RM0493
Reference manual

Multiprotocol wireless Bluetooth® LE and IEEE802.15.4, STM32WBA5xxx Arm®-based 32-bit MCUs

Introduction

This document is addressed to application developers. It provides complete information on how to use the STM32WBA50/52/54/55xx (hereinafter referred to as STM32WBA5xxx) microcontrollers.

These MCUs include ST state-of-the-art patented technology, and feature 2.4 GHz wireless radio, embedded memories, and peripherals. They are based on a single core (Arm® Cortex®-M33), with the following memory configurations:

- For STM32WBA52/54/55xx devices
  - 1-Mbyte flash + 128-Kbyte SRAM (STM32WBA52/54/55xGx)
  - 512-Kbyte flash + 96-Kbyte SRAM (STM32WBA52/54/55xEx)
- For STM32WBA50xx devices
  - 1-Mbyte flash + 64-Kbyte SRAM (STM32WBA50xGx)

Related documents (available from STMicroelectronics web site *www.st.com*):

- STM32WBA5xxx datasheets
- STM32WBA5xxx erratas

Contents

List of tables

List of figures

1 Documentation conventions

1.1 General information

The STM32WBA5xxx 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.

1.3 Register reset value

Bits (binary notation) or bits nibbles (hexadecimal notation) of which the reset value is undefined are marked as X.

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.

Bits (binary notation) or bits nibbles (hexadecimal notation) of which the reset value is unmodified are marked as U.

1.4 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.

2 Memory and bus architecture

2.1 System architecture

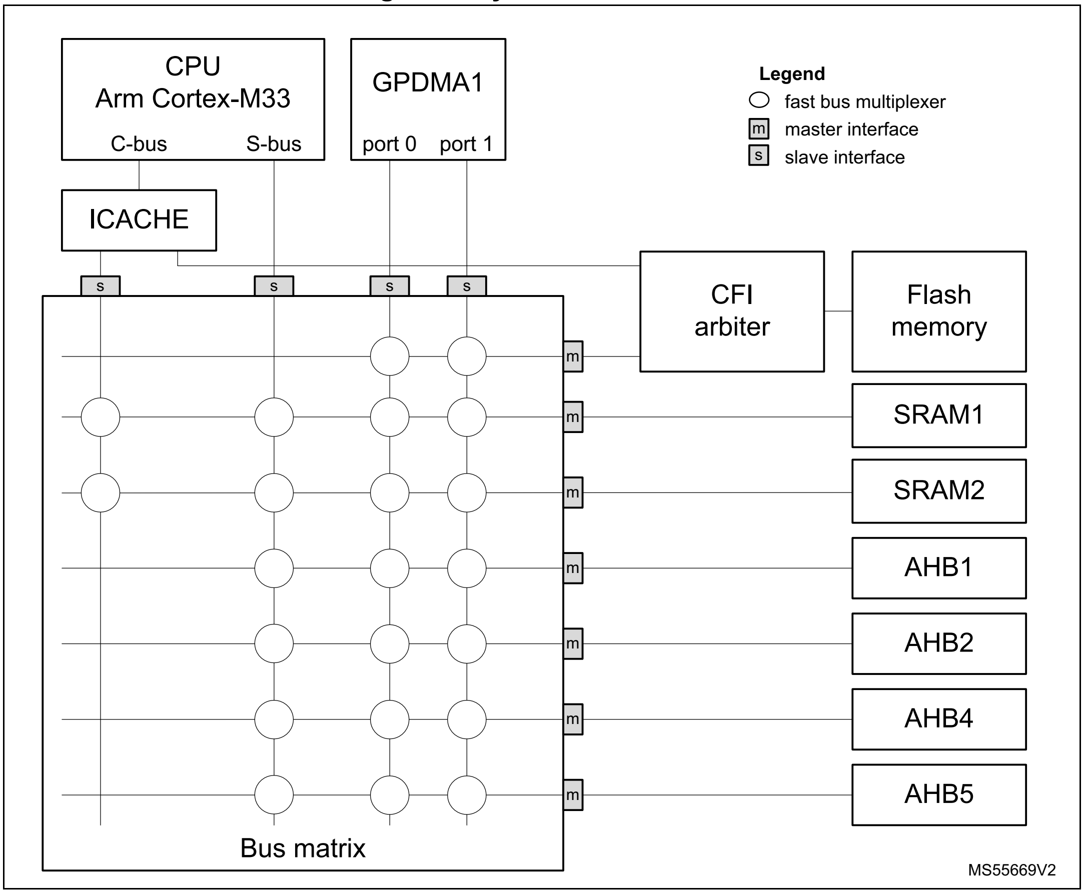
The device architecture relies on an Arm® Cortex®-M33 core optimized for execution, thanks to an instruction cache with a direct access to the embedded flash memory.

The architecture features a 32-bit multilayer AHB bus matrix interconnecting masters and slaves:

- Masters:
  - CPU core C-bus through 128-bit wide ICACHE connecting to flash memory
  - CPU core C-bus through 32-bit wide ICACHE connecting to SRAM
  - CPU core S-bus
  - GPDMA1 port 0
  - GPDMA1 port 1
- Slaves:
  - Internal flash memory on the CPU core C-bus
  - Internal flash memory on the GPDMA1 bus
  - Internal SRAM1
  - Internal SRAM2
  - AHB1 peripherals including AHB to APB bridges and APB peripherals (connected to APB1 and APB2)
  - AHB2 peripherals
  - AHB4 peripherals including AHB to APB bridges and APB peripherals (connected to APB7)
  - AHB5 with the 2.4 GHz RADIO peripheral

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. The architecture is shown in *Figure 1*.

Figure 1. System architecture



2.1.1 CPU C-bus

This bus connects the C-bus of the CPU 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 SRAM (SRAM1 and SRAM2) via the ICACHE address remap function.

2.1.2 CPU S-bus

This bus connects the system bus of the CPU to the bus matrix, and it is used by the core to access data located in a peripheral or SRAM area. This bus targets the internal SRAM (SRAM1 and SRAM2), the AHB1 peripherals including the APB1 and APB2 peripherals, AHB2, AHB4 peripherals including the APB7, and AHB5 peripherals.

2.1.3 GPDMA1-bus

The buses connect the two AHB master interfaces of the GPDMA1 to the bus matrix. This targets the internal flash memory, the internal SRAMs (SRAM1, SRAM2), the AHB1 peripherals including the APB1 and APB2 peripherals, the AHB2 peripherals, AHB4 peripherals including the APB7, and AHB5 peripherals.

2.1.4 Bus matrix

The bus matrix manages the access arbitration (based on fixed priority) between masters, and features a bus multiplexer used to connect each master to a given slave without latency.

Table 1. Bus matrix access arbitration

Master	Priority
CPU core S-bus	1 (highest)
ICACHE slow port	2
GPDMA1 port 0	3
GPDMA1 port 1	4 (lowest)

2.1.5 AHB/APB bridges

The three bridges (AHB1 to APB1, AHB1 to ABP2, and AHB4 to APB7) provide full synchronous connections between the AHB and the APB buses, resulting in 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, the clock of peripherals having an xxEN bit in the RCC is disabled. Before using a peripheral, its clock must be enabled in the RCC_AHBxENR and RCC_APBxENR registers.

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

AHB5 is a semisynchronous bus, connected through a bridge to the bus matrix.

2.2 TrustZone® security architecture[a]

The security architecture is based on Arm® TrustZone with the Armv8-M mainline extension.

The TZEN option bit in the FLASH_OPTR register activates TrustZone security.

When the TrustZone is enabled, the SAU (security-attribution unit) and IDAU (implementation-defined-attribution unit) defines the access permissions based on secure and nonsecure states.

- SAU: up to eight SAU configurable regions are available for security attribution.
- IDAU: provides a first memory partition as nonsecure or nonsecure callable attributes. The IDAU memory map partition is not configurable and fixed by hardware implementation (refer to *Figure 2: Memory map*). It is then combined with the results from the SAU security attribution, and the higher security state is selected.

a. Available only on STM32WBA52/4/5xx devices.

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

Table 2 shows a typical example of eight SAU regions mapping, based on IDAU regions. The user can split and choose the secure, nonsecure or NSC regions for external memories according to application needs.

Table 2. Memory map security attribution example vs. SAU configuration regions[1]

Region description	Address range	IDAU security attribution	SAU security attribution typical configuration	Final security attribution
Reserved	0x0000 0000 to 0x07FF FFFF	Nonsecure	Secure, nonsecure or NSC	
Code Flash and SRAM	0x0800 0000 to 0x0BFF FFFF	Nonsecure	Nonsecure	
	0x0C00 0000 to 0x0FFF FFFF	NSC	Secure or NSC	
Reserved	0x1000 0000 to 0x17FF FFFF	Nonsecure		
	0x1800 0000 to 0x1FFF FFFF			
SRAM	0x2000 0000 to 0x2FFF FFFFF	Nonsecure		
	0x3000 0000 to 0x3FFF FFFFF	NSC	Secure or NSC	
Peripherals	0x4000 0000 to 0x4FFF FFFFF	Nonsecure	Nonsecure	
	0x5000 0000 to 0x5FFF FFFFF	NSC	Secure or NSC	
Reserved	0x6000 0000 to 0xDFFF FFFF	Nonsecure	Secure, nonsecure or NSC	

1. NSC = nonsecure callable

2.2.1 Default TrustZone security state

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

- CPU:
  - CPU1 Cortex-M33 is in secure state after reset. The boot address must point to a secure area.
- 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 nonsecure after reset.
- SRAMs:
  - All SRAMs are secure after reset. MPCBBx (block-based memory protection controller) are secure.
- Peripherals (see *Table 3 and Table 4 for a list of securable and TrustZone-aware peripherals)*
  - Securable peripherals are nonsecure after reset.
  - TrustZone-aware peripherals are nonsecure after reset. Their secure configuration registers are secure.
- All GPIOs are secure after reset.
- Interrupts:
  - NVIC: All interrupts are secure after reset. NVIC is banked for secure and nonsecure state.
  - TZIC: All illegal access interrupts are disabled after reset (see *Section 5.5: GTZC interrupts*).

2.2.2 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 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 *Section 5: Global TrustZone controller (GTZC)* for more details.

Table 3 and *Table 4* list the securable and TrustZone-aware peripherals within the system.

Table 3. Securable peripherals by TZSC

Bus	Peripheral
AHB5	2.4 GHz RADIO + SEQRAM
	PTACONV(1)
AHB4	ADC4

Table 3. Securable peripherals by TZSC (continued)

Bus	Peripheral
AHB2	SAES
	PKA
	RNG
	HASH
	AES
AHB1	ICACHE registers
	TSC
	CRC
	RAMCFG
APB7	LPTIM1
	I2C3
	COMP(1)
	LPUART1
	SPI3
APB2	TIM17
	TIM16
	SAI1(1)
	USART1
	SPI1
	TIM1
APB1	LPTIM2
	I2C1
	USART2
	IWDG
	WWDG
	TIM3
	TIM2

1. Available only on STM32WBA54/55xx devices.

Table 4. TrustZone-aware peripherals

Bus	Peripheral
AHB4	EXTI
	RCC
	PWR

Table 4. TrustZone-aware peripherals (continued)

Bus	Peripheral
AHB2	HSEM
	GPIOH
	GPIOC
	GPIOB
	GPIOA
AHB1	GTZC-MCPBB6
	GTZC-MCPBB2
	GTZC-MCPBB1
	GTZC-TZSC
	GTZC-TZIC
	FLASH interface
	GPDMA1
APB7	TAMP
	RTC
	SYSCFG

2.3 Memory organization

2.3.1 Introduction

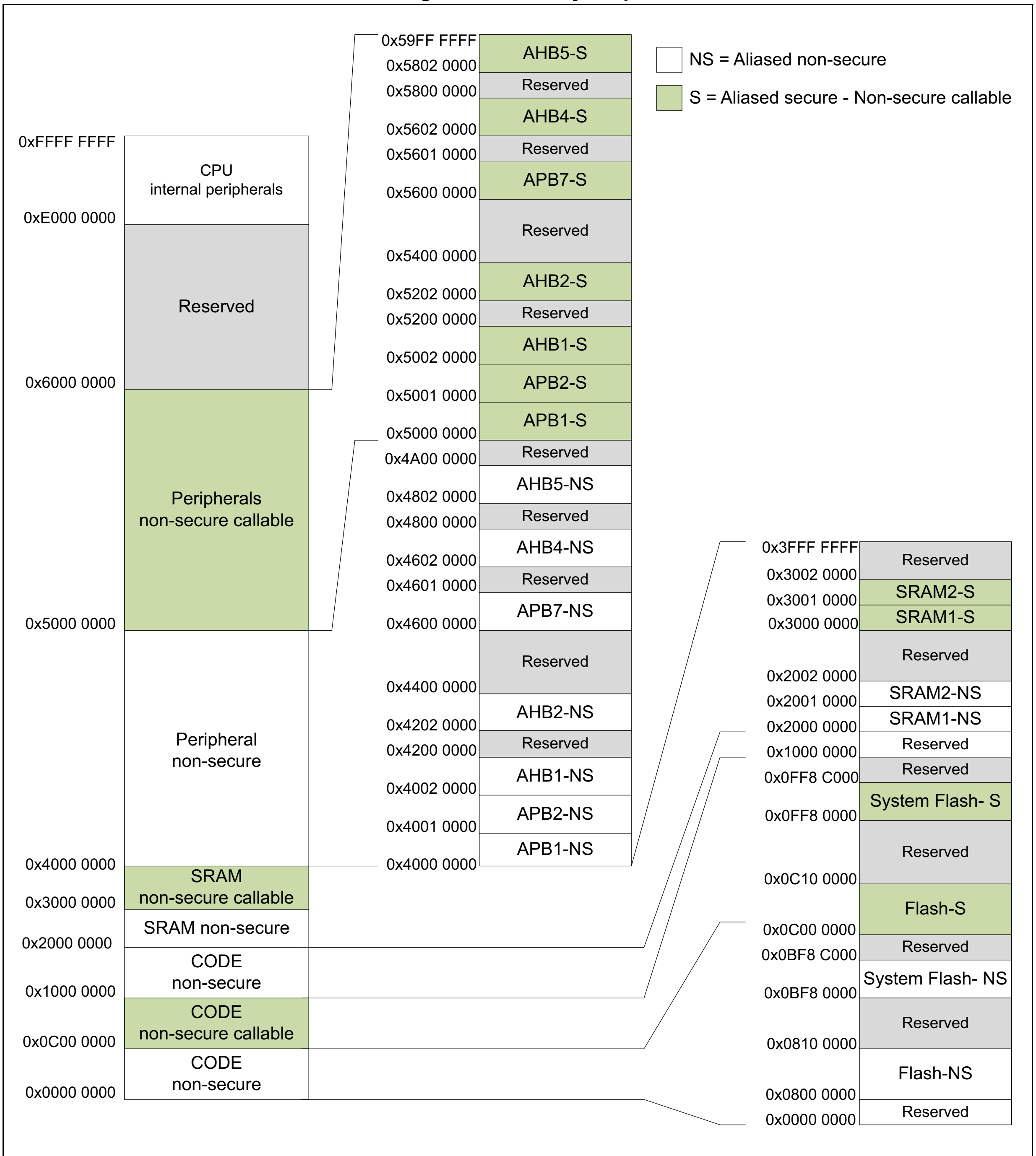
Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space.

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.

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

2.3.2 Memory map and register boundary addresses

Figure 2. Memory map



STM32WBA52/54/55xG devices contain a 1-Mbyte flash memory from address offset 0x00 0000 to 0x0F FFFF, and SRAM1 64-Kbyte from address offset 0x0 0000 to 0x0 FFFF. (continuous SRAM space with SRAM2).

STM32WBA52/54/55xE devices contain a 512-Kbyte flash memory from address offset 0x00 0000 to 0x07 FFFF, and SRAM1 32-Kbyte from address offset 0x0 0000 to 0x0 7FFF (non-continuous SRAM space with SRAM2).

STM32WBA50xG devices contain a 1-Mbyte flash memory from address offset 0x00 000 to 0x0F FFFF, SRAM1 16-Kbyte from address offset 0x0 0000 to 0x0 3FFF (non-continuous SRAM space with SRAM2), and SRAM2 48-Kbyte from address offset 0x1 0000 to 0x1 BFFF.

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

Table 5 gives the boundary addresses of the peripherals available in the device.

Table 5. Memory map and peripheral register boundary addresses

Bus	Nonsecure callable boundary address(1)	Nonsecure boundary address(1)	Size (bytes)	Peripheral	Peripheral register map
-	0x5A00 0000 - 0xDFFF FFFF	0x4A00 0000 - 0x4FFF FFFF	-	Reserved	-
AHB5	0x5803 8400 - 0x59FF FFFF	0x4803 8400 - 0x49FF FFFF	-	Reserved	-
	0x5803 8000 - 0x5803 83FF	0x4803 8000 - 0x4803 83FF	1 K	PTACONV(2)	*Section 10.5.4 on page 272*
	0x5802 C000 - 0x5803 7FFF	0x4802 C000 - 0x4803 7FFF	-	Reserved	-
	0x5802 8000 - 0x5802 BFFF	0x4802 8000 - 0x4802 BFFF	16 K	RXTXRAM	*Section 6.6.11 on page 175*
	0x5802 1200 - 0x5802 7FFF	0x4802 1200 - 0x4802 7FFF	-	Reserved	-
	0x5802 1000 - 0x5802 11FF	0x4802 1000 - 0x4802 11FF	0.5 K	SEQRAM	*Section 6.6.11 on page 175*
	0x5802 0000 - 0x5802 0FFF	0x4802 0000 - 0x4802 0FFF	4 K	2.4 GHz RADIO	-
-	0x5800 0000 - 0x5801 FFFF	0x4800 0000 - 0x4801 FFFF	-	Reserved	-
AHB4	0x5602 2400 - 0x57FF FFFF	0x4602 2400 - 0x47FF FFFF	-	Reserved	-
	0x5602 2000 - 0x5602 23FF	0x4602 2000 - 0x4602 23FF	1 K	EXTI	*Section 19.6.15 on page 615*
	0x5602 1400 - 0x5602 1FFF	0x4602 1400 - 0x4602 1FFF	-	Reserved	-
	0x5602 1000 - 0x5602 13FF	0x4602 1000 - 0x4602 13FF	1 K	ADC4	*Section 21.7.18 on page 684*
	0x5602 0C00 - 0x5602 0FFF	0x4602 0C00 - 0x4602 0FFF	1 K	RCC	*Section 12.8.54 on page 429*
	0x5602 0800 - 0x5602 0BFF	0x4602 0800 - 0x4602 0BFF	1 K	PWR	*Section 11.10.25 on page 328*
	0x5602 0000 - 0x5602 07FF	0x4602 0000 - 0x4602 07FF	-	Reserved	-
-	0x5601 0000 - 0x5601 FFFF	0x4601 0000 - 0x4601 FFFF	-	Reserved	-

Table 5. Memory map and peripheral register boundary addresses (continued)

Bus	Nonsecure callable boundary address(1)	Nonsecure boundary address(1)	Size (bytes)	Peripheral	Peripheral register map
APB7	0x5600 8000 - 0x5600 FFFF	0x4600 8000 - 0x4600 FFFF	-	Reserved	-
	0x5600 7C00 - 0x5600 7FFF	0x4600 7C00 - 0x4600 7FFF	1 K	TAMP	
	0x5600 7800 - 0x5600 7BFF	0x4600 7800 - 0x4600 7BFF	1 K	RTC	
	0x5600 5800 - 0x5600 77FF	0x4600 5800 - 0x4600 77FF	-	Reserved	-
	0x5600 5400 - 0x5600 57FF	0x4600 5400 - 0x4600 57FF	1 K	COMP(2)	
	0x5600 4800 - 0x5600 53FF	0x4600 4800 - 0x4600 53FF	-	Reserved	-
	0x5600 4400 - 0x5600 47FF	0x4600 4400 - 0x4600 47FF	1 K	LPTIM1	Section 32.7.16 on page 1257
	0x5600 2C00 - 0x5600 43FF	0x4600 2C00 - 0x4600 43FF	-	Reserved	-
	0x5600 2800 - 0x5600 2BFF	0x4600 2800 - 0x4600 2BFF	1 K	I2C3	Section 38.9.13 on page 1449
	0x5600 2400 - 0x5600 27FF	0x4600 2400 - 0x4600 27FF	1 K	LPUART1	Section 40.7.15 on page 1593
	0x5600 2000 - 0x5600 23FF	0x4600 2000 - 0x4600 23FF	1 K	SPI3	Section 41.8.15 on page 1647
	0x5600 0800 - 0x5600 1FFF	0x4600 0800 - 0x4600 1FFF	-	Reserved	-
	0x5600 0400 - 0x5600 07FF	0x4600 0400 - 0x4600 07FF	1 K	SYSCFG	Section 15.3.12 on page 517
	0x5600 0000 - 0x5600 03FF	0x4600 0000 - 0x4600 03FF	-	Reserved	-
-	0x5400 0000 - 0x55FF FFFF	0x4400 0000- 0x45FF FFFF	-	Reserved	-
AHB2	0x520C 4000 - 0x53FF FFFF	0x420C 4000 - 0x43FF FFFF	-	Reserved	-
	0x520C 3400 - 0x520C 3FFF	0x420C 3400 - 0x420C 3FFF	8 K	PKA continue	Section 28.7.5 on page 888
	0x520C 2400 - 0x520C 33FF	0x420C 2400 - 0x420C 33FF		PKA RAM	
	0x520C 2000 - 0x520C 23FF	0x420C 2000 - 0x420C 23FF		PKA	
	0x520C 1C00 - 0x520C 1FFF	0x420C 1C00 - 0x420C 1FFF	1 K	HSEM	Section 13.4.15 on page 455
	0x520C 1000 - 0x520C 1BFF	0x420C 1000 - 0x420C 1BFF	-	Reserved	-
	0x520C 0C00 - 0x520C 0FFF	0x420C 0C00 - 0x420C 0FFF	1 K	SAES(3)	Section 26.8.21 on page 832
	0x520C 0800 - 0x520C 0BFF	0x420C 0800 - 0x420C 0BFF	1 K	RNG	Section 24.7.6 on page 732
	0x520C 0400 - 0x520C 07FF	0x420C 0400 - 0x420C 07FF	1 K	HASH	Section 27.6.8 on page 854
	0x520C 0000 - 0x520C 03FF	0x420C 0000 - 0x420C 03FF	1 K	AES	Section 25.8.21 on page 776
	0x5202 2000 - 0x520B FFFF	0x4202 2000 - 0x420B FFFF	-	Reserved	-
	0x5202 1C00 - 0x5202 1FFF	0x4202 1C00 - 0x4202 1FFF	1 K	GPIOH	Section 14.8.15 on page 502
	0x5202 0C00 - 0x5202 1BFF	0x4202 0C00 - 0x4202 1BFF	-	Reserved	-
	0x5202 0800 - 0x5202 0BFF	0x4202 0800 - 0x4202 0BFF	1 K	GPIOC	Section 14.8.14 on page 501
	0x5202 0400 - 0x5202 07FF	0x4202 0400 - 0x4202 07FF	1 K	GPIOB	Section 14.8.13 on page 499
	0x5202 0000 - 0x5202 03FF	0x4202 0000 - 0x4202 03FF	1 K	GPIOA	Section 14.8.12 on page 497
-	0x5200 0000 - 0x5201 FFFF	0x4200 0000 - 0x4201 FFFF	-	Reserved	-

Table 5. Memory map and peripheral register boundary addresses (continued)

