowed I/O speed, 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. (SDR104 SDMMC_CK speed limited to maximum allowed I/O speed, SPI mode and UHS-II mode not supported).
- Data transfer up to 208 Mbyte/s for the 8-bit mode. (depending maximum allowed I/O speed).
- Data and command output enable signals to control external bidirectional drivers.
- IDMA linked list support

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.

61.2 SDMMC implementation

Table 600. SDMMC features

SDMMC modes/features(1)	SDMMC1	SDMMC2
Variable delay (SDR104, HS200)	X	X
SDMMC_CKIN	X	-
SDMMC_CDIR, SDMMC_D0DIR	X	-
SDMMC_D123DIR	X	-

1. X = supported.

61.3 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 729. SDMMC "no response" and "no data" operations

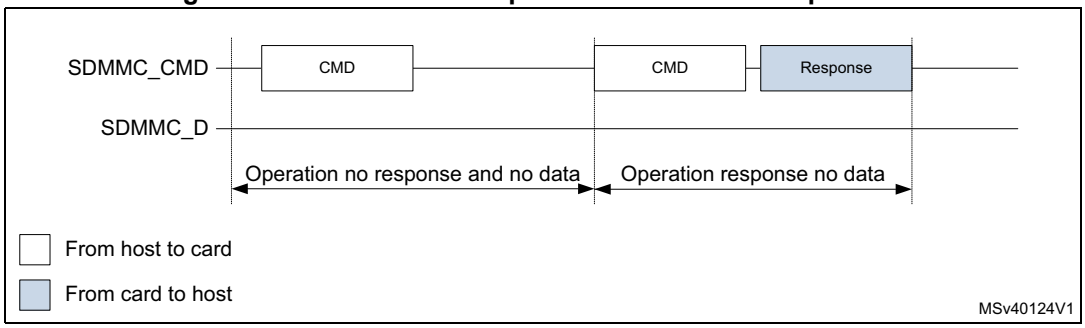
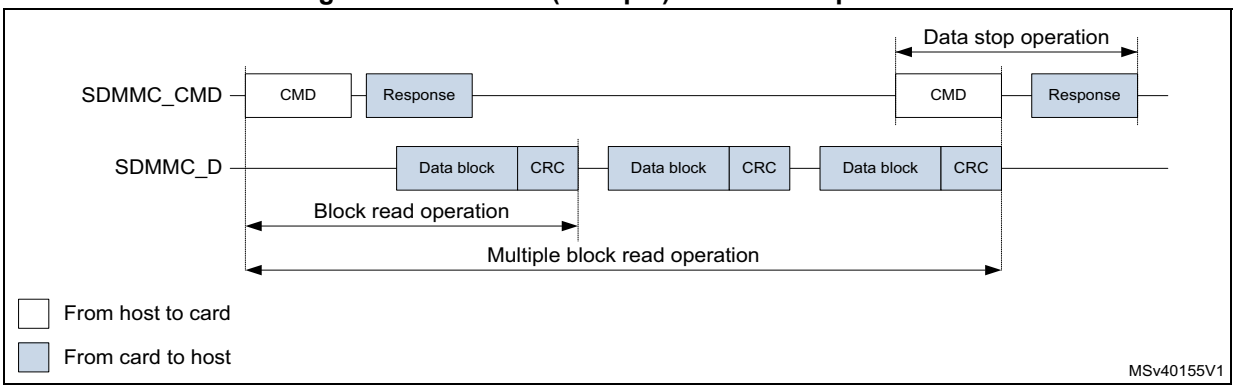
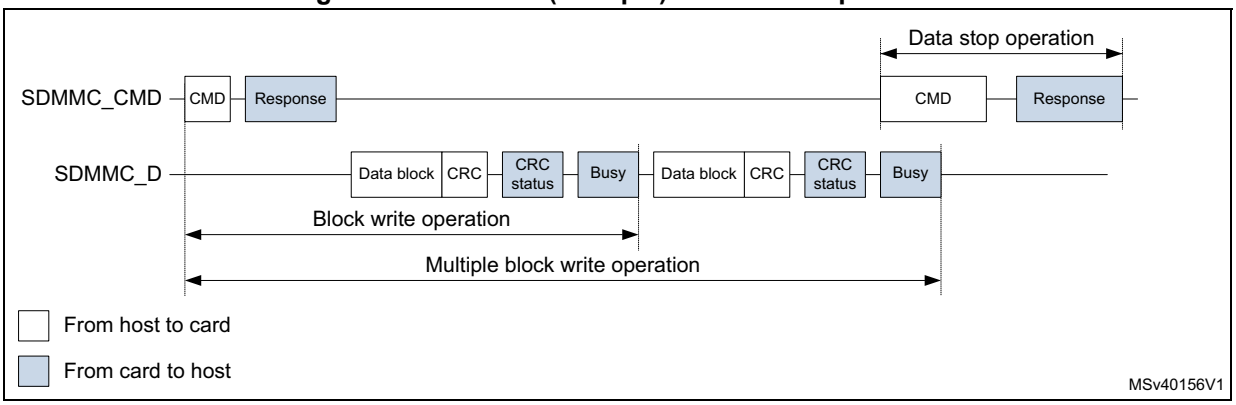


Figure 730. SDMMC (multiple) block read operation



Note: The Stop Transmission command is not required at the end of a e•MMC multiple block read with predefined block count.

Figure 731. 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 732. SDMMC (sequential) stream read operation

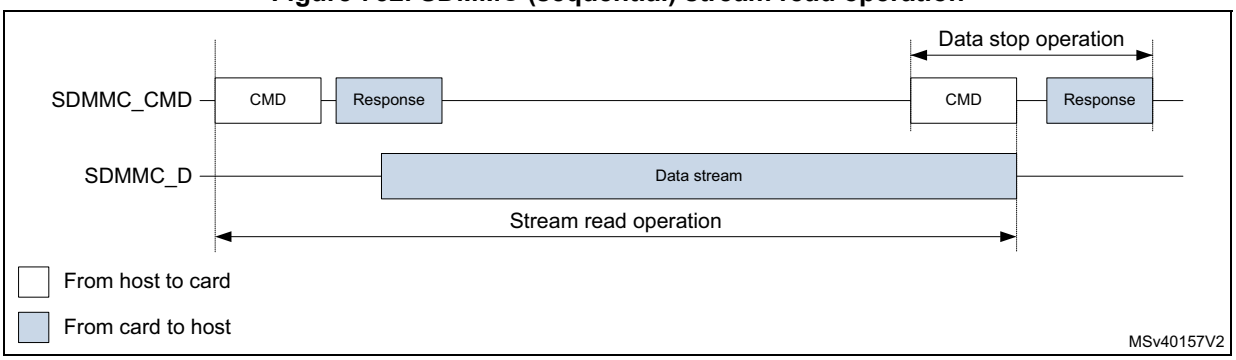
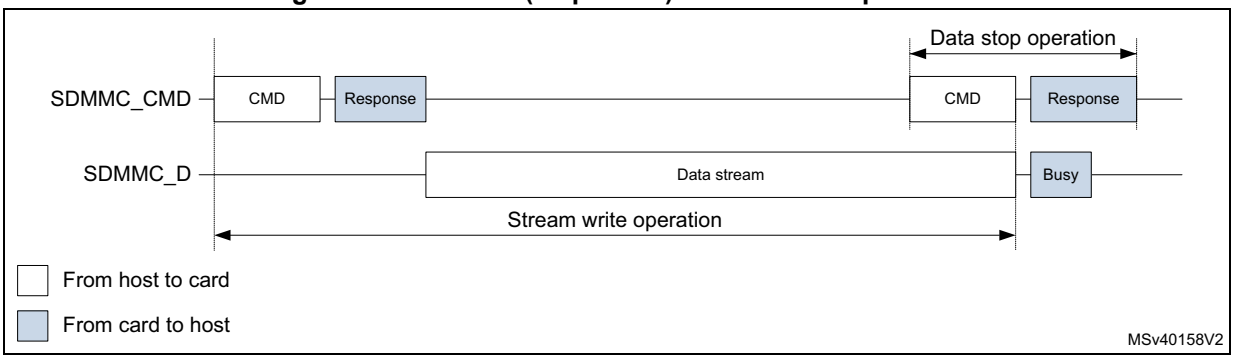


Figure 733. SDMMC (sequential) stream write operation



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).

61.4 SDMMC operation modes

Table 601. 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
SDR104	104	208	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.

SDR104 mode requires variable delay support using sampling point tuning. The use of variable delay is optional for SDR50 mode.

Table 602. 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
High speed HS200	200	200	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.

HS200 mode requires variable delay support using sampling point tuning.

61.5 SDMMC functional description

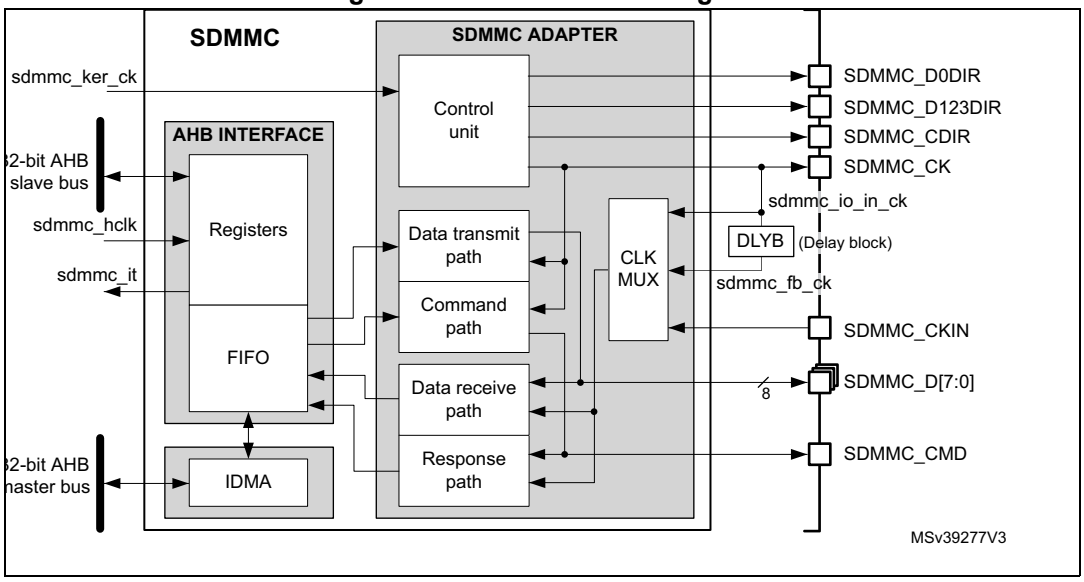
The SDMMC consists of four parts:

- 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.
- A delay block (DLYB) taking care of the receive data sample clock alignment. The delay block is NOT part of the SDMMC. A delay block is mandatory when supporting SDR104 or HS200.

61.5.1 SDMMC block diagram

Figure 734 shows the SDMMC block diagram.

Figure 734. SDMMC block diagram



61.5.2 SDMMC pins and internal signals

Table 603 lists the SDMMC internal input/output signals, *Table 604* the SDMMC pins (alternate functions).

Table 603. 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

Table 603. SDMMC internal input/output signals (continued)

Signal name	Signal type	Description
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).
sdmmc_fb_ck	Digital input	SD/SDIO/*e*•MMC card tuned feedback clock after DLYB delay block (for SDR50, DDR50, SDR104, HS200)

Table 604. 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].

61.5.3 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 735*.

Figure 735. SDMMC Command and data phase relation

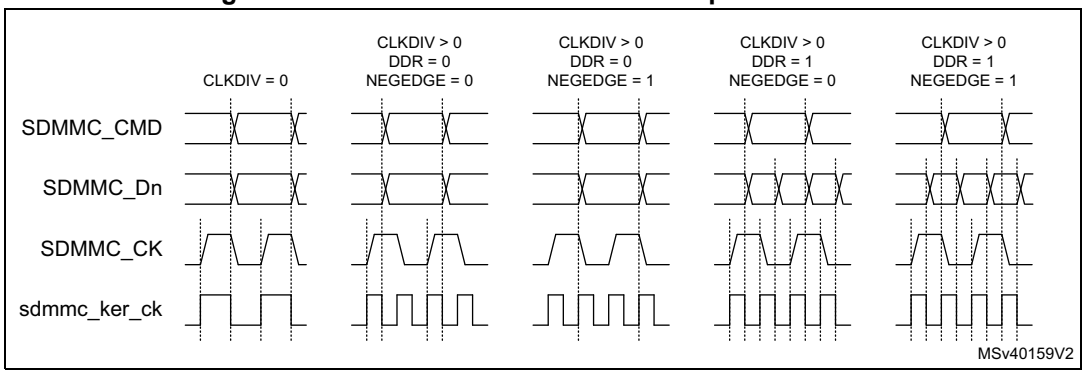


Table 605. SDMMC Command and data phase selection

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.

For tuning the phase of the sampling clock to accommodate the receive data timing, the DLYB delay block available on the device can be connected between **sdmmc_io_in_ck** signal (DLYB input dlyb_in_ck) and **sdmmc_fb_ck** clock input of SDMMC (DLYB output dlyb_out_ck). Selecting the **sdmmc_fb_ck** clock input in the receive path then enables using the phase-tuned sampling clock for the incoming data. This is required for SDMMC to support the SDR104 and HS200 operating mode and optional for SDR50 and DDR50 modes.

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.

61.5.4 SDMMC adapter

The SDMMC adapter (see *Figure 734: 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
- Delay block (DLYB), external to the SDMMC
- 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, or from the sdmmc_fb_ck generated by DLYB.

The DLYB delay block on the device can be used in conjunction with the SDMMC adapter, to tune the phase of the sampling clock for incoming data in SDMMC receive mode. It is required for the SDMMC to support the SDR104 and HS200 operating mode and optional for SDR50 and DDR50 modes.

Adapter register block

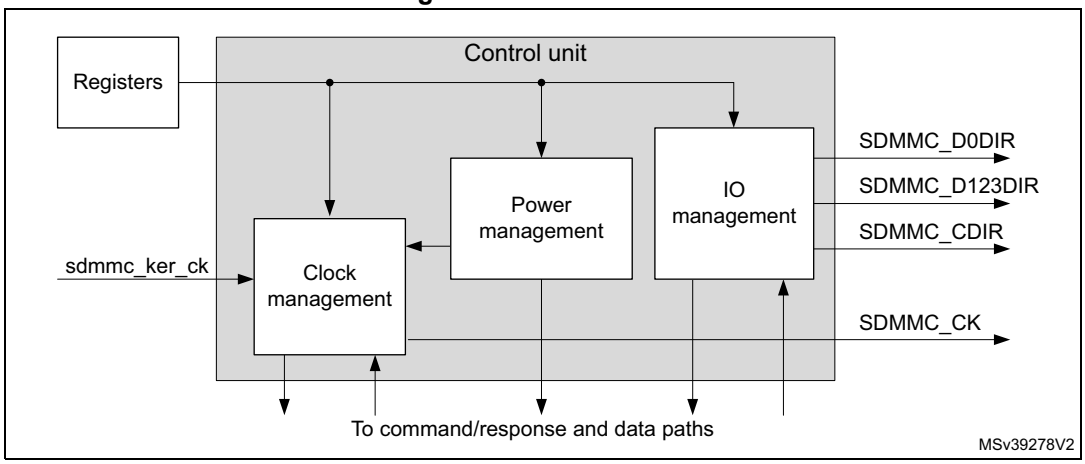
The adapter register block contains all system control registers, the SDMMC command and response registers and the data FIFO.

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 736*, contains the power management functions, the SDMMC_CK clock management with divider, and the I/O direction management.

Figure 736. 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:

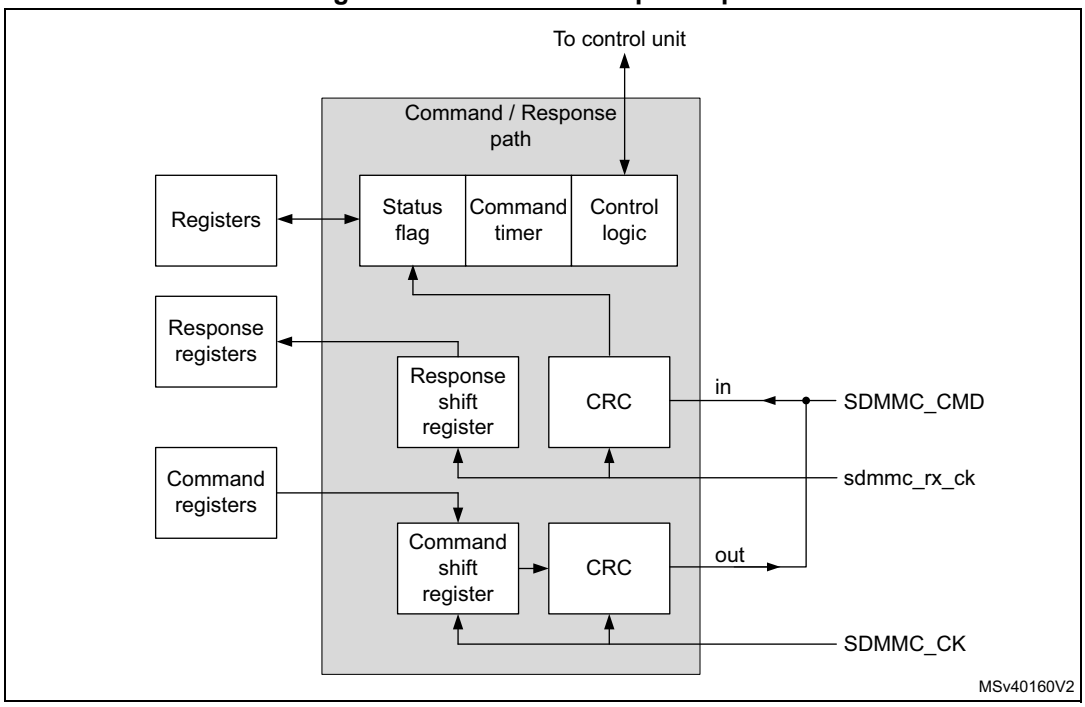
- after reset
- during the power-off or power-up phases
- if the power saving mode (register bit PWRSAV) is enabled and the card bus is in the Idle state for eight clock periods. The clock is stopped eight cycles after both the command/response CPSM and data path DPSM subunits have enter the Idle phase. 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 737. Command/response path

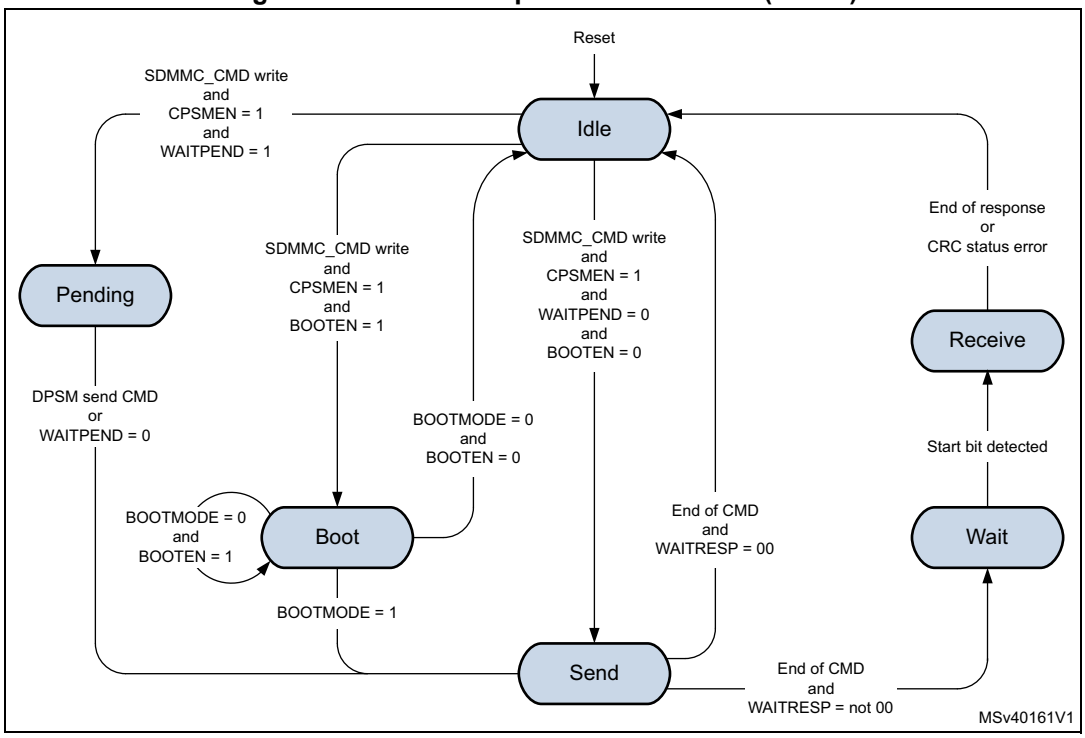


Command/response path state machine (CPSM)

- 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 738. 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 606.*

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 606. 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 607*, *Table 608* and *Table 609*.

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 607. 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 608. 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 609. 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 735*.

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) = (first bit) * x^n + (second bit) * x^{n-1}+... + (last bit before CRC) * x^0

Where n = 39 or 119.

The CPSM can send a number of specific commands to handle various operating modes when CPSMEN is set, see *Table 610*.

Table 610. 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 611*:

Table 611. 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 612*:

Table 612. Command path error handling

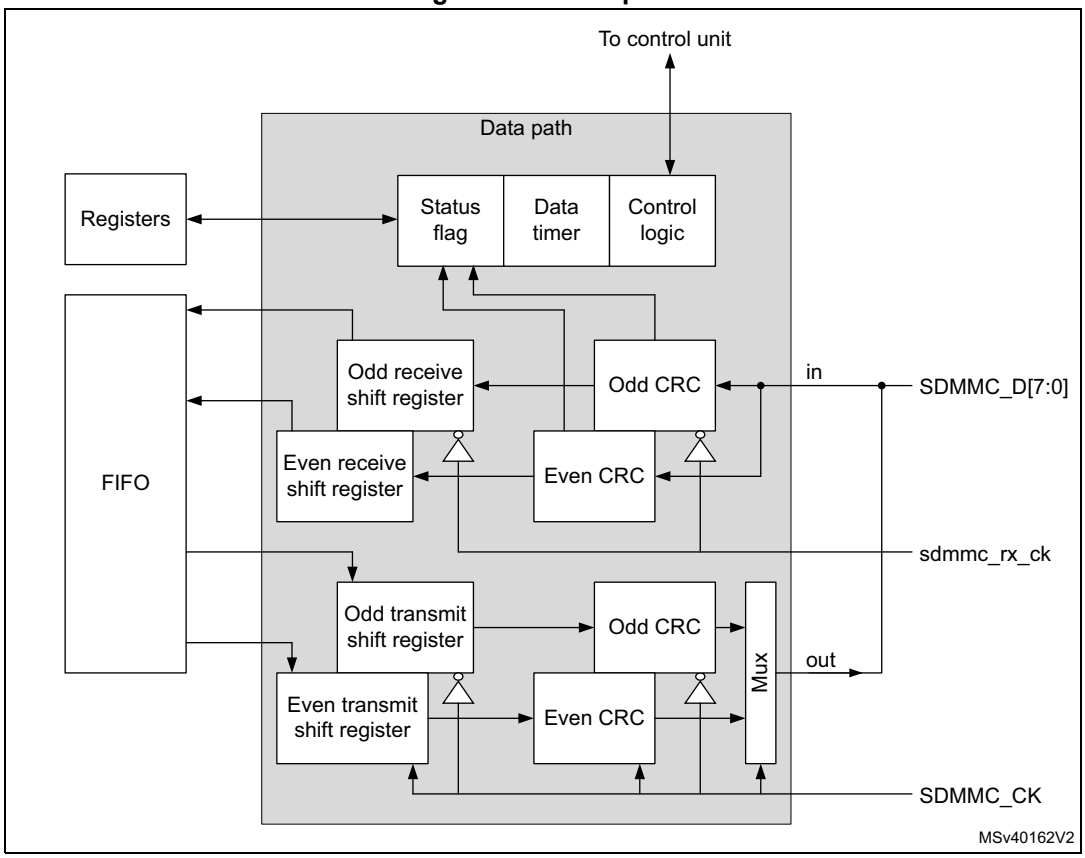
Error	CPSM state	Cause	Card action	Host action	CPSM action
Timeout	Wait	No start bit in time	Unknown	Reset or cycle power card(1)	Move to Idle
CRC status	Receive	Negative status	Command ignored	Resend command(1)	Move to Idle
		Transmission error	Command accepted	Resend command(1)	

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 739* shows the data path block diagram.

Figure 739. 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 740* and *Figure 741*:

- 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 740. DDR mode data packet clocking

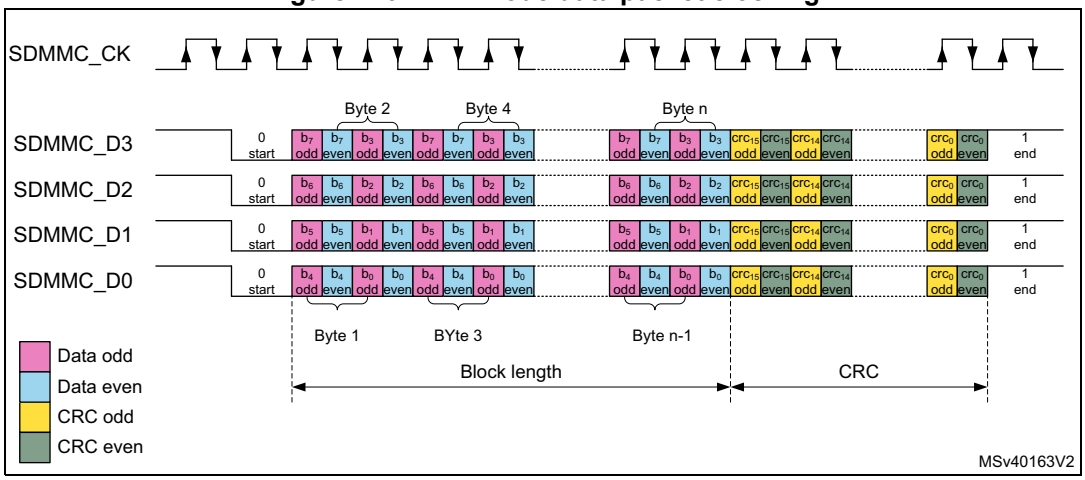
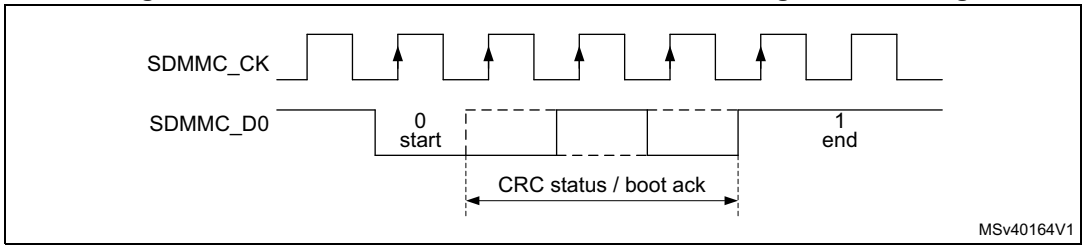


Figure 741. DDR mode CRC status / boot acknowledgment clocking



Data path state machine (DPSM)

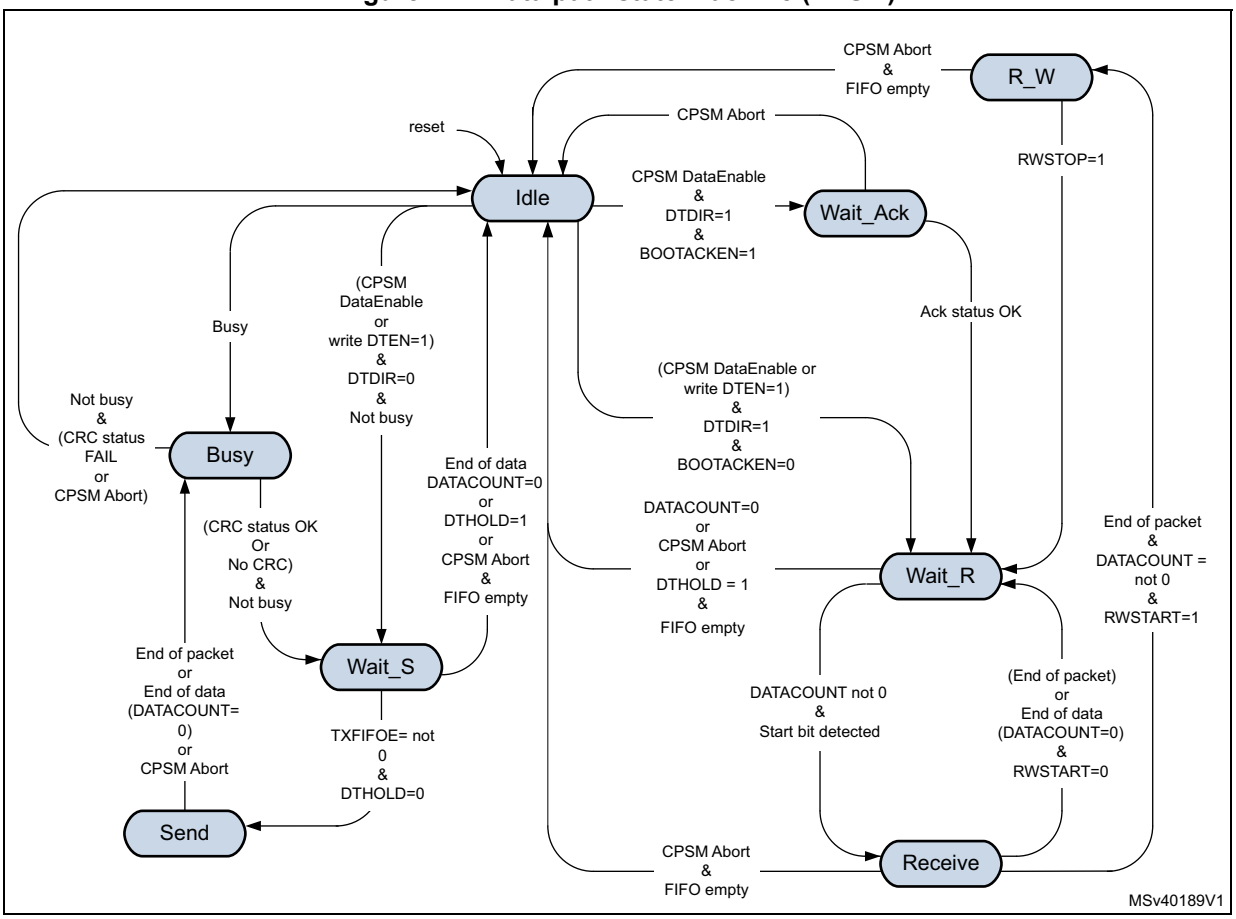
Depending on the transfer direction (send or receive), the data path state machine (DPSM) moves to the Wait_S or Wait_R state when it is enabled:

- 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 742*. When ever the DPSM is active, i.e. not in the Idle state, the DPSMACT bit is set.

Figure 742. 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 613. 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 614*:

Table 614. 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 614. 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 615*:

Table 615. 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 616*.

Table 616. 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 617*.

Table 617. 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 618*.

Table 618. 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:

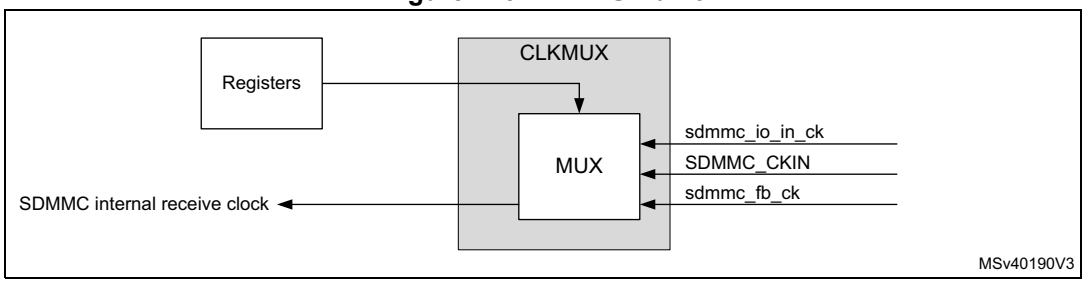
- sdmmc_io_in_ck bus master main feedback clock.
- SDMMC_CKIN external bus feedback clock.
- sdmmc_fb_ck bus tuned feedback clock.

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.

The sdmmc_fb_ck clock input must be selected when the DLYB block on the device is used with SDR104, HS200 and optionally with SDR50 and DDR50 modes.

Figure 743. CLKMUX unit



The sdmmc_rx_ck source must be changed when the CPSM and DPSM are in the Idle state.

61.5.5 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.

61.5.6 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
- linked list 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 IDMABASE. When the IDMA has finished transferring all data the and the DPSM has completed the transfer the DATAEND flag is set.

Linked list channel

In linked list configuration, IDMAMODE = 1, the data at the memory side is subsequently accessed from linked buffers, located at base address IDMABASE. The size of the memory buffers is defined by IDMABSIZE. The buffer size must be an integer multiple of the burst size. The bit ULA is used to indicate if a new linked list buffer configuration has to be loaded from the linked list table. A new linked list configuration is loaded when the ULA bit for the current linked list item is set.

The first linked list item configuration is programmed by firmware directly in the SDMMC registers.

When the IDMA has finished transferring all the data of one linked list buffer, according IDMABSIZE, and when the linked list item ULA bit is set, the IDMA loads the new linked list item from the linked list table, and continues transferring data from the next linked list buffer. When the IDMA has finished transferring all data, according IDMABSIZE and ULA, and the DPSM has completed the transfer, according DATALENGHT, the DATAEND flag is set.

In the following cases, the linked list provides more buffer space than the data to transfer which means the current linked list buffer data has not completely be transfered:

- the ULA bit is set, and all SDMMC data according DATALENGTH has been transfered (DATAEND flag)
- a transfer error (DCRCFAIL when DATACOUNT > 0, RXOVERR, TXUNDERR) occurs
- a transfer is hold (DTHOLD)

In all above cases, the IDMA linked list is stopped and the FIFO is flushed/reset. Before starting or restarting a new SDMMC transfer, the software must initialize a new linked list with correct IDMABASE and IDMABSIZE.

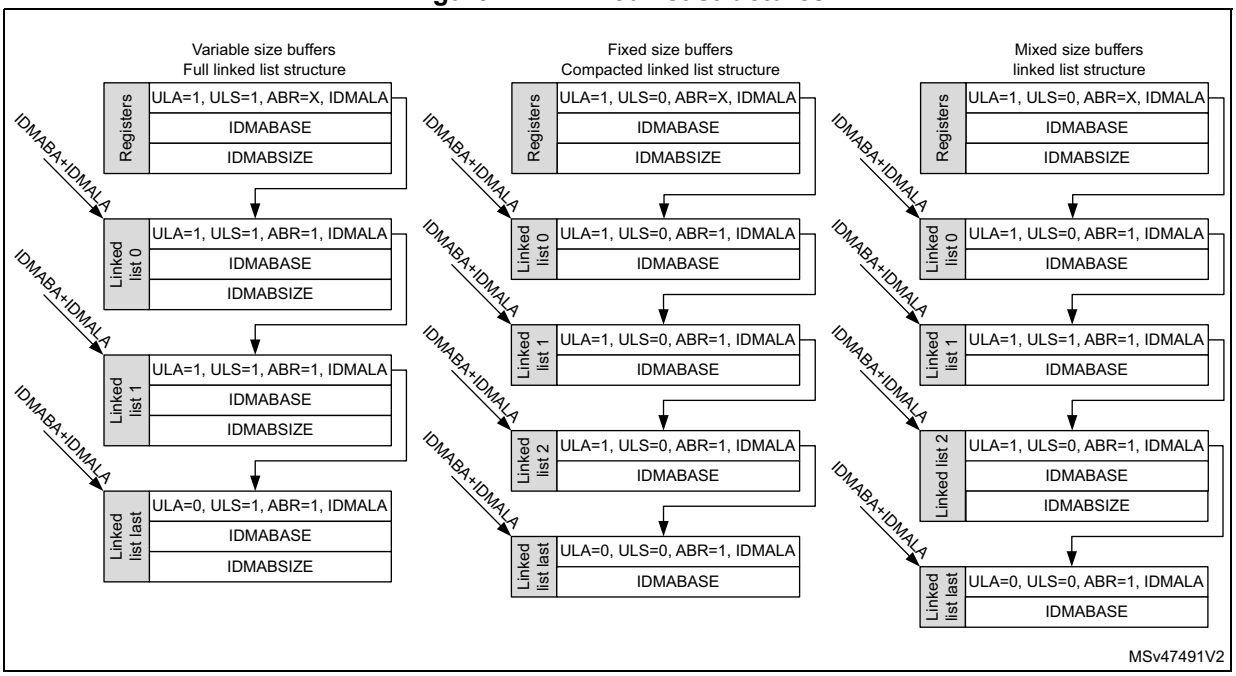
When a IDMA transfer error occurs (see *Section : IDMA transfer error management*) or when the linked list does not provide sufficient buffer space:

- the linked list ends with ULA = 0 and all last linked list buffer data has been transfered, and not all SDMMC data according DATALENGTH has been transfered. The SDMMC transfer is stopped and an IDMA transfer error is generated (see *Section : IDMA transfer error management*).

For a given linked list item, the base address is given by the linked list base IDMABA register value plus the linked list offset IDMALA register value.

The content of each linked list item can be specified by the ULS bit, which makes possible to optionally load the IDMABSIZE, resulting in a 3-word linked list structure. When the IDMABSIZE is not to be loaded (i.e. fixed size buffers) a compacted reduced 2-word linked list structure can be used containing only the IDMABASER and the IDMALAR values.

Figure 744. Linked list structures



There is no restriction on mixing both linked list item structures in a single list, this enables the IDMABSIZE to be updated only when needed.

Whenever a linked list buffer has been transfered and the current buffer ULA = 1, an end-of-linked-list-buffer-transfer-complete interrupt (IDMABTC) may be generated (if interrupt is enabled).

Linked list acknowledgment

In the case where software dynamically updates the linked list, during the SDMMC transfer, the availability of a new linked list buffer can be acknowledged by the acknowledge buffer ready (ABR) bit.

When ABR acknowledges that the new linked list buffer is ready, the IDMA continues transferring data from the new linked list buffer.

When ABR indicates that the new linked list buffer is not ready, an IDMA transfer error is generated (see *Section : IDMA transfer error management*). Depending when the IDMA transfer error occurs, it normally causes the generation of an TXUNDERR or RXOVERR error. When a linked list buffer is not acknowledged in time the SDMMC transfer is stopped.

The ABR information is “don't care” when starting the linked list from software programmed register information. The first linked list buffer must be ready to be used before starting the SDMMC transfer.

IDMA transfer error management

An IDMA transfer error can occur:

- When reading or writing a reserved address space (for data or linked list information).
- When there is no more linked list buffer space to store received SDMMC data.
- When all linked list buffer data has been transfered and still more SDMMC data needs to be sent.
- When the availability of a linked list buffer is not acknowledged.

On a IDMA transfer error subsequent IDMA transfers are disabled and an IDMATE flag is set and hardware flow control is disabled. 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).

61.5.7 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 619. 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
e•MMC HS200	8	200	150
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
SD SDR104	4	208	78

61.6 Card functional description

61.6.1 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 620. 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]).

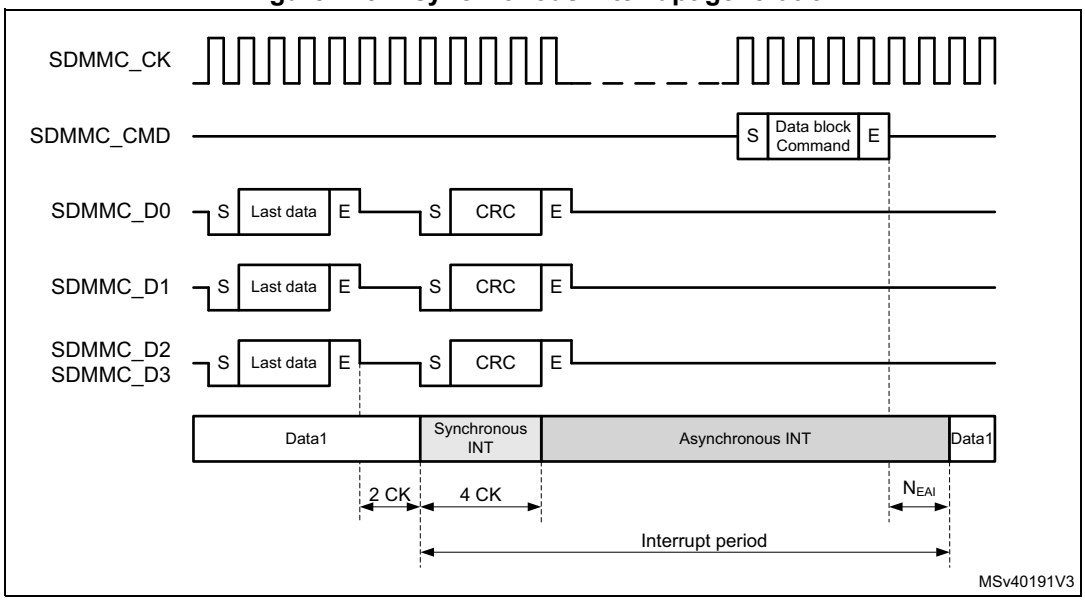
In SD 1-bit mode pin 8 is dedicated to the interrupt function (IRQ), and there are no timing constraints on interrupts.

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 620*.

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 745. 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 746. Synchronous interrupt period data read

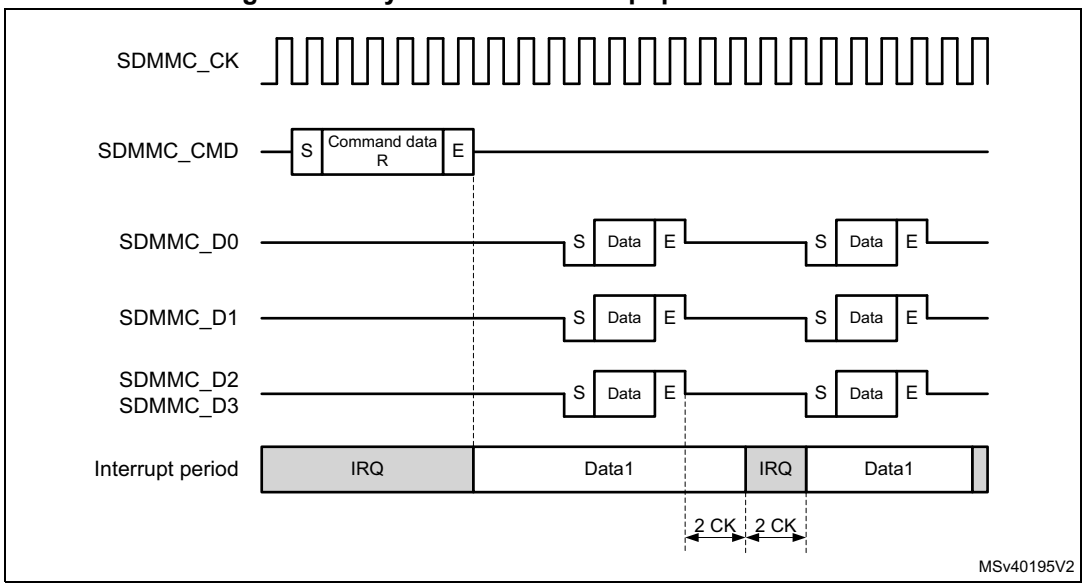
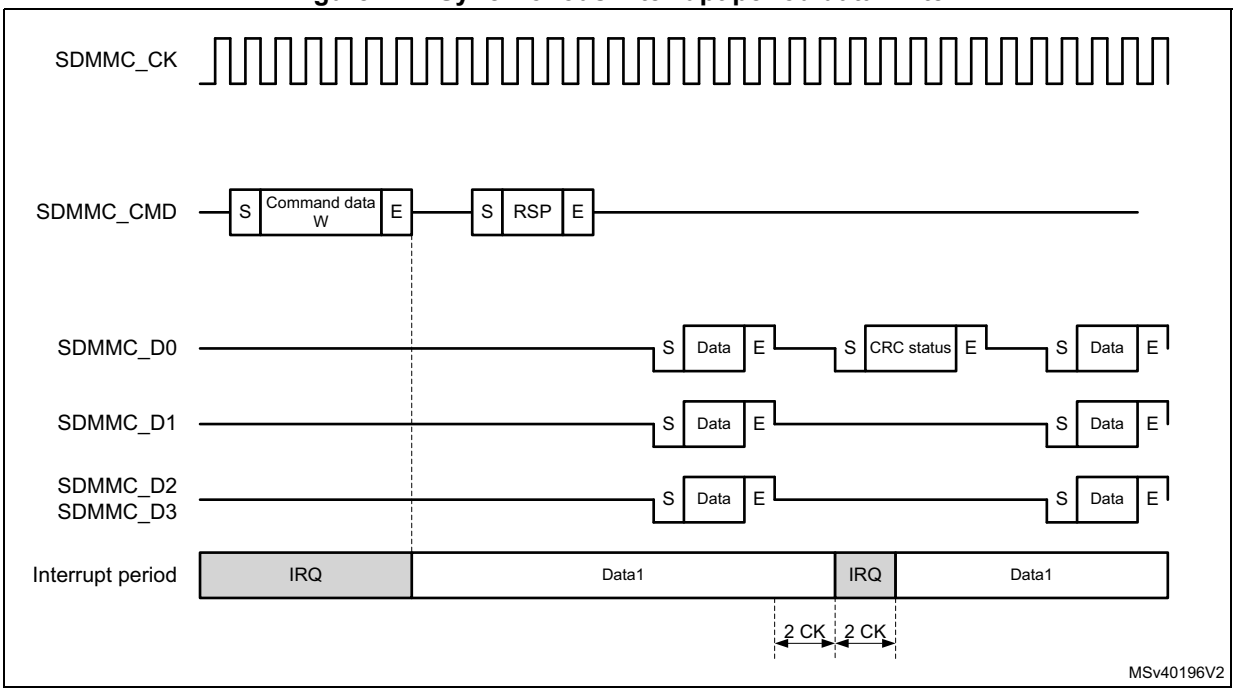


Figure 747. Synchronous interrupt period data write



In SDR50, SDR104, 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 748. Asynchronous interrupt period data read

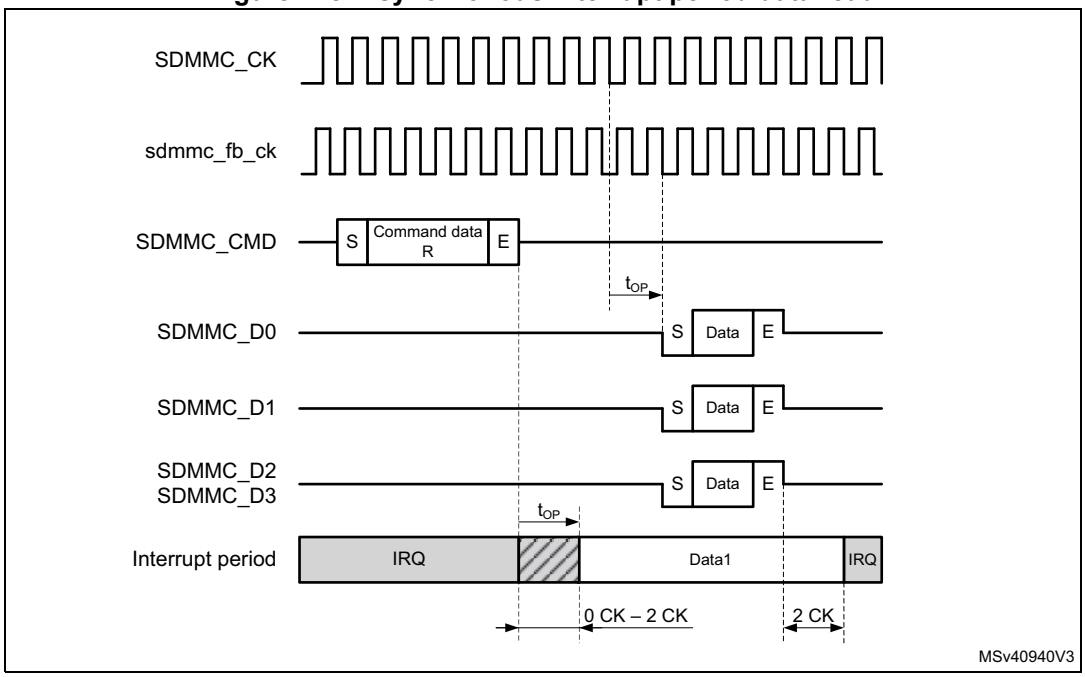
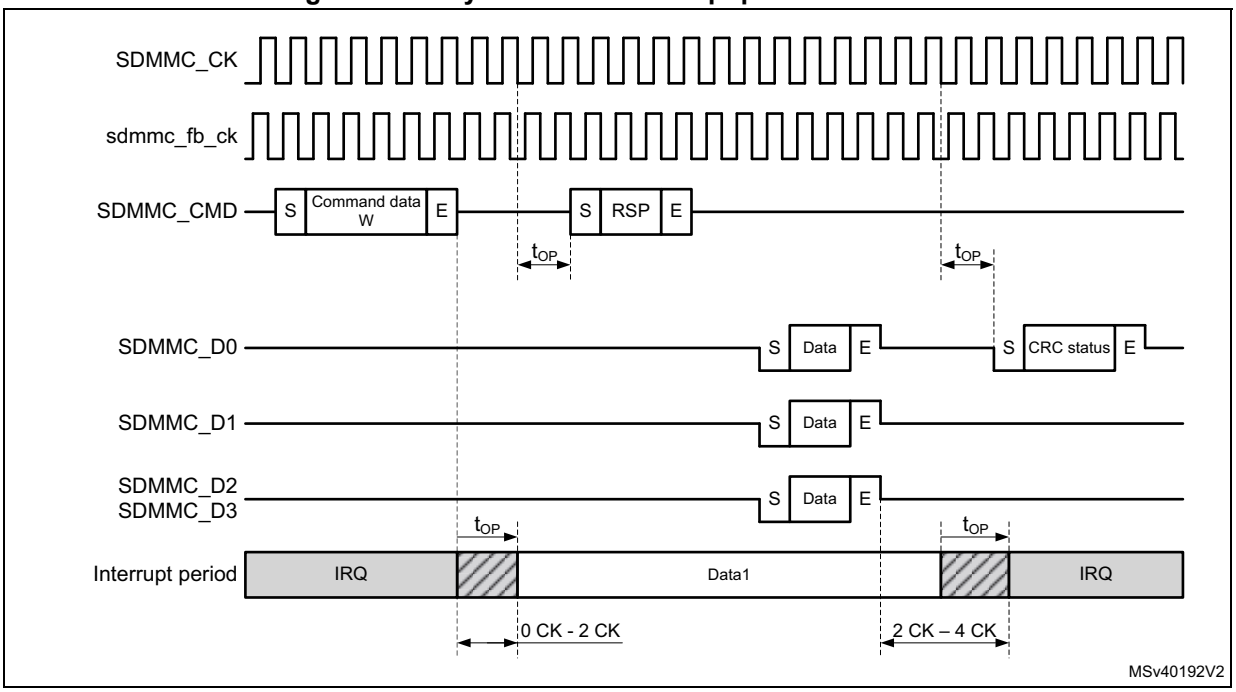


Figure 749. Asynchronous interrupt period data write



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 621*.

Table 621. 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 makes possible 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 makes possible 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:

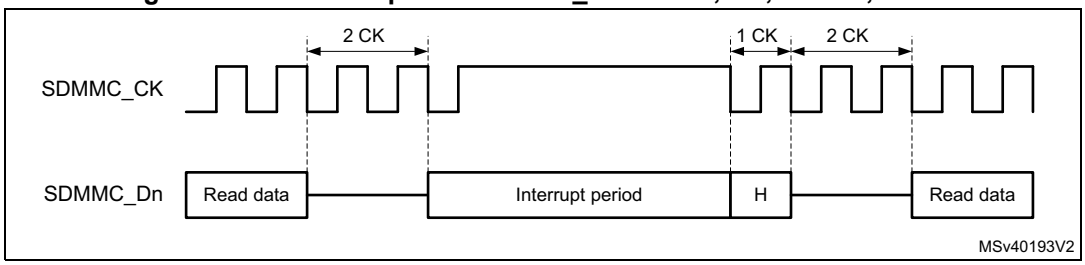
1. Stopping the SDMMC_CK.
2. Using Read Wait signaling on SDMMC_D2.

The SDMMC can perform a Read Wait with register settings according *Table 620*.

Depending the SDMMC operation mode (DS, HS, SDR12, SDR25) or (SDR50, SDR104, DDR) each method has a different characteristic.

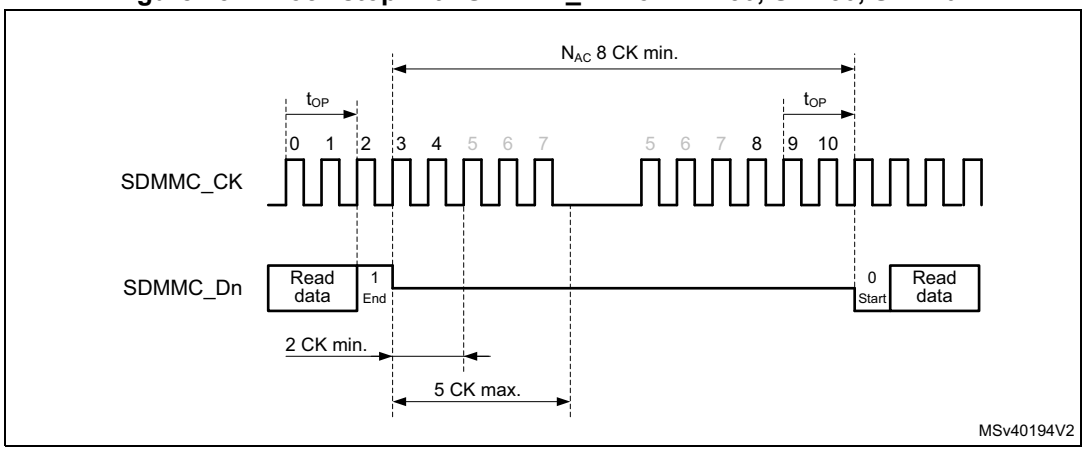
The timing for pause read operation by stopping the SDMMC_CK for DS, HS, SDR12, and SDR25, the SDMMC_CK may be stopped 2 SDMMC_CK cycles after the end bit. When ready the host resumes by restarting clock, see *Figure 750*.

Figure 750. Clock stop with SDMMC_CK for DS, HS, SDR12, SDR25



The timing for pause read operation by stopping the SDMMC_CK for SDR50, SDR104, 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 751*. (In DDR50 mode the SDMMC_CK must only be stopped after the falling edge, when the clock line is low.)

Figure 751. Clock stop with SDMMC_CK for DDR50, SDR50, SDR104



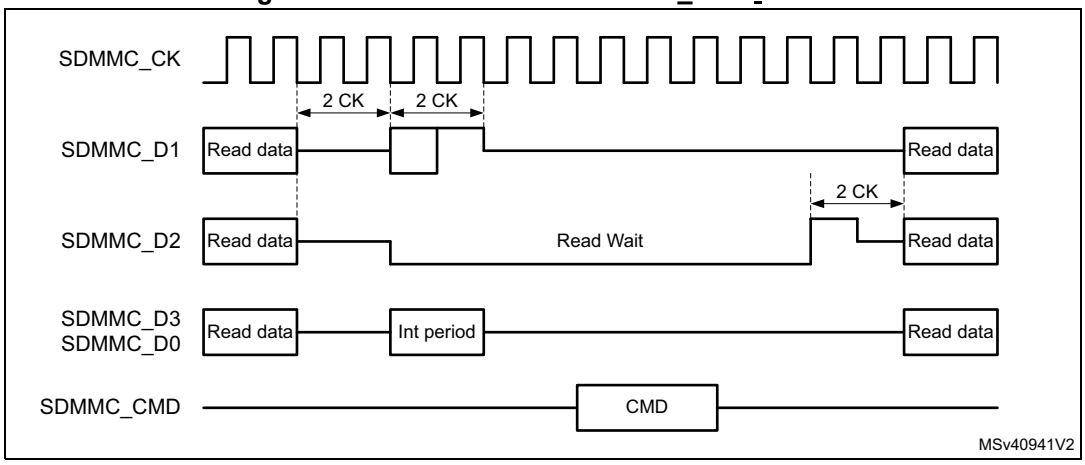
In Read Wait SDMMC_CK clock stopping, when RWSTART is set, the DSPM stops the clock after the end bit of the current received data block CRC. The clock start again after writing 1 to the RWSTOP bit, where after the DPSM waits for a start bit from the card.

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 enables 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.

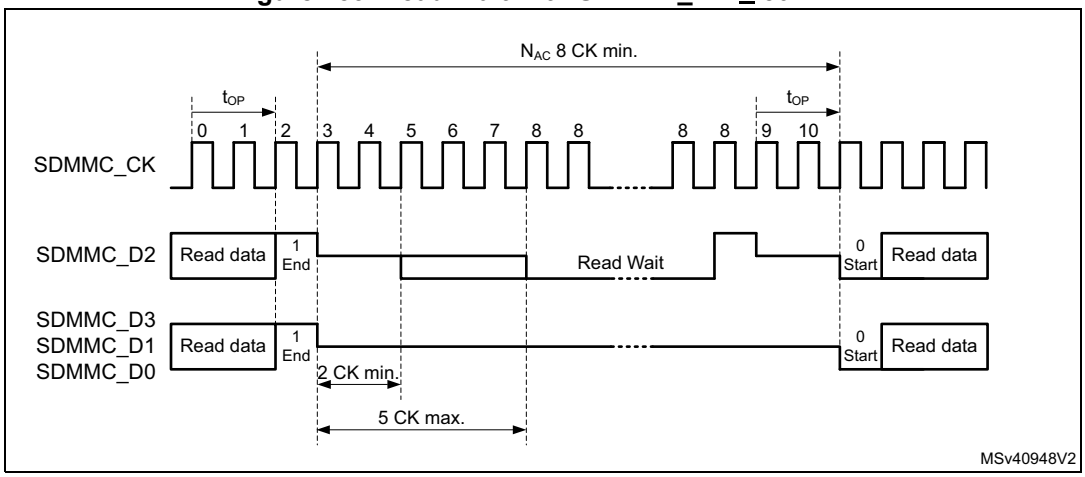
The timing for Read Wait with a SDMMC_CK less then 50MHz (DS, HS, SDR12, SDR25) is based on the interrupt period generated by the card on SDMMC_D1. The host by asserting SDMMC_D2 low during the interrupt period requests the card to enter Read Wait. To exit Read Wait the host must raise SDMMC_D2 high during one SDMMC_CK cycles before making it Hi-Z, see *Figure 752*.

Figure 752. Read Wait with SDMMC_CK < 50 MHz



For SDR50, SDR104 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 753*).

Figure 753. 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.

61.6.2 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 622. 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.
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:

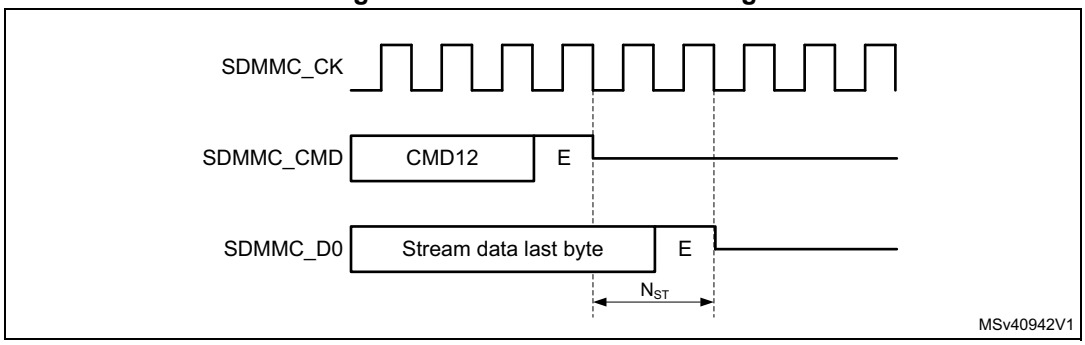
1. Load CMD12 Stop Transmission command in registers and set the CMDSTOP bit.
   a) This causes the CPSM Abort signal to be generated when the command is sent to the DPSM.
2. Configure the CPSM to send a command immediately (clear WAITPEND bit).
   a) The card, when sending data, stops data transfer 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 754. 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.

61.6.3 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 can be switched off

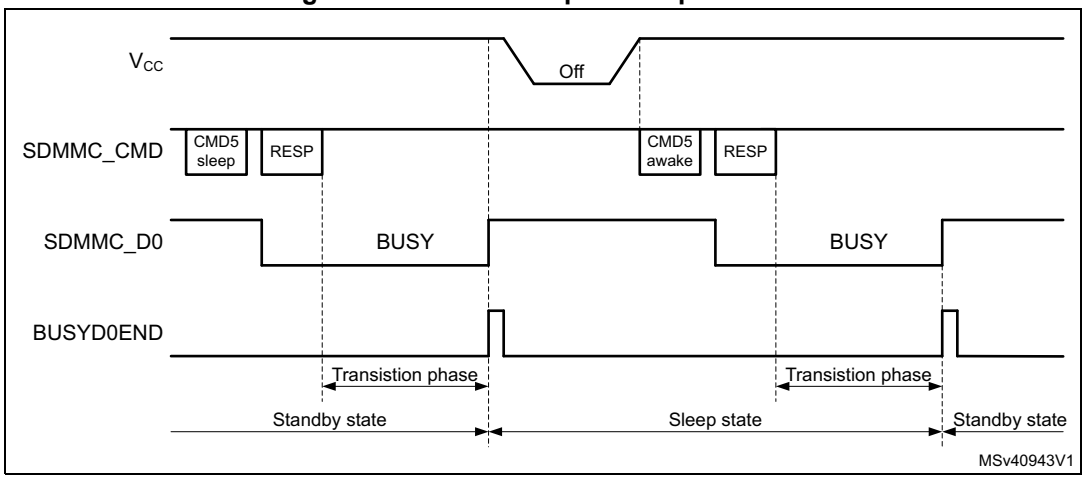
The CMD5 (AWAKE) is used to initiate the state transition from Sleep state to Standby state. The card indicates Busy, pulling down SDMMC_D0, during the transition phase. The Standby state is reached when the card stops pulling down the SDMMC_DO line.