Bus	Nonsecure callable boundary address(1)	Nonsecure boundary address(1)	Size (bytes)	Peripheral	Peripheral register map
AHB1	0x5003 4400 - 0x51FF FFFF	0x4003 4400 - 0x41FF FFFF	-	Reserved	-
	0x5003 4000 - 0x5003 43FF	0x4003 4000 - 0x4003 43FF	1 K	GTZC_MPCBB6(3)	Section 5.8.6 on page 164
	0x5003 3400 - 0x5003 3FFF	0x4003 3400 - 0x4003 3FFF	-	Reserved	-
	0x5003 3000 - 0x5003 33FF	0x4003 3000 - 0x4003 33FF	1 K	GTZC_MPCBB2(3)	Section 5.8.5 on page 163
	0x5003 2C00 - 0x5003 2FFF	0x4003 2C00 - 0x4003 2FFF	1 K	GTZC_MPCBB1(3)	
	0x5003 2800 - 0x5003 2BFF	0x4003 2800 - 0x4003 2BFF	1 K	GTZC_TZIC(3)	Section 5.7.13 on page 159
	0x5003 2400 - 0x5003 27FF	0x4003 2400 - 0x4003 27FF	1 K	GTZC_TZSC(3)	Section 5.6.8 on page 141
	0x5003 0800 - 0x5003 23FF	0x4003 0800 - 0x4003 23FF	-	Reserved	-
	0x5003 0400 - 0x5003 07FF	0x4003 0400 - 0x4003 07FF	1 K	ICACHE	Section 8.7.8 on page 257
	0x5002 7000 - 0x5003 03FF	0x4002 7000 - 0x4003 03FF	-	Reserved	-
	0x5002 6000 - 0x5002 6FFF	0x4002 6000 - 0x4002 6FFF	4 K	RAMCFG	Section 6.6.11 on page 175
	0x5002 4400 - 0x5002 5FFF	0x4002 4400 - 0x4002 5FFF	-	Reserved	-
	0x5002 4000 - 0x5002 43FF	0x4002 4000 - 0x4002 43FF	1 K	TSC	Section 23.6.11 on page 714
	0x5002 3400 - 0x5002 3FFF	0x4002 3400 - 0x4002 3FFF	-	Reserved	-
	0x5002 3000 - 0x5002 33FF	0x4002 3000 - 0x4002 33FF	1 K	CRC	Section 20.4.6 on page 623
	0x5002 2400 - 0x5002 2FFF	0x4002 2400 - 0x4002 2FFF	-	Reserved	-
	0x5002 2000 - 0x5002 23FF	0x4002 2000 - 0x4002 23FF	1 K	FLASH interface	Section 7.9.30 on page 237
	0x5002 1000 - 0x5002 1FFF	0x4002 1000 - 0x4002 1FFF	-	Reserved	-
	0x5002 0000 - 0x5002 0FFF	0x4002 0000 - 0x4002 0FFF	4 K	GPDMA1	Section 17.8.16 on page 589
APB2	0x5001 5800 - 0x5001 FFFF	0x4001 5800 - 0x4001 FFFF	-	Reserved	-
	0x5001 5400 - 0x5001 57FF	0x4001 5400 - 0x4001 57FF	1 K	SAI1(2)	
	0x5001 4C00 - 0x5001 53FF	0x4001 4C00 - 0x4001 53FF	-	Reserved	-
	0x5001 4800 - 0x5001 4BFF	0x4001 4800 - 0x4001 4BFF	1 K	TIM17(3)	Section 31.6.22 on page 1211
	0x5001 4400 - 0x5001 47FF	0x4001 4400 - 0x4001 47FF	1 K	TIM16	Section 31.6.22 on page 1211
	0x5001 3C00 - 0x5001 43FF	0x4001 3C00 - 0x4001 43FF	-	Reserved	-
	0x5001 3800 - 0x5001 3BFF	0x4001 3800 - 0x4001 3BFF	1 K	USART1	Section 39.8.17 on page 1538
	0x5001 3400 - 0x5001 37FF	0x4001 3400 - 0x4001 37FF	-	Reserved	-
	0x5001 3000 - 0x5001 33FF	0x4001 3000 - 0x4001 33FF	1 K	SPI1(3)	Section 41.8.15 on page 1647
	0x5001 2C00 - 0x5001 2FFF	0x4001 2C00 - 0x4001 2FFF	1 K	TIM1	Section 29.6.31 on page 1029
	0x5001 0000 - 0x5001 2BFF	0x4001 0000 - 0x4001 2BFF	-	Reserved	-

Table 5. Memory map and peripheral register boundary addresses (continued)

Bus	Nonsecure callable boundary address(1)	Nonsecure boundary address(1)	Size (bytes)	Peripheral	Peripheral register map
APB1	0x5000 9800 - 0x5000 FFFF	0x4000 9800 - 0x4000 FFFF	-	Reserved	-
	0x5000 9400 - 0x5000 97FF	0x4000 9400 - 0x4000 97FF	1 K	LPTIM2(3)	Section 32.7.16 on page 1257
	0x5000 5800 - 0x5000 93FF	0x4000 5800 - 0x4000 93FF	-	Reserved	-
	0x5000 5400 - 0x5000 57FF	0x4000 5400 - 0x4000 57FF	1 K	I2C1(3)	Section 38.9.13 on page 1449
	0x5000 4800 - 0x5000 53FF	0x4000 4800 - 0x4000 53FF	-	Reserved	-
	0x5000 4400 - 0x5000 47FF	0x4000 4400 - 0x4000 47FF	1 K	USART2(3)	Section 39.8.17 on page 1538
	0x5000 3400 - 0x5000 43FF	0x4000 3400 - 0x4000 43FF	-	Reserved	-
	0x5000 3000 - 0x5000 33FF	0x4000 3000 - 0x4000 33FF	1 K	IWDG	Section 34.7.7 on page 1275
	0x5000 2C00 - 0x5000 2FFF	0x4000 2C00 - 0x4000 2FFF	1 K	WWDG	Section 35.6.4 on page 1282
	0x5000 0800 - 0x5000 2BFF	0x4000 0800 - 0x4000 2BFF	-	Reserved	-
	0x5000 0400 - 0x5000 07FF	0x4000 0400 - 0x4000 07FF	1 K	TIM3(3)	Section 30.5.31 on page 1145
	0x5000 0000 - 0x5000 03FF	0x4000 0000 - 0x4000 03FF	1 K	TIM2	Section 30.5.31 on page 1145
AHB	0x3002 0000 - 0x4FFF FFFF	0x2002 0000 - 0x3FFF FFFF	-	Reserved	-
	0x3001 0000 - 0x3001 FFFF	0x2001 0000 - 0x2001 FFFF	64 K(4)	SRAM2	-
	0x3000 0000 - 0x3000 FFFF	0x2000 0000 - 0x2000 FFFF	64 K(5)	SRAM1	-
	0x0FF9 8000 - 0x2FFF FFFF	0x0FF9 8000 - 0x1FFF FFFF	-	Reserved	-
	0x0FF9 4000 - 0x0FF9 7FFF	0x0BF9 4000 - 0x0BF9 7FFF	16 K	Flash user options	Section 7.9.30 on page 237
	0x0FF9 0500 - 0x0FF9 3FFF	0x0BF9 0500 - 0x0BF9 3FFF	14.75 K	DESIG	Section 44.1.13 on page 1839
	0x0FF9 0200 - 0x0FF9 04FF	0x0BF9 0200 - 0x0BF9 04FF	-	Reserved	-
	0x0FF9 0000 - 0x0FF9 01FF	0x0BF9 0000 - 0x0BF9 01FF	512	OTP	-
	0x0FF8 8000 - 0x0FF8 FFFF	0x0BF8 8000 - 0x0BF8 FFFF	32 K	Bootloader	-
	0x0FF8 6000 - 0x0FF8 7FFF	0x0BF8 6000 - 0x0BF8 7FFF	8 K	RSS-Lib(3)	-
	0x0FF8 0000 - 0x0FF8 5FFF	0x0BF8 0000 - 0x0BF8 5FFF	24 K	RSS-Boot(3)	-
	0x0C10 0000 - 0x0FF7 FFFF	0x0810 0000 - 0x0BF7 FFFF	-	Reserved	-
	0x0C00 0000 - 0x0C0F FFFF	0x0800 0000 - 0x080F FFFF	1 M(6)	User flash	-
	0x0000 0000 - 0x0BFF FFFF	0x0000 0000 - 0x07FF FFFF	-	Reserved	-

1. Gray shaded fields are reserved.
2. Available only on STM32WBA54/5xx devices.
3. Available only on STM32WBA52/54/55xx devices.
4. Device-dependent (STM32WBA52/54/55xx 64-Kbyte SRAM2 offset 0x1 0000 - 0x1 FFFF, STM32WBA50xG 48-Kbyte SRAM2 offset 0x1 0000 - 0x1 BFFF, where 0x1 C000 - 0x1 FFFF is reserved).
5. Device-dependent (STM32WBA52/54/55xG 64-Kbyte SRAM1 offset 0x0 0000 - 0x0 FFFF, STM32WBA52/4/5xE 32-Kbyte SRAM1 offset 0x0 0000 - 0x 0 7FFF, where 0x0 8000 - 0x0 FFFF is reserved, STM32WBA50xx 16-Kbyte SRAM1 offset 0x0 0000 - 0x0 3FFF, where 0x0 4000 - 0x0 FFFF is reserved).
6. Device-dependent (STM32WBA5xxG 1-Mbyte User flash offset 0x00 0000 - 0x0F FFFF, STM32WBA5xxE 512-Kbyte user flash offset 0x00 0000 - 0x07 FFFF, where 0x08 0000 - 0x0F FFFF is reserved).

2.3.3 Embedded SRAM

The devices feature up to 128-Kbyte SRAMs:

- SRAM1 up to 64-Kbyte (STM32WBA52/54/55xG 64-Kbyte, STM32WBA52/54/55xE 32-Kbyte, STM32WBA50xG 16-Kbyte)
- SRAM2 up to 64-Kbyte (STM32WBA52/54/55xx 64-Kbyte, STM32WBA50xG 48-Kbyte)

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 DMA.

The CPU can access the SRAM1, and SRAM2 through the system bus or, when remapped in the I-CACHE, through the C-bus, depending on the selected address.

When TrustZone security is enabled, all SRAMs are secure after reset. The SRAM can be programmed as nonsecure 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 memory operations (program/erase) controlled through the FLASH registers, plus the security access control features. Refer to *Section 7: Embedded flash memory (FLASH)* for more details.

3 System security

The device is designed with a comprehensive set of security features, some of which are based on the standard Arm TrustZone® technology[a].

These 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 reuse, improving the interoperability, and minimizing the API fragmentation.

This section explains the different security features available on the device.

3.1 Key security features

- Resource isolation using privileged 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[a]
  - leveraging the on-chip immutable bootloader that supports the download of image through USART, I^2C, SPI, and JTAG
- Secure boot thanks to the unique boot entry feature and hide-protect area (HDP) mechanism[a]
- Secure storage, featuring:
  - Nonvolatile on-chip secure storage, protected with secure and HDP areas[a]
  - 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 enhanced storage technology, using hardware secret nonvolatile derived hardware unique key (DHUK), and application-defined volatile secret boot hardware key (BHK), both loadable by hardware to the DPA-resistant SAES engine[a]
- General purpose cryptographic acceleration
  - AES 256-bit engine, supporting ECB, CBC, CTR, GCM, and CCM chaining modes
  - Secure AES 256-bit security coprocessor, supporting ECB, CBC, CTR, GCM, and CCM chaining modes with side-channel countermeasures and mitigations[a]
  - 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 precertified
- 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

a. Available only on STM32WBA52/4/5x devices.

- Protected firmware distribution scheme, using TrustZone and RDP Level 0.5[a]
- Active tamper and protection
  - Six active inputs, six active output tamper pins, available in all power modes

3.2 Secure firmware install[a]

The secure firmware install (SFI) is an immutable secure service embedded in the devices. The SFI allows secure and counted installation of OEM firmware in an untrusted production environment (such as OEM contract manufacturer).

The confidentiality of the installed images written in the internal flash memory is 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 3.10*)

Further information can be found in AN4992 *STM32 MCUs secure firmware install (SFI) overview*, available on *www.st.com*.

3.3 Secure boot[a]

Secure boot, whose specifically designed features are described in this section, 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.

The device Trusted Firmware-M (TFM) application, supported by the STM32 ecosystem, provides a root of trust solution including secure boot functions.

In the devices, the secure boot benefits 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 3.10*)

a. Available only on STM32WBA52/4/5x devices.

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 *Section 3.3.2*), 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 perform 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 SYSCFG_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.

3.4 Secure firmware update[a]

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 nonsecure parts, or on the secure (respectively nonsecure) part of the firmware image, independently.

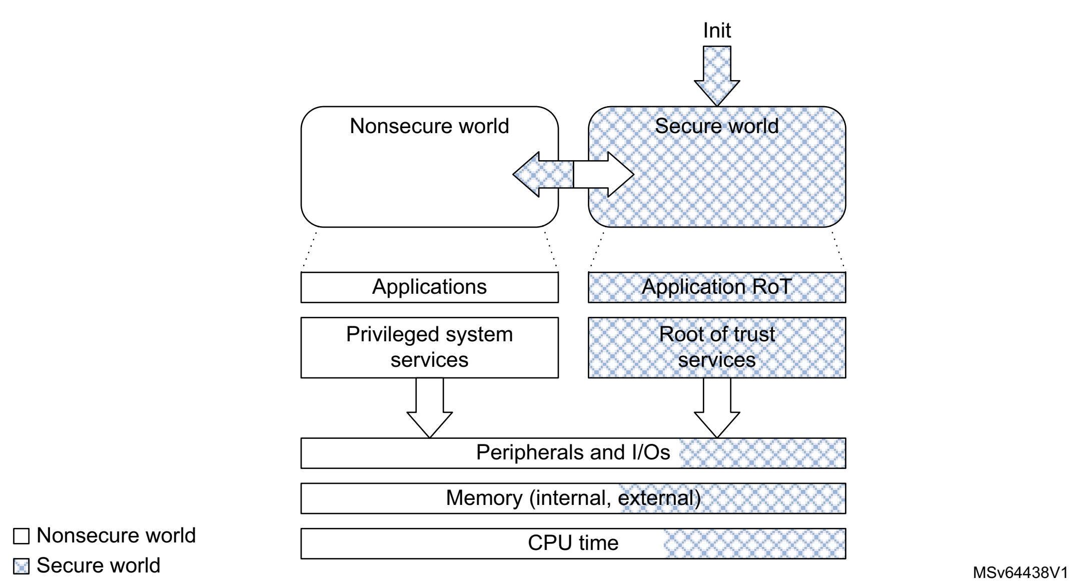
In the devices, the secure update application leverages the same hardware security as the firmware install described in *Section 3.2*.

3.5 Resource isolation using TrustZone[a]

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

a. Available only on STM32WBA52/4/5x devices.

Figure 3. Secure/nonsecure 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 nonsecure world.

Note: The secure code is typically small and rarely modified, while nonsecure 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 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/nonsecure world change
- two memory map views of a shared 4-Gbyte address space
- a low interrupt latency for both secure and nonsecure domains, and a new interrupt configuration for security grouping and priority setting
- separated exception vector tables for the secure and nonsecure exceptions
- Micro-coded context preservation
- banking of specific registers across secure/nonsecure states, including stack pointers with stack-limit checkers
- banking of the following Cortex-M33 programmable components (two separate units for secure and nonsecure):
  - SysTick timer
  - MPU configuration registers (eight MPU regions in secure, eight in nonsecure)
  - some of the system control block (SCB) registers
- new system exception (SecureFault) for handling of security violations
- configurable debug support, as defined in *Section 3.12*

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

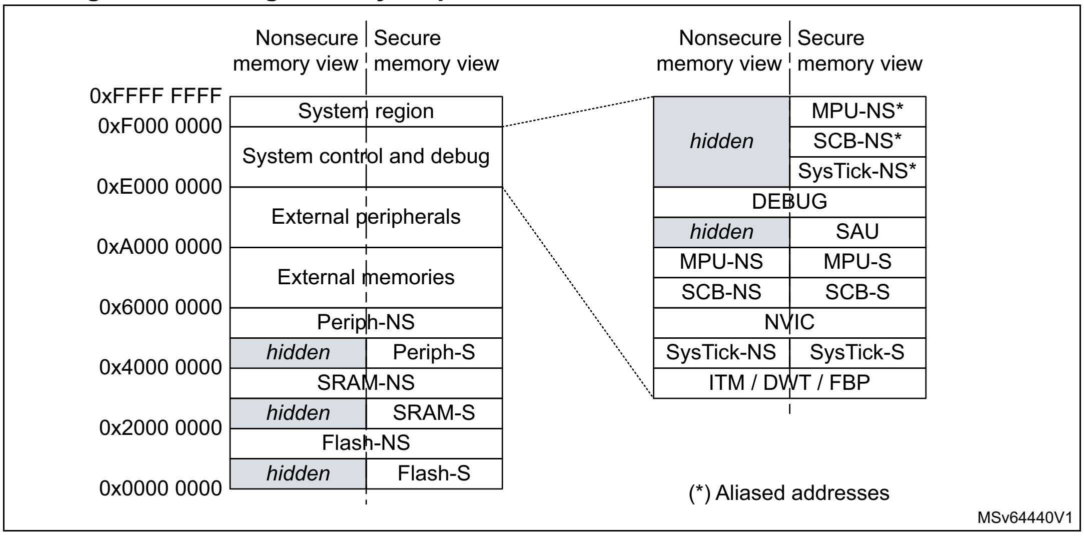
3.5.3 Memory and peripheral allocation using IDAU/SAU

Security attributes

The Armv8-M nonsecure memory view (see *Figure 4*) 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.

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

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



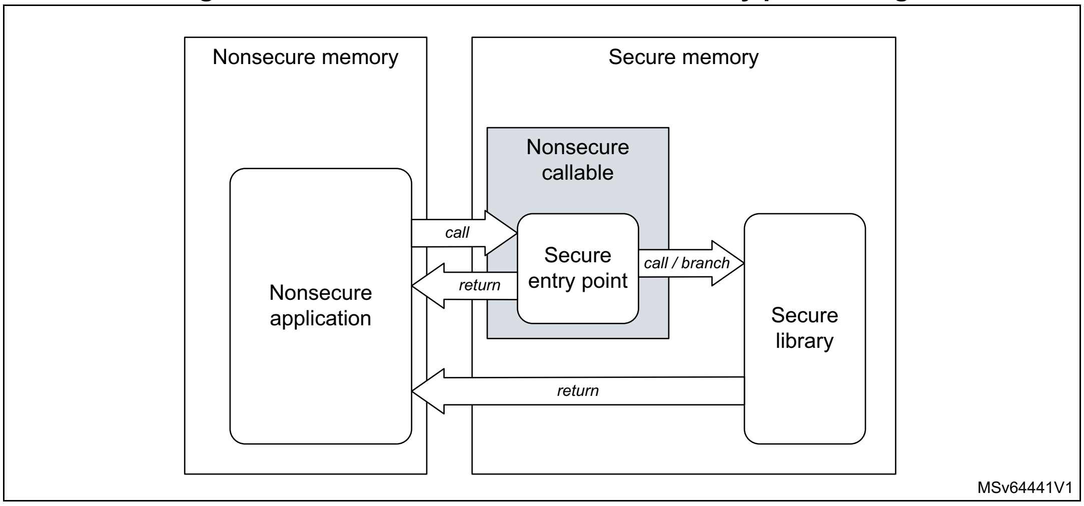
Note: *The devices have no support for external memories and external peripherals.*

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 nonsecure state only executes from nonsecure (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 nonsecure state, the CPU is limited to nonsecure memories.

To manage transitions to the secure world, developers must create nonsecure 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 nonsecure code to call a secure function (see *Figure 5*).

Figure 5. Secure world transition and memory partitioning



Programming security attributes

In Cortex-M33, the static implementation-defined attribution unit (IDAU) works with the programmable security attribution unit (SAU) to assign a specific security attribute (S, NS, or NSC) to a specific address, as shown in *Table 6*.

Table 6. Configuring security attributes with IDAU and SAU

IDAU security attribution	SAU security attribution(1)	Final security attribution
Nonsecure	Secure	Secure
	Secure-NSC	Secure-NSC
	Nonsecure	Nonsecure
Secure-NSC	Secure	Secure
	Nonsecure	Secure-NSC

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

The SAU can be configured by the Cortex-M33 only in the secure-privileged 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 *Table 6*.

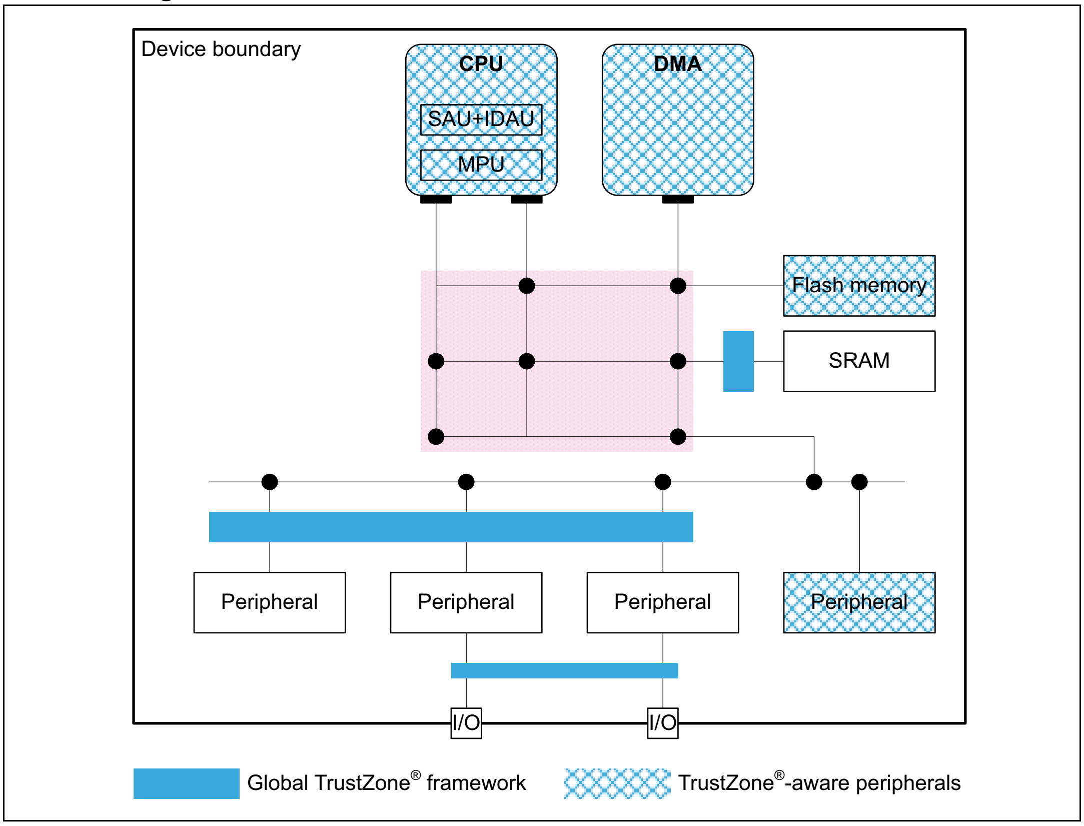
Note: The SAU/IDAU settings are applicable only to the Cortex-M33. *The other masters (like DMA) are not affected by these policies.*

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 nonsecure worlds. Unlike the SAU/IDAU, the GTZC can protect legacy memories and peripherals against nonsecure transactions coming from other masters than the Cortex-M33.

Figure 6. Global TrustZone framework and TrustZone awareness



Securing peripherals with TZSC

When the TrustZone security is active, a peripheral is either securable through the TZSC block 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, 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.

TrustZone-aware AHB masters like Cortex-M33 or DMAs, drive a secure signal in the AHB interconnect, according to their security mode, independently from the TZSC.

Note: Like with TrustZone, a peripheral can be made privileged-only with TZSC (see Section 3.6.3). In this case, if this peripheral is master on the interconnect, it automatically issues privileged transactions.

Securing memories with MPCBB

The MPCBB blocks in GTZC provide the capability to configure the security and privilege of embedded SRAM blocks, as defined in *Table 7*.

Table 7. MPCBBx resources

Memory	MPC resource
SRAM1	GTZC1_MPCBB1
SRAM2	GTZC1_MPCBB2
2.4 GHz RADIO RXTXRAM	GTZC1_MPCBB6

For more information, refer to *Section 5: Global TrustZone controller (GTZC)*.

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 region size can be changed by a secure software. This software 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 3.6.3).

When the TrustZone is activated, peripherals are set as nonsecure and unprivileged after reset.

TrustZone-aware peripherals

The devices include the following TrustZone-aware peripherals:

- GPIOA to GPIOC and GPIOH
- GTZC1_MPCBBx, GTZC1_TZIC and GTZC1_TZSC (GTZC blocks)
- EXTI
- Flash memory
- RCC and PWR
- GPDMA
- SYSCFG
- RTC and TAMP
- MCU debug unit DBGMCU
- HSEM

Illegal accesses to those peripherals are monitored through the TZIC registers, as described in *Section 5: Global TrustZone controller (GTZC)*. For more details, refer to *Section 3.5.5*.

Illegal access controller with TZIC

The TZIC block in GTZC gathers all illegal access events originated from sources protected by GTZC or TrustZone-aware peripherals, generating a 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 towards the NVIC.

Note: By default, all events are masked.

3.5.5 Managing security in TrustZone-aware peripherals

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 *Figure 7*, are available to the application:

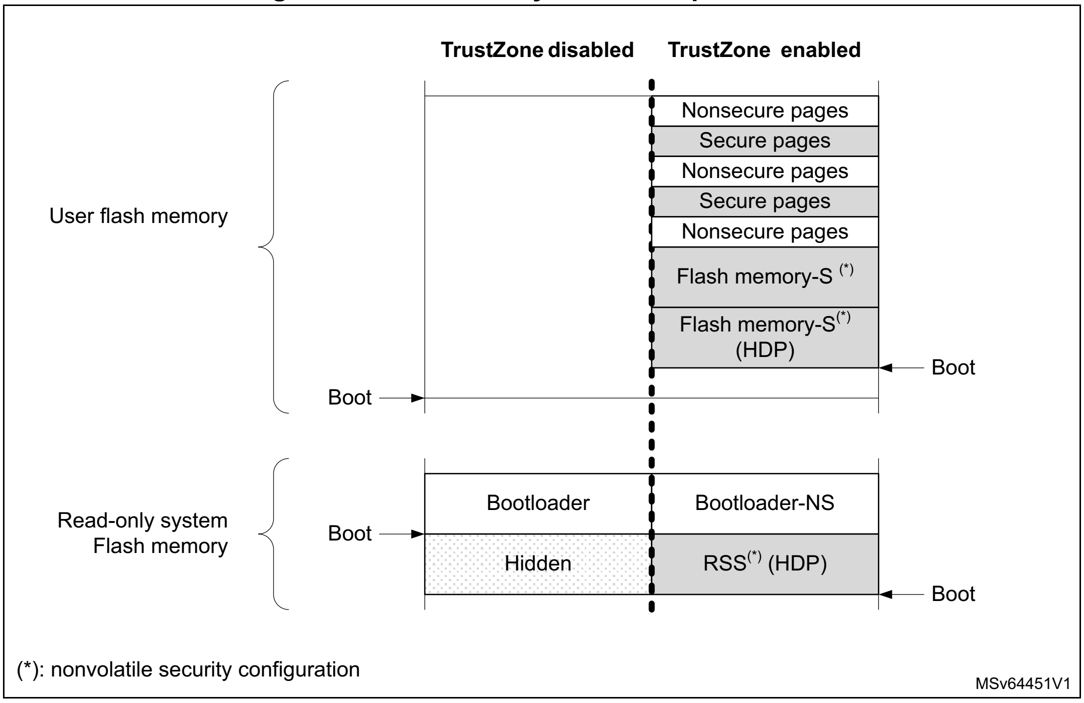
- nonvolatile user secure areas, defined with nonvolatile secure user option bytes
  - watermark-based secure only area
  - secure hide protection (HDP) area, stickily hidden after boot
- volatile user secure pages, defined with volatile secure registers (lost after reset)
  - any page set as nonsecure (such as 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 nonsecure in Cortex-M33 IDAU/SAU. In this case, nonsecure accesses by the CPU to the flash memory are denied.

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

Figure 7. Flash memory TrustZone protections



As shown above, when TrustZone is activated (TZEN = 1), the application code can use the HDP area that is part of the flash watermark-based secure area. When the application sets the HDP_ACCDIS bit, data read, write, and instruction fetch on this HDP area are denied until the 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.

Direct memory access controller (GPDMA)

When a DMA channel x is defined as secure (SECx = 1 in GPDMA_SECCFGR), the source and destination transfers can be independently set as secure or nonsecure by a secure application using SSEC and DSEC bits in GPDMA_CxTR1. *Table 8* summarizes the security options available in each DMA channel.

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

Destination type	Secure DMA channel x (SECx = 1)		Nonsecure DMA channel y (SECy = 0)	
	Secure source	Nonsecure source	Secure source	Nonsecure source
Secure destination	OK	OK(2)	Transfer blocked	
Nonsecure destination	OK(3)	OK(4)	Transfer blocked	OK

1. When a transfer is blocked, it 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 possible only if SSEC = 0, otherwise the transfer is blocked.
3. If the destination is a memory, the transfer is possible only if DSEC = 0, otherwise the transfer is blocked.
4. The memory-to-memory transfer is possible only if SSEC = 0 and DSEC = 0, otherwise it is blocked.

When a channel is configured as secure:

- Registers allocated to this channel (excluding GPDMA_SECCFGR, GPDMA_PRIVCFGR and GPDMA_RCFGLOCKR) are read as zero. Write are ignored for nonsecure accesses. A secure illegal access event may also be triggered toward the TZIC peripheral.
  - Writes to GPDMA_SECCFGR and GPDMA_RCFGLOCKR must be secure. For each bit in GPDMA_PRIVCFGR, write must be secure if the corresponding bit in 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 nonsecure 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 nonsecure in linked-list mode, the loading of the next linked-list data structure from memory is performed with nonsecure transfers. See 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
- wake-up (WKUP) pins definition
- voltage detection
- backup domain control

Other PWR configuration bits become secure:

- when the system clock selection is secure in the RCC: the voltage scaling (VOS) configuration become secure
- when a GPIO is configured as secure: its corresponding bit for standby mode retention configuration becomes secure
- when the 2.4 GHz RADIO is configured as secure: its corresponding control bits become secure

See *Section 11: 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.

When a peripheral is defined as secure, its related clock, reset, and clock source selection in the RCC, become secure and in some case the selection of clock source during low-power modes as well.

Note: Refer to Table 3 and Table 4 for the list of securable and TrustZone-aware peripherals.

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

- external clock (such as HSE32 or LSE), internal oscillator (such as HSI16, or LSI)
- PLL
- AHB and APB prescalers
- system clock source, SysTick, and MCO clock output selection
- reset flag clearing

See *Section 12: Reset and clock control (RCC)* for details.

Real time clock (RTC)

Like all TrustZone-aware peripherals, a nonsecure 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 nonsecure modes. The secure boot code can then change its security setup, making registers Alarm A, alarm B, wake-up 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 nonsecure 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 an NRST pin reset.

See *Section 36: Real-time clock (RTC)* for more details.

Tamper and backup registers (TAMP)

Like all TrustZone-aware peripherals, a nonsecure 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 nonsecure modes. The secure boot code can change its 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 (except for TAMP_SECCFGR, TAMP_PRIVCFGR and TAMP_MISR) returns zero if the access is nonsecure. 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 secure, write secure
  - protection zone 2 is read nonsecure, write secure
  - protection zone 3 is read nonsecure, write nonsecure

Note: The TAMP security configuration is not affected by an NRST reset.

See *Section 37: 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 the 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 nonsecure access.

When digital alternate function is used (input/output mode), 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 *Table 9*.

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	-
Nonsecure	Secure	I/O data	Peripheral data	Out of reset configuration
Secure	Nonsecure	0	0	-
Nonsecure	Nonsecure	I/O data	Peripheral data	-

When an analog function with an analog switch is used, the connection to the peripherals listed in *Table 10*, is blocked by hardware when the peripheral is nonsecure and the I/O is secure.

Table 10. Nonsecure peripheral functions not connected to secure I/Os

Peripheral	Analog function(1)	Input	Output	How to set the peripheral or the function as secure
ADC4	ADC4_INy (y = 1 to 10)	X	-	Set ADC4SEC in GTZC1_TZSC
COMPx(2) (x = 1, 2)	COMPx_INM1 COMPx_INP1			Set COMPSEC in GTZC1_TZSC

1. Used to find the I/O corresponding to the signal/function on the package (refer to the product datasheet).
2. Available only on STM32WBA54/55xx devices.

Table 11 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 nonsecure.

For example, when a secure application sets an I/O port pin as secure to be used as LPTIM2_OUT, if the RTC_OUTx is nonsecure, it can be programmed to output data to the I/O port pin, potentially causing malfunction to the secure application.

Similarly, when a secure application sets an I/O port pin as secure to be used as USART1_TX, if TAMP_INx is nonsecure, it can be programmed to capture the USART input traffic.

It is important that, for each case described in *Table 11*, 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), the application must configure the TAMP as secure, even if it does not use it.

Note: How to make a peripheral secure is summarized in the right column of Table 11.

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

Peripheral	Signal(1)	Input	Output	How to set the peripheral or the function as secure
TAMP	TAMP_INx (x = 1 to 6)	X	-	Set TAMPSEC in TAMP_SECCFGR
	TAMP_OUTx (x = 1 to 6)	-	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 15)	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 14: 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 nonsecure 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 19: 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 nonsecure 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.

Hardware semaphore (HSEM)

Like all TrustZone-aware peripherals, when the TrustZone security is activated (TZEN = 1), a nonsecure read/write access to a secured HSEM register is RAZ/WI. Such access also generates an illegal access event that triggers a secure illegal access interrupt if the HSEM illegal access event is not masked in the TZIC.

See *Section 13: Hardware semaphore (HSEM)* for more details.

Microcontroller debug unit (DBGMCU)

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

- low-power modes behavior during debug
- peripheral freeze during debug, applicable to I2Cs, IWDG, WWDG, timers, low-power timers, and DMA channels

The DBGCMU is a TrustZone-aware peripheral, managing accesses to its control registers as described in *Table 12*.

Table 12. TrustZone-aware DBGMCU accesses management

Debug profile	Peripheral status(1)	DBG_xx_STOP control bits	
		Write access	Read access
Nonsecure 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 or the TrustZone-aware peripheral feature.
2. Secure access from debugger is converted to nonsecure access in the device.

Refer to *Section 45.13: Microcontroller debug unit (DBGMCU)* for more details.

3.5.6 Activating TrustZone security

The TrustZone is deactivated by default in the device. 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 nonvolatile secure areas (with the HDP zone), are defined with nonvolatile registers FLASH_SECWMR. Default secure option bytes setup is all user flash secure, without HDP area defined.
  - Volatile block-based security attributions of the flash memory are nonsecure.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 no-secure.
- All GPIOs are secure
- All GPDMA channels are nonsecure
- Backup registers are nonsecure
- Peripherals and GTZC:
  - Securable peripherals are nonsecure and unprivileged.
  - TrustZone-aware peripherals are nonsecure, with their secure configuration registers being secure.
  - All illegal access interrupts in GTZC_TZIC are disabled.

Note: Refer to Table 3 and Table 4 for the list of securable and TrustZone-aware peripherals.

3.5.7 Deactivating TrustZone security

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

Note: Such RDP regression triggers the erase of embedded memories (flash memory, SRAM1, SRAM2, PKA SRAM and TAMP backup registers), and the reset of all peripherals, including all crypto engines.

Note: Before launching an RDP regression, the software must invalidate the ICACHE and wait for the BUSYF bit to get cleared.

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

- The nonvolatile secure area of the embedded flash memory is deactivated, including the HDP area.
- The NVIC only manages nonsecure 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 3.6.3 for details).

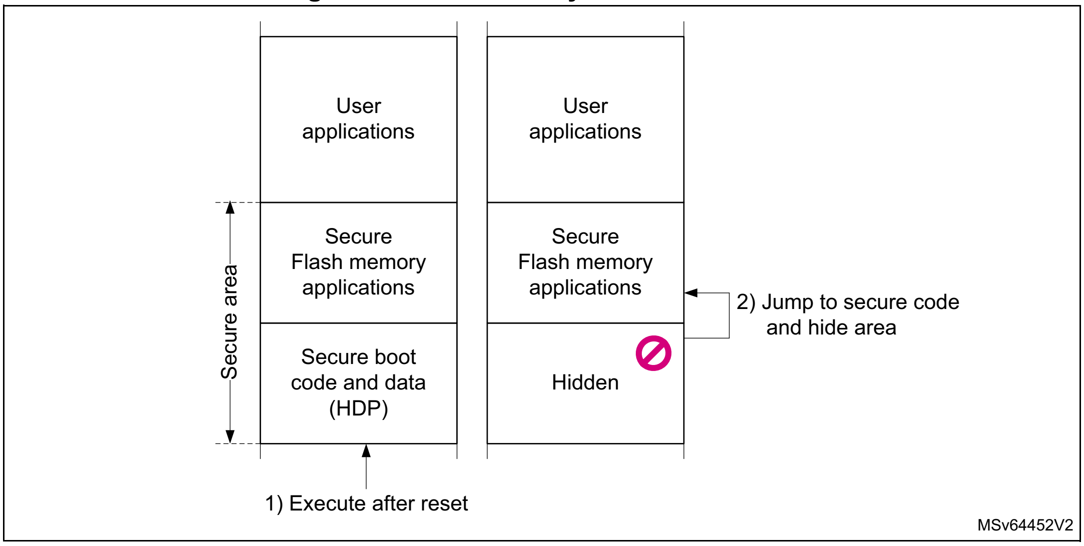
3.6 Isolation of other resources[a]

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 in the watermarked-secure area (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 shown in *Figure 8*.

Figure 8. Flash memory secure HDP area



When the HDPEN and HDP_ACCDIS bits are set, data read, write, and instruction fetch on the area defined by SECWM_PSTRT and HDP_PEND option bytes are denied until the next device reset.

Note: Mass 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 RSSLIB functions

The RSS provides runtime services thanks to RSS library. As other microcontroller peripherals features and mapping, the RSS library functions are exposed to user within the CMSIS device header file provided by the STM32Cube firmware package. Please refer to UM3131 to get more details regarding STM32Cube firmware package. RSS library functions are named RSSLIB functions hereafter.

The user firmware calls RSSLIB functions using RSSLIB_PFUNC C defined macro, that points to a location within nonsecure system memory. Hence prior calling RSSLIB functions, the secure user firmware must define a nonsecure region above this location within SAU of the Cortex®-M33. This nonsecure region starts from

a. Available only on STM32WBA52/4/5xx devices.

RSSLIB_SYS_FLASH_NS_PFUNC_START up to RSSLIB_SYS_FLASH_NS_PFUNC_END. These last addresses are provided within the CMSIS device header file. The user can set this nonsecure region either by using the CMSIS system partition header file or by implementing its own code for SAU setup. The CMSIS system partition header file is part of the STM32Cube firmware package.

RSSLIB functions are split between non-secure callable and secure callable function.

The RSS library functions are described within sections hereafter.

CloseExitHDP

Bootloader ID:

CloseExitHDP1 function

Secure attribute:

Secure callable function.

Prototype:

`uint32_t CloseExitHDP(uint32_t HdpArea, uint32_t VectorTableAddr)`

Arguments:

- HdpArea:
  Values can be: RSSLIB_HDP_AREA1_Msk.
- VectorTableAddr:
  Input parameter, address of the next vector table to apply.
  The vector table format is the one used by the Cortex®-M33 core.

Description:

The user calls CloseExitHDP to close flash HDP secure memory area and jump to the reset handler embedded within the vector table which address is passed as input parameter.

CloseExitHDP sets the SP provided by the passed vector table, however it is up to the caller to first set the new vector table. Then it clears all general-purpose.

A Cortex®-M33 registers (r0, r1, …) before jumping to new vector table reset handler.

On successful execution, the function does not return and does not push LR onto the stack.

In case of failure (bad input parameter value), this function returns RSSLIB_ERROR.

Refer to section *Section 7.5.3: Secure hide protection (HDP)* to get more details on flash memory HDP protection.

3.6.3 Resource isolation using Cortex privileged mode

In parallel to the TrustZone isolation described in *Section 3.5*, the hardware and software resources 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.

- 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 privileged-only.

    The list of securable peripherals can be found in *Table 3*.
- Securing memories with MPCBB (privileged-only)

  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 nonsecure privileged bit NSPRIV is set in FLASH_PRIVCFGR, reading and writing the flash nonsecure 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 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 register FLASH_PRIVBBRx. An unprivileged page is accessible either by a privileged or by an 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 to unprivileged code only for unprivileged pages or memory.

- *Direct memory access controllers (GPDMA)*
  When a DMA channel x is defined as privileged (PRIVx = 1 in GPDMA_PRIVCFGR), special rules apply when accessing privileged/unprivileged source or destination (see *Table 13*).

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 0s.

  See *Section 17: General purpose direct memory access controller (GPDMA)* for more details.

- *Power control (PWR)*
  By default, after a 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 nonsecure privileged bit NSPRIV is set in PWR_PRIVCFGR, reading and writing the PWR nonsecure registers are possible only in privileged mode.
  See *Section 11: Power control (PWR)* for details.
- *Secure clock and reset (RCC)*
  By default, after a 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 the nonsecure privileged bit NSPRIV is set in RCC_PRIVCFGR, reading and writing the RCC nonsecure bits are possible only in privileged mode.
  See *Section 12: 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 the 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 36: Real-time clock (RTC)* for details.

- *Tamper and backup registers (TAMP)*

  By default after a 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 privileged and unprivileged modes. All the other TAMP registers can be read only in privileged 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 nonsecure).

- *Extended interrupts and event controller (EXTI)*

  The EXTI peripheral can 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 19: 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_CSLCKR, 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 the Cortex-M33 operating mode (privileged, unprivileged), and the data/instruction fetch.

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

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 areais defined through option bytes, writing the start and end addresses: two write-protected areas can be defined, with page size granularity.

WRP areas can be modified through option byte changes unless the corresponding FLASH_WRPA/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: Mass erase aborts when it contains a write-protected area (WRP or HDP area).

a. Available only on STM32WBA52/4/5xx devices.

3.7.3 Tamper detection and response

Principle

The devices include active protection of critical security assets attacks, with the following features:

- erasing of device secrets upon tamper detection
- improved guarantee of safe execution for the CPU and its associated security peripherals, including:
  - out-of-range clocking LSE detection
  - security watchdog IWDG clocked by the internal oscillator LSI
  - possible selection of internal oscillator HSI16 as system clock
  - secure ADC4 window watchdog monitoring

See *Section 37: Tamper and backup registers (TAMP)* for more details.

Tamper detection sources

The devices support six active input/output pins, allowing three independent active-tamper meshes, or up to five meshes if the same output pin is shared by several input pins (for a total of six active-tamper I/Os).

The active pins when clocked by the LSE are functional in all system operating modes (Run, Sleep, Stop, and Standby).

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

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

The internal tamper sources are listed in *Table 14*.

Table 14. Internal tampers in TAMP

Tamper input	Tamper source
itamp3	LSE clock security monitoring
itamp5	RTC calendar overflow (rtc_calovf)
itamp6	JTAG/SWD access when RDP > 0
itamp7, 12, 13	Voltage monitoring (V_{CORE}, V_{SENSE}) through ADC4 analog watchdog, functional down to Stop 1 mode
itamp8	TAMP monotonic counter overflow
itamp9	Fault generation for cryptographic peripherals (SAES, PKA, AES, RNG)
itamp11	IWDG timeout and potential tamper (IWDG reset when at least one enabled tamper flag is set)

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 TAMPxPOM bit in TAMP_CR2 (for tamper pins) or ITAMPxPOM bit in TAMP_CR3 (for internal tamper) is cleared. The erasable secrets are:
  - keys stored in backup registers (x32), in AES and HASH
  - keys stored in PKA SRAM
  - other secrets stored in SRAM2 and CPU instruction cache memory
  - nonvolatile information used to derive the DHUK in SAES is zeroed until complete SRAM2 erase

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.

Device secrets erase is also triggered by setting the BKERASE bit in TAMP_CR2.

Software filtering mechanism

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

In such a 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, SRAM2: read-as-zero, write-ignored
- AES, SAES, and HASH peripherals: automatically reset.
- 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

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.

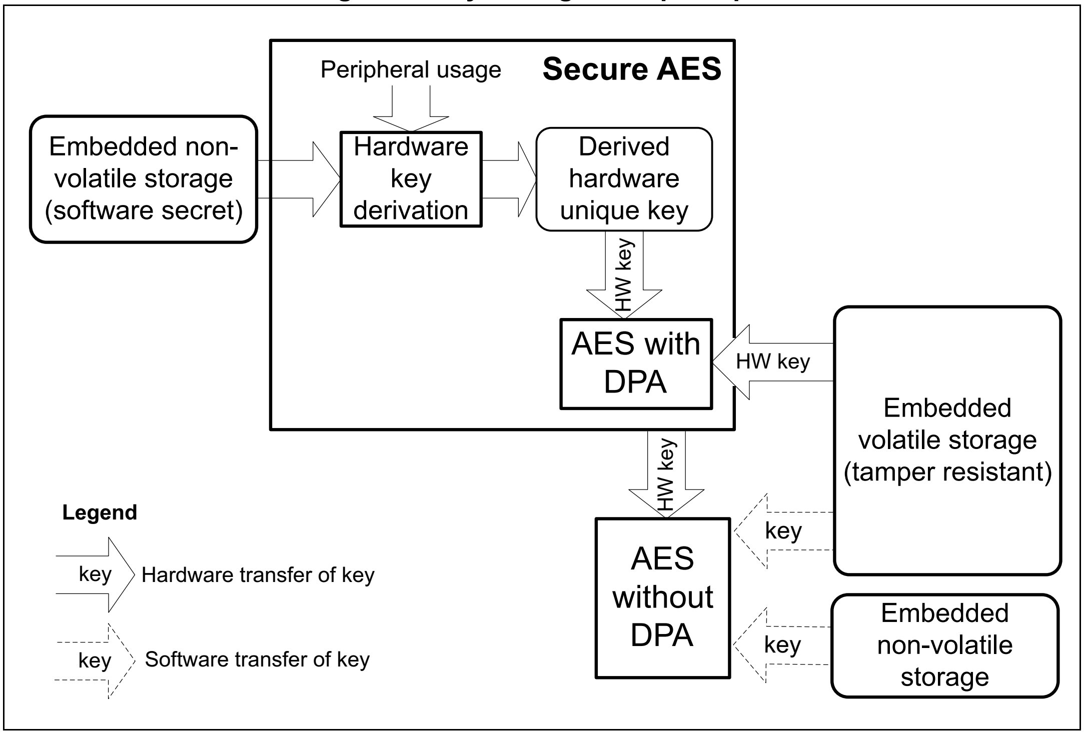
3.8 Secure storage[a]

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

Figure 9 shows how key management service application can use the AES engine, for example to compute decryption keys. A nonvolatile key can be stored in the embedded secure HDP area (see *Section 3.6.1*), while volatile keys can be stored in registers in the TrustZone-aware TAMP. *Figure 9* also shows keys that are managed only by hardware (like DHUK). More information on those hardware keys can be found in *Section 3.8.1*.

a. Available only on STM32WBA52/4/5xx devices.

Figure 9. 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:

1. DHUK: derived key based on a 256-bit nonvolatile device unique secret in flash memory. The generation of this key takes into account the TrustZone state and key use state (KMOD).
2. 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 the next reset.
3. XORK: result of an XOR of BHK and DHUK

Those keys can be used:

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

3.9 Unique ID

The devices store a 96-bit ID that is unique to each device (see *Section 46.1.2: DESIG 96-bit unique device ID register 1 (DESIG_UIDR1)*).

Application services can use this unique identity key to identify the product in the cloud network, or to 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*).

3.10 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 engine 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 purposes, 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.