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 can be switched off only after the Sleep state has been reached. The Vcc supply must be reinstalled before CMD5 (AWAKE) is sent.

Figure 755. CMD5 Sleep Awake procedure



61.6.4 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.

61.6.5 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:

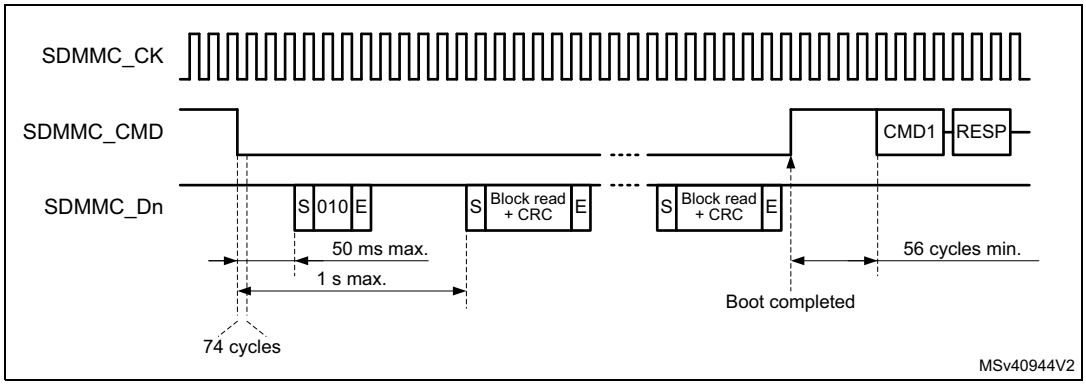
- The partition from which boot data is read (EXT_CSD Byte[179])
- The boot data size (EXT_CSD Byte[226])
- The bus configuration during boot (EXT_CSD Byte[177])
- Receiving boot acknowledgment from the card. (EXT_CSD Byte[179])

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 756. Normal boot mode operation



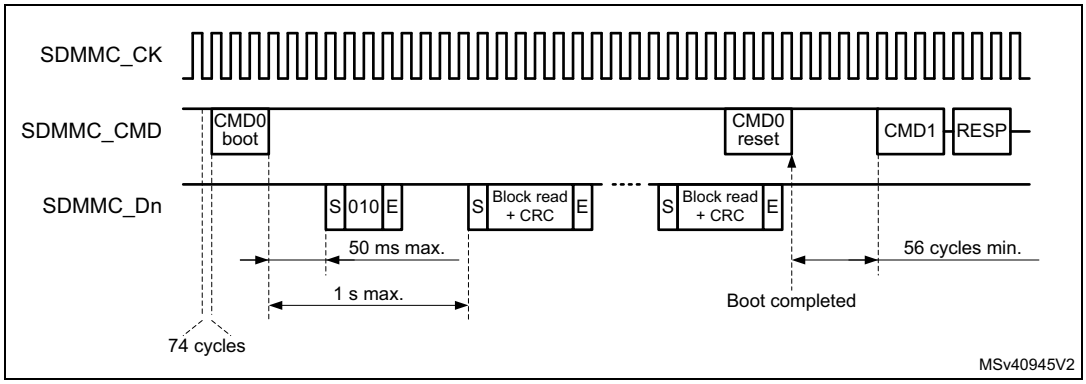
To perform the normal boot procedure the following steps needed:

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 757. 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:

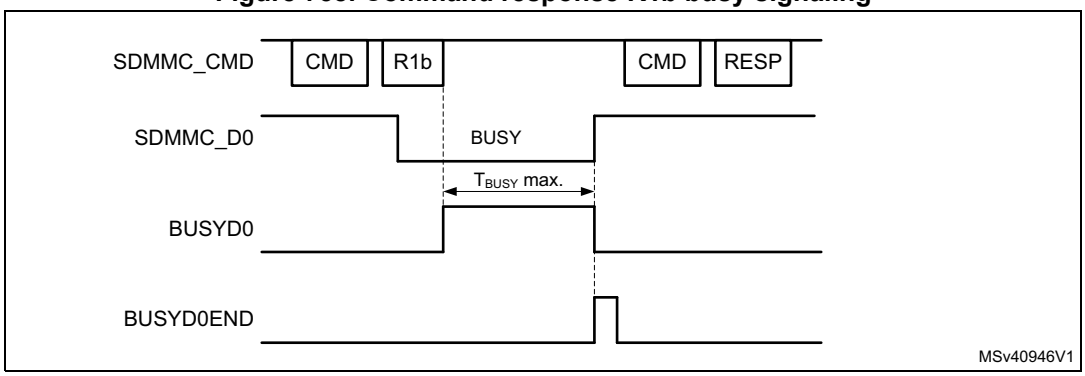
- the loaded command and argument to be sent out. (BOOTMODE = alternative boot).
- the ACK timeout to start.
- DPSM to be enabled.

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.

61.6.6 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 758. 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.

61.6.7 Reset and card cycle power

Reset

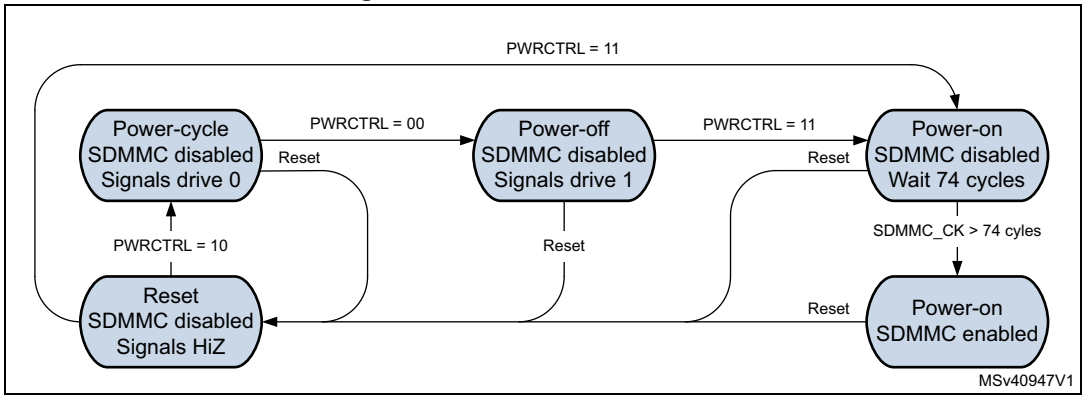
Following reset the SDMMC is in the reset state. In this state the SDMMC is disabled and no command nor data can be transfered. The SDMMC_D[7:0], and SDMMC_CMD are in HiZ and the SDMMC_CK is driven low.

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 759.*.

Figure 759. SDMMC state control

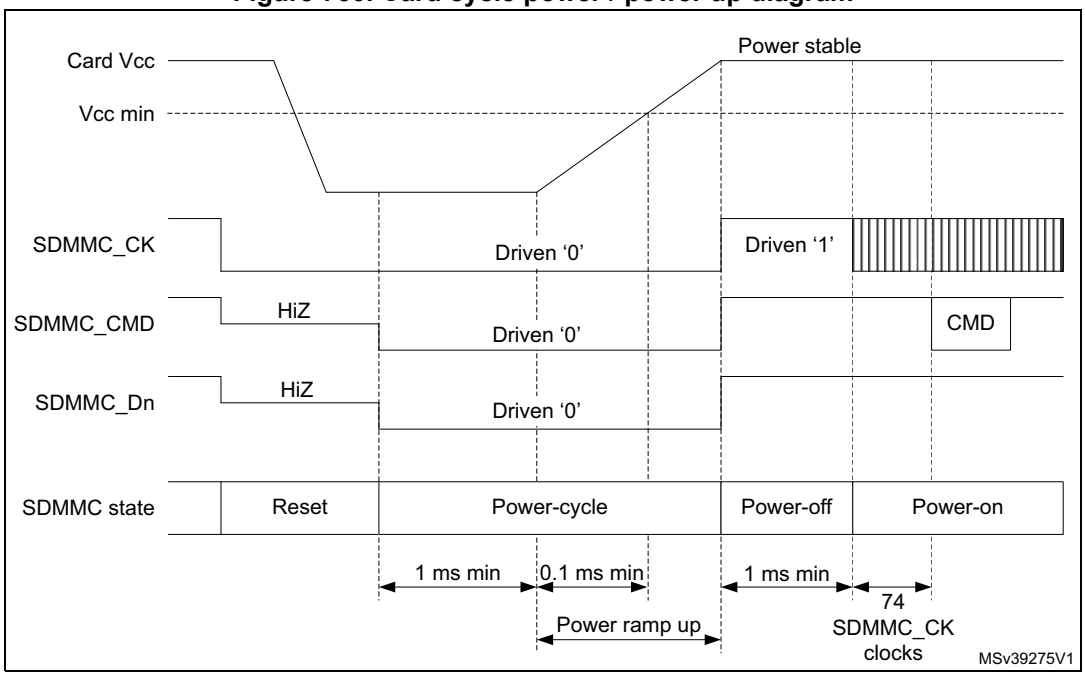


Card cycle power

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 760. Card cycle power / power up diagram



61.7 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.

On an IDMA linked list transfer error, the hardware flow control is disabled. As a consequence, depending on when the IDMA linked list transfer error occurs, an underrun or overrun error may also occur (see *Section : IDMA transfer error management*).

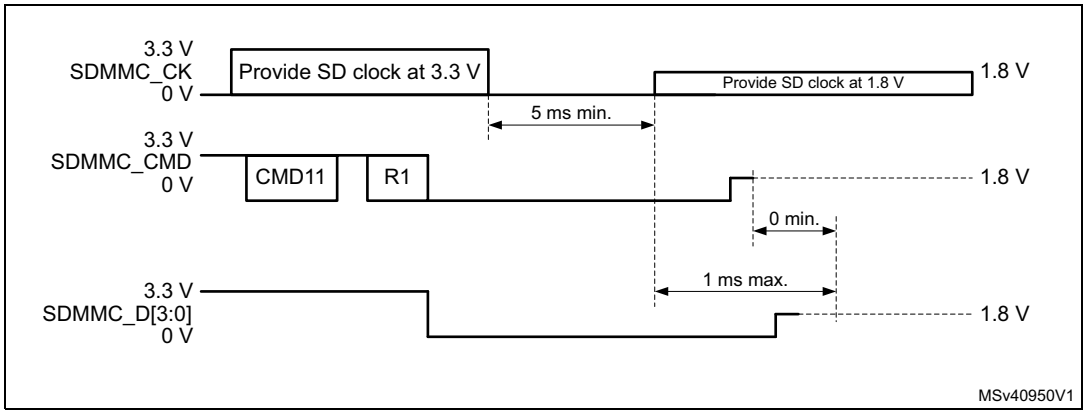
To enable hardware flow control during data transfer, the HWFC_EN register bit must be set to 1. After reset hardware flow control is disabled.

Hardware flow control must only be used when the SDMMC_Dn data is cycle-aligned with the SDMMC_CK. Whenever the sdmmc_fb_ck from the DLYB delay block is used, i.e in the case of SDR104 mode with a t_{OP} and Dt_{OP} delay > 1 cycle, hardware flow control can not be used.

61.8 Ultra-high-speed phase I (UHS-I) voltage switch

UHS-I mode (SDR12, SDR25, SDR50, SDR104, 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 761. 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.

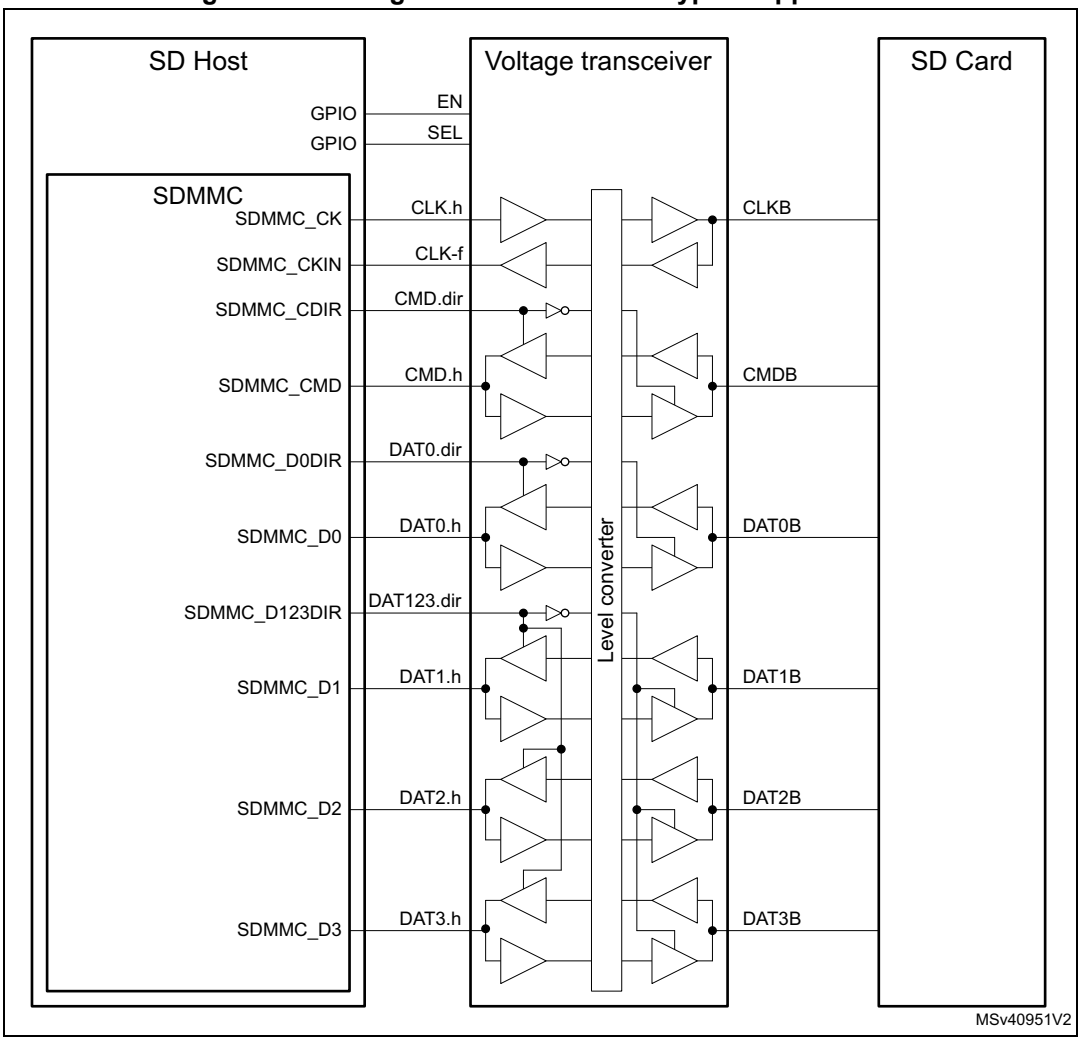
6. The card after detecting SDMMC_CK low begins switching signaling voltage to 1.8 V.
7. The host SDMMC hardware after at least 5 ms restarts the SDMMC_CK.
8. The card within 1 ms from detecting SDMMC_CK transition drives CMD and DAT[3:0] high for at least 1 SDMMC_CK cycle and then stop driving CMD and 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 762. 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.

61.9 SDMMC interrupts

Table 623. 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 623. 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

61.10 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.

61.10.1 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.

61.10.2 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: sdmmc_fb_ck tuned feedback clock selected as receive clock.

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, SDR104, HS200 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, see SDMMC status register definition in *Section 61.10.11*.

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.

61.10.3 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.

61.10.4 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.

61.10.5 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.

61.10.6 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
CARDSTATUS[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
CARDSTATUS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CARDSTATUS[31:0]**: Card status according table below
See *Table 624*.

The card status size is 32 or 128 bits, depending on the response type.

Table 624. 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.

61.10.7 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.

61.10.8 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.

61.10.9 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: Block length = 2^0 = 1 byte

0001: Block length = 2^1 = 2 bytes

0010: Block length = 2^2 = 4 bytes

0011: Block length = 2^3 = 8 bytes

0100: Block length = 2^4 = 16 bytes

0101: Block length = 2^5 = 32 bytes

0110: Block length = 2^6 = 64 bytes

0111: Block length = 2^7 = 128 bytes

1000: Block length = 2^8 = 256 bytes

1001: Block length = 2^9 = 512 bytes

1010: Block length = 2^{10} = 1024 bytes

1011: Block length = 2^{11} = 2048 bytes

1100: Block length = 2^{12} = 4096 bytes

1101: Block length = 2^{13} = 8192 bytes

1110: Block length = 2^{14} = 16384 bytes

1111: 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.

61.10.10 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.

61.10.11 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 (masked by hardware when IDMA is enabled)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 4 **TXUNDERR**: Transmit FIFO underrun error (masked by hardware when IDMA is enabled)
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.

61.10.12 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.	IDMABTCC	IDMATEC	CKSTOPC	VSWENDC	ACKTIMEOUTC	ACKFAILC	SDIOITC	BUSYD0ENDC	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.	DABORTC	DBCKENDC	DHOLDC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTIMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC
				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

61.10.13 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

61.10.14 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.

61.10.15 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 enables 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.

61.10.16 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 enables 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.	Res.	IDMABMODE	IDMAEN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

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: Linked list 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

61.10.17 SDMMC IDMA buffer size register (SDMMC_IDMABSIZER)

Address offset: 0x054

Reset value: 0x0000 0000

The SDMMC_IDMABSIZER register contains the buffer size when in linked list 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.	IDMABNDT[11]
															rw

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

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

Bits 16:5 **IDMABNDT[11:0]**: Number of bytes per buffer

This 12-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 = 0x001: buffer size = 8 words = 32 bytes.
Example: IDMABNDT = 0x800: buffer size = 16384 words = 64 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.

61.10.18 SDMMC IDMA buffer base address register (SDMMC_IDMABASER)

Address offset: 0x058

Reset value: 0x0000 0000

The SDMMC_IDMABASER register contains the memory buffer base address in single buffer configuration and linked list configuration.

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

Bits 31:0 **IDMABASE[31:0]**: Buffer 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).

61.10.19 SDMMC IDMA linked list address register (SDMMC_IDMALAR)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ULA	ULS	ABR	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
IDMALA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **ULA**: Update SDMMC_IDMALAR from linked list when in linked list mode (SDMMC_IDMACTRLR.IDMABMODE select linked list mode)
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: SDMMC_IDMALAR is not to be updated, last linked list item.
1: SDMMC_IDMALAR is to be updated from linked list table.

Bit 30 **ULS**: Update SDMMC_IDMABSIZE from the next linked list when in linked list mode (SDMMC_IDMACTRLR.IDMABMODE select linked list mode and ULA = 1)
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: SDMMC_IDMABSIZER is not to be updated from next linked list table.
1: SDMMC_IDMABSIZER is to be updated from next linked list table.

Bit 29 **ABR**: Acknowledge linked list buffer ready
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
This bit is not taken into account when starting the first linked list buffer from the software programmed register information. ABR is only taken into account on subsequent loaded linked list items.
0: Loaded linked list buffer is not ready (this causes a linked list IDMA transfer error to be generated).
1: Loaded linked list buffer ready acknowledge. Linked list buffer data are transfered by IDMA.

Bits 28:16 Reserved, must be kept at reset value.

Bits 15:2 **IDMALA[13:0]**: Word aligned linked list item address offset
Linked list item offset pointer to the base of the next linked list item structure.
Linked list item base address is IDMABA + IDMALA.
These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Bits 1:0 Reserved, must be kept at reset value.

61.10.20 SDMMC IDMA linked list memory base register (SDMMC_IDMABAR)

Address offset: 0x068

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDMABA[29:14]															
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
IDMABA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:2 **IDMABA[29:0]**: Word aligned Linked list memory base address
Linked list memory base pointer.
These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Bits 1:0 Reserved, must be kept at reset value.

61.10.21 SDMMC register map

Table 625. 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	CARDSTATUS[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	CARDSTATUS[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	CARDSTATUS[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	CARDSTATUS[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
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

Table 625. 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
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.	Res.	IDMABMODE	IDMAEN
	Reset value																															0	0
0x054	SDMMC_ IDMABSIZER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABNDT[11:0]												Res.	Res.	Res.	Res.	Res.
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0					
0x058	SDMMC_ IDMABASER	IDMABASE[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 - 0x060	Reserved	Res.																															
0x064	SDMMC_ IDMALAR	ULA	ULS	ABR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMALA[13:0]														Res.	Res.
	Reset value	0	0	0														0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x068	SDMMC_ IDMABAR	IDMABA[29:0]																													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	0		
0x06C - 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 115* for the register boundary addresses.

62 FD controller area network (FDCAN)

62.1 Introduction

The controller area network (CAN) subsystem (see *Figure 763*) 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.

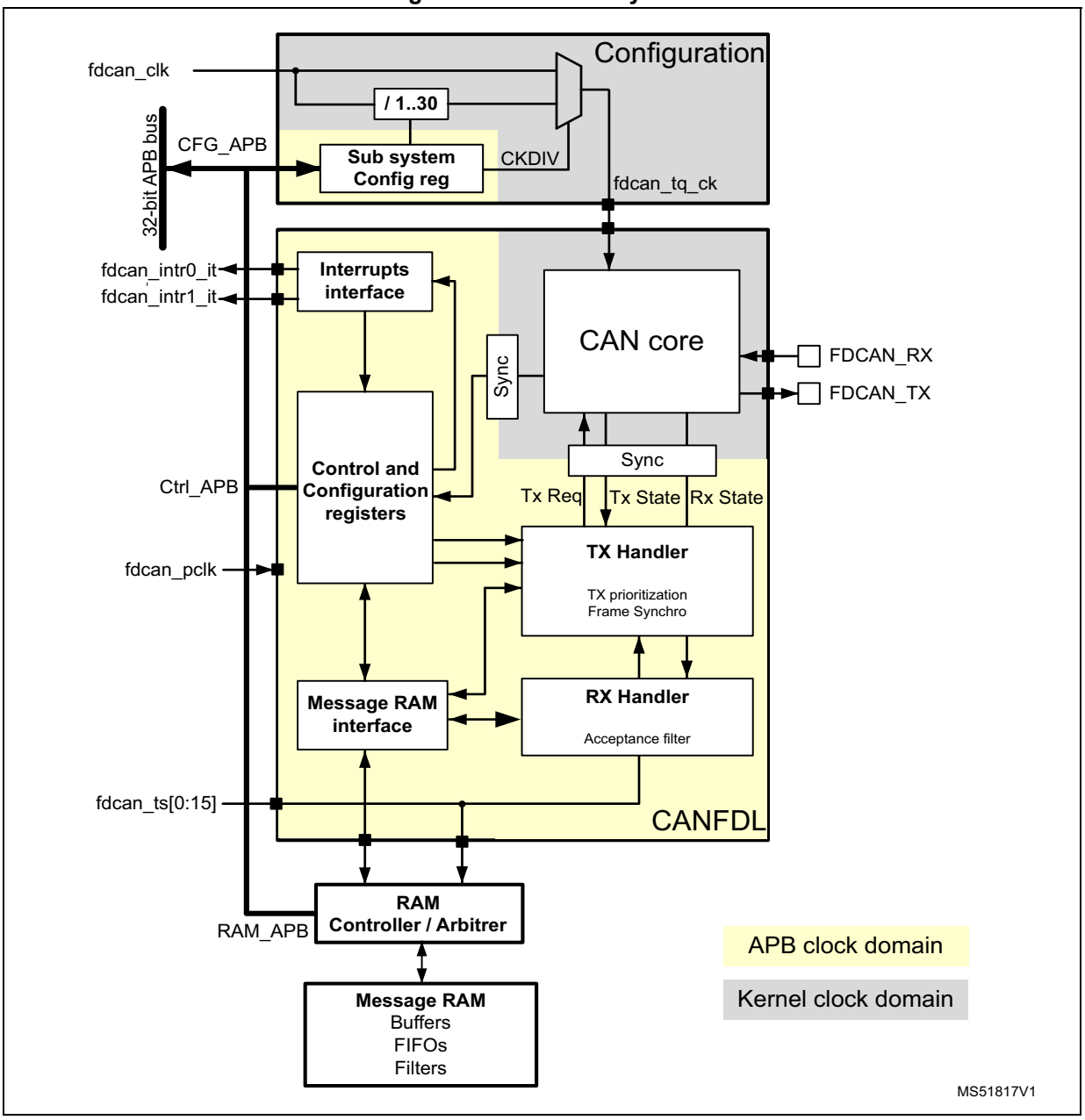
A 0.8-Kbyte message RAM per FDCAN instance implements filters, receive FIFOs, transmit event FIFOs and transmit FIFOs.

The CAN subsystem I/O signals and pins are detailed, respectively, in *Table 626* and *Figure 763*.

Table 626. 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
fdcan_intr0_it	Digital output	FDCAN interrupt0
fdcan_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 763. CAN subsystem

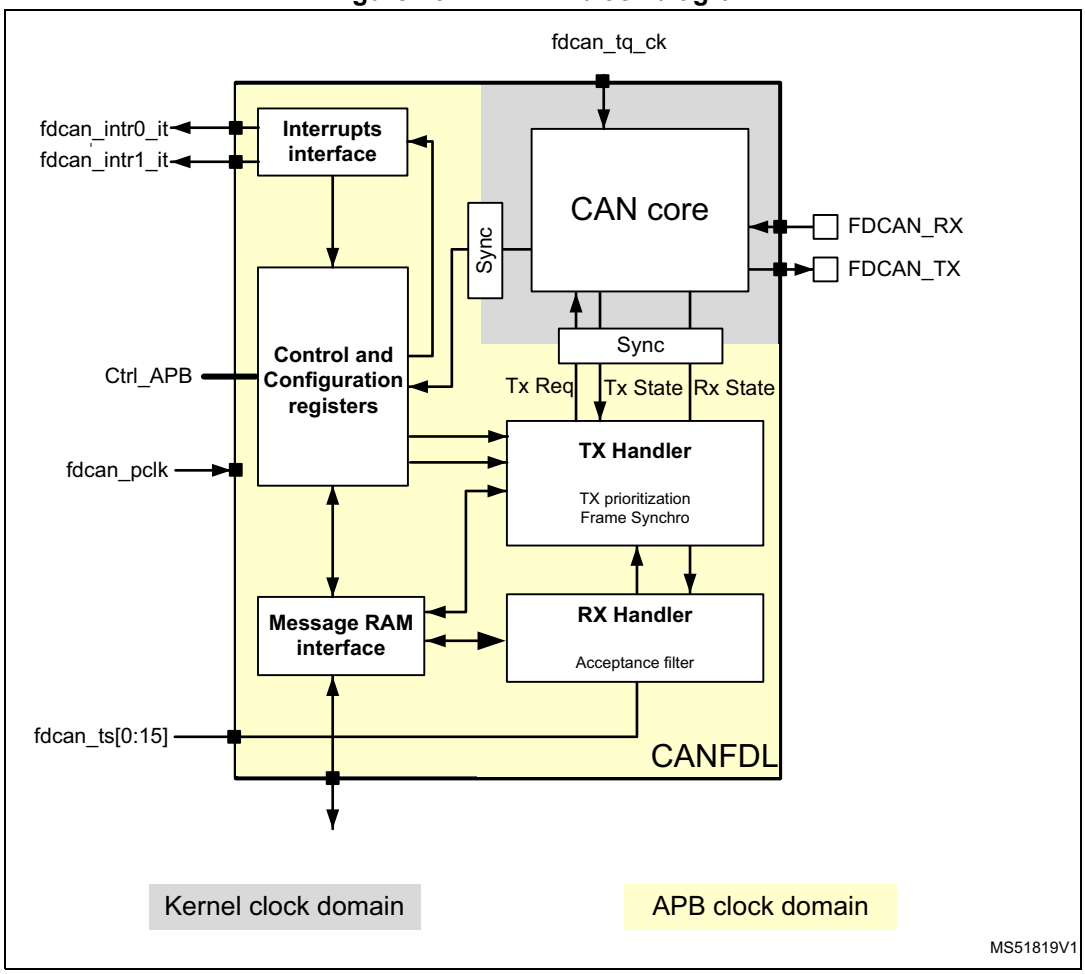


62.2 FDCAN main features

- 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

62.3 FDCAN functional description

Figure 764. 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.

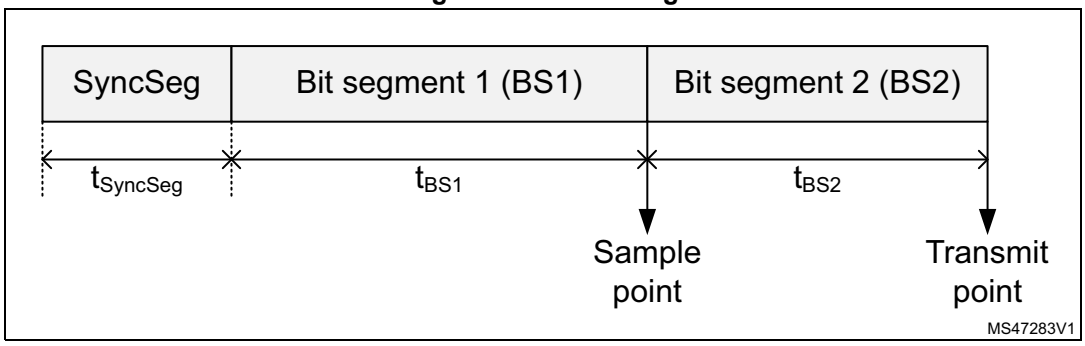
62.3.1 Bit timing

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 765*, 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 765. 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 765* 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.

62.3.2 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 627*.

Table 627. 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).

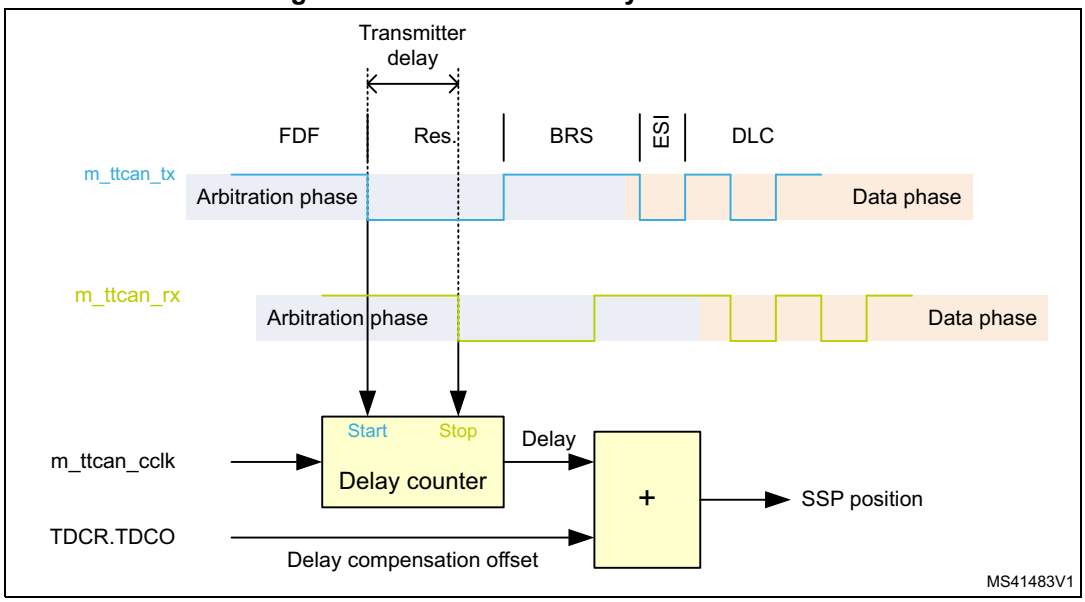
PSR.TDCV shows the actual transmitter delay compensation value. PSR.TDCV is cleared when CCCR.INIT is set and is updated at each transmission of an FD frame while DBTP.TDC is set.

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 766. 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.

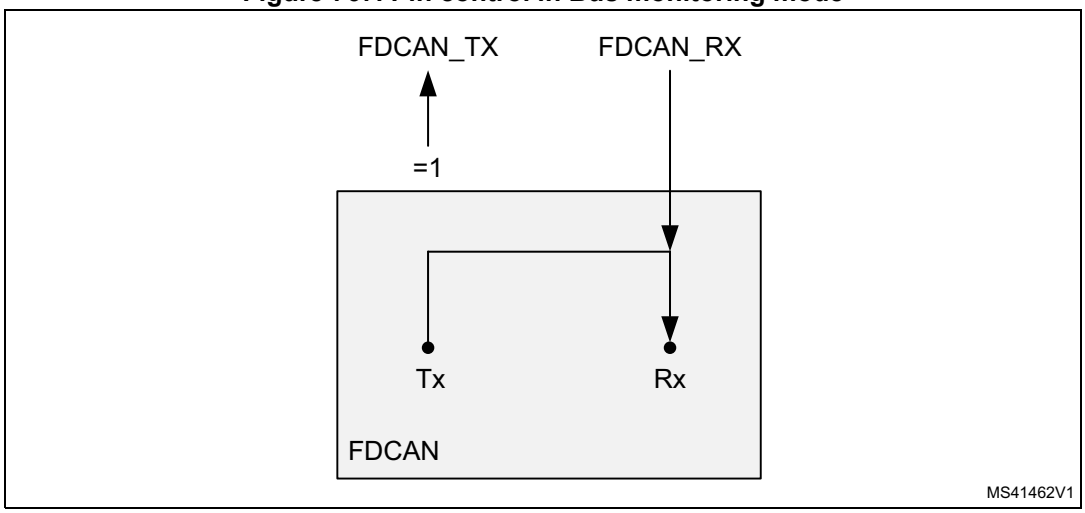
Note: The Restricted operation mode must not be combined with the Loop back mode (internal or external).

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 767* shows the connection of FDCAN_TX and FDCAN_RX signals to the FDCAN in Bus monitoring mode.

Figure 767. 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

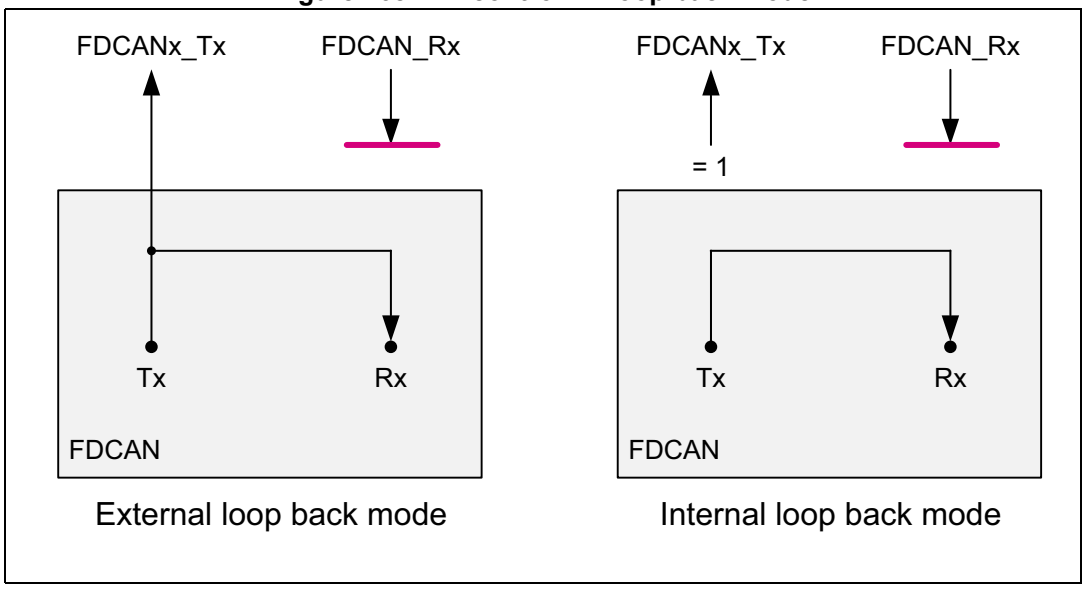
The FDCAN can be set in External loop back mode by programming TEST.LBCK to 1. In Loop Back mode, the FDCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. *Figure 768* shows the connection of transmit and receive signals FDCAN_TX 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 768* shows the connection of FDCAN_TX and FDCAN_RX pins to the FDCAN in case of Internal loop back mode.

Figure 768. 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.

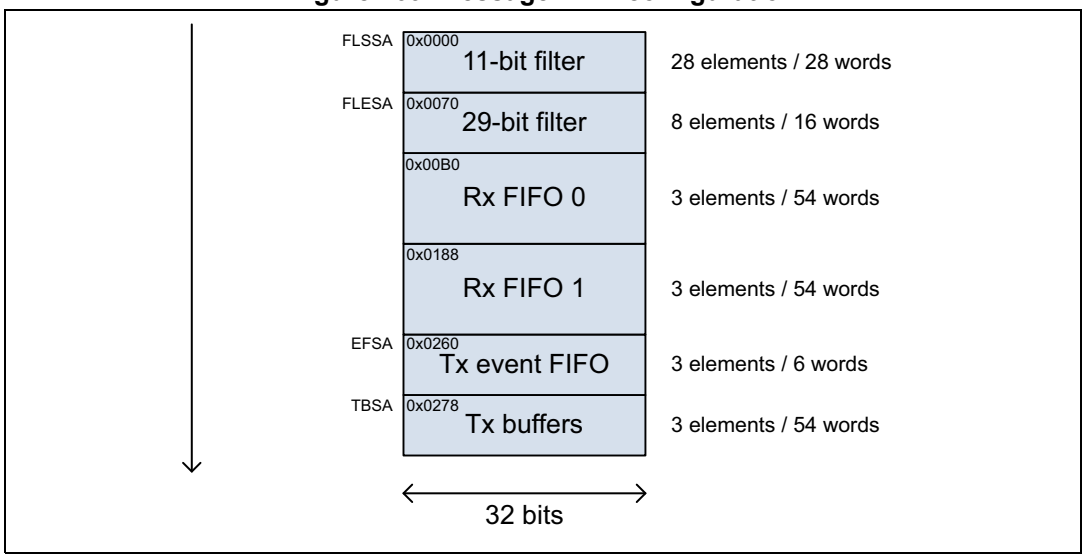
When the counter reaches 0, interrupt flag IR[TOO] is set. In Continuous mode, the counter is immediately restarted at TOCC[TOP].

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.

62.3.3 Message RAM

The Message RAM has a 32-bit width, 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 769*.

Figure 769. 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 0x034C (as in *Figure 769*)
- FDCAN2:
  - start address = 0x034C (FDCAN1 end address) + 4 = 0x0350
  - end address = 0x0350 (FDCAN2 start address) + 0x0350 - 4 = 0x069C.

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.

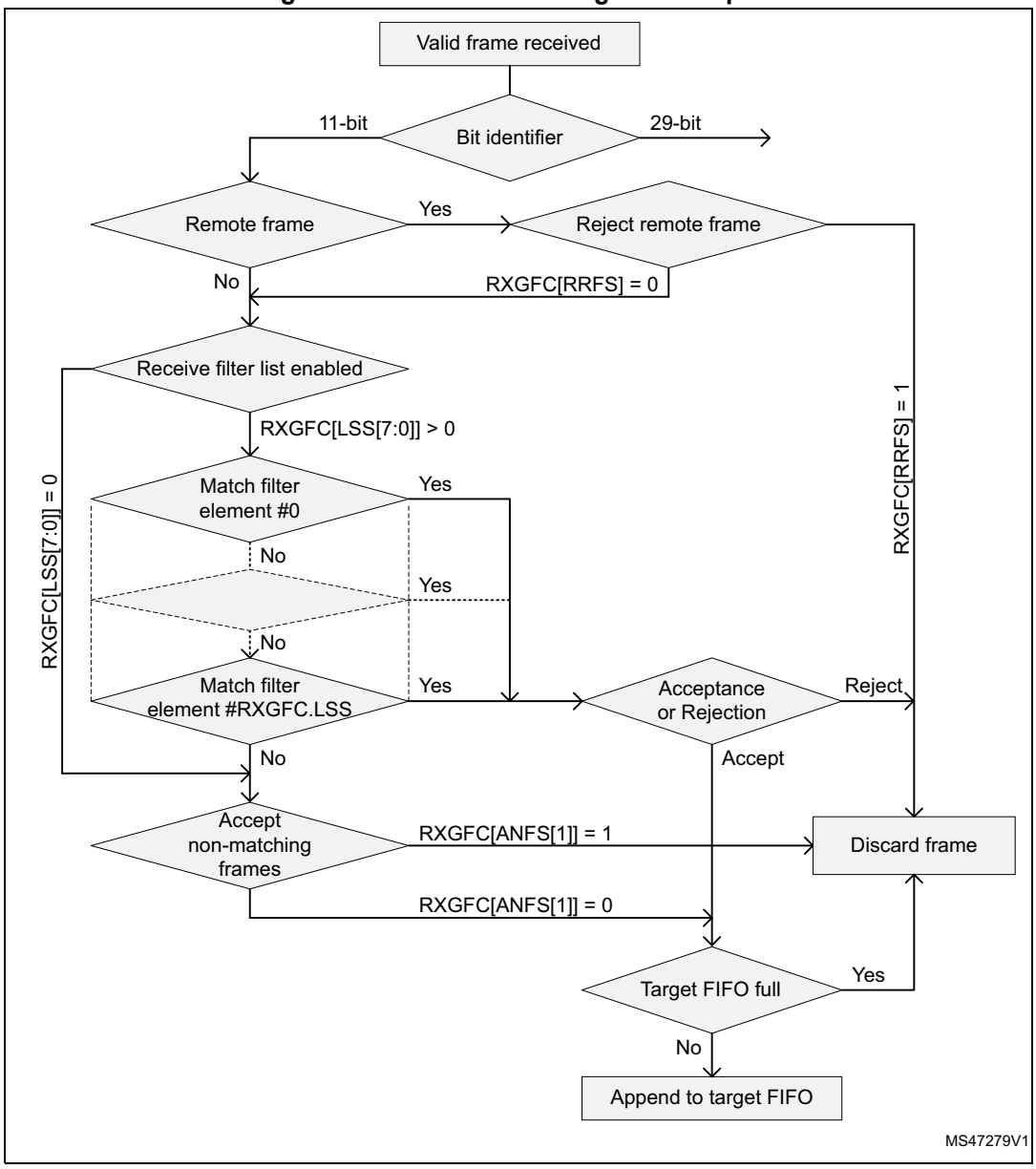
In case all mask bits are one, a match occurs only when the received Message ID and the Message ID filter are identical. If all mask bits are 0, all Message IDs match.

Standard message ID filtering

Figure 770 shows the flow for standard message ID (11-bit Identifier) filtering. The standard message ID filter element is described in *Section 62.3.8*.

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 770. Standard Message ID filter path

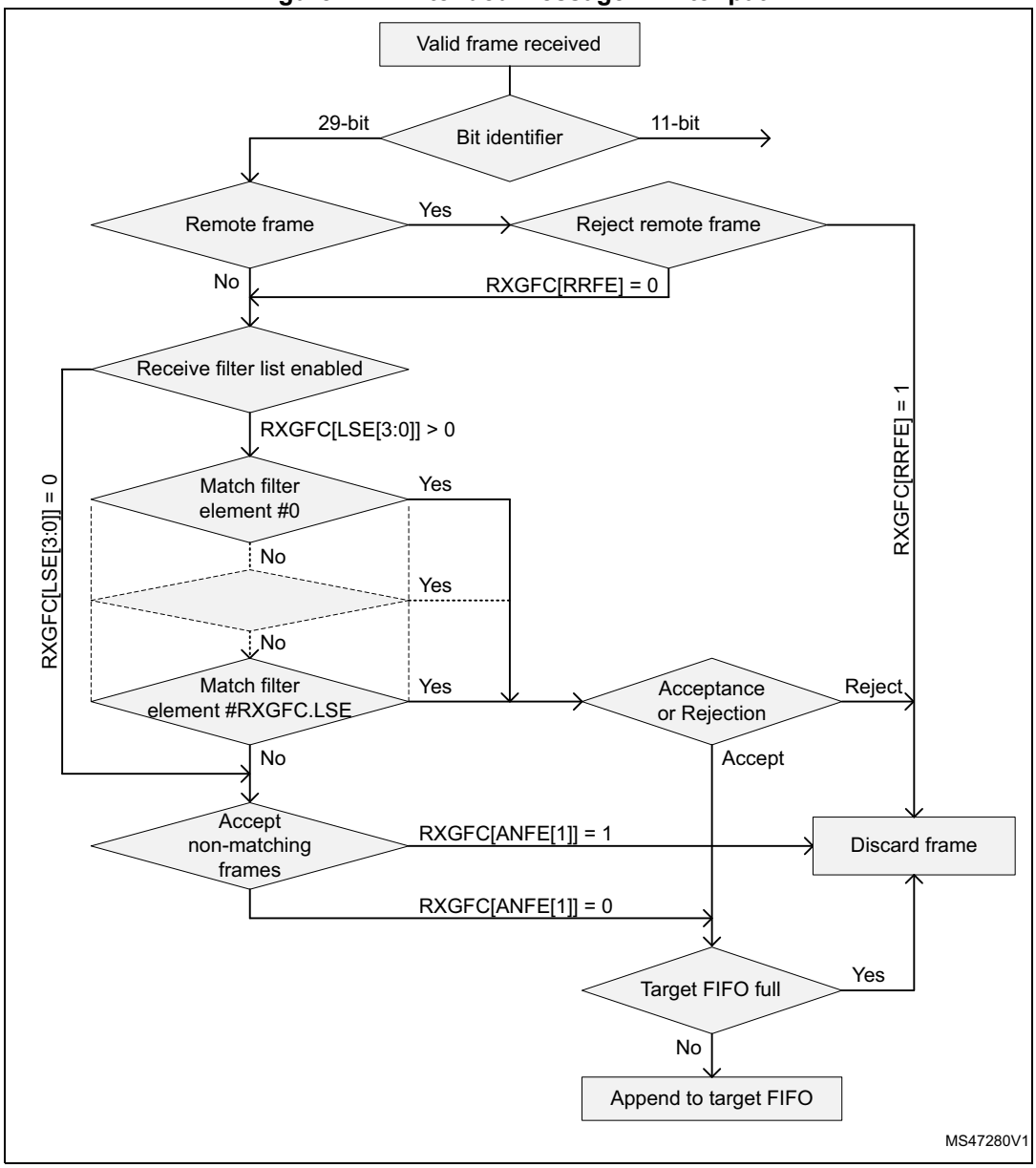


Extended message ID filtering

Figure 771 shows the flow for extended message ID (29-bit Identifier) filtering. The Extended Message ID filter element is described in *Section 62.3.9*.

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 771. 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 62.3.5*.

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 628. 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.

62.3.4 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 62.4.23* and *Section 62.4.25*. 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.

62.3.5 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 629*, the description is provided in *Table 630*.

Table 629. 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 630. 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 630. 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 630. 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

62.3.6 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 631. 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 632. 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 632. 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.

62.3.7 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 633. 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 634. 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 634. 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].

62.3.8 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 635. 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 636. 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.

62.3.9 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 637. 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 638. 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.

62.4 FDCAN registers

62.4.1 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

62.4.2 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.

62.4.3 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.

62.4.4 FDCAN test register (FDCAN_TEST)

Write access to the this register is enabled by setting bit CCCR[TEST] to 1. All 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.

62.4.5 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.

62.4.6 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.

62.4.7 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.

62.4.8 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.

62.4.9 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.

62.4.10 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*.

62.4.11 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.

62.4.12 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.

62.4.13 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.

62.4.14 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.

62.4.15 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

62.4.16 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

62.4.17 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

62.4.18 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

62.4.19 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 770* and *Figure 771*.

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.

62.4.20 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.

62.4.21 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.

62.4.22 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.

62.4.23 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].

62.4.24 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.

62.4.25 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].

62.4.26 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.

62.4.27 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).

62.4.28 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.

62.4.29 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.

62.4.30 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

62.4.31 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

62.4.32 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

62.4.33 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

62.4.34 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

62.4.35 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.

62.4.36 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].

62.4.37 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.

62.4.38 FDCAN register map

Table 639. 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 639. FDCAN register map and reset values(1) (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0044	FDCAN_PSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[6:0]							Res.	PXE	REDL	RBRSRESI1	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]		
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x0048	FDCAN_TDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCO[6:0]							Res.	TDCF[6:0]							
	Reset value																		0	0	0	0	0	0		0	0	0	0	0	0	0	0
0x004C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x0050	FDCAN_IR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARA	PED	PEA	WDI	BO	EW	EP	ELO	TOO	MRAF	TSW	TEFL	TEFF	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1N	RF0L	RF0F	RF0N
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0054	FDCAN_IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	TOOE	MRAFE	TSWE	TEFLE	TEFFE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1NE	RF0LE	RF0FE	RF0NE
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0058	FDCAN_ILS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERR	BERR	MISC	TFERR	SMSG	RXFIFO1	RXFIFO0
	Reset value	0																									0	0	0	0	0	0	0
0x005C	FDCAN_ILE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EINT1	EINT0
	Reset value																															0	0
0x0060 to 0x007C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x0080	FDCAN _RXGFC	Res.	Res.	Res.	Res.	LSE[3:0].				Res.	Res.	Res.	LSS[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	F0OM	F1OM	Res.	Res.	ANFS[1:0]		ANFE[1:0]		RRFS	RRFE
	Reset value					0	0	0	0				0	0	0	0	0							0	0			0	0	0	0	0	0
0x0084	FDCAN _XIDAM	Res.	Res.	Res.	EIDM[28:0]																												
	Reset value				1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0088	FDCAN_HPMS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLST	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.	RF0L	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

Table 639. 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
0x0098	FDCAN_RXF1S	Res.	Res.	Res.	Res.	Res.	Res.	RF1L	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
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.	TEFL	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

Table 639. 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
0x00E8	FDCAN _TXEFA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EFAI [1:0]	
	Reset value																															0	0
0x0100	FDCAN _CKDIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PDIV[3:0]			
	Reset value																													0	0	0	0

1. R = Read, S = Set on read, X = Reset on read, W = Write, P = Protected write, p = Protected set, C = Clear/preset on write.

Refer to *Section 2.3 on page 115* for the register boundary addresses.

63 USB on-the-go full-speed (OTG_FS)

63.1 Introduction

This section presents the architecture and the programming model of the OTG_FS controller.

The following acronyms are used throughout the section:

FS	Full-speed
LS	Low-speed
MAC	Media access controller
OTG	On-the-go
PFC	Packet FIFO controller
PHY	Physical layer
USB	Universal serial bus
UTMI	USB 2.0 Transceiver Macrocell interface (UTMI)
LPM	Link power management
BCD	Battery charging detector
HNP	Host negotiation protocol
SRP	Session request protocol

References are made to the following documents:

- USB On-The-Go Supplement, Revision 2.0
- Universal Serial Bus Revision 2.0 Specification
- USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification, July 16, 2007
- Errata for USB 2.0 ECN: Link Power Management (LPM) - 7/2007
- Battery Charging Specification, Revision 1.2

The USB OTG is a dual-role device (DRD) controller that supports both device and host functions and is fully compliant with the *On-The-Go Supplement to the USB 2.0 Specification*. It can also be configured as a host-only or device-only controller, fully compliant with the *USB 2.0 Specification.* OTG_FS supports the speeds defined in the *Table 640: OTG_FS speeds supported* below. The USB OTG supports both HNP and SRP. The only external device required is a charge pump for V_{BUS} in OTG mode.

Table 640. OTG_FS speeds supported

-	HS (480 Mb/s)	FS (12 Mb/s)	LS (1.5 Mb/s)
Host mode	-	X	X
Device mode	-	X	-

63.2 OTG_FS main features

The main features can be divided into three categories: general, host-mode and device-mode features.

63.2.1 General features

The OTG_FS interface general features are the following:

- It is USB-IF certified to the Universal Serial Bus Specification Rev 2.0
- OTG_FS supports the following PHY interface:
  - An on-chip full-speed PHY
- It includes full support (PHY) for the optional On-The-Go (OTG) protocol detailed in the On-The-Go Supplement Rev 2.0 specification
  - Integrated support for A-B device identification (ID line)
  - Integrated support for host Negotiation protocol (HNP) and session request protocol (SRP)
  - It allows host to turn V_{BUS} off to conserve battery power in OTG applications
  - It supports OTG monitoring of V_{BUS} levels with internal comparators
  - It supports dynamic host-peripheral switch of role
- It is software-configurable to operate as:
  - SRP capable USB FS Peripheral (B-device)
  - SRP capable USB FS/LS host (A-device)
  - USB On-The-Go Full-Speed Dual Role device
- It supports FS SOF and LS Keep-alives with
  - SOF pulse PAD connectivity
  - SOF pulse internal connection to timer (TIMx)
  - Configurable framing period
  - Configurable end of frame interrupt
- It includes power saving features such as system stop during USB suspend, switch-off of clock domains internal to the digital core, PHY and DFIFO power management.
- It features a dedicated RAM of 1.25 Kbytes with advanced FIFO control:
  - Configurable partitioning of RAM space into different FIFOs for flexible and efficient use of RAM
  - Each FIFO can hold multiple packets
  - Dynamic memory allocation
  - Configurable FIFO sizes that are not powers of 2 to allow the use of contiguous memory locations
- It guarantees max USB bandwidth for up to one frame (1 ms) without system intervention.
- It supports charging port detection as described in Battery Charging Specification Revision 1.2.

63.2.2 Host-mode features

The OTG_FS interface main features and requirements in host-mode are the following:

- External charge pump for V_{BUS} voltage generation.
- Up to 12 host channels (pipes): each channel is dynamically reconfigurable to allocate any type of USB transfer.
- Built-in hardware scheduler holding:
  - Up to 12 interrupt plus isochronous transfer requests in the periodic hardware queue
  - Up to 12 control plus bulk transfer requests in the non-periodic hardware queue
- Management of a shared Rx FIFO, a periodic Tx FIFO and a nonperiodic Tx FIFO for efficient usage of the USB data RAM.

63.2.3 Peripheral-mode features

The OTG_FS interface main features in peripheral-mode are the following:

- 1 bidirectional control endpoint0
- 5 IN endpoints (EPs) configurable to support bulk, interrupt or isochronous transfers
- 5 OUT endpoints configurable to support bulk, interrupt or isochronous transfers
- Management of a shared Rx FIFO and a Tx-OUT FIFO for efficient usage of the USB data RAM
- Management of up to 6 dedicated Tx-IN FIFOs (one for each active IN EP) to put less load on the application
- Support for the soft disconnect feature.

63.3 OTG_FS implementation

Table 641. OTG_FS implementation(1)

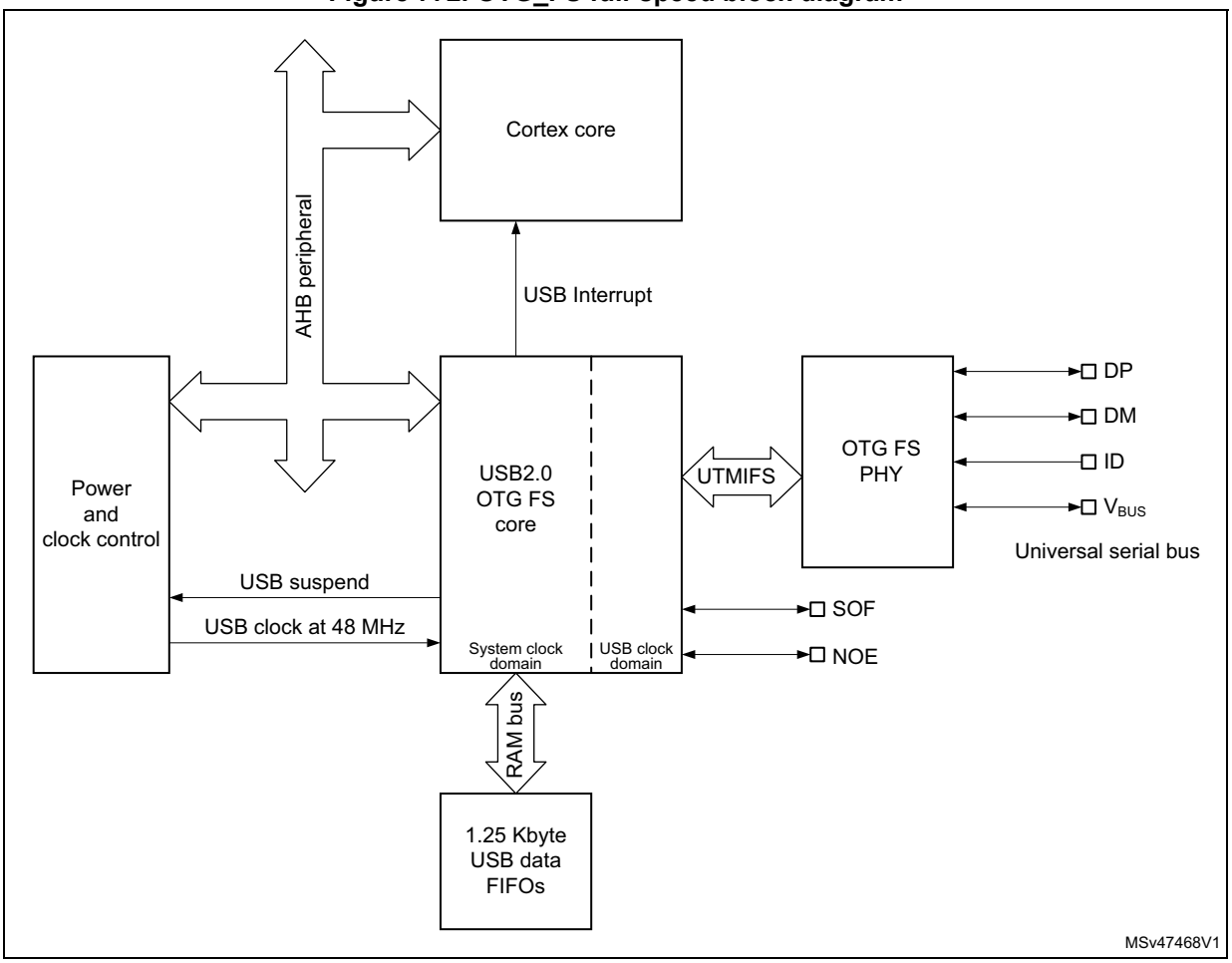
USB features	OTG_FS
Device bidirectional endpoints (including EP0)	6
Host mode channels	12
Size of dedicated SRAM	1.2 KB
USB 2.0 link power management (LPM) support	X
OTG revision supported	2.0
Attach detection protocol (ADP) support	-
Battery charging detection (BCD) support	X

1. “X” = supported, “-” not supported

63.4 OTG_FS functional description

63.4.1 OTG_FS block diagram

Figure 772. OTG_FS full-speed block diagram



63.4.2 OTG_FS pin and internal signals

Table 642. OTG_FS input/output pins

Signal name	Signal type	Description
OTG_FS_DP	Digital input/output	USB OTG D+ line
OTG_FS_DM	Digital input/output	USB OTG D- line
OTG_FS_ID	Digital input	USB OTG ID
OTG_FS_VBUS	Analog input	USB OTG VBUS
OTG_FS_SOF	Digital output	USB OTG Start Of Frame (visibility)
OTG_FS_NOE	Digital output	USB OTG output enable for D+/D- (visibility)

Table 643. OTG_FS input/output signals

Signal name	Signal type	Description
usb_sof	Digital output	USB OTG start-of-frame event for on chip peripherals
usb_wkup	Digital output	USB OTG wakeup event output
usb_gbl_it	Digital output	USB OTG global interrupt

63.4.3 OTG_FS core

The USB OTG_FS receives the 48 MHz clock from the reset and clock controller (RCC). This clock is used for driving the 48 MHz domain at full-speed (12 Mbit/s) and must be enabled prior to configuring the OTG core.

The CPU reads and writes from/to the OTG core registers through the AHB peripheral bus. It is informed of USB events through the single USB OTG interrupt line described in *Section 63.13: OTG_FS interrupts*.

The CPU submits data over the USB by writing 32-bit words to dedicated OTG locations (push registers). The data are then automatically stored into Tx-data FIFOs configured within the USB data RAM. There is one Tx FIFO push register for each in-endpoint (peripheral mode) or out-channel (host mode).

The CPU receives the data from the USB by reading 32-bit words from dedicated OTG addresses (pop registers). The data are then automatically retrieved from a shared Rx FIFO configured within the 1.25-Kbyte USB data RAM. There is one Rx FIFO pop register for each out-endpoint or in-channel.

The USB protocol layer is driven by the serial interface engine (SIE) and serialized over the USB by the transceiver module within the on-chip physical layer (PHY).

Caution: To guarantee a correct operation for the USB OTG FS peripheral, the AHB frequency should be higher than 14.2 MHz.

63.4.4 Embedded full-speed OTG PHY connected to OTG_FS

The embedded full-speed OTG PHY is controlled by the OTG FS core and conveys USB control & data signals through the full-speed subset of the UTMI+ Bus (UTMIFS). It provides the physical support to USB connectivity.
The full-speed OTG PHY includes the following components:

- FS/LS transceiver module used by both host and device. It directly drives transmission and reception on the single-ended USB lines.
- DP/DM integrated pull-up and pull-down resistors controlled by the OTG_FS core depending on the current role of the device. As a peripheral, it enables the DP pull-up resistor to signal full-speed peripheral connections as soon as V_{BUS} is sensed to be at a valid level (B-session valid). In host mode, pull-down resistors are enabled on both DP/DM. Pull-up and pull-down resistors are dynamically switched when the role of the device is changed via the host negotiation protocol (HNP).
- Pull-up/pull-down resistor ECN circuit. The DP pull-up consists of two resistors controlled separately from the OTG_FS as per the resistor Engineering Change Notice applied to USB Rev2.0. The dynamic trimming of the DP pull-up strength allows for better noise rejection and Tx/Rx signal quality.