3.10.1 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 28: Public key accelerator (PKA)* for more details.

Table 16. Accelerated cryptographic operations

Operations	Algorithm	Specification	Key length (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
Authenticated encryption or decryption		NIST SP800-38C NIST SP800-38D		GCM, CCM
Cipher-based message authentication code		NIST SP800-38D		GMAC

Table 16. Accelerated cryptographic operations (continued)

Operations	Algorithm	Specification	Key length (in bit)	Modes
Checksum	MD5	IETF RFC 1321	N/A	Digest 128-bit
	SHA-1	FIPS PUB 180-4		Digest 160-bit
Cryptographic hash	SHA2-224	FIPS PUB 180-4		Digest 224-bit
	SHA2-256			Digest 256-bit
Keyed-hashing for message authentication	HMAC	FIPS PUB 198-1 IETF RFC 2104	Short, long (>64 bytes)	-
Encryption/decryption key-pair generation[3]	RSA	IETF RFC 8017 NIST SP800-56B	Up to 4160	RSAES-OAEP
Signature[3] 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	*Table 241: Family of supported curves for ECC operations*
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. Private key cryptography protected against side-channel and timing attacks.

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

3.10.2 Secure AES coprocessor (SAES)[a]

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

Due to these counter-measures the SAES is slower than the AES, to provide best-in-class side-channel protections.

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.

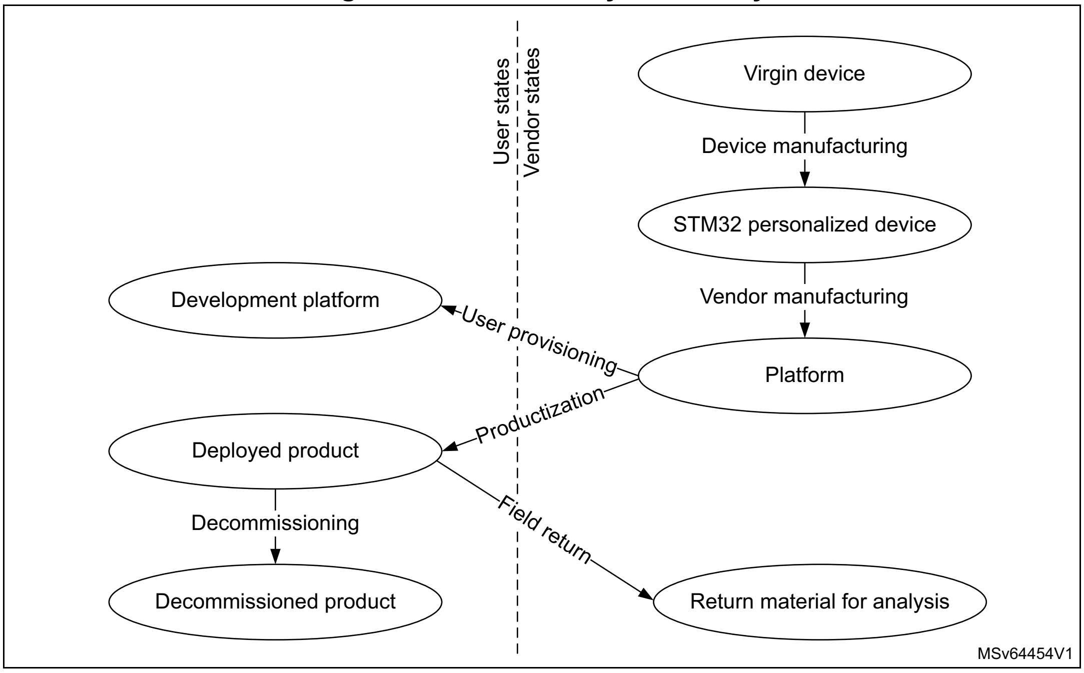
For more information, refer to *Section 26: Secure AES coprocessor (SAES)*.

3.11 Product life cycle

A typical IoT device life cycle is summarized in *Figure 10*. For each step, the devices propose secure life cycle management mechanisms embedded in the hardware.

a. Available only on STM32WBA52/4/5xx devices.

Figure 10. 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 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.

3.11.1 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 *Table 18*.

Table 18. Typical product life-cycle phases

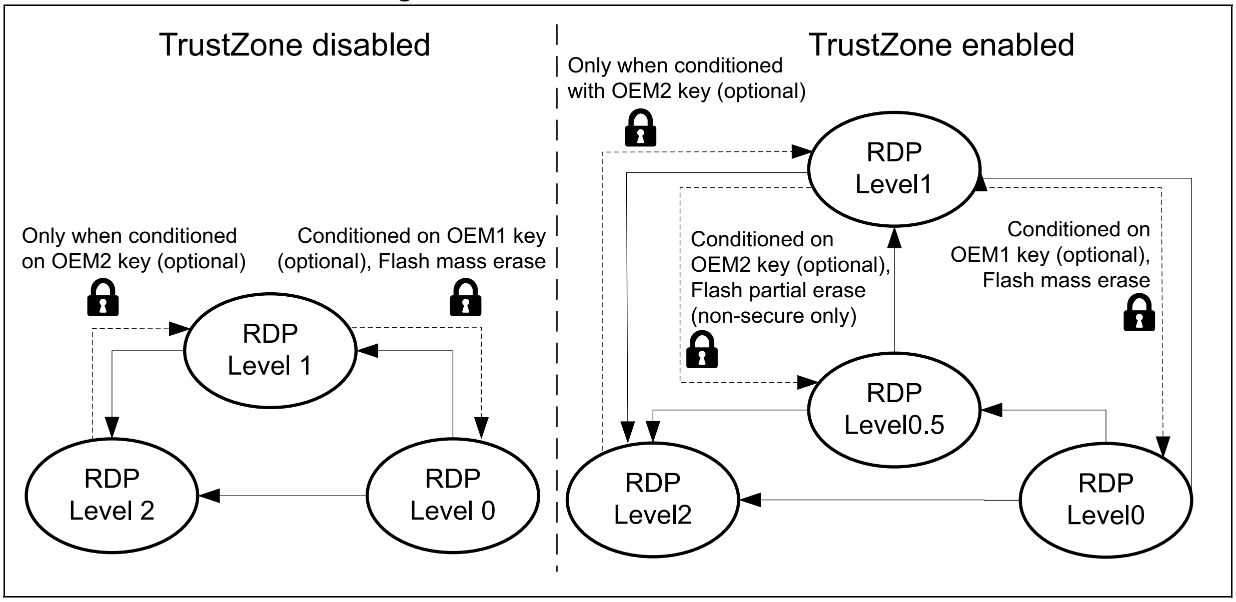
RDP protection level		Debug	Comments
0	Device open	Secure[(1)] and nonsecure	Boot address must target a secure area when TrustZone is enabled (secure SRAM, secure flash memory, RSS in system flash memory). Both OEM1 and OEM2 unlocking keys can be provisioned in the User options.
0.5[(2)]	Device partially closed (closed secure)	Nonsecure 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 nonsecure flash memory is allowed when debug is connected. Both OEM1 and OEM2 unlocking keys can be provisioned in the User options.
1	Device memories protected	Nonsecure only (conditioned)	Boot address must target the secure user flash memory. Accesses to nonsecure flash memory, SRAM2, and backup registers are not allowed when debug is connected. Both OEM1 and OEM2 unlocking keys can be provisioned in the User options.
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*).

1. Debug is not available when executing RSS code.
2. Only applicable when TrustZone security is activated in the product (STM32WBA52/4/5xx devices only).

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 *Figure 11*, can be requested (when available) through the debug interface or via the system bootloader.

Figure 11. RDP level transition scheme



The user flash memory is automatically erased, either partially or in totality, during an RDP regression from level 1. 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 nonsecure embedded flash memory is erased, keeping functional, for example, the secure boot and the secure firmware update. In all regressions from RDP Level 1 the OTP area in the flash memory is kept, and targeted device secrets are erased. No secrets must be stored in the OTP as they are nonsecure. The same secrets are erased, as those erased as response to tamper, defined in *Section 3.7.3*.

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

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

RDP regression and erase

When regressing RDP to level 0.5 or level 0, the SRAM1, SRAM2, PKA SRAM memories and TAMP backup registers are erased.

Note: Before launching an RDP regression, the software must invalidate the ICACHE and wait for the BUSYF bit to get cleared.

RDP unlocking sequences

The use of the two OEM password keys described in *Figure 11* is 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. OEM key 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 unlock sequence A (see below)
				2	Automatic regression to Level 1 triggered upon successful OEM2 unlock sequence B (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 level 2 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 nonsecure 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 unlock sequence A, 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 nonsecure part of the embedded flash memory and the 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 unlock sequence B, 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 applied.
  - In case of mismatch value, the RDP regression is blocked and RDP Level 2 protections are enforced until the 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 deactivated (OEM2LOCK = 0 and RDP Level = 2), a 32-bit device specific quantity can always be read through 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.

3.11.2 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
  - nonsecure boot address option bytes set in user flash memory
- If TrustZone is enabled (TZEN = 1)[a]
  - 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 OEM2 key has been successfully provisioned, and OEM2LOCK option bit is set.

3.12 Access controlled debug

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

3.12.1 Debug protection with readout protection (RDP)

As described in *Section 3.11.1*, the hardware RDP mechanism automatically controls the accesses to the device debug and test. The protection of these debug features are defined in *Table 20*. Possible password-based regressions are described in *Section 3.11.1*.

Table 20. Debug protection with RDP

RDP protection level		Debug features protection
0	Device open	Any debug[1]
0.5[2]	Device partially closed	Secure debug is no longer available.
1	Device memories protected	Nonsecure debug can no longer debug code and data stored in the embedded flash memory, SRAM2, and backup registers.
2	Device closed	JTAG is physically deactivated, unless it is kept operational only for password key injection (OEM2LOCK = 1). See *Section 3.11.1*.

1. Including ST engineering test modes, used for field returns.
2. Only applicable when TrustZone security is activated in the product (STM32WBA52/4/5xx devices only).

a. Available only on STM32WBA52/4/5xx devices.

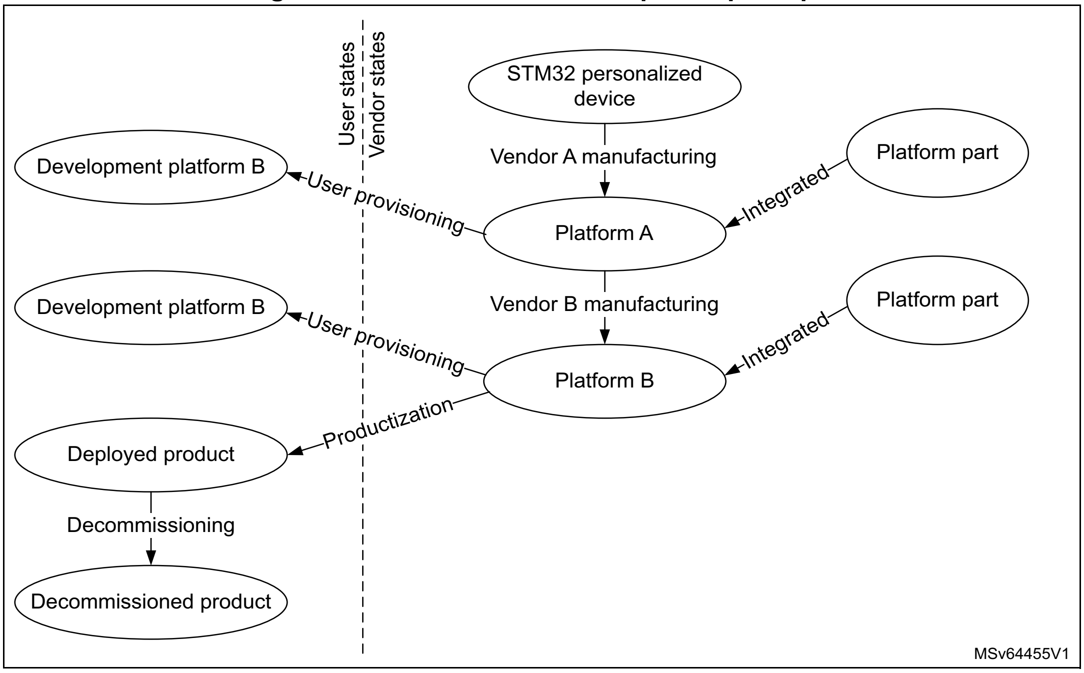
3.13 Software intellectual property protection and collaborative development

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 *Figure 12*. Starting from a personalized device sold by STMicroelectronics, a vendor A can integrate a portion of hardware and software on a platform A, which 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 12. Collaborative development principle



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

3.13.1 Software intellectual property protection with RDP

As described in *Section 3.11.1*, the hardware RDP mechanism automatically controls the accesses to secrets provisioned in the device. The protection of these secrets is defined in *Table 21*.

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, SRAM2 and backup registers are no more accessible via debugger.
Level 2	Device closed	All data and code stored in the device cannot be dumped, debugged or traced.

1. Only applicable when TrustZone® security is activated in the product (STM32WBA52/4/5xx devices only).

3.13.2 Other software intellectual property protections

The device additional protections to software intellectual property are:

- Invasive attacks such as physical tampering 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, NSWBOOT0, NSBOOTADDx/SECBOOTADD0, and TZEN option bytes are used to select the boot memory address, which 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 the option bit NBOOT0, 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 program the flash memory by using USART, I2C, or SPI in device mode.

Table 22 details the boot modes when TrustZone is disabled, and *Table 23* when enabled(a).

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

BOOT0(1)	Boot address option bytes selection		Boot initial VTOR_NS
	Nonsecure	ST programmed default value	
0	NSBOOTADD0	0x0800 0000 - User flash	NSBOOTADD0
1	NSBOOTADD1	0x0BF8 8000 - Bootloader	NSBOOTADD1

1. BOOT0 is either not NBOOT0 when NSWBOOT0 = 0 or pin PH3-BOOT0 when NSWBOOT0 = 1

When TrustZone is enabled by setting the TZEN option bit, the boot space must be in the secure area. The SECBOOTADD0 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. In this case, all other boot options are ignored.

Table 23. Boot modes when TrustZone is enabled (TZEN = 1)

<table>
<tr><th rowspan="2">BOOT_LOCK</th><th rowspan="2">BOOT0(1)</th><th rowspan="2">RSS command</th><th colspan="4">Boot address option bytes selection</th><th rowspan="2">Boot initial VTOR_S</th></tr>
<tr><th>Secure</th><th>ST programmed default value</th><th>Nonsecure</th><th>ST programmed default value</th></tr>
<tr><td rowspan="4">0</td><td>0</td><td>0</td><td>SECBOOTADD0</td><td>0x0C00 0000 - User flash</td><td>NSBOOTADD0</td><td>0x0800 0000(2)</td><td>SECBOOTADD0</td></tr>
<tr><td>1</td><td>0</td><td>N/A</td><td>0x0FF8 0000 - RSS</td><td>NSBOOTADD1</td><td>0x0BF8 8000</td><td>0x0FF8 0000</td></tr>
<tr><td>0</td><td rowspan="2">≠ 0</td><td rowspan="2">N/A</td><td rowspan="2">0x0FF8 0000 - RSS</td><td>NSBOOTADD0</td><td>0x0800 0000(2)</td><td rowspan="2">0x0FF8 0000</td></tr>
<tr><td>1</td><td>NSBOOTADD1</td><td>0x0BF8 8000</td></tr>
</table>

a. STM32WBA52/4/5xx devices only.

Table 23. Boot modes when TrustZone is enabled (TZEN = 1) (continued)

BOOT_LOCK	BOOT0(1)	RSS command	Boot address option bytes selection: Secure	ST programmed default value	Nonsecure	ST programmed default value	Boot initial VTOR_S
1	0	x	SECBOOTADD0	0x0C00 0000 - User flash	NSBOOTADD0	0x0800 0000(2)	SECBOOTADD0
	1				NSBOOTADD1	0x0BF8 8000(3)	

1. BOOT0 is either not NBOOT0 when NSWBOOT0 = 0 or pin PH3-BOOT0 when NSWBOOT0 = 1
2. The default NSBOOTADD0 points to a secure flash memory area, at boot privileged software must write Cortex-M33 VTOR_NS to point to a nonsecure area.
3. The default NSBOOTADD1 points to the bootloader, at boot privileged software must write Cortex-M33 VTOR_NS to point to a nonsecure user area.

The boot address option bytes are used to program any boot memory address. However, the allowed address space depends on flash memory read protection RDP level.

If the programmed boot memory address is out of the allowed memory mapped area when TZEN = 0 and RDP level is 2, or when TZEN = 1 and RDP level is 0.5 or more, the default boot fetch address is forced either in the secure flash memory, or the nonsecure flash memory, depending on TrustZone security option, as described in *Table 24*.

Table 24. Boot space versus RDP protection

RDP	TZEN = 1(1): Valid address range SECBOOTADD	TZEN = 0: Valid address range NSBOOTADDx
0	Any boot address	Any boot address
0.5	Boot address only in RSS 0x0FF8 0000 or in secure flash memory: 0x0C00 0000- 0x0C0F FFFF. Otherwise boot address forced to RSS: 0x0FF8 0000.	N/A
1		Any boot address
2		Boot address only in flash memory: 0x0800 0000 - 0x080F FFFF. Otherwise boot address forced to 0x0800 0000.

1. The initial NSBOOTADDx can point to a secure area. At boot, privileged software must write Cortex-M33 VTOR_NS to point to a nonsecure user area.

The BOOT0 value (coming from the pin or from the user option) is latched upon reset release. It is up to the user to set BOOT0 or NBOOT0 values, to select the required boot mode.

The BOOT0 pin or user option NBOOT0 (depending on NSWBOOT0 in FLASH_OPTR) is also resampled when exiting standby modes. Consequently, the BOOT0 pin or user option must be kept in the required Boot mode configuration in standby modes. After the 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 user option), 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 AN2606 "*STM32 microcontroller system memory boot mode"*.

Embedded root security services (RSS)[a]

The embedded RSS are located in the secure information block, programmed by ST during production. Refer to the AN4992 "*Introduction to secure firmware install (SFI) for STM32 MCUs"*.

a. STM32WBA52/4/5xx devices only.

5 Global TrustZone® controller (GTZC)

Available only on STM32WBA52/54/55xx devices.

5.1 Introduction

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

- **TZSC**: TrustZone security controller

  This sub-block defines the secure/privileged state of slave peripherals. 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:

- independent 32-bit AHB interface for TZSC, TZIC, and MPCBBs
- TZIC accessible only with secure transactions
- Secure and nonsecure 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 access mode for securable peripherals

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 nonsecure 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/nonsecure and privileged/unprivileged transaction towards APB agents
- PPC (peripheral protection controller) used as secure gate to block or propagate secure/nonsecure 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)
- **Nonsecure 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 has Armv8-M core or DMA)

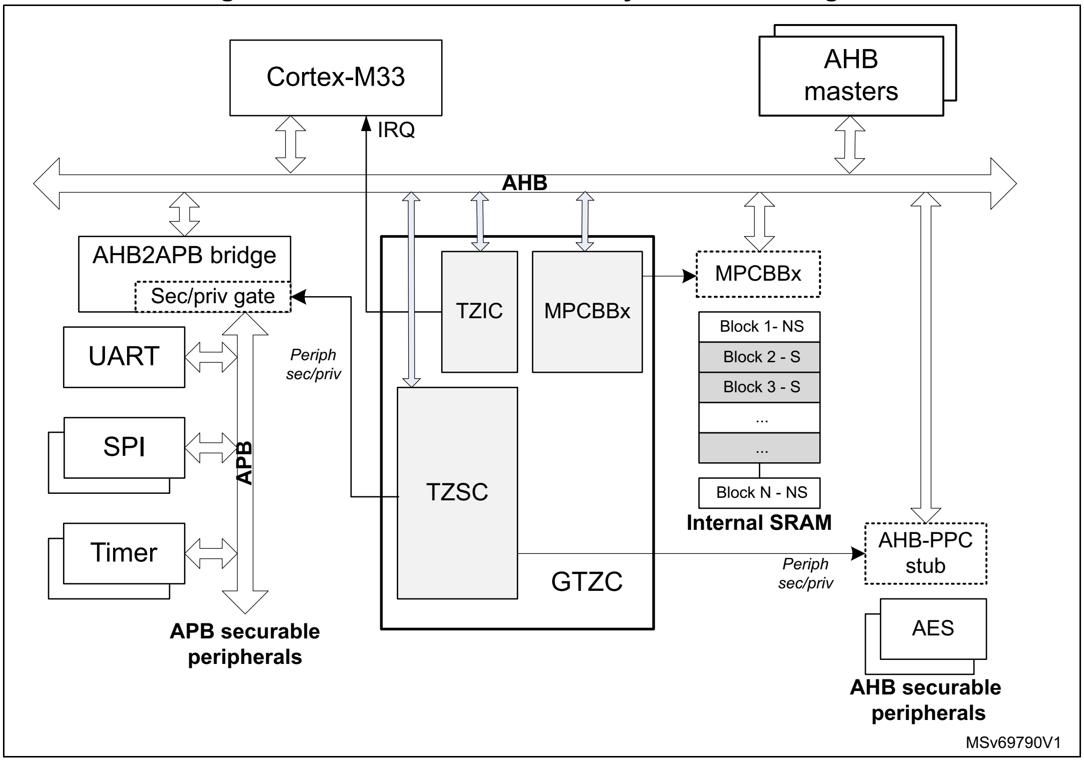
Application information

The TZSC, TZIC, and MPCBBs 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, MPCBB secure blocks can be changed by secure software executing from the secure FLASH region or secure SRAM. The applies to the GTZC1_TZSC_SECCFGRn/PRIVCFGRn registers that define secure/privileged state of each peripheral.

The Armv8-M security architecture with secure, securable and TrustZone-aware peripherals is shown in *Figure 13*.

Figure 13. GTZC in Armv8-M subsystem block diagram



5.3 GTZC implementation

The devices embed two instances of GTZC.

Table 25. GTZC features

GTZC sub-blocks	GTZC
TZSC	X
TZIC	X
MPCBB (number of MPCBB)	X (3)

The following table shows the GTZC sub-blocks address offset versus GTZC base address (refer to *Section 2.3* for GTZC base address).

Table 26. GTZC sub-blocks address offset

GTZC sub-blocks	Address offset
GTZC1_TZSC	0x0000
GTZC1_TZIC	0x0400
GTZC1_MPCBB1	0x0800
GTZC1_MPCBB2	0x0C00
GTZC1_MPCBB6	0x1C00

The following table describes the characteristics of the available MPCBB.

Table 27. MPCBB resource assignment

<table>
<tr><th>MPC</th><th>Resource</th><th>Memory size (Kbytes)</th><th>Block size (bytes)</th><th>Number of blocks</th><th>Number of super-blocks</th></tr>
<tr><td rowspan="2">MPCBB1</td><td rowspan="2">SRAM1</td><td>64 (STM32WBA5xxG)</td><td rowspan="4">512</td><td>128</td><td>4</td></tr>
<tr><td>32 (STM32WBA5xxE)</td><td>64</td><td>2</td></tr>
<tr><td>MPCBB2</td><td>SRAM2</td><td>64</td><td>128</td><td>4</td></tr>
<tr><td>MPCBB6</td><td>2.4 GHz RADIO RXTXRAM</td><td>16</td><td>32</td><td>1</td></tr>
</table>

5.4 GTZC functional description

5.4.1 GTZC block diagram

Figure 14 describes the combined feature of TZSC, TZIC, and MPCBBs. Each sub-block is controlled by its own AHB configuration port.