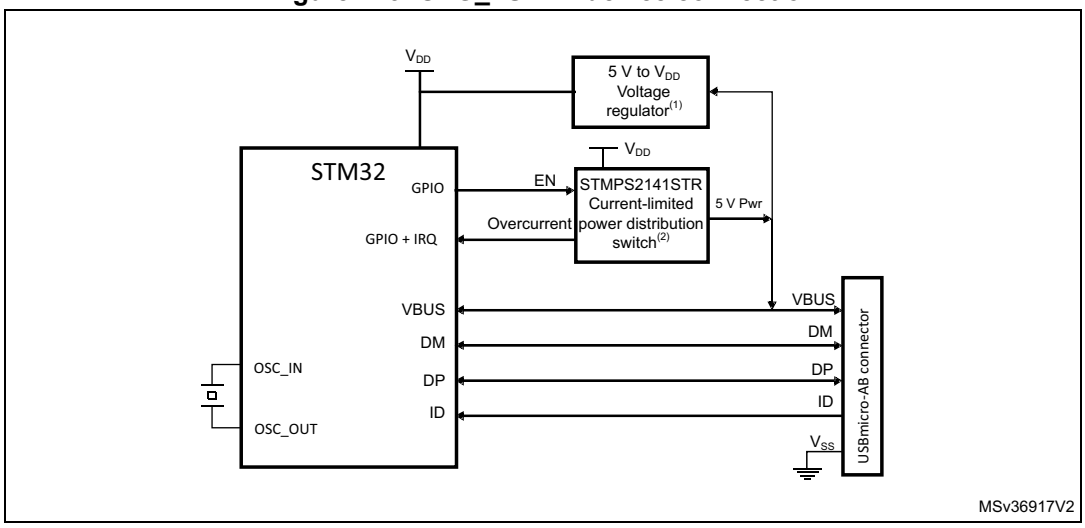
63.4.5 OTG detections

Additionally the OTG_FS uses the following functions:

- integrated ID pull-up resistor used to sample the ID line for A/B device identification.
- V_{BUS} sensing comparators with hysteresis used to detect V_{BUS} valid, A-B session valid and session-end voltage thresholds. They are used to drive the session request protocol (SRP), detect valid startup and end-of-session conditions, and constantly monitor the V_{BUS} supply during USB operations.

63.5 OTG_FS dual role device (DRD)

Figure 773. OTG_FS A-B device connection



1. External voltage regulator only needed when building a VBUS powered device.
2. STMPS2141STR needed only if the application has to support a VBUS powered device. A basic power switch can be used if 5 V are available on the application board.

63.5.1 ID line detection

The host or peripheral (the default) role is assumed depending on the ID input pin. The ID line status is determined on plugging in the USB cable, depending on whether a MicroA or MicroB plug is connected to the micro-AB receptacle.

- If the B-side of the USB cable is connected with a floating ID wire, the integrated pull-up resistor detects a high ID level and the default peripheral role is confirmed. In this configuration the OTG_FS complies with the standard FSM described in section 4.2.4: ID pin of the On-the-Go specification Rev2.0, supplement to the USB2.0.
- If the A-side of the USB cable is connected with a grounded ID, the OTG_FS issues an ID line status change interrupt (CIDSCHG bit in OTG_GINTSTS) for host software initialization, and automatically switches to the host role. In this configuration the OTG_FS complies with the standard FSM described by section 4.2.4: ID pin of the On-the-Go specification Rev2.0, supplement to the USB2.0.

63.5.2 HNP dual role device

The HNP capable bit in the Global USB configuration register (HNPCAP bit in OTG_ GUSBCFG) enables the OTG_FS core to dynamically change its role from A-host to A-peripheral and vice-versa, or from B-Peripheral to B-host and vice-versa according to the host negotiation protocol (HNP). The current device status can be read by the combined values of the connector ID status bit in the Global OTG control and status register (CIDSTS bit in OTG_GOTGCTL) and the current mode of operation bit in the global interrupt and status register (CMOD bit in OTG_GINTSTS).

The HNP program model is described in detail in *Section 63.16: OTG_FS programming model*.

63.5.3 SRP dual role device

The SRP capable bit in the global USB configuration register (SRPCAP bit in OTG_GUSBCFG) enables the OTG_FS core to switch off the generation of V_{BUS} for the A-device to save power. Note that the A-device is always in charge of driving V_{BUS} regardless of the host or peripheral role of the OTG_FS.

The SRP A/B-device program model is described in detail in *Section 63.16: OTG_FS programming model*.

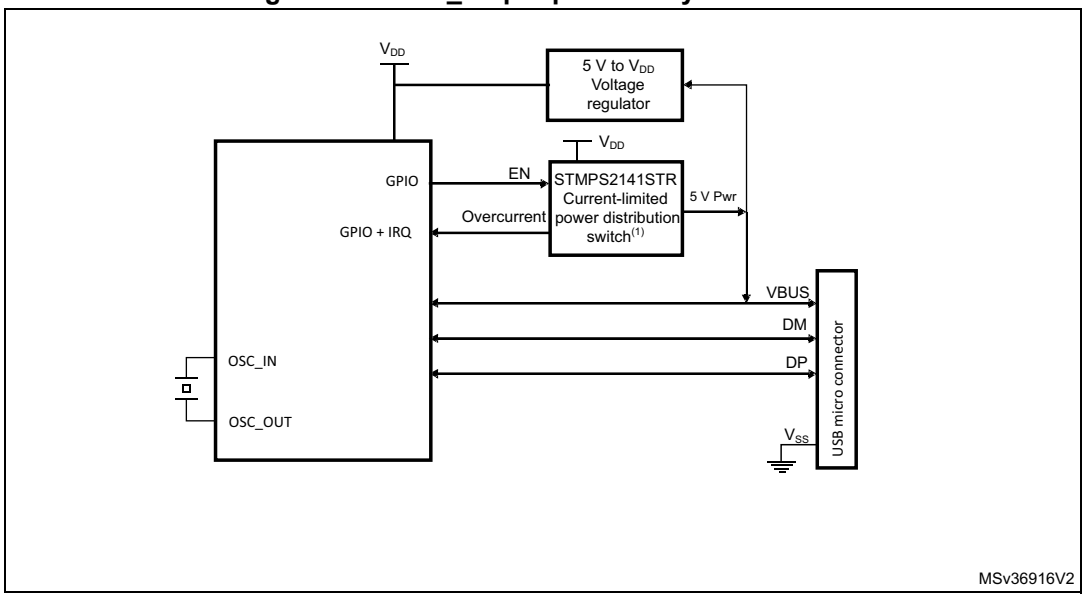
63.6 OTG_FS as a USB peripheral

This section gives the functional description of the OTG_FS in the USB peripheral mode. The OTG_FS works as an USB peripheral in the following circumstances:

- OTG B-Peripheral
  - OTG B-device default state if B-side of USB cable is plugged in
- OTG A-Peripheral
  - OTG A-device state after the HNP switches the OTG_FS to its peripheral role
- B-device
  - If the ID line is present, functional and connected to the B-side of the USB cable, and the HNP-capable bit in the Global USB Configuration register (HNPCAP bit in OTG_GUSBCFG) is cleared.
- Peripheral only (see *Figure 774: OTG_FS peripheral-only connection*)
  - The force device mode bit (FDMOD) in the *Section 63.15.4: OTG USB configuration register (OTG_GUSBCFG)* is set to 1, forcing the OTG_FS core to work as an USB peripheral-only. In this case, the ID line is ignored even if it is present on the USB connector.

Note: To build a bus-powered device implementation in case of the B-device or peripheral-only configuration, an external regulator has to be added, that generates the necessary power-supply from V_{BUS}.

Figure 774. OTG_FS peripheral-only connection



1. Use a regulator to build a bus-powered device.

63.6.1 SRP-capable peripheral

The SRP capable bit in the Global USB configuration register (SRPCAP bit in OTG_GUSBCFG) enables the OTG_FS to support the session request protocol (SRP). In this way, it allows the remote A-device to save power by switching off V_{BUS} while the USB session is suspended.

The SRP peripheral mode program model is described in detail in the *B-device session request protocol* section.

63.6.2 Peripheral states

Powered state

The V_{BUS} input detects the B-session valid voltage by which the USB peripheral is allowed to enter the powered state (see USB2.0 section 9.1). The OTG_FS then automatically connects the DP pull-up resistor to signal full-speed device connection to the host and generates the session request interrupt (SRQINT bit in OTG_GINTSTS) to notify the powered state.

The V_{BUS} input also ensures that valid V_{BUS} levels are supplied by the host during USB operations. If a drop in V_{BUS} below B-session valid happens to be detected (for instance because of a power disturbance or if the host port has been switched off), the OTG_FS automatically disconnects and the session end detected (SEDET bit in OTG_GOTGINT) interrupt is generated to notify that the OTG_FS has exited the powered state.

In the powered state, the OTG_FS expects to receive some reset signaling from the host. No other USB operation is possible. When a reset signaling is received the reset detected interrupt (USBRST in OTG_GINTSTS) is generated. When the reset signaling is complete, the enumeration done interrupt (ENUMDNE bit in OTG_GINTSTS) is generated and the OTG_FS enters the Default state.

Soft disconnect

The powered state can be exited by software with the soft disconnect feature. The DP pull-up resistor is removed by setting the soft disconnect bit in the device control register (SDIS bit in OTG_DCTL), causing a device disconnect detection interrupt on the host side even though the USB cable was not really removed from the host port.

Default state

In the Default state the OTG_FS expects to receive a SET_ADDRESS command from the host. No other USB operation is possible. When a valid SET_ADDRESS command is decoded on the USB, the application writes the corresponding number into the device address field in the device configuration register (DAD bit in OTG_DCFG). The OTG_FS then enters the address state and is ready to answer host transactions at the configured USB address.

Suspended state

The OTG_FS peripheral constantly monitors the USB activity. After counting 3 ms of USB idleness, the early suspend interrupt (ESUSP bit in OTG_GINTSTS) is issued, and confirmed 3 ms later, if appropriate, by the suspend interrupt (USBSUSP bit in OTG_GINTSTS). The device suspend bit is then automatically set in the device status register (SUSPSTS bit in OTG_DSTS) and the OTG_FS enters the suspended state.

The suspended state may optionally be exited by the device itself. In this case the application sets the remote wakeup signaling bit in the device control register (RWUSIG bit in OTG_DCTL) and clears it after 1 to 15 ms.

When a resume signaling is detected from the host, the resume interrupt (WKUPINT bit in OTG_GINTSTS) is generated and the device suspend bit is automatically cleared.

63.6.3 Peripheral endpoints

The OTG_FS core instantiates the following USB endpoints:

- Control endpoint 0:
  - Bidirectional and handles control messages only
  - Separate set of registers to handle in and out transactions
  - Proper control (OTG_DIEPCTL0/OTG_DOEPCTL0), transfer configuration (OTG_DIEPTSIZ0/OTG_DOEPTSIZ0), and status-interrupt (OTG_DIEPINT0/)OTG_DOEPINT0) registers. The available set of bits inside the control and transfer size registers slightly differs from that of other endpoints
- 5 IN endpoints
  - Each of them can be configured to support the isochronous, bulk or interrupt transfer type
  - Each of them has proper control (OTG_DIEPCTL*x*), transfer configuration (OTG_DIEPTSIZ*x*), and status-interrupt (OTG_DIEPINT*x*) registers
  - The device IN endpoints common interrupt mask register (OTG_DIEPMSK) is available to enable/disable a single kind of endpoint interrupt source on all of the IN endpoints (EP0 included)
  - Support for incomplete isochronous IN transfer interrupt (IISOIXFR bit in OTG_GINTSTS), asserted when there is at least one isochronous IN endpoint on

which the transfer is not completed in the current frame. This interrupt is asserted along with the end of periodic frame interrupt (OTG_GINTSTS/EOPF).

- 5 OUT endpoints
  - Each of them can be configured to support the isochronous, bulk or interrupt transfer type
  - Each of them has a proper control (OTG_DOEPCTL*x*), transfer configuration (OTG_DOEPTSIZ*x*) and status-interrupt (OTG_DOEPINT*x*) register
  - Device OUT endpoints common interrupt mask register (OTG_DOEPMSK) is available to enable/disable a single kind of endpoint interrupt source on all of the OUT endpoints (EP0 included)
  - Support for incomplete isochronous OUT transfer interrupt (INCOMPISOOUT bit in OTG_GINTSTS), asserted when there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the end of periodic frame interrupt (OTG_GINTSTS/EOPF).

Endpoint control

- The following endpoint controls are available to the application through the device endpoint-*x* IN/OUT control register (OTG_DIEPCTL*x*/OTG_DOEPCTL*x*):
  - Endpoint enable/disable
  - Endpoint activate in current configuration
  - Program USB transfer type (isochronous, bulk, interrupt)
  - Program supported packet size
  - Program Tx FIFO number associated with the IN endpoint
  - Program the expected or transmitted data0/data1 PID (bulk/interrupt only)
  - Program the even/odd frame during which the transaction is received or transmitted (isochronous only)
  - Optionally program the NAK bit to always negative-acknowledge the host regardless of the FIFO status
  - Optionally program the STALL bit to always stall host tokens to that endpoint
  - Optionally program the SNOOP mode for OUT endpoint not to check the CRC field of received data

Endpoint transfer

The device endpoint-*x* transfer size registers (OTG_DIEPTSIZ*x*/OTG_DOEPTSIZ*x*) allow the application to program the transfer size parameters and read the transfer status. Programming must be done before setting the endpoint enable bit in the endpoint control register. Once the endpoint is enabled, these fields are read-only as the OTG_FS core updates them with the current transfer status.

The following transfer parameters can be programmed:

- Transfer size in bytes
- Number of packets that constitute the overall transfer size

Endpoint status/interrupt

The device endpoint-*x* interrupt registers (OTG_DIEPINT*x*/OTG_DOPEPINT*x)* indicate the status of an endpoint with respect to USB- and AHB-related events. The application must read these registers when the OUT endpoint interrupt bit or the IN endpoint interrupt bit in

the core interrupt register (OEPINT bit in OTG_GINTSTS or IEPINT bit in OTG_GINTSTS, respectively) is set. Before the application can read these registers, it must first read the device all endpoints interrupt (OTG_DAINT) register to get the exact endpoint number for the device endpoint-*x* interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers

The peripheral core provides the following status checks and interrupt generation:

- Transfer completed interrupt, indicating that data transfer was completed on both the application (AHB) and USB sides
- Setup stage has been done (control-out only)
- Associated transmit FIFO is half or completely empty (in endpoints)
- NAK acknowledge has been transmitted to the host (isochronous-in only)
- IN token received when Tx FIFO was empty (bulk-in/interrupt-in only)
- Out token received when endpoint was not yet enabled
- Babble error condition has been detected
- Endpoint disable by application is effective
- Endpoint NAK by application is effective (isochronous-in only)
- More than 3 back-to-back setup packets were received (control-out only)
- Timeout condition detected (control-in only)
- Isochronous out packet has been dropped, without generating an interrupt

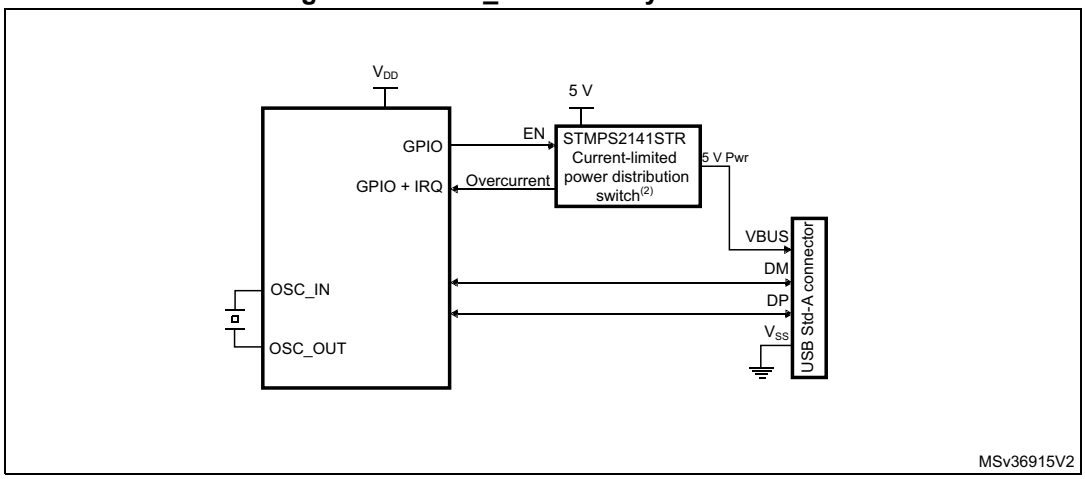
63.7 OTG_FS as a USB host

This section gives the functional description of the OTG_FS in the USB host mode. The OTG_FS works as a USB host in the following circumstances:

- OTG A-host
  - OTG A-device default state when the A-side of the USB cable is plugged in
- OTG B-host
  - OTG B-device after HNP switching to the host role
- A-device
  - If the ID line is present, functional and connected to the A-side of the USB cable, and the HNP-capable bit is cleared in the Global USB Configuration register (HNPCAP bit in OTG_GUSBCFG). Integrated pull-down resistors are automatically set on the DP/DM lines.
- Host only
  - The force host mode bit (FHMOD) in the *OTG USB configuration register (OTG_GUSBCFG)* forces the OTG_FS core to work as a USB host-only. In this case, the ID line is ignored even if present on the USB connector. Integrated pull-down resistors are automatically set on the DP/DM lines.

Note: On-chip 5 V V_{BUS} generation is not supported. For this reason, a charge pump or, if 5 V are available on the application board, a basic power switch must be added externally to drive the 5 V V_{BUS} line. The external charge pump can be driven by any GPIO output. This is required for the OTG A-host, A-device and host-only configurations.

Figure 775. OTG_FS host-only connection



1. V_{DD} range is between 2 V and 3.6 V.

63.7.1 SRP-capable host

SRP support is available through the SRP capable bit in the global USB configuration register (SRPCAP bit in OTG_GUSBCFG). With the SRP feature enabled, the host can save power by switching off the V_{BUS} power while the USB session is suspended.

The SRP host mode program model is described in detail in the *A-device session request protocol*) section.

63.7.2 USB host states

Host port power

On-chip 5 V V_{BUS} generation is not supported. For this reason, a charge pump or, if 5 V are available on the application board, a basic power switch, must be added externally to drive the 5 V V_{BUS} line. The external charge pump can be driven by any GPIO output or via an I^2C interface connected to an external PMIC (power management IC). When the application decides to power on V_{BUS}, it must also set the port power bit in the host port control and status register (PPWR bit in OTG_HPRT).

V_{BUS} valid

When HNP or SRP is enabled the VBUS sensing pin should be connected to V_{BUS}. The V_{BUS} input ensures that valid V_{BUS} levels are supplied by the charge pump during USB operations. Any unforeseen V_{BUS} voltage drop below the V_{BUS} valid threshold (4.4 V) leads to an OTG interrupt triggered by the session end detected bit (SEDET bit in OTG_GOTGINT). The application is then required to remove the V_{BUS} power and clear the port power bit.

When HNP and SRP are both disabled, the VBUS sensing pin does not need to be connected to V_{BUS}.

The charge pump overcurrent flag can also be used to prevent electrical damage. Connect the overcurrent flag output from the charge pump to any GPIO input and configure it to generate a port interrupt on the active level. The overcurrent ISR must promptly disable the V_{BUS} generation and clear the port power bit.

Host detection of a peripheral connection

If SRP or HNP are enabled, even if USB peripherals or B-devices can be attached at any time, the OTG_FS does not detect any bus connection until V_{BUS} is no longer sensed at a valid level (5 V). When V_{BUS} is at a valid level and a remote B-device is attached, the OTG_FS core issues a host port interrupt triggered by the device connected bit in the host port control and status register (PCDET bit in OTG_HPRT).

When HNP and SRP are both disabled, USB peripherals or B-device are detected as soon as they are connected. The OTG_FS core issues a host port interrupt triggered by the device connected bit in the host port control and status (PCDET bit in OTG_HPRT).

Host detection of peripheral a disconnection

The peripheral disconnection event triggers the disconnect detected interrupt (DISCINT bit in OTG_GINTSTS).

Host enumeration

After detecting a peripheral connection the host must start the enumeration process by sending USB reset and configuration commands to the new peripheral.

Before starting to drive a USB reset, the application waits for the OTG interrupt triggered by the debounce done bit (DBCDNE bit in OTG_GOTGINT), which indicates that the bus is stable again after the electrical debounce caused by the attachment of a pull-up resistor on DP (FS) or DM (LS).

The application drives a USB reset signaling (single-ended zero) over the USB by keeping the port reset bit set in the host port control and status register (PRST bit in OTG_HPRT) for a minimum of 10 ms and a maximum of 20 ms. The application takes care of the timing count and then of clearing the port reset bit.

Once the USB reset sequence has completed, the host port interrupt is triggered by the port enable/disable change bit (PENCHNG bit in OTG_HPRT). This informs the application that the speed of the enumerated peripheral can be read from the port speed field in the host port control and status register (PSPD bit in OTG_HPRT) and that the host is starting to drive SOFs (FS) or Keep alives (LS). The host is now ready to complete the peripheral enumeration by sending peripheral configuration commands.

Host suspend

The application decides to suspend the USB activity by setting the port suspend bit in the host port control and status register (PSUSP bit in OTG_HPRT). The OTG_FS core stops sending SOFs and enters the suspended state.

The suspended state can be optionally exited on the remote device’s initiative (remote wakeup). In this case the remote wakeup interrupt (WKUPINT bit in OTG_GINTSTS) is generated upon detection of a remote wakeup signaling, the port resume bit in the host port control and status register (PRES bit in OTG_HPRT) self-sets, and resume signaling is automatically driven over the USB. The application must time the resume window and then clear the port resume bit to exit the suspended state and restart the SOF.

If the suspended state is exited on the host initiative, the application must set the port resume bit to start resume signaling on the host port, time the resume window and finally clear the port resume bit.

63.7.3 Host channels

The OTG_FS core instantiates 12 host channels. Each host channel supports an USB host transfer (USB pipe). The host is not able to support more than 12 transfer requests at the same time. If more than 12 transfer requests are pending from the application, the host controller driver (HCD) must re-allocate channels when they become available from previous duty, that is, after receiving the transfer completed and channel halted interrupts.

Each host channel can be configured to support in/out and any type of periodic/nonperiodic transaction. Each host channel makes us of proper control (OTG_HCCHAR*x*), transfer configuration (OTG_HCTSIZ*x*) and status/interrupt (OTG_HCINT*x*) registers with associated mask (OTG_HCINTMSK*x*) registers.

Host channel control

- The following host channel controls are available to the application through the host channel-*x* characteristics register (OTG_HCCHAR*x*):
  - Channel enable/disable
  - Program the FS/LS speed of target USB peripheral
  - Program the address of target USB peripheral
  - Program the endpoint number of target USB peripheral
  - Program the transfer IN/OUT direction
  - Program the USB transfer type (control, bulk, interrupt, isochronous)
  - Program the maximum packet size (MPS)
  - Program the periodic transfer to be executed during odd/even frames

Host channel transfer

The host channel transfer size registers (OTG_HCTSIZ*x*) allow the application to program the transfer size parameters, and read the transfer status. Programming must be done before setting the channel enable bit in the host channel characteristics register. Once the endpoint is enabled the packet count field is read-only as the OTG_FS core updates it according to the current transfer status.

- The following transfer parameters can be programmed:
  - transfer size in bytes
  - number of packets making up the overall transfer size
  - initial data PID

Host channel status/interrupt

The host channel-*x* interrupt register (OTG_HCINT*x*) indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read these register when the host channels interrupt bit in the core interrupt register (HCINT bit in OTG_GINTSTS) is set. Before the application can read these registers, it must first read the host all channels interrupt (OTG_HAINT) register to get the exact channel number for the host channel-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_HAINT and OTG_GINTSTS registers.

The mask bits for each interrupt source of each channel are also available in the OTG_HCINTMSKx register.

- The host core provides the following status checks and interrupt generation:
  - Transfer completed interrupt, indicating that the data transfer is complete on both the application (AHB) and USB sides
  - Channel has stopped due to transfer completed, USB transaction error or disable command from the application
  - Associated transmit FIFO is half or completely empty (IN endpoints)
  - ACK response received
  - NAK response received
  - STALL response received
  - USB transaction error due to CRC failure, timeout, bit stuff error, false EOP
  - Babble error
  - frame overrun
  - data toggle error

63.7.4 Host scheduler

The host core features a built-in hardware scheduler which is able to autonomously re-order and manage the USB transaction requests posted by the application. At the beginning of each frame the host executes the periodic (isochronous and interrupt) transactions first, followed by the nonperiodic (control and bulk) transactions to achieve the higher level of priority granted to the isochronous and interrupt transfer types by the USB specification.

The host processes the USB transactions through request queues (one for periodic and one for nonperiodic). Each request queue can hold up to 8 entries. Each entry represents a pending transaction request from the application, and holds the IN or OUT channel number along with other information to perform a transaction on the USB. The order in which the requests are written to the queue determines the sequence of the transactions on the USB interface.

At the beginning of each frame, the host processes the periodic request queue first, followed by the nonperiodic request queue. The host issues an incomplete periodic transfer interrupt (IPXFR bit in OTG_GINTSTS) if an isochronous or interrupt transaction scheduled for the current frame is still pending at the end of the current frame. The OTG_FS core is fully responsible for the management of the periodic and nonperiodic request queues.The periodic transmit FIFO and queue status register (OTG_HPTXSTS) and nonperiodic transmit FIFO and queue status register (OTG_HNPTXSTS) are read-only registers which can be used by the application to read the status of each request queue. They contain:

- The number of free entries currently available in the periodic (nonperiodic) request queue (8 max)
- Free space currently available in the periodic (nonperiodic) Tx FIFO (out-transactions)
- IN/OUT token, host channel number and other status information.

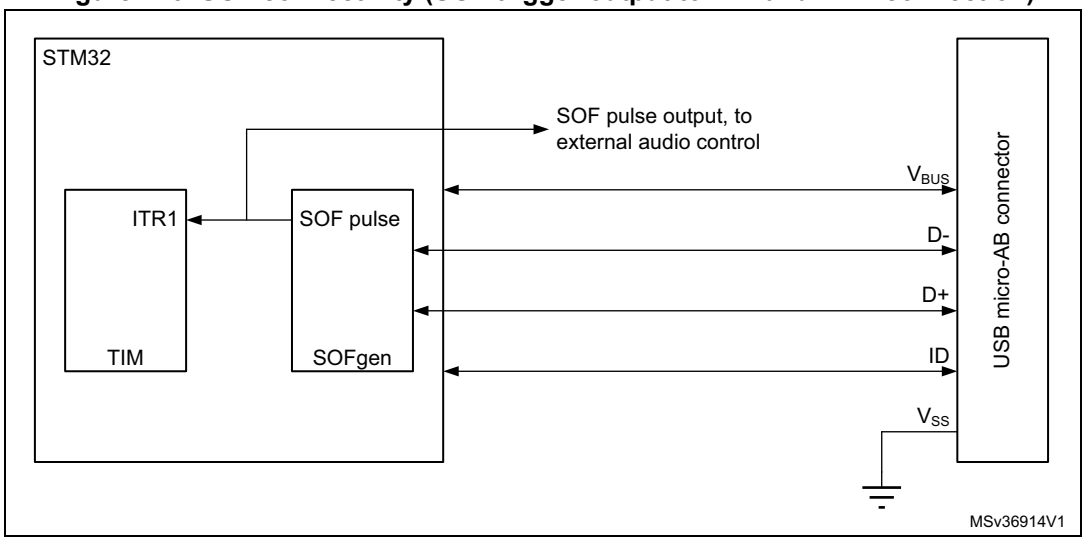
As request queues can hold a maximum of 8 entries each, the application can push to schedule host transactions in advance with respect to the moment they physically reach the SB for a maximum of 8 pending periodic transactions plus 8 pending non-periodic transactions.

To post a transaction request to the host scheduler (queue) the application must check that there is at least 1 entry available in the periodic (nonperiodic) request queue by reading the

PTXQSAV bits in the OTG_HNPTXSTS register or NPTQXSAV bits in the OTG_HNPTXSTS register.

63.8 OTG_FS SOF trigger

Figure 776. SOF connectivity (SOF trigger output to TIM and ITR1 connection)



The OTG_FS core provides means to monitor, track and configure SOF framing in the host and peripheral, as well as an SOF pulse output connectivity feature.

Such utilities are especially useful for adaptive audio clock generation techniques, where the audio peripheral needs to synchronize to the isochronous stream provided by the PC, or the host needs to trim its framing rate according to the requirements of the audio peripheral.

63.8.1 Host SOFs

In host mode the number of PHY clocks occurring between the generation of two consecutive SOF (FS) or Keep-alive (LS) tokens is programmable in the host frame interval register (HFIR), thus providing application control over the SOF framing period. An interrupt is generated at any start of frame (SOF bit in OTG_GINTSTS). The current frame number and the time remaining until the next SOF are tracked in the host frame number register (HFNUM).

A SOF pulse signal, is generated at any SOF starting token and with a width of 20 HCLK cycles. The SOF pulse is also internally connected to the input trigger of the timer, so that the input capture feature, the output compare feature and the timer can be triggered by the SOF pulse.

63.8.2 Peripheral SOFs

In device mode, the start of frame interrupt is generated each time an SOF token is received on the USB (SOF bit in OTG_GINTSTS). The corresponding frame number can be read from the device status register (FNSOF bit in OTG_DSTS). A SOF pulse signal with a width of 20 HCLK cycles is also generated.The SOF pulse signal is also internally connected to the TIM input trigger, so that the input capture feature, the output compare feature and the timer can be triggered by the SOF pulse.

The end of periodic frame interrupt (OTG_GINTSTS/EOPF) is used to notify the application when 80%, 85%, 90% or 95% of the time frame interval elapsed depending on the periodic frame interval field in the device configuration register (PFIVL bit in OTG_DCFG). This feature can be used to determine if all of the isochronous traffic for that frame is complete.

63.9 OTG_FS low-power modes

Table 644 below defines the STM32 low power modes and their compatibility with the OTG.

Table 644. Compatibility of STM32 low power modes with the OTG

Mode	Description	USB compatibility
Run	MCU fully active	Required when USB not in suspend state.
Sleep	USB suspend exit causes the device to exit Sleep mode. Peripheral registers content is kept.	Available while USB is in suspend state.
Stop	USB suspend exit causes the device to exit Stop mode. Peripheral registers content is kept(1).	Available while USB is in suspend state.
Standby	Powered-down. The peripheral must be reinitialized after exiting Standby mode.	Not compatible with USB applications.

1. Within Stop mode there are different possible settings. Some restrictions may also exist, refer to *Section 10: Power control (PWR)* to understand which (if any) restrictions apply when using OTG.

The following bits and procedures reduce power consumption.

The power consumption of the OTG PHY is controlled by two or three bits in the general core configuration register, depending on OTG revision supported.

- PHY power down (OTG_GCCFG/PWRDWN)

  It switches on/off the full-speed transceiver module of the PHY. It must be preliminarily set to allow any USB operation
- V_{BUS} detection enable (OTG_GCCFG/VBDEN)

  It switches on/off the V_{BUS} sensing comparators associated with OTG operations

Power reduction techniques are available while in the USB suspended state, when the USB session is not yet valid or the device is disconnected.

- Stop PHY clock (STPPCLK bit in OTG_PCGCCTL)

  When setting the stop PHY clock bit in the clock gating control register, most of the 48 MHz clock domain internal to the OTG core is switched off by clock gating. The dynamic power consumption due to the USB clock switching activity is cut even if the 48 MHz clock input is kept running by the application

  Most of the transceiver is also disabled, and only the part in charge of detecting the asynchronous resume or remote wakeup event is kept alive.
- Gate HCLK (GATEHCLK bit in OTG_PCGCCTL)

  When setting the Gate HCLK bit in the clock gating control register, most of the system clock domain internal to the OTG_FS core is switched off by clock gating. Only the register read and write interface is kept alive. The dynamic power consumption due to

the USB clock switching activity is cut even if the system clock is kept running by the application for other purposes.

- USB system stop

  When the OTG_FS is in the USB suspended state, the application may decide to drastically reduce the overall power consumption by a complete shut down of all the clock sources in the system. USB System Stop is activated by first setting the Stop PHY clock bit and then configuring the system deep sleep mode in the power control system module (PWR).

  The OTG_FS core automatically reactivates both system and USB clocks by asynchronous detection of remote wakeup (as an host) or resume (as a device) signaling on the USB.

To save dynamic power, the USB data FIFO is clocked only when accessed by the OTG_FS core.

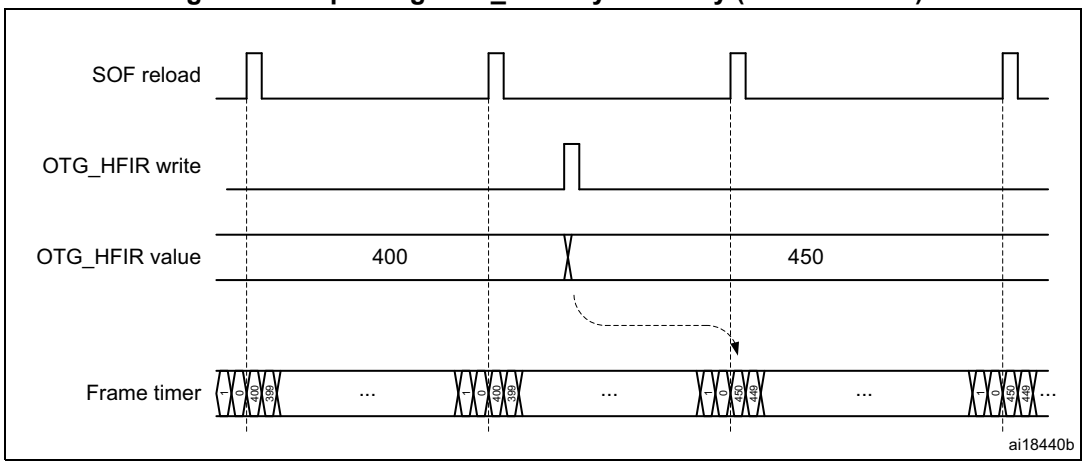
63.10 OTG_FS Dynamic update of the OTG_HFIR register

The USB core embeds a dynamic trimming capability of SOF framing period in host mode allowing to synchronize an external device with the SOF frames.

When the OTG_HFIR register is changed within a current SOF frame, the SOF period correction is applied in the next frame as described in *Figure 777*.

For a dynamic update, it is required to set RLDCTRL=1.

Figure 777. Updating OTG_HFIR dynamically (RLDCTRL = 1)

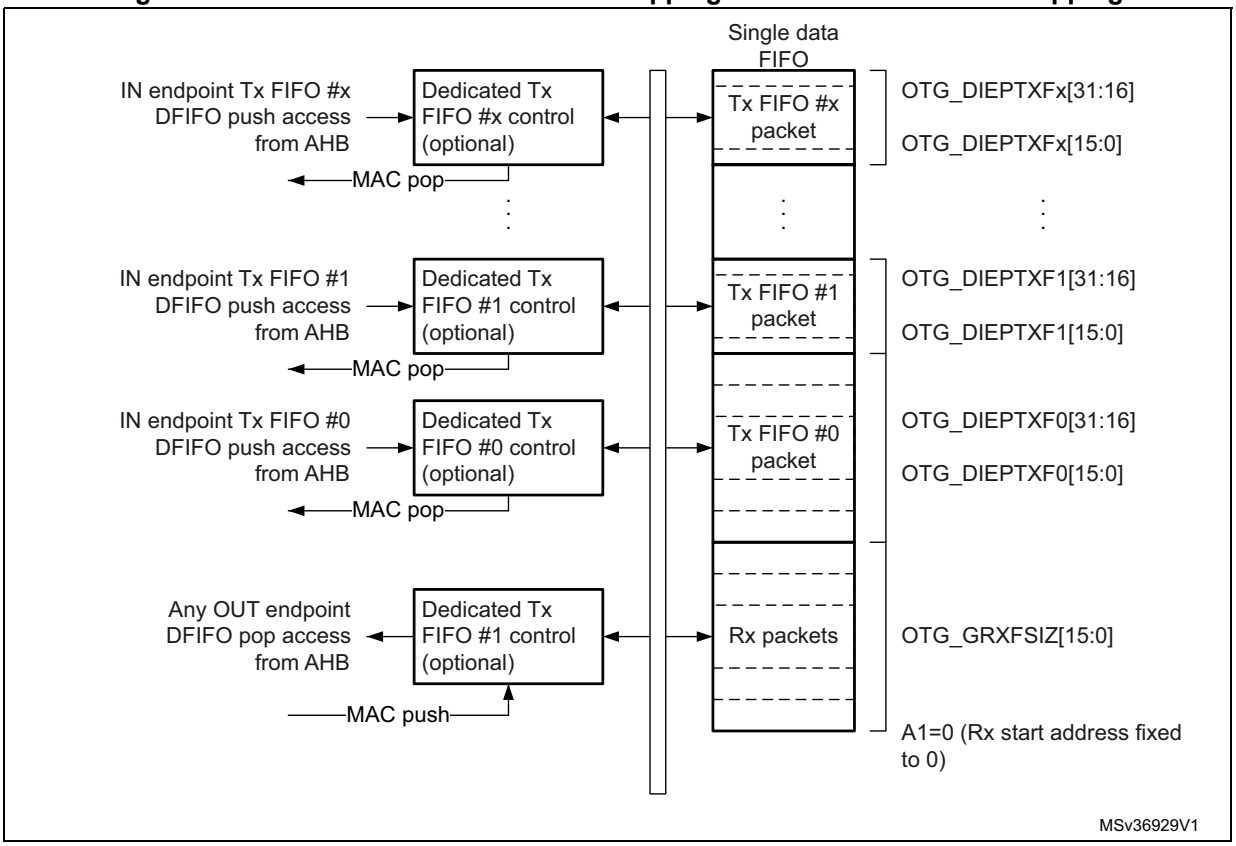


63.11 OTG_FS data FIFOs

The USB system features 1.25 Kbytes of dedicated RAM with a sophisticated FIFO control mechanism. The packet FIFO controller module in the OTG_FS core organizes RAM space into Tx FIFOs into which the application pushes the data to be temporarily stored before the USB transmission, and into a single Rx FIFO where the data received from the USB are temporarily stored before retrieval (popped) by the application. The number of instructed FIFOs and how these are organized inside the RAM depends on the device's role. In peripheral mode an additional Tx FIFO is instructed for each active IN endpoint. Any FIFO size is software configured to better meet the application requirements.

63.11.1 Peripheral FIFO architecture

Figure 778. Device-mode FIFO address mapping and AHB FIFO access mapping



Peripheral Rx FIFO

The OTG peripheral uses a single receive FIFO that receives the data directed to all OUT endpoints. Received packets are stacked back-to-back until free space is available in the Rx FIFO. The status of the received packet (which contains the OUT endpoint destination number, the byte count, the data PID and the validity of the received data) is also stored by the core on top of the data payload. When no more space is available, host transactions are NACKed and an interrupt is received on the addressed endpoint. The size of the receive FIFO is configured in the receive FIFO size register (OTG_GRXFSIZ).

The single receive FIFO architecture makes it more efficient for the USB peripheral to fill in the receive RAM buffer:

- All OUT endpoints share the same RAM buffer (shared FIFO)
- The OTG_FS core can fill in the receive FIFO up to the limit for any host sequence of OUT tokens

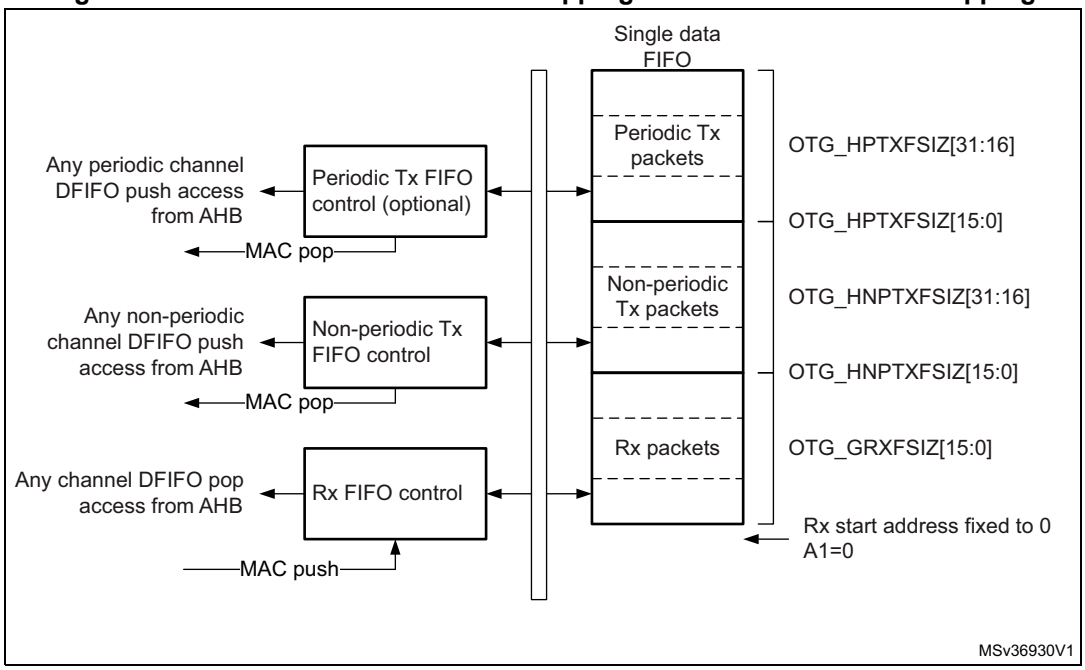
The application keeps receiving the Rx FIFO non-empty interrupt (RXFLVL bit in OTG_GINTSTS) as long as there is at least one packet available for download. It reads the packet information from the receive status read and pop register (OTG_GRXSTSP) and finally pops data off the receive FIFO by reading from the endpoint-related pop address.

Peripheral Tx FIFOs

The core has a dedicated FIFO for each IN endpoint. The application configures FIFO sizes by writing the endpoint 0 transmit FIFO size register (OTG_DIEPTXF0) for IN endpoint0 and the device IN endpoint transmit FIFOx registers (OTG_DIEPTXFx) for IN endpoint-x.

63.11.2 Host FIFO architecture

Figure 779. Host-mode FIFO address mapping and AHB FIFO access mapping



Host Rx FIFO

The host uses one receiver FIFO for all periodic and nonperiodic transactions. The FIFO is used as a receive buffer to hold the received data (payload of the received packet) from the USB until it is transferred to the system memory. Packets received from any remote IN endpoint are stacked back-to-back until free space is available. The status of each received packet with the host channel destination, byte count, data PID and validity of the received data are also stored into the FIFO. The size of the receive FIFO is configured in the receive FIFO size register (OTG_GRXFSIZ).

The single receive FIFO architecture makes it highly efficient for the USB host to fill in the receive data buffer:

- All IN configured host channels share the same RAM buffer (shared FIFO)
- The OTG_FS core can fill in the receive FIFO up to the limit for any sequence of IN tokens driven by the host software

The application receives the Rx FIFO not-empty interrupt as long as there is at least one packet available for download. It reads the packet information from the receive status read and pop register and finally pops the data off the receive FIFO.

Host Tx FIFOs

The host uses one transmit FIFO for all non-periodic (control and bulk) OUT transactions and one transmit FIFO for all periodic (isochronous and interrupt) OUT transactions. FIFOs are used as transmit buffers to hold the data (payload of the transmit packet) to be transmitted over the USB. The size of the periodic (nonperiodic) Tx FIFO is configured in the host periodic (nonperiodic) transmit FIFO size OTG_HPTXFSIZ / OTG_HNPTXFSIZ) register.

The two Tx FIFO implementation derives from the higher priority granted to the periodic type of traffic over the USB frame. At the beginning of each frame, the built-in host scheduler processes the periodic request queue first, followed by the nonperiodic request queue.

The two transmit FIFO architecture provides the USB host with separate optimization for periodic and nonperiodic transmit data buffer management:

- All host channels configured to support periodic (nonperiodic) transactions in the OUT direction share the same RAM buffer (shared FIFOs)
- The OTG_FS core can fill in the periodic (nonperiodic) transmit FIFO up to the limit for any sequence of OUT tokens driven by the host software

The OTG_FS core issues the periodic Tx FIFO empty interrupt (PTXFE bit in OTG_GINTSTS) as long as the periodic Tx FIFO is half or completely empty, depending on the value of the periodic Tx FIFO empty level bit in the AHB configuration register (PTXFELVL bit in OTG_GAHBCFG). The application can push the transmission data in advance as long as free space is available in both the periodic Tx FIFO and the periodic request queue. The host periodic transmit FIFO and queue status register (OTG_HPTXSTS) can be read to know how much space is available in both.

OTG_FS core issues the non periodic Tx FIFO empty interrupt (NPTXFE bit in OTG_GINTSTS) as long as the nonperiodic Tx FIFO is half or completely empty depending on the non periodic Tx FIFO empty level bit in the AHB configuration register (TXFELVL bit in OTG_GAHBCFG). The application can push the transmission data as long as free space is available in both the nonperiodic Tx FIFO and nonperiodic request queue. The host nonperiodic transmit FIFO and queue status register (OTG_HNPTXSTS) can be read to know how much space is available in both.

63.11.3 FIFO RAM allocation

Device mode

Receive FIFO RAM allocation: the application should allocate RAM for SETUP packets:

- 10 locations must be reserved in the receive FIFO to receive SETUP packets on control endpoint. The core does not use these locations, which are reserved for SETUP packets, to write any other data.
- One location is to be allocated for Global OUT NAK.
- Status information is written to the FIFO along with each received packet. Therefore, a minimum space of (largest packet size / 4) + 1 must be allocated to receive packets. If multiple isochronous endpoints are enabled, then at least two (largest packet size / 4) + 1 spaces must be allocated to receive back-to-back packets. Typically, two (largest packet size / 4) + 1 spaces are recommended so that when the previous packet is being transferred to the CPU, the USB can receive the subsequent packet.
- Along with the last packet for each endpoint, transfer complete status information is also pushed to the FIFO. One location for each OUT endpoint is recommended.

Device RxFIFO =

(5 * number of control endpoints + 8) + ((largest USB packet used / 4) + 1 for status information) + (2 * number of OUT endpoints) + 1 for Global NAK

Example: The MPS is 1,024 bytes for a periodic USB packet and 512 bytes for a non-periodic USB packet. There are three OUT endpoints, three IN endpoints, one control endpoint, and three host channels.

Device RxFIFO = (5 * 1 + 8) + ((1,024 / 4) +1) + (2 * 4) + 1 = 279

Transmit FIFO RAM allocation: the minimum RAM space required for each IN endpoint Transmit FIFO is the maximum packet size for that particular IN endpoint.

Note: More space allocated in the transmit IN endpoint FIFO results in better performance on the USB.

Host mode

Receive FIFO RAM allocation:

Status information is written to the FIFO along with each received packet. Therefore, a minimum space of (largest packet size / 4) + 1 must be allocated to receive packets. If multiple isochronous channels are enabled, then at least two (largest packet size / 4) + 1 spaces must be allocated to receive back-to-back packets. Typically, two (largest packet size / 4) + 1 spaces are recommended so that when the previous packet is being transferred to the CPU, the USB can receive the subsequent packet.

Along with the last packet in the host channel, transfer complete status information is also pushed to the FIFO. So one location must be allocated for this.

Host RxFIFO = (largest USB packet used / 4) + 1 for status information + 1 transfer complete

Example: Host RxFIFO = ((1,024 / 4) + 1) + 1 = 258

Transmit FIFO RAM allocation:

The minimum amount of RAM required for the host Non-periodic Transmit FIFO is the largest maximum packet size among all supported non-periodic OUT channels.

Typically, two largest packet sizes worth of space is recommended, so that when the current packet is under transfer to the USB, the CPU can get the next packet.

Non-Periodic TxFIFO = largest non-periodic USB packet used / 4

Example: Non-Periodic TxFIFO = (512 / 4) = 128

The minimum amount of RAM required for host periodic Transmit FIFO is the largest maximum packet size out of all the supported periodic OUT channels. If there is at least one isochronous OUT endpoint, then the space must be at least two times the maximum packet size of that channel.

Host Periodic TxFIFO = largest periodic USB packet used / 4

Example: Host Periodic TxFIFO = (1,024 / 4) = 256

Note: More space allocated in the Transmit Non-periodic FIFO results in better performance on the USB.

63.12 OTG_FS system performance

Best USB and system performance is achieved owing to the large RAM buffers, the highly configurable FIFO sizes, the quick 32-bit FIFO access through AHB push/pop registers and, especially, the advanced FIFO control mechanism. Indeed, this mechanism allows the OTG_FS to fill in the available RAM space at best regardless of the current USB sequence. With these features:

- The application gains good margins to calibrate its intervention in order to optimize the CPU bandwidth usage:
  - It can accumulate large amounts of transmission data in advance compared to when they are effectively sent over the USB
  - It benefits of a large time margin to download data from the single receive FIFO
- The USB core is able to maintain its full operating rate, that is to provide maximum full-speed bandwidth with a great margin of autonomy versus application intervention:
  - It has a large reserve of transmission data at its disposal to autonomously manage the sending of data over the USB
  - It has a lot of empty space available in the receive buffer to autonomously fill it in with the data coming from the USB

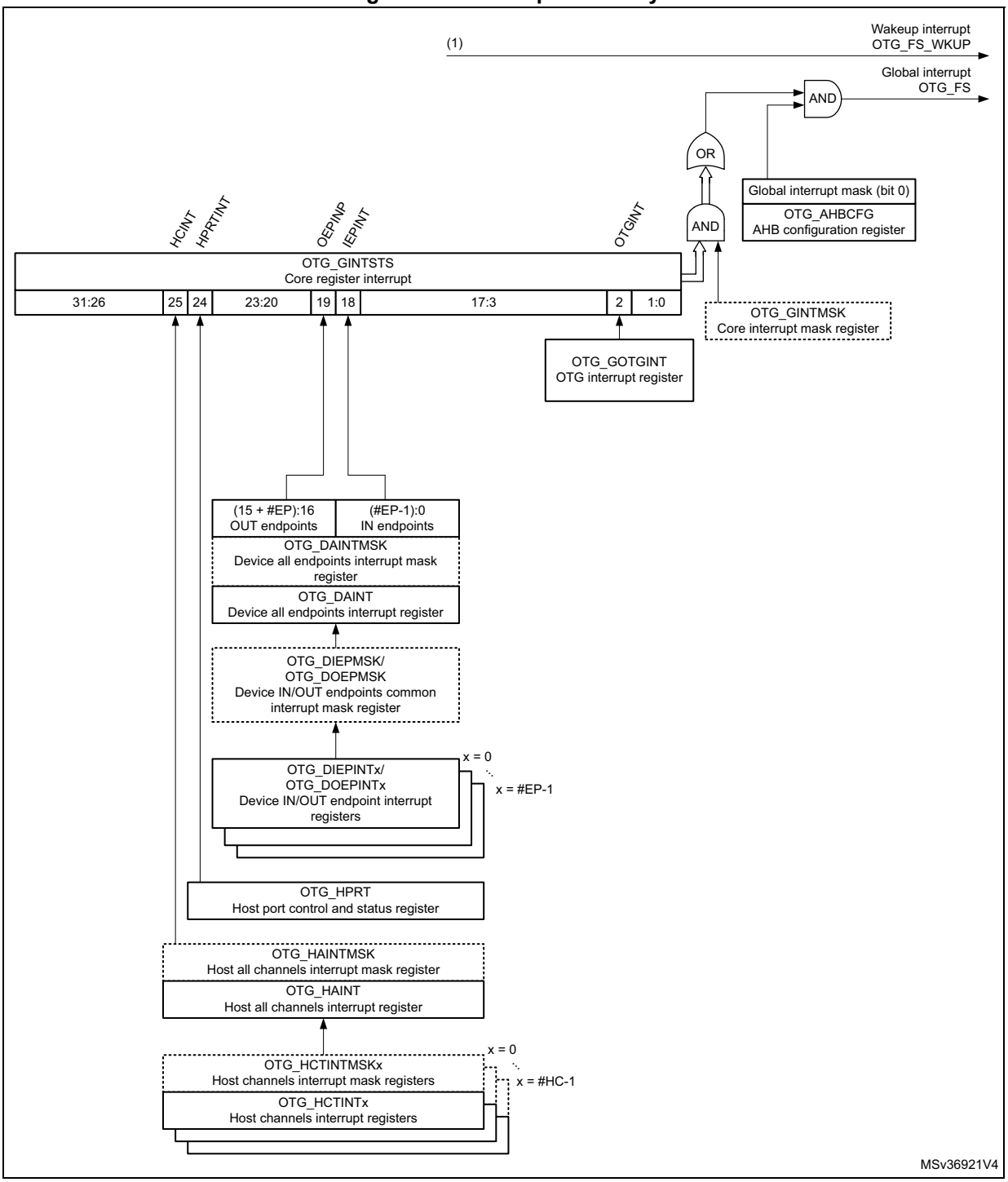
As the OTG_FS core is able to fill in the 1.25-Kbyte RAM buffer very efficiently, and as 1.25-Kbyte of transmit/receive data is more than enough to cover a full speed frame, the USB system is able to withstand the maximum full-speed data rate for up to one USB frame (1 ms) without any CPU intervention.

63.13 OTG_FS interrupts

When the OTG_FS controller is operating in one mode, either device or host, the application must not access registers from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and reflected in the core interrupt register (MMIS bit in the OTG_GINTSTS register). When the core switches from one mode to the other, the registers in the new mode of operation must be reprogrammed as they would be after a power-on reset.

Figure 780 shows the interrupt hierarchy.

Figure 780. Interrupt hierarchy



1. OTG_FS_WKUP becomes active (high state) when resume condition occurs during L1 SLEEP or L2 SUSPEND states.

63.14 OTG_FS control and status registers

By reading from and writing to the control and status registers (CSRs) through the AHB slave interface, the application controls the OTG_FS controller. These registers are 32 bits wide, and the addresses are 32-bit block aligned. The OTG_FS registers must be accessed by words (32 bits).

CSRs are classified as follows:

- Core global registers
- Host-mode registers
- Host global registers
- Host port CSRs
- Host channel-specific registers
- Device-mode registers
- Device global registers
- Device endpoint-specific registers
- Power and clock-gating registers
- Data FIFO (DFIFO) access registers

Only the core global, power and clock-gating, data FIFO access, and host port control and status registers can be accessed in both host and device modes. When the OTG_FS controller is operating in one mode, either device or host, the application must not access registers from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and reflected in the core interrupt register (MMIS bit in the OTG_GINTSTS register). When the core switches from one mode to the other, the registers in the new mode of operation must be reprogrammed as they would be after a power-on reset.

63.14.1 CSR memory map

The host and device mode registers occupy different addresses. All registers are implemented in the AHB clock domain.

Global CSR map

These registers are available in both host and device modes.

Table 645. Core global control and status registers (CSRs)

Acronym	Address offset	Register name
OTG_GOTGCTL	0x000	*Section 63.15.1: OTG control and status register (OTG_GOTGCTL)*
OTG_GOTGINT	0x004	*Section 63.15.2: OTG interrupt register (OTG_GOTGINT)*
OTG_GAHBCFG	0x008	*Section 63.15.3: OTG AHB configuration register (OTG_GAHBCFG)*
OTG_GUSBCFG	0x00C	*Section 63.15.4: OTG USB configuration register (OTG_GUSBCFG)*
OTG_GRSTCTL	0x010	*Section 63.15.5: OTG reset register (OTG_GRSTCTL)*
OTG_GINTSTS	0x014	*Section 63.15.6: OTG core interrupt register (OTG_GINTSTS)*
OTG_GINTMSK	0x018	*Section 63.15.7: OTG interrupt mask register (OTG_GINTMSK)*

Table 645. Core global control and status registers (CSRs) (continued)

Acronym	Address offset	Register name
OTG_GRXSTSR	0x01C	*Section 63.15.8: OTG receive status debug read register (OTG_GRXSTSR)*
		Section 63.15.9: OTG receive status debug read [alternate] (OTG_GRXSTSR)
OTG_GRXSTSP	0x020	*Section 63.15.10: OTG status read and pop registers (OTG_GRXSTSP)*
		Section 63.15.11: OTG status read and pop registers [alternate] (OTG_GRXSTSP)
OTG_GRXFSIZ	0x024	*Section 63.15.12: OTG receive FIFO size register (OTG_GRXFSIZ)*
OTG_HNPTXFSIZ/ OTG_DIEPTXF0(1)	0x028	*Section 63.15.13: OTG host non-periodic transmit FIFO size register (OTG_HNPTXFSIZ)/Endpoint 0 Transmit FIFO size (OTG_DIEPTXF0)*
OTG_HNPTXSTS	0x02C	*Section 63.15.14: OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS)*
OTG_GCCFG	0x038	*Section 63.15.15: OTG general core configuration register (OTG_GCCFG)*
OTG_CID	0x03C	*Section 63.15.16: OTG core ID register (OTG_CID)*
OTG_GLPMCFG	0x54	*Section 63.15.17: OTG core LPM configuration register (OTG_GLPMCFG)*
OTG_HPTXFSIZ	0x100	*Section 63.15.18: OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ)*
OTG_DIEPTXFx	0x104 0x108 ... 0x114	*Section 63.15.19: OTG device IN endpoint transmit FIFO x size register (OTG_DIEPTXFx)*

1. The general rule is to use OTG_HNPTXFSIZ for host mode and OTG_DIEPTXF0 for device mode.

Host-mode CSR map

These registers must be programmed every time the core changes to host mode.

Table 646. Host-mode control and status registers (CSRs)

Acronym	Offset address	Register name
OTG_HCFG	0x400	*Section 63.15.21: OTG host configuration register (OTG_HCFG)*
OTG_HFIR	0x404	*Section 63.15.22: OTG host frame interval register (OTG_HFIR)*
OTG_HFNUM	0x408	*Section 63.15.23: OTG host frame number/frame time remaining register (OTG_HFNUM)*
OTG_HPTXSTS	0x410	*Section 63.15.24: OTG_Host periodic transmit FIFO/queue status register (OTG_HPTXSTS)*
OTG_HAINT	0x414	*Section 63.15.25: OTG host all channels interrupt register (OTG_HAINT)*
OTG_HAINTMSK	0x418	*Section 63.15.26: OTG host all channels interrupt mask register (OTG_HAINTMSK)*

Table 646. Host-mode control and status registers (CSRs) (continued)

Acronym	Offset address	Register name
OTG_HPRT	0x440	*Section 63.15.27: OTG host port control and status register (OTG_HPRT)*
OTG_HCCHARx	0x500 0x520 ... 0x660	*Section 63.15.28: OTG host channel x characteristics register (OTG_HCCHARx)*
OTG_HCINTx	0x508 0x528 .... 0x668	*Section 63.15.29: OTG host channel x interrupt register (OTG_HCINTx)*
OTG_HCINTMSKx	0x50C 0x52C 0x66C	*Section 63.15.30: OTG host channel x interrupt mask register (OTG_HCINTMSKx)*
OTG_HCTSIZx	0x510 0x530 .... 0x670	*Section 63.15.31: OTG host channel x transfer size register (OTG_HCTSIZx)*

Device-mode CSR map

These registers must be programmed every time the core changes to device mode.

Table 647. Device-mode control and status registers

Acronym	Offset address	Register name
OTG_DCFG	0x800	*Section 63.15.33: OTG device configuration register (OTG_DCFG)*
OTG_DCTL	0x804	*Section 63.15.34: OTG device control register (OTG_DCTL)*
OTG_DSTS	0x808	*Section 63.15.35: OTG device status register (OTG_DSTS)*
OTG_DIEPMSK	0x810	*Section 63.15.36: OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK)*
OTG_DOEPMSK	0x814	*Section 63.15.37: OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK)*
OTG_DAINT	0x818	*Section 63.15.38: OTG device all endpoints interrupt register (OTG_DAINT)*
OTG_DAINTMSK	0x81C	*Section 63.15.39: OTG all endpoints interrupt mask register (OTG_DAINTMSK)*
OTG_DVBUSDIS	0x828	*Section 63.15.40: OTG device V_{BUS} discharge time register (OTG_DVBUSDIS)*
OTG_DVBUSPULSE	0x82C	*Section 63.15.41: OTG device V_{BUS} pulsing time register (OTG_DVBUSPULSE)*

Table 647. Device-mode control and status registers (continued)

Acronym	Offset address	Register name
OTG_DIEPEMPMSK	0x834	*Section 63.15.42: OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK)*
OTG_DIEPCTL0	0x900	*Section 63.15.43: OTG device control IN endpoint 0 control register (OTG_DIEPCTL0)*
OTG_DIEPCTLx	0x920 0x940 ... 0x9A0	*Section 63.15.44: OTG device IN endpoint x control register (OTG_DIEPCTLx)*
OTG_DIEPINTx	0x908 0x928 .... 0x988	*Section 63.15.45: OTG device IN endpoint x interrupt register (OTG_DIEPINTx)*
OTG_DIEPTSIZ0	0x910	*Section 63.15.46: OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0)*
OTG_DTXFSTSx	0x918 0x938 .... 0x998	*Section 63.15.47: OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx)*
OTG_DIEPTSIZx	0x930 0x950 ... 0x9B0	*Section 63.15.48: OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx)*
OTG_DOEPCTL0	0xB00	*Section 63.15.49: OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0)*
OTG_DOEPINTx	0xB08 0xB28 ... 0xBA8	*Section 63.15.50: OTG device OUT endpoint x interrupt register (OTG_DOEPINTx)*
OTG_DOEPTSIZ0	0xB10	*Section 63.15.51: OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0)*
OTG_DOEPCTLx	0xB20 0xB40 ... 0xBA0	*Section 63.15.52: OTG device OUT endpoint x control register (OTG_DOEPCTLx)*
OTG_DOEPTSIZx	0xB30 0xB50 ... 0xBB0	*Section 63.15.53: OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx)*

Data FIFO (DFIFO) access register map

These registers, available in both host and device modes, are used to read or write the FIFO space for a specific endpoint or a channel, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel.