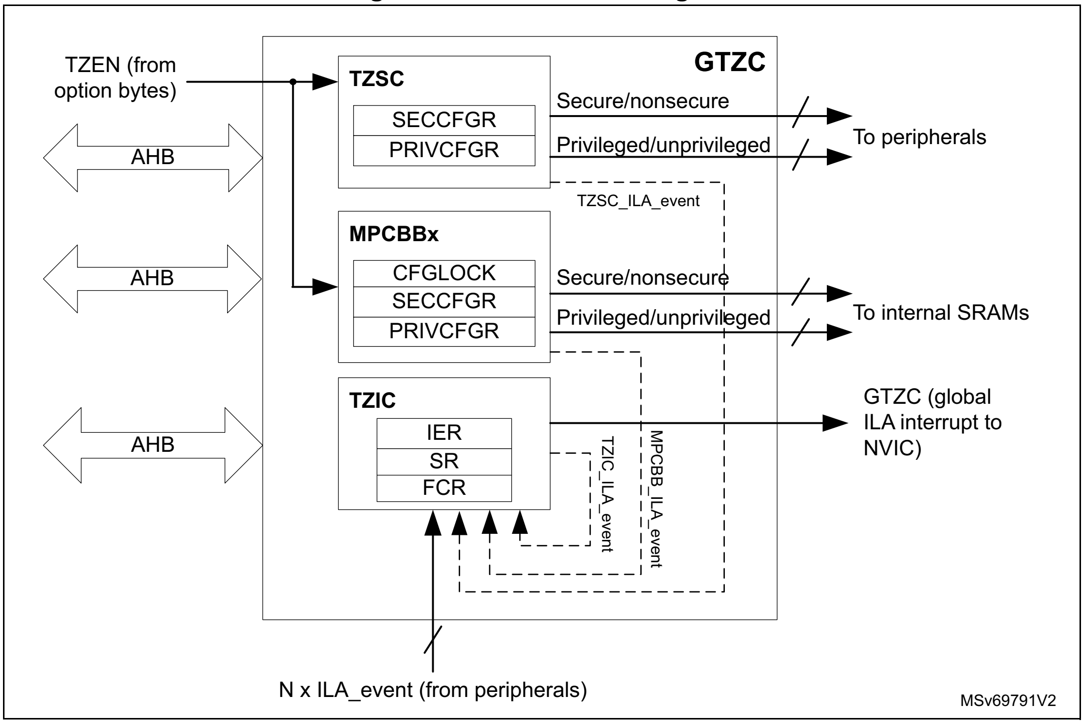
The TZSC defines which peripheral is secured and/or privileged. The privileged configuration bit of a peripheral can be modified by a secure privileged transaction when the peripheral is configured as secure. Otherwise, a privileged transaction (nonsecure) is sufficient.

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 14. GTZC block diagram



5.4.2 Illegal access definition

Three different types of illegal access (ILA) exist:

- Illegal nonsecure access

  Any nonsecure transaction trying to write a secure resource is considered as illegal: consequently, the addressed resource generates an illegal access interrupt for illegal write access and a bus error for illegal fetch access. Some exceptions exist on secure and privileged configuration registers: the latter ones authorize nonsecure read access to secure registers (see GTZC1_TZSC_SECCFGRn and GTZC1_TZSC_PRIVCFGRn).
- Illegal secure access

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

  Correct TZIC settings enable the capture of the associated event and then generate the GTZC1_IRQn interrupt to the NVIC. This applies to read, write, and execute accesses.

  Concerning the MPCBB controller, there is an option to ignore secure data read/write access on nonsecure SRAM blocks, by setting the SRWILADIS bit in the GTZC1_MPCBB_CR register. Secure read and write data transactions are then allowed on nonsecure SRAM blocks, while secure execution access remains not allowed.

  Any secure execute transaction trying to access a nonsecure peripheral register is considered as illegal and generates 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)

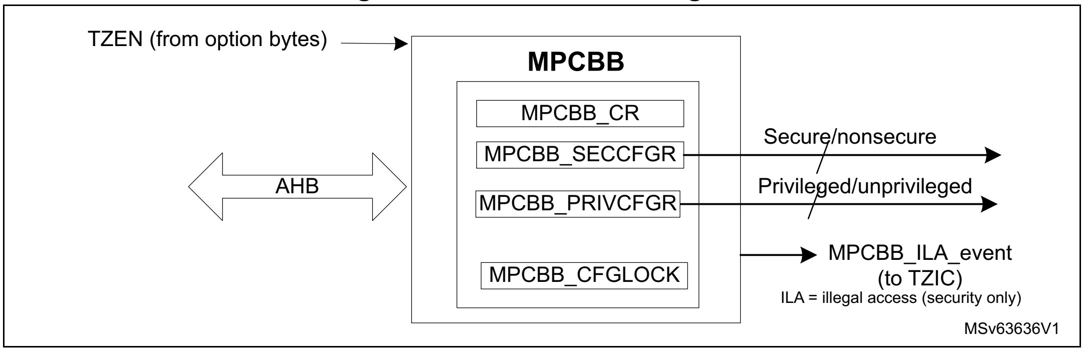
The TZSC is composed of a configurable set of registers, providing the control of secure and privileged state for all peripherals, done through:

- GTZC1_TZSC_SECCFGRn registers to control AHB/APB firewall stubs for the securable peripherals
- GTZC1_TZSC_PRIVCFGRn registers to control AHB/APB firewall stubs for the privileged peripherals

5.4.4 Memory protection controller - block based (MPCBB)

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

Figure 15. MPCBB block diagram



In order to set up 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 GTZC1_MPCBB_SECCFGRn.
- Privileged firmware must define which memory blocks are privileged by setting the correct bits in GTZC1_MPCBB_PRIVCFGRn.

An MPCBB super-block is made of 32 consecutive blocks. For each super-block, a secure application can lock all related secure/privileged bits using the correct bits in GTZC1_MPCBB_CFGLOCK. This lock remains active until the next system reset.

Note: The block size is 512 bytes. The super-block size is 512 x 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 triggered the secure illegal access interrupt. Register masks (GTZC1_TZIC_IERx) are available to filter unwanted event. On unmasked illegal event, TZIC generates the GTZC1_IRQn interrupt to the NVIC.

For each illegal event source, a status flag and a clear bit exist (respectively within GTZC1_TZIC_SRx and GTZC1_TZIC_FCRx). The reset value of mask registers (GTZC1_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 GTZC1_TZSC_SECCFGRn and GTZC1_TZSC_PRIVCFGRn, meaning that all securable peripherals are respectively set to nonsecure and unprivileged.

For internal SRAMs, all GTZC1_MPCBB_SECCFGRn and GTZC1_MPCBB_PRIVCFGRn 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 nonsecure and unprivileged 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, which systematically generates an illegal access event when accessed by a nonsecure access. The MPCBB and TZSC are TrustZone-aware peripherals, meaning that secure and nonsecure registers coexist within the peripheral.

Table 28. 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 GTZC1_TZIC_SRx	All bits in GTZC1_TZIC_IERx	Write 1 in the bit GTZC1_TZIC_FCRx	Yes	No	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_SECCFGRn and GTZC1_TZSC_PRIVCFGRn 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_SECCFGRn and GTZC1_TZSC_PRIVCFGRn registers not locked

1: Configuration of all GTZC1_TZSC_SECCFGRn and GTZC1_TZSC_PRIVCFGRn 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 the corresponding GTZC1_TZSC_PRIVCFGR1 register bit 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.	LPTIM2SEC	Res.
														rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	I2C1SEC	Res.	Res.	Res.	USART2SEC	Res.	IWDGSEC	WWDGSEC	Res.	Res.	Res.	Res.	TIM3SEC	TIM2SEC
		rw				rw		rw	rw					rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **LPTIM2SEC**: Secure access mode for LPTIM2
0: Nonsecure
1: Secure

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **I2C1SEC**: Secure access mode for I2C1
0: Nonsecure
1: Secure

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

Bit 9 **USART2SEC**: Secure access mode for USART2
0: Nonsecure
1: Secure

Bit 8 Reserved, must be kept at reset value.

Bit 7 **IWDGSEC**: Secure access mode for IWDG
0: Nonsecure
1: Secure

Bit 6 **WWDGSEC**: Secure access mode for WWDG
0: Nonsecure
1: Secure

Bits 5:2 Reserved, must be kept at reset value.

Bit 1 **TIM3SEC**: Secure access mode for TIM3
0: Nonsecure
1: Secure

Bit 0 **TIM2SEC**: Secure access mode for TIM2
0: Nonsecure
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 the corresponding GTZC1_TZSC_PRIVCFGR2 register bit 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.	ADC4SEC	COMPSEC	Res.	Res.	Res.	LPTIM1SEC	I2C3SEC	LPUART1SEC	SPI3SEC
							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.	SAI1SEC	TIM17SEC	TIM16SEC	Res.	USART1SEC	Res.	SPI1SEC	TIM1SEC
								rw	rw	rw		rw		rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **ADC4SEC**: Secure access mode for ADC4
0: Nonsecure
1: Secure

Bit 23 **COMPSEC**: Secure access mode for COMP
0: Nonsecure
1: Secure
Note that bit 23 is reserved on STM32WBA52xx devices.

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

Bit 19 **LPTIM1SEC**: Secure access mode for LPTIM1
0: Nonsecure
1: Secure

Bit 18 **I2C3SEC**: Secure access mode for I2C3
0: Nonsecure
1: Secure

Bit 17 **LPUART1SEC:** Secure access mode for LPUART1
0: Nonsecure
1: Secure

Bit 16 **SPI3SEC:** Secure access mode for SPI3
0: Nonsecure
1: Secure

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

Bit 7 **SAI1SEC:** Secure access mode for SAI1
0: Nonsecure
1: Secure
Note that bit 7 is reserved on STM32WBA52xx devices.

Bit 6 **TIM17SEC**: Secure access mode for TIM17
0: Nonsecure
1: Secure

Bit 5 **TIM16SEC**: Secure access mode for TIM16
0: Nonsecure
1: Secure

Bit 4 Reserved, must be kept at reset value.

Bit 3 **USART1SEC**: Secure access mode for USART1
0: Nonsecure
1: Secure

Bit 2 Reserved, must be kept at reset value.

Bit 1 **SPI1SEC**: Secure access mode for SPI1
0: Nonsecure
1: Secure

Bit 0 **TIM1SEC**: Secure access mode for TIM1
0: Nonsecure
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 the corresponding GTZC1_TZSC_PRIVCFGR3 register bit 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.	PTACONVSEC	RADIOSEC	RAMCFGSEC	Res.	Res.	Res.	Res.	Res.	PKASEC
							rw	rw	rw						rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SAESSEC	RNGSEC	HASHSEC	AESSEC	Res.	Res.	Res.	Res.	ICACHE_REGSEC	Res.	TSCSEC	CRCSEC	Res.	Res.	Res.
	rw	rw	rw	rw					rw		rw	rw			

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **PTACONVSEC**: Secure access mode for PTACONV
0: Nonsecure
1: Secure
Note that bit 24 is reserved on STM32WBA52xx devices.

Bit 23 **RADIOSEC**: Secure access mode for 2.4 GHz RADIO
0: Nonsecure
1: Secure

Bit 22 **RAMCFGSEC**: Secure access mode for RAMCFG
0: Nonsecure
1: Secure

Bits 21:17 Reserved, must be kept at reset value.

Bit 16 **PKASEC**: Secure access mode for PKA
0: Nonsecure
1: Secure

Bit 15 Reserved, must be kept at reset value.

Bit 14 **SAESSEC**: Secure access mode for SAES
0: Nonsecure
1: Secure

Bit 13 **RNGSEC**: Secure access mode for RNG
0: Nonsecure
1: Secure

Bit 12 **HASHSEC**: Secure access mode for HASH
0: Nonsecure
1: Secure

Bit 11 **AESSEC**: Secure access mode for AES
0: Nonsecure
1: Secure

Bits 10:7 Reserved, must be kept at reset value.

Bit 6 **ICACHE_REGSEC**: Secure access mode for ICACHE registers
0: Nonsecure
1: Secure

Bit 5 Reserved, must be kept at reset value.

Bit 4 **TSCSEC**: Secure access mode for TSC
0: Nonsecure
1: Secure

Bit 3 **CRCSEC**: Secure access mode for CRC
0: Nonsecure
1: Secure

Bits 2:0 Reserved, must be kept at reset value.

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 written only by secure privileged transaction when the corresponding GTZC1_TZSC_SECCFGR1 register bit is set to 1. If a given SEC bit is not set, the equivalent PRIV bit can be written by nonsecure 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.	LPTIM2PRIV	Res.
														rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	I2C1PRIV	Res.	Res.	Res.	USART2PRIV	Res.	IWDGPRIV	WWDGPRIV	Res.	Res.	Res.	Res.	TIM3PRIV	TIM2PRIV
		rw				rw		rw	rw					rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **LPTIM2PRIV**: Privileged access mode for LPTIM2
0: Unprivileged
1: Privileged

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **I2C1PRIV**: Privileged access mode for I2C1
0: Unprivileged
1: Privileged

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

Bit 9 **USART2PRIV**: Privileged access mode for USART2
0: Unprivileged
1: Privileged

Bit 8 Reserved, must be kept at reset value.

Bit 7 **IWDGPRIV**: Privileged access mode for IWDG
0: Unprivileged
1: Privileged

Bit 6 **WWDGPRIV**: Privileged access mode for WWDG
0: Unprivileged
1: Privileged

Bits 5:2 Reserved, must be kept at reset value.

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 written only by secure privileged transaction when the corresponding GTZC1_TZSC_SECCFGR2 register bit is set to 1. If a given SEC bit is not set, the equivalent PRIV bit can be written by nonsecure 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.	ADC4PRIV	COMPPRIV	Res.	Res.	Res.	LPTIM1PRIV	I2C3PRIV	LPUART1PRIV	SPI3PRIV
							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.	SAI1PRIV	TIM17PRIV	TIM16PRIV	Res.	USART1PRIV	Res.	SPI1PRIV	TIM1PRIV
								rw	rw	rw		rw		rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **ADC4PRIV**: Privileged access mode for ADC4
0: Unprivileged
1: Privileged

Bit 23 **COMPPRIV**: Privileged access mode for COMP
0: Unprivileged
1: Privileged
Note that bit 23 is reserved on STM32WBA52xx devices.

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

Bit 19 **LPTIM1PRIV**: Privileged access mode for LPTIM1
0: Unprivileged
1: Privileged

Bit 18 **I2C3PRIV**: Privileged access mode for I2C3
0: Unprivileged
1: Privileged

Bit 17 **LPUART1PRIV**: Privileged access mode for LPUART1
0: Unprivileged
1: Privileged

Bit 16 **SPI3PRIV**: Privileged access mode for SPI3
0: Unprivileged
1: Privileged

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

Bit 7 **SAI1PRIV**: Privileged access mode for SAI1
0: Unprivileged
1: Privileged
Note that bit 7 is reserved on STM32WBA52xx devices.

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 Reserved, must be kept at reset value.

Bit 3 **USART1PRIV**: Privileged access mode for USART1
0: Unprivileged
1: Privileged

Bit 2 Reserved, must be kept at reset value.

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 written only by secure privileged transaction when the corresponding GTZC1_TZSC_SECCFGR3 register bit is set to 1. If a given SEC bit is not set, the equivalent PRIV bit can be written by nonsecure 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.	PTACONVPRIV	RADIOPRIV	RAMCFGPRIV	Res.	Res.	Res.	Res.	Res.	PKAPRIV
							rw	rw	rw						rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SAESPRIV	RNGPRIV	HASHPRIV	AESPRIV	Res.	Res.	Res.	Res.	ICACHE_REGPRIV	Res.	TSCPRIV	CRCPRIV	Res.	Res.	Res.
	rw	rw	rw	rw					rw		rw	rw			

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **PTACONVPRIV**: Privileged access mode for PTACONV
0: Unprivileged
1: Privileged
Note that bit 24 is reserved on STM32WBA52xx devices.

Bit 23 **RADIOPRIV**: Privileged access mode for 2.4 GHz RADIO
0: Unprivileged
1: Privileged

Bit 22 **RAMCFGPRIV**: Privileged access mode for RAMCFG
0: Unprivileged
1: Privileged

Bits 21:17 Reserved, must be kept at reset value.

Bit 16 **PKAPRIV**: Privileged access mode for PKA
0: Unprivileged
1: Privileged

Bit 15 Reserved, must be kept at reset value.

Bit 14 **SAESPRIV**: Privileged access mode for SAES
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

Bits 10:7 Reserved, must be kept at reset value.

Bit 6 **ICACHE_REGPRIV**: Privileged access mode for ICACHE registers
0: Unprivileged
1: Privileged

Bit 5 Reserved, must be kept at reset value.

Bit 4 **TSCPRIV**: Privileged access mode for TSC
0: Unprivileged
1: Privileged

Bit 3 **CRCPRIV**: Privileged access mode for CRC
0: Unprivileged
1: Privileged

Bits 2:0 Reserved, must be kept at reset value.

5.6.8 GTZC1 TZSC register map

Table 29. GTZC1 TZSC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	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 to 0x00C	Reserved	Reserved																															
0x010	GTZC1_TZSC_ SECCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2SEC	Res.	Res.	Res.	I2C1SEC	Res.	Res.	Res.	USART2SEC	Res.	IWDGSEC	WWDGSEC	Res.	Res.	Res.	Res.	TIM3SEC	TIM2SEC
	Reset value															0				0				0		0	0					0	0
0x014	GTZC1_TZSC_ SECCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC4SEC	COMPSEC(1)	Res.	Res.	Res.	LPTIM1SEC	I2C3SEC	LPUART1SEC	SPI3SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI1SEC(1)	TIM17SEC	TIM16SEC	Res.	USART1SEC	Res.	SPI1SEC	TIM1SEC
	Reset value								0	0				0	0	0	0									0	0	0		0		0	0
0x018	GTZC1_TZSC_ SECCFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTACONVSEC(1)	RADIOSEC	RAMCFGSEC	Res.	Res.	Res.	Res.	Res.	PKASEC	Res.	SAESSEC	RNGSEC	HASHSEC	AESSEC	Res.	Res.	Res.	Res.	ICACHE_REGSEC	Res.	TSCSEC	CRCSEC	Res.	Res.	Res.
	Reset value								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.	Res.	Res.	LPTIM2PRIV	Res.	Res.	Res.	I2C1PRIV	Res.	Res.	Res.	USART2PRIV	Res.	IWDGPRIV	WWDGPRIV	Res.	Res.	Res.	Res.	TIM3PRIV	TIM2PRIV
	Reset value															0				0				0		0	0					0	0
0x024	GTZC1_TZSC_ PRIVCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC4PRIV	COMPPRIV(1)	Res.	Res.	Res.	LPTIM1PRIV	I2C3PRIV	LPUART1PRIV	SPI3PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI1PRIV(1)	TIM17PRIV	TIM16PRIV	Res.	USART1PRIV	Res.	SPI1PRIV	TIM1PRIV
	Reset value								0	0				0	0	0	0									0	0	0		0		0	0
0x028	GTZC1_TZSC_ PRIVFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTACONVPRIV(1)	RADIOPRIV	RAMCFGPRIV	Res.	Res.	Res.	Res.	Res.	PKAPRIV	Res.	SAESPRIV	RNGPRIV	HASHPRIV	AESPRIV	Res.	Res.	Res.	Res.	ICACHE_REGPRIV	Res.	TSCPRIV	CRCPRIV	Res.	Res.	Res.
	Reset value								0	0	0						0		0	0	0	0					0		0	0			

1. Bit(s) reserved on STM32WBA52xx devices.

Refer to *Table 26: GTZC 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.	Res.	Res.	LPTIM2IE	Res.
														rw	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	I2C1IE	Res.	Res.	Res.	USART2IE	Res.	IWDGIE	WWDGIE	Res.	Res.	Res.	Res.	TIM3IE	TIM2IE
		rw				rw		rw	rw					rw	rw

Bits 31:18 Reserved, must be kept at reset value.

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

Bits 16:14 Reserved, must be kept at reset value.

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

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

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

Bit 8 Reserved, must be kept at reset value.

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

Bits 5:2 Reserved, must be kept at reset value.

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.	ADC4IE	COMPIE	Res.	Res.	Res.	LPTIM1IE	I2C3IE	LPUART1IE	SPI3IE
							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.	SAI1IE	TIM17IE	TIM16IE	Res.	USART1IE	Res.	SPI1IE	TIM1IE
								rw	rw	rw		rw		rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **ADC4IE**: Illegal access interrupt enable for ADC4
0: Interrupt disabled
1: Interrupt enabled

Bit 23 **COMPIE**: Illegal access interrupt enable for COMP
0: Interrupt disabled
1: Interrupt enabled
Note that bit 23 is reserved on STM32WBA52xx devices.

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

Bit 19 **LPTIM1IE**: Illegal access interrupt enable for LPTIM1
0: Interrupt disabled
1: Interrupt enabled

Bit 18 **I2C3IE**: Illegal access interrupt enable for I2C3
0: Interrupt disabled
1: Interrupt enabled

Bit 17 **LPUART1IE**: Illegal access interrupt enable for LPUART1
0: Interrupt disabled
1: Interrupt enabled

Bit 16 **SPI3IE**: Illegal access interrupt enable for SPI3
0: Interrupt disabled
1: Interrupt enabled

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

Bit 7 **SAI1IE**: Illegal access interrupt enable for SAI1
0: Interrupt disabled
1: Interrupt enabled
Note that bit 7 is reserved on STM32WBA52xx devices.

Bit 6 **TIM17IE**: Illegal access interrupt enable for TIM17
0: Interrupt disabled
1: Interrupt enabled

Bit 5 **TIM16IE**: Illegal access interrupt enable for TIM16
0: Interrupt disabled
1: Interrupt enabled

Bit 4 Reserved, must be kept at reset value.

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

Bit 2 Reserved, must be kept at reset value.

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.	PTACONVIE	RADIOIE	RAMCFGIE	Res.	Res.	Res.	Res.	Res.	PKAIE
							rw	rw	rw						rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HSEMIE	SAESIE	RNGIE	HASHIE	AESIE	Res.	Res.	Res.	Res.	ICACHEIE	Res.	TSCIE	CRCIE	Res.	Res.	Res.
rw	rw	rw	rw	rw					rw		rw	rw			

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **PTACONVIE**: Illegal access interrupt enable for PTACONV
0: Interrupt disabled
1: Interrupt enabled
Note that bit 24 is reserved on STM32WBA52xx devices.

Bit 23 **RADIOIE**: Illegal access interrupt enable for 2.4 GHz RADIO
0: Interrupt disabled
1: Interrupt enabled

Bit 22 **RAMCFGIE**: Illegal access interrupt enable for RAMCFG
0: Interrupt disabled
1: Interrupt enabled

Bits 21:17 Reserved, must be kept at reset value.

Bit 16 **PKAIE**: Illegal access interrupt enable for PKA
0: Interrupt disabled
1: Interrupt enabled

Bit 15 **HSEMIE**: Illegal access interrupt enable for HSEM
0: Interrupt disabled
1: Interrupt enabled

Bit 14 **SAESIE**: Illegal access interrupt enable for SAES
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

Bits 10:7 Reserved, must be kept at reset value.

Bit 6 **ICACHEIE**: Illegal access interrupt enable for ICACHE registers
0: Interrupt disabled
1: Interrupt enabled

Bit 5 Reserved, must be kept at reset value.

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

Bits 2:0 Reserved, must be kept at reset value.

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
MPCBB6IE	SRAM6IE	Res.	Res.	Res.	Res.	MPCBB2IE	SRAM2IE	MPCBB1IE	SRAM1IE	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw					rw	rw	rw	rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TZICIE	TZSCIE	EXTIIE	Res.	RCCIE	PWRIE	TAMPIE	RTCIE	SYSCFGIE	Res.	Res.	Res.	Res.	FLASH_REGIE	FLASHIE	GPDMA1IE
rw	rw	rw		rw	rw	rw	rw	rw					rw	rw	rw

Bit 31 **MPCBB6IE**: Illegal access interrupt enable for MPCBB6
0: Interrupt disabled
1: Interrupt enabled

Bit 30 **SRAM6IE**: Illegal access interrupt enable for 2.4 GHz RXTXRAM memory
0: Interrupt disabled
1: Interrupt enabled

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **MPCBB2IE**: Illegal access interrupt enable for MPCBB2
0: Interrupt disabled
1: Interrupt enabled

Bit 24 **SRAM2IE**: Illegal access interrupt enable for SRAM2 memory
0: Interrupt disabled
1: Interrupt enabled

Bit 23 **MPCBB1IE**: Illegal access interrupt enable for MPCBB1
0: Interrupt disabled
1: Interrupt enabled

Bit 22 **SRAM1IE**: Illegal access interrupt enable for SRAM1
0: Interrupt disabled
1: Interrupt enabled

Bits 21:16 Reserved, must be kept at reset value.

Bit 15 **TZICIE**: Illegal access interrupt enable for GTZC1 TZIC
0: Interrupt disabled
1: Interrupt enabled

Bit 14 **TZSCIE**: Illegal access interrupt enable for GTZC1 TZSC
0: Interrupt disabled
1: Interrupt enabled

Bit 13 **EXTIIE**: Illegal access interrupt enable for EXTI
0: Interrupt disabled
1: Interrupt enabled

Bit 12 Reserved, must be kept at reset value.

Bit 11 **RCCIE**: Illegal access interrupt enable for RCC
0: Interrupt disabled
1: Interrupt enabled

Bit 10 **PWRIE**: Illegal access interrupt enable for PWR
0: Interrupt disabled
1: Interrupt enabled

Bit 9 **TAMPIE**: Illegal access interrupt enable for TAMP
0: Interrupt disabled
1: Interrupt enabled

Bit 8 **RTCIE**: Illegal access interrupt enable for RTC
0: Interrupt disabled
1: Interrupt enabled

Bit 7 **SYSCFGIE**: Illegal access interrupt enable for SYSCFG
0: Interrupt disabled
1: Interrupt enabled

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **FLASH_REGIE**: Illegal access interrupt enable for FLASH interface
0: Interrupt disabled
1: Interrupt enabled

Bit 1 **FLASHIE**: Illegal access interrupt enable for FLASH memory
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.	Res.	Res.	LPTIM2F	Res.
														r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	I2C1F	Res.	Res.	Res.	USART2F	Res.	IWDGF	WWDGF	Res.	Res.	Res.	Res.	TIM3F	TIM2F
		r				r		r	r					r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **LPTIM2F**: Illegal access flag for LPTIM2
0: No illegal access event
1: Illegal access event

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **I2C1F**: Illegal access flag for I2C1
0: No illegal access event
1: Illegal access event

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

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

Bit 8 Reserved, must be kept at reset value.

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

Bits 5:2 Reserved, must be kept at reset value.

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.	ADC4F	COMPF	Res.	Res.	Res.	LPTIM1F	I2C3F	LPUART1F	SPI3F
							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.	SAI1F	TIM17F	TIM16F	Res.	USART1F	Res.	SPI1F	TIM1F
								r	r	r		r		r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **ADC4F**: Illegal access flag for ADC4
0: No illegal access event
1: Illegal access event

Bit 23 **COMPF**: Illegal access flag for COMP
0: No illegal access event
1: Illegal access event
Note that bit 23 is reserved on STM32WBA52xx devices.

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

Bit 19 **LPTIM1F**: Illegal access flag for LPTIM1
0: No illegal access event
1: Illegal access event

Bit 18 **I2C3F**: Illegal access flag for I2C3
0: No illegal access event
1: Illegal access event

Bit 17 **LPUART1F**: Illegal access flag for LPUART1
0: No illegal access event
1: Illegal access event

Bit 16 **SPI3F**: Illegal access flag for SPI3
0: No illegal access event
1: Illegal access event

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

Bit 7 **SAI1F**: Illegal access flag for SAI1
0: No illegal access event
1: Illegal access event
Note that bit 7 is reserved on STM32WBA52xx devices.

Bit 6 **TIM17F**: Illegal access flag for TIM17
0: No illegal access event
1: Illegal access event

Bit 5 **TIM16F**: Illegal access flag for TIM16
0: No illegal access event
1: Illegal access event

Bit 4 Reserved, must be kept at reset value.

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

Bit 2 Reserved, must be kept at reset value.

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.	PTACONVF	RADIOF	RAMCFGF	Res.	Res.	Res.	Res.	Res.	PKAF
							r	r	r						r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSEMF	SAESF	RNGF	HASHF	AESF	Res.	Res.	Res.	Res.	ICACHEF	Res.	TSCF	CRCF	Res.	Res.	Res.
r	r	r	r	r					r		r	r			

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **PTACONVF**: Illegal access flag for PTACONV
0: No illegal access event
1: Illegal access event
Note that bit 24 is reserved on STM32WBA52xx devices.

Bit 23 **RADIOF**: Illegal access flag for 2.4 GHz RADIO
0: No illegal access event
1: Illegal access event

Bit 22 **RAMCFGF**: Illegal access flag for RAMCFG
0: No illegal access event
1: Illegal access event

Bits 21:17 Reserved, must be kept at reset value.

Bit 16 **PKAF**: Illegal access flag for PKA
0: No illegal access event
1: Illegal access event

Bit 15 **HSEMF**: Illegal access flag for HSEM
0: No illegal access event
1: Illegal access event

Bit 14 **SAESF**: Illegal access flag for SAES
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

Bits 10:7 Reserved, must be kept at reset value.

Bit 6 **ICACHEF**: Illegal access flag for ICACHE registers
0: No illegal access event
1: Illegal access event

Bit 5 Reserved, must be kept at reset value.

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

Bits 2:0 Reserved, must be kept at reset value.

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
MPCBB6F	SRAM6F	Res.	Res.	Res.	Res.	MPCBB2F	SRAM2F	MPCBB1F	SRAM1F	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
TZICF	TZSCF	EXTIF	Res.	RCCF	PWRF	TAMPF	RTCF	SYSCFGF	Res.	Res.	Res.	Res.	FLASH_REGF	FLASHF	GPDMA1F
r	r	r		r	r	r	r	r					r	r	r

Bit 31 **MPCBB6F**: Illegal access flag for MPCBB6
0: No illegal access event
1: Illegal access event

Bit 30 **SRAM6F**: Illegal access flag for 2.4 GHZ RADIO RXTXRAM memory
0: No illegal access event
1: Illegal access event

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **MPCBB2F**: Illegal access flag for MPCBB2
0: No illegal access event
1: Illegal access event

Bit 24 **SRAM2F**: Illegal access flag for SRAM2 memory
0: No illegal access event
1: Illegal access event

Bit 23 **MPCBB1F**: Illegal access flag for MPCBB1
0: No illegal access event
1: Illegal access event

Bit 22 **SRAM1F**: Illegal access flag for SRAM1
0: No illegal access event
1: Illegal access event

Bits 21:16 Reserved, must be kept at reset value.

Bit 15 **TZICF**: Illegal access flag for GTZC1 TZIC
0: No illegal access event
1: Illegal access event

Bit 14 **TZSCF**: Illegal access flag for GTZC1 TZSC
0: No illegal access event
1: Illegal access event

Bit 13 **EXTIF**: Illegal access flag for EXTI
0: No illegal access event
1: Illegal access event

Bit 12 Reserved, must be kept at reset value.

Bit 11 **RCCF**: Illegal access flag for RCC
0: No illegal access event
1: Illegal access event

Bit 10 **PWRF**: Illegal access flag for PWR
0: No illegal access event
1: Illegal access event

Bit 9 **TAMPF**: Illegal access flag for TAMP
0: No illegal access event
1: Illegal access event

Bit 8 **RTCF**: Illegal access flag for RTC
0: No illegal access event
1: Illegal access event

Bit 7 **SYSCFGF**: Illegal access flag for SYSCFG
0: No illegal access event
1: Illegal access event

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **FLASH_REGF**: Illegal access flag for FLASH interface
0: No illegal access event
1: Illegal access event

Bit 1 **FLASHF**: Illegal access flag for FLASH memory
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.	Res.	Res.	CLPTIM2F	Res.
														w	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	CI2C1F	Res.	Res.	Res.	CUSART2F	Res.	CIWDGF	CWWDGF	Res.	Res.	Res.	Res.	CTIM3F	CTIM2F
		w				w		w	w					w	w

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **CLPTIM2F**: Clear the illegal access flag for LPTIM2
0: No action
1: Status flag cleared

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **CI2C1F**: Clear the illegal access flag for I2C1
0: No action
1: Status flag cleared

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

Bit 9 **CUSART2F**: Clear the illegal access flag for USART2
0: No action
1: Status flag cleared

Bit 8 Reserved, must be kept at reset value.

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

Bits 5:2 Reserved, must be kept at reset value.

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.	CADC4F	CCOMPF	Res.	Res.	Res.	CLPTIM1F	CI2C3F	CLPUART1F	CSPI3F
							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.	CSAI1F	CTIM17F	CTIM16F	Res.	CUSART1F	Res.	CSPI1F	CTIM1F
								w	w	w		w		w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **CADC4F**: Clear the illegal access flag for ADC4
0: No action
1: Status flag cleared

Bit 23 **CCOMPF**: Clear the illegal access flag for COMP
0: No action
1: Status flag cleared
Note that bit 23 is reserved on STM32WBA52xx devices.

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

Bit 19 **CLPTIM1F**: Clear the illegal access flag for LPTIM1
0: No action
1: Status flag cleared

Bit 18 **CI2C3F**: Clear the illegal access flag for I2C3
0: No action
1: Status flag cleared

Bit 17 **CLPUART1F**: Clear the illegal access flag for LPUART1
0: No action
1: Status flag cleared

Bit 16 **CSPI3F**: Clear the illegal access flag for SPI3
0: No action
1: Status flag cleared

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

Bit 7 **CSAI1F**: Clear the illegal access flag for SAI1
0: No action
1: Status flag cleared
Note that bit 7 is reserved on STM32WBA52xx devices.

Bit 6 **CTIM17F**: Clear the illegal access flag for TIM17
0: No action
1: Status flag cleared

Bit 5 **CTIM16F**: Clear the illegal access flag for TIM16
0: No action
1: Status flag cleared

Bit 4 Reserved, must be kept at reset value.

Bit 3 **CUSART1F**: Clear the illegal access flag for USART1
0: No action
1: Status flag cleared

Bit 2 Reserved, must be kept at reset value.

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 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.	CPTACONVF	CRADIOF	CRAMCFGF	Res.	Res.	Res.	Res.	Res.	CPKAF
							w	w	w						w
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CHSEMF	CSAESF	CRNGF	CHASHF	CAESF	Res.	Res.	Res.	Res.	CICACHEF	Res.	CTSCF	CCRCF	Res.	Res.	Res.
w	w	w	w	w					w		w	w			

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **CPTACONVF**: Clear the illegal access flag for PTACONV
0: No action
1: Status flag cleared
Note that bit 24 is reserved on STM32WBA52xx devices.

Bit 23 **CRADIOF**: Clear the illegal access flag for 2.4 GHz RADIO
0: No action
1: Status flag cleared

Bit 22 **CRAMCFGF**: Clear the illegal access flag for RAMCFG
0: No action
1: Status flag cleared

Bits 21:17 Reserved, must be kept at reset value.

Bit 16 **CPKAF**: Clear the illegal access flag for PKA
0: No action
1: Status flag cleared

Bit 15 **CHSEMF**: Clear the illegal access flag for HSEM
0: No action
1: Status flag cleared

Bit 14 **CSAESF**: Clear the illegal access flag for SAES
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

Bits 10:7 Reserved, must be kept at reset value.

Bit 6 **CICACHEF**: Clear the illegal access flag for ICACHE registers
0: No action
1: Status flag cleared

Bit 5 Reserved, must be kept at reset value.

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

Bits 2:0 Reserved, must be kept at reset value.

5.7.12 GTZC1 TZIC flag clear register 4 (GTZC1_TZIC_FCR4)

Address offset: 0x02C

Reset value: 0x0000 0000

Secure privileged access only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMPCBB6F	CSRAM6F	Res.	Res.	Res.	Res.	CMPCBB2F	CSRAM2F	CMPCBB1F	CSRAM1F	Res.	Res.	Res.	Res.	Res.	Res.
w	w					w	w	w	w						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTZICF	CTZSCF	CEXTIF	Res.	CRCCF	CPWRF	CTAMPF	CRTCF	CSYSCFGF	Res.	Res.	Res.	Res.	CFLASH_REGF	CFLASHF	CGPDMA1F
w	w	w		w	w	w	w	w					w	w	w

Bit 31 **CMPCBB6F**: Clear the illegal access flag for MPCBB6
0: No action
1: Status flag cleared

Bit 30 **CSRAM6F**: Clear the illegal access flag for 2.4 GHz RADIO RXTXRAM
0: No action
1: Status flag cleared

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **CMPCBB2F**: Clear the illegal access flag for MPCBB2
0: No action
1: Status flag cleared

Bit 24 **CSRAM2F**: Clear the illegal access flag for SRAM2
0: No action
1: Status flag cleared

Bit 23 **CMPCBB1F**: Clear the illegal access flag for MPCBB1
0: No action
1: Status flag cleared

Bit 22 **CSRAM1F**: Clear the illegal access flag for SRAM1
0: No action
1: Status flag cleared

Bits 21:16 Reserved, must be kept at reset value.

Bit 15 **CTZICF**: Clear the illegal access flag for GTZC1 TZIC
0: No action
1: Status flag cleared

Bit 14 **CTZSCF**: Clear the illegal access flag for GTZC1 TZSC
0: No action
1: Status flag cleared

Bit 13 **CEXTIF**: Clear the illegal access flag for EXTI
0: No action
1: Status flag cleared

Bit 12 Reserved, must be kept at reset value.

Bit 11 **CRCCF**: Clear the illegal access flag for RCC
0: No action
1: Status flag cleared

Bit 10 **CPWRF**: Clear the illegal access flag for PWR
0: No action
1: Status flag cleared

Bit 9 **CTAMPF**: Clear the illegal access flag for TAMP
0: No action
1: Status flag cleared

Bit 8 **CRTCF**: Clear the illegal access flag for RTC
0: No action
1: Status flag cleared

Bit 7 **CSYSCFGF**: Clear the illegal access flag for SYSCFG
0: No action
1: Status flag cleared

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **CFLASH_REGF**: Clear the illegal access flag for FLASH interface
0: No action
1: Status flag cleared

Bit 1 **CFLASHF**: Clear the illegal access flag for FLASH memory
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 30. GTZC1 TZIC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	GTZC1_TZIC_IER1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2IE	Res.	Res.	Res.	I2C1IE	Res.	Res.	Res.	USART2IE	Res.	IWDGIE	WWDGIE	Res.	Res.	Res.	Res.	TIM3IE	TIM2IE
	Reset value															0				0				0		0	0					0	0
0x004	GTZC1_TZIC_IER2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC4IE	COMPIE[1]	Res.	Res.	Res.	LPTIM1IE	I2C3IE	LPUART1IE	SPI3IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI1IE[1]	TIM17IE	TIM16IE	Res.	USART1IE	Res.	SPI1IE	TIM1IE
	Reset value								0	0				0	0	0	0									0	0	0		0		0	0
0x008	GTZC1_TZIC_IER3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTACONVFI[1]	RADIOIE	RAMCFGIE	Res.	Res.	Res.	Res.	Res.	PKAIE	HSEMIE	SAESIE	RNGIE	HASHIE	AESIE	Res.	Res.	Res.	Res.	ICACHE_REGIE	Res.	TSCIE	CRCIE	Res.	Res.	Res.
	Reset value								0	0	0						0	0	0	0	0	0					0		0	0			
0x00C	GTZC1_TZIC_IER4	MPCBB6IE	SRAM6IE	Res.	Res.	Res.	Res.	MPCBB2IE	SRAM2IE	MPCBB1IE	SRAM1IE	Res.	Res.	Res.	Res.	Res.	Res.	TZICIE	TZSCIE	EXTIIE	Res.	RCCIE	PWRIE	TAMPIE	RTCIE	SYSCFGIE	Res.	Res.	Res.	Res.	FLASH_REGIE	FLASHIE	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.	Res.	Res.	LPTIM2F	Res.	Res.	Res.	I2C1F	Res.	Res.	Res.	USART2F	Res.	IWDGF	WWDGF	Res.	Res.	Res.	Res.	TIM3F	TIM2F
	Reset value															0				0				0		0	0					0	0
0x014	GTZC1_TZIC_SR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC4F	COMPF[1]	Res.	Res.	Res.	LPTIM1F	I2C3F	LPUART1F	SPI3F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI1F[1]	TIM17F	TIM16F	Res.	USART1F	Res.	SPI1F	TIM1F
	Reset value								0	0				0	0	0	0									0	0	0		0		0	0
0x018	GTZC1_TZIC_SR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTACONVF[1]	RADIOF	RAMCFGF	Res.	Res.	Res.	Res.	Res.	PKAF	HSEMF	SAESF	RNGF	HASHF	AESF	Res.	Res.	Res.	Res.	ICACHEF	Res.	TSCF	CRCF	Res.	Res.	Res.
	Reset value								0	0	0						0	0	0	0	0	0					0		0	0			
0x01C	GTZC1_TZIC_SR4	MPCBB6F	SRAM6F	Res.	Res.	Res.	Res.	MPCBB2F	SRAM2F	MPCBB1F	SRAM1F	Res.	Res.	Res.	Res.	Res.	Res.	TZICF	TZSCF	EXTIF	Res.	RCCF	PWRF	TAMPF	RTCF	SYSCFGF	Res.	Res.	Res.	Res.	FLASH_REGF	FLASHF	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.	Res.	Res.	CLPTIM2F	Res.	Res.	Res.	CI2C1F	Res.	Res.	Res.	CUSART2F	Res.	CIWDGF	CWWDGF	Res.	Res.	Res.	Res.	CTIM3F	CTIM2F
	Reset value															0				0				0		0	0					0	0
0x024	GTZC1_TZIC_FCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CADC4F	CCOMPF[1]	Res.	Res.	Res.	CLPTIM1F	CI2C3F	CLPUART1F	CSPI3F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSAI1F[1]	CTIM17F	CTIM16F	Res.	CUSART1F	Res.	CSPI1F	CTIM1F
	Reset value								0	0				0	0	0	0									0	0	0		0		0	0

Table 30. GTZC1 TZIC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x028	GTZC1_TZIC_ FCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTACONVF(1)	CRADIOF	CRAMCFGF	Res.	Res.	Res.	Res.	Res.	CPKAF	CHSEMF	CSAESF	CRNGF	CHASHF	CAESF	Res.	Res.	Res.	Res.	CICACHEF	Res.	CTSCF	CCRCF	Res.	Res.	Res.
	Reset value								0	0	0						0	0	0	0	0	0					0		0	0			
0x02C	GTZC1_TZIC_ FCR4	CMPCBB6F	CSRAM6F	Res.	Res.	Res.	Res.	CMPCBB2F	CSRAM2F	CMPCBB1F	CSRAM1F	Res.	Res.	Res.	Res.	Res.	Res.	CTZICF	CTZSCF	CEXTIF	Res.	CRCCF	CPWRF	CTAMPF	CRTCF	CSYSCFGF	Res.	Res.	Res.	Res.	CFLASH_REGF	CFLASHF	CGPDMA1F
	Reset value	0	0					0	0	0	0							0	0	0		0	0	0	0	0					0	0	0

1. Bit reserved on STM32WBA52xx devices.

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

5.8 GTZC1 MPCBB registers

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

5.8.1 GTZC1 MPCBB control register (GTZC1_MPCBB_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 nonsecure blocks of the block-based SRAM (secure fetch on nonsecure block is always considered illegal).

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

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

Bit 30 **INVSECSTATE**: SRAM clocks security state

This bit is used to define the internal SRAM clocks control in RCC as secure or not.

0: SRAM clock is secured if a secure area exists in the MPCBB. It is non secure if there is no secure area.

1: SRAM clock is nonsecure 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 MPCBB configuration lock register (GTZC1_MPCBB_CFGLOCK)

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.	SPLCK3	SPLCK2	SPLCK1	SPLCK0
												rs	rs	rs	rs

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **SPLCK[3:0]**: Security/privilege configuration lock super-block (n = 0 to 3)

This bit is set by software and can be cleared only by system reset.

0: GTZC1_MPCBB_SECCFGRn and GTZC1_MPCBB_PRIVCFGRn can be written.

1: Writes to GTZC1_MPCBB_SECCFGRn and GTZC1_MPCBB_PRIVCFGRn are ignored

Note that bits [3:2] are reserved on STM32WBA5xxE devices for MPCBB1.

5.8.3 GTZC1 MPCBB security configuration for super-block n register (GTZC1_MPCBB_SECCFGRn)

Address offset: 0x100 + 0x04 * n, (n = 0 to 3)

Reset value: 0xFFFF FFFF

MPCBB1 registers (n = 2 to 3) are reserved on STM32WBA5xxE devices.

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 **SEC[31:0]**: Security configuration for block y (y = 0 to 31) in super block n
0: Nonsecure access only to block y, belonging to super-block n. Secure access is also allowed if the SRWILADIS bit is set in GTZC1_MPCBB_CR.
1: Secure access only to block y, belonging to super-block n.
Unprivileged write to this bit is ignored if PRIVy bit is set in GTZC1_MPCBB_PRIVCFGRn.
Writes are ignored if SPLCKn bit is set in GTZC1_MPCBB_CFGLOCK.

5.8.4 GTZC1 MPCBB privileged configuration for super-block n register (GTZC1_MPCBB_PRIVCFGRn)

Address offset: 0x200 + 0x04 * n, (n = 0 to 3)

Reset value: 0xFFFF FFFF

MPCBB1 registers (n = 2 to 3) are reserved on STM32WBA5xxE devices.

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 **PRIV[31:0]**: Privileged configuration for block y (y = 0 to 31), belonging to super-block n.
0: Privileged and unprivileged access to block y, belonging to super-block n
1: Only privileged access to block y, belonging to super-block n
Nonsecure write to this bit is ignored if SECy bit is set in GTZC1_MPCBB_SECCFGRn.
Writes are ignored if SPLCKn bit is set in GTZC1_MPCBB_CFGLOCK.

5.8.5 GTZC1 MPCBB1 and MPCBB2 register map

Table 31. GTZC1 MPCBB1 and MPCBB2 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	GTZC1_MPCBB_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_MPCBB_CFGLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPLCK3[1]	SPLCK2[1]	SPLCK1	SPLCK0
	Reset value																													0	0	0	0
0x014 to 0x0FC	Reserved	Reserved																															
0x100	GTZC1_MPCBB_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	GTZC1_MPCBB_SECCFGR1	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
0x108)	GTZC1_MPCBB_SECCFGR2[2]	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
0x10C	GTZC1_MPCBB_SECCFGR3[2]	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
0x110 to 0x1FC	Reserved	Reserved																															
0x200	GTZC1_MPCBB_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
0x204	GTZC1_MPCBB_PRIVCFGR1	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
0x208	GTZC1_MPCBB_PRIVCFGR2[2]	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
0x20C	GTZC1_MPCBB_PRIVCFGR3[2]	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
0x210 to 0x3FC	Reserved	Reserved																															

1. In MPCBB1 bit reserved on STM32WBA5xxE devices.
2. Register in MPCBB1 is reserved on STM32WBA5xxE devices.

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

5.8.6 GTZC1 MPCBB6 register map

Table 32. GTZC1 MPCBB6 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	GTZC1_MPCBB_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_MPCBB_CFGLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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 to 0x0FC	Reserved	Reserved																															
0x100	GTZC1_MPCBB_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 to 0x1FC	Reserved	Reserved																															
0x200	GTZC1_MPCBB_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
0x204 to 0x3FC	Reserved	Reserved																															

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

6 RAMs configuration controller (RAMCFG)

6.1 Introduction

The RAMCFG configures the features of the internal SRAMs (SRAM1 and SRAM2).

6.2 RAMCFG main features

The internal SRAMs support some of the features listed hereafter, configured in RAMCFG:

- Parity error detection with failing address indication
- Write protection (1-Kbyte granularity)
- Programmable wait states
- SRAM software erase

6.3 RAMCFG functional description

6.3.1 Internal SRAMs features

Two main SRAMs (SRAM1 and SRAM2) are embedded in the devices, together with 2.4 GHz RADIO RXTXRAM and Sequence SRAM, each with specific features:

- SRAM power/retention control
  - SRAM1 and/or SRAM2 powered down in Stop modes to reduce consumption
  - SRAM1, SRAM2 and/or 2.4 GHz RADIO SRAMs retained in Standby mode.