Table 648. Data FIFO (DFIFO) access register map

FIFO access register section	Offset address	Access
Device IN endpoint 0/Host OUT Channel 0: DFIFO write access Device OUT endpoint 0/Host IN Channel 0: DFIFO read access	0x1000–0x1FFC	w r
Device IN endpoint 1/Host OUT Channel 1: DFIFO write access Device OUT endpoint 1/Host IN Channel 1: DFIFO read access	0x2000–0x2FFC	w r
...	...	...
Device IN endpoint x(1)/Host OUT Channel x(1): DFIFO write access Device OUT endpoint x(1)/Host IN Channel x(1): DFIFO read access	0xX000–0xXFFC	w r

1. Where x is 5in device mode and 11 in host mode.

Power and clock gating CSR map

There is a single register for power and clock gating. It is available in both host and device modes.

Table 649. Power and clock gating control and status registers

Acronym	Offset address	Register name
OTG_PCGCCTL	0xE00–0xE04	*Section 63.15.54: OTG power and clock gating control register (OTG_PCGCCTL)*

63.15 OTG_FS registers

These registers are available in both host and device modes, and do not need to be reprogrammed when switching between these modes.

Bit values in the register descriptions are expressed in binary unless otherwise specified.

63.15.1 OTG control and status register (OTG_GOTGCTL)

Address offset: 0x000

Reset value: 0x0001 0000

The OTG_GOTGCTL register controls the behavior and reflects the status of the OTG function of the core.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CUR MOD	OTG VER	BSVLD	ASVLD	DBCT	CID STS
										r	rw	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	EHEN	DHNP EN	HSHNP EN	HNP RQ	HNG SCS	BVALO VAL	BVALO EN	AVALO VAL	AVALO EN	VBVAL OVAL	VBVAL OEN	SRQ	SRQ SCS
			rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CURMOD:** Current mode of operation

Indicates the current mode (host or device).

0: Device mode

1: Host mode

Bit 20 **OTGVER:** OTG version

Selects the OTG revision.

0:OTG Version 1.3. OTG1.3 is obsolete for new product development.

1:OTG Version 2.0. In this version the core supports only data line pulsing for SRP.

Bit 19 **BSVLD:** B-session valid

Indicates the device mode transceiver status.

0: B-session is not valid.

1: B-session is valid.

In OTG mode, the user can use this bit to determine if the device is connected or disconnected.

Note: Only accessible in device mode.

Bit 18 **ASVLD:** A-session valid

Indicates the host mode transceiver status.

0: A-session is not valid

1: A-session is valid

Note: Only accessible in host mode.

Bit 17 **DBCT:** Long/short debounce time

Indicates the debounce time of a detected connection.

0: Long debounce time, used for physical connections (100 ms + 2.5 µs)

1: Short debounce time, used for soft connections (2.5 µs)

Note: Only accessible in host mode.

Bit 16 **CIDSTS:** Connector ID status

Indicates the connector ID status on a connect event.

0: The OTG_FS controller is in A-device mode

1: The OTG_FS controller is in B-device mode

Note: Accessible in both device and host modes.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **EHEN:** Embedded host enable

It is used to select between OTG A device state machine and embedded host state machine.

0: OTG A device state machine is selected

1: Embedded host state machine is selected

Bit 11 **DHNPEN:** Device HNP enabled

The application sets this bit when it successfully receives a SetFeature.SetHNPEnable command from the connected USB host.

0: HNP is not enabled in the application

1: HNP is enabled in the application

Note: Only accessible in device mode.

Bit 10 **HSHNPEN:** host set HNP enable

The application sets this bit when it has successfully enabled HNP (using the SetFeature.SetHNPEnable command) on the connected device.

0: Host Set HNP is not enabled

1: Host Set HNP is enabled

Note: Only accessible in host mode.

Bit 9 **HNPRQ:** HNP request

The application sets this bit to initiate an HNP request to the connected USB host. The application can clear this bit by writing a 0 when the host negotiation success status change bit in the OTG_GOTGINT register (HNSSCHG bit in OTG_GOTGINT) is set. The core clears this bit when the HNSSCHG bit is cleared.

0: No HNP request

1: HNP request

Note: Only accessible in device mode.

Bit 8 **HNGSCS:** Host negotiation success

The core sets this bit when host negotiation is successful. The core clears this bit when the HNP request (HNPRQ) bit in this register is set.

0: Host negotiation failure

1: Host negotiation success

Note: Only accessible in device mode.

Bit 7 **BVALOVAL:** B-peripheral session valid override value.

This bit is used to set override value for Bvalid signal when BVALOEN bit is set.

0: Bvalid value is '0' when BVALOEN = 1

1: Bvalid value is '1' when BVALOEN = 1

Note: Only accessible in device mode.

Bit 6 **BVALOEN:** B-peripheral session valid override enable.

This bit is used to enable/disable the software to override the Bvalid signal using the BVALOVAL bit.

0:Override is disabled and Bvalid signal from the respective PHY selected is used internally by the core

1:Internally Bvalid received from the PHY is overridden with BVALOVAL bit value

Note: Only accessible in device mode.

Bit 5 **AVALOVAL:** A-peripheral session valid override value.

This bit is used to set override value for Avalid signal when AVALOEN bit is set.

0: Avalid value is '0' when AVALOEN = 1

1: Avalid value is '1' when AVALOEN = 1

Note: Only accessible in host mode.

Bit 4 **AVALOEN:** A-peripheral session valid override enable.

This bit is used to enable/disable the software to override the Avalid signal using the AVALOVAL bit.

0:Override is disabled and Avalid signal from the respective PHY selected is used internally by the core

1:Internally Avalid received from the PHY is overridden with AVALOVAL bit value

Note: Only accessible in host mode.

Bit 3 **VBVALOVAL:** V_{BUS} valid override value.

This bit is used to set override value for vbusvalid signal when VBVALOEN bit is set.

0: vbusvalid value is '0' when VBVALOEN = 1

1: vbusvalid value is '1' when VBVALOEN = 1

Note: Only accessible in host mode.

Bit 2 **VBVALOEN:** V_{BUS} valid override enable.

This bit is used to enable/disable the software to override the vbusvalid signal using the VBVALOVAL bit.

0: Override is disabled and vbusvalid signal from the respective PHY selected is used internally by the core

1: Internally vbusvalid received from the PHY is overridden with VBVALOVAL bit value

Note: Only accessible in host mode.

Bit 1 **SRQ:** Session request

The application sets this bit to initiate a session request on the USB. The application can clear this bit by writing a 0 when the host negotiation success status change bit in the OTG_GOTGINT register (HNSSCHG bit in OTG_GOTGINT) is set. The core clears this bit when the HNSSCHG bit is cleared.

If the user uses the USB 1.1 full-speed serial transceiver interface to initiate the session request, the application must wait until V_{BUS} discharges to 0.2 V, after the B-session valid bit in this register (BSVLD bit in OTG_GOTGCTL) is cleared.

0: No session request

1: Session request

Note: Only accessible in device mode.

Bit 0 **SRQSCS:** Session request success

The core sets this bit when a session request initiation is successful.

0: Session request failure

1: Session request success

Note: Only accessible in device mode.

63.15.2 OTG interrupt register (OTG_GOTGINT)

Address offset: 0x04

Reset value: 0x0000 0000

The application reads this register whenever there is an OTG interrupt and clears the bits in this register to clear the OTG interrupt.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBC DNE	ADTO CHG	HNG DET	Res.
												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.	HNSS CHG	SRSS CHG	Res.	Res.	Res.	Res.	Res.	SEDET	Res.	Res.
						rc_w1	rc_w1						rc_w1		

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

Bit 19 **DBCDNE:** Debounce done

The core sets this bit when the debounce is completed after the device connect. The application can start driving USB reset after seeing this interrupt. This bit is only valid when the HNP Capable or SRP Capable bit is set in the OTG_GUSBCFG register (HNPCAP bit or SRPCAP bit in OTG_GUSBCFG, respectively).

Note: Only accessible in host mode.

Bit 18 **ADTOCHG:** A-device timeout change

The core sets this bit to indicate that the A-device has timed out while waiting for the B-device to connect.

Note: Accessible in both device and host modes.

Bit 17 **HNGDET:** Host negotiation detected

The core sets this bit when it detects a host negotiation request on the USB.

Note: Accessible in both device and host modes.

Bits 16:10 Reserved, must be kept at reset value.

Bit 9 **HNSSCHG:** Host negotiation success status change

The core sets this bit on the success or failure of a USB host negotiation request. The application must read the host negotiation success bit of the OTG_GOTGCTL register (HNGSCS bit in OTG_GOTGCTL) to check for success or failure.

Note: Accessible in both device and host modes.

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

Bit 8 **SRSSCHG:** Session request success status change

The core sets this bit on the success or failure of a session request. The application must read the session request success bit in the OTG_GOTGCTL register (SRQSCS bit in OTG_GOTGCTL) to check for success or failure.

Note: Accessible in both device and host modes.

Bit 2 **SEDET:** Session end detected

The core sets this bit to indicate that the level of the voltage on V_{BUS} is no longer valid for a B-Peripheral session when $V_{BUS} < 0.8$ V.

Note: Accessible in both device and host modes.

Bits 1:0 Reserved, must be kept at reset value.

63.15.3 OTG AHB configuration register (OTG_GAHBCFG)

Address offset: 0x008

Reset value: 0x0000 0000

This register can be used to configure the core after power-on or a change in mode. This register mainly contains AHB system-related configuration parameters. Do not change this register after the initial programming. The application must program this register before starting any transactions on either the AHB or the USB.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTXFE LVL	TXFE LVL	Res.	Res.	Res.	Res.	Res.	Res.	GINT MSK
							rw	rw							rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **PTXFELVL:** Periodic Tx FIFO empty level

Indicates when the periodic Tx FIFO empty interrupt bit in the OTG_GINTSTS register (PTXFE bit in OTG_GINTSTS) is triggered.

0: PTXFE (in OTG_GINTSTS) interrupt indicates that the Periodic Tx FIFO is half empty

1: PTXFE (in OTG_GINTSTS) interrupt indicates that the Periodic Tx FIFO is completely empty

Note: Only accessible in host mode.

Bit 7 **TXFELVL:** Tx FIFO empty level

In device mode, this bit indicates when IN endpoint Transmit FIFO empty interrupt (TXFE in OTG_DIEPINTx) is triggered:

0:The TXFE (in OTG_DIEPINTx) interrupt indicates that the IN endpoint Tx FIFO is half empty

1:The TXFE (in OTG_DIEPINTx) interrupt indicates that the IN endpoint Tx FIFO is completely empty

In host mode, this bit indicates when the nonperiodic Tx FIFO empty interrupt (NPTXFE bit in OTG_GINTSTS) is triggered:

0:The NPTXFE (in OTG_GINTSTS) interrupt indicates that the nonperiodic Tx FIFO is half empty

1:The NPTXFE (in OTG_GINTSTS) interrupt indicates that the nonperiodic Tx FIFO is completely empty

Bits 6:1 Reserved, must be kept at reset value.

Bit 0 **GINTMSK:** Global interrupt mask

The application uses this bit to mask or unmask the interrupt line assertion to itself. Irrespective of this bit's setting, the interrupt status registers are updated by the core.

0: Mask the interrupt assertion to the application.

1: Unmask the interrupt assertion to the application.

Note: Accessible in both device and host modes.

63.15.4 OTG USB configuration register (OTG_GUSBCFG)

Address offset: 0x00C

Reset value: 0x0000 1440

This register can be used to configure the core after power-on or a changing to host mode or device mode. It contains USB and USB-PHY related configuration parameters. The application must program this register before starting any transactions on either the AHB or the USB. Do not make changes to this register after the initial programming.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	FD MOD	FH MOD	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.	TRDT[3:0]				HNP CAP	SRP CAP	Res.	PHY SEL	Res.	Res.	Res.	TOCAL[2:0]		
		rw	rw	rw	rw	rw	rw		r				rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **FDMOD:** Force device mode

Writing a 1 to this bit, forces the core to device mode irrespective of the OTG_ID input pin.
0: Normal mode
1: Force device mode
After setting the force bit, the application must wait at least 25 ms before the change takes effect.

Note: Accessible in both device and host modes.

Bit 29 **FHMOD:** Force host mode

Writing a 1 to this bit, forces the core to host mode irrespective of the OTG_ID input pin.
0: Normal mode
1: Force host mode
After setting the force bit, the application must wait at least 25 ms before the change takes effect.

Note: Accessible in both device and host modes.

Bits 28:26 Reserved, must be kept at reset value.

Bits 25:23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bits 21:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bits 13:10 **TRDT[3:0]:** USB turnaround time

These bits are used to set the turnaround time in PHY clocks. They must be configured according to *Table 650: TRDT values*, depending on the application AHB frequency. Higher TRDT values allow stretching the USB response time to IN tokens in order to compensate for longer AHB read access latency to the data FIFO.

Note: Only accessible in device mode.

Bit 9 **HNPCAP:** HNP-capable

The application uses this bit to control the OTG_FS controller's HNP capabilities.

0: HNP capability is not enabled.

1: HNP capability is enabled.

Note: Accessible in both device and host modes.

Bit 8 **SRPCAP:** SRP-capable

The application uses this bit to control the OTG_FS controller's SRP capabilities. If the core operates as a non-SRP-capable

B-device, it cannot request the connected A-device (host) to activate V_{BUS} and start a session.

0: SRP capability is not enabled.

1: SRP capability is enabled.

Note: Accessible in both device and host modes.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **PHYSEL:** Full Speed serial transceiver mode select

This bit is always 1 with read-only access.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **TOCAL[2:0]:** FS timeout calibration

The number of PHY clocks that the application programs in this field is added to the full-speed interpacket timeout duration in the core to account for any additional delays introduced by the PHY. This can be required, because the delay introduced by the PHY in generating the line state condition can vary from one PHY to another.

The USB standard timeout value for full-speed operation is 16 to 18 (inclusive) bit times. The application must program this field based on the speed of enumeration. The number of bit times added per PHY clock is 0.25 bit times.

Table 650. TRDT values

AHB frequency range (MHz)		TRDT minimum value
Min	Max	
14.2	15	0xF
15	16	0xE
16	17.2	0xD
17.2	18.5	0xC
18.5	20	0xB
20	21.8	0xA
21.8	24	0x9
24	27.5	0x8
27.5	32	0x7
32	-	0x6

63.15.5 OTG reset register (OTG_GRSTCTL)

Address offset: 0x10

Reset value: 0x8000 0000

The application uses this register to reset various hardware features inside the core.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AHB IDL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	TXFNUM[4:0]					TXF FLSH	RXF FLSH	Res.	FCRST	PSRST	CSRST
					rw	rw	rw	rw	rw	rs	rs		rs	rs	r

Bit 31 **AHBIDL:** AHB master idle

Indicates that the AHB master state machine is in the Idle condition.

Note: Accessible in both device and host modes.

Bits 30:11 Reserved, must be kept at reset value.

Bits 10:6 **TXFNUM[4:0]:** Tx FIFO number

This is the FIFO number that must be flushed using the Tx FIFO Flush bit. This field must not be changed until the core clears the Tx FIFO Flush bit.

00000:

- Non-periodic Tx FIFO flush in host mode
- Tx FIFO 0 flush in device mode

00001:

- Periodic Tx FIFO flush in host mode
- Tx FIFO 1 flush in device mode

00010: Tx FIFO 2 flush in device mode

...

01111: Tx FIFO 15 flush in device mode

10000: Flush all the transmit FIFOs in device or host mode.

Note: Accessible in both device and host modes.

Bit 5 **TXFFLSH:** Tx FIFO flush

This bit selectively flushes a single or all transmit FIFOs, but cannot do so if the core is in the midst of a transaction.

The application must write this bit only after checking that the core is neither writing to the Tx FIFO nor reading from the Tx FIFO. Verify using these registers:

Read—NAK Effective interrupt ensures the core is not reading from the FIFO

Write—AHBIDL bit in OTG_GRSTCTL ensures the core is not writing anything to the FIFO.

Flushing is normally recommended when FIFOs are reconfigured. FIFO flushing is also recommended during device endpoint disable. The application must wait until the core clears this bit before performing any operations. This bit takes eight clocks to clear, using the slower clock of phy_clk or hclk.

Note: Accessible in both device and host modes.

Bit 4 **RXFFLSH:** Rx FIFO flush

The application can flush the entire Rx FIFO using this bit, but must first ensure that the core is not in the middle of a transaction.

The application must only write to this bit after checking that the core is neither reading from the Rx FIFO nor writing to the Rx FIFO.

The application must wait until the bit is cleared before performing any other operations. This bit requires 8 clocks (slowest of PHY or AHB clock) to clear.

Note: Accessible in both device and host modes.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FCRST:** Host frame counter reset

The application writes this bit to reset the frame number counter inside the core. When the frame counter is reset, the subsequent SOF sent out by the core has a frame number of 0.

When application writes '1' to the bit, it might not be able to read back the value as it gets cleared by the core in a few clock cycles.

Note: Only accessible in host mode.

Bit 1 **PSRST:** Partial soft reset

Resets the internal state machines but keeps the enumeration info. Could be used to recover some specific PHY errors.

Note: Accessible in both device and host modes.

Bit 0 **CSRST:** Core soft reset

Resets the HCLK and PHY clock domains as follows:

Clears the interrupts and all the CSR register bits except for the following bits:

– GATEHCLK bit in OTG_PCGCCTL
– STPPCLK bit in OTG_PCGCCTL
– FSLSPCS bits in OTG_HCFG
– DSPD bit in OTG_DCFG
– SDIS bit in OTG_DCTL
– OTG_GCCFG register

All module state machines (except for the AHB slave unit) are reset to the Idle state, and all the transmit FIFOs and the receive FIFO are flushed.

Any transactions on the AHB Master are terminated as soon as possible, after completing the last data phase of an AHB transfer. Any transactions on the USB are terminated immediately.

The application can write to this bit any time it wants to reset the core. This is a self-clearing bit and the core clears this bit after all the necessary logic is reset in the core, which can take several clocks, depending on the current state of the core. Once this bit has been cleared, the software must wait at least 3 PHY clocks before accessing the PHY domain (synchronization delay). The software must also check that bit 31 in this register is set to 1 (AHB Master is Idle) before starting any operation.

Typically, the software reset is used during software development and also when the user dynamically changes the PHY selection bits in the above listed USB configuration registers. When the user changes the PHY, the corresponding clock for the PHY is selected and used in the PHY domain. Once a new clock is selected, the PHY domain has to be reset for proper operation.

Note: Accessible in both device and host modes.

63.15.6 OTG core interrupt register (OTG_GINTSTS)

Address offset: 0x014

Reset value: 0x0400 0020

This register interrupts the application for system-level events in the current mode (device mode or host mode).

Some of the bits in this register are valid only in host mode, while others are valid in device mode only. This register also indicates the current mode. To clear the interrupt status bits of the rc_w1 type, the application must write 1 into the bit.

The FIFO status interrupts are read-only; once software reads from or writes to the FIFO while servicing these interrupts, FIFO interrupt conditions are cleared automatically.

The application must clear the OTG_GINTSTS register at initialization before unmasking the interrupt bit to avoid any interrupts generated prior to initialization.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WKUP INT	SRQ INT	DISC INT	CIDS CHG	LPM INT	PTXFE	HCINT	HPRT INT	RST DET	Res.	IPXFR/ IN COMP ISO OUT	IISOI XFR	OEP INT	IEPINT	Res.	Res.
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	r	r	r	rc_w1		rc_w1	rc_w1	r	r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOPF	ISOO DRP	ENUM DNE	USB RST	USB SUSP	ESUSP	Res.	Res.	GO NAK EFF	GI NAK EFF	NPTXF E	RXF LVL	SOF	OTG INT	MMIS	CMOD
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1			r	r	r	r	rc_w1	r	rc_w1	r

Bit 31 **WKUPINT:** Resume/remote wakeup detected interrupt

Wakeup interrupt during suspend(L2) or LPM(L1) state.

– During suspend(L2):

In device mode, this interrupt is asserted when a resume is detected on the USB. In host mode, this interrupt is asserted when a remote wakeup is detected on the USB.

– During LPM(L1):

This interrupt is asserted for either host initiated resume or device initiated remote wakeup on USB.

Note: Accessible in both device and host modes.

Bit 30 **SRQINT:** Session request/new session detected interrupt

In host mode, this interrupt is asserted when a session request is detected from the device. In device mode, this interrupt is asserted when V_{BUS} is in the valid range for a B-peripheral device. Accessible in both device and host modes.

Bit 29 **DISCINT:** Disconnect detected interrupt

Asserted when a device disconnect is detected.

Note: Only accessible in host mode.

Bit 28 **CIDSCHG:** Connector ID status change

The core sets this bit when there is a change in connector ID status.

Note: Accessible in both device and host modes.

Bit 27 **LPMINT:** LPM interrupt

In device mode, this interrupt is asserted when the device receives an LPM transaction and responds with a non-ERRORed response.

In host mode, this interrupt is asserted when the device responds to an LPM transaction with a non-ERRORed response or when the host core has completed LPM transactions for the programmed number of times (RETRYCNT bit in OTG_GLPMCFG).

This field is valid only if the LPMEN bit in OTG_GLPMCFG is set to 1.

Bit 26 **PTXFE:** Periodic Tx FIFO empty

Asserted when the periodic transmit FIFO is either half or completely empty and there is space for at least one entry to be written in the periodic request queue. The half or completely empty status is determined by the periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (PTXFELVL bit in OTG_GAHBCFG).

Note: Only accessible in host mode.

Bit 25 **HCINT:** Host channels interrupt

The core sets this bit to indicate that an interrupt is pending on one of the channels of the core (in host mode). The application must read the OTG_HAINT register to determine the exact number of the channel on which the interrupt occurred, and then read the corresponding OTG_HCINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the OTG_HCINTx register to clear this bit.

Note: Only accessible in host mode.

Bit 24 **HPRTINT:** Host port interrupt

The core sets this bit to indicate a change in port status of one of the OTG_FS controller ports in host mode. The application must read the OTG_HPRT register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_HPRT register to clear this bit.

Note: Only accessible in host mode.

Bit 23 **RSTDET:** Reset detected interrupt

In device mode, this interrupt is asserted when a reset is detected on the USB in partial power-down mode when the device is in suspend.

Note: Only accessible in device mode.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **IPXFR:** Incomplete periodic transfer

In host mode, the core sets this interrupt bit when there are incomplete periodic transactions still pending, which are scheduled for the current frame.

INCOMPISOOUT: Incomplete isochronous OUT transfer

In device mode, the core sets this interrupt to indicate that there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

Bit 20 **IISOIXFR:** Incomplete isochronous IN transfer

The core sets this interrupt to indicate that there is at least one isochronous IN endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

Note: Only accessible in device mode.

Bit 19 **OEPINT:** OUT endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the OUT endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the OUT endpoint on which the interrupt occurred, and then read the corresponding OTG_DOEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DOEPINTx register to clear this bit.

Note: Only accessible in device mode.

Bit 18 **IEPINT:** IN endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the IN endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the IN endpoint on which the interrupt occurred, and then read the corresponding OTG_DIEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DIEPINTx register to clear this bit.

Note: Only accessible in device mode.

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **EOPF:** End of periodic frame interrupt

Indicates that the period specified in the periodic frame interval field of the OTG_DCFG register (PFIVL bit in OTG_DCFG) has been reached in the current frame.

Note: Only accessible in device mode.

Bit 14 **ISOODRP:** Isochronous OUT packet dropped interrupt

The core sets this bit when it fails to write an isochronous OUT packet into the Rx FIFO because the Rx FIFO does not have enough space to accommodate a maximum size packet for the isochronous OUT endpoint.

Note: Only accessible in device mode.

Bit 13 **ENUMDNE:** Enumeration done

The core sets this bit to indicate that speed enumeration is complete. The application must read the OTG_DSTS register to obtain the enumerated speed.

Note: Only accessible in device mode.

Bit 12 **USBRST:** USB reset

The core sets this bit to indicate that a reset is detected on the USB.

Note: Only accessible in device mode.

Bit 11 **USBSUSP:** USB suspend

The core sets this bit to indicate that a suspend was detected on the USB. The core enters the suspended state when there is no activity on the data lines for an extended period of time.

Note: Only accessible in device mode.

Bit 10 **ESUSP:** Early suspend

The core sets this bit to indicate that an Idle state has been detected on the USB for 3 ms.

Note: Only accessible in device mode.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GONAKEFF:** Global OUT NAK effective

Indicates that the Set global OUT NAK bit in the OTG_DCTL register (SGONAK bit in OTG_DCTL), set by the application, has taken effect in the core. This bit can be cleared by writing the Clear global OUT NAK bit in the OTG_DCTL register (CGONAK bit in OTG_DCTL).

Note: Only accessible in device mode.

Bit 6 **GINAKEFF:** Global IN non-periodic NAK effective

Indicates that the Set global non-periodic IN NAK bit in the OTG_DCTL register (SGINAK bit in OTG_DCTL), set by the application, has taken effect in the core. That is, the core has sampled the Global IN NAK bit set by the application. This bit can be cleared by clearing the Clear global non-periodic IN NAK bit in the OTG_DCTL register (CGINAK bit in OTG_DCTL).

This interrupt does not necessarily mean that a NAK handshake is sent out on the USB. The STALL bit takes precedence over the NAK bit.

Note: Only accessible in device mode.

Bit 5 **NPTXFE:** Non-periodic Tx FIFO empty

This interrupt is asserted when the non-periodic Tx FIFO is either half or completely empty, and there is space for at least one entry to be written to the non-periodic transmit request queue. The half or completely empty status is determined by the non-periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (TXFELVL bit in OTG_GAHBCFG).

Note: Accessible in host mode only.

Bit 4 **RXFLVL:** Rx FIFO non-empty

Indicates that there is at least one packet pending to be read from the Rx FIFO.

Note: Accessible in both host and device modes.

Bit 3 **SOF:** Start of frame

In host mode, the core sets this bit to indicate that an SOF (FS), or Keep-Alive (LS) is transmitted on the USB. The application must write a 1 to this bit to clear the interrupt.

In device mode, in the core sets this bit to indicate that an SOF token has been received on the USB. The application can read the OTG_DSTS register to get the current frame number. This interrupt is seen only when the core is operating in FS.

Note: This register may return '1' if read immediately after power on reset. If the register bit reads '1' immediately after power on reset it does not indicate that an SOF has been sent (in case of host mode) or SOF has been received (in case of device mode). The read value of this interrupt is valid only after a valid connection between host and device is established. If the bit is set after power on reset the application can clear the bit.

Note: Accessible in both host and device modes.

Bit 2 **OTGINT:** OTG interrupt

The core sets this bit to indicate an OTG protocol event. The application must read the OTG interrupt status (OTG_GOTGINT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_GOTGINT register to clear this bit.

Note: Accessible in both host and device modes.

Bit 1 **MMIS:** Mode mismatch interrupt

The core sets this bit when the application is trying to access:

– A host mode register, when the core is operating in device mode

– A device mode register, when the core is operating in host mode

The register access is completed on the AHB with an OKAY response, but is ignored by the core internally and does not affect the operation of the core.

Note: Accessible in both host and device modes.

Bit 0 **CMOD:** Current mode of operation

Indicates the current mode.

0: Device mode

1: Host mode

Note: Accessible in both host and device modes.

63.15.7 OTG interrupt mask register (OTG_GINTMSK)

Address offset: 0x018

Reset value: 0x0000 0000

This register works with the core interrupt register to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the core interrupt (OTG_GINTSTS) register bit corresponding to that interrupt is still set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WUIM	SRQIM	DISCINT	CIDSCHGM	LPMINTM	PTXFEM	HCIM	PRTIM	RSTDETM	Res.	IPXFRM/IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Res.	Res.	GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	

Bit 31 **WUIM:** Resume/remote wakeup detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both host and device modes.

Bit 30 **SRQIM:** Session request/new session detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both host and device modes.

Bit 29 **DISCINT:** Disconnect detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in host mode.

Bit 28 **CIDSCHGM:** Connector ID status change mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both host and device modes.

Bit 27 **LPMINTM:** LPM interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both host and device modes.

Bit 26 **PTXFEM:** Periodic Tx FIFO empty mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in host mode.

Bit 25 **HCIM:** Host channels interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in host mode.

Bit 24 **PRTIM:** Host port interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in host mode.

Bit 23 **RSTDETM:** Reset detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **IPXFRM:** Incomplete periodic transfer mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in host mode.
IISOOXFRM: Incomplete isochronous OUT transfer mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 20 **IISOIXFRM:** Incomplete isochronous IN transfer mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 19 **OEPINT:** OUT endpoints interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 18 **IEPINT:** IN endpoints interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **EOPFM:** End of periodic frame interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 14 **ISOODRPM:** Isochronous OUT packet dropped interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 13 **ENUMDNEM:** Enumeration done mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 12 **USBRST:** USB reset mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 11 **USBSUSPM:** USB suspend mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 10 **ESUSPM:** Early suspend mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GONAKEFFM:** Global OUT NAK effective mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 6 **GINAKEFFM:** Global non-periodic IN NAK effective mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 5 **NPTXFEM:** Non-periodic Tx FIFO empty mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in host mode.

Bit 4 **RXFLVLM:** Receive FIFO non-empty mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both device and host modes.

Bit 3 **SOFM:** Start of frame mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both device and host modes.

Bit 2 **OTGINT:** OTG interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both device and host modes.

Bit 1 **MMISM:** Mode mismatch interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both device and host modes.

Bit 0 Reserved, must be kept at reset value.

63.15.8 OTG receive status debug read register (OTG_GRXSTSR)

Address offset for read: 0x01C

Reset value: 0x0000 0000

This description is for register OTG_GRXSTSR in Device mode.

A read to the receive status debug read register returns the contents of the top of the receive FIFO.

The core ignores the receive status read when the receive FIFO is empty and returns a value of 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM[3:0]				PKTSTS[3:0]				DPID[1]
				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
DPID[0]	BCNT[10:0]											EPNUM[3:0]			
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 **STSPHST:** Status phase start

Indicates the start of the status phase for a control write transfer. This bit is set along with the OUT transfer completed PKTSTS pattern.

Bits 26:25 Reserved, must be kept at reset value.

Bits 24:21 **FRMNUM[3:0]:** Frame number

This is the least significant 4 bits of the frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.

Bits 20:17 **PKTSTS[3:0]:** Packet status

Indicates the status of the received packet
0001: Global OUT NAK (triggers an interrupt)
0010: OUT data packet received
0011: OUT transfer completed (triggers an interrupt)
0100: SETUP transaction completed (triggers an interrupt)
0110: SETUP data packet received
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID

Indicates the data PID of the received OUT data packet
00: DATA0
10: DATA1

Bits 14:4 **BCNT[10:0]:** Byte count

Indicates the byte count of the received data packet.

Bits 3:0 **EPNUM[3:0]:** Endpoint number

Indicates the endpoint number to which the current received packet belongs.

63.15.9 OTG receive status debug read [alternate] (OTG_GRXSTSR)

Address offset for read: 0x01C

Reset value: 0x0000 0000

This description is for register OTG_GRXSTSR in Host mode.

A read to the receive status debug read register returns the contents of the top of the receive FIFO.

The core ignores the receive status read when the receive FIFO is empty and returns a value of 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS[3:0]				DPID
											r	r	r	r	r

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

Bits 20:17 **PKTSTS[3:0]:** Packet status
Indicates the status of the received packet
0010: IN data packet received
0011: IN transfer completed (triggers an interrupt)
0101: Data toggle error (triggers an interrupt)
0111: Channel halted (triggers an interrupt)
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID
Indicates the data PID of the received packet
00: DATA0
10: DATA1

Bits 14:4 **BCNT[10:0]:** Byte count
Indicates the byte count of the received IN data packet.

Bits 3:0 **CHNUM[3:0]:** Channel number
Indicates the channel number to which the current received packet belongs.

63.15.10 OTG status read and pop registers (OTG_GRXSTSP)

Address offset for pop: 0x020

Reset value: 0x0000 0000

This description is for register OTG_GRXSTSP in Device mode.

Similarly to OTG_GRXSTSR (receive status debug read register) where a read returns the contents of the top of the receive FIFO, a read to OTG_GRXSTSP (receive status read and pop register) additionally pops the top data entry out of the Rx FIFO.

The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the receive status FIFO when the receive FIFO non-empty bit of the core interrupt register (RXFLVL bit in OTG_GINTSTS) is asserted.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM[3:0]				PKTSTS[3:0]				DPID[1]
				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
DPID[0]	BCNT[10:0]											EPNUM[3:0]			
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 **STSPHST:** Status phase start

Indicates the start of the status phase for a control write transfer. This bit is set along with the OUT transfer completed PKTSTS pattern.

Bits 26:25 Reserved, must be kept at reset value.

Bits 24:21 **FRMNUM[3:0]:** Frame number

This is the least significant 4 bits of the frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.

Bits 20:17 **PKTSTS[3:0]:** Packet status

Indicates the status of the received packet
0001: Global OUT NAK (triggers an interrupt)
0010: OUT data packet received
0011: OUT transfer completed (triggers an interrupt)
0100: SETUP transaction completed (triggers an interrupt)
0110: SETUP data packet received
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID

Indicates the data PID of the received OUT data packet
00: DATA0
10: DATA1

Bits 14:4 **BCNT[10:0]:** Byte count

Indicates the byte count of the received data packet.

Bits 3:0 **EPNUM[3:0]:** Endpoint number

Indicates the endpoint number to which the current received packet belongs.

63.15.11 OTG status read and pop registers [alternate] (OTG_GRXSTSP)

Address offset for pop: 0x020

Reset value: 0x0000 0000

This description is for register OTG_GRXSTSP in Host mode.

Similarly to OTG_GRXSTSR (receive status debug read register) where a read returns the contents of the top of the receive FIFO, a read to OTG_GRXSTSP (receive status read and pop register) additionally pops the top data entry out of the Rx FIFO.

The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the receive status FIFO when the receive FIFO non-empty bit of the core interrupt register (RXFLVL bit in OTG_GINTSTS) is asserted.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS[3:0]				DPID
											r	r	r	r	r

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

Bits 20:17 **PKTSTS[3:0]:** Packet status
Indicates the status of the received packet
0010: IN data packet received
0011: IN transfer completed (triggers an interrupt)
0101: Data toggle error (triggers an interrupt)
0111: Channel halted (triggers an interrupt)
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID
Indicates the data PID of the received packet
00: DATA0
10: DATA1

Bits 14:4 **BCNT[10:0]:** Byte count
Indicates the byte count of the received IN data packet.

Bits 3:0 **CHNUM[3:0]:** Channel number
Indicates the channel number to which the current received packet belongs.

63.15.12 OTG receive FIFO size register (OTG_GRXFSIZ)

Address offset: 0x024

Reset value: 0x0000 0200

The application can program the RAM size that must be allocated to the Rx 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
RXFD[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 **RXFD[15:0]:** Rx FIFO depth

This value is in terms of 32-bit words.

Minimum value is 16

Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

63.15.13 OTG host non-periodic transmit FIFO size register (OTG_HNPTXFSIZ)/Endpoint 0 Transmit FIFO size (OTG_DIEPTXF0)

Address offset: 0x028

Reset value: 0x0200 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NPTXFD/TX0FD[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
NPTXFSA/TX0FSA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Host mode

Bits 31:16 **NPTXFD[15:0]:** Non-periodic Tx FIFO depth

This value is in terms of 32-bit words.

Minimum value is 16

Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

Bits 15:0 **NPTXFSA[15:0]:** Non-periodic transmit RAM start address

This field configures the memory start address for non-periodic transmit FIFO RAM.

Device mode

Bits 31:16 **TX0FD:** Endpoint 0 Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

Bits 15:0 **TX0FSA:** Endpoint 0 transmit RAM start address
This field configures the memory start address for the endpoint 0 transmit FIFO RAM.

63.15.14 OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS)

Address offset: 0x02C

Reset value: 0x0008 0200

Note: In device mode, this register is not valid.

This read-only register contains the free space information for the non-periodic Tx FIFO and the non-periodic transmit request queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	NPTXQTOP[6:0]							NPTQXSAV[7:0]							
	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
NPTXFSAV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **NPTXQTOP[6:0]:** Top of the non-periodic transmit request queue

Entry in the non-periodic Tx request queue that is currently being processed by the MAC.
Bits 30:27: Channel/endpoint number
Bits 26:25:
00: IN/OUT token
01: Zero-length transmit packet (device IN/host OUT)
11: Channel halt command
Bit 24: Terminate (last entry for selected channel/endpoint)

Bits 23:16 **NPTQXSAV[7:0]:** Non-periodic transmit request queue space available

Indicates the amount of free space available in the non-periodic transmit request queue. This queue holds both IN and OUT requests.
0: Non-periodic transmit request queue is full
1: 1 location available
2: locations available
n: n locations available (0 ≤ n ≤ 8)
Others: Reserved

Bits 15:0 **NPTXFSAV[15:0]:** Non-periodic Tx FIFO space available

Indicates the amount of free space available in the non-periodic Tx FIFO.
Values are in terms of 32-bit words.
0: Non-periodic Tx FIFO is full
1: 1 word available
2: 2 words available
n: n words available (where 0 ≤ n ≤ 512)
Others: Reserved

63.15.15 OTG general core configuration register (OTG_GCCFG)

Address offset: 0x038

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.	VBDEN	SDEN	PDEN	DCD EN	BCDEN	PWR DWN
										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.	PS2 DET	SDET	PDET	DCDET
												r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **VBDEN:** USB V_{BUS} detection enable

Enables V_{BUS} sensing comparators to detect V_{BUS} valid levels on the V_{BUS} PAD for USB host and device operation. If HNP and/or SRP support is enabled, V_{BUS} comparators are automatically enabled independently of VBDEN value.

0 = V_{BUS} detection disabled

1 = V_{BUS} detection enabled

Bit 20 **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 19 **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 18 **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 17 **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.

Bit 16 **PWRDWN:** Power down control of FS PHY

Used to activate the FS PHY in transmission/reception. When reset, the PHY is kept in power-down. When set, the BCD function must be off (BCDEN=0).

0 = USB FS PHY disabled

1 = USB FS PHY enabled

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **PS2DET:** DM pull-up detection status

This bit is active only during PD and gives the result of comparison between DM voltage level and VLGC 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, CDP or DCP)

1: PS2 port or proprietary charger detected

Bit 2 **SDET:** Secondary detection (SD) status

This bit gives the result of SD.

0: CDP detected

1: DCP detected

Bit 1 **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 CDP or DCP).

Bit 0 **DCDET:** Data contact detection (DCD) status

This bit gives the result of DCD.

0: data lines contact not detected

1: data lines contact detected

63.15.16 OTG core ID register (OTG_CID)

Address offset: 0x03C

Reset value: 0x0000 3000

This is a register containing the Product ID as reset value.

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

Bits 31:0 **PRODUCT_ID[31:0]:** Product ID field
Application-programmable ID field.

63.15.17 OTG core LPM configuration register (OTG_GLPMCFG)

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.	EN BESL	LPMRCNTSTS[2:0]			SND LPM	LPMRCNT[2:0]			LPMCHIDX[3:0]				L1RSM OK
			rw	r	r	r	rs	rw	rw	rw	rw	rw	rw	rw	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SLP STS	LPMRSP[1:0]		L1DS EN	BESLTHRS[3:0]				L1SS EN	REM WAKE	BESL[3:0]				LPM ACK	LPM EN
r	r	r	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 **ENBESL:** Enable best effort service latency
This bit enables the BESL feature as defined in the LPM errata:
0:The core works as described in the following document:
USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification, July 16, 2007
1:The core works as described in the LPM Errata:
Errata for USB 2.0 ECN: Link Power Management (LPM) - 7/2007
Note: Only the updated behavior (described in LPM Errata) is considered in this document and so the ENBESL bit should be set to '1' by application SW.

Bits 27:25 **LPMRCNTSTS[2:0]:** LPM retry count status
Number of LPM host retries still remaining to be transmitted for the current LPM sequence.
Note: Accessible only in host mode.

Bit 24 **SNDLPM:** Send LPM transaction

When the application software sets this bit, an LPM transaction containing two tokens, EXT and LPM is sent. The hardware clears this bit once a valid response (STALL, NYET, or ACK) is received from the device or the core has finished transmitting the programmed number of LPM retries.

Note: This bit must be set only when the host is connected to a local port.

Note: Accessible only in host mode.

Bits 23:21 **LPMRCNT[2:0]:** LPM retry count

When the device gives an ERROR response, this is the number of additional LPM retries that the host performs until a valid device response (STALL, NYET, or ACK) is received.

Note: Accessible only in host mode.

Bits 20:17 **LPMCHIDX[3:0]:** LPM Channel Index

The channel number on which the LPM transaction has to be applied while sending an LPM transaction to the local device. Based on the LPM channel index, the core automatically inserts the device address and endpoint number programmed in the corresponding channel into the LPM transaction.

Note: Accessible only in host mode.

Bit 16 **L1RSMOK:** Sleep state resume OK

Indicates that the device or host can start resume from Sleep state. This bit is valid in LPM sleep (L1) state. It is set in sleep mode after a delay of 50 µs ($T_{L1Residency}$).

This bit is reset when SLPSTS is 0.

1: The application or host can start resume from Sleep state

0: The application or host cannot start resume from Sleep state

Bit 15 **SLPSTS:** Port sleep status

Device mode:

This bit is set as long as a Sleep condition is present on the USB bus. The core enters the Sleep state when an ACK response is sent to an LPM transaction and the $T_{L1TokenRetry}$ timer has expired. To stop the PHY clock, the application must set the STPPCLK bit in OTG_PCGCCTL, which asserts the PHY suspend input signal.

The application must rely on SLPSTS and not ACK in LPMRSP to confirm transition into sleep.

The core comes out of sleep:

– When there is any activity on the USB linestate

– When the application writes to the RWUSIG bit in OTG_DCTL or when the application resets or soft-disconnects the device.

Host mode:

The host transitions to Sleep (L1) state as a side-effect of a successful LPM transaction by the core to the local port with ACK response from the device. The read value of this bit reflects the current Sleep status of the port.

The core clears this bit after:

– The core detects a remote L1 wakeup signal,

– The application sets the PRST bit or the PRES bit in the OTG_HPRT register, or

– The application sets the L1Resume/ remote wakeup detected interrupt bit or disconnect detected interrupt bit in the core interrupt register (WKUPINT or DISCINT bit in OTG_GINTSTS, respectively).

0: Core not in L1

1: Core in L1

Bits 14:13 **LPMRSP[1:0]:** LPM response

Device mode:

The response of the core to LPM transaction received is reflected in these two bits.

Host mode:

Handshake response received from local device for LPM transaction

11: ACK

10: NYET

01: STALL

00: ERROR (No handshake response)

Bit 12 **L1DSEN:** L1 deep sleep enable

Enables suspending the PHY in L1 Sleep mode. For maximum power saving during L1 Sleep mode, this bit should be set to '1' by application SW in all the cases.

Bits 11:8 **BESLTHRS[3:0]:** BESL threshold

Device mode:

The core puts the PHY into deep low power mode in L1 when BESL value is greater than or equal to the value defined in this field BESL_Thres[3:0].

Host mode:

The core puts the PHY into deep low power mode in L1. BESLTHRS[3:0] specifies the time for which resume signaling is to be reflected by host ($T_{L1HubDrvResume2}$) on the USB bus when it detects device initiated resume.

BESLTHRS must not be programmed with a value greater than 1100b in host mode, because this exceeds maximum $T_{L1HubDrvResume2}$.

Thres[3:0] host mode resume signaling time (µs):

0000: 75

0001: 100

0010: 150

0011: 250

0100: 350

0101: 450

0110: 950

All other values: reserved

Bit 7 **L1SSEN:** L1 Shallow Sleep enable

Enables suspending the PHY in L1 Sleep mode. For maximum power saving during L1 Sleep mode, this bit should be set to '1' by application SW in all the cases.

Bit 6 **REMWAKE:** bRemoteWake value

Host mode:

The value of remote wake up to be sent in the wIndex field of LPM transaction.

Device mode (read-only):

This field is updated with the received LPM token bRemoteWake bmAttribute when an ACK, NYET, or STALL response is sent to an LPM transaction.

Bits 5:2 **BESL[3:0]:** Best effort service latency

Host mode:

The value of BESL to be sent in an LPM transaction. This value is also used to initiate resume for a duration $T_{L1HubDrvResume1}$ for host initiated resume.

Device mode (read-only):

This field is updated with the received LPM token BESL bmAttribute when an ACK, NYET, or STALL response is sent to an LPM transaction.

BESL[3:0]T_{BESL} (µs)

0000: 125
0001: 150
0010: 200
0011: 300
0100: 400
0101: 500
0110: 1000
0111: 2000
1000: 3000
1001: 4000
1010: 5000
1011: 6000
1100: 7000
1101: 8000
1110: 9000
1111: 10000

Bit 1 **LPMACK:** LPM token acknowledge enable

Handshake response to LPM token preprogrammed by device application software.

1: ACK

Even though ACK is preprogrammed, the core device responds with ACK only on successful LPM transaction. The LPM transaction is successful if:

– No PID/CRC5 errors in either EXT token or LPM token (else ERROR)
– Valid bLinkState = 0001B (L1) received in LPM transaction (else STALL)
– No data pending in transmit queue (else NYET).

0: NYET

The preprogrammed software bit is over-ridden for response to LPM token when:

– The received bLinkState is not L1 (STALL response), or
– An error is detected in either of the LPM token packets because of corruption (ERROR response).

Note: Accessible only in device mode.

Bit 0 **LPMEN:** LPM support enable

The application uses this bit to control the OTG_FS core LPM capabilities.

If the core operates as a non-LPM-capable host, it cannot request the connected device or hub to activate LPM mode.

If the core operates as a non-LPM-capable device, it cannot respond to any LPM transactions.

0: LPM capability is not enabled
1: LPM capability is enabled

63.15.18 OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ)

Address offset: 0x100

Reset value: 0x0200 0400

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

Bits 31:16 **PTXFSIZ[15:0]:** Host periodic Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16

Bits 15:0 **PTXSA[15:0]:** Host periodic Tx FIFO start address
This field configures the memory start address for periodic transmit FIFO RAM.

63.15.19 OTG device IN endpoint transmit FIFO x size register (OTG_DIEPTXFx)

Address offset: 0x104 + 0x04 * (x - 1), (x = 1 to 5)

Reset value: Block 1: 0x0200 0400

Reset value: Block 2: 0x0200 0600

Reset value: Block 3: 0x0200 0800

Reset value: Block 4: 0x0200 0A00

Reset value: Block 5: 0x0200 0C00

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

Bits 31:16 **INEPTXFD[15:0]:** IN endpoint Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16

Bits 15:0 **INEPTXSA[15:0]:** IN endpoint FIFOx transmit RAM start address
This field contains the memory start address for IN endpoint transmit FIFOx. The address must be aligned with a 32-bit memory location.

63.15.20 Host-mode registers

Bit values in the register descriptions are expressed in binary unless otherwise specified.

Host-mode registers affect the operation of the core in the host mode. Host mode registers must not be accessed in device mode, as the results are undefined. Host mode registers can be categorized as follows:

63.15.21 OTG host configuration register (OTG_HCFG)

Address offset: 0x400

Reset value: 0x0000 0000

This register configures the core after power-on. Do not make changes to this register after initializing the host.

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

15	14	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.	FSLSS	FSLSPCS[1:0]	
													r	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **FSLSS:** FS- and LS-only support

The application uses this bit to control the core's enumeration speed. Using this bit, the application can make the core enumerate as an FS host, even if the connected device supports HS traffic. Do not make changes to this field after initial programming.
1: FS/LS-only, even if the connected device can support HS (read-only).

Bits 1:0 **FSLSPCS[1:0]:** FS/LS PHY clock select

When the core is in FS host mode
01: PHY clock is running at 48 MHz
Others: Reserved
When the core is in LS host mode
00: Reserved
01: Select 48 MHz PHY clock frequency
10: Select 6 MHz PHY clock frequency
11: Reserved

Note: The FSLSPCS must be set on a connection event according to the speed of the connected device (after changing this bit, a software reset must be performed).

63.15.22 OTG host frame interval register (OTG_HFIR)

Address offset: 0x404

Reset value: 0x0000 EA60

This register stores the frame interval information for the current speed to which the OTG_FS controller has enumerated.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRIVL[15:0]															
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 **RLDCTRL:** Reload control

This bit allows dynamic reloading of the HFIR register during run time.

0: The HFIR cannot be reloaded dynamically

1: The HFIR can be dynamically reloaded during run time.

This bit needs to be programmed during initial configuration and its value must not be changed during run time.

Caution: RLDCTRL = 0 is not recommended.

Bits 15:0 **FRIVL[15:0]:** Frame interval

The value that the application programs to this field, specifies the interval between two consecutive SOFs (FS) or Keep-Alive tokens (LS). This field contains the number of PHY clocks that constitute the required frame interval. The application can write a value to this register only after the port enable bit of the host port control and status register (PENA bit in OTG_HPRT) has been set. If no value is programmed, the core calculates the value based on the PHY clock specified in the FS/LS PHY clock select field of the host configuration register (FSLSPCS in OTG_HCFG). Do not change the value of this field after the initial configuration, unless the RLDCTRL bit is set. In such case, the FRIVL is reloaded with each SOF event.

– Frame interval = 1 ms × (FRIVL - 1)

63.15.23 OTG host frame number/frame time remaining register (OTG_HFNUM)

Address offset: 0x408

Reset value: 0x0000 3FFF

This register indicates the current frame number. It also indicates the time remaining (in terms of the number of PHY clocks) in the current frame.

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

Bits 31:16 **FTREM[15:0]:** Frame time remaining

Indicates the amount of time remaining in the current frame, in terms of PHY clocks. This field decrements on each PHY clock. When it reaches zero, this field is reloaded with the value in the Frame interval register and a new SOF is transmitted on the USB.

Bits 15:0 **FRNUM[15:0]:** Frame number

This field increments when a new SOF is transmitted on the USB, and is cleared to 0 when it reaches 0x3FFF.

63.15.24 OTG_Host periodic transmit FIFO/queue status register (OTG_HPTXSTS)

Address offset: 0x410

Reset value: 0x0008 0100

This read-only register contains the free space information for the periodic Tx FIFO and the periodic transmit request queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PTXQTOP[7:0]								PTXQSAV[7:0]							
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
PTXFSAVL[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **PTXQTOP[7:0]:** Top of the periodic transmit request queue

This indicates the entry in the periodic Tx request queue that is currently being processed by the MAC.

This register is used for debugging.

Bit 31: Odd/Even frame

0: send in even frame

1: send in odd frame

Bits 30:27: Channel/endpoint number

Bits 26:25: Type

00: IN/OUT

01: Zero-length packet

11: Disable channel command

Bit 24: Terminate (last entry for the selected channel/endpoint)

Bits 23:16 **PTXQSAV[7:0]:** Periodic transmit request queue space available

Indicates the number of free locations available to be written in the periodic transmit request queue. This queue holds both IN and OUT requests.

00: Periodic transmit request queue is full

01: 1 location available

10: 2 locations available

bxn: n locations available (0 ≤ n ≤ 8)

Others: Reserved

Bits 15:0 **PTXFSAVL[15:0]:** Periodic transmit data FIFO space available

Indicates the number of free locations available to be written to in the periodic Tx FIFO.

Values are in terms of 32-bit words

0000: Periodic Tx FIFO is full

0001: 1 word available

0010: 2 words available

bxn: n words available (where 0 ≤ n ≤ PTXFD)

Others: Reserved

63.15.25 OTG host all channels interrupt register (OTG_HAINT)

Address offset: 0x414

Reset value: 0x0000 0000

When a significant event occurs on a channel, the host all channels interrupt register interrupts the application using the host channels interrupt bit of the core interrupt register (HCINT bit in OTG_GINTSTS). This is shown in *Figure 780*. There is one interrupt bit per channel, up to a maximum of 16 bits. Bits in this register are set and cleared when the application sets and clears bits in the corresponding host channel-x interrupt 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.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HAINT[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 **HAINT[15:0]:** Channel interrupts
One bit per channel: Bit 0 for Channel 0, bit 15 for Channel 15

63.15.26 OTG host all channels interrupt mask register (OTG_HAINTMSK)

Address offset: 0x418

Reset value: 0x0000 0000

The host all channel interrupt mask register works with the host all channel interrupt register to interrupt the application when an event occurs on a channel. There is one interrupt mask bit per channel, up to a maximum of 16 bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HAINTM[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 **HAINTM[15:0]:** Channel interrupt mask
0: Masked interrupt
1: Unmasked interrupt
One bit per channel: Bit 0 for channel 0, bit 15 for channel 15

63.15.27 OTG host port control and status register (OTG_HPRT)

Address offset: 0x440

Reset value: 0x0000 0000

This register is available only in host mode. Currently, the OTG host supports only one port.

A single register holds USB port-related information such as USB reset, enable, suspend, resume, connect status, and test mode for each port. It is shown in *Figure 780*. The rc_w1 bits in this register can trigger an interrupt to the application through the host port interrupt bit of the core interrupt register (HPRTINT bit in OTG_GINTSTS). On a port interrupt, the application must read this register and clear the bit that caused the interrupt. For the rc_w1 bits, the application must write a 1 to the bit to clear the interrupt.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSPD[1:0]		PTCTL[3]
													r	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTCTL[2:0]			PPWR	PLSTS[1:0]		Res.	PRST	PSUSP	PRES	POC CHNG	POCA	PEN CHNG	PENA	PCDET	PCSTS
rw	rw	rw	rw	r	r		rw	rs	rw	rc_w1	r	rc_w1	rc_w1	rc_w1	r

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

Bits 18:17 **PSPD[1:0]:** Port speed

Indicates the speed of the device attached to this port.
01: Full speed
10: Low speed
11: Reserved

Bits 16:13 **PTCTL[3:0]:** Port test control

The application writes a nonzero value to this field to put the port into a Test mode, and the corresponding pattern is signaled on the port.
0000: Test mode disabled
0001: Test_J mode
0010: Test_K mode
0011: Test_SE0_NAK mode
0100: Test_Packet mode
0101: Test_Force_Enable
Others: Reserved

Bit 12 **PPWR:** Port power

The application uses this field to control power to this port, and the core clears this bit on an overcurrent condition.
0: Power off
1: Power on

Bits 11:10 **PLSTS[1:0]:** Port line status

Indicates the current logic level USB data lines
Bit 10: Logic level of OTG_DP
Bit 11: Logic level of OTG_DM

Bit 9 Reserved, must be kept at reset value.

Bit 8 **PRST:** Port reset

When the application sets this bit, a reset sequence is started on this port. The application must time the reset period and clear this bit after the reset sequence is complete.

0: Port not in reset

1: Port in reset

The application must leave this bit set for a minimum duration of at least 10 ms to start a reset on the port. The application can leave it set for another 10 ms in addition to the required minimum duration, before clearing the bit, even though there is no maximum limit set by the USB standard.

High speed: 50 ms

Full speed/Low speed: 10 ms

Bit 7 **PSUSP:** Port suspend

The application sets this bit to put this port in suspend mode. The core only stops sending SOFs when this is set. To stop the PHY clock, the application must set the port clock stop bit, which asserts the suspend input pin of the PHY.

The read value of this bit reflects the current suspend status of the port. This bit is cleared by the core after a remote wakeup signal is detected or the application sets the port reset bit or port resume bit in this register or the resume/remote wakeup detected interrupt bit or disconnect detected interrupt bit in the core interrupt register (WKUPINT or DISCINT in OTG_GINTSTS, respectively).

0: Port not in suspend mode

1: Port in suspend mode

Bit 6 **PRES:** Port resume

The application sets this bit to drive resume signaling on the port. The core continues to drive the resume signal until the application clears this bit.

If the core detects a USB remote wakeup sequence, as indicated by the port resume/remote wakeup detected interrupt bit of the core interrupt register (WKUPINT bit in OTG_GINTSTS), the core starts driving resume signaling without application intervention and clears this bit when it detects a disconnect condition. The read value of this bit indicates whether the core is currently driving resume signaling.

0: No resume driven

1: Resume driven

When LPM is enabled and the core is in L1 state, the behavior of this bit is as follow:

1. The application sets this bit to drive resume signaling on the port.

2. The core continues to drive the resume signal until a predetermined time specified in BESLTHRS[3:0] field of OTG_GLPMCFG register.

3. If the core detects a USB remote wakeup sequence, as indicated by the port L1Resume/Remote L1Wakeup detected interrupt bit of the core interrupt register (WKUPINT in OTG_GINTSTS), the core starts driving resume signaling without application intervention and clears this bit at the end of resume.This bit can be set or cleared by both the core and the application. This bit is cleared by the core even if there is no device connected to the host.

Bit 5 **POCCHNG:** Port overcurrent change

The core sets this bit when the status of the port overcurrent active bit (bit 4) in this register changes.

Bit 4 **POCA:** Port overcurrent active

Indicates the overcurrent condition of the port.

0: No overcurrent condition

1: Overcurrent condition

Bit 3 **PENCHNG:** Port enable/disable change

The core sets this bit when the status of the port enable bit 2 in this register changes.

Bit 2 **PENA:** Port enable

A port is enabled only by the core after a reset sequence, and is disabled by an overcurrent condition, a disconnect condition, or by the application clearing this bit. The application cannot set this bit by a register write. It can only clear it to disable the port. This bit does not trigger any interrupt to the application.

0: Port disabled

1: Port enabled

Bit 1 **PCDET:** Port connect detected

The core sets this bit when a device connection is detected to trigger an interrupt to the application using the host port interrupt bit in the core interrupt register (HPRTINT bit in OTG_GINTSTS). The application must write a 1 to this bit to clear the interrupt.

Bit 0 **PCSTS:** Port connect status

0: No device is attached to the port

1: A device is attached to the port

63.15.28 OTG host channel x characteristics register (OTG_HCCHARx)

Address offset: 0x500 + 0x20 * x, (x = 0 to 11)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CHENA	CHDIS	ODD FRM	DAD[6:0]							MCNT[1:0]		EPTYP[1:0]		LSDEV	Res.
rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPDIR	EPNUM[3:0]				MPSIZ[10:0]										
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **CHENA:** Channel enable

This field is set by the application and cleared by the OTG host.

0: Channel disabled

1: Channel enabled

Bit 30 **CHDIS:** Channel disable

The application sets this bit to stop transmitting/receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel disabled interrupt before treating the channel as disabled.

Bit 29 **ODDFRM:** Odd frame

This field is set (reset) by the application to indicate that the OTG host must perform a transfer in an odd frame. This field is applicable for only periodic (isochronous and interrupt) transactions.

0: Even frame

1: Odd frame

Bits 28:22 **DAD[6:0]:** Device address

This field selects the specific device serving as the data source or sink.

Bits 21:20 **MCNT[1:0]:** Multicount

This field indicates to the host the number of transactions that must be executed per frame for this periodic endpoint. For non-periodic transfers, this field is not used

00: Reserved. This field yields undefined results

01: 1 transaction

10: 2 transactions per frame to be issued for this endpoint

11: 3 transactions per frame to be issued for this endpoint

Note: This field must be set to at least 01.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

Indicates the transfer type selected.

00: Control

01: Isochronous

10: Bulk

11: Interrupt

Bit 17 **LSDEV:** Low-speed device

This field is set by the application to indicate that this channel is communicating to a low-speed device.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **EPDIR:** Endpoint direction

Indicates whether the transaction is IN or OUT.

0: OUT

1: IN

Bits 14:11 **EPNUM[3:0]:** Endpoint number

Indicates the endpoint number on the device serving as the data source or sink.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

Indicates the maximum packet size of the associated endpoint.

63.15.29 OTG host channel x interrupt register (OTG_HCINTx)

Address offset: 0x508 + 0x20 * x, (x = 0 to 11)

Reset value: 0x0000 0000

This register indicates the status of a channel with respect to USB- and AHB-related events. It is shown in *Figure 780*. The application must read this register when the host channels interrupt bit in the core interrupt register (HCINT bit in OTG_GINTSTS) is set. Before the application can read this register, it must first read the host all channels interrupt (OTG_HAINT) register to get the exact channel number for the host channel-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_HAINT and OTG_GINTSTS registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DTERR	FRM OR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRC
					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 **DTERR:** Data toggle error.

Bit 9 **FRMOR:** Frame overrun.

Bit 8 **BBERR:** Babble error.

Bit 7 **TXERR:** Transaction error.

Indicates one of the following errors occurred on the USB.
CRC check failure
Timeout
Bit stuff error
False EOP

Bit 6 Reserved, must be kept at reset value.

Bit 5 **ACK:** ACK response received/transmitted interrupt.

Bit 4 **NAK:** NAK response received interrupt.

Bit 3 **STALL:** STALL response received interrupt.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CHH:** Channel halted.

Indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application.

Bit 0 **XFRC:** Transfer completed.

Transfer completed normally without any errors.

63.15.30 OTG host channel x interrupt mask register (OTG_HCINTMSKx)

Address offset: 0x50C + 0x20 * x, (x = 0 to 11)

Reset value: 0x0000 0000

This register reflects the mask for each channel status described in the previous 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.	DTERRM	FRMORM	BBERRM	TXERRM	Res.	ACKM	NAKM	STALLM	Res.	CHHM	XFRCM
					rw	rw	rw	rw		rw	rw	rw		rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DTERRM:** Data toggle error mask.

0: Masked interrupt
1: Unmasked interrupt

Bit 9 **FRMORM:** Frame overrun mask.

0: Masked interrupt
1: Unmasked interrupt

Bit 8 **BBERRM:** Babble error mask.
0: Masked interrupt
1: Unmasked interrupt

Bit 7 **TXERRM:** Transaction error mask.
0: Masked interrupt
1: Unmasked interrupt

Bit 6 Reserved, must be kept at reset value.

Bit 5 **ACKM:** ACK response received/transmitted interrupt mask.
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **NAKM:** NAK response received interrupt mask.
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **STALLM:** STALL response received interrupt mask.
0: Masked interrupt
1: Unmasked interrupt

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CHHM:** Channel halted mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed mask
0: Masked interrupt
1: Unmasked interrupt

63.15.31 OTG host channel x transfer size register (OTG_HCTSIZx)

Address offset: 0x510 + 0x20 * x, (x = 0 to 11)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DO PNG	DPID[1:0]		PKTCNT[9:0]										XFRSIZ[18:16]		
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
XFRSIZ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DOPNG:** Do Ping

This bit is used only for OUT transfers. Setting this field to 1 directs the host to do PING protocol.