  Refer to *Section 11: Power control (PWR)* for more details.
- SRAM erase on RDP regression:
  - SRAM1 and SRAM2 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.
- SRAM erase on reset:
  - SRAM2 is erased when a system reset occurs if the SRAM2_RST option bit is selected in the flash memory user option bytes.
  - SRAM1 is erased when a system reset occurs if the SRAM1_RST option bit is selected in the flash memory user option bytes.

  Refer to *Section 7: Embedded flash memory (FLASH)* for more details.
- Tamper protection
  - SRAM2 is protected by the tamper detection circuit, and is erased by hardware in case of tamper detection
  - SRAM2 is also erased by hardware in case of a V_{DD} BOR0 (Backup domain supply power-on)

  Refer to *Section 37: Tamper and backup registers (TAMP)* for more details.
- Parity and write protection:
  - SRAM2 features parity and write protection

  Refer to *Section 6.3.2* and *Section 6.3.3* for more details.

- Wait states and software reset:
  - SRAM1 and SRAM2 feature programmable wait state configuration.
  - SRAM1 and SRAM2 feature software erase.

  Refer to *Section 6.3.4* and *Section 6.3.5* for more details.

Table 33. SRAMs features

Feature	SRAM1	SRAM2	2.4 GHz RADIO SRAMs (SRAM6)
BAM in Stop 0 mode	X	X	-
Optional retention in Stop mode	X	X	(1)
Optional retention in Standby mode	X	X	X
Erased with RDP regression	X	X	-
Erased with tamper detection	-	X	-
Backup domain reset	-	-	-
Optionally erased with system reset	X	X	-
Software erase	X	X	-
Parity	-	X	-
Write protection	-	X	-
Wait states	X	X	-

1. 2.4 GHz RADIO SRAMs are always retained in Stop modes.

6.3.2 Internal SRAM parity

Parity is supported by SRAM2 when enabled with the SRAM2_PE user option bit, and when, furthermore, either PEIE or PENMI is enabled. Refer to *Section 7: Embedded flash memory (FLASH)* for more details.

Four parity bits are added per 32 bits of SRAM (1 bit per byte) to increase memory robustness.

The parity bits are computed and stored when writing into the SRAM2, then they are automatically checked when reading. If one byte parity fails, an NMI or IRQ can be generated. The same error can also be linked to the break input (BRK_IN) of TIM1, TIM16, and TIM17, with the SYSCFG_CFGR2.SPL control bit.

Note: When enabling SRAM2 parity, initialize by software the whole RAM at the beginning of the code, to avoid getting parity errors when reading noninitialized addresses.

When a parity error is detected, the PED and CPED bits are set in *RAMCFG SRAM2 interrupt status register (RAMCFG_M2ISR)* and *RAMCFG SRAM2 interrupt clear register (RAMCFG_M2ICR)*. An interrupt or NMI can be generated if enabled by the PEIE or PENMI bits in the *RAMCFG SRAM2 interrupt enable register (RAMCFG_M2IER)*. When the ALE bit is set in *RAMCFG SRAM2 control register (RAMCFG_M2CR)*, the failing parity SRAM offset word address is stored in the *RAMCFG SRAM2 parity error address register (RAMCFG_M2PEAR)*, together with the failing byte(s) parity error flag and the AHB bus master ID.

When the PED and CPED bits are already set any subsequent parity failing access SRAM offset word address is no longer updated. A new parity error SRAM offset word address is only latched when PED and CPED are 0.

Unaligned word and half word accesses can generate a parity error on bytes not belonging to the accessed data (see *Table 34*).

Table 34. SRAM parity access error

Access	Parity error on bytes not belonging to data
Aligned	All reported errors belong to the bytes read.
Unaligned word at byte address offset 0x1	An error on word offset address byte 0 does not belong to the read data.
Unaligned word at byte address offset 0x2	All reported errors belong to the bytes read.
Unaligned word at byte address offset 0x3	An error on word offset address byte 3 does not belong to the read data.
Unaligned half word at byte address offset 0x1	An error on word offset address byte 0 or 3 does not belong to the read data.
Unaligned half word at byte address offset 0x3	All reported errors belong to the bytes read.

Table 35. SRAM parity error bus master ID

SRAM	CPU	Debugger	DMA port 0	DMA port 1
SRAM2	0b010	0b011	0b110	0b111

6.3.3 Internal SRAM write protection

The SRAM2 is made of 64 1-Kbyte write protect pages, each can be write-protected by setting its corresponding PxWP (x = 0 to 63) bit in the *RAMCFG SRAM2 write protection register 1 (RAMCFG_M2WPR1)* and *RAMCFG SRAM2 write protection register 2 (RAMCFG_M2WPR2)*.

When writing to a write-protected page, the write is ignored and a bus error is generated.

Any SRAM erase operation erases the write-protected pages as well.

6.3.4 Internal SRAM read access latency

To read correctly data, the number of wait states (WS) must be correctly programmed in the WSC[2:0] field of the RAM control register, depending upon the voltage scaling range for SRAM1 and SRAM2, and upon AHB hclk1 clock frequency (see *Table 36*).

Table 36. Number of wait states versus hclk frequency and voltage range scaling

Wait states (latency)	AHB hclk1 (MHz)	
	V_{CORE} range 1	V_{CORE} range 2
0 WS (1 AHB cycle)	≤ 100	≤ 12
1 WS (2 AHB cycle)	-	≤ 16

After reset, the SRAM1 and SRAM2 hclk1 frequency is 16 MHz, 1 wait state (WSC) is configured in RAMCFG_MxCR.

Before entering Stop 1 mode, the software must set SRAM1 and SRAM2 wait states to at least 1 in RAMCFG_MxCR, to comply with the SYSCLK 16 MHz and range 2 configuration when exiting the mode.

6.3.5 Internal SRAM erase

SRAM erase can be requested by executing this software sequence:

1. Write 0xCA in the *RAMCFG SRAMx erase key register (RAMCFG_MxERKEYR)*.
2. Write 0x53 in the *RAMCFG SRAMx erase key register (RAMCFG_MxERKEYR)*.
3. Write 1 in the SRAMER bit of the *RAMCFG SRAM1 control register (RAMCFG_M1CR)*.

SRAM erase can also start because of a tamper (see *Section 37: Tamper and backup registers (TAMP)*), or upon a reset condition (see *Section 7.4: FLASH option bytes*).

SRAMBUSY flag is set in the related SRAM interrupt status register as long as the erase is ongoing.

The total duration of each SRAM erase is N x AHB clock cycles, where N is the size of the SRAM in 32-bit words.

If the SRAM is read while an erase is ongoing, default zero data is read on the AHB bus.

If the SRAM is written while an erase is ongoing, wait states are inserted on the AHB bus until the end of the erase operation.

An SRAM erase operation also erases write-protected pages.

6.4 RAMCFG low-power modes

Table 37. 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 SRAM2 parity is functional and an interrupt or NMI causes the device to exit Stop 0 mode.
Standby	The RAMCFG peripheral is powered down and must be reinitialized after exiting Standby mode.

6.5 RAMCFG interrupts

Table 38. RAMCFG interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop mode	Exit Standby mode
RAMCFG	Parity error detection	PED	PEIE and PENMI = 0	Write 1 in CPED	Yes	Yes(1)	No
NMI			PENMI				

1. Stop 0 mode only.

6.6 RAMCFG registers

In the registers described below, x refers to SRAM index.

Access to all RAMCFG registers can be protected by GTZC_TZSC RAMCFGSEC and GTZC_TZSC RAMCFGSPRIV.

6.6.1 RAMCFG SRAM1 control register (RAMCFG_M1CR)

Address offset: 0x000

Reset value: 0x0001 0000

31	30	29	28	27	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.	Res.	Res.	Res.	Res.	Res.
							rs								

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

Bits 18:16 **WSC[2:0]**: SRAM1 wait state configuration

This field is used to program the number of wait states inserted on the AHB when reading the SRAM1, depending on its access time.

000: Zero wait states

001: One wait state

...

111: Seven wait states

Note: Before entering Stop 1 mode, software must set SRAM1 wait states to at least 1.

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

Bit 8 **SRAMER**: SRAM1 erase

This bit can be set by software only after writing the unlock sequence in the ERASEKEY field of the RAMCFG_M1ERKEYR register. Setting this bit starts the SRAM1 erase. This bit is automatically cleared by hardware at the end of the erase operation.

0: No erase operation ongoing

1: Erase operation ongoing

Bits 7:0 Reserved, must be kept at reset value.

6.6.2 RAMCFG SRAM1 interrupt status register (RAMCFG_M1ISR)

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.	SRAM BUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							r								

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

Bit 8 **SRAMBUSY**: SRAM busy with erase operation.

0: No memory erase operation ongoing

1: Memory erase operation ongoing

Note: Depending on the SRAM, the erase operation can be performed due to software request, system reset if the enabled by user option, tamper detection or RDP regression. Refer to Table 33.

Bits 7:0 Reserved, must be kept at reset value.

6.6.3 RAMCFG SRAMx erase key register (RAMCFG_MxERKEYR)

Address offset: 0x028 + 0x040 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

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

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.

a) Write 0xCA into ERASEKEY[7:0]

b) Write 0x53 into ERASEKEY[7:0]

Note: Writing a wrong key reactivates the write protection.

6.6.4 RAMCFG SRAM2 control register (RAMCFG_M2CR)

Address offset: 0x040

Reset value: 0x0001 0000

31	30	29	28	27	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.	Res.
							rs				rw				

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

Bits 18:16 **WSC[2:0]**: SRAM2 wait state configuration

This field is used to program the number of wait states inserted on the AHB when reading the SRAM2, depending on its access time.

000: 0 wait state

001: 1 wait state

...

111: 7 wait states

Note: Before entering Stop 1 mode software must set SRAM2 wait states to at least 1.

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

Bit 8 **SRAMER**: SRAM2 erase

This bit can be set by software only after writing the unlock sequence in the ERASEKEY field of the RAMCFG_M2ERKEYR register. Setting this bit starts the SRAM2 erase. This bit is automatically cleared by hardware at the end of the erase operation.

0: No erase operation ongoing

1: Erase operation ongoing

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **ALE**: SRAM2 parity fail address latch enable

0: Failing address not stored in the SRAM2 parity error address register RAMCFG_M2PEAR

1: Failing address stored in the SRAM2 parity error address register RAMCFG_M2PEAR

Bits 3:0 Reserved, must be kept at reset value.

6.6.5 RAMCFG SRAM2 interrupt enable register (RAMCFG_M2IER)

Address offset: 0x044

Reset value: 0x0000 0000

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

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **PENMI**: Parity error NMI.

This bit is set by software and cleared only by a global RAMCFG reset

0: NMI not generated in case of parity error

1: NMI generated in case of parity error

Note: When PENMI bit is set, the RAMCFG maskable interrupt is not generated for a parity error whatever PEIE bit value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **PEIE**: Parity error interrupt enable

0: Parity error interrupt disabled

1: Parity error interrupt enabled

Bit 0 Reserved, must be kept at reset value.

6.6.6 RAMCFG SRAM2 interrupt status register (RAMCFG_M2ISR)

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAM BUSY	Res.	Res.	Res.	Res.	Res.	Res.	PED	Res.
							r							r	

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

Bit 8 **SRAMBUSY**: SRAM2 busy with erase operation.
0: No memory erase operation ongoing
1: Memory erase operation ongoing

Note: Depending on the SRAM2, the erase operation can be performed due to software request, system reset if the enabled by user option, tamper detection or RDP regression. Refer to Table 33.

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

Bit 1 **PED**: Parity error detected
0: No parity error detected
1: Parity error detected

Bit 0 Reserved, must be kept at reset value.

6.6.7 RAMCFG SRAM2 parity error address register (RAMCFG_M2PEAR)

Address offset: 0x050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BYTE[3:0]				ID[3:0]				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**
PEA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **BYTE[3:0]**: Byte parity error flag.
When ALE bit is set in RAMCFG_M2CR register, this field is updated when PED and CPED are zero and a new parity error is detected, with:
1xxx: parity error detected on fourth byte in word aligned address
x1xx: parity error detected on third byte in word aligned address
xx1x: parity error detected on second byte in word aligned address
xxx1: parity error detected on first byte in word aligned address

Bits 27:24 **ID[3:0]**: Parity error AHB bus master ID.
When ALE bit is set in RAMCFG_M2CR register, this field is updated when PED and CPED are zero and a new parity error is detected, with:
010: parity error detected on CPU access
011: parity error detected on Debugger access
110: parity error detected on DMA master port o access
111: parity error detected on DMA master port 1 access
Others: reserved

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:0 **PEA[15:0]**: Parity error SRAM word aligned address offset.PEA[1:0] read 0b00.
When ALE bit is set in RAMCFG_M2CR register, this field is updated when PED and CPED are zero and a new parity error is detected, with the SRAM word aligned address offset corresponding to the parity error.

6.6.8 RAMCFG SRAM2 interrupt clear register (RAMCFG_M2ICR)

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.	Res.	Res.
15	**14**	**13**	**12**	**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.	CPED	Res.
														rc_w1	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **CPED**: Clear parity error detect bit
Writing 1 to this bit clears the PED bit in RAMCFG_M2ISR.
Reading this bit returns the value of the RAMCFG_M2ISR PED bit.

Bit 0 Reserved, must be kept at reset value.

6.6.9 RAMCFG SRAM2 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 write protect page y write protection (y = 31 to 0)
These bits are set by software and cleared only by a system reset.
0: Write protection of SRAM2 1-Kbyte write protect page y is disabled.
1: Write protection of SRAM2 1-Kbyte write protect page y is enabled.

6.6.10 RAMCFG SRAM2 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 write protect page y write protection (y = 63 to 32)
These bits are set by software and cleared only by a system reset.
0: Write protection of SRAM2 1-Kbyte write protect page y is disabled.
1: Write protection of SRAM2 1-Kbyte write protect page y is enabled.

6.6.11 RAMCFG register map

Table 39. 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
0x000	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.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	1								0								
0x004	Reserved	Reserved																															
0x008	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								
0x00C to 0x024	Reserved	Reserved																															
0x028	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
0x02C to 0x03C	Reserved	Reserved																															
0x040	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.	Res.
	Reset value														0	0	1								0				0				
0x044	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.	PENMI	Res.	PEIE	Res.
	Reset value																													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.	PED	Res.
	Reset value																								0							0	
0x04C	Reserved	Reserved																															
0x050	RAMCFG_M2PEAR	BYTE[3:0]				ID[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEA[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
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.	CPED	Res.
	Reset value																															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 to 0x064	Reserved	Reserved																															
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 0x3FC	Reserved	Reserved																															

Refer to *Section 2.3: Memory organization* 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 DMA. 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

- Memory organization
  - Single bank architecture
  - Main memory: 1 Mbyte on STM32WBA5xxG devices, 512 Kbytes on STM32WBA5xxE devices
  - Information block: 64.5 Kbytes
- Standard and burst programming modes
- Read, program and erase operations in all voltage ranges
- 10 kcycles endurance on the whole flash memory. 100 kcycles on 256 Kbytes
- Product security activated by TrustZone option bit (TZEN)[a]
- Device life cycle managed by readout protection option bit (RDP)
- Two write protection areas)
- TrustZone® support[a]:
  - One secure area
  - One secure HDP (hide protection) area part of the secure area
- 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, power-down mode)

a. Available only on STM32WBA52/4/5xx devices.

7.3 FLASH functional description

7.3.1 Flash memory organization

The main features of the flash memory are the following:

- Single bank mode:
  - 8 Kbytes page size
  - 137 bits wide data read and write (128 effective bits plus 9 ECC bits)
  - Page and mass erase

The flash memory is organized as follows:

- Main memory block organized as single bank of 1-Mbyte containing 128 pages of 8 Kbytes (STM32WBA5xxG devices), 512-Kbyte containing 64 pages of 8 Kbytes (STM32WBA5xxE devices)
- 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 I2C (DFU). The system memory is programmed by STMicroelectronics when the device is manufactured. For further details, refer to AN2606 "*STM32 microcontroller system memory boot mode*", available on *www.st.com*.
  - 32 Kbytes immutable secure area containing the root security services (RSS and RSS library) developed by STMicroelectronics
  - Up to 512 bytes OTP (one-time programmable) bytes for user data (up to 32 quad-words). The OTP data cannot be erased and can be written only once. If only one bit is at 0, the entire quad-word cannot be written anymore, even with the value 0x0000 0000 0000 0000 0000 0000 0000 0000.
    OTP can only be written by nonsecure access and read by secure and nonsecure access.
  - 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 is based on a main area and an information block, as detailed in *Table 40*.

Table 40. Flash module 1-Mbyte single bank organization

Flash area	Flash memory address secure(1)	Flash memory address nonsecure(1)	Size (bytes)	Name
Main memory(2)	0x0C00 0000 - 0x0C00 1FFF	0x0800 0000 - 0x0800 1FFF	8 K	Page 0
	0x0C00 2000 - 0x0C00 3FFF	0x0800 2000 - 0x0800 3FFF	8 K	Page 1
	⋮	⋮	⋮	⋮
	0x0C07 C000 - 0x0C07 DFFF	0x0807 C000 - 0x0807 DFFF	8 K	Page 62
	0x0C07 E000 - 0x0C07 FFFF	0x0807 E000 - 0x0807 FFFF	8 K	Page 63
	0x0C08 0000 - 0x0C08 1FFF	0x0808 0000 - 0x0808 1FFF	8 K	Page 64
	⋮	⋮	⋮	⋮
	0x0C0F E000 - 0x0C0F FFFF	0x080F E000 - 0x080F FFFF	8 K	Page 127

Table 40. Flash module 1-Mbyte single bank organization (continued)

Flash area	Flash memory address secure[(1)]	Flash memory address nonsecure[(1)]	Size (bytes)	Name
-	0x0C10 0000 - 0x0FF7 FFFF	0x0810 0000 - 0x0BF7 FFFF	-	Reserved
Information block	0x0FF8 0000 - 0x0FF8 5FFF	0x0BF8 0000 - 0x0BF8 7FFF	24 K	RSS[(3)]
	0x0FF8 6000 - 0x0FF8 7FFF		8 K	RSS library[(3)]
Information block	0x0FF8 8000 - 0x0FF9 01FF	0x0BF8 8000 - 0x0BF8 FFFF	32 K	Boot loader
		0x0BF9 0000 - 0x0BF9 01FF	512	OTP area
-	0x0FF9 0200 - 0x0FF9 04FF	0x0BF9 0200 - 0x0BF9 04FF	-	Reserved
Information block	0x0FF9 0500 - 0x0FF9 3FFF	0x0BF9 0500 - 0x0BF9 3FFF	14.75 K	Engineering bytes
	0x0FF9 4000 - 0x0FF9 7FFF	0x0BF9 4000 - 0xBF9 7FFF	16 K	User options

1. Gray shaded areas are reserved.
2. Product dependent: pages 0 to 127 are available on STM32WBA5xxG devices, only pages 0 to 63 are available on STM32WBA5xxE devices.
3. Available only on STM32WBA52/4/5xx devices.

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, as nine bits are added per quad-word (128 bits). The ECC mechanism supports one error detection and correction, and 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 is saved in ADDR_ECC[19:0] in the *FLASH ECC register (FLASH_ECCR)*. ADDR_ECC[3:0] are always cleared.

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

Note: For an erased flash line, only one error is detected and corrected. Two errors detection is not supported.

Note: 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 prefetch buffer or in ICACHE, even if ECCC and ECCD are cleared.

The following addresses in the flash system memory are used to store words including ECC errors to allow software run-time tests on ECC correction detection capability:

- 0x0BF9 1F00 embeds a word with 1-bit error
- 0x0BF9 1F80 embeds a word with 2-bit errors

Note: If the second address is read, for instance by the debugger memory viewer, an 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 (hclk1) and the internal voltage range of the device V_{CORE}. Refer to *Section 11.5.4: Dynamic voltage scaling management*.

Table 41 shows the correspondence between wait states and CPU clock frequency.

Table 41. Number of wait states according to CPU clock (hclk1) frequency (LPM = 0)

Wait states (WS) (latency)	hclk1 (MHz)	
	V_{CORE} range 1	V_{CORE} range 2
0 WS (1 CPU cycle)	≤ 32	≤ 8
1 WS (2 CPU cycles)	≤ 64	≤ 16
2 WS (3 CPU cycles)	≤ 96	-
3 WS (4 CPU cycles)	≤ 100	-

The flash memory supports a low-power read mode when setting the LPM bit in the *FLASH access control register (FLASH_ACR)*. *Table 42* shows the correspondence between wait states and CPU clock frequency when LPM bit is set.

Table 42. Number of wait states according to CPU clock (hclk1) frequency (LPM = 1)

Wait states (WS) (latency)	hclk1 (MHz)	
	V_{CORE} range 1	V_{CORE} range 2
0 WS (1 CPU cycle)	WS ≥ hclk1 (MHz) / 10 - 1 Maximum hclk1 frequency is given by *Table 41*.	≤ 8
1 WS (2 CPU cycles)		≤ 16
2 WS (3 CPU cycles)		-
3 WS (4 CPU cycles)		-
⋮		-
9 WS (10 CPU cycles)		-

After reset, the CPU clock frequency is 16 MHz, 1 wait state (WS) is configured in the *FLASH access control register (FLASH_ACR)* and the normal read mode is selected (LPM = 0).

Before entering Stop 1 mode software must set FLASH wait states to at least 1 (LATENCY) in FLASH_ACR. This to comply with the SYSCLK 16 MHz and range 2 configuration when exiting Stop 1 mode.

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 and 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 executed by the CPU or filled in instruction cache.

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, use it carefully in low-power applications.

CPU frequency change

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)*.

To modify the read mode apply the software sequences detailed below.

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 Flash power-down mode

After reset, the flash memory is in normal mode. To reduce power consumption, it can be put in power-down mode by setting the PDREQ bit in the *FLASH access control register (FLASH_ACR)*.

Request entry in power-down mode

- Check that the flash memory is not in power-down mode and no request to put it in power-down mode is pending (PD bit in *FLASH status register (FLASH_NSSR)* and PDREQ bit in *FLASH access control register (FLASH_ACR)* must be reset).
- Unlock PDREQ write access in PDKEYR with correct keys. Refer to *FLASH power-down key register (FLASH_PDKEYR)*
- Set the PDREQ bit in the *FLASH access control register (FLASH_ACR)*.
- Check that the PD bit in the *FLASH power-down key register (FLASH_PDKEYR)* is set. The PDREQ bit in the *FLASH access control register (FLASH_ACR)* is automatically reset and write access is locked.

Note: If the memory is being accessed, the power-down request is delayed until the access is completed.

Requesting power-down entry for the flash memory already in power-down mode has no effect. The PDREQ bit in the *FLASH access control register (FLASH_ACR)* is automatically reset and the PDKEYR is locked.

Return to normal mode

Any access to flash memory in power-down mode automatically wakes up the memory, at least 5 µs are needed to wake it up.

The flash memory wake up is done in one of the following cases:

- upon a valid read access to flash memory
- upon a valid write access to flash memory
- 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 wake-up time by enabling HSI16 before waking up the flash.

7.3.5 Flash memory program and erase operations

The embedded flash memory can be programmed using in-circuit (ICP) or in-application (IAP) programming.

The 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, the IAP can use any communication interface supported by the microcontroller (such as I/Os, UART, I2C, SPI, or 2.4 GHz RADIO) 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 halts CPU when accessing flash. The CPU operation from flash proceeds correctly once the program/erase operation is completed.

Program and erase operations can be suspended to guarantee flash read and execution access for real time critical software.

The MCU supports Trustzone that defines secure and nonsecure areas in the flash memory. Program and erase operations can be performed in secure mode through the secure registers or in nonsecure mode through the nonsecure registers. For more information, refer to *Section 7.5*.