Note: Do not set this bit for IN transfers. If this bit is set for IN transfers, it disables the channel.

Bits 30:29 **DPID[1:0]:** Data PID

The application programs this field with the type of PID to use for the initial transaction. The host maintains this field for the rest of the transfer.

00: DATA0

10: DATA1

11: SETUP (control) / reserved (non-control)

Bits 28:19 **PKTCNT[9:0]:** Packet count

This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN).

The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion.

Bits 18:0 **XFRSIZ[18:0]:** Transfer size

For an OUT, this field is the number of data bytes the host sends during the transfer.

For an IN, this field is the buffer size that the application has reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).

63.15.32 Device-mode registers

These registers must be programmed every time the core changes to device mode

63.15.33 OTG device configuration register (OTG_DCFG)

Address offset: 0x800

Reset value: 0x0220 0000

This register configures the core in device mode after power-on or after certain control commands or enumeration. Do not make changes to this register after initial programming.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRATIM	Res.	Res.	PFIVL[1:0]		DAD[6:0]							Res.	NZLSOHSK	DSPD[1:0]	
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 **ERRATIM:** Erratic error interrupt mask
1: Mask early suspend interrupt on erratic error
0: Early suspend interrupt is generated on erratic error

Bit 14 Reserved, must be kept at reset value.

Bit 13 Reserved, must be kept at reset value.

Bits 12:11 **PFIVL[1:0]:** Periodic frame interval
Indicates the time within a frame at which the application must be notified using the end of periodic frame interrupt. This can be used to determine if all the isochronous traffic for that frame is complete.
00: 80% of the frame interval
01: 85% of the frame interval
10: 90% of the frame interval
11: 95% of the frame interval

Bits 10:4 **DAD[6:0]:** Device address
The application must program this field after every SetAddress control command.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **NZLSOHSK:** Non-zero-length status OUT handshake
The application can use this field to select the handshake the core sends on receiving a nonzero-length data packet during the OUT transaction of a control transfer's status stage.
1:Send a STALL handshake on a nonzero-length status OUT transaction and do not send the received OUT packet to the application.
0:Send the received OUT packet to the application (zero-length or nonzero-length) and send a handshake based on the NAK and STALL bits for the endpoint in the device endpoint control register.

Bits 1:0 **DSPD[1:0]:** Device speed
Indicates the speed at which the application requires the core to enumerate, or the maximum speed the application can support. However, the actual bus speed is determined only after the chirp sequence is completed, and is based on the speed of the USB host to which the core is connected.
00: Reserved
01: Reserved
10: Reserved
11: Full speed (USB 1.1 transceiver clock is 48 MHz)

63.15.34 OTG device control register (OTG_DCTL)

Address offset: 0x804

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.	DSBESLRJCT	Res.	Res.
													rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL[2:0]			GONSTS	GINSTS	SDIS	RWUSIG
				rw	w	w	w	w	rw	rw	rw	r	r	rw	rw

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

Bit 18 **DSBESLRJCT:** Deep sleep BESL reject

Core rejects LPM request with BESL value greater than BESL threshold programmed. NYET response is sent for LPM tokens with BESL value greater than BESL threshold. By default, the deep sleep BESL reject feature is disabled.

Bits 17:12 Reserved, must be kept at reset value.

Bit 11 **POPRGDNE:** Power-on programming done

The application uses this bit to indicate that register programming is completed after a wakeup from power down mode.

Bit 10 **CGONAK:** Clear global OUT NAK

Writing 1 to this field clears the Global OUT NAK.

Bit 9 **SGONAK:** Set global OUT NAK

Writing 1 to this field sets the Global OUT NAK.
The application uses this bit to send a NAK handshake on all OUT endpoints.
The application must set the this bit only after making sure that the Global OUT NAK effective bit in the core interrupt register (GONAKEFF bit in OTG_GINTSTS) is cleared.

Bit 8 **CGINAK:** Clear global IN NAK

Writing 1 to this field clears the Global IN NAK.

Bit 7 **SGINAK:** Set global IN NAK

Writing 1 to this field sets the Global non-periodic IN NAK.The application uses this bit to send a NAK handshake on all non-periodic IN endpoints.
The application must set this bit only after making sure that the Global IN NAK effective bit in the core interrupt register (GINAKEFF bit in OTG_GINTSTS) is cleared.

Bits 6:4 **TCTL[2:0]:** Test control

000: Test mode disabled
001: Test_J mode
010: Test_K mode
011: Test_SE0_NAK mode
100: Test_Packet mode
101: Test_Force_Enable
Others: Reserved

Bit 3 **GONSTS:** Global OUT NAK status

0:A handshake is sent based on the FIFO status and the NAK and STALL bit settings.

1:No data is written to the Rx FIFO, irrespective of space availability. Sends a NAK handshake on all packets, except on SETUP transactions. All isochronous OUT packets are dropped.

Bit 2 **GINSTS:** Global IN NAK status

0:A handshake is sent out based on the data availability in the transmit FIFO.

1:A NAK handshake is sent out on all non-periodic IN endpoints, irrespective of the data availability in the transmit FIFO.

Bit 1 **SDIS:** Soft disconnect

The application uses this bit to signal the USB OTG core to perform a soft disconnect. As long as this bit is set, the host does not see that the device is connected, and the device does not receive signals on the USB. The core stays in the disconnected state until the application clears this bit.

0:Normal operation. When this bit is cleared after a soft disconnect, the core generates a device connect event to the USB host. When the device is reconnected, the USB host restarts device enumeration.

1:The core generates a device disconnect event to the USB host.

Bit 0 **RWUSIG:** Remote wakeup signaling

When the application sets this bit, the core initiates remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the suspend state. As specified in the USB 2.0 specification, the application must clear this bit 1 ms to 15 ms after setting it.

If LPM is enabled and the core is in the L1 (sleep) state, when the application sets this bit, the core initiates L1 remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the sleep state. As specified in the LPM specification, the hardware automatically clears this bit 50 µs ($T_{L1DevDrvResume}$) after being set by the application. The application must not set this bit when bRemoteWake from the previous LPM transaction is zero (refer to REMWAKE bit in GLPMCFG register).

Table 651 contains the minimum duration (according to device state) for which the Soft disconnect (SDIS) bit must be set for the USB host to detect a device disconnect. To accommodate clock jitter, it is recommended that the application add some extra delay to the specified minimum duration.

Table 651. Minimum duration for soft disconnect

Operating speed	Device state	Minimum duration
Full speed	Suspended	1 ms + 2.5 µs
Full speed	Idle	2.5 µs
Full speed	Not Idle or suspended (Performing transactions)	2.5 µs

63.15.35 OTG device status register (OTG_DSTS)

Address offset: 0x808

Reset value: 0x0000 0010

This register indicates the status of the core with respect to USB-related events. It must be read on interrupts from the device all interrupts (OTG_DAINT) 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.	DEVLNSTS[1:0]		FNSOF[13:8]					
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FNSOF[7:0]								Res.	Res.	Res.	Res.	EERR	ENUMSPD[1:0]		SUSP STS
r	r	r	r	r	r	r	r					r	r	r	r

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

Bits 23:22 **DEVLNSTS[1:0]:** Device line status

Indicates the current logic level USB data lines.
Bit [23]: Logic level of D+
Bit [22]: Logic level of D-

Bits 21:8 **FNSOF[13:0]:** Frame number of the received SOF

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

Bit 3 **EERR:** Erratic error

The core sets this bit to report any erratic errors.
Due to erratic errors, the OTG_FS controller goes into suspended state and an interrupt is generated to the application with Early suspend bit of the OTG_GINTSTS register (ESUSP bit in OTG_GINTSTS). If the early suspend is asserted due to an erratic error, the application can only perform a soft disconnect recover.

Bits 2:1 **ENUMSPD[1:0]:** Enumerated speed

Indicates the speed at which the OTG_FS controller has come up after speed detection through a chirp sequence.
11: Full speed using embedded FS PHY
Others: reserved

Bit 0 **SUSPSTS:** Suspend status

In device mode, this bit is set as long as a suspend condition is detected on the USB. The core enters the suspended state when there is no activity on the USB data lines for a period of 3 ms. The core comes out of the suspend:

– When there is an activity on the USB data lines
– When the application writes to the remote wakeup signaling bit in the OTG_DCTL register (RWUSIG bit in OTG_DCTL).

63.15.36 OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK)

Address offset: 0x810

Reset value: 0x0000 0000

This register works with each of the OTG_DIEPINTx registers for all endpoints to generate an interrupt per IN endpoint. The IN endpoint interrupt for a specific status in the OTG_DIEPINTx register can be masked by writing to the corresponding bit in this register. Status bits are masked by default.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NAKM	Res.	Res.	Res.	Res.	Res.	Res.	INEPNEM	INEPNMM	ITTXFEMSK	TOM	Res.	EPDM	XFRCM
		rw							rw	rw	rw	rw		rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAKM:** NAK interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 12:10 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.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **INEPNEM:** IN endpoint NAK effective mask
0: Masked interrupt
1: Unmasked interrupt

Bit 5 **INEPNMM:** IN token received with EP mismatch mask
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **ITTXFEMSK:** IN token received when Tx FIFO empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **TOM:** Timeout condition mask (Non-isochronous endpoints)
0: Masked interrupt
1: Unmasked interrupt

Bit 2 Reserved, must be kept at reset value.

Bit 1 **EPDM:** Endpoint disabled interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed interrupt mask
0: Masked interrupt
1: Unmasked interrupt

63.15.37 OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK)

Address offset: 0x814

Reset value: 0x0000 0000

This register works with each of the OTG_DOEPINTx registers for all endpoints to generate an interrupt per OUT endpoint. The OUT endpoint interrupt for a specific status in the OTG_DOEPINTx register can be masked by writing into the corresponding bit in this register. Status bits are masked by default.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NAK MSK	BERR M	Res.	Res.	Res.	OUT PKT ERRM	Res.	Res.	STS PHSR XM	OTEPD M	STUPM	Res.	EPDM	XFRC M
		rw	rw				rw			rw	rw	rw		rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAKMSK:** NAK interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 12 **BERRM**: Babble error interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **OUTPKTERRM:** Out packet error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **STSPHSRXM**: Status phase received for control write mask
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **OTEPDM:** OUT token received when endpoint disabled mask. Applies to control OUT endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **STUPM:** STUPM: SETUP phase done mask. Applies to control endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 2 Reserved, must be kept at reset value.

Bit 1 **EPDM:** Endpoint disabled interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed interrupt mask
0: Masked interrupt
1: Unmasked interrupt

63.15.38 OTG device all endpoints interrupt register (OTG_DAINT)

Address offset: 0x818

Reset value: 0x0000 0000

When a significant event occurs on an endpoint, a OTG_DAINT register interrupts the application using the device OUT endpoints interrupt bit or device IN endpoints interrupt bit of the OTG_GINTSTS register (OEPINT or IEPINT in OTG_GINTSTS, respectively). There is one interrupt bit per endpoint, up to a maximum of 16 bits for OUT endpoints and 16 bits for IN endpoints. For a bidirectional endpoint, the corresponding IN and OUT interrupt bits are used. Bits in this register are set and cleared when the application sets and clears bits in the corresponding device endpoint-x interrupt register (OTG_DIEPINTx/OTG_DOEPINTx).

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

Bits 31:16 **OEPINT[15:0]:** OUT endpoint interrupt bits
One bit per OUT endpoint:
Bit 16 for OUT endpoint 0, bit 19 for OUT endpoint 3.

Bits 15:0 **IEPINT[15:0]:** IN endpoint interrupt bits
One bit per IN endpoint:
Bit 0 for IN endpoint 0, bit 3 for endpoint 3.

63.15.39 OTG all endpoints interrupt mask register (OTG_DAINTMSK)

Address offset: 0x81C

Reset value: 0x0000 0000

The OTG_DAINTMSK register works with the device endpoint interrupt register to interrupt the application when an event occurs on a device endpoint. However, the OTG_DAINT register bit corresponding to that interrupt is still set.

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

Bits 31:16 **OEPM[15:0]:** OUT EP interrupt mask bits
One per OUT endpoint:
Bit 16 for OUT EP 0, bit 19 for OUT EP 3
0: Masked interrupt
1: Unmasked interrupt

Bits 15:0 **IEPM[15:0]:** IN EP interrupt mask bits
One bit per IN endpoint:
Bit 0 for IN EP 0, bit 3 for IN EP 3
0: Masked interrupt
1: Unmasked interrupt

63.15.40 OTG device V_{BUS} discharge time register (OTG_DVBUSDIS)

Address offset: 0x0828

Reset value: 0x0000 17D7

This register specifies the V_{BUS} discharge time after V_{BUS} pulsing during SRP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VBUSDT[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 **VBUSDT[15:0]:** Device V_{BUS} discharge time
Specifies the V_{BUS} discharge time after V_{BUS} pulsing during SRP. This value equals:
V_{BUS} discharge time in PHY clocks / 1 024
Depending on V_{BUS} load, this value may need adjusting.

63.15.41 OTG device V_{BUS} pulsing time register (OTG_DVBUSPULSE)

Address offset: 0x082C

Reset value: 0x0000 05B8

This register specifies the V_{BUS} pulsing time during SRP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DVBUSP[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 **DVBUSP[15:0]:** Device V_{BUS} pulsing time. This feature is only relevant to OTG1.3.
Specifies the V_{BUS} pulsing time during SRP. This value equals:
V_{BUS} pulsing time in PHY clocks / 1 024

63.15.42 OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK)

Address offset: 0x834

Reset value: 0x0000 0000

This register is used to control the IN endpoint FIFO empty interrupt generation (TXFE_OTG_DIEPINTx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INEPTXFEM[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 **INEPTXFEM[15:0]:** IN EP Tx FIFO empty interrupt mask bits
These bits act as mask bits for OTG_DIEPINTx.
TXFE interrupt one bit per IN endpoint:
Bit 0 for IN endpoint 0, bit 3 for IN endpoint 3
0: Masked interrupt
1: Unmasked interrupt

63.15.43 OTG device control IN endpoint 0 control register (OTG_DIEPCTL0)

Address offset: 0x900

Reset value: 0x0000 0000

This section describes the OTG_DIEPCTL0 register for USB_OTG FS. Nonzero control endpoints use registers for endpoints 1–3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	Res.	Res.	SNAK	CNAK	TXFNUM[3:0]				STALL	Res.	EPTYP[1:0]		NAK STS	Res.
rs	rs			w	w	rw	rw	rw	rw	rs		r	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBAEP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPSIZ[1:0]	
r														rw	rw

Bit 31 **EPENA:** Endpoint enable
The application sets this bit to start transmitting data on the endpoint 0.
The core clears this bit before setting any of the following interrupts on this endpoint:
– Endpoint disabled
– Transfer completed

Bit 30 **EPDIS:** Endpoint disable
The application sets this bit to stop transmitting data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

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

Bit 27 **SNAK:** Set NAK
A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for an endpoint after a SETUP packet is received on that endpoint.

Bit 26 **CNAK:** Clear NAK
A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 **TXFNUM[3:0]:** Tx FIFO number
This value is set to the FIFO number that is assigned to IN endpoint 0.

Bit 21 **STALL:** STALL handshake
The application can only set this bit, and the core clears it when a SETUP token is received for this endpoint. If a NAK bit, a Global IN NAK or Global OUT NAK is set along with this bit, the STALL bit takes priority.

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **EPTYP[1:0]:** Endpoint type
Hardcoded to '00' for control.

Bit 17 **NAKSTS:** NAK status
Indicates the following:
0: The core is transmitting non-NAK handshakes based on the FIFO status
1: The core is transmitting NAK handshakes on this endpoint.
When this bit is set, either by the application or core, the core stops transmitting data, even if there are data available in the Tx FIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **USBAEP:** USB active endpoint
This bit is always set to 1, indicating that control endpoint 0 is always active in all configurations and interfaces.

Bits 14:2 Reserved, must be kept at reset value.

Bits 1:0 **MPSIZ[1:0]:** Maximum packet size
The application must program this field with the maximum packet size for the current logical endpoint.
00: 64 bytes
01: 32 bytes
10: 16 bytes
11: 8 bytes

63.15.44 OTG device IN endpoint x control register (OTG_DIEPCTLx)

Address offset: 0x900 + 0x20 * x, (x = 1 to 5)

Reset value: 0x0000 0000

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	SODD FRM	SD0 PID/ SEVN FRM	SNAK	CNAK	TXFNUM[3:0]				STALL	Res.	EPTYP[1:0]		NAK STS	EO NUM/ DPID
rs	rs	w	w	w	w	rw	rw	rw	rw	rw		rw	rw	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	MPSIZ[10:0]										
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EPENA:** Endpoint enable

The application sets this bit to start transmitting data on an endpoint.
The core clears this bit before setting any of the following interrupts on this endpoint:
– SETUP phase done
– Endpoint disabled
– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

Bit 29 **SODDFRM:** Set odd frame

Applies to isochronous IN and OUT endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to odd frame.

Bit 28 **SD0PID:** Set DATA0 PID

Applies to interrupt/bulk IN endpoints only.
Writing to this field sets the endpoint data PID (DPID) field in this register to DATA0.

SEVNFRM: Set even frame

Applies to isochronous IN endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to even frame.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 **TXFNUM[3:0]:** Tx FIFO number

These bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.
This field is valid only for IN endpoints.

Bit 21 **STALL:** STALL handshake

Applies to non-control, non-isochronous IN endpoints only (access type is rw).
The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

This is the transfer type supported by this logical endpoint.
00: Control
01: Isochronous
10: Bulk
11: Interrupt

Bit 17 **NAKSTS:** NAK status

It indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit:

For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there are data available in the Tx FIFO.

For isochronous IN endpoints: The core sends out a zero-length data packet, even if there are data available in the Tx FIFO.

Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **EONUM:** Even/odd frame

Applies to isochronous IN endpoints only.

Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM fields in this register.

0: Even frame

1: Odd frame

DPID: Endpoint data PID

Applies to interrupt/bulk IN endpoints only.

Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application uses the SD0PID register field to program either DATA0 or DATA1 PID.

0: DATA0

1: DATA1

Bit 15 **USBAEP:** USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

63.15.45 OTG device IN endpoint x interrupt register (OTG_DIEPINTx)

Address offset: 0x908 + 0x20 * x, (x = 0 to 5)

Reset value: 0x0000 0080

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in *Figure 780*. The application must read this register when the IN endpoints interrupt bit of the core interrupt register (IEPINT in OTG_GINTSTS) is set. Before the application can read this register, it must first read the device all endpoints interrupt (OTG_DAINT) register to get the exact endpoint number for the device endpoint-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NAK	Res.	PKTD RPSTS	Res.	Res.	Res.	TXFE	IN EPNE	IN EPNM	ITTXFE	TOC	Res.	EP DISD	XFRC
		rc_w1		rc_w1				r	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAK:** NAK input

The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO.

Bit 12 Reserved, must be kept at reset value.

Bit 11 **PKTDRPSTS:** Packet dropped status

This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt.

Bit 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.

Bit 7 **TXFE:** Transmit FIFO empty

This interrupt is asserted when the Tx FIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the Tx FIFO Empty Level bit in the OTG_GAHBCFG register (TXFELVL bit in OTG_GAHBCFG).

Bit 6 **INEPNE:** IN endpoint NAK effective

This bit can be cleared when the application clears the IN endpoint NAK by writing to the CNAK bit in OTG_DIEPCTLx.

This interrupt indicates that the core has sampled the NAK bit set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit set by the application has taken effect in the core.

This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.

Bit 5 **INEPNM**: IN token received with EP mismatch

Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received.

Bit 4 **ITTXFE:** IN token received when Tx FIFO is empty

Indicates that an IN token was received when the associated Tx FIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received.

Bit 3 **TOC:** Timeout condition

Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **EPDISD:** Endpoint disabled interrupt

This bit indicates that the endpoint is disabled per the application's request.

Bit 0 **XFRC:** Transfer completed interrupt

This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

63.15.46 OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0)

Address offset: 0x910

Reset value: 0x0000 0000

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the endpoint enable bit in the device control endpoint 0 control registers (EPENA in OTG_DIEPCTL0), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTCNT[1:0]		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.	XFRSIZ[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:19 **PKTCNT[1:0]:** Packet count

Indicates the total number of USB packets that constitute the transfer size amount of data for endpoint 0.

This field is decremented every time a packet (maximum size or short packet) is read from the Tx FIFO.

Bits 18:7 Reserved, must be kept at reset value.

Bits 6:0 **XFRSIZ[6:0]:** Transfer size

Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet from the external memory is written to the Tx FIFO.

63.15.47 OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx)

Address offset: 0x918 + 0x20 * x, (x = 0 to 5)

Reset value: 0x0000 0200

This read-only register contains the free space information for the device IN endpoint Tx 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
INEPTFSAV[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 **INEPTFSAV[15:0]:** IN endpoint Tx FIFO space available

Indicates the amount of free space available in the endpoint Tx FIFO.

Values are in terms of 32-bit words:

0x0: Endpoint Tx FIFO is full

0x1: 1 word available

0x2: 2 words available

0xn: n words available

Others: Reserved

63.15.48 OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx)

Address offset: 0x910 + 0x20 * x, (x = 1 to 5)

Reset value: 0x0000 0000

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using the endpoint enable bit in the OTG_DIEPCTLx registers (EPENA bit in OTG_DIEPCTLx), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MCNT[1:0]		PKTCNT[9:0]										XFRSIZ[18:16]		
	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
XFRSIZ[15:0]															
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.

Bits 30:29 **MCNT[1:0]:** Multi count

For periodic IN endpoints, this field indicates the number of packets that must be transmitted per frame on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints.

01: 1 packet

10: 2 packets

11: 3 packets

Bits 28:19 **PKTCNT[9:0]:** Packet count

Indicates the total number of USB packets that constitute the transfer size amount of data for this endpoint.

This field is decremented every time a packet (maximum size or short packet) is read from the Tx FIFO.

Bits 18:0 **XFRSIZ[18:0]:** Transfer size

This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet from the external memory is written to the Tx FIFO.

63.15.49 OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0)

Address offset: 0xB00

Reset value: 0x0000 8000

This section describes the OTG_DOEPCTL0 register. Nonzero control endpoints use registers for endpoints 1–3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	Res.	Res.	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP[1:0]		NAK STS	Res.
w	r			w	w					rs	rw	r	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPSIZ[1:0]	
r														r	r

Bit 31 **EPENA:** Endpoint enable

The application sets this bit to start transmitting data on endpoint 0.
The core clears this bit before setting any of the following interrupts on this endpoint:

– SETUP phase done
– Endpoint disabled
– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application cannot disable control OUT endpoint 0.

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

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL:** STALL handshake

The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 **SNPM:** Snoop mode

This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

Hardcoded to 2'b00 for control.

Bit 17 **NAKSTS:** NAK status

Indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit, the core stops receiving data, even if there is space in the Rx FIFO to accommodate the incoming packet. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **USBAEP:** USB active endpoint

This bit is always set to 1, indicating that a control endpoint 0 is always active in all configurations and interfaces.

Bits 14:2 Reserved, must be kept at reset value.

Bits 1:0 **MPSIZ[1:0]:** Maximum packet size

The maximum packet size for control OUT endpoint 0 is the same as what is programmed in control IN endpoint 0.

00: 64 bytes

01: 32 bytes

10: 16 bytes

11: 8 bytes

63.15.50 OTG device OUT endpoint x interrupt register (OTG_DOEPINTx)

Address offset: 0xB08 + 0x20 * x, (x = 0 to 5)

Reset value: 0x0000 0080

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in *Figure 780*. The application must read this register when the OUT endpoints interrupt bit of the OTG_GINTSTS register (OEPINT bit in OTG_GINTSTS) is set. Before the application can read this register, it must first read the OTG_DAINT register to get the exact endpoint number for the OTG_DOEPINTx register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NAK	BERR	Res.	Res.	Res.	Res.	Res.	Res.	STSPH SRX	OTEP DIS	STUP	Res.	EP DISD	XFRC
		rc_w1	rc_w1							rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAK:** NAK input

The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO.

Bit 12 **BERR:** Babble error interrupt

The core generates this interrupt when babble is received for the endpoint.

Bits 11:10 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.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **STSPHSRX**: Status phase received for control write

This interrupt is valid only for control OUT endpoints. This interrupt is generated only after OTG_FS has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a control write transfer. The application can use this interrupt to ACK or STALL the status phase, after it has decoded the data phase.

Bit 4 **OTEPDIS:** OUT token received when endpoint disabled

Applies only to control OUT endpoints.
Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received.

Bit 3 **STUP:** SETUP phase done

Applies to control OUT endpoint only.Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **EPDISD:** Endpoint disabled interrupt

This bit indicates that the endpoint is disabled per the application’s request.

Bit 0 **XFRC:** Transfer completed interrupt

This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

63.15.51 OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0)

Address offset: 0xB10

Reset value: 0x0000 0000

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the endpoint enable bit in the OTG_DOEPCTL0 registers (EPENA bit in OTG_DOEPCTL0), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–5.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	STUPCNT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTCNT	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.	XFRSIZ[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **STUPCNT[1:0]:** SETUP packet count

This field specifies the number of back-to-back SETUP data packets the endpoint can receive.
01: 1 packet
10: 2 packets
11: 3 packets

Bits 28:20 Reserved, must be kept at reset value.

Bit 19 **PKTCNT:** Packet count

This field is decremented to zero after a packet is written into the Rx FIFO.

Bits 18:7 Reserved, must be kept at reset value.

Bits 6:0 **XFRSIZ[6:0]:** Transfer size

Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.
The core decrements this field every time a packet is read from the Rx FIFO and written to the external memory.

63.15.52 OTG device OUT endpoint x control register (OTG_DOEPCTLx)

Address offset: 0xB00 + 0x20 * x, (x = 1 to 5)

Reset value: 0x0000 0000

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	SD1 PID/ SODD FRM	SD0 PID/ SEVN FRM	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP[1:0]		NAK STS	EO NUM/ DPID
rs	rs	w	w	w	w					rw	rw	rw	rw	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	MPSIZ[10:0]										
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EPENA:** Endpoint enable

Applies to IN and OUT endpoints.

The application sets this bit to start transmitting data on an endpoint.

The core clears this bit before setting any of the following interrupts on this endpoint:

– SETUP phase done

– Endpoint disabled

– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

Bit 29 **SD1PID:** Set DATA1 PID

Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the endpoint data PID (DPID) field in this register to DATA1.

SODDFRM: Set odd frame

Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd frame (EONUM) field to odd frame.

Bit 28 **SD0PID:** Set DATA0 PID

Applies to interrupt/bulk OUT endpoints only.

Writing to this field sets the endpoint data PID (DPID) field in this register to DATA0.

SEVNFRM: Set even frame

Applies to isochronous OUT endpoints only.

Writing to this field sets the Even/Odd frame (EONUM) field to even frame.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.

Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL:** STALL handshake

Applies to non-control, non-isochronous OUT endpoints only (access type is rw).

The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.

Applies to control endpoints only (access type is rs).

The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 **SNPM:** Snoop mode

This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

This is the transfer type supported by this logical endpoint.

00: Control

01: Isochronous

10: Bulk

11: Interrupt

Bit 17 **NAKSTS:** NAK status

Indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit:

The core stops receiving any data on an OUT endpoint, even if there is space in the Rx FIFO to accommodate the incoming packet.

Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **EONUM:** Even/odd frame

Applies to isochronous IN and OUT endpoints only.

Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM fields in this register.

0: Even frame

1: Odd frame

DPID: Endpoint data PID

Applies to interrupt/bulk OUT endpoints only.

Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application uses the SD0PID register field to program either DATA0 or DATA1 PID.

0: DATA0

1: DATA1

Bit 15 **USBAEP:** USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

63.15.53 OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx)

Address offset: 0xB10 + 0x20 * x, (x = 1 to 5)

Reset value: 0x0000 0000

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using endpoint enable bit of the OTG_DOEPCTLx registers (EPENA bit in OTG_DOEPCTLx), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	RXDPID/ STUPCNT[1:0]		PKTCNT[9:0]										XFRSIZ[18:16]		
	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
XFRSIZ[15:0]															
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.

Bits 30:29 **RXDPID[1:0]:** Received data PID

Applies to isochronous OUT endpoints only.

This is the data PID received in the last packet for this endpoint.

00: DATA0

10: DATA1

STUPCNT[1:0]: SETUP packet count

Applies to control OUT endpoints only.

This field specifies the number of back-to-back SETUP data packets the endpoint can receive.

01: 1 packet

10: 2 packets

11: 3 packets

Bits 28:19 **PKTCNT[9:0]:** Packet count

Indicates the total number of USB packets that constitute the transfer size amount of data for this endpoint.

This field is decremented every time a packet (maximum size or short packet) is written to the Rx FIFO.

Bits 18:0 **XFRSIZ[18:0]:** Transfer size

This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet is read from the Rx FIFO and written to the external memory.

63.15.54 OTG power and clock gating control register (OTG_PCGCCTL)

Address offset: 0xE00

Reset value: 0x200B 8000

This register is available in host and device modes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	PHY SLEEP	ENL1 GTG	PHY SUSP	Res.	Res.	GATE HCLK	STPP CLK
								r	r	rw	r			rw	rw

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

Bit 7 **SUSP:** Deep Sleep

This bit indicates that the PHY is in Deep Sleep when in L1 state.

Bit 6 **PHYSLEEP:** PHY in Sleep

This bit indicates that the PHY is in the Sleep state.

Bit 5 **ENL1GTG:** Enable sleep clock gating

When this bit is set, core internal clock gating is enabled in Sleep state if the core cannot assert utmi_l1_suspend_n. When this bit is not set, the PHY clock is not gated in Sleep state.

Bit 4 **PHYSUSP:** PHY suspended

Indicates that the PHY has been suspended. This bit is updated once the PHY is suspended after the application has set the STPPCLK bit.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **GATEHCLK:** Gate HCLK

The application sets this bit to gate HCLK to modules other than the AHB Slave and Master and wakeup logic when the USB is suspended or the session is not valid. The application clears this bit when the USB is resumed or a new session starts.

Bit 0 **STPPCLK:** Stop PHY clock

The application sets this bit to stop the PHY clock when the USB is suspended, the session is not valid, or the device is disconnected. The application clears this bit when the USB is resumed or a new session starts.

63.15.55 OTG_FS register map

The table below gives the USB OTG register map and reset values.

Table 652. OTG_FS 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	OTG_GOTGCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CURMOD	OTGVER	BSVLD	ASVLD	DBCT	CIDSTS	Res.	Res.	Res.	EHEN	DHNPEN	HSHNPEN	HNPRQ	HNGSCS	BVALOVAL	BVALOEN	AVALOVAL	AVALOEN	VBVALOVA	VBVALOEN	SRQ	SRQSCS
	Reset value											0	0	0	0	0	1				0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	OTG_GOTGINT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBCDNE	ADTOCHG	HNGDET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HNSSCHG	SRSSCHG	Res.	Res.	Res.	Res.	Res.	SEDET	Res.	Res.
	Reset value													0	0	0							-	0	0						0		
0x008	OTG_GAHBCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTXFELVL	TXFELVL	Res.	Res.	Res.	Res.	Res.	Res.	GINTMSK
	Reset value																								0	0							0
0x00C	OTG_GUSBCFG	Res.	FDMOD	FHMOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRDT				HNPCAP	SRPCAP	Res.	PHYSEL	Res.	Res.	Res.	TOCAL		
	Reset value		0	0																0	1	0	1	0	0		1				0	0	0
0x010	OTG_GRSTCTL	AHBIDL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFNUM					TXFFLSH	RXFFLSH	Res.	FCRST	PSRST	CSRST
	Reset value	1																					0	0	0	0	0	0	0		0	0	0

Table 652. OTG_FS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x014	OTG_ GINTSTS	WKUPINT	SRQINT	DISCINT	CIDSCHG	LPMINT	PTXFE	HCINT	HPRTINT	RSTDET	Res.	IPXFR/INCOMPISOOUT	IISOIXFR	OEPINT	IEPINT	Res.	Res.	EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Res.	Res.	GONAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD
	Reset value	0	0	0	0	0	1	0	0	0		0	0	0	0			0	0	0	0	0	0			0	0	1	0	0	0	0	0
0x018	OTG_ GINTMSK	WUIM	SRQIM	DISCINT	CIDSCHGM	LPMINTM	PTXFEM	HCIM	PRTIM	RSTDETM	Res.	IPXFRM/IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	Res.	Res.	EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Res.	Res.	GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	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	
0x01C	OTG_ GRXSTSR (Device mode)	Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM				PKTSTS				DPID		BCNT											EPNUM			
	Reset value					0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	OTG_ GRXSTSR (Host mode)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS				DPID		BCNT											CHNUM			
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	OTG_ GRXSTSP (Device mode)	Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM				PKTSTS				DPID		BCNT											EPNUM			
	Reset value					0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	OTG_ GRXSTSP (Host mode)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS				DPID		BCNT											CHNUM			
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	OTG_ GRXFSIZ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFD															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x028	OTG_ HNPTXFSIZ/ OTG_ DIEPTXF0	NPTXFD/TX0FD																NPTXFSA/TX0FSA															
	Reset value	0	0	0	0	0	0	1	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
0x02C	OTG_ HNPTXSTS	Res.	NPTXQTOP							NPTQXSAV								NPTXFSAV															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x038	OTG_ GCCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBDEN	SDEN	PDEN	DCDEN	BCDEN	PWRDWN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS2DET	SDET	PDET	DCDET
	Reset value											0	0	0	0	0	0													X	X	X	X

Table 652. OTG_FS 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
0x03C	OTG_CID	PRODUCT_ID																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0x054	OTG_ GLPMCFG	Res.	Res.	Res.	ENBESL	LPMR CNTSTS			SNDLPM	LPM RCNT			LPMCHIDX				L1RSMOK	SLPSTS	LPM RSP		L1DSEN	BESLTHRS				L1SSEN	REMWAKE	BESL				LPMACK	LPMEN
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100	OTG_ HPTXFSIZ	PTXFSIZ																PTXSA															
	Reset value	0	0	0	0	0	0	1	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
0x104	OTG_ DIEPTXF1	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	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
0x108	OTG_ DIEPTXF2	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
. . . .	. . . .	. . . .																															
0x114	OTG_ DIEPTXF5	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
0x400	OTG_ HCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSLSS	FSLS PCS	
	Reset value																														0	0	0
0x404	OTG_ HFIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLDCTRL	FRIVL															
	Reset value																0	1	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0
0x408	OTG_ HFNUM	FTREM																FRNUM															
	Reset value	0	0	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
0x410	OTG_ HPTXSTS	PTXQTOP								PTXQSAV								PTXFSAVL															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x414	OTG_ HAINT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HAINT															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x418	OTG_ HAINTMSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HAINTM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x440	OTG_ HPRT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSPD	PTCTL			PPWR	PLSTS		Res.	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS		
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 652. OTG_FS 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
0x500	**OTG_ HCCHAR0**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Res.	EPDIR	EPNUM				MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x508	**OTG_ HCINT0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x508	**OTG_ HCINT0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x50C	**OTG_ HCINTMSK0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERRM	FRMORM	BBERRM	TXERRM		ACKM	NAKM	STALLM	Res.	CHHM	XFRCM
	Reset value																						0	0	0	0		0	0	0		0	0
0x510	**OTG_ HCTSIZ0**	DOPNG	DPID		PKTCNT											XFRSIZ																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x520	**OTG_ HCCHAR1**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Res.	EPDIR	EPNUM				MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x528	**OTG_ HCINT1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x528	**OTG_ HCINT1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x52C	**OTG_ HCINTMSK1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERRM	FRMORM	BBERRM	TXERRM		ACKM	NAKM	STALLM	Res.	CHHM	XFRCM
	Reset value																						0	0	0	0		0	0	0		0	0
0x530	**OTG_ HCTSIZ1**	DOPNG	DPID		PKTCNT											XFRSIZ																	
	Reset value	0	0	0	0	0	0	0	0	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 652. OTG_FS 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
0x660	**OTG_ HCCHAR11**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Res.	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x668	**OTG_ HCINT11**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x66C	**OTG_ HCINTMSK11**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERRM	FRMORM	BBERRM	TXERRM		ACKM	NAKM	STALLM	Res.	CHHM	XFRCM
	Reset value																						0	0	0	0		0	0	0		0	0
0x670	**OTG_ HCTSIZ11**	DOPNG	DPID		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x800	**OTG_ DCFG**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRATIM	XCVRDLY	Res.	PFIVL		DAD							Res.	NZLSOHSK	DSPD	
	Reset value																	0	0		0	0	0	0	0	0	0	0	0		0	0	0
0x804	**OTG_ DCTL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSBESLRJCT	Res.	Res.	Res.	Res.	Res.	Res.	POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL			GONSTS	GINSTS	SDIS	RWUSIG
	Reset value														0							0	0	0	0	0	0	0	0	0	0	1	0
0x808	**OTG_ DSTS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEV LN STS		FNSOF											Res.	Res.	Res.	Res.				EERR	ENUMSPD		SUSPSTS
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0
0x810	**OTG_ DIEPMSK**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAKM	Res.	Res.	Res.	Res.	Res.	Res.	INEPNEM	INEPNMM	ITTXFEMSK	TOM	Res.	EPDM	XFRCM
	Reset value																			0							0	0	0	0		0	0
0x814	**OTG_ DOEPMSK**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAKMSK	BERRM	Res.	Res.	Res.	OUTPKTERRM	Res.	Res.	Res.	OTEPDM	STUPM	Res.	EPDM	XFRCM
	Reset value																			0	0				0				0	0		0	0
0x818	**OTG_ DAINT**	OEPINT																IEPINT															
	Reset value	0	0	0	0	0	0	0	0	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 652. OTG_FS 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
0x81C	**OTG_ DAINTMSK**	OEPM																IEPM															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x828	**OTG_ DVBUSDIS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBUSDT															
	Reset value																	0	0	0	1	0	1	1	1	1	1	0	1	0	1	1	1
0x82C	**OTG_DVB USPULSE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DVBUSP															
	Reset value																	0	0	0	0	0	1	0	1	1	0	1	1	1	0	0	0
0x834	**OTG_DIE PEMPMSK**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTXFEM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x900	**OTG_ DIEPCTL0**	EPENA	EPDIS	Res.	Res.	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	Res.	USBAEP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPSIZ	
	Reset value	0	0			0	0	0	0	0	0	0		0	0	0		1														0	0
0x908	**OTG_ DIEPINT0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAK	Res.	PKTDRPSTS	Res.	Res.	Res.	TXFE	INEPNE	INEPNM	ITTXFE	TOC	Res.	EPDISD	XFRC
	Reset value																			0		0				1	0	0	0	0		0	0
0x910	**OTG_ DIEPTSIZ0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKT CNT		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ						
	Reset value												0	0													0	0	0	0	0	0	0
0x918	**OTG_ DTXFSTS0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x920	**OTG_ DIEPCTL1**	EPENA	EPDIS	SODDFRM/SD1PID	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x928	**OTG_ DIEPINT1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAK	Res.	PKTDRPSTS	Res.	Res.	Res.	TXFE	INEPNE	INEPNM	ITTXFE	TOC	Res.	EPDISD	XFRC
	Reset value																			0		0				1	0	0	0	0		0	0
0x930	**OTG_ DIEPTSIZ1**	Res.	MCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x938	**OTG_ DTXFSTS1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Table 652. OTG_FS 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
0x940	OTG_ DIEPCTL2	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
. . . .	. . . .	.																															
0x9A0	OTG_ DIEPCTL5	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
. . . .	. . . .	.																															
0x9A8	OTG_ DIEPINT5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAK	Res.	PKTDRPSTS	Res.	Res.	Res.	TXFE	INEPNE	INEPNM	ITTXFE	TOC	Res.	EPDISD	XFRC
	Reset value																			0		0				1	0	0	0	0		0	0
. . . .	. . . .	.																															
0x9B8	OTG_ DTXFSTS5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
. . . .	. . . .	.																															
0x9B0	OTG_ DIEPTSIZ5	Res.	MCNT		PKTCNT											XFRSIZ																	
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB00	OTG_ DOEPCTL0	EPENA	EPDIS	Res.	Res.	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP		NAKSTS	Res.	USBAEP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPSIZ	
	Reset value	0	0			0	0					0	0	0	0	0		1														0	0
0xB08	OTG_ DOEPINT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAK	BERR	Res.	Res.	Res.	Res.	Res.	Res.	STSPHSRX	OTEPDIS	STUP	Res.	EPDISD	XFRC
	Reset value																			0	0							0	0	0		0	0

Table 652. OTG_FS 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
0xB10	**OTG_ DOEPTSIZ0**	Res.	STUP CNT		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTCNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ						
	Reset value		0	0										0													0	0	0	0	0	0	0
0xB20	**OTG_ DOEPCTL1**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EP TYP		NAKSTS	EONUM/DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xB28	**OTG_ DOEPINT1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAK	BERR	Res.	Res.	Res.	Res.	Res.	Res.	STSPHSRX	OTEPDIS	STUP	Res.	EPDISD	XFRC
	Reset value																			0	0							0	0	0		0	0
0xB30	**OTG_ DOEPTSIZ1**	Res.	RXDPID/ STUPCNT		PKTCNT											XFRSIZ																	
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
. . . .	. . . .	.																															
0xBA0	**OTG_ DOEPCTL5**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xBA8	**OTG_ DOEPINT5**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAK	BERR	Res.	Res.	Res.	Res.	Res.	Res.	STSPHSRX	OTEPDIS	STUP	Res.	EPDISD	XFRC
	Reset value																			0	0					-		0	0	0		0	0
0xBB0	**OTG_ DOEPTSIZ5**	Res.	RXDPID/ STUPCNT		PKTCNT											XFRSIZ																	
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE00	**OTG_ PCGCCTL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	PHYSLEEP	ENL1GTG	PHYSUSP	Res.	Res.	GATEHCLK	STPPCLK
	Reset value																									0	0	0	0			0	0

Refer to *Section 2.3 on page 115* for the register boundary addresses.

63.16 OTG_FS programming model

63.16.1 Core initialization

The application must perform the core initialization sequence. If the cable is connected during power-up, the current mode of operation bit in the OTG_GINTSTS (CMOD bit in OTG_GINTSTS) reflects the mode. The OTG_FS controller enters host mode when an “A” plug is connected or device mode when a “B” plug is connected.

This section explains the initialization of the OTG_FS controller after power-on. The application must follow the initialization sequence irrespective of host or device mode operation. All core global registers are initialized according to the core’s configuration:

1. Program the following fields in the OTG_GAHBCFG register:
   - Global interrupt mask bit GINTMSK = 1
   - Rx FIFO non-empty (RXFLVL bit in OTG_GINTSTS)
   - Periodic Tx FIFO empty level
2. Program the following fields in the OTG_GUSBCFG register:
   - HNP capable bit
   - SRP capable bit
   - OTG_FS timeout calibration field
   - USB turnaround time field
3. The software must unmask the following bits in the OTG_GINTMSK register:

   OTG interrupt mask

   Mode mismatch interrupt mask
4. The software can read the CMOD bit in OTG_GINTSTS to determine whether the OTG_FS controller is operating in host or device mode.

63.16.2 Host initialization

To initialize the core as host, the application must perform the following steps:

1. Program the HPRTINT in the OTG_GINTMSK register to unmask
2. Program the OTG_HCFG register to select full-speed host
3. Program the PPWR bit in OTG_HPRT to 1. This drives V_{BUS} on the USB.
4. Wait for the PCDET interrupt in OTG_HPRT0. This indicates that a device is connecting to the port.
5. Program the PRST bit in OTG_HPRT to 1. This starts the reset process.
6. Wait at least 10 ms for the reset process to complete.
7. Program the PRST bit in OTG_HPRT to 0.
8. Wait for the PENCHNG interrupt in OTG_HPRT.
9. Read the PSPD bit in OTG_HPRT to get the enumerated speed.
10. Program the HFIR register with a value corresponding to the selected PHY clock 1
11. Program the FSLSPCS field in the OTG_HCFG register following the speed of the device detected in step 9. If FSLSPCS has been changed a port reset must be performed.
12. Program the OTG_GRXFSIZ register to select the size of the receive FIFO.
13. Program the OTG_HNPTXFSIZ register to select the size and the start address of the Non-periodic transmit FIFO for non-periodic transactions.
14. Program the OTG_HPTXFSIZ register to select the size and start address of the periodic transmit FIFO for periodic transactions.

To communicate with devices, the system software must initialize and enable at least one channel.

63.16.3 Device initialization

The application must perform the following steps to initialize the core as a device on power-up or after a mode change from host to device.

1. Program the following fields in the OTG_DCFG register:
   - Device speed
   - Non-zero-length status OUT handshake
   - Periodic Frame Interval
2. Clear the DCTL.SDIS bit. The core issues a connect after this bit is cleared.
3. Program the OTG_GINTMSK register to unmask the following interrupts:
   - USB reset
   - Enumeration done
   - Early suspend
   - USB suspend
   - SOF
4. Wait for the USBRST interrupt in OTG_GINTSTS. It indicates that a reset has been detected on the USB that lasts for about 10 ms on receiving this interrupt.
5. Wait for the ENUMDNE interrupt in OTG_GINTSTS. This interrupt indicates the end of reset on the USB. On receiving this interrupt, the application must read the OTG_DSTS

register to determine the enumeration speed and perform the steps listed in *Endpoint initialization on enumeration completion on page 2792*.

At this point, the device is ready to accept SOF packets and perform control transfers on control endpoint 0.

63.16.4 Host programming model

Channel initialization

The application must initialize one or more channels before it can communicate with connected devices. To initialize and enable a channel, the application must perform the following steps:

1. Program the OTG_GINTMSK register to unmask the following:
2. Channel interrupt
   - Non-periodic transmit FIFO empty for OUT transactions (applicable when operating in pipelined transaction-level with the packet count field programmed with more than one).
   - Non-periodic transmit FIFO half-empty for OUT transactions (applicable when operating in pipelined transaction-level with the packet count field programmed with more than one).
3. Program the OTG_HAINTMSK register to unmask the selected channels' interrupts.
4. Program the OTG_HCINTMSK register to unmask the transaction-related interrupts of interest given in the host channel interrupt register.
5. Program the selected channel's OTG_HCTSIZx register with the total transfer size, in bytes, and the expected number of packets, including short packets. The application must program the PID field with the initial data PID (to be used on the first OUT transaction or to be expected from the first IN transaction).
6. Program the OTG_HCCHARx register of the selected channel with the device's endpoint characteristics, such as type, speed, direction, and so forth. (The channel can be enabled by setting the channel enable bit to 1 only when the application is ready to transmit or receive any packet).

Halting a channel

The application can disable any channel by programming the OTG_HCCHARx register with the CHDIS and CHENA bits set to 1. This enables the OTG_FS host to flush the posted requests (if any) and generates a channel halted interrupt. The application must wait for the CHH interrupt in OTG_HCINTx before reallocating the channel for other transactions. The OTG_FS host does not interrupt the transaction that has already been started on the USB.

Before disabling a channel, the application must ensure that there is at least one free space available in the non-periodic request queue (when disabling a non-periodic channel) or the periodic request queue (when disabling a periodic channel). The application can simply flush the posted requests when the request queue is full (before disabling the channel), by programming the OTG_HCCHARx register with the CHDIS bit set to 1 which automatically clears the CHENA bit to 0.

The application is expected to disable a channel on any of the following conditions:

1. When an STALL, TXERR, BBERR or DTERR interrupt in OTG_HCINTx is received for an IN or OUT channel. The application must be able to receive other interrupts (DTERR, Nak, data, TXERR) for the same channel before receiving the halt.
2. When a DISCINT (disconnect device) interrupt in OTG_GINTSTS is received. (The application is expected to disable all enabled channels).
3. When the application aborts a transfer before normal completion.

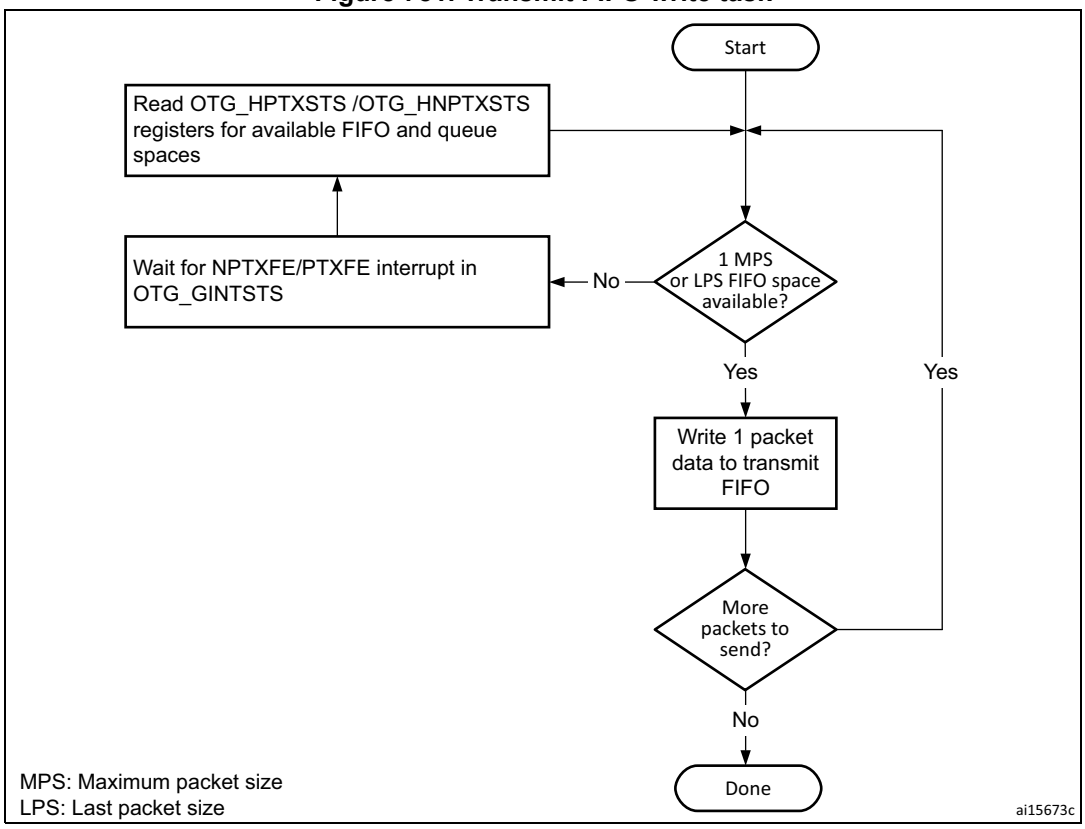
Operational model

The application must initialize a channel before communicating to the connected device. This section explains the sequence of operation to be performed for different types of USB transactions.

- **Writing the transmit FIFO**

  The OTG_FS host automatically writes an entry (OUT request) to the periodic/non-periodic request queue, along with the last 32-bit word write of a packet. The application must ensure that at least one free space is available in the periodic/non-periodic request queue before starting to write to the transmit FIFO. The application must always write to the transmit FIFO in 32-bit words. If the packet size is non-32-bit word aligned, the application must use padding. The OTG_FS host determines the actual packet size based on the programmed maximum packet size and transfer size.

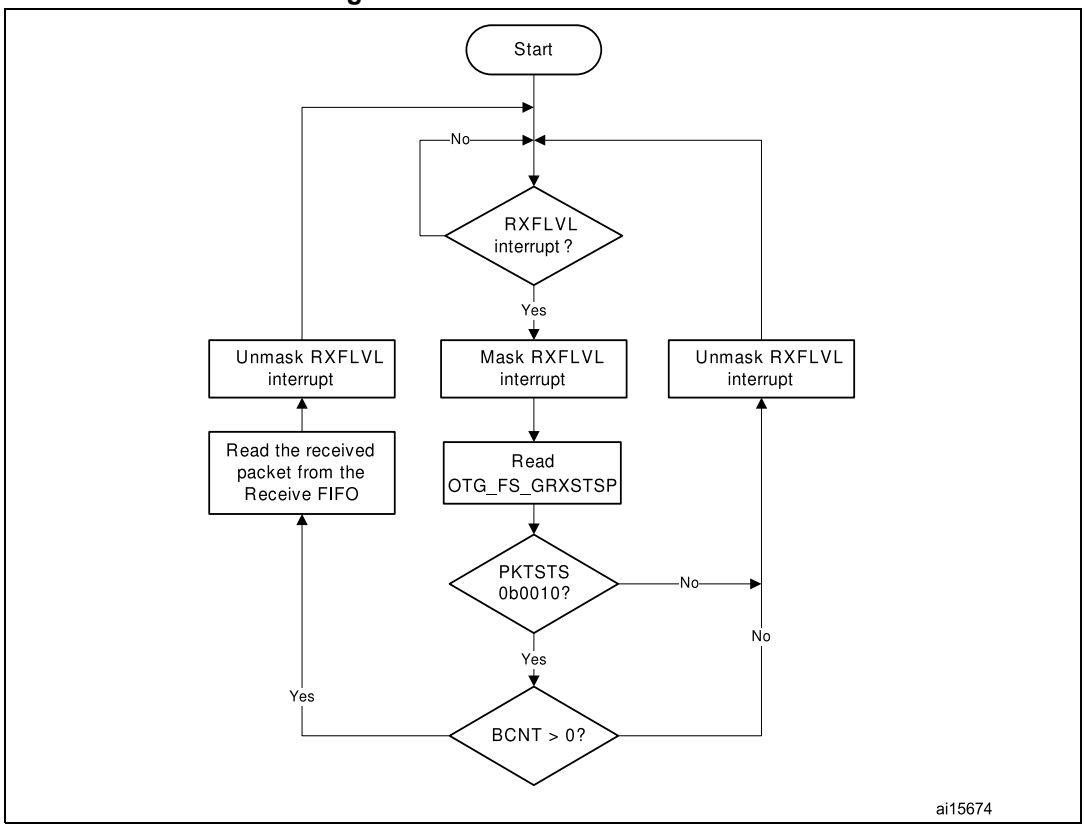
Figure 781. Transmit FIFO write task



- **Reading the receive FIFO**

  The application must ignore all packet statuses other than IN data packet (bx0010).

Figure 782. Receive FIFO read task



- **Bulk and control OUT/SETUP transactions**

  A typical bulk or control OUT/SETUP pipelined transaction-level operation is shown in *Figure 783*. See channel 1 (ch_1). Two bulk OUT packets are transmitted. A control SETUP transaction operates in the same way but has only one packet. The assumptions are:

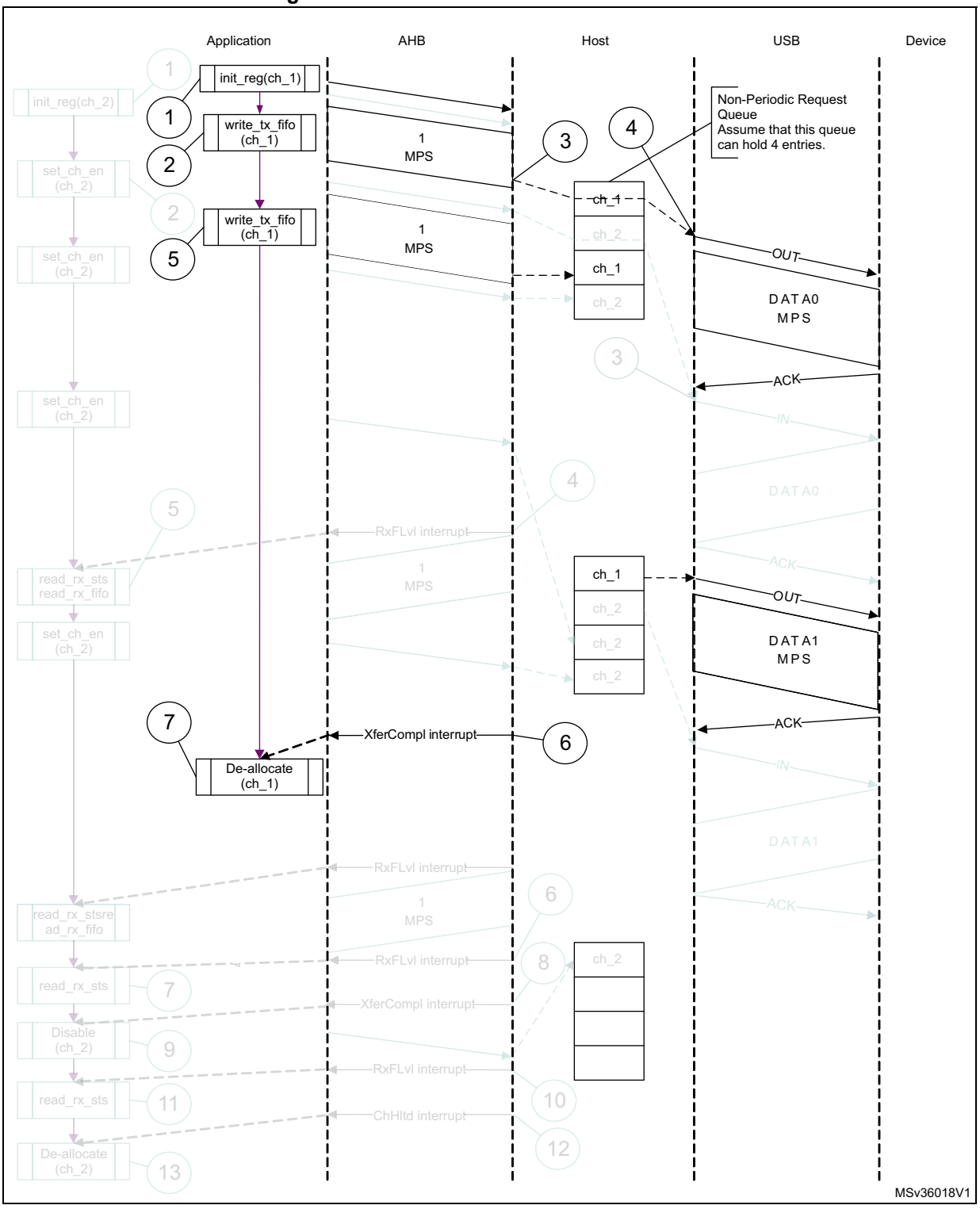
  – The application is attempting to send two maximum-packet-size packets (transfer size = 1, 024 bytes).
  – The non-periodic transmit FIFO can hold two packets (128 bytes for FS).
  – The non-periodic request queue depth = 4.

- **Normal bulk and control OUT/SETUP operations**

  The sequence of operations in (channel 1) is as follows:

1. Initialize channel 1
2. Write the first packet for channel 1
3. Along with the last word write, the core writes an entry to the non-periodic request queue
4. As soon as the non-periodic queue becomes non-empty, the core attempts to send an OUT token in the current frame
5. Write the second (last) packet for channel 1
6. The core generates the XFRC interrupt as soon as the last transaction is completed successfully
7. In response to the XFRC interrupt, de-allocate the channel for other transfers
8. Handling non-ACK responses

Figure 783. Normal bulk/control OUT/SETUP



1. The grayed elements are not relevant in the context of this figure.

The channel-specific interrupt service routine for bulk and control OUT/SETUP transactions is shown in the following code samples.

- **Interrupt service routine for bulk/control OUT/SETUP and bulk/control IN transactions**

  a) Bulk/control OUT/SETUP

```
Unmask (NAK/TXERR/STALL/XFRC)
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   De-allocate Channel
   }
else if (STALL)
   {
   Transfer Done = 1
   Unmask CHH
   Disable Channel
   }
else if (NAK or TXERR )
   {
   Rewind Buffer Pointers
   Unmask CHH
   Disable Channel
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      }
   else
      {
      Reset Error Count
      }
      }
else if (CHH)
   {
   Mask CHH
   if (Transfer Done or (Error_count == 3))
      {
      De-allocate Channel
      }
   else
      {
      Re-initialize Channel
      }
   }
```

```
   else if (ACK)
      {
      Reset Error Count
      Mask ACK
      }
```

The application is expected to write the data packets into the transmit FIFO when the space is available in the transmit FIFO and the request queue. The application can make use of the NPTXFE interrupt in OTG_GINTSTS to find the transmit FIFO space.

b) Bulk/control IN

```
Unmask (TXERR/XFRC/BBERR/STALL/DTERR)
if (XFRC)
   {
   Reset Error Count
   Unmask CHH
   Disable Channel
   Reset Error Count
   Mask ACK
   }
else if (TXERR or BBERR or STALL)
   {
   Unmask CHH
   Disable Channel
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      }
   }
else if (CHH)
   {
   Mask CHH
   if (Transfer Done or (Error_count == 3))
      {
      De-allocate Channel
      }
   else
      {
      Re-initialize Channel
      }
   }
else if (ACK)
   {
   Reset Error Count
   Mask ACK
   }
```

```
else if (DTERR)
   {
   Reset Error Count
   }
```

The application is expected to write the requests as and when the request queue space is available and until the XFRC interrupt is received.

- **Bulk and control IN transactions**

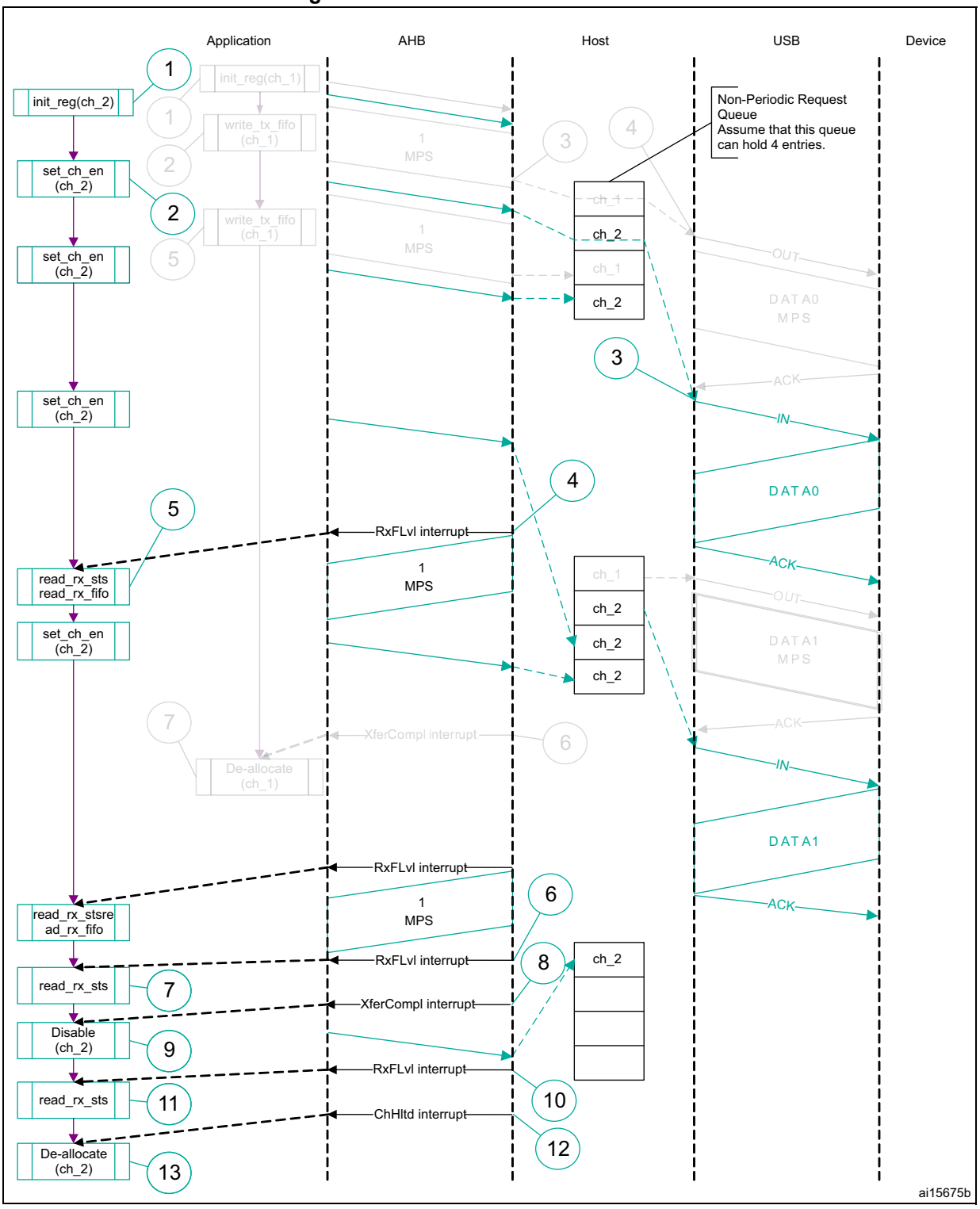
  A typical bulk or control IN pipelined transaction-level operation is shown in *Figure 784*. See channel 2 (ch_2). The assumptions are:

  – The application is attempting to receive two maximum-packet-size packets (transfer size = 1 024 bytes).

  – The receive FIFO can contain at least one maximum-packet-size packet and two status words per packet (72 bytes for FS).

  – The non-periodic request queue depth = 4.

Figure 784. Bulk/control IN transactions



1. The grayed elements are not relevant in the context of this figure.

The sequence of operations is as follows:

1. Initialize channel 2.
2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the non-periodic request queue.
3. The core attempts to send an IN token after completing the current OUT transaction.
4. The core generates an RXFLVL interrupt as soon as the received packet is written to the receive FIFO.
5. In response to the RXFLVL interrupt, mask the RXFLVL interrupt and read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. Following this, unmask the RXFLVL interrupt.
6. The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO.
7. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in OTG_GRXSTSR ≠ 0b0010).
8. The core generates the XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, disable the channel and stop writing the OTG_HCCHAR2 register for further requests. The core writes a channel disable request to the non-periodic request queue as soon as the OTG_HCCHAR2 register is written.
10. The core generates the RXFLVL interrupt as soon as the halt status is written to the receive FIFO.
11. Read and ignore the receive packet status.
12. The core generates a CHH interrupt as soon as the halt status is popped from the receive FIFO.
13. In response to the CHH interrupt, de-allocate the channel for other transfers.
14. Handling non-ACK responses

- **Control transactions**

  Setup, data, and status stages of a control transfer must be performed as three separate transfers. setup-, data- or status-stage OUT transactions are performed similarly to the bulk OUT transactions explained previously. Data- or status-stage IN transactions are performed similarly to the bulk IN transactions explained previously. For all three stages, the application is expected to set the EPTYP field in

OTG_HCCHAR1 to control. During the setup stage, the application is expected to set the PID field in OTG_HCTSIZ1 to SETUP.

- **Interrupt OUT transactions**

  A typical interrupt OUT operation is shown in *Figure 785*. The assumptions are:

  – The application is attempting to send one packet in every frame (up to 1 maximum packet size), starting with the odd frame (transfer size = 1 024 bytes)

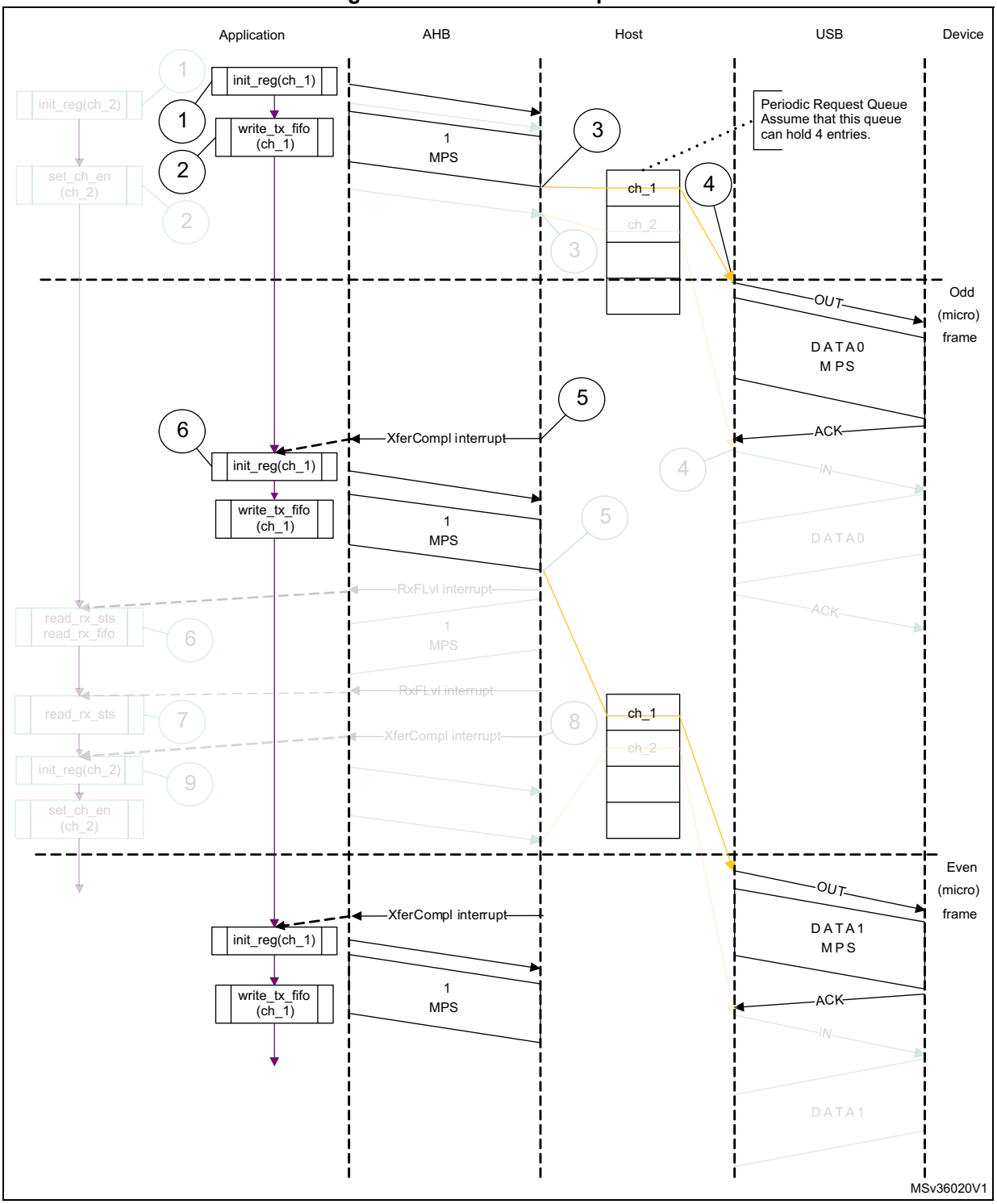
  – The periodic transmit FIFO can hold one packet (1 Kbyte)

  – Periodic request queue depth = 4

  The sequence of operations is as follows:

1. Initialize and enable channel 1. The application must set the ODDFRM bit in OTG_HCCHAR1.
2. Write the first packet for channel 1.
3. Along with the last word write of each packet, the OTG_FS host writes an entry to the periodic request queue.
4. The OTG_FS host attempts to send an OUT token in the next (odd) frame.
5. The OTG_FS host generates an XFRC interrupt as soon as the last packet is transmitted successfully.
6. In response to the XFRC interrupt, reinitialize the channel for the next transfer.

Figure 785. Normal interrupt OUT



1. The grayed elements are not relevant in the context of this figure.

- **Interrupt service routine for interrupt OUT/IN transactions**

  a) Interrupt OUT

`Unmask (NAK/TXERR/STALL/XFRC/FRMOR)`

```
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   De-allocate Channel
   }
else
   if (STALL or FRMOR)
      {
      Mask ACK
      Unmask CHH
      Disable Channel
      if (STALL)
         {
         Transfer Done = 1
         }
      }
   else
      if (NAK or TXERR)
         {
         Rewind Buffer Pointers
         Reset Error Count
         Mask ACK
         Unmask CHH
         Disable Channel
         }
      else
         if (CHH)
         {
         Mask CHH
         if (Transfer Done or (Error_count == 3))
            {
            De-allocate Channel
            }
         else
            {
            Re-initialize Channel (in next b_interval - 1 Frame)
            }
         }
      else
         if (ACK)
         {
         Reset Error Count
         Mask ACK
         }
```

```
The application uses the NPTXFE interrupt in OTG_GINTSTS to find the
transmit FIFO space.
Interrupt IN
Unmask (NAK/TXERR/XFRC/BBERR/STALL/FRMOR/DTERR)
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   if (OTG_HCTSIZx.PKTCNT == 0)
      {
      De-allocate Channel
      }
   else
      {
      Transfer Done = 1
      Unmask CHH
      Disable Channel
      }
   }
else
   if (STALL or FRMOR or NAK or DTERR or BBERR)
      {
      Mask ACK
      Unmask CHH
      Disable Channel
      if (STALL or BBERR)
         {
         Reset Error Count
         Transfer Done = 1
         }
      else
         if (!FRMOR)
         {
         Reset Error Count
         }
      }
else
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      Unmask CHH
      Disable Channel
      }
else
```

```
   if (CHH)
      {
      Mask CHH
      if (Transfer Done or (Error_count == 3))
         {
         De-allocate Channel
         }
      else
         Re-initialize Channel (in next b_interval - 1 /Frame)
      }
   }
else
   if (ACK)
      {
      Reset Error Count
      Mask ACK
      }
```

- **Interrupt IN transactions**

  The assumptions are:

  – The application is attempting to receive one packet (up to 1 maximum packet size) in every frame, starting with odd (transfer size = 1 024 bytes).

  – The receive FIFO can hold at least one maximum-packet-size packet and two status words per packet (1 031 bytes).

  – Periodic request queue depth = 4.

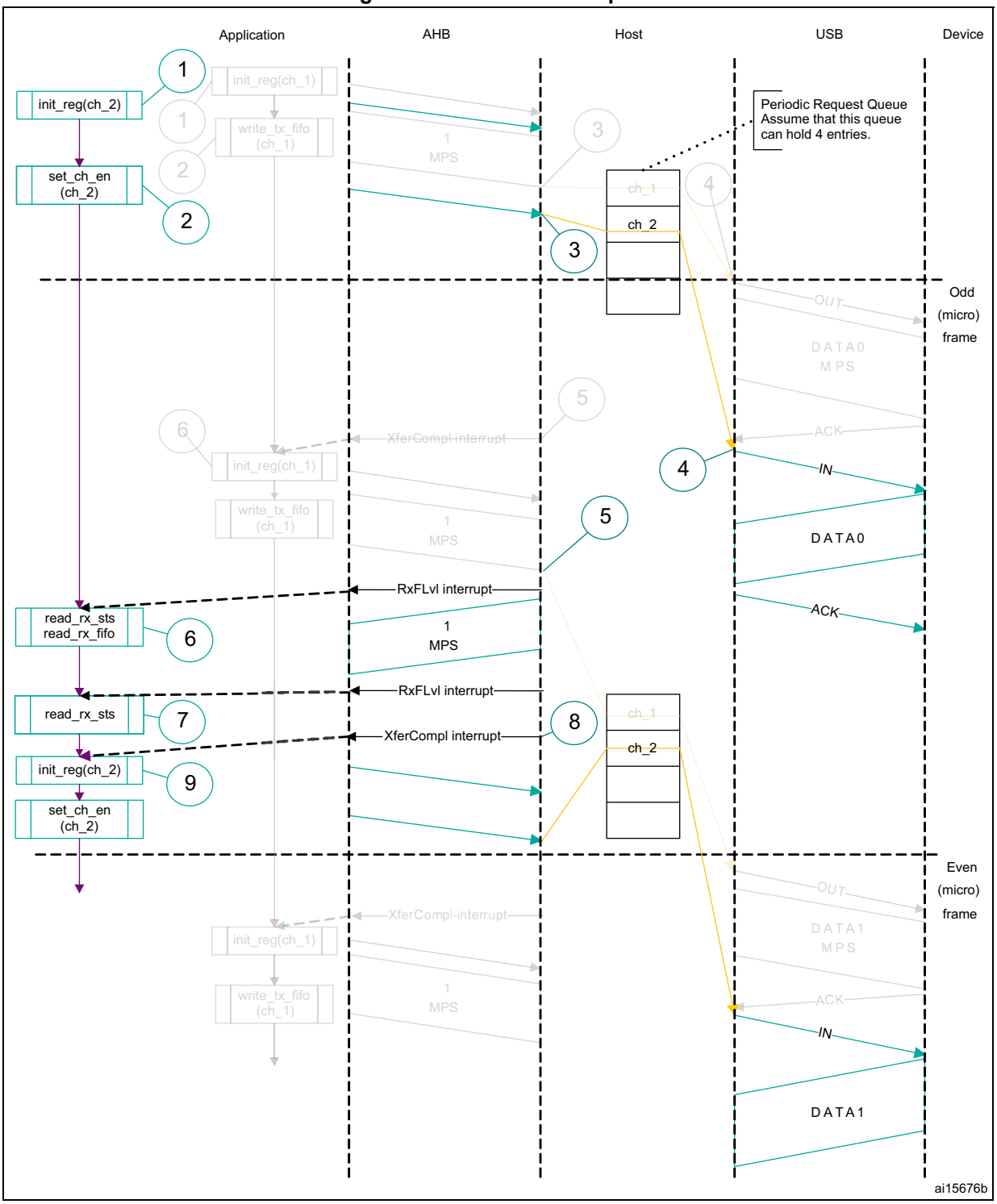
- **Normal interrupt IN operation**

  The sequence of operations is as follows:

1. Initialize channel 2. The application must set the ODDFRM bit in OTG_HCCHAR2.
2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the periodic request queue.
3. The OTG_FS host writes an IN request to the periodic request queue for each OTG_HCCHAR2 register write with the CHENA bit set.
4. The OTG_FS host attempts to send an IN token in the next (odd) frame.
5. As soon as the IN packet is received and written to the receive FIFO, the OTG_FS host generates an RXFLVL interrupt.
6. In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask after reading the entire packet.
7. The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in GRXSTSR ≠ 0b0010).
8. The core generates an XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, read the PKTCNT field in OTG_HCTSIZ2. If the PKTCNT bit in OTG_HCTSIZ2 is not equal to 0, disable the channel before re-

initializing the channel for the next transfer, if any). If PKTCNT bit in OTG_HCTSIZ2 = 0, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG_HCCHAR2.

Figure 786. Normal interrupt IN



1. The grayed elements are not relevant in the context of this figure.

- **Isochronous OUT transactions**

  A typical isochronous OUT operation is shown in *Figure 787*. The assumptions are:

  – The application is attempting to send one packet every frame (up to 1 maximum

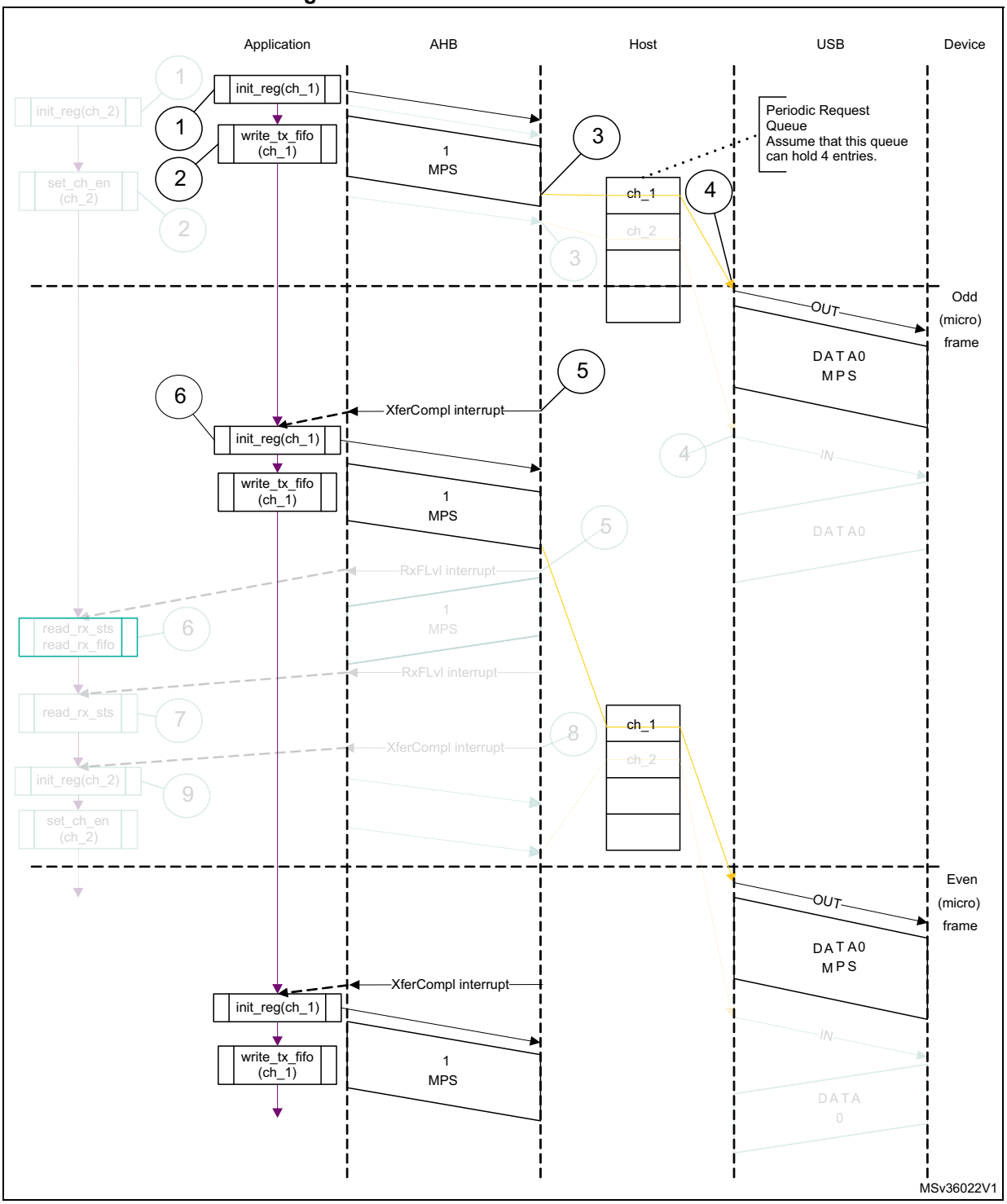
packet size), starting with an odd frame. (transfer size = 1 024 bytes).

– The periodic transmit FIFO can hold one packet (1 Kbyte).
– Periodic request queue depth = 4.

The sequence of operations is as follows:

1. Initialize and enable channel 1. The application must set the ODDFRM bit in OTG_HCCHAR1.
2. Write the first packet for channel 1.
3. Along with the last word write of each packet, the OTG_FS host writes an entry to the periodic request queue.
4. The OTG_FS host attempts to send the OUT token in the next frame (odd).
5. The OTG_FS host generates the XFRC interrupt as soon as the last packet is transmitted successfully.
6. In response to the XFRC interrupt, reinitialize the channel for the next transfer.
7. Handling non-ACK responses

Figure 787. Isochronous OUT transactions



1. The grayed elements are not relevant in the context of this figure.

- **Interrupt service routine for isochronous OUT/IN transactions**

  Code sample: isochronous OUT

```
Unmask (FRMOR/XFRC)
if (XFRC)
```

```
   {
   De-allocate Channel
   }
else
   if (FRMOR)
      {
      Unmask CHH
      Disable Channel
      }
   else
   if (CHH)
      {
      Mask CHH
      De-allocate Channel
      }
Code sample: Isochronous IN
Unmask (TXERR/XFRC/FRMOR/BBERR)
if (XFRC or FRMOR)
   {
   if (XFRC and (OTG_HCTSIZx.PKTCNT == 0))
      {
      Reset Error Count
      De-allocate Channel
      }
   else
      {
      Unmask CHH
      Disable Channel
      }
   }
   else
      if (TXERR or BBERR)
         {
         Increment Error Count
         Unmask CHH
         Disable Channel
         }
      else
         if (CHH)
            {
            Mask CHH
            if (Transfer Done or (Error_count == 3))
               {
               De-allocate Channel
               }
```

```
else
   {
   Re-initialize Channel
   }
}
```

- **Isochronous IN transactions**

  The assumptions are:

  – The application is attempting to receive one packet (up to 1 maximum packet size) in every frame starting with the next odd frame (transfer size = 1 024 bytes).

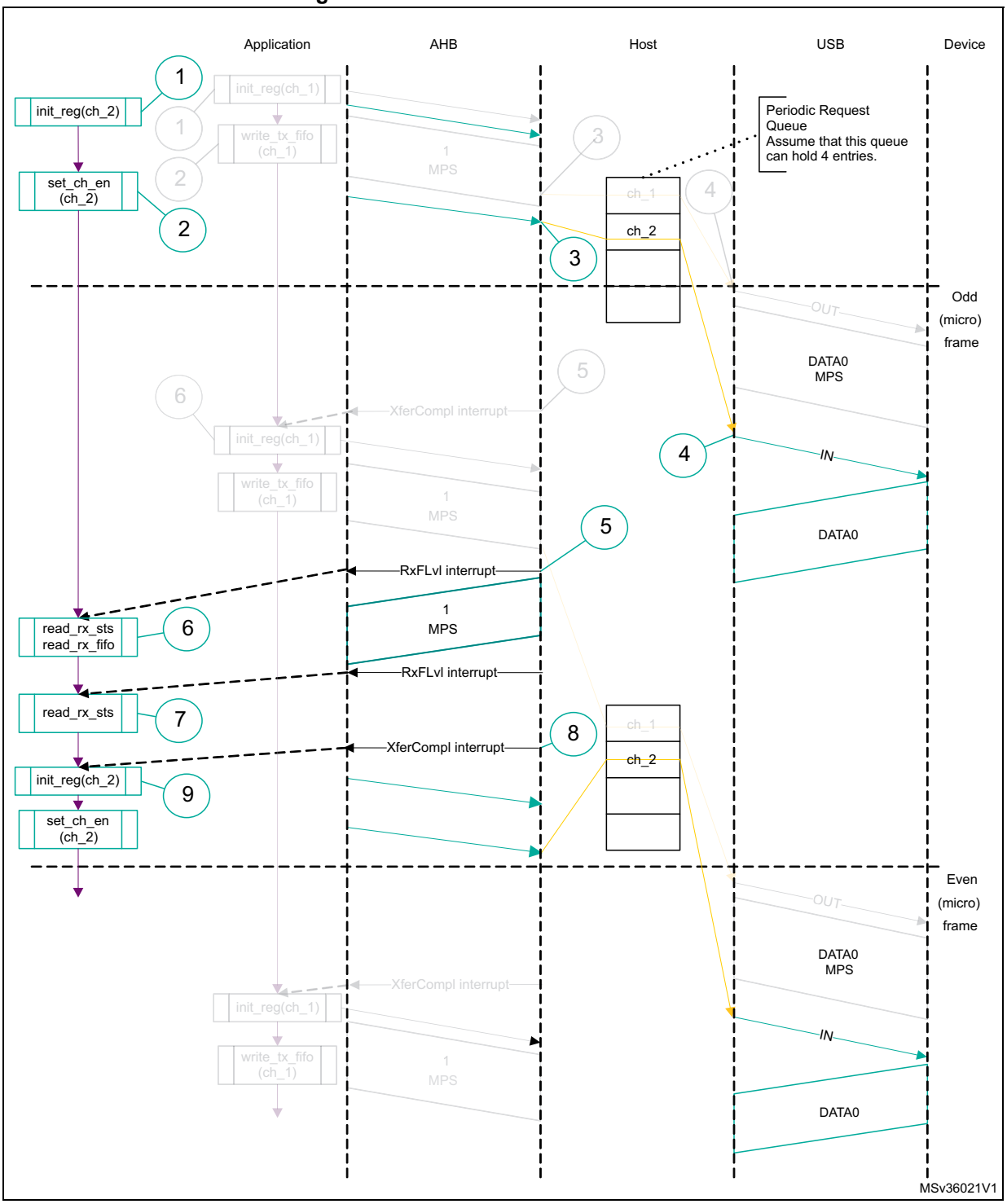
  – The receive FIFO can hold at least one maximum-packet-size packet and two status word per packet (1 031 bytes).

  – Periodic request queue depth = 4.

  The sequence of operations is as follows:

1. Initialize channel 2. The application must set the ODDFRM bit in OTG_HCCHAR2.
2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the periodic request queue.
3. The OTG_FS host writes an IN request to the periodic request queue for each OTG_HCCHAR2 register write with the CHENA bit set.
4. The OTG_FS host attempts to send an IN token in the next odd frame.
5. As soon as the IN packet is received and written to the receive FIFO, the OTG_FS host generates an RXFLVL interrupt.
6. In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask it after reading the entire packet.
7. The core generates an RXFLVL interrupt for the transfer completion status entry in the receive FIFO. This time, the application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS bit in OTG_GRXSTSR ≠ 0b0010).
8. The core generates an XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, read the PKTCNT field in OTG_HCTSIZ2. If PKTCNT ≠ 0 in OTG_HCTSIZ2, disable the channel before re-initializing the channel for the next transfer, if any. If PKTCNT = 0 in OTG_HCTSIZ2, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG_HCCHAR2.

Figure 788. Isochronous IN transactions



1. The grayed elements are not relevant in the context of this figure.

- **Selecting the queue depth**

  Choose the periodic and non-periodic request queue depths carefully to match the number of periodic/non-periodic endpoints accessed.

  The non-periodic request queue depth affects the performance of non-periodic

transfers. The deeper the queue (along with sufficient FIFO size), the more often the core is able to pipeline non-periodic transfers. If the queue size is small, the core is able to put in new requests only when the queue space is freed up.

The core's periodic request queue depth is critical to perform periodic transfers as scheduled. Select the periodic queue depth, based on the number of periodic transfers scheduled in a microframe. If the periodic request queue depth is smaller than the periodic transfers scheduled in a microframe, a frame overrun condition occurs.

- **Handling babble conditions**

  OTG_FS controller handles two cases of babble: packet babble and port babble. Packet babble occurs if the device sends more data than the maximum packet size for the channel. Port babble occurs if the core continues to receive data from the device at EOF2 (the end of frame 2, which is very close to SOF).

  When OTG_FS controller detects a packet babble, it stops writing data into the Rx buffer and waits for the end of packet (EOP). When it detects an EOP, it flushes already written data in the Rx buffer and generates a Babble interrupt to the application.

  When OTG_FS controller detects a port babble, it flushes the Rx FIFO and disables the port. The core then generates a port disabled interrupt (HPRTINT in OTG_GINTSTS, PENCHNG in OTG_HPRT). On receiving this interrupt, the application must determine that this is not due to an overcurrent condition (another cause of the port disabled interrupt) by checking POCA in OTG_HPRT, then perform a soft reset. The core does not send any more tokens after it has detected a port babble condition.

63.16.5 Device programming model

Endpoint initialization on USB reset

1. Set the NAK bit for all OUT endpoints
   - SNAK = 1 in OTG_DOEPCTLx (for all OUT endpoints)
2. Unmask the following interrupt bits
   - INEP0 = 1 in OTG_DAINTMSK (control 0 IN endpoint)
   - OUTEP0 = 1 in OTG_DAINTMSK (control 0 OUT endpoint)
   - STUPM = 1 in OTG_DOEPMSK
   - XFRCM = 1 in OTG_DOEPMSK
   - XFRCM = 1 in OTG_DIEPMSK
   - TOM = 1 in OTG_DIEPMSK
3. Set up the data FIFO RAM for each of the FIFOs
   - Program the OTG_GRXFSIZ register, to be able to receive control OUT data and setup data. If thresholding is not enabled, at a minimum, this must be equal to 1 max packet size of control endpoint 0 + 2 words (for the status of the control OUT data packet) + 10 words (for setup packets).
   - Program the OTG_DIEPTXF0 register (depending on the FIFO number chosen) to be able to transmit control IN data. At a minimum, this must be equal to 1 max packet size of control endpoint 0.
4. Program the following fields in the endpoint-specific registers for control OUT endpoint 0 to receive a SETUP packet
   - STUPCNT = 3 in OTG_DOEPTSIZ0 (to receive up to 3 back-to-back SETUP packets)

At this point, all initialization required to receive SETUP packets is done.

Endpoint initialization on enumeration completion

1. On the Enumeration Done interrupt (ENUMDNE in OTG_GINTSTS), read the OTG_DSTS register to determine the enumeration speed.
2. Program the MPSIZ field in OTG_DIEPCTL0 to set the maximum packet size. This step configures control endpoint 0. The maximum packet size for a control endpoint depends on the enumeration speed.

At this point, the device is ready to receive SOF packets and is configured to perform control transfers on control endpoint 0.

Endpoint initialization on SetAddress command

This section describes what the application must do when it receives a SetAddress command in a SETUP packet.

1. Program the OTG_DCFG register with the device address received in the SetAddress command
2. Program the core to send out a status IN packet

Endpoint initialization on SetConfiguration/SetInterface command

This section describes what the application must do when it receives a SetConfiguration or SetInterface command in a SETUP packet.

1. When a SetConfiguration command is received, the application must program the endpoint registers to configure them with the characteristics of the valid endpoints in the new configuration.
2. When a SetInterface command is received, the application must program the endpoint registers of the endpoints affected by this command.
3. Some endpoints that were active in the prior configuration or alternate setting are not valid in the new configuration or alternate setting. These invalid endpoints must be deactivated.
4. Unmask the interrupt for each active endpoint and mask the interrupts for all inactive endpoints in the OTG_DAINTMSK register.
5. Set up the data FIFO RAM for each FIFO.
6. After all required endpoints are configured; the application must program the core to send a status IN packet.

At this point, the device core is configured to receive and transmit any type of data packet.

Endpoint activation

This section describes the steps required to activate a device endpoint or to configure an existing device endpoint to a new type.

1. Program the characteristics of the required endpoint into the following fields of the OTG_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG_DOEPCTLx register (for OUT or bidirectional endpoints).
   – Maximum packet size
   – USB active endpoint = 1
   – Endpoint start data toggle (for interrupt and bulk endpoints)
   – Endpoint type
   – Tx FIFO number
2. Once the endpoint is activated, the core starts decoding the tokens addressed to that endpoint and sends out a valid handshake for each valid token received for the endpoint.

Endpoint deactivation

This section describes the steps required to deactivate an existing endpoint.

1. In the endpoint to be deactivated, clear the USB active endpoint bit in the OTG_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG_DOEPCTLx register (for OUT or bidirectional endpoints).
2. Once the endpoint is deactivated, the core ignores tokens addressed to that endpoint, which results in a timeout on the USB.

Note: The application must meet the following conditions to set up the device core to handle traffic:
NPTXFEM and RXFLVLM in the OTG_GINTMSK register must be cleared.

Operational model

SETUP and OUT data transfers:

This section describes the internal data flow and application-level operations during data OUT transfers and SETUP transactions.

- **Packet read**

This section describes how to read packets (OUT data and SETUP packets) from the receive FIFO.

1. On catching an RXFLVL interrupt (OTG_GINTSTS register), the application must read the receive status pop register (OTG_GRXSTSP).
2. The application can mask the RXFLVL interrupt (in OTG_GINTSTS) by writing to RXFLVLM = 0 (in OTG_GINTMSK), until it has read the packet from the receive FIFO.
3. If the received packet's byte count is not 0, the byte count amount of data is popped from the receive data FIFO and stored in memory. If the received packet byte count is 0, no data is popped from the receive data FIFO.
4. The receive status readout of the packet of FIFO indicates one of the following:
   a) Global OUT NAK pattern:
   PKTSTS = Global OUT NAK, BCNT = 0x000, EPNUM = (0x0),
   DPID = (0b00).
   These data indicate that the global OUT NAK bit has taken effect.
   b) SETUP packet pattern:
   PKTSTS = SETUP, BCNT = 0x008, EPNUM = Control EP Num,

DPID = DATA0. These data indicate that a SETUP packet for the specified endpoint is now available for reading from the receive FIFO.

c) Setup stage done pattern:
PKTSTS = Setup Stage Done, BCNT = 0x0, EPNUM = Control EP Num, DPID = (0b00).
These data indicate that the setup stage for the specified endpoint has completed and the data stage has started. After this entry is popped from the receive FIFO, the core asserts a setup interrupt on the specified control OUT endpoint.

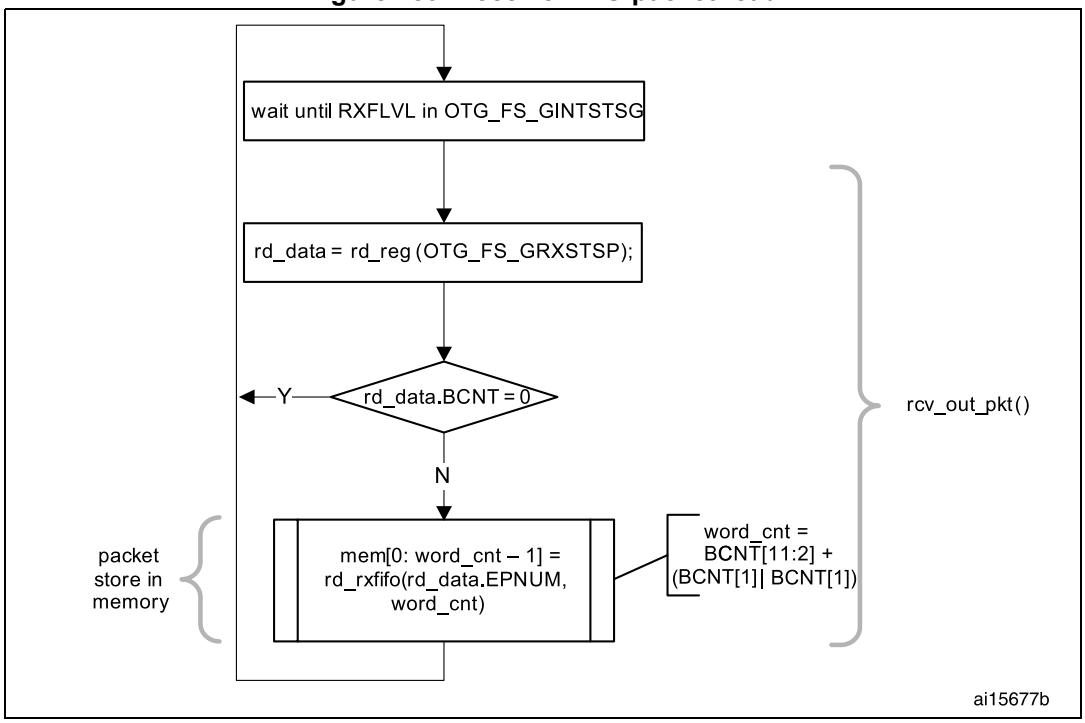
d) Data OUT packet pattern:
PKTSTS = DataOUT, BCNT = size of the received data OUT packet (0 ≤ BCNT ≤ 1 024), EPNUM = EPNUM on which the packet was received, DPID = Actual Data PID.

e) Data transfer completed pattern:
PKTSTS = Data OUT transfer done, BCNT = 0x0, EPNUM = OUT EP Num on which the data transfer is complete, DPID = (0b00).
These data indicate that an OUT data transfer for the specified OUT endpoint has completed. After this entry is popped from the receive FIFO, the core asserts a transfer completed interrupt on the specified OUT endpoint.

5. After the data payload is popped from the receive FIFO, the RXFLVL interrupt (OTG_GINTSTS) must be unmasked.
6. Steps 1–5 are repeated every time the application detects assertion of the interrupt line due to RXFLVL in OTG_GINTSTS. Reading an empty receive FIFO can result in undefined core behavior.

Figure 789 provides a flowchart of the above procedure.

Figure 789. Receive FIFO packet read



SETUP transactions

This section describes how the core handles SETUP packets and the application's sequence for handling SETUP transactions.

- **Application requirements**

1. To receive a SETUP packet, the STUPCNT field (OTG_DOEPTSIZx) in a control OUT endpoint must be programmed to a non-zero value. When the application programs the STUPCNT field to a non-zero value, the core receives SETUP packets and writes them to the receive FIFO, irrespective of the NAK status and EPENA bit setting in OTG_DOEPCTLx. The STUPCNT field is decremented every time the control endpoint receives a SETUP packet. If the STUPCNT field is not programmed to a proper value before receiving a SETUP packet, the core still receives the SETUP packet and decrements the STUPCNT field, but the application may not be able to determine the correct number of SETUP packets received in the setup stage of a control transfer.
   – STUPCNT = 3 in OTG_DOEPTSIZx
2. The application must always allocate some extra space in the receive data FIFO, to be able to receive up to three SETUP packets on a control endpoint.
   – The space to be reserved is 10 words. Three words are required for the first SETUP packet, 1 word is required for the setup stage done word and 6 words are required to store two extra SETUP packets among all control endpoints.
   – 3 words per SETUP packet are required to store 8 bytes of SETUP data and 4 bytes of SETUP status (setup packet pattern). The core reserves this space in the receive data FIFO to write SETUP data only, and never uses this space for data packets.
3. The application must read the 2 words of the SETUP packet from the receive FIFO.
4. The application must read and discard the setup stage done word from the receive FIFO.

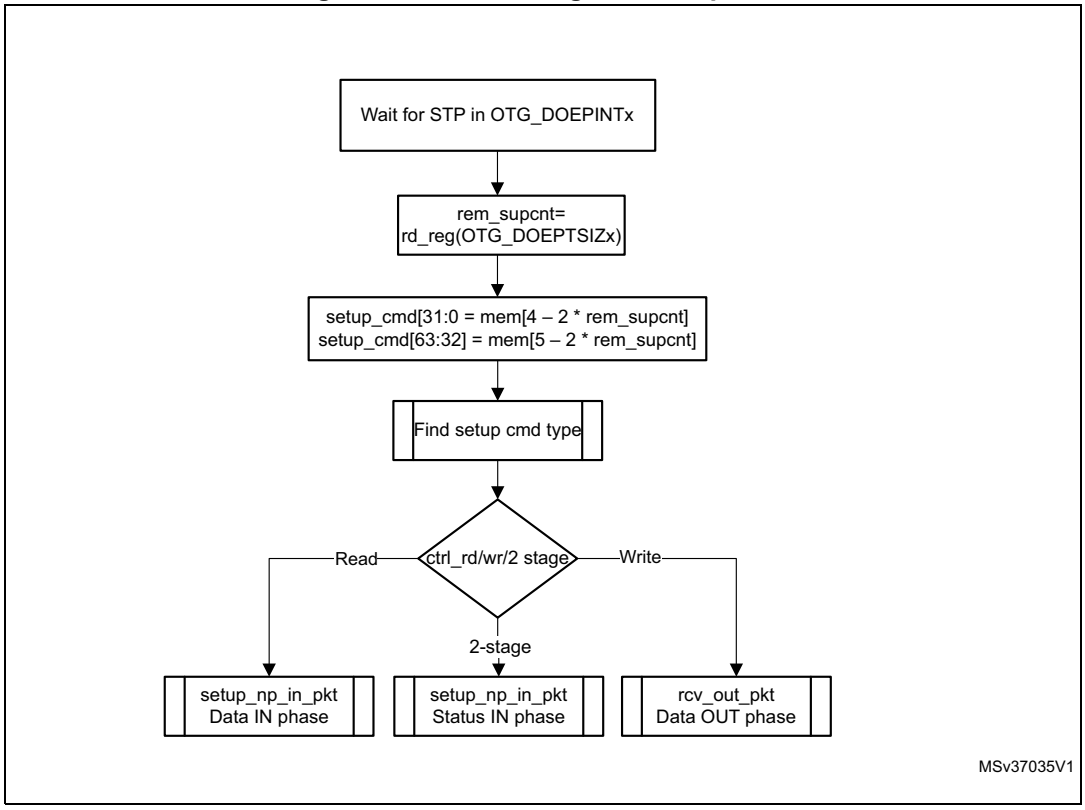
- **Internal data flow**

1. When a SETUP packet is received, the core writes the received data to the receive FIFO, without checking for available space in the receive FIFO and irrespective of the endpoint's NAK and STALL bit settings.
   – The core internally sets the IN NAK and OUT NAK bits for the control IN/OUT endpoints on which the SETUP packet was received.
2. For every SETUP packet received on the USB, 3 words of data are written to the receive FIFO, and the STUPCNT field is decremented by 1.
   – The first word contains control information used internally by the core
   – The second word contains the first 4 bytes of the SETUP command
   – The third word contains the last 4 bytes of the SETUP command
3. When the setup stage changes to a data IN/OUT stage, the core writes an entry (setup stage done word) to the receive FIFO, indicating the completion of the setup stage.
4. On the AHB side, SETUP packets are emptied by the application.
5. When the application pops the setup stage done word from the receive FIFO, the core interrupts the application with an STUP interrupt (OTG_DOEPINTx), indicating it can process the received SETUP packet.
6. The core clears the endpoint enable bit for control OUT endpoints.

- **Application programming sequence**

1. Program the OTG_DOEPTSIZx register.
   – STUPCNT = 3
2. Wait for the RXFLVL interrupt (OTG_GINTSTS) and empty the data packets from the receive FIFO.
3. Assertion of the STUP interrupt (OTG_DOEPINTx) marks a successful completion of the SETUP data transfer.
   – On this interrupt, the application must read the OTG_DOEPTSIZx register to determine the number of SETUP packets received and process the last received SETUP packet.

Figure 790. Processing a SETUP packet



- **Handling more than three back-to-back SETUP packets**

Per the USB 2.0 specification, normally, during a SETUP packet error, a host does not send more than three back-to-back SETUP packets to the same endpoint. However, the USB 2.0 specification does not limit the number of back-to-back SETUP packets a host can send to the same endpoint. When this condition occurs, the OTG_FS controller generates an interrupt (B2BSTUP in OTG_DOEPINTx).

- **Setting the global OUT NAK**

Internal data flow:

1. When the application sets the Global OUT NAK (SGONAK bit in OTG_DCTL), the core stops writing data, except SETUP packets, to the receive FIFO. Irrespective of the

space availability in the receive FIFO, non-isochronous OUT tokens receive a NAK handshake response, and the core ignores isochronous OUT data packets

2. The core writes the Global OUT NAK pattern to the receive FIFO. The application must reserve enough receive FIFO space to write this data pattern.
3. When the application pops the Global OUT NAK pattern word from the receive FIFO, the core sets the GONAKEFF interrupt (OTG_GINTSTS).
4. Once the application detects this interrupt, it can assume that the core is in Global OUT NAK mode. The application can clear this interrupt by clearing the SGONAK bit in OTG_DCTL.

Application programming sequence:

1. To stop receiving any kind of data in the receive FIFO, the application must set the Global OUT NAK bit by programming the following field:
   – SGONAK = 1 in OTG_DCTL
2. Wait for the assertion of the GONAKEFF interrupt in OTG_GINTSTS. When asserted, this interrupt indicates that the core has stopped receiving any type of data except SETUP packets.
3. The application can receive valid OUT packets after it has set SGONAK in OTG_DCTL and before the core asserts the GONAKEFF interrupt (OTG_GINTSTS).
4. The application can temporarily mask this interrupt by writing to the GONAKEFFM bit in the OTG_GINTMSK register.
   – GONAKEFFM = 0 in the OTG_GINTMSK register
5. Whenever the application is ready to exit the Global OUT NAK mode, it must clear the SGONAK bit in OTG_DCTL. This also clears the GONAKEFF interrupt (OTG_GINTSTS).
   – CGONAK = 1 in OTG_DCTL
6. If the application has masked this interrupt earlier, it must be unmasked as follows:
   – GONAKEFFM = 1 in OTG_GINTMSK

- **Disabling an OUT endpoint**

The application must use this sequence to disable an OUT endpoint that it has enabled.

Application programming sequence:

1. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core.
   – SGONAK = 1 in OTG_DCTL
2. Wait for the GONAKEFF interrupt (OTG_GINTSTS)
3. Disable the required OUT endpoint by programming the following fields:
   – EPDIS = 1 in OTG_DOEPCTLx
   – SNAK = 1 in OTG_DOEPCTLx
4. Wait for the EPDISD interrupt (OTG_DOEPINTx), which indicates that the OUT endpoint is completely disabled. When the EPDISD interrupt is asserted, the core also clears the following bits:
   – EPDIS = 0 in OTG_DOEPCTLx
   – EPENA = 0 in OTG_DOEPCTLx
5. The application must clear the Global OUT NAK bit to start receiving data from other non-disabled OUT endpoints.
   – SGONAK = 0 in OTG_DCTL

- **Transfer Stop Programming for OUT endpoints**

The application must use the following programing sequence to stop any transfers (because of an interrupt from the host, typically a reset).

Sequence of operations:

1. Enable all OUT endpoints by setting
   – EPENA = 1 in all NA_DOEPCTLx registers.
2. Flush the RxFIFO as follows
   – Poll NA_GRSTCTL.AHBIDL until it is 1. This indicates that AHB master is idle.
   – Perform read modify write operation on NA_GRSTCTL.RXFFLSH =1
   – Poll NA_GRSTCTL.RXFFLSH until it is 0, but also using a timeout of less than 10 milli-seconds (corresponds to minimum reset signaling duration). If 0 is seen before the timeout, then the RxFIFO flush is successful. If at the moment the timeout occurs, there is still a 1, (this may be due to a packet on EP0 coming from the host) then go back (once only) to the previous step (“Perform read modify write operation”).
3. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core, according to the instructions in “*Setting the global OUT NAK on page 2796*”. This ensures that data in the RxFIFO is sent to the application successfully. Set SGONAK = 1 in NA_DCTL
4. Wait for the GONAKEFF interrupt (NA_GINTSTS)
5. Disable all active OUT endpoints by programming the following register bits:
   – EPDIS = 1 in registers NA_DOEPCTLx
   – SNAK = 1 in registers NA_DOEPCTLx
6. Wait for the EPDIS interrupt in NA_DOEPINTx for each OUT endpoint programmed in the previous step. The EPDIS interrupt in NA_DOEPINTx indicates that the

corresponding OUT endpoint is completely disabled. When the EPDIS interrupt is asserted, the following bits are cleared:

– EPENA = 0 in registers NA_DOEPCTLx
– EPDIS = 0 in registers NA_DOEPCTLx
– SNAK = 0 in registers NA_DOEPCTLx

- **Generic non-isochronous OUT data transfers**

This section describes a regular non-isochronous OUT data transfer (control, bulk, or interrupt).

Application requirements:

1. Before setting up an OUT transfer, the application must allocate a buffer in the memory to accommodate all data to be received as part of the OUT transfer.
2. For OUT transfers, the transfer size field in the endpoint's transfer size register must be a multiple of the maximum packet size of the endpoint, adjusted to the word boundary.
   – transfer size[EPNUM] = *n* × (MPSIZ[EPNUM] + 4 – (MPSIZ[EPNUM] mod 4))
   – packet count[EPNUM] = *n*
   – *n* > 0
3. On any OUT endpoint interrupt, the application must read the endpoint's transfer size register to calculate the size of the payload in the memory. The received payload size can be less than the programmed transfer size.
   – Payload size in memory = application programmed initial transfer size – core updated final transfer size
   – Number of USB packets in which this payload was received = application programmed initial packet count – core updated final packet count

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers, clear the NAK bit, and enable the endpoint to receive the data.
2. Once the NAK bit is cleared, the core starts receiving data and writes it to the receive FIFO, as long as there is space in the receive FIFO. For every data packet received on the USB, the data packet and its status are written to the receive FIFO. Every packet (maximum packet size or short packet) written to the receive FIFO decrements the packet count field for that endpoint by 1.
   – OUT data packets received with bad data CRC are flushed from the receive FIFO automatically.
   – After sending an ACK for the packet on the USB, the core discards non-isochronous OUT data packets that the host, which cannot detect the ACK, re-sends. The application does not detect multiple back-to-back data OUT packets on the same endpoint with the same data PID. In this case the packet count is not decremented.
   – If there is no space in the receive FIFO, isochronous or non-isochronous data packets are ignored and not written to the receive FIFO. Additionally, non-isochronous OUT tokens receive a NAK handshake reply.
   – In all the above three cases, the packet count is not decremented because no data are written to the receive FIFO.

3. When the packet count becomes 0 or when a short packet is received on the endpoint, the NAK bit for that endpoint is set. Once the NAK bit is set, the isochronous or non-isochronous data packets are ignored and not written to the receive FIFO, and non-isochronous OUT tokens receive a NAK handshake reply.
4. After the data are written to the receive FIFO, the application reads the data from the receive FIFO and writes it to external memory, one packet at a time per endpoint.
5. At the end of every packet write on the AHB to external memory, the transfer size for the endpoint is decremented by the size of the written packet.
6. The OUT data transfer completed pattern for an OUT endpoint is written to the receive FIFO on one of the following conditions:
   – The transfer size is 0 and the packet count is 0
   – The last OUT data packet written to the receive FIFO is a short packet (0 ≤ packet size < maximum packet size)
7. When either the application pops this entry (OUT data transfer completed), a transfer completed interrupt is generated for the endpoint and the endpoint enable is cleared.

Application programming sequence:

1. Program the OTG_DOEPTSIZx register for the transfer size and the corresponding packet count.
2. Program the OTG_DOEPCTLx register with the endpoint characteristics, and set the EPENA and CNAK bits.
   – EPENA = 1 in OTG_DOEPCTLx
   – CNAK = 1 in OTG_DOEPCTLx
3. Wait for the RXFLVL interrupt (in OTG_GINTSTS) and empty the data packets from the receive FIFO.
   – This step can be repeated many times, depending on the transfer size.
4. Asserting the XFRC interrupt (OTG_DOEPINTx) marks a successful completion of the non-isochronous OUT data transfer.
5. Read the OTG_DOEPTSIZx register to determine the size of the received data payload.

- **Generic isochronous OUT data transfer**

This section describes a regular isochronous OUT data transfer.

Application requirements:

1. All the application requirements for non-isochronous OUT data transfers also apply to isochronous OUT data transfers.
2. For isochronous OUT data transfers, the transfer size and packet count fields must always be set to the number of maximum-packet-size packets that can be received in a single frame and no more. Isochronous OUT data transfers cannot span more than 1 frame.
3. The application must read all isochronous OUT data packets from the receive FIFO (data and status) before the end of the periodic frame (EOPF interrupt in OTG_GINTSTS).
4. To receive data in the following frame, an isochronous OUT endpoint must be enabled after the EOPF (OTG_GINTSTS) and before the SOF (OTG_GINTSTS).

Internal data flow:

1. The internal data flow for isochronous OUT endpoints is the same as that for non-isochronous OUT endpoints, but for a few differences.
2. When an isochronous OUT endpoint is enabled by setting the endpoint enable and clearing the NAK bits, the Even/Odd frame bit must also be set appropriately. The core receives data on an isochronous OUT endpoint in a particular frame only if the following condition is met:
   – EONUM (in OTG_DOEPCTLx) = FNSOF[0] (in OTG_DSTS)
3. When the application completely reads an isochronous OUT data packet (data and status) from the receive FIFO, the core updates the RXDPID field in OTG_DOEPTSIZx with the data PID of the last isochronous OUT data packet read from the receive FIFO.

Application programming sequence:

1. Program the OTG_DOEPTSIZx register for the transfer size and the corresponding packet count
2. Program the OTG_DOEPCTLx register with the endpoint characteristics and set the endpoint enable, ClearNAK, and Even/Odd frame bits.
   – EPENA = 1
   – CNAK = 1
   – EONUM = (0: Even/1: Odd)
3. Wait for the RXFLVL interrupt (in OTG_GINTSTS) and empty the data packets from the receive FIFO
   – This step can be repeated many times, depending on the transfer size.
4. The assertion of the XFRC interrupt (in OTG_DOEPINTx) marks the completion of the isochronous OUT data transfer. This interrupt does not necessarily mean that the data in memory are good.
5. This interrupt cannot always be detected for isochronous OUT transfers. Instead, the application can detect the INCOMPISOOUT interrupt in OTG_GINTSTS.
6. Read the OTG_DOEPTSIZx register to determine the size of the received transfer and to determine the validity of the data received in the frame. The application must treat the data received in memory as valid only if one of the following conditions is met:
   – RXDPID = DATA0 (in OTG_DOEPTSIZx) and the number of USB packets in which this payload was received = 1
   – RXDPID = DATA1 (in OTG_DOEPTSIZx) and the number of USB packets in which this payload was received = 2

   The number of USB packets in which this payload was received = Application programmed initial packet count – core updated final packet count

   The application can discard invalid data packets.

- **Incomplete isochronous OUT data transfers**

This section describes the application programming sequence when isochronous OUT data packets are dropped inside the core.

Internal data flow:

1. For isochronous OUT endpoints, the XFRC interrupt (in OTG_DOEPINTx) may not always be asserted. If the core drops isochronous OUT data packets, the application

could fail to detect the XFRC interrupt (OTG_DOEPINTx) under the following circumstances:

- When the receive FIFO cannot accommodate the complete ISO OUT data packet, the core drops the received ISO OUT data
- When the isochronous OUT data packet is received with CRC errors
- When the isochronous OUT token received by the core is corrupted
- When the application is very slow in reading the data from the receive FIFO

2. When the core detects an end of periodic frame before transfer completion to all isochronous OUT endpoints, it asserts the incomplete isochronous OUT data interrupt (INCOMPISOOUT in OTG_GINTSTS), indicating that an XFRC interrupt (in OTG_DOEPINTx) is not asserted on at least one of the isochronous OUT endpoints. At this point, the endpoint with the incomplete transfer remains enabled, but no active transfers remain in progress on this endpoint on the USB.

Application programming sequence:

1. Asserting the INCOMPISOOUT interrupt (OTG_GINTSTS) indicates that in the current frame, at least one isochronous OUT endpoint has an incomplete transfer.
2. If this occurs because isochronous OUT data is not completely emptied from the endpoint, the application must ensure that the application empties all isochronous OUT data (data and status) from the receive FIFO before proceeding.
   - When all data are emptied from the receive FIFO, the application can detect the XFRC interrupt (OTG_DOEPINTx). In this case, the application must re-enable the endpoint to receive isochronous OUT data in the next frame.
3. When it receives an INCOMPISOOUT interrupt (in OTG_GINTSTS), the application must read the control registers of all isochronous OUT endpoints (OTG_DOEPCTLx) to determine which endpoints had an incomplete transfer in the current microframe. An endpoint transfer is incomplete if both the following conditions are met:
   - EONUM bit (in OTG_DOEPCTLx) = FNSOF[0] (in OTG_DSTS)
   - EPENA = 1 (in OTG_DOEPCTLx)
4. The previous step must be performed before the SOF interrupt (in OTG_GINTSTS) is detected, to ensure that the current frame number is not changed.
5. For isochronous OUT endpoints with incomplete transfers, the application must discard the data in the memory and disable the endpoint by setting the EPDIS bit in OTG_DOEPCTLx.
6. Wait for the EPDISD interrupt (in OTG_DOEPINTx) and enable the endpoint to receive new data in the next frame.
   - Because the core can take some time to disable the endpoint, the application may not be able to receive the data in the next frame after receiving bad isochronous data.

- **Stalling a non-isochronous OUT endpoint**

This section describes how the application can stall a non-isochronous endpoint.

1. Put the core in the Global OUT NAK mode.
2. Disable the required endpoint
   – When disabling the endpoint, instead of setting the SNAK bit in OTG_DOEPCTL, set STALL = 1 (in OTG_DOEPCTL).

     The STALL bit always takes precedence over the NAK bit.
3. When the application is ready to end the STALL handshake for the endpoint, the STALL bit (in OTG_DOEPCTLx) must be cleared.
4. If the application is setting or clearing a STALL for an endpoint due to a SetFeature.Endpoint Halt or ClearFeature.Endpoint Halt command, the STALL bit must be set or cleared before the application sets up the status stage transfer on the control endpoint.

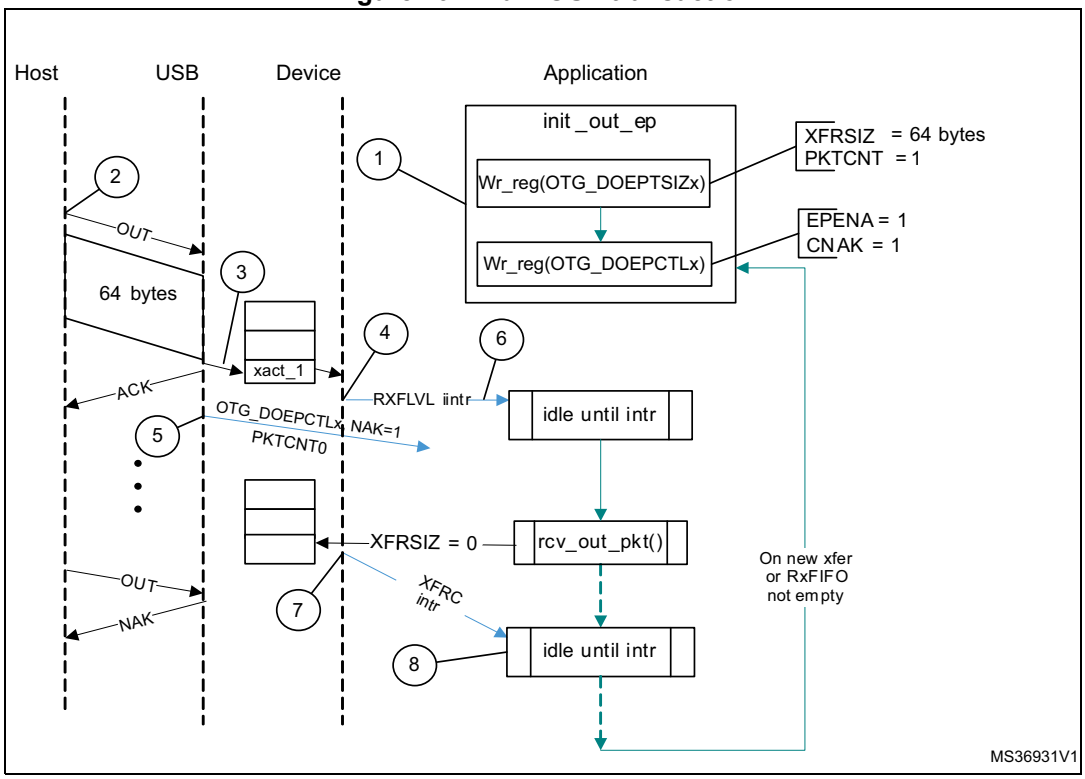
Examples

This section describes and depicts some fundamental transfer types and scenarios.

- Bulk OUT transaction

Figure 791 depicts the reception of a single Bulk OUT data packet from the USB to the AHB and describes the events involved in the process.

Figure 791. Bulk OUT transaction



After a SetConfiguration/SetInterface command, the application initializes all OUT endpoints by setting CNAK = 1 and EPENA = 1 (in OTG_DOEPCTLx), and setting a suitable XFRSIZ and PKTCNT in the OTG_DOEPTSIZx register.

1. host attempts to send data (OUT token) to an endpoint.
2. When the core receives the OUT token on the USB, it stores the packet in the Rx FIFO because space is available there.
3. After writing the complete packet in the Rx FIFO, the core then asserts the RXFLVL interrupt (in OTG_GINTSTS).
4. On receiving the PKTCNT number of USB packets, the core internally sets the NAK bit for this endpoint to prevent it from receiving any more packets.
5. The application processes the interrupt and reads the data from the Rx FIFO.
6. When the application has read all the data (equivalent to XFRSIZ), the core generates an XFRC interrupt (in OTG_DOEPINTx).
7. The application processes the interrupt and uses the setting of the XFRC interrupt bit (in OTG_DOEPINTx) to determine that the intended transfer is complete.

IN data transfers

- **Packet write**

This section describes how the application writes data packets to the endpoint FIFO when dedicated transmit FIFOs are enabled.

1. The application can either choose the polling or the interrupt mode.
   - In polling mode, the application monitors the status of the endpoint transmit data FIFO by reading the OTG_DTXFSTSx register, to determine if there is enough space in the data FIFO.
   - In interrupt mode, the application waits for the TXFE interrupt (in OTG_DIEPINTx) and then reads the OTG_DTXFSTSx register, to determine if there is enough space in the data FIFO.
   - To write a single non-zero length data packet, there must be space to write the entire packet in the data FIFO.
   - To write zero length packet, the application must not look at the FIFO space.
2. Using one of the above mentioned methods, when the application determines that there is enough space to write a transmit packet, the application must first write into the endpoint control register, before writing the data into the data FIFO. Typically, the application, must do a read modify write on the OTG_DIEPCTLx register to avoid modifying the contents of the register, except for setting the endpoint enable bit.

The application can write multiple packets for the same endpoint into the transmit FIFO, if space is available. For periodic IN endpoints, the application must write packets only for one microframe. It can write packets for the next periodic transaction only after getting transfer complete for the previous transaction.

- **Setting IN endpoint NAK**

Internal data flow:

1. When the application sets the IN NAK for a particular endpoint, the core stops transmitting data on the endpoint, irrespective of data availability in the endpoint's transmit FIFO.
2. Non-isochronous IN tokens receive a NAK handshake reply
   – Isochronous IN tokens receive a zero-data-length packet reply
3. The core asserts the INEPNE (IN endpoint NAK effective) interrupt in OTG_DIEPINTx in response to the SNAK bit in OTG_DIEPCTLx.
4. Once this interrupt is seen by the application, the application can assume that the endpoint is in IN NAK mode. This interrupt can be cleared by the application by setting the CNAK bit in OTG_DIEPCTLx.

Application programming sequence:

1. To stop transmitting any data on a particular IN endpoint, the application must set the IN NAK bit. To set this bit, the following field must be programmed.
   – SNAK = 1 in OTG_DIEPCTLx
2. Wait for assertion of the INEPNE interrupt in OTG_DIEPINTx. This interrupt indicates that the core has stopped transmitting data on the endpoint.
3. The core can transmit valid IN data on the endpoint after the application has set the NAK bit, but before the assertion of the NAK Effective interrupt.
4. The application can mask this interrupt temporarily by writing to the INEPNEM bit in OTG_DIEPMSK.
   – INEPNEM = 0 in OTG_DIEPMSK
5. To exit endpoint NAK mode, the application must clear the NAK status bit (NAKSTS) in OTG_DIEPCTLx. This also clears the INEPNE interrupt (in OTG_DIEPINTx).
   – CNAK = 1 in OTG_DIEPCTLx
6. If the application masked this interrupt earlier, it must be unmasked as follows:
   – INEPNEM = 1 in OTG_DIEPMSK

- **IN endpoint disable**

Use the following sequence to disable a specific IN endpoint that has been previously enabled.

Application programming sequence:

1. The application must stop writing data on the AHB for the IN endpoint to be disabled.
2. The application must set the endpoint in NAK mode.
   – SNAK = 1 in OTG_DIEPCTLx
3. Wait for the INEPNE interrupt in OTG_DIEPINTx.
4. Set the following bits in the OTG_DIEPCTLx register for the endpoint that must be disabled.
   – EPDIS = 1 in OTG_DIEPCTLx
   – SNAK = 1 in OTG_DIEPCTLx
5. Assertion of the EPDISD interrupt in OTG_DIEPINTx indicates that the core has completely disabled the specified endpoint. Along with the assertion of the interrupt, the core also clears the following bits:
   – EPENA = 0 in OTG_DIEPCTLx
   – EPDIS = 0 in OTG_DIEPCTLx
6. The application must read the OTG_DIEPTSIZx register for the periodic IN EP, to calculate how much data on the endpoint were transmitted on the USB.
7. The application must flush the data in the endpoint transmit FIFO, by setting the following fields in the OTG_GRSTCTL register:
   – TXFNUM (in OTG_GRSTCTL) = Endpoint transmit FIFO number
   – TXFFLSH in (OTG_GRSTCTL) = 1

The application must poll the OTG_GRSTCTL register, until the TXFFLSH bit is cleared by the core, which indicates the end of flush operation. To transmit new data on this endpoint, the application can re-enable the endpoint at a later point.

- **Transfer Stop Programming for IN endpoints**

The application must use the following programing sequence to stop any transfers (because of an interrupt from the host, typically a reset).

Sequence of operations:

1. Disable the IN endpoint by setting:
   – EPDIS = 1 in all NA_DIEPCTLx registers
2. Wait for the EPDIS interrupt in NA_DIEPINTx, which indicates that the IN endpoint is completely disabled. When the EPDIS interrupt is asserted the following bits are cleared:
   – EPDIS = 0 in NA_DIEPCTLx
   – EPENA = 0 in NA_DIEPCTLx
3. Flush the TxFIFO by programming the following bits:
   – TXFFLSH = 1 in NA_GRSTCTL
   – TXFNUM = “FIFO number specific to endpoint” in NA_GRSTCTL
4. The application can start polling till TXFFLSH in NA_GRSTCTL is cleared. When this bit is cleared, it ensures that there is no data left in the Tx FIFO.

- **Generic non-periodic IN data transfers**

Application requirements:

1. Before setting up an IN transfer, the application must ensure that all data to be transmitted as part of the IN transfer are part of a single buffer.
2. For IN transfers, the transfer size field in the endpoint transfer size register denotes a payload that constitutes multiple maximum-packet-size packets and a single short packet. This short packet is transmitted at the end of the transfer.
   – To transmit a few maximum-packet-size packets and a short packet at the end of the transfer:

     Transfer size[EPNUM] = *x* × MPSIZ[EPNUM] + sp

     If (sp > 0), then packet count[EPNUM] = x + 1.
     Otherwise, packet count[EPNUM] = *x*
   – To transmit a single zero-length data packet:

     Transfer size[EPNUM] = 0

     Packet count[EPNUM] = 1
   – To transmit a few maximum-packet-size packets and a zero-length data packet at the end of the transfer, the application must split the transfer into two parts. The first sends maximum-packet-size data packets and the second sends the zero-length data packet alone.

     First transfer: transfer size[EPNUM] = *x* × MPSIZ[epnum]; packet count = *n*;

     Second transfer: transfer size[EPNUM] = 0; packet count = 1;
3. Once an endpoint is enabled for data transfers, the core updates the transfer size register. At the end of the IN transfer, the application must read the transfer size register to determine how much data posted in the transmit FIFO have already been sent on the USB.
4. Data fetched into transmit FIFO = Application-programmed initial transfer size – core-updated final transfer size
   – Data transmitted on USB = (application-programmed initial packet count – core updated final packet count) × MPSIZ[EPNUM]
   – Data yet to be transmitted on USB = (Application-programmed initial transfer size – data transmitted on USB)

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The application must also write the required data to the transmit FIFO for the endpoint.
3. Every time a packet is written into the transmit FIFO by the application, the transfer size for that endpoint is decremented by the packet size. The data is fetched from the memory by the application, until the transfer size for the endpoint becomes 0. After writing the data into the FIFO, the “number of packets in FIFO” count is incremented (this is a 3-bit count, internally maintained by the core for each IN endpoint transmit FIFO. The maximum number of packets maintained by the core at any time in an IN endpoint FIFO is eight). For zero-length packets, a separate flag is set for each FIFO, without any data in the FIFO.
4. Once the data are written to the transmit FIFO, the core reads them out upon receiving an IN token. For every non-isochronous IN data packet transmitted with an ACK

handshake, the packet count for the endpoint is decremented by one, until the packet count is zero. The packet count is not decremented on a timeout.

5. For zero length packets (indicated by an internal zero length flag), the core sends out a zero-length packet for the IN token and decrements the packet count field.
6. If there are no data in the FIFO for a received IN token and the packet count field for that endpoint is zero, the core generates an “IN token received when Tx FIFO is empty” (ITTXFE) interrupt for the endpoint, provided that the endpoint NAK bit is not set. The core responds with a NAK handshake for non-isochronous endpoints on the USB.
7. The core internally rewinds the FIFO pointers and no timeout interrupt is generated.
8. When the transfer size is 0 and the packet count is 0, the transfer complete (XFRC) interrupt for the endpoint is generated and the endpoint enable is cleared.

Application programming sequence:

1. Program the OTG_DIEPTSIZx register with the transfer size and corresponding packet count.
2. Program the OTG_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA (endpoint enable) bits.
3. When transmitting non-zero length data packet, the application must poll the OTG_DTXFSTSx register (where x is the FIFO number associated with that endpoint) to determine whether there is enough space in the data FIFO. The application can optionally use TXFE (in OTG_DIEPINTx) before writing the data.

- **Generic periodic IN data transfers**

This section describes a typical periodic IN data transfer.

Application requirements:

1. Application requirements 1, 2, 3, and 4 of *Generic non-periodic IN data transfers on page 2806* also apply to periodic IN data transfers, except for a slight modification of requirement 2.
   – The application can only transmit multiples of maximum-packet-size data packets or multiples of maximum-packet-size packets, plus a short packet at the end. To

transmit a few maximum-packet-size packets and a short packet at the end of the transfer, the following conditions must be met:

transfer size[EPNUM] = x × MPSIZ[EPNUM] + sp
(where *x* is an integer ≥ 0, and 0 ≤ sp < MPSIZ[EPNUM])

If (sp > 0), packet count[EPNUM] = *x* + 1
Otherwise, packet count[EPNUM] = *x*;

MCNT[EPNUM] = packet count[EPNUM]

– The application cannot transmit a zero-length data packet at the end of a transfer. It can transmit a single zero-length data packet by itself. To transmit a single zero-length data packet:
– transfer size[EPNUM] = 0

packet count[EPNUM] = 1

MCNT[EPNUM] = packet count[EPNUM]

2. The application can only schedule data transfers one frame at a time.
   – (MCNT – 1) × MPSIZ ≤ XFERSIZ ≤ MCNT × MPSIZ
   – PKTCNT = MCNT (in OTG_DIEPTSIZx)
   – If XFERSIZ < MCNT × MPSIZ, the last data packet of the transfer is a short packet.
   – Note that: MCNT is in OTG_DIEPTSIZx, MPSIZ is in OTG_DIEPCTLx, PKTCNT is in OTG_DIEPTSIZx and XFERSIZ is in OTG_DIEPTSIZx
3. The complete data to be transmitted in the frame must be written into the transmit FIFO by the application, before the IN token is received. Even when 1 word of the data to be transmitted per frame is missing in the transmit FIFO when the IN token is received, the core behaves as when the FIFO is empty. When the transmit FIFO is empty:
   – A zero data length packet would be transmitted on the USB for isochronous IN endpoints
   – A NAK handshake would be transmitted on the USB for interrupt IN endpoints

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The application must also write the required data to the associated transmit FIFO for the endpoint.
3. Every time the application writes a packet to the transmit FIFO, the transfer size for that endpoint is decremented by the packet size. The data are fetched from application memory until the transfer size for the endpoint becomes 0.
4. When an IN token is received for a periodic endpoint, the core transmits the data in the FIFO, if available. If the complete data payload (complete packet, in dedicated FIFO

mode) for the frame is not present in the FIFO, then the core generates an IN token received when Tx FIFO empty interrupt for the endpoint.

- A zero-length data packet is transmitted on the USB for isochronous IN endpoints
- A NAK handshake is transmitted on the USB for interrupt IN endpoints

5. The packet count for the endpoint is decremented by 1 under the following conditions:
   - For isochronous endpoints, when a zero- or non-zero-length data packet is transmitted
   - For interrupt endpoints, when an ACK handshake is transmitted
   - When the transfer size and packet count are both 0, the transfer completed interrupt for the endpoint is generated and the endpoint enable is cleared.
6. At the “Periodic frame Interval” (controlled by PFIVL in OTG_DCFG), when the core finds non-empty any of the isochronous IN endpoint FIFOs scheduled for the current frame non-empty, the core generates an IISOIXFR interrupt in OTG_GINTSTS.

Application programming sequence:

1. Program the OTG_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA bits.
2. Write the data to be transmitted in the next frame to the transmit FIFO.
3. Asserting the ITTXFE interrupt (in OTG_DIEPINTx) indicates that the application has not yet written all data to be transmitted to the transmit FIFO.
4. If the interrupt endpoint is already enabled when this interrupt is detected, ignore the interrupt. If it is not enabled, enable the endpoint so that the data can be transmitted on the next IN token attempt.
5. Asserting the XFRC interrupt (in OTG_DIEPINTx) with no ITTXFE interrupt in OTG_DIEPINTx indicates the successful completion of an isochronous IN transfer. A read to the OTG_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
6. Asserting the XFRC interrupt (in OTG_DIEPINTx), with or without the ITTXFE interrupt (in OTG_DIEPINTx), indicates the successful completion of an interrupt IN transfer. A read to the OTG_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
7. Asserting the incomplete isochronous IN transfer (IISOIXFR) interrupt in OTG_GINTSTS with none of the aforementioned interrupts indicates the core did not receive at least 1 periodic IN token in the current frame.

- **Incomplete isochronous IN data transfers**

This section describes what the application must do on an incomplete isochronous IN data transfer.

Internal data flow:

1. An isochronous IN transfer is treated as incomplete in one of the following conditions:
   a) The core receives a corrupted isochronous IN token on at least one isochronous IN endpoint. In this case, the application detects an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG_GINTSTS).
   b) The application is slow to write the complete data payload to the transmit FIFO and an IN token is received before the complete data payload is written to the FIFO. In this case, the application detects an IN token received when Tx FIFO empty interrupt in OTG_DIEPINTx. The application can ignore this interrupt, as it

eventually results in an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG_GINTSTS) at the end of periodic frame.

The core transmits a zero-length data packet on the USB in response to the received IN token.

2. The application must stop writing the data payload to the transmit FIFO as soon as possible.
3. The application must set the NAK bit and the disable bit for the endpoint.
4. The core disables the endpoint, clears the disable bit, and asserts the endpoint disable interrupt for the endpoint.

Application programming sequence:

1. The application can ignore the IN token received when Tx FIFO empty interrupt in OTG_DIEPINTx on any isochronous IN endpoint, as it eventually results in an incomplete isochronous IN transfer interrupt (in OTG_GINTSTS).
2. Assertion of the incomplete isochronous IN transfer interrupt (in OTG_GINTSTS) indicates an incomplete isochronous IN transfer on at least one of the isochronous IN endpoints.
3. The application must read the endpoint control register for all isochronous IN endpoints to detect endpoints with incomplete IN data transfers.
4. The application must stop writing data to the Periodic Transmit FIFOs associated with these endpoints on the AHB.
5. Program the following fields in the OTG_DIEPCTLx register to disable the endpoint:
   – SNAK = 1 in OTG_DIEPCTLx
   – EPDIS = 1 in OTG_DIEPCTLx
6. The assertion of the endpoint disabled interrupt in OTG_DIEPINTx indicates that the core has disabled the endpoint.
   – At this point, the application must flush the data in the associated transmit FIFO or overwrite the existing data in the FIFO by enabling the endpoint for a new transfer in the next microframe. To flush the data, the application must use the OTG_GRSTCTL register.

- **Stalling non-isochronous IN endpoints**

This section describes how the application can stall a non-isochronous endpoint.

Application programming sequence:

1. Disable the IN endpoint to be stalled. Set the STALL bit as well.
2. EPDIS = 1 in OTG_DIEPCTLx, when the endpoint is already enabled
   – STALL = 1 in OTG_DIEPCTLx
   – The STALL bit always takes precedence over the NAK bit
3. Assertion of the endpoint disabled interrupt (in OTG_DIEPINTx) indicates to the application that the core has disabled the specified endpoint.
4. The application must flush the non-periodic or periodic transmit FIFO, depending on the endpoint type. In case of a non-periodic endpoint, the application must re-enable the other non-periodic endpoints that do not need to be stalled, to transmit data.
5. Whenever the application is ready to end the STALL handshake for the endpoint, the STALL bit must be cleared in OTG_DIEPCTLx.
6. If the application sets or clears a STALL bit for an endpoint due to a SetFeature.Endpoint Halt command or ClearFeature.Endpoint Halt command, the STALL bit must be set or cleared before the application sets up the status stage transfer on the control endpoint.

Special case: stalling the control OUT endpoint

The core must stall IN/OUT tokens if, during the data stage of a control transfer, the host sends more IN/OUT tokens than are specified in the SETUP packet. In this case, the application must enable the ITTXFE interrupt in OTG_DIEPINTx and the OTEPDIS interrupt in OTG_DOEPINTx during the data stage of the control transfer, after the core has transferred the amount of data specified in the SETUP packet. Then, when the application receives this interrupt, it must set the STALL bit in the corresponding endpoint control register, and clear this interrupt.

63.16.6 Worst case response time

When the OTG_FS controller acts as a device, there is a worst case response time for any tokens that follow an isochronous OUT. This worst case response time depends on the AHB clock frequency.

The core registers are in the AHB domain, and the core does not accept another token before updating these register values. The worst case is for any token following an isochronous OUT, because for an isochronous transaction, there is no handshake and the next token could come sooner. This worst case value is 7 PHY clocks when the AHB clock is the same as the PHY clock. When the AHB clock is faster, this value is smaller.

If this worst case condition occurs, the core responds to bulk/interrupt tokens with a NAK and drops isochronous and SETUP tokens. The host interprets this as a timeout condition for SETUP and retries the SETUP packet. For isochronous transfers, the Incomplete isochronous IN transfer interrupt (IISOIXFR) and Incomplete isochronous OUT transfer interrupt (IISOOXFR) inform the application that isochronous IN/OUT packets were dropped.

Choosing the value of TRDT in OTG_GUSBCFG

The value in TRDT (OTG_GUSBCFG) is the time it takes for the MAC, in terms of PHY clocks after it has received an IN token, to get the FIFO status, and thus the first data from the PFC block. This time involves the synchronization delay between the PHY and AHB clocks. The worst case delay for this is when the AHB clock is the same as the PHY clock. In this case, the delay is 5 clocks.

Once the MAC receives an IN token, this information (token received) is synchronized to the AHB clock by the PFC (the PFC runs on the AHB clock). The PFC then reads the data from the SPRAM and writes them into the dual clock source buffer. The MAC then reads the data out of the source buffer (4 deep).

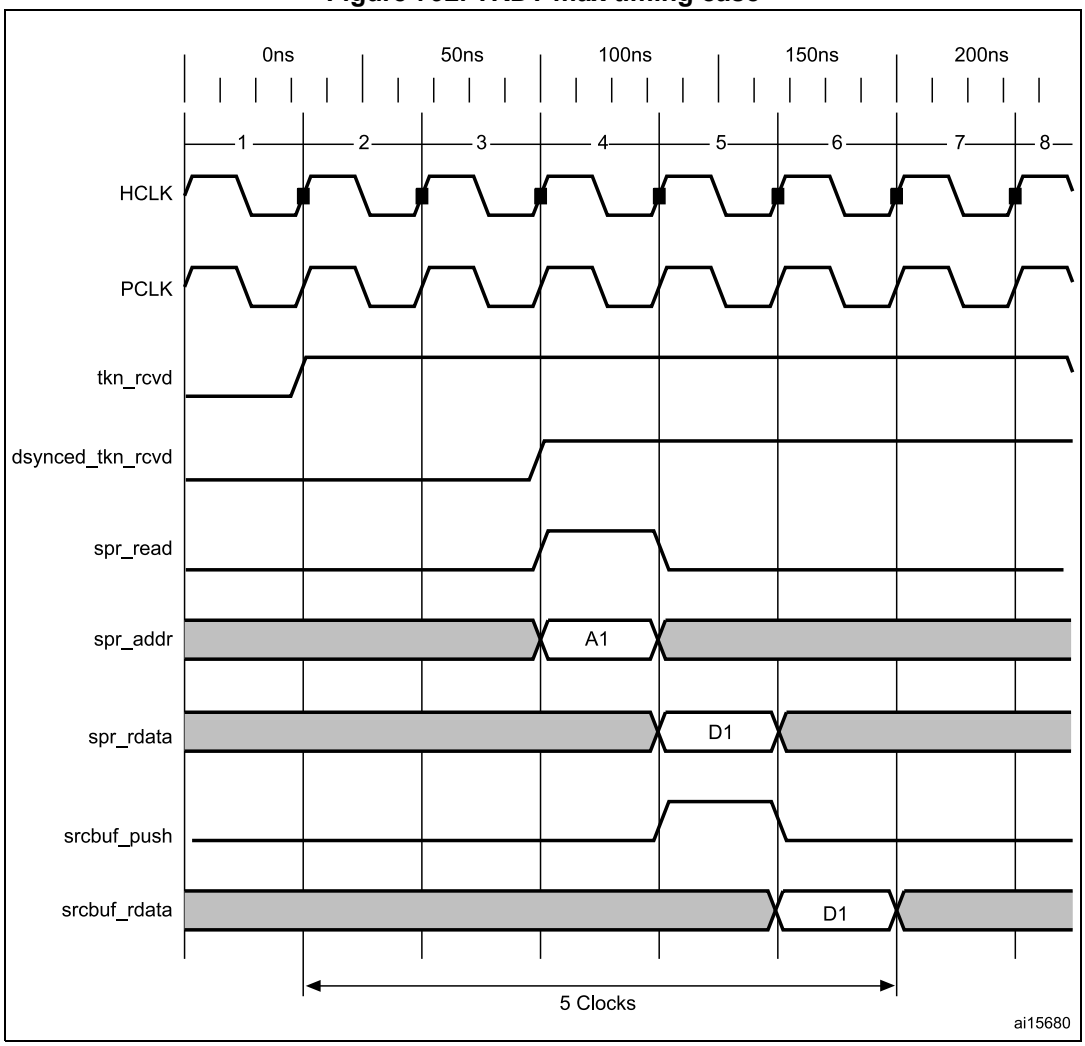
If the AHB is running at a higher frequency than the PHY, the application can use a smaller value for TRDT (in OTG_GUSBCFG).

Figure 792 has the following signals:

- tkn_rcvd: Token received information from MAC to PFC
- dynced_tkn_rcvd: Doubled sync tkn_rcvd, from PCLK to HCLK domain
- spr_read: Read to SPRAM
- spr_addr: Address to SPRAM
- spr_rdata: Read data from SPRAM
- srcbuf_push: Push to the source buffer
- srcbuf_rdata: Read data from the source buffer. Data seen by MAC

To calculate the value of TRDT, refer to *Table 650: TRDT values*.

Figure 792. TRDT max timing case



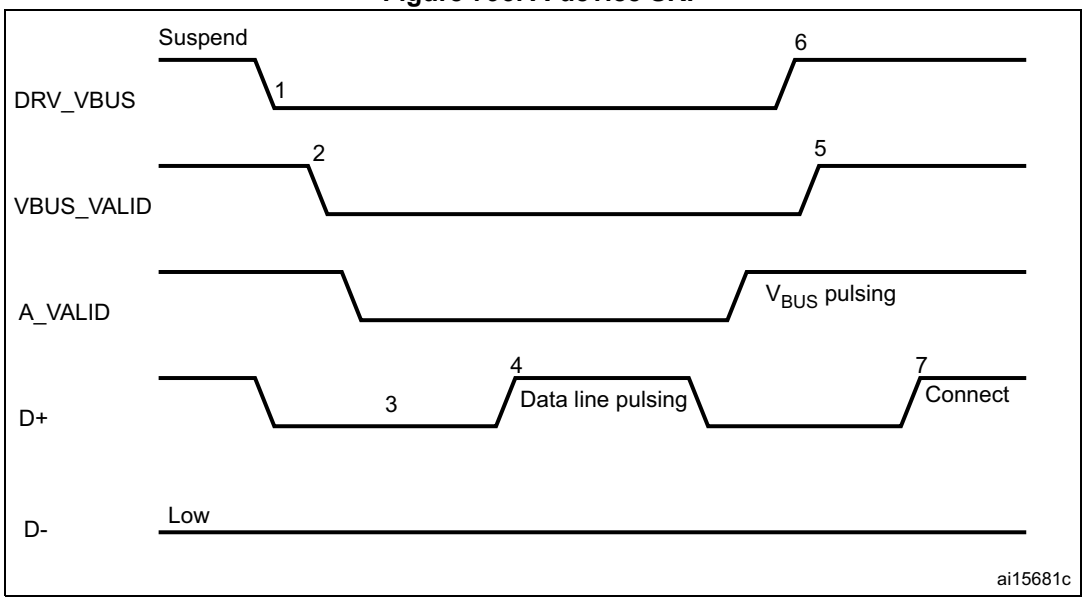
63.16.7 OTG programming model

The OTG_FS controller is an OTG device supporting HNP and SRP. When the core is connected to an “A” plug, it is referred to as an A-device. When the core is connected to a “B” plug it is referred to as a B-device. In host mode, the OTG_FS controller turns off V_{BUS} to conserve power. SRP is a method by which the B-device signals the A-device to turn on V_{BUS} power. A device must perform both data-line pulsing and V_{BUS} pulsing, but a host can detect either data-line pulsing or V_{BUS} pulsing for SRP. HNP is a method by which the B-device negotiates and switches to host role. In Negotiated mode after HNP, the B-device suspends the bus and reverts to the device role.

A-device session request protocol

The application must set the SRP-capable bit in the core USB configuration register. This enables the OTG_FS controller to detect SRP as an A-device.

Figure 793. A-device SRP



1. DRV_VBUS = V_{BUS} drive signal to the PHY
   VBUS_VALID = V_{BUS} valid signal from PHY
   A_VALID = A-peripheral V_{BUS} level signal to PHY
   D+ = Data plus line
   D- = Data minus line

The following points refer and describe the signal numeration shown in the *Figure 793*:

1. To save power, the application suspends and turns off port power when the bus is idle by writing the port suspend and port power bits in the host port control and status register.
2. PHY indicates port power off by deasserting the VBUS_VALID signal.
3. The device must detect SE0 for at least 2 ms to start SRP when V_{BUS} power is off.
4. To initiate SRP, the device turns on its data line pull-up resistor for 5 to 10 ms. The OTG_FS controller detects data-line pulsing.
5. The device drives V_{BUS} above the A-device session valid (2.0 V minimum) for V_{BUS} pulsing.
   The OTG_FS controller interrupts the application on detecting SRP. The session

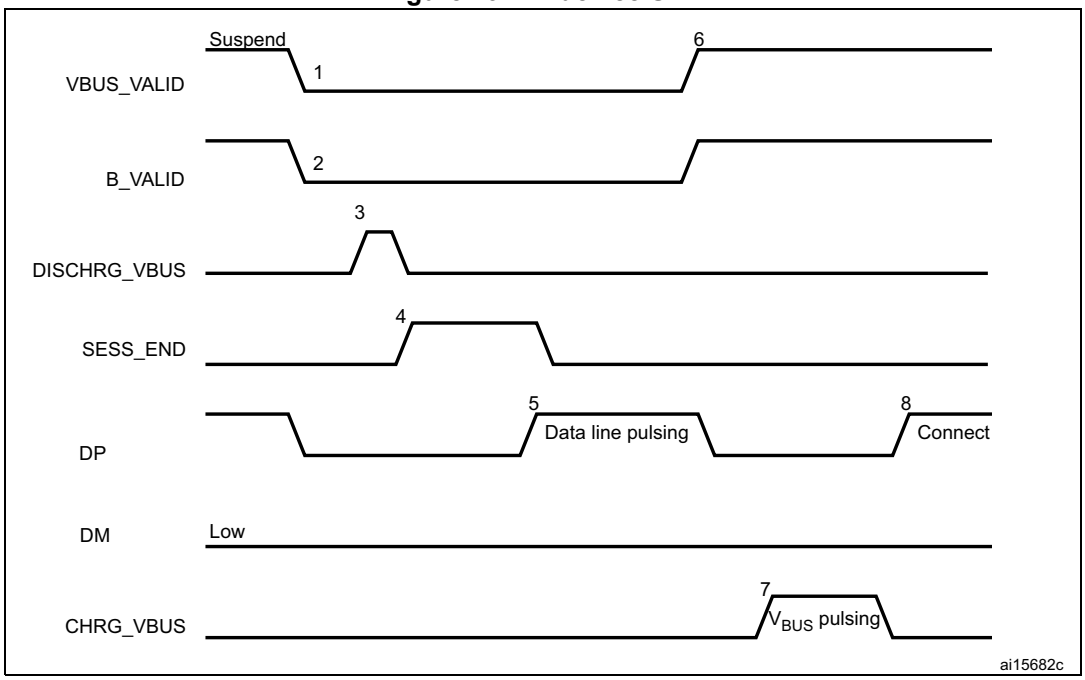
request detected bit is set in Global interrupt status register (SRQINT set in OTG_GINTSTS).

6. The application must service the session request detected interrupt and turn on the port power bit by writing the port power bit in the host port control and status register. The PHY indicates port power-on by asserting the VBUS_VALID signal.
7. When the USB is powered, the device connects, completing the SRP process.

B-device session request protocol

The application must set the SRP-capable bit in the core USB configuration register. This enables the OTG_FS controller to initiate SRP as a B-device. SRP is a means by which the OTG_FS controller can request a new session from the host.

Figure 794. B-device SRP



1. VBUS_VALID = V_{BUS} valid signal from PHY
B_VALID = B-peripheral valid session to PHY
DISCHRG_VBUS = discharge signal to PHY
SESS_END = session end signal to PHY
CHRG_VBUS = charge V_{BUS} signal to PHY
DP = Data plus line
DM = Data minus line

The following points refer and describe the signal numeration shown in the *Figure 794*:

1. To save power, the host suspends and turns off port power when the bus is idle.
The OTG_FS controller sets the early suspend bit in the core interrupt register after 3 ms of bus idleness. Following this, the OTG_FS controller sets the USB suspend bit in the core interrupt register.
The OTG_FS controller informs the PHY to discharge V_{BUS}.
2. The PHY indicates the session's end to the device. This is the initial condition for SRP.
The OTG_FS controller requires 2 ms of SE0 before initiating SRP.
For a USB 1.1 full-speed serial transceiver, the application must wait until V_{BUS} discharges to 0.2 V after BSVLD (in OTG_GOTGCTL) is deasserted. This discharge

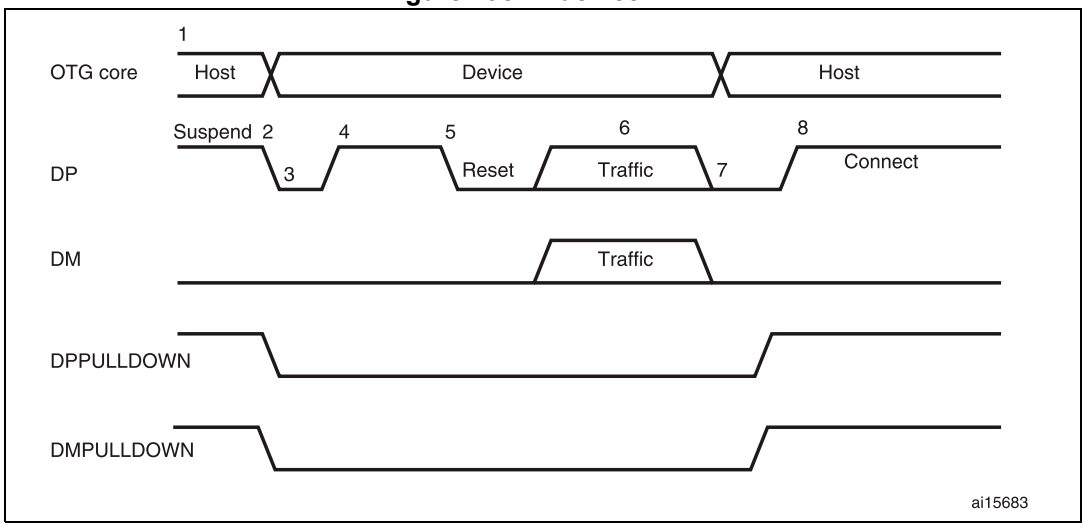
time can be obtained from the transceiver vendor and varies from one transceiver to another.

3. The OTG_FS core informs the PHY to speed up V_{BUS} discharge.
4. The application initiates SRP by writing the session request bit in the OTG control and status register. The OTG_FS controller perform data-line pulsing followed by V_{BUS} pulsing.
5. The host detects SRP from either the data-line or V_{BUS} pulsing, and turns on V_{BUS}. The PHY indicates V_{BUS} power-on to the device.
6. The OTG_FS controller performs V_{BUS} pulsing.
   The host starts a new session by turning on V_{BUS}, indicating SRP success. The OTG_FS controller interrupts the application by setting the session request success status change bit in the OTG interrupt status register. The application reads the session request success bit in the OTG control and status register.
7. When the USB is powered, the OTG_FS controller connects, completing the SRP process.

A-device host negotiation protocol

HNP switches the USB host role from the A-device to the B-device. The application must set the HNP-capable bit in the core USB configuration register to enable the OTG_FS controller to perform HNP as an A-device.

Figure 795. A-device HNP



1. DPPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DP line inside the PHY.
   DMPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DM line inside the PHY.

The following points refer and describe the signal numeration shown in the *Figure 795*:

1. The OTG_FS controller sends the B-device a SetFeature b_hnp_enable descriptor to enable HNP support. The B-device's ACK response indicates that the B-device supports HNP. The application must set host Set HNP enable bit in the OTG control

and status register to indicate to the OTG_FS controller that the B-device supports HNP.

2. When it has finished using the bus, the application suspends by writing the port suspend bit in the host port control and status register.
3. When the B-device observes a USB suspend, it disconnects, indicating the initial condition for HNP. The B-device initiates HNP only when it must switch to the host role; otherwise, the bus continues to be suspended.

   The OTG_FS controller sets the host negotiation detected interrupt in the OTG interrupt status register, indicating the start of HNP.

   The OTG_FS controller deasserts the DM pull down and DM pull down in the PHY to indicate a device role. The PHY enables the OTG_DP pull-up resistor to indicate a connect for B-device.

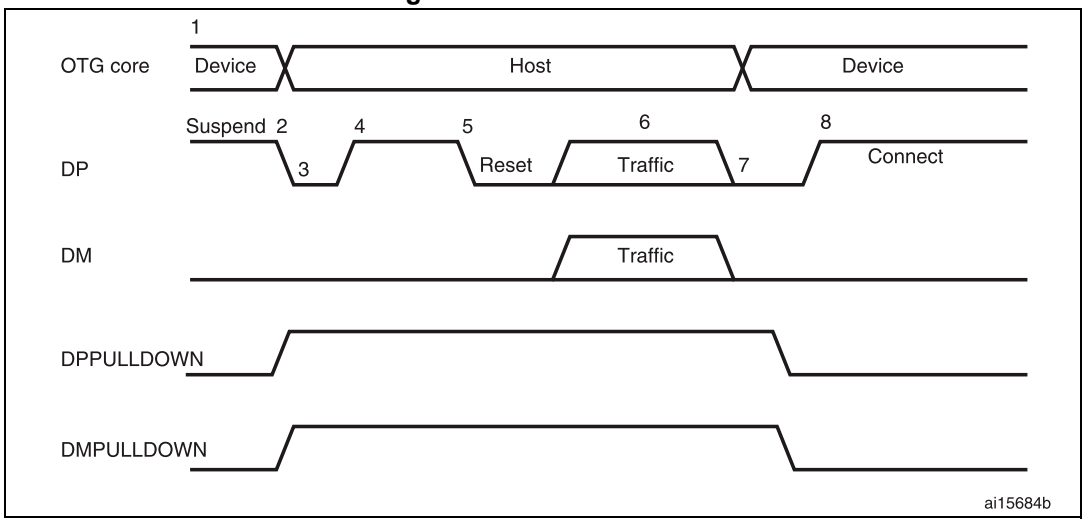
   The application must read the current mode bit in the OTG control and status register to determine device mode operation.
4. The B-device detects the connection, issues a USB reset, and enumerates the OTG_FS controller for data traffic.
5. The B-device continues the host role, initiating traffic, and suspends the bus when done.

   The OTG_FS controller sets the early suspend bit in the core interrupt register after 3 ms of bus idleness. Following this, the OTG_FS controller sets the USB suspend bit in the core interrupt register.
6. In Negotiated mode, the OTG_FS controller detects the suspend, disconnects, and switches back to the host role. The OTG_FS controller asserts the DM pull down and DM pull down in the PHY to indicate its assumption of the host role.
7. The OTG_FS controller sets the connector ID status change interrupt in the OTG interrupt status register. The application must read the connector ID status in the OTG control and status register to determine the OTG_FS controller operation as an A-device. This indicates the completion of HNP to the application. The application must read the Current mode bit in the OTG control and status register to determine host mode operation.
8. The B-device connects, completing the HNP process.

B-device host negotiation protocol

HNP switches the USB host role from B-device to A-device. The application must set the HNP-capable bit in the core USB configuration register to enable the OTG_FS controller to perform HNP as a B-device.

Figure 796. B-device HNP



1. DPPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DP line inside the PHY. DMPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DM line inside the PHY.

The following points refer and describe the signal numeration shown in the *Figure 796*:

1. The A-device sends the SetFeature b_hnp_enable descriptor to enable HNP support. The OTG_FS controller's ACK response indicates that it supports HNP. The application must set the device HNP enable bit in the OTG control and status register to indicate HNP support.

   The application sets the HNP request bit in the OTG control and status register to indicate to the OTG_FS controller to initiate HNP.

2. When it has finished using the bus, the A-device suspends by writing the port suspend bit in the host port control and status register.

   The OTG_FS controller sets the Early suspend bit in the core interrupt register after 3 ms of bus idleness. Following this, the OTG_FS controller sets the USB suspend bit in the core interrupt register.

   The OTG_FS controller disconnects and the A-device detects SE0 on the bus, indicating HNP. The OTG_FS controller asserts the DP pull down and DM pull down in the PHY to indicate its assumption of the host role.

   The A-device responds by activating its OTG_DP pull-up resistor within 3 ms of detecting SE0. The OTG_FS controller detects this as a connect.

   The OTG_FS controller sets the host negotiation success status change interrupt in the OTG interrupt status register, indicating the HNP status. The application must read the host negotiation success bit in the OTG control and status register to determine host

negotiation success. The application must read the current Mode bit in the core interrupt register (OTG_GINTSTS) to determine host mode operation.

3. The application sets the reset bit (PRST in OTG_HPRT) and the OTG_FS controller issues a USB reset and enumerates the A-device for data traffic.
4. The OTG_FS controller continues the host role of initiating traffic, and when done, suspends the bus by writing the port suspend bit in the host port control and status register.
5. In Negotiated mode, when the A-device detects a suspend, it disconnects and switches back to the host role. The OTG_FS controller deasserts the DP pull down and DM pull down in the PHY to indicate the assumption of the device role.
6. The application must read the current mode bit in the core interrupt (OTG_GINTSTS) register to determine the host mode operation.
7. The OTG_FS controller connects, completing the HNP process.

64 USB Type-C®/USB Power Delivery interface (UCPD)

64.1 Introduction

The USB Type-C/USB Power Delivery interface complies with:

- Universal Serial Bus Type-C Cable and Connector Specification: release 2.1, May 2021
- Universal Serial Bus Power Delivery specifications:
  - revision 2.0, version 1.3, January 12, 2017
  - revision 3.1, version 1.3, January 2022

It integrates the physical layer of the Power Delivery (PD) specification, with CC signaling method (no VBUS), for operation with Type-C cables.

64.2 UCPD main features

- Compliance with USB Type-C specification release 2.0
- Compliance with USB Power Delivery specifications revision 2.0 and 3.1
  - 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

64.3 UCPD implementation

The devices have one UCPD controller to support one USB Type-C port.

Table 653. UCPD implementation(1)

UCPD feature	STM32U575/585 Rev. X	Other STM32U5xx device variants
Dead battery support via UCPD1_DBCC1 and UCPD1_DBCC2 external signals	X	X
UCPD1_FRSTX as alternate function pin	X	X
Fully automatic trimming	X(2)	-(3)
USB PD receiver hardware filter control	X	X
Discrete component PHY support	-	-

1. "X" = supported, "-" = not supported
2. No software trimming required.
3. Apply software trimming as described in *Section 64.5.5: UCPD software trimming*.

The following table gives the memory locations of trim data stored in the non-volatile memory, to use in the software trimming procedure described in *Section 64.5.5: UCPD software trimming*.

Table 654. UCPD software trim data

Name	Non-volatile memory location	
	Address	Bits
3A0_CC1[3:0]	0x0BFA0545	3:0
3A0_CC2[3:0]	0x0BFA0547	3:0
1A5_CC1[3:0]	0x0BFA07A7	3:0
1A5_CC2[3:0]	0x0BFA07A8	3:0
Rd_CC1[3:0]	0x0BFA0544	3:0
Rd_CC2[3:0]	0x0BFA0546	3:0

64.4 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".

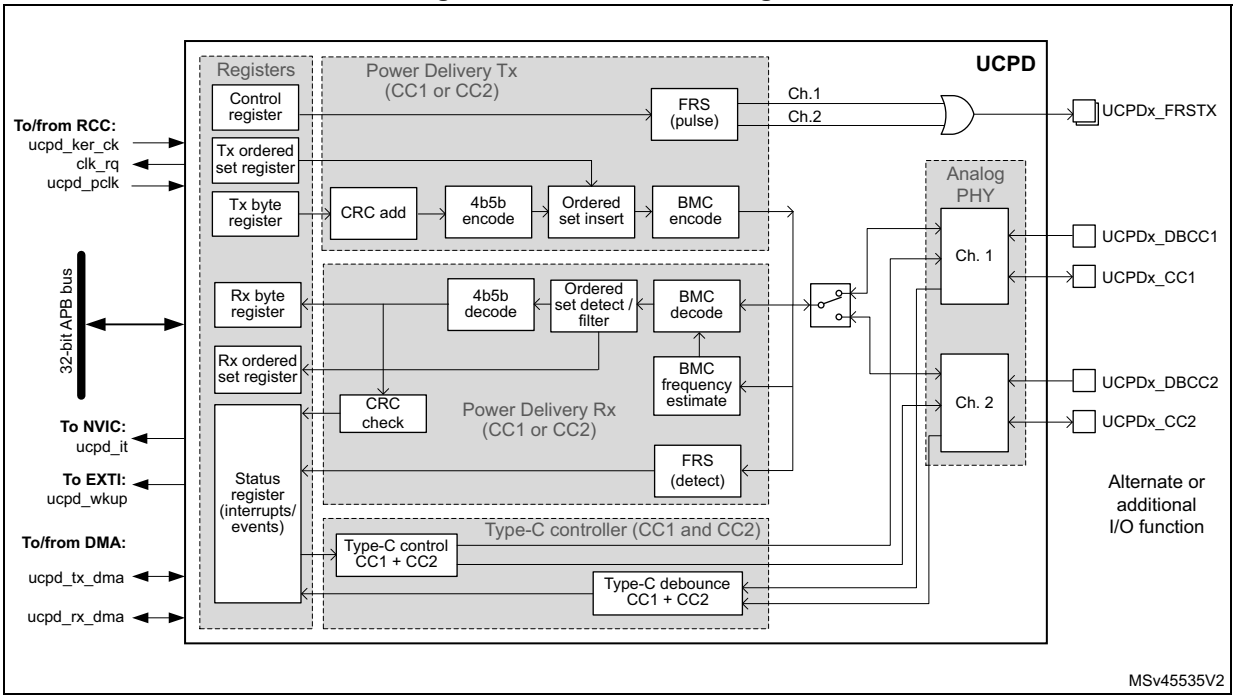
The UCPD receiver detects SOP, BMC-decodes the incoming stream, recovers the preamble, 4b5b-decodes payload data, detects EOP, and checks CRC. It automatically detects five K-code SOP and two Reset ordered sets, plus two software-defined patterns

(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.

64.4.1 UCPD block diagram

Figure 797. UCPD block diagram



The following table lists external signals (alternate or additional I/O functions).

Table 655. 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.

Table 655. UCPD signals on pins (continued)

Pin name	Signal type	Description
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 656. 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

64.4.2 UCPD reset and clocks

The peripheral has a single reset signal (APB bus reset).

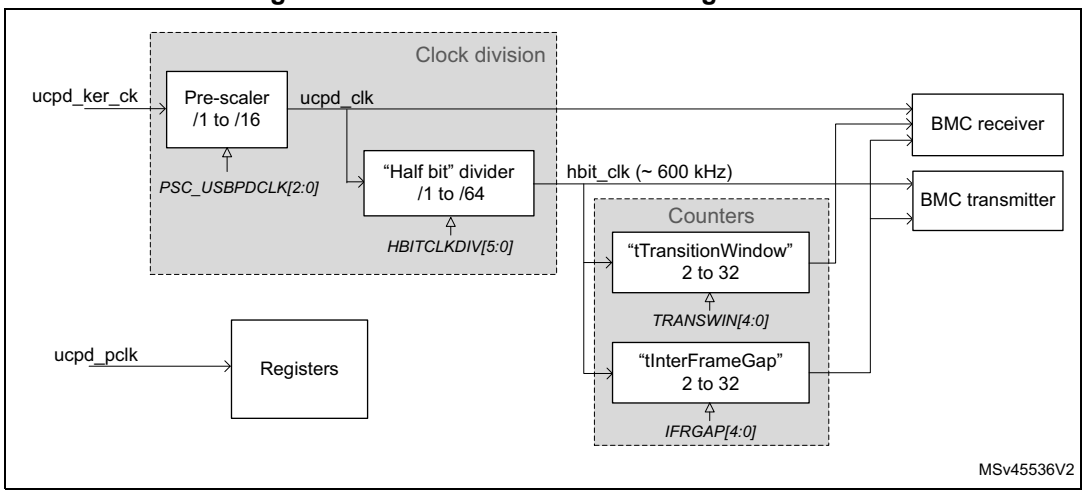
The register section is clocked with the APB clock (ucpd_pclk).

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 798. 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.

64.4.3 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.

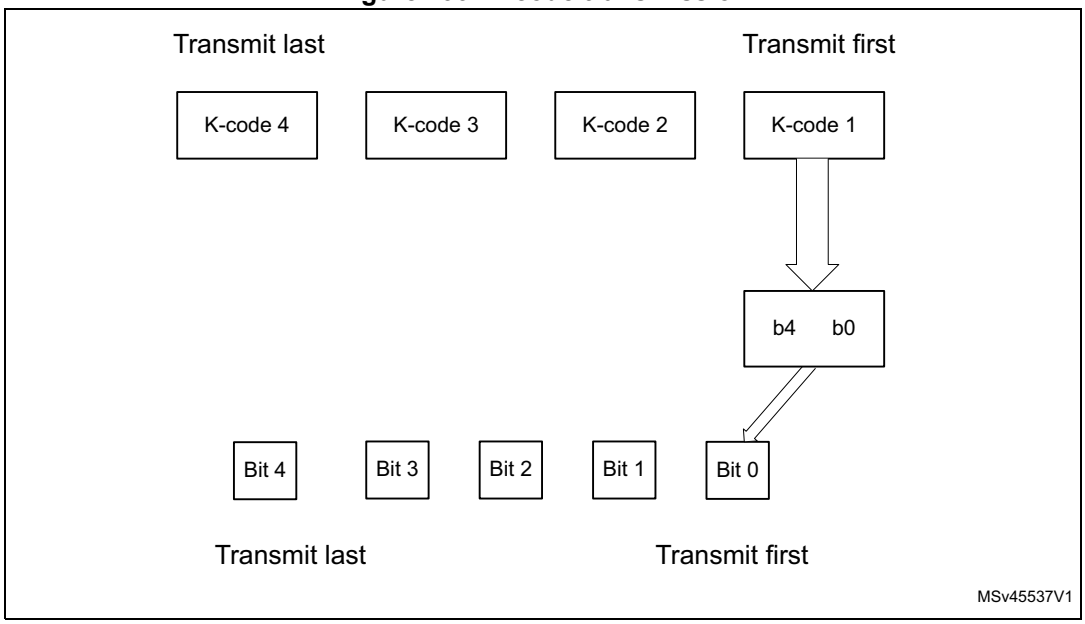
Table 657. 4b5b symbol encoding table

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
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 799. 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 ongoing Tx message.

Table 658. 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 659. 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 659. 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

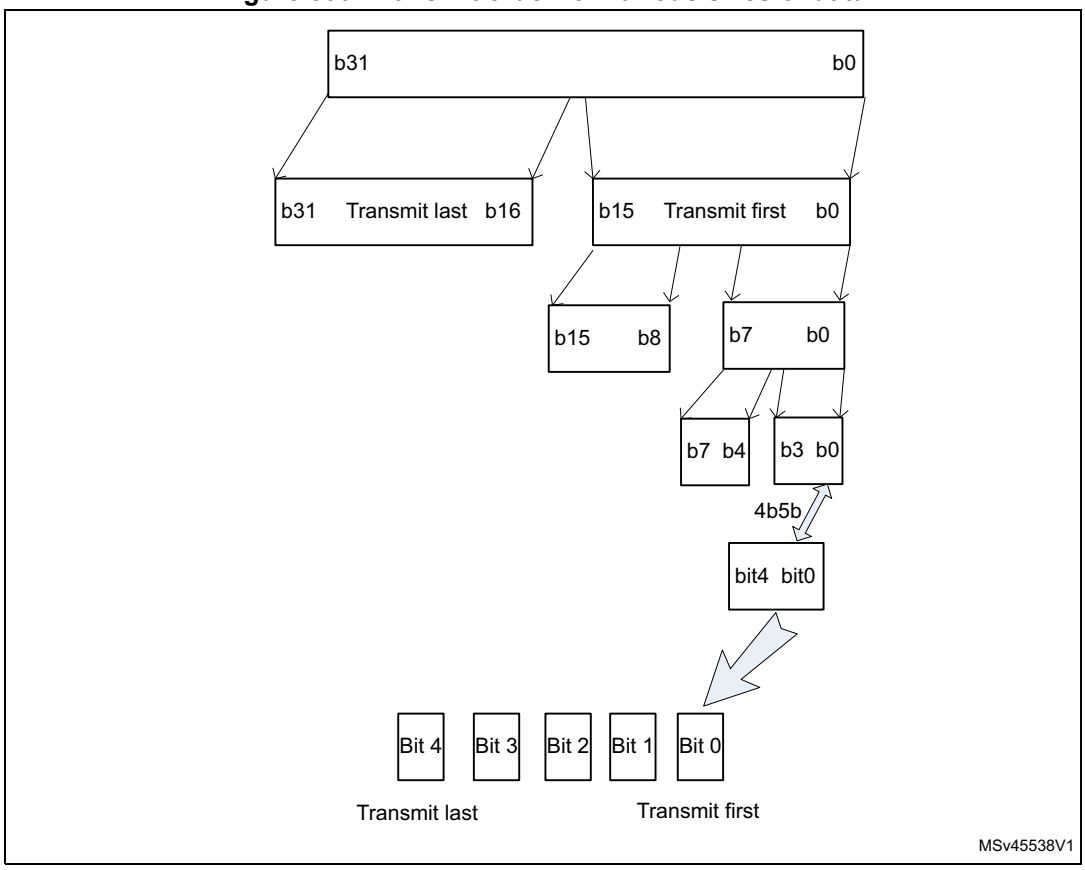
Allowed transmission data units / data sizes are in the following table.

Table 660. 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 800. Transmit order for various sizes of data

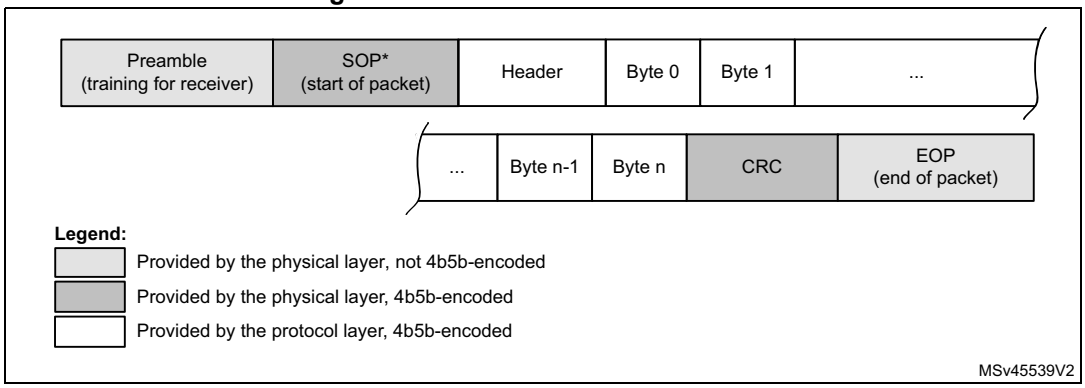


Packet format

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 801. Packet format



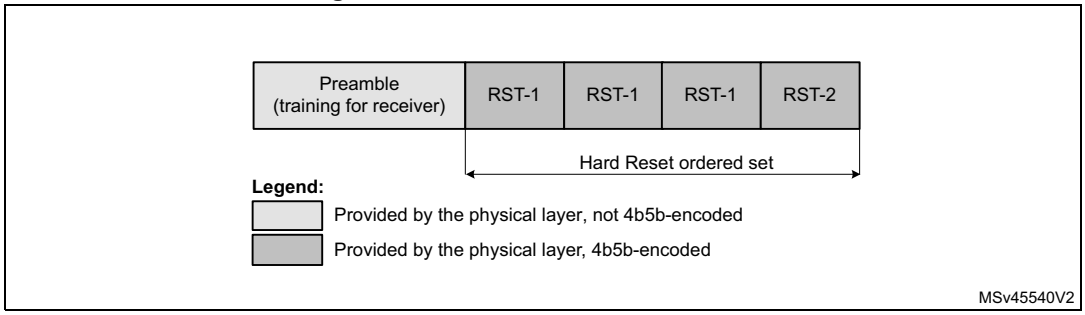
Hard Reset

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 ongoing 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.

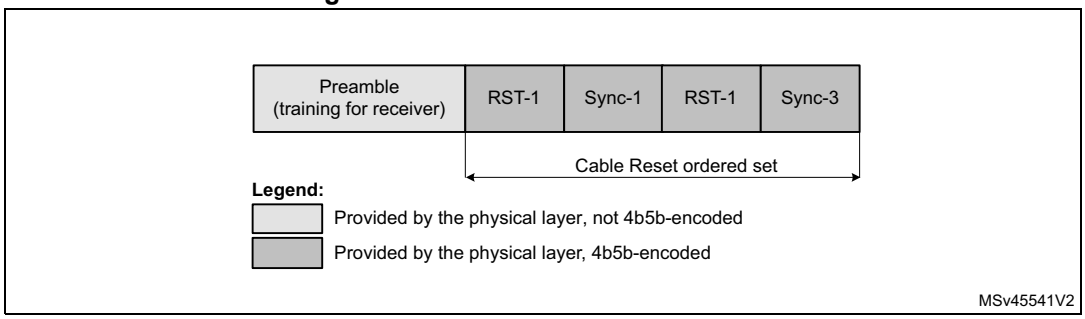
Figure 802. Line format of Hard Reset



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 803. 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 3.1 also requires to manage the Rp 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:

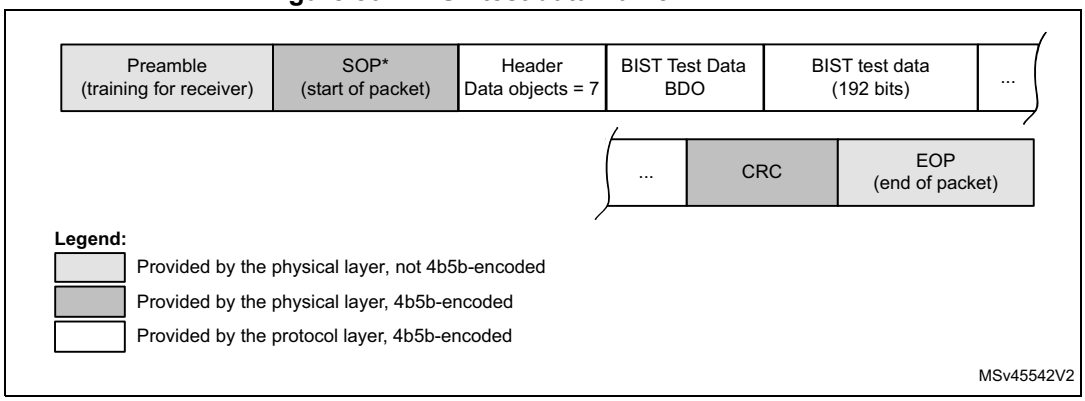
- BIST Test Data (192 bit pattern), applies to Tx and Rx. In the case of Rx, the message is received (but discarded rather than passing to the protocol layer, which must nevertheless still generate a GoodCRC Tx message in acknowledgment).
- BIST Carrier Mode 2 (single pattern, infinite length message), 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 804. BIST test data frame



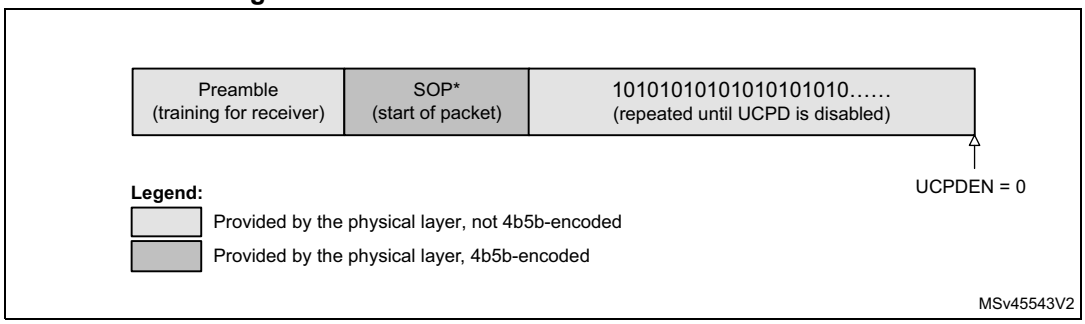
This is a fixed length test data pattern. In reality the only aspect that marks its difference from the general packet format already shown in *Figure 801: Packet format* is the contents of the Header. As UCPD receives the Tx Header contents via programming (it is simply viewed as part of the payload), it is only this programming (and not the block's behavior) 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.

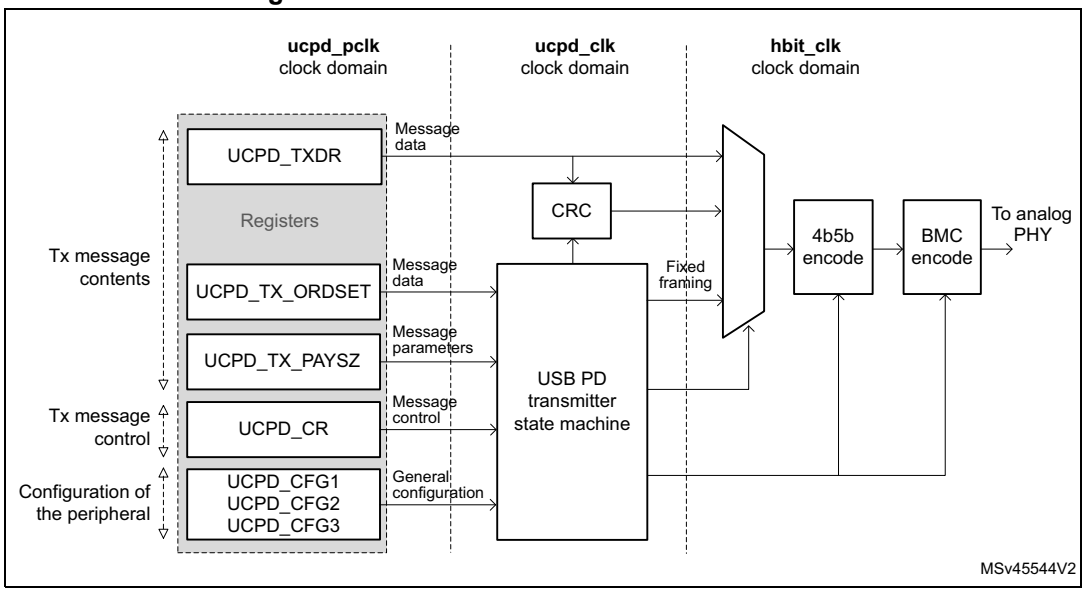
Figure 805. BIST Carrier Mode 2 frame



64.4.4 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 806. 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 ongoing 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.

64.4.5 UCPD BMC receiver

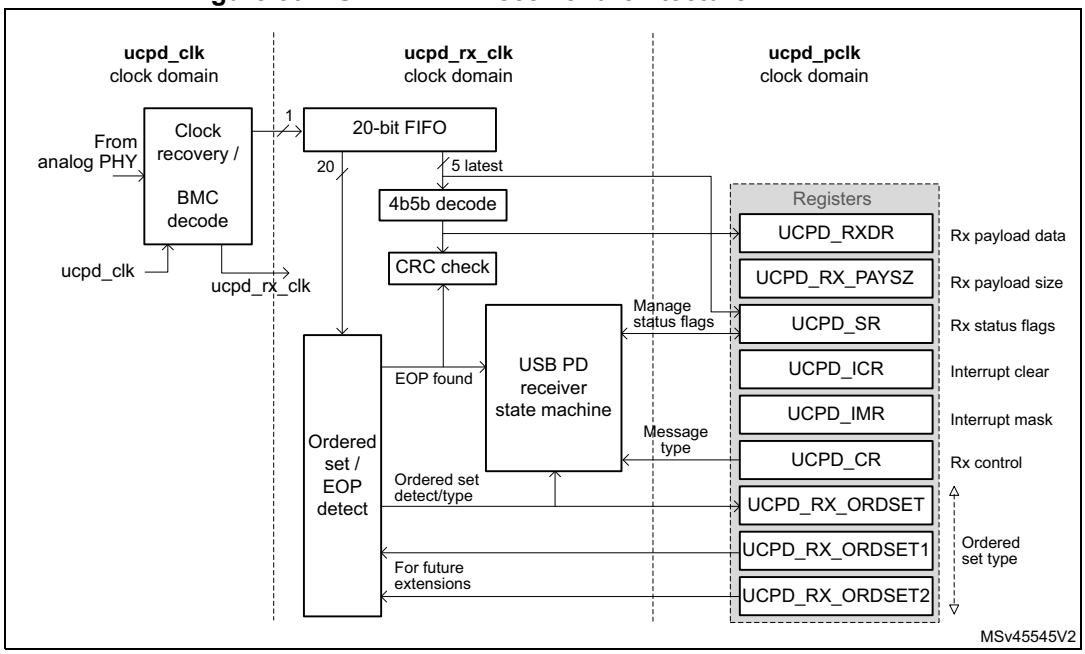
The UCPD BMC receiver performs:

- 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 807. 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.

64.4.6 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.

64.4.7 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.

64.4.8 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.

64.4.9 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.

64.4.10 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

64.5 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.

64.5.1 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.
3. Enable the analog Rx filter of either CC line, via the RXAFILTEN bit of the UCPD_CFGR2 register.

64.5.2 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 661. 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 defines 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, 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 behavior 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 662. Type-C sequence (source: 3A); cable/sink connected (Rd on CC1; Ra on CC2)

Type-C state	ANAMODE; ANASUBMODE[1:0]	CCENABLE	PHYCCSEL	RDCH	CC[x] VCONNEN(1)	Event => go to next line	Comments
Unattached. SRC	0:Source; 11:Rp3A0	11:both enabled	0 (don't care)	0: [Normal]	00: [neither]	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]

1. Two GPIOs to enable VCONN through external load switch components

64.5.3 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

64.5.4 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.

64.5.5 UCPD software trimming

The CC pull-up (Rp) and pull-down (Rd) devices must be trimmed on each part, to meet the required accuracy. The trimming values are saved in the non-volatile memory.

To trim the CC pull-up and pull-down devices by software, apply the following procedure:

1. Retrieve the trim values from the non-volatile memory (refer to *Table 654: UCPD software trim data*)
2. At initialization, write the trim values to the UCPD_CFGR3 register bitfields as follows:
   - 3A0_CC1[3:0] to TRIM_CC1_RP[3:0]
   - 3A0_CC2[3:0] to TRIM_CC2_RP[3:0]
   - Rd_CC1[3:0] to TRIM_CC1_RD[3:0]
   - Rd_CC2[3:0] to TRIM_CC2_RD[3:0]
3. At each setting of ANASUBMODE to 1A5 or 3A0, respectively, write the trimming values to the UCPD_CFGR3 register bitfields as follows:
   - 1A5_CC1[3:0] or 3A0_CC1[3:0], respectively, to TRIM_CC1_RP[3:0]
   - 1A5_CC2[3:0] or 3A0_CC2[3:0], respectively, to TRIM_CC2_RP[3:0]

64.6 UCPD low-power modes

A summary of low-power modes is shown below in *Table 663: Effect of low power modes on the UCPD*.

Table 663. Effect of low power modes on the UCPD

Mode	Description
Sleep	No effect
Stop	Detection of events (Type-C, BMC Rx, FRS detection) remains operational and can wake up the MCU.
Standby	UCPD is not operating, and cannot wake up the MCU. Pull-downs remain active if configured.
Unpowered	Dead battery pull-downs remain active.

The UCPD is able to wakeup the MCU from Stop mode when it recognizes a relevant event, either:

- Type-C event relating to a change in the voltage range seen on either of the CC lines, visible in TYPEC_VSTATE_CCx
- Power delivery receive message with an ordered set matching those filtered according to RXORDSETEN[8:0], visible by reading RXORDSET

Wakeup from Stop mode is enabled by setting the WUPEN bit in the UCPD_CFG2 register.

At UCPD level three types of event requiring kernel clock activity may occur during Stop mode:

- Activity on the analog PHY voltage threshold detectors which could later be confirmed to be a stable change between voltage ranges defined in the Type-C specification
- Activity on Power Delivery BMC receiver (coming from the selected CC line) which could potentially generate an Rx message event (that is, RXORDSET) later
- Activity on Power Delivery FRS detector which could potentially generate an FRS signaling detection event (that is, FRSEVT) later

It order to function correctly with the RCC, the clock request signal is activated (conditional on WUPEN) when there is asynchronous activity on:

- Type-C voltage threshold detectors (coming from either CC line)
- Power Delivery receiver signal (from the selected CC line)
- FRS detection signal (from the selected CC line)

64.7 UCPD interrupts

The table below lists the UCPD event flags, with the associated flag clear bits and interrupt enable bits.

Table 664. 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

Table 664. UCPD interrupt requests (continued)

Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit
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.

64.8 UCPD registers

64.8.1 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

64.8.2 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.	RXAFILTEN	Res.	Res.	Res.	Res.	WUPEN	FORCECLK	RXFILT2N3	RXFILTDIS
							rw					rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **RXAFILTEN**: Rx analog filter enable
Setting the bit enables the Rx analog filter required for optimum Power Delivery reception.
0: Disable
1: Enable

Bits 7:4 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).

64.8.3 UCPD configuration register 3 (UCPD_CFGR3)

Address offset: 0x008

Reset value: 0x0000 0000

Configuration of UCPD trimming of the CC pull-up and pull-down devices. The trimming is managed by hardware until the first software write into this register.

The register is reserved (must not be written) for devices that support the fully automatic trimming. Refer to *Table 653: UCPD implementation*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TRIM_CC2_RP[3:0]				Res.	Res.	Res.	Res.	Res.	TRIM_CC2_RD[3:0]			
			rw	rw	rw	rw						rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TRIM_CC1_RP[3:0]				Res.	Res.	Res.	Res.	Res.	TRIM_CC1_RD[3:0]			
			rw	rw	rw	rw						rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:25 **TRIM_CC2_RP[3:0]**: SW trim value for Rp current sources on the CC2 line

Bits 24:20 Reserved, must be kept at reset value.

Bits 19:16 **TRIM_CC2_RD[3:0]**: SW trim value for Rd resistor on the CC2 line

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:9 **TRIM_CC1_RP[3:0]**: SW trim value for Rp current sources on the CC1 line

Bits 8:4 Reserved, must be kept at reset value.

Bits 3:0 **TRIM_CC1_RD[3:0]**: SW trim value for Rd resistor on the CC1 line

64.8.4 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 3.1.

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 661: 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 661: 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. As this mode prevents reception of the header (containing MessageID), software has to auto-increment a received MessageID counter for inclusion in the GoodCRC acknowledge that must still be transmitted during this test.

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).

64.8.5 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.
											r				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPECEVT2IE	TYPECEVT1IE	Res.	RXMSGENDIE	RXOVRIE	RXHRSTDETIE	RXORDDETIE	RXNEIE	Res.	TXUNDIE	HRSTSENTIE	HRSTDISCIE	TXMSGABTIE	TXMSGSENTIE	TXMSGDISCIE	TXISIE
rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **FRSEVTIE**: FRSEVT interrupt enable
0: Disable
1: Enable

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **TYPECEVT2IE**: TYPECEVT2 interrupt enable
0: Disable
1: Enable

Bit 14 **TYPECEVT1IE**: TYPECEVT1 interrupt enable

Bit 13 Reserved, must be kept at reset value.

Bit 12 **RXMSGENDIE**: RXMSGEND interrupt enable
0: Disable
1: Enable

Bit 11 **RXOVRIE**: RXOVR interrupt enable
0: Disable
1: Enable

Bit 10 **RXHRSTDETIE**: RXHRSTDET interrupt enable
0: Disable
1: Enable

Bit 9 **RXORDDETIE**: RXORDDET interrupt enable
0: Disable
1: Enable

Bit 8 **RXNEIE**: RXNE interrupt enable
0: Disable
1: Enable

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TXUNDIE**: TXUND interrupt enable
0: Disable
1: Enable

Bit 5 **HRSTSENTIE**: HRSTSENT interrupt enable
0: Disable
1: Enable

Bit 4 **HRSTDISCIE**: HRSTDISC interrupt enable
0: Disable
1: Enable

Bit 3 **TXMSGABTIE**: TXMSGABT interrupt enable
0: Disable
1: Enable

Bit 2 **TXMSGSENTIE**: TXMSGSENT interrupt enable
0: Disable
1: Enable

Bit 1 **TXMSGDISCIE**: TXMSGDISC interrupt enable
0: Disable
1: Enable

Bit 0 **TXISIE**: TXIS interrupt enable
0: Disable
1: Enable

64.8.6 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

64.8.7 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.

64.8.8 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.

64.8.9 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)

64.8.10 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

64.8.11 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

64.8.12 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).

64.8.13 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

64.8.14 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.

64.8.15 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.

64.8.16 UCPD register map

Table 665. UCPD 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	UCPD_CFGR1	UCPDEN	RXDMAEN	TXDMAEN	RXORDSETEN[8:0]									PSC_USBPDCLK[2:0]			Res.	TRANSWIN[4:0]					IFRGAP[4:0]					HBITCLKDIV[5:0]						
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	UCPD_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXAFILTEN	Res.	Res.	Res.	Res.	WUPEN	FORCECLK	RXFILT2N3	RXFILTDIS
	Reset value																								0					0	0	0	0
0x008	UCPD_CFGR3	Res.	Res.	Res.	TRIM_CC2_RP[3:0]				Res.	Res.	Res.	Res.	Res.	TRIM_CC2_RD[3:0]				Res.	Res.	Res.	Res.	TRIM_CC1_RP[3:0]			Res.	Res.	Res.	Res.	TRIM_CC1_RD[3:0]			
	Reset value				0	0	0	0						0	0	0	0				0	0	0	0						0	0	0	0
0x00C	UCPD_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2TCDIS	CC1TCDIS	Res.	RDCH	FRSTX	FRSRXEN	Res.	Res.	Res.	Res.	CCENABLE[1:0]		ANAMODE	ANASUBMODE[1:0]		PHYCCSEL	PHYRXEN	RXMODE	TXHRST	TXSEND	TXMODE[1:0]	
	Reset value											0	0		0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
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

Table 665. UCPD 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
0x018	UCPD_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRSEVTCF	Res.	Res.	Res.	Res.	TYPECEVT2CF	TYPECEVT1CF	Res.	RXMSGENDCF	RXOVRCF	RXHRSTDETCF	RXORDDETCF	Res.	Res.	TXUNDCF	HRSTSENTCF	HRSTDISCCF	TXMSGABTCF	TXMSGSENTCF	TXMSGDISCCF	Res.
	Reset value												0					0	0		0	0	0	0			0	0	0	0	0	0	
0x01C	UCPD_TX_ORDSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXORDSET[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	UCPD_TX_PAYSZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXPAYSZ[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x024	UCPD_TXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x028	UCPD_RX_ORDSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPKINVALID[2:0]			RXSOP3OF4	RXORDSET[2:0]		
	Reset value																										0	0	0	0	0	0	0
0x02C	UCPD_RX_PAYSZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXPAYSZ[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x030	UCPD_RXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x034	UCPD_RX_ORDEXTR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPX1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	UCPD_RX_ORDEXTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPX2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C - 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 115* for the register boundary addresses.

65 Debug support (DBG)

65.1 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)

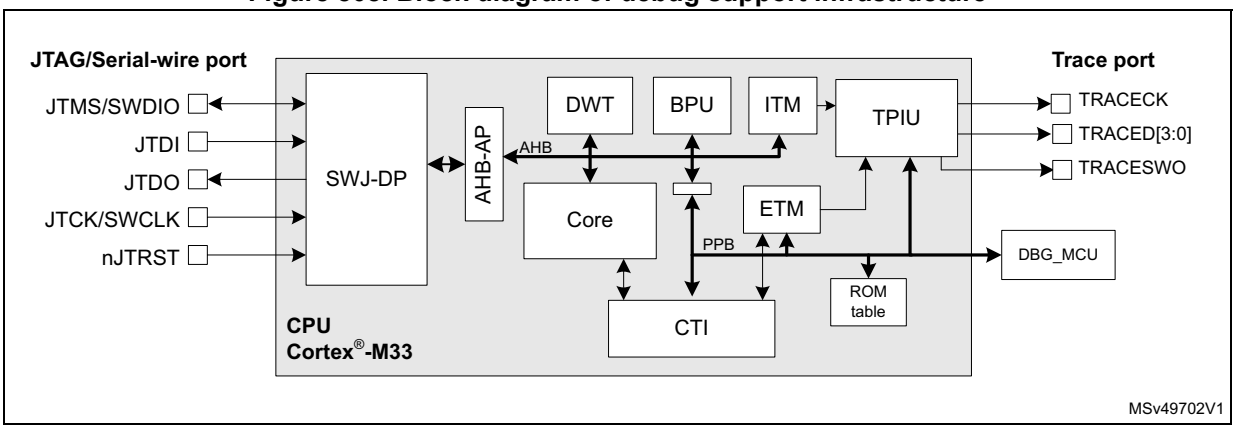
The debug features are accessible by the debugger via the AHB-AP.

Additional information can be found in the Arm documents referenced in *Section 65.13*.

65.2 DBG functional description

65.2.1 DBG block diagram

Figure 808. Block diagram of debug support infrastructure



65.2.2 DBG pins and internal signals

Table 666. 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 can help to decide whether the pad is used as TDI or as CC.

Table 667. Trace port pins

Pin name	Type	Description	Pin assignment
TRACED0	O	Trace synchronous data out 0	Refer to the datasheet
TRACED1		Trace synchronous data out 1	
TRACED2		Trace synchronous data out 2	
TRACED3		Trace synchronous data out 3	
TRACECK		Trace clock	

Table 668. 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.

65.2.3 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. This clock 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, that 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 *DP control and status register (DP_CTRL/STATR)*. 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 DP_CTRL/STATR. 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.

65.2.4 DBG power domains

The debug components are located in the core power domain. This means that the debugger connection is not possible in Shutdown or Standby low-power mode. To avoid losing the connection when the device enters Standby mode, the power can be maintained to the core by setting a bit in the *DBGMCU configuration register (DBGMCU_CR)*. This also keeps the processor clocks active and holds off the reset, so that the debug session is maintained.

65.2.5 Debug and low-power modes

The 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 if the core is not clocked, all debug components are inaccessible to the debugger. To avoid this, power-saving mode emulation is implemented. If the 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.

The emulation mode is programmed in the microcontroller debug (DBGMCU) unit. For more information refer to *Section 65.12: Microcontroller debug unit (DBGMCU)*.

65.2.6 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.

The following authentication signals are used by the system to determine which debug features are enabled or disabled:

- **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 readout protection (RDP) level (see *Section 7.6.2: Readout protection (RDP)*), as shown in the table below:

Table 669. 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. The 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. The debugger access to secure memory is disabled.

Table 669. 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. The 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 the 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 memories 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.

The 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 control/status word register (AP_CSWR)*.

The debugger access is disabled while the processor is booting from system Flash memory (RSS), whatever the RDP level, if security features are enabled (TZEN = 1).

65.3 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)

These 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.

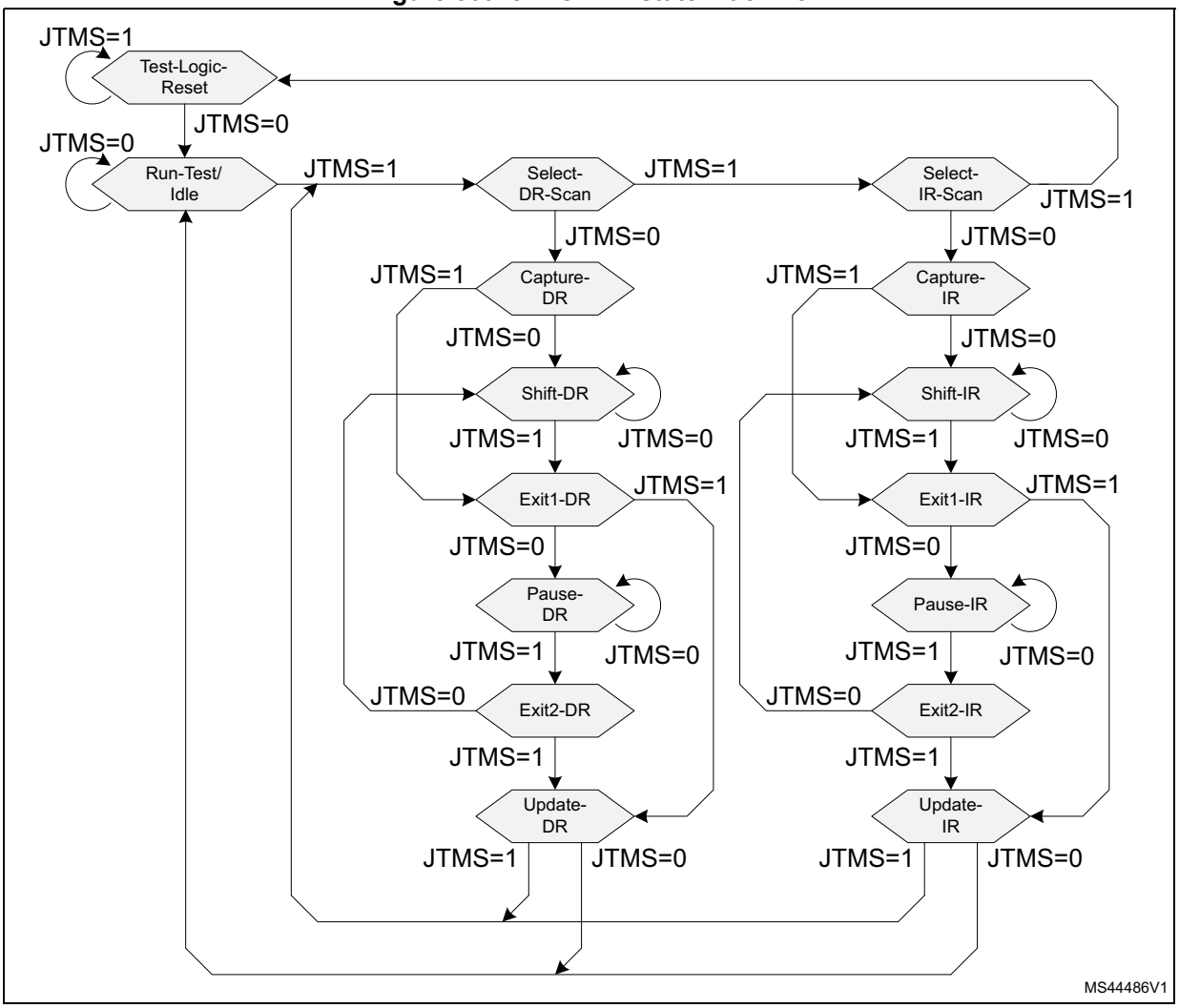
In SW-DP mode, the unused JTAG pins JTDI, JTDO and nJTRST can be used for other functions.

Note: All SWJ port I/Os can be reconfigured to other functions by software, but debugging is no longer possible.

65.3.1 JTAG debug port

The JTAG-DP implements a TAP state machine (TAPSM), shown in the figure below, 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 809. 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 to 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 to the instruction register.
4. When the TAPSM goes through the Capture-DR state, a value is transferred from one of the data registers to 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, are selected.
6. This data is then shifted while the TAPSM is in the Shift-DR state, in the same manner as the IR shifts 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 to 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 the table below.

Table 670. 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 670. 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 (see *Section 65.13*).

65.3.2 Serial-wire debug port

The serial-wire debug protocol uses the following pins:

- SWCLK: clock from host to target
- SWDIO: bi-directional serial data

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 671*)
2. acknowledge response (3 bits) transmitted by the target (see *Table 672*)
3. data transfer (33 bits) transmitted by the host (in case of a write) or target (in case of a read) (see *Table 673*)

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 671. Packet request

Bit field	Name	Description
0	Start	Must be 1.
1	APnDP	– 0: DP register access - see *Section 65.3.3: Debug port registers* – 1: AP register access - see *Section 65.4: 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 674* or *Table 675*)
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 672. ACK response

Bit field	Name	Description
2:0	ACK	– 000: FAULT – 010: WAIT – 100: OK

Table 673. 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.

65.3.3 Debug port registers

Both the SW-DP and the JTAG-DP access the debug port (DP) registers listed in *Table 674*.

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 671*). 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 65.2.5: Debug and low-power modes*.
0: DCLK gated
1: DCLK enabled

Bit 28 **CDBGPWRUPREQ**: debug power-up request
This bit 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
This bit 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
This bit 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)
This bit indicates that an error occurred in an AP transaction.It 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).
This bit indicates that an overrun occurred (new transaction received before previous transaction completed). This bit is only set if the ORUNDETECT bit is set. it 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: 0xXXXX 0041

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 A, Y, Z
0x2: revision B, X
0x3: revision C

Bits 27:12 **TPARTNO[15:0]**: target part number
0x4820: STM32U575/585

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
this field 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[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w	w	w	w	w	w	w	w								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBANKSEL[3:0]				DPBANKSEL[3:0]			
								w	w	w	w	w	w	w	w

Bits 31:24 **APSEL[7:0]**: access port select
This field selects the access port for the next transaction.
0x0: AP0 - Cortex-M33 debug access port (AHB-AP)
others: reserved

Bits 23:8 Reserved, must be kept at reset value.

Bits 7:4 **APBANKSEL[3:0]**: AP register bank select
This field selects the 4-word register bank on the active AP for the next transaction.

Bits 3:0 **DPBANKSEL[3:0]**: DP register bank select
This field 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.

Debug port register map

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 674. 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

Table 674. 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
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	Res.	DAPABORT
	Reset value																												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	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	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[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
0x8	DP_SELECTR	APSEL[7:0]								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	x	x	x	x
0xC	DP_RDBUFFR	RDBUFF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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.

65.4 Access ports

There is one access port (AP) attached to the DP. It enables the access to the debug and trace features integrated in the Cortex-M33 processor core via its internal AHB bus.

65.4.1 Access port registers

The access port is of MEM-AP type: 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 675*.

The address of the AP registers is composed of the following fields:

- bits [7:4]: content of the APBANKSEL[3:0] field in the *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 670*), or of the SW-DP packet request (see *Table 671*), depending on the debug interface used
- bits [1:0]: always set to 0

The content of the DP_SELECTR.APSEL[3:0] field defines which MEM-AP is being accessed.

The debugger can access the AP registers as follows:

1. Program the APSEL[3:0] field in the *DP access port select register (DP_SELECTR)* to choose the AP, and the APBANKSEL[3:0] field in DP_SELECTR to select the register bank to be accessed.
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 671*). 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 *AP transfer address register (AP_TAR)*.
2. Program the *AP control/status word register (AP_CSWR)*, if necessary, with the transfer parameters (AddrInc for example).
3. Write to or read from the *AP data read/write register (AP_DRWR)* to initiate a bus transaction at the address held in AP_TAR. Alternatively, a read or write to the *AP banked data n register (AP_BDnR)* triggers an access to TAR[31:4] + n address, 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

Access port register map

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 675. 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																															

Table 675. 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
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
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

65.5 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, as well as for the MCU debug unit.

The MCU ROM table (see the table below) occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00F E000 to 0xE00F EFFC.

Table 676. 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	DBGMCU	0xE004 4000	0xFFF4 6000	4	0xFFF4 6003

Table 676. MCU ROM table (continued)

Address in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0xE00F E00C	Reserved	-	-	-	0x1FF0 2002
0xE00F E010	Top of table	-	-	-	0x0000 0000
0xE00F E014 to 0xE00F EFC8	Reserved	-	-	-	0x0000 0000
0xE00F EFCC to 0xE00F EFFC	ROM table registers	-	-	-	See *Table 678*

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.

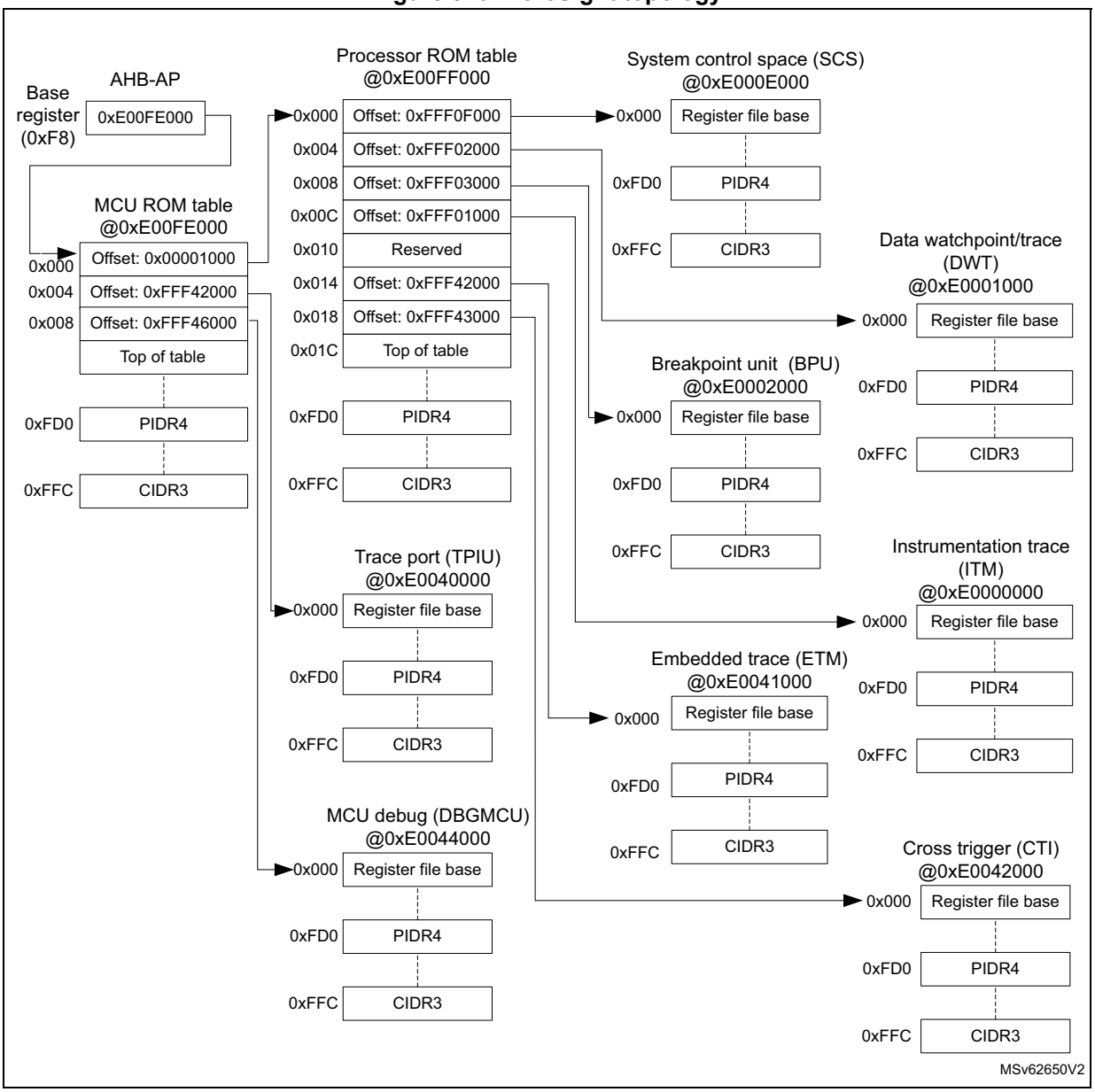
The processor ROM table (see the table below) occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00F F000 to 0xE00F FFFC.

Table 677. 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 679*

The topology for the CoreSight components in the Cortex -M33 is shown in the figure below.

Figure 810. CoreSight topology



65.5.1 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.	SIZE[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 **SIZE[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 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.	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]
0x82: STM32U575/585

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: STM32U575/585

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

MCU ROM table register map

Table 678. 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.	SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4 to FDC	Reserved	Reserved																															

Table 678. MCU ROM table register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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																									X	X	X	X	X	X	X	X
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 *Table 676: MCU ROM table* for register boundary addresses.

65.5.2 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.	SIZE[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 **SIZE[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

Processor ROM table register map

Table 679. 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.	SIZE[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

Table 679. CPU ROM table register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFFC	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 *Table 677: Processor ROM table* for register boundary addresses.

65.6 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 (COMP 1 only)
- clock cycle counter comparator (COMP 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 comparator x register (DWT_COMPxR)* with one of the following:

- a data address
- an instruction address
- a data value
- the cycle-count value, for COMP 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 Armv8-M Architecture Reference Manual *[3]*.

65.6.1 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 *[3]*.

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 *[3]*.

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 *[3]*

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 *[3]*

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.	SIZE[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 **SIZE[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

DWT register map

The DWT registers are located at address range 0xE000 1000 to 0xE000 1FFC.

Table 680. 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	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
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																															

Table 680. 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
0x028	DWT_FUNCTR0	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	1			0													0	0					0	0	0	0	0	0
0x02C	Reserved	Reserved																															
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]					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	0	1	0			0													0	0					0	0	0	0	0	0
0x03C	Reserved	Reserved																															
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																															
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
0x04C	Reserved	Reserved																															
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
0xFCC	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.	SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															

Table 680. 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
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
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	0	0	1	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 *Table 677: Processor ROM table* for register boundary addresses.

65.7 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 *[3]*.

65.7.1 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.	SIZE[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 **SIZE[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

ITM register map

The ITM registers are located at address range 0xE000 0000 to 0xE000 0FFC.

Table 681. 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_STIMR0 to ITM_STIMR31	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

Table 681. 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
0xE04 to 0xE3C	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]											PRESEN	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.	SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															
0xFE0	ITM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	ITM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	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 *Table 677: Processor ROM table* for register boundary addresses.

65.8 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 *[3]*.

65.8.1 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 14:12, 7:4 **NUM_CODE[6:0]**: number of instruction address comparators supported
0x08: 8 instruction comparators supported

Bits 11:8, 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.	SIZE[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 **SIZE[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

BPU register map

The BPU registers are located at address range 0xE000 2000 to 0xE000 2FFC.

Table 682. 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																															

Table 682. 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
0x008 to 0x024	BPU_COMP0R to BPU_COMP7R	BPADDR[31:1]																															BE
	Reset value	0	0	0	0	0	0	0	0	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]											PRESEN	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	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.	SIZE[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
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 *Table 677: Processor ROM table* for register boundary addresses.

65.9 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

STM32U575/585, 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 *[5]*.

65.9.1 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 behavior
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[3:0].
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.	SIZE[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 **SIZE[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

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 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: 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

ETM register map

The ETM registers are accessed by the debugger at address range 0xE0041000 to 0xE0041FFC.

Table 683. 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.	PMSTABLE	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																															

Table 683. 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
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.	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																															
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																															

Table 683. 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
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
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.	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.	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	STATUS	Res.	Res.	Res.	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																															

Table 683. 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
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																															
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	Res.	Res.	Res.
	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
0xFA8 to 0xFB4	Reserved	Reserved																															
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]											PRESEN	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	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.	SIZE[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
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

Table 683. ETM register map and reset values (continued)

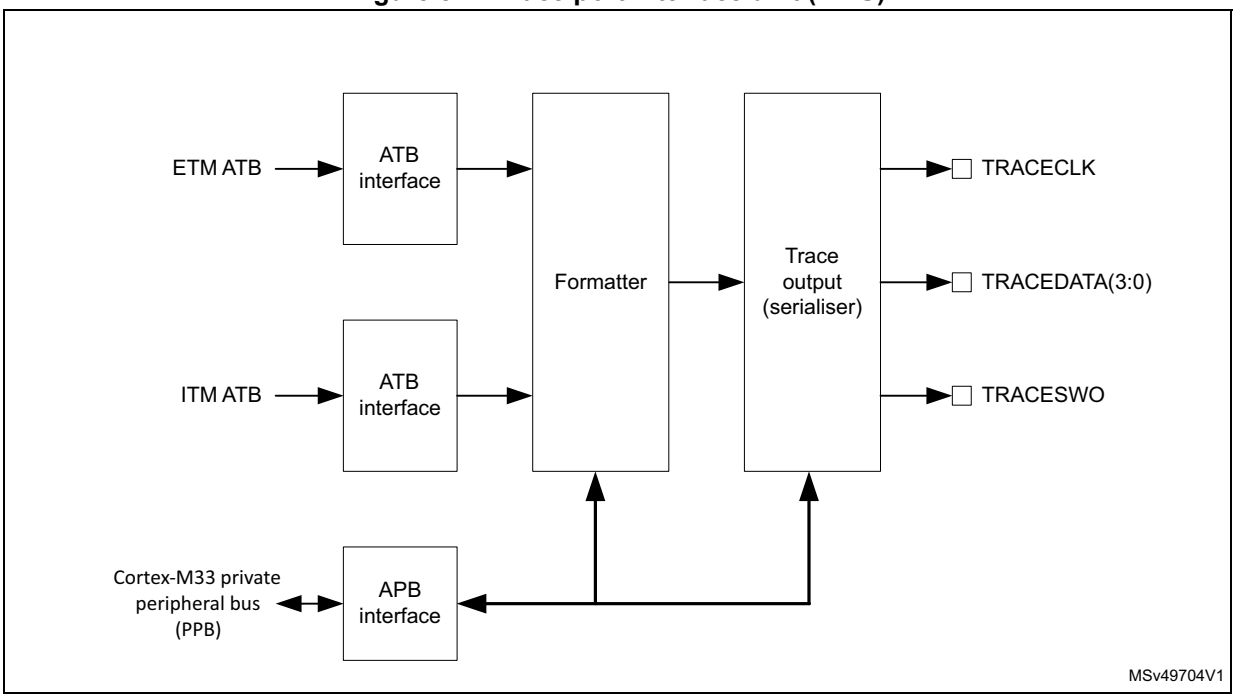
Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF0	ETM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									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[3:0]				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 *Table 677: Processor ROM table* for register boundary addresses.

65.10 Trace port interface unit (TPIU)

The TPIU formats the trace stream and outputs it on the external trace port signals. As shown in the figure below, 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 811. 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 *[4]*.

65.10.1 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

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.	SIZE[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 **SIZE[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

TPIU register map

Table 684. 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.	FTNONSTOP	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																															

Table 684. TPIU register map and reset values (continued)

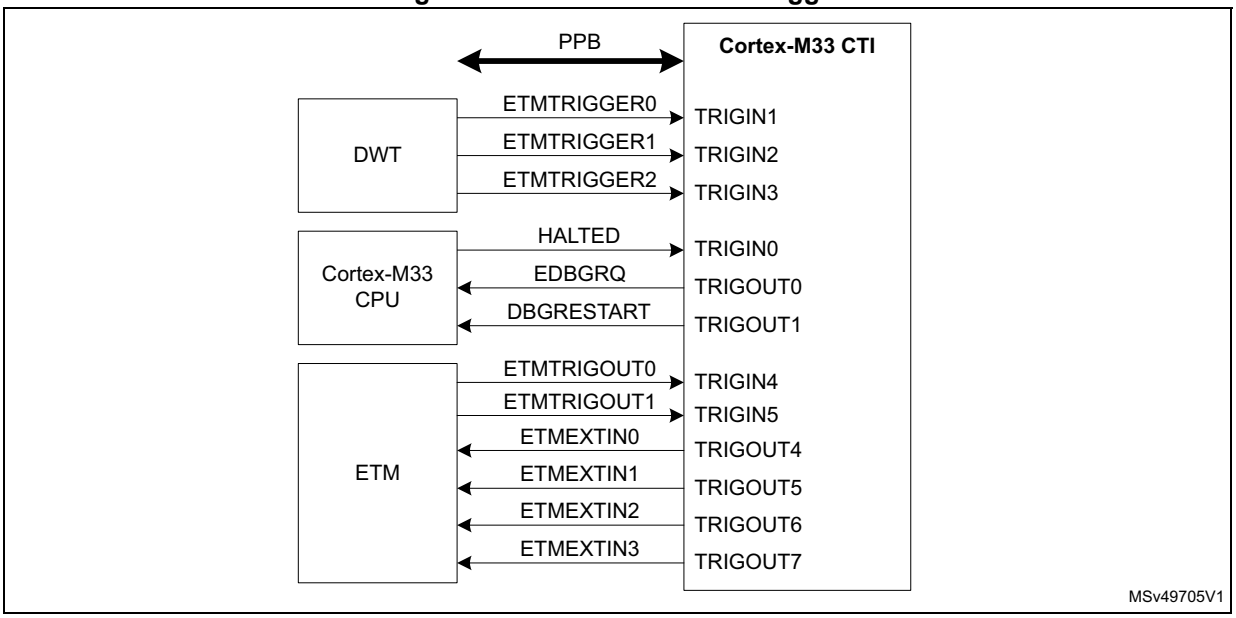
Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFA0	TPIU_CLAIMSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET [3:0]			
	Reset value																													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.	SWOUARTNRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MaXNUM[4:0]				
	Reset value																					1	1	0	0	1	0	1	0	0	0	0	1
0xFCC	TPIU_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]				MAJORTYPE [3:0]			
	Reset value																									0	0	0	1	0	0	0	1
0xFD0	TPIU_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.	SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFE0	TPIU_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	TPIU_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	TPIU_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	TPIU_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	TPIU_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	TPIU_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	TPIU_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	TPIU_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 676: MCU ROM table* for register boundary addresses.

65.11 Cross-trigger interface (CTI)

The CTI allows cross triggering between the processor and the ETM (see the figure below).

Figure 812. 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 the tables below.

Table 685. 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 686. 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 686. CTI outputs (continued)

Number	Source signal	Destination component	Comments
2	-	-	Not used
3	-	-	Not used
4	ETMEXTIN0	ETM	ETM event input 0
5	ETMEXTIN1	ETM	ETM event input 1
6	ETMEXTIN2	ETM	ETM event input 2
7	ETMEXTIN3	ETM	ETM event input 3

For more information on the cross-trigger interface CoreSight component, refer to the Arm CoreSight SoC-400 Technical Reference Manual *[2]*.

65.11.1 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.	SIZE[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 **SIZE[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 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: 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

CTI register map

Table 687. 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

Table 687. 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
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
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.	CHINSTATUS [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.	CHOUTSTATUS[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.	SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															

Table 687. 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
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	1	0	0	0	1
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
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 *Table 677: Processor ROM table* for register boundary addresses.

65.12 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

65.12.1 Device ID

The DBGMCU includes an identity code register, DBGMCU_IDCODE. This register contains the ID code for the device. Debug tools can locate this register via the CoreSight discovery procedure described in *Section 65.5: ROM tables*.

65.12.2 Low-power mode emulation

When the device enters either Stop mode (clocks are stopped) or Standby mode (core power is switched off), the debugger can no longer access the debug access port and loses the connection with the device. To avoid this, the debugger (or software) can set the

DBG_STANDBY and/or DBG_STOP bits in the *DBGMCU configuration register (DBGMCU_CR)*. These bits, when set, maintain the clock and power to the processor while the device is in the corresponding low-power mode. The processor remains in Sleep mode, and exits the low-power mode in the normal way. However, peripheral devices continue to operate, so the device behaviour may not be identical to the one in the actual low-power mode.

65.12.3 Peripheral clock freeze

The DBGMCU peripheral clock freeze registers allow the operation of certain peripherals to be suspended in debug mode. The peripheral units which support this feature are listed in the table below.

Table 688. Peripheral clock freeze control bits

Bus	Control register	Peripheral	Description
APB1L	DBGMCU_APB1LFZR	I2C2	I2C2 SMBUS timeout
		I2C1	I2C1 SMBUS timeout
		IWDG	Independent watchdog
		WWDG	Window watchdog
		TIM7	General purpose timer 7
		TIM6	General purpose timer 6
		TIM5	General purpose timer 5
		TIM4	General purpose timer 4
		TIM3	General purpose timer 3
		TIM2	General purpose timer 2
APB1H	DGBMCU_APB1HFZR	LPTIM2	Low power timer 2
		I2C4	I2C4 SMBUS timeout
APB2	DBGMCU_APB2FZR	TIM17	General purpose timer 17
		TIM16	General purpose timer 16
		TIM15	General purpose timer 15
		TIM8	General purpose timer 8
		TIM1	General purpose timer 1
APB3	DBGMCU_APB3FZR	RTC	Real time clock
		LPTIM4	Low power timer 4
		LPTIM3	Low power timer 3
		LPTIM1	Low power timer 1
		I2C3	I2C3 SMBUS timeout
AHB1	DBGMCU_AHB1FZR	GPDMA 0 to 15	General purpose DMA channels 0 to 15
AHB3	DBGMCU_AHB3FZR	LPDMA 0 to 3	Low power DMA channels 0 to 3

Each peripheral unit or DMA channel has a corresponding control bit, DBG_xxx_STOP, where xxx is the acronym of the peripheral (or DMA channel). The control bits are organized in DBGMCU_zzzFZR registers, where zzz corresponds to the name of the bus (AHB or APB). For example, DBGMCU_APB1LFZR contains the control bits for peripherals on the APB1L bus.

The control bit, when set, causes the corresponding peripheral operation to be suspended when the CPU is stopped in debug (HALTED = 1), according to the table below:

Table 689. Peripheral behavior in debug mode

HALTED	DBG_xxx_STOP	Peripheral behaviour
0	X	The operation continues.
1	0	The operation continues.
1	1	The operation is suspended.

The accessibility of the bits DBG_xxx_STOP by the debugger depends on the state of the authentication signal spiden.

When spiden = 1 (secure privilege debug enabled), all bits can be modified by a secure access. Only bits corresponding to non-secure peripherals (or DMA channels) can be modified by a non-secure access. All bits can be read by both non-secure or secure accesses.

When spiden = 0 (secure privilege debug disabled), only non-secure accesses are possible (secure access requests by the debugger are converted to non-secure by the CPU). Only bits corresponding to non-secure peripherals (or DMA channels) can be modified. All bits can be read. This is summarized in the table below.

Table 690. Debugger access to freeze register bits

spiden	Peripheral xxx status	Access security attribute	DBG_xxx_STOP can be modified?	DBG_xxx_STOP can be read?
0	Non-secure	Non-secure	Yes	Yes
		Secure	Yes(1)	Yes(1)
	Secure	Non-secure	No	Yes
		Secure	No(1)	Yes(1)
1	Non-secure	Non-secure	Yes	Yes
		Secure	Yes	Yes
	Secure	Non-secure	No	Yes
		Secure	Yes	Yes

1. When spiden = 0, secure access requests by the debugger are converted to non-secure.

The status (secure or non-secure) of a TrustZone–aware peripheral or a DMA channel, is signaled to the DBGMCU by the peripheral.

The CPU access to the DBG_xxx_STOP bits does not depend on spiden. This access depends only on the security status of the peripheral. The bits corresponding to a secure

peripheral (or DMA channel) can only be modified by a secure access (when CPU is in secure state).

65.12.4 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, and to software, at base address 0xE004 4000.

DBGMCU identity code register (DBGMCU_IDCODE)

Address offset: 0x00

Reset value: 0xXXXX 6XXX

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
0x2001: revision X
0x3000: revision C

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV_ID[11:0]**: device identification
0x482: STM32U575/585

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 4 ((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 APB1L peripheral freeze register (DBGMCU_APB1LFZR)

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.	DBG_I2C2_STOP	DBG_I2C1_STOP	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.	DBG_IWDG_STOP	DBG_WWDG_STOP	Res.	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

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

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.

Bits 10:6 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 APB1H peripheral freeze register (DBGMCU_APB1HFZR)

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.	DBG_LPTIM2_STOP	Res.	Res.	Res.	DBG_I2C4_STOP	Res.
										rw				rw	

Bits 31:6 Reserved, must be kept at reset value.

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.

DBGMCU APB3 peripheral freeze register (DBGMCU_APB3FZR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DBG_RTC_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM4_STOP	DBG_LPTIM3_STOP	DBG_LPTIM1_STOP	Res.
	rw											rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DBG_I2C3_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
					rw										

Bit 31 Reserved, must be kept at reset value.

Bit 30 **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 29:20 Reserved, must be kept at reset value.

Bit 19 **DBG_LPTIM4_STOP**: LPTIM4 stop in debug
0: normal operation. LPTIM4 continues to operate while CPU is in debug mode.
1: stop in debug. LPTIM4 is frozen while CPU is in debug mode.

Bit 18 **DBG_LPTIM3_STOP**: LPTIM3 stop in debug
0: normal operation. LPTIM3 continues to operate while CPU is in debug mode.
1: stop in debug. LPTIM3 is frozen while CPU is in debug mode.

Bit 17 **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 16:11 Reserved, must be kept at reset value.

Bit 10 **DBG_I2C3_STOP**: I2C3 stop in debug
0: normal operation. I2C3 continues to operate while CPU is in debug mode.
1: stop in debug. I2C3 is frozen while CPU is in debug mode.

Bits 9:0 Reserved, must be kept at reset value.

DBGMCU AHB1 peripheral freeze register (DBGMCU_AHB1FZR)

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
DBG_GPDMA15_STOP	DBG_GPDMA14_STOP	DBG_GPDMA13_STOP	DBG_GPDMA12_STOP	DBG_GPDMA11_STOP	DBG_GPDMA10_STOP	DBG_GPDMA9_STOP	DBG_GPDMA8_STOP	DBG_GPDMA7_STOP	DBG_GPDMA6_STOP	DBG_GPDMA5_STOP	DBG_GPDMA4_STOP	DBG_GPDMA3_STOP	DBG_GPDMA2_STOP	DBG_GPDMA1_STOP	DBG_GPDMA0_STOP
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **DBG_GPDMA15_STOP**: GPDMA channel 15 stop in debug
0: normal operation. GPDMA channel 15 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA channel 15 is frozen while CPU is in debug mode.

Bit 14 **DBG_GPDMA14_STOP**: GPDMA channel 14 stop in debug
0: normal operation. GPDMA channel 14 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA channel 14 is frozen while CPU is in debug mode.

Bit 13 **DBG_GPDMA13_STOP**: GPDMA channel 13 stop in debug
0: normal operation. GPDMA channel 13 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA channel 13 is frozen while CPU is in debug mode.

Bit 12 **DBG_GPDMA12_STOP**: GPDMA channel 12 stop in debug
0: normal operation. GPDMA channel 12 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA channel 12 is frozen while CPU is in debug mode.

Bit 11 **DBG_GPDMA11_STOP**: GPDMA channel 11 stop in debug
0: normal operation. GPDMA channel 11 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA channel 11 is frozen while CPU is in debug mode.

Bit 10 **DBG_GPDMA10_STOP**: GPDMA channel 10 stop in debug
0: normal operation. GPDMA channel 10 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA channel 10 is frozen while CPU is in debug mode.

Bit 9 **DBG_GPDMA9_STOP**: GPDMA channel 9 stop in debug
0: normal operation. GPDMA channel 9 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA channel 9 is frozen while CPU is in debug mode.

Bit 8 **DBG_GPDMA8_STOP**: GPDMA channel 8 stop in debug
0: normal operation. GPDMA channel 8 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA channel 8 is frozen while CPU is in debug mode.

Bit 7 **DBG_GPDMA7_STOP**: GPDMA channel 7 stop in debug
0: normal operation. GPDMA channel 7 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA channel 7 is frozen while CPU is in debug mode.

Bit 6 **DBG_GPDMA6_STOP**: GPDMA channel 6 stop in debug
0: normal operation. GPDMA channel 6 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA channel 6 is frozen while CPU is in debug mode.

Bit 5 **DBG_GPDMA5_STOP**: GPDMA channel 5 stop in debug
0: normal operation. GPDMA channel 5 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA channel 5 is frozen while CPU is in debug mode.

Bit 4 **DBG_GPDMA4_STOP**: GPDMA channel 4 stop in debug
0: normal operation. GPDMA channel 4 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA channel 4 is frozen while CPU is in debug mode.

Bit 3 **DBG_GPDMA3_STOP**: GPDMA channel 3 stop in debug
0: normal operation. GPDMA channel 3 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA channel 3 is frozen while CPU is in debug mode.

Bit 2 **DBG_GPDMA2_STOP**: GPDMA channel 2 stop in debug
0: normal operation. GPDMA channel 2 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA channel 2 is frozen while CPU is in debug mode.

Bit 1 **DBG_GPDMA1_STOP**: GPDMA channel 1 stop in debug
0: normal operation. GPDMA channel 1 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA channel 1 is frozen while CPU is in debug mode.

Bit 0 **DBG_GPDMA0_STOP**: GPDMA channel 0 stop in debug
0: normal operation. GPDMA channel 0 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA channel 0 is frozen while CPU is in debug mode.

DBGMCU AHB3 peripheral freeze register (DBGMCU_AHB3FZR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPDMA3_STOP	DBG_LPDMA2_STOP	DBG_LPDMA1_STOP	DBG_LPDMA0_STOP
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **DBG_LPDMA3_STOP**: LPDMA channel 3 stop in debug
0: normal operation. LPDMA channel 3 continues to operate while CPU is in debug mode.
1: stop in debug. LPDMA channel 3 is frozen while CPU is in debug mode.

Bit 2 **DBG_LPDMA2_STOP**: LPDMA channel 2 stop in debug
0: normal operation. LPDMA channel 2 continues to operate while CPU is in debug mode.
1: stop in debug. LPDMA channel 2 is frozen while CPU is in debug mode.

Bit 1 **DBG_LPDMA1_STOP**: LPDMA channel 1 stop in debug
0: normal operation. LPDMA channel 1 continues to operate while CPU is in debug mode.
1: stop in debug. LPDMA channel 1 is frozen while CPU is in debug mode.

Bit 0 **DBG_LPDMA0_STOP**: LPDMA channel 0 stop in debug
0: normal operation. LPDMA channel 0 continues to operate while CPU is in debug mode.
1: stop in debug. LPDMA channel 0 is frozen while CPU is in debug mode.

DBGMCU status register (DBGMCU_SR)

Address offset: 0xFC

Reset value: 0x0000 XX01

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

Bits 31:16 **AP_ENABLED[15:0]**: Bit n identifies whether access port AP n is open (can be accessed via the debug port) or locked (debug access to the AP is blocked)
Bit n = 0: APn locked
Bit n = 1: APn enabled

Bits 15:0 **AP_PRESENT[15:0]**: Bit n identifies whether access port AP n is present in device
Bit n = 0: APn absent
Bit n = 1: APn present

DBGMCU debug host authentication register (DBGMCU_DBG_AUTH_HOST)

Address offset: 0x100

Reset value: 0xXXXX XXXX

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

Bits 31:0 **AUTH_KEY[31:0]**: Device authentication key

The device specific 64-bit authentication key (OEM key) must be written to this register (in two successive 32-bit writes, least significant word first) to permit RDP regression. Writing a wrong key locks access to the device and prevent code execution from the Flash memory.

DBGMCU debug device authentication register (DBGMCU_DBG_AUTH_DEVICE)

Address offset: 0x104

Reset value: 0xXXXX XXXX

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

Bits 31:0 **AUTH_ID[31:0]**: Device specific ID

Device specific ID used for RDP regression.

DBGMCU CoreSight peripheral identity register 4 (DBGMCU_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.	SIZE[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 **SIZE[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 code

DBGMCU CoreSight peripheral identity register 0 (DBGMCU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0000

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

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]
0x00: DBGMCU part number

DBGMCU CoreSight peripheral identity register 1 (DBGMCU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 0000

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

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]
0x0: DBGMCU part number

DBGMCU CoreSight peripheral identity register 2 (DBGMCU_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: 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]
0x2: STMicroelectronics JEDEC code

DBGMCU CoreSight peripheral identity register 3 (DBGMCU_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

DBGMCU CoreSight component identity register 0 (DBGMCU_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

DBGMCU CoreSight component identity register 1 (DBGMCU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00F0

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

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
0xF: Non-CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]
0x0: common identification value

DBGMCU CoreSight component identity register 2 (DBGMCU_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

DBGMCU CoreSight component identity register 3 (DBGMCU_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

DBGMCU register map

Table 691. DBGMCU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DBGMCU_IDCODE	REV_ID[15:0]																Res.	Res.	Res.	Res.	DEV_ID[11:0]											
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0x004	DBGMCU_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACE_MODE [1:0]		TRACE_EN	TRACE_IOEN	Res.	DBG_STANDBY	DBG_STOP	Res.
	Reset value																									0	0	0	0		0	0	

Table 691. DBGMCU 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
0x008	DBGMCU_ APB1LFZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I2C2_STOP	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_IWDG_STOP	DBG_WWDG_STOP	Res.	Res.	Res.	Res.	Res.	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
	Reset value										0	0									0	0						0	0	0	0	0	0
0x0C0	DBGMCU_ APB1HFZR	Res.	Res.	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_LPTIM2_STOP	Res.	Res.	Res.	DBG_I2C4_STOP	Res.
	Reset value																											0				0	
0x010	DBGMCU_ APB2FZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP	Res.	Res.	Res.	DBG_TIM8_STOP	Res.	DBG_TIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0	0				0		0											
0x04	DBGMCU_ APB3FZR	Res.	DBG_RTC_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM4_STOP	DBG_LPTIM3_STOP	DBG_LPTIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I2C3_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		0											0	0	0							0										
0x018 to 0x01C	Reserved	Reserved																															
0x020	DBGMCU_ AHB1FZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_GPDMA15_STOP	DBG_GPDMA14_STOP	DBG_GPDMA13_STOP	DBG_GPDMA12_STOP	DBG_GPDMA11_STOP	DBG_GPDMA10_STOP	DBG_GPDMA9_STOP	DBG_GPDMA8_STOP	DBG_GPDMA7_STOP	DBG_GPDMA6_STOP	DBG_GPDMA5_STOP	DBG_GPDMA4_STOP	DBG_GPDMA3_STOP	DBG_GPDMA2_STOP	DBG_GPDMA1_STOP	DBG_GPDMA0_STOP
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	Reserved	Reserved																															
0x028	DBGMCU_ AHB3FZR	Res.	Res.	Res.	Res.	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_LPDMA3_STOP	DBG_LPDMA2_STOP	DBG_LPDMA1_STOP	DBG_LPDMA0_STOP
	Reset value																													0	0	0	0
0x02C to 0x0F8	Reserved	Reserved																															
0x0FC	DBGMCU_SR	AP_ENABLED[15:0]																AP_PRESENT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	1

Table 691. DBGMCU 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	DBGMCU_DBG_AUTH_HOST	AUTH_KEY[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
0x104	DBGMCU_DBG_AUTH_DEVICE	AUTH_ID[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
0x108 to 0xFBC	Reserved	Reserved																															
0xFD0	DBGMCU_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.	SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															
0xFE0	DBGMCU_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0xFE4	DBGMCU_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	0	0	0
0xFE8	DBGMCU_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	DBGMCU_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	DBGMCU_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	DBGMCU_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	1	0	0	0	0
0xFF8	DBGMCU_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	DBGMCU_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.

65.13 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 2015DDI 0314H - Arm CoreSight Components Technical Reference Manual, Issue H, 10 July, 2009
3. DDI 0553A (ID092917) - Arm v8-M Architecture Reference Manual, Issue A.f, 29 September 2017
4. 100230_0002_00_en - Arm Cortex-M33 Processor r0p2 Technical Reference Manual, Issue 0002-00, 10 May 2017
5. 100232_0001_00_en - Arm CoreSight ETM-M33 r0p1 Technical Reference Manual, Issue 0001-00, 3 February 2017

66 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 devices.

66.1 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

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 0700

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)

66.2 Flash size data register

Base address: 0x0BFA 07A0

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 field indicates the size of the device Flash memory expressed in Kbytes.
As an example, 0x800 corresponds to 2048 Kbytes.

66.3 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: LQFP48
00111: UFBGA169
01000: LQFP64 SMPS
01001: WLSCP90 SMPS
01010: LQFP100 SMPS
01011: UFBGA132 SMPS
01100: LQFP144 SMPS
01101: LQFP48 SMPS
01111: UFBGA169 SMPS
Others: reserved

67 Revision history

Table 692. Document revision history

Date	Revision	Changes
22-Jun-2021	1	Initial release
20-Sep-2021	2	Updated: – *Figure 1: System architecture* – *Figure 3: Memory map based on IDAU mapping for STM32U575/585* – End of *Section 6.3.2: Error code correction (SRAM2, SRAM3, BKPSRAM)* – Sentence added in intro of *Section 7.6.2: Readout protection (RDP)* – Desc of bit 21 in *Section 7.9.14: FLASH option register (FLASH_OPTR)* – First sentence of *Section 10.5.3: LDO and SMPS step down converter fast startup* – One sentence removed in *Exiting Stop 0 mode*, *Exiting Stop 2 mode* and *Exiting Stop 3 mode* – Note added on BREN bit in *Section 10.10.9: PWR Backup domain control register 1 (PWR_BDCR1)* – Desc of MSISSRANGE and MSIKSRANGE in *Section 11.8.50: RCC control/status register (RCC_CSR)* – *Table 125: Peripherals interconnect matrix* – *Table 124: Programmed GPDMA1 request* – Bit 7 in *Section 25.7.1: OCTOSPI control register (OCTOSPI_CR)* – *Section 37.2: PSSI main features* – *Table 212: RNG configurations* – *Section 34.4.13: AES data registers and data swapping* – *Section 42.4.14: SAES operation with shared keys* – Address offset of *Section 23.6.19: TIMx option register 1 (TIMx_OR1)(x = 16 to 17)* – *Table 121: LPTIM1/2/3 input/output pins*, *Table 122: LPTIM4 input/output pins* and *Table 146: LPTIM1/2/3/4 external trigger connection* – *Table 533: I2C1, I2C2, I2C4 interconnection* and *Table 534: I2C3 interconnection* – *Table 294: Comparison of analog vs. digital filters* – usart/lpuart_trg6/7 in *Table 553: USART interconnection (USART1/2/3 and UART4/5)* – New notes in *Section 31.5.20: Continuous communication using USART and DMA* – New *Determining the maximum USART baud rate that enables to correctly wake up the microcontroller from low-power mode* – *Table 322: LPUART input/output pins* – usart/lpuart_trg6/7 in *Table 565: LPUART interconnections (LPUART1)* – New – *Table 576: SPI interconnection (SPI1 and SPI2)* and *Table 577: SPI interconnection (SPI3)* – New *Control of the I/Os* – Structure of *Section 65.3: Serial-wire and JTAG debug port (SWJ-DP)* – APSEL range in *DP access port select register (DP_SELECTR)* – REV_ID[15:0] in *DBGMCU identity code register (DBGMCU_IDCODE)* – TREVISION[3:0] in *DP target identification register (DP_TARGETIDR)*

Table 692. Document revision history

Date	Revision	Changes
16-Mar-2022	3	Updated: – *Related documents* – New note in *Section 2.1.9: SmartRun domain (SRD)* – *Section 3.1: Key security features* – *Section 3.2: Secure install* – *Section 3.3.2: Immutable root of trust in system Flash memory* – *Section 3.4: Secure update* – *Table 10: Non-secure peripheral functions that are not connected to secure I/Os* – *Section 3.5.7: Deactivating TrustZone security* – New note in *Section 3.8.1: Hardware secret key management* – *Section 3.11: Access controlled debug* – DCMISEC desc in *Section 5.6.4: GTZC1 TZSC secure configuration register 3 (GTZC1_TZSC_SECCFGR3)* – DCMIPRIV desc in *Section 5.6.7: GTZC1 TZSC privilege configuration register 3 (GTZC1_TZSC_PRIVCFGR3)* – DCMIIE desc in *Section 5.7.3: GTZC1 TZIC interrupt enable register 3 (GTZC1_TZIC_IER3)* – DCMIF desc in *Section 5.7.7: GTZC1 TZIC status register 3 (GTZC1_TZIC_SR3)* – CDCMIF desc in *Section 5.7.11: GTZC1 TZIC flag clear register 3 (GTZC1_TZIC_FCR3)* – New warning in *Section 6.3.2: Error code correction (SRAM2, SRAM3, BKPSRAM)* – Sentence in *Section 7.3.1: Flash memory organization* – New ST production value in *Section 7.9.13: FLASH option register (FLASH_OPTR)* to *Section 7.9.24: FLASH WPR2 area B address register (FLASH_WRP2BR)* – *Figure 29: Power supply overview* – V_{BAT} in *Section 10.4.1: External power supplies* – *Section 10.4.7: Battery Backup domain* – PSSI added in *Table 89: Functionalities depending on the working mode* – ULPMEN desc in *Section 10.10.1: PWR control register 1 (PWR_CR1)* – *Section 10.6.1: Brownout reset (BOR)* – *Section 10.6.2: Programmable voltage detector (PVD)* – *Section 10.7.8: Stop 2 mode* and *Section 10.7.9: Stop 3 mode* – Note on *Table 99: PWR interrupt requests* – *Section 11.4.6: PLL* – *Section 11.4.7: LSE clock* – *Hardware auto calibration with LSE (PLL-mode)* – *Section 11.4.12: Clock security system on LSE* – *Section 11.4.18: Clock-out capability* – *Section 11.5.1: RCC TrustZone security protection modes* – LSECSS, MSI_PLL_UNLOCK, and notes of *Table 107: Interrupt sources and control* – MSIBIAS desc in *Section 11.8.2: RCC internal clock sources calibration register 1 (RCC_ICSCR1)* – PLL1P/R desc in *Section 11.8.12: RCC PLL1 dividers register (RCC_PLL1DIVR)* – LSESYSEN desc in *Section 11.8.49: RCC Backup domain control register (RCC_BDCR)* – LSECSS, MSI_PLL_UNLOCK in *Table 153: STM32U575/585 vector table*

Table 692. Document revision history

Date	Revision	Changes
16-Mar-2022	3 (cont'd)	– Line 23 in *Table 156: EXTI line connections* – Bit 23 in *Section 21.6.1* to *Section 21.6.7*, *Section 21.6.10*, and *Section 21.6.11* – *Section 21.6.8: EXTI external interrupt selection register (EXTI_EXTICRm)* – *Bulb sampling mode* – CALINDEX[3:0] in ADC_CR – New *Extended calibration mode* – *Table 249: Memory location of the temperature sensor calibration values* – *Section 30.7.13: ADC power register (ADC_PWRR)* – New *Section 55.3: TAMP implementation* – *Active tamper detection* – ATCKSEL desc in *Section 55.7.5: TAMP active tamper control register 1 (TAMP_ATCR1)* – Note in *Table 507: STM32U575/585 IWDG features* – *Configuring the IWDG when the window option is enabled* – New *Updating the window comparator* – *Section 52.4.6: Register access protection* – KEY[15:0] desc in *Section 52.7.1: IWDG key register (IWDG_KR)* – *Table 653: UCPD implementation* and *Table 654: UCPD software trim data* – New *Section 64.5.5: UCPD software trimming* – *Section 64.8.3: UCPD configuration register 3 (UCPD_CFGR3)* – TREVISION desc in *DP target identification register (DP_TARGETIDR)* – REV_ID desc in *DBGMCU identity code register (DBGMCU_IDCODE)*

Index

A

B

C

D

E

F

G

H

I

L

M

O

P

R

S

T

U

V

W

IMPORTANT NOTICE – PLEASE READ CAREFULLY

The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST's customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

- ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture (www.psacertified.org) and/or Security Evaluation Standard for IoT Platforms (www.trustcb.com). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on *www.st.com* for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers should re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers should not assume it is certified.

- Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.

- Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.

- While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.

- All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.

ST reserves the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products, and ST assumes no liability for application assistance or the design of purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to *www.st.com/trademarks*. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.