Unlock the secure/nonsecure flash control registers

After reset, write is not allowed in the flash control registers (*FLASH secure control register (FLASH_SECCR1)* and *FLASH control register (FLASH_NSCR1)*) 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 key register (FLASH_NSKEYR)*.
2. Write KEY2 = 0xCDEF89AB in the *FLASH secure key register (FLASH_SECKEYR)* or *FLASH key register (FLASH_NSKEYR)*.

Any wrong sequence locks up the *FLASH secure control register (FLASH_SECCR1)* or *FLASH control register (FLASH_NSCR1)* 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_NSCR1 (resp. FLASH_SECCR1) register can be locked again by software by setting the LOCK bit in the FLASH_NSCR1 (resp. FLASH_SECCR1) register.

*Note: The FLASH*_NSCR1 and FLASH_SECCR1 *registers cannot be written when the BSY bits are set. Any attempt to write them with the BSY bits set, causes the write to be ignored and generated a bus error and hard fault interrupt.*

Wait for data-to-write flags (WDW)

The WDW flags in the *FLASH status register (FLASH_NSSR)* and *FLASH secure status register (FLASH_SECSR)* are both set when a secure or nonsecure 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.

Secure and nonsecure busy flags

The BSY flags in the *FLASH status register (FLASH_NSSR)* and *FLASH secure status register (FLASH_SECSR)* are both set when a secure or nonsecure flash operation is started:

- Erase operation: setting the STRT bit in the *FLASH control register (FLASH_NSCR1)* or *FLASH secure control register (FLASH_SECCR1)*
- Write operation: setting the PG bit in the *FLASH control register (FLASH_NSCR1)* or *FLASH secure control register (FLASH_SECCR1)* and writing a quad-word in the flash memory
- Option bytes programming: setting the OPTSTRT in the *FLASH control register (FLASH_NSCR1)*

7.3.6 Flash memory erase sequences

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

Page erase

A page erase is possible only in two cases:

- when page is nonsecure, by a nonsecure page erase request through *FLASH control register (FLASH_NSCR1)*
- when page is secure, by a secure page erase request through *FLASH secure control register (FLASH_SECCR1)*

To erase a page, follow the procedure below:

1. Check that no flash memory operation is ongoing by checking the BSY bit in the *FLASH status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*. The best guarantee to have only a single software thread starting a flash operation is by using a hardware semaphore in HSEM.
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) in the *FLASH control register (FLASH_NSCR1)* or *FLASH secure control register (FLASH_SECCR1)*.
4. Set the STRT bit in the *FLASH control register (FLASH_NSCR1)* or *FLASH secure control register (FLASH_SECCR1)*.
5. Wait for the BSY bit to be cleared in the *FLASH status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.

Mass erase

A mass erase is possible only in two cases:

1. When full flash is nonsecure, by a nonsecure mass erase request through *FLASH control register (FLASH_NSCR1)*
2. When full flash is secure, by a secure mass erase request through *FLASH secure control register (FLASH_SECCR1)*

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 status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*. The best guarantee to have only a single software thread starting a flash operation is by using a hardware semaphore in HSEM.
2. Check and clear all nonsecure error programming flags due to a previous programming. If not, the PGSERR bit is set.
3. Set the MER bit in the *FLASH control register (FLASH_NSCR1)* or *FLASH secure control register (FLASH_SECCR1)*.
4. Set the STRT bit in the *FLASH control register (FLASH_NSCR1)* or *FLASH secure control register (FLASH_SECCR1)*.
5. Wait for the BSY bit to be cleared in the *FLASH status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.
6. The MER bit can be cleared if no more mass erase is requested.

Note: Page and mass erase are only started when erase operations have not been suspended by ES in FLASH control 2 register (FLASH_NSCR2) or FLASH secure control 2 register (FLASH_SECCR2).

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 or to perform a mass erase, the software must have sufficient privilege (see Table 60 and Table 61).

7.3.7 Flash memory programming sequences

The flash memory is programmed 137 bits at a time (128-bit data + 9-bit ECC).

Programming in a previously programmed address is not allowed, except if the data to write is full zero, any attempt sets the PROGERR flag in the *FLASH status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR).*

It is possible only to program quad-word (4 x 32-bit data).

- Any attempt to write byte or half-word sets the SIZERR flag in the *FLASH 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 status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR).*

Flash memory programming

Flash memory programming is possible only in two cases:

- when program location is nonsecure, by a nonsecure program request through *FLASH control register (FLASH_NSCR1)* and nonsecure quad-word write sequence
- when program location is secure, by a secure program request through *FLASH secure control register (FLASH_SECCR1)* and secure quad-word write sequence.

The programming sequence is as follows:

1. Check that no flash main memory operation is ongoing by checking the BSY bit in the *FLASH status register (FLASH_NSSR)* or *FLASH secure status register*

(FLASH_SECSR). The best guarantee to have only a single software thread starting a flash operation is by using a hardware semaphore in HSEM.

2. Check that the write buffer is empty by checking the WDW bit in the *FLASH 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 control register (FLASH_NSCR1)* or *FLASH secure control register (FLASH_SECCR1)*.
5. Perform the data write operation at the desired flash memory address, or in the OTP area. Only a quad-word can be programmed:
   – Write a first word in an address aligned on a quad-word address. The WDW bits in the *FLASH 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 both the WDW bit and BSY bit are cleared in the *FLASH status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*. The software must first check that the WDW is cleared before checking that the BSY is cleared.
8. If the EOP flag is set in the *FLASH 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 control register (FLASH_NSCR1)* or *FLASH secure control register (FLASH_SECCR1)* if there is no more programming request.

Note: Flash program operations start only when program operations have not been suspended by PS in FLASH control 2 register (FLASH_NSCR2) or FLASH secure control 2 register (FLASH_SECCR2).

Note: When the flash memory interface receives a good program 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.

Option bytes modifications or erase requests are not allowed when the WDW bit is set.

Programming is possible only if the privileged and security attributes are respected. Refer to Section 7.7.

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.

Burst programming (8 quad-words)

This programming is possible only in two cases:

- when program location is nonsecure, by a nonsecure program request through *FLASH control register (FLASH_NSCR1)* and nonsecure quad-word write sequence.
- when program location is secure, by a secure program request through *FLASH secure control register (FLASH_SECCR1)* and secure quad-word write sequence.

The burst programming sequence is as follows:

1. Check that no operation is ongoing in the flash main memory by checking the BSY bit in the *FLASH 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 status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*.
3. Check and clear all error programming flags due to previous programming(s). If none, PGSERR is set.
4. Set the BWR and PG bits in the *FLASH control register (FLASH_NSCR1)* or *FLASH secure control register (FLASH_SECCR1)*.
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 x quad-word address (multiple of 0x80). The WDW bits in the *FLASH 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 both the WDW bit and BSY bit are cleared in the *FLASH status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)*. The software must first check that the WDW bit is cleared before checking that the BSY bit is cleared.
8. If the EOP flag is set in the *FLASH status register (FLASH_NSSR)* or *FLASH secure status register (FLASH_SECSR)* (meaning that the programming operation has been successful and the EOPIE bit is set), it has to be cleared by software.
9. Clear the BWR and PG bits in the *FLASH control register (FLASH_NSCR1)* or *FLASH secure control register (FLASH_SECCR1)* if there is no more programming request.

Note: Program operation can started only when program operations have not been suspended by PS in FLASH control 2 register (FLASH_NSCR2) or FLASH secure control 2 register (FLASH_SECCR2).

Note: When the flash memory interface receives 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.

7.3.8 Flash memory programming erase suspend

The prevent real time critical program execution from being interrupted by flash main memory program or erase operations, these operations can be suspended during the real time program execution phase. For this, a program and a erase suspend feature is available which prevents any new program or erase operation to be started. These operations are suspended (delayed) until after the suspend feature has been disabled. During the suspend period and after any ongoing program or erase operation has been completed, program execution and memory read operation from flash are still available.

Due to the difference in timing constrains for program and erase operation, these can be suspended separately by the program suspend register PS and ES bits in *FLASH control 2 register (FLASH_NSCR2)* and *FLASH secure control 2 register (FLASH_SECCR2)*, where PS suspends any new program operation and ES suspends any new erase operation.

Furthermore, program and erase suspend is available as secure and nonsecure request, resulting in possibility of separate program and erase suspend control from the secure and nonsecure sides.

When suspending flash programming by setting a PS bit the following happens:

- Any ongoing secure or nonsecure program operation, started before the PS has been set, completes. The maximum latency for a flash main memory program suspend to guarantee uninterrupted execution from flash is the time to program the flash (see data sheet for times).
- A new requested secure and nonsecure program operation is suspended until all PS bits are cleared. The suspended program operation has the BSY bit set from the moment the last quad-word has been written in a flash program sequence or the last of the 32 words in a flash burst program sequence.

Note: Program suspend is applied to both flash main memory and OTP programming.

Warning: When programming is suspended and a new program operation is requested after any good program sequence, any additional flash write or write to FLASH_NSCR1 and FLASH_SECCR1 halts the bus master (CPU or DMA) that writes.

When suspending flash erase by setting an ES bit the following happens:

- Any ongoing secure or nonsecure erase operation, started before the ES has been set, completes. The maximum latency for a flash main memory erase suspend to guarantee uninterrupted execution from flash is the time to erase the memory (see data sheet for times).
- A new requested secure and nonsecure erase operation is suspended until all ES bits are cleared. The suspended erase operation has the BSY bit set from the moment the erase STRT bit is set. Both page erase and mass erase are suspended.

Note: Erase suspend has no effect when the flash memory is erased by RDP regression. Flash erase requested via RDP regression is performed even when ES bits are set.

Table 43. Program and erase suspend control

<table>
<tr><th colspan="2">PS</th><th colspan="2">ES</th><th rowspan="2">Description</th></tr>
<tr><th>Secure</th><th>Nonsecure</th><th>Secure</th><th>Nonsecure</th></tr>
<tr><td>0</td><td>0</td><td colspan="2" rowspan="2">0</td><td>Program and erase operation granted</td></tr>
<tr><td colspan="2">Not 00</td><td>Any new start of program operation suspended
Erase operations granted</td></tr>
<tr><td>0</td><td>0</td><td colspan="2" rowspan="2">Not 00</td><td>Program operations granted
Any new start of erase operation suspended</td></tr>
<tr><td colspan="2">Not 00</td><td>Any new start of program or erase operation suspended</td></tr>
</table>

7.3.9 Flash memory endurance

Each memory page can be written and erased 10000 times. In addition, up to 256 Kbytes (32 pages) feature an increased endurance of 100 kcycles, which can be used for data storage that usually needs more intensive cycling capability compared to code storage. Any page can be chosen to be cycled more than 10000 times (up to 100000 times). It is the application responsibility to limit the size of the flash area cycled more than 10000 times to these 32 pages.

7.3.10 Flash memory errors flags

Flash programming errors

Several kind of errors can be detected during secure and nonsecure 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 nonsecure flags are only set during a nonsecure operation.

- **PROGERR**: secure/nonsecure programming error

  It is set when the word to program is pointing to an address:
  - not previously erased
  - already fully programmed to 0
  - already programmed and the new value to program is not full 0
  - for OTP programming, when the address is already programmed
- **SIZERR**: secure/nonsecure 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/nonsecure 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 x 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 MER 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 MER and PER are cleared.
  - If page and mass erase are requested at the same time, STRT and PER are set while MER 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 nonsecure 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 nonsecure 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 56* to *Table 61* for the conditions of WRPERR flag setting. In addition, WRPERR flag is set when start a user option programming with OPTSTRT in RDP level 2.
- **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 47*
- 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 SECBOOTADD0 option bit while BOOT_LOCK is set
- to modify SECWMR2 while HDP_ACCDIS is set
- to modify the option bytes, when RDP is level 2
- 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 OEM1 key is shifted through JTAG
- to regress from RDP level 1 to RDP level 0.5 while OEM2LOCK bit is set and a wrong OEM2 key is shifted through JTAG
- 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 nonsecure program or erase operation, one of the following programming error flags is set:

- nonsecure programming error flags: PROGERR, SIZERR, PGAERR, PGSERR, OPTWRERR or WRPERR is set in the *FLASH status register (FLASH_NSSR)*.

  If the nonsecure error interrupt enable bit ERRIE is set in the *FLASH control register (FLASH_NSCR1)*, 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_SECCR1)*, 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.

7.3.11 Power-down during programming or erase operations

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 programming or erase operations

The contents of the flash memory currently being accessed during a flash memory program or erase operation are not guaranteed if a reset occurs. When a system reset occurs during a flash memory program or erase operation the status of the flash memory can be recovered from the CODE_OP in *FLASH operation status register (FLASH_OPSR)*, and the associated accessed address can be recovered from ADDR_OP.

It is the software responsibility to check the status of the flash memory and to take corrective actions. This is advised to be done after each system reset and before performing any other programming or erase operation.

CODE_OP and ADDR_OP are reinitialized when starting a program or erase operation. When a program or erase operation completes successfully CODE_OP and ADDR_OP are reset to 0.

Table 44 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 44. Flash operation interrupted by a system reset

CODE_OP	Operation interrupted	Address	System flash	Corrective action
0x0	No operation	Reserved		None
0x1	Single write	ADDR_OP	SYSF_OP	Page erase and single write at same location
0x2	Burst write	ADDR_OP	SYSF_OP	Page erase and burst write at same location
0x3	Page erase	ADDR_OP	Reserved	Erase same page
0x4	Reserved			
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 flash memory locations with interrupted program operation, may no longer be writable and on read may generate a ECC error. Consequently, the remaining page content must be saved before performing a page erase and restored afterwards.

For OTP locations, it is not possible to perform a page erase. An OTP quad-word with interrupted program operation, 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*). The user option bytes are accessible through the flash memory registers interface.

Table 45 describes the organization of the user option bytes available in the flash memory interface registers.

Table 45. User option byte organization mapping

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register map
TZEN	Res.	Res.	Res.	NBOOT0	NSWBOOT0	SRAM2_RST	SRAM2_PE	Res.	Res.	Res.	Res.	WWDG_SW	IWDG_STDBY	IWDG_STOP	IDWG_SW	SRAM1_RST	Res.	NRST_STDBY	NRST_STOP	Res.	BOR_LEV[2:0]			RDP[7:0]								*Section 7.9.13*
NSBOOTADD0[24:0]																									Res.	Res.	Res.	Res.	Res.	Res.	Res.	*Section 7.9.16*
NSBOOTADD1[24:0]																									Res.	Res.	Res.	Res.	Res.	Res.	Res.	*Section 7.9.17*
SECBOOTADD0[24:0]																									Res.	Res.	Res.	Res.	Res.	Res.	BOOT_LOCK	*Section 7.9.18*
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM_PSTRT[6:0]							*Section 7.9.19*
HDPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.							*Section 7.9.20*
UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPA_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPA_PSTRT[6:0]							*Section 7.9.20*
UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPB_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPB_PSTRT[6:0]							*Section 7.9.21*
OEM1KEY[31:0]																																*Section 7.9.22*
OEM1KEY[63:32]																																*Section 7.9.22*
OEM2KEY[31:0]																																*Section 7.9.23*
OEM2KEY[63:32]																																*Section 7.9.24*

7.4.2 Option bytes programming

After reset, the options related operation bits OPTSTRT and OBL_LAUNCH in the FLASH_NSCR1 register are write-protected. To run any operation on the option bytes page, the option lock bit OPTLOCK in the *FLASH control register (FLASH_NSCR1)* must be cleared.

The following sequence is used to unlock this register:

1. Unlock the FLASH_NSCR1 register with the LOCK clearing sequence (refer to *Unlock the secure/nonsecure 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_NSCR1 *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 in the FLASH_NSCR1 register.
5. Wait for the BSY bit to be cleared.
6. Disable LSE oscillator by setting LSEON to 0 and waiting for LSERDY to be 0.
7. Set the OBL_LAUNCH option bit in the in the FLASH_NSCR1 register to start option bytes loading.

Note: If the OPTWERR or PGSERR error bit is set, the old option byte values are kept.

*Note: The FLASH_*NSCR1 *register cannot be written when the OBL_LAUNCH bit is set. Any attempt to write it with the OBL_LAUNCH bit set, causes the write to be ignored and generated a bus error and hard fault interrupt.*

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 control register (FLASH_NSCR1)*
- after a power reset (BOR0 reset) or exit from Standby modes

After a system reset, internal option registers are copied into option registers and factory-programmed values are loaded. The internal option 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.

Note: The factory-programmed value for LSETRIM is only loaded when SBF is cleared. They are not loaded on exit from Standby modes. See Section 12.4.4: LSE clock and Section 11.7.8: PWR Standby mode for more information on these features.

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 possible only 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 possible only when RDP is changing from level 1 to level 0 or when RDP is in level 0.
- SECBOOTADD0 option bytes
  - It cannot be modified when BOOT_LOCK option bit is set.
- SECWMRy option bits
  - It is not possible to modify secure (SECWM_PSTRT and SECWM_PEND option bits) and HDP (HDP_PEND and HDPEN option bits) area when the HDP_ACCDIS bit is set.
- RDP option bits
  - Depending on OEMx key RDP lock state, the following regressions are blocked: from RDP level 2 to level 1, from RDP level 1 to level 0.5 or from level 1 to level 0. Refer to *Device life cycle managed by readout protection (RDP) transitions*.
- WRPxR 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, HDP areas
- Secure or nonsecure 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 nonsecure 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, no HDP area as shown in *Table 46*.

Table 46. Default secure option bytes after TZEN activation

Secure watermark option bytes values after OBL when TZEN is activated (from 0 to 1)	Security attribute
SECWM_PSTRT = 0 and SECWM_PEND = 0x7F	All flash memory secure
HDPEN = 0 and HDP_PEND = 0	No secure HDP area

Illegal access generation

A nonsecure 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 GTZC_TZIC_IERx register.

A nonsecure access to a secure flash register is RAZ/WI and generates an illegal access event. An illegal access interrupt is generated if the FLASH_REGIE illegal access interrupt is enabled in the GTZC_TZIC_IERx register.

Deactivate TrustZone security

Deactivation of TZEN (from 1 to 0) is possible only 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*)
- Secure interrupts (refer to *Section 7.8*)
- 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 nonsecure read and write accesses. One non-volatile secure areas can be defined by option bytes and can be read or written by a secure access only: one area can be selected with a page granularity.

The secure areas are defined by a start-page offset and end-page offset using the SECWM_PSTRT and SECWM_PEND option bytes. These offsets are defined in the secure watermark address registers *FLASH secure watermark register 1 (FLASH_SECWMR1)*.

The SECWM_PSTRT and SECWM_PEND option bytes can only be modified by secure firmware when the HDP_ACCDIS bit is reset. If the HDP_ACCDIS bit is set, the SECWM_PSTRT and SECWM_PEND cannot be modified until next system reset.

Table 47. Secure watermark-based area

Secure watermark option bytes values (x = 1, 2)	Secure watermark protection area
SECWM_PSTRT > SECWM_PEND	No secure area
SECWM_PSTRT = SECWM_PEND	One page defined by SECWM_PSTRT is secure watermark-based protected
SECWM_PSTRT < SECWM_PEND	The area between SECWM_PSTRT and SECWM_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 nonsecure 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 HDP_ACCDIS bit in the *FLASH secure HDP control register (FLASH_SECHDPCR)*.

When the HDP_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 HDP_ACCDIS bit can be cleared only by a system reset.

Note: It is the software responsibility to take any appropriate action to protect the HDP code before resetting the HDPEN bit such as erasing the HDP area and flushing the instruction cache.

One flash secure hide protection (HDP) area can be defined with a page granularity.

The secure HDP area is enabled by the HDPEN. When the HDPEN bit is reset, there is no HDP area. The HDPEN bit can be set or reset on the fly by the secure firmware if the HDP_ACCDIS bit is reset. If the HDP_ACCDIS bit is set, the HDPEN 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 HDP_PEND option bytes while the start-page offset is already defined by SECWM_PSTRT option bytes.These offsets are defined in the secure watermark address registers: *FLASH secure watermark register 1 (FLASH_SECWMR1)*, and *FLASH secure watermark register 2 (FLASH_SECWMR2)*.

For example, to protect by HDP from the address 0x0C00 4000 (included) to the address 0x0C00 5FFF (included):

- The option bytes registers must be programmed with:
  – SECWM_PSTRT = 0x2
  – HDP_PEND = 0x3

If an invalid secure HDP area is defined (as described in *Table 48*), the OPTWERR flag error is set and option bytes modification is discarded.

Table 48. Secure hide protection

HDPx watermark option bytes values (x = 1, 2)		Hide protection area
HDPEN = 0	-	No secure HDP area
HDPEN = 1	SECWM_PSTRT ≤ HDP_PEND ≤ SECWM_PEND	The area between SECWM_PSTRT and HDP_PEND is secure HDP protected.
	Others	Invalid secure area. Hide protection 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 nonsecure using the block-based configuration registers FLASH_SECBBRx to configure the security attribute.

When the page security attribute, bit SECBBy in SECBBRx, 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 SECBBy bit in SECBBRx 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 in SECBBRx.

Caution: Switching a page or memory block from secure to nonsecure does not erase the content of the associated block. User secure software must perform the following needed operations before switching a block between secure and nonsecure attribute:

- Erase page content,
- Invalidate the instruction cache.

Note: For SECBBRx bit i access control, refer to Table 62.

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_SECCR1 register.

The RCC FLASHEN and FLASHSMEN bits security attributes 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_SECCR1 register. A secure firmware setting this INV bit allows a nonsecure firmware to disable the flash memory clock when the flash memory is in power down or when the MCU enters low-power modes.

Table 49. Flash security state

Secure area	INV bit	Flash security state
None	0	Nonsecure
	1	Secure
Yes	0	Secure
	1	Nonsecure

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_PRIVBBRx to configure the privilege attribute.

When the page privilege attribute, bit PRIVBBy in PRIVBBRx, 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 in PRIVBBRx.

Caution: Switching a page or memory block from privileged to unprivileged does not erase the content of the associated block.

Note: For PRIVBBRx bit access control, refer to Table 63 and Table 64.

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_PRIFCFGR)*, with the following rules:

- When the SPRIV (resp. NSPRIV) bit is reset, all secure (resp. nonsecure) 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. nonsecure) flash memory registers can be read and written by privileged access only. Unprivileged access to a privileged registers is RAZ/WI.

Table 59 summarizes the flash memory registers access control.

7.6 Flash memory protection

The flash memory interface implements the following protection mechanisms:

- readout protection (RDP)
- two write protection (WRP) non-volatile areas
- additional secure protections when TrustZone is active (refer to *Section 7.5*)
  - one secure watermark-based non-volatile area
  - one secure hide protection non-volatile area
  - 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, with page granularity.

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

The WRP “y” area (y = A,B) is defined as follows:

- from the address: flash base address + [WRPy_PSTRT x 0x2000] (included)
- to the address: flash base address + [(WRPy_PEND+1) x 0x2000] (excluded)

For example, to protect by WRP from the address 0x0806 2000 (included) to the address 0x0807 3FFF (included):

- FLASH_WRPAR register must be programmed with:
  - WRPA_PSTRT = 0x31
  - WRPA_PEND = 0x39

  WRPB_PSTRT and WRPB_PEND in FLASH_WRPBR can be used instead (area “B” in flash).

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 nonsecure 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 system flash memory and to the 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 nonsecure) 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 50. WRP protection

WRP registers values (y = A / B)	WRP area
WRPy_PSTRT = WRPy_PEND	Page WRPy is protected
WRPy_PSTRT > WRPy_PEND	No WRPy area
WRPy_PSTRT < WRPy_PEND	The pages from WRPy_PSTRT to WRPy_PEND are protected

Write protection lock

Each WRP area can be independently locked by writing 0 to the UNLOCK bit in the *FLASH WRP area A address register (FLASH_WRPAR)*, or *FLASH WRP area B address register*

(FLASH_WRPBR). 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.

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 OEM1 key in order to prevent a regression to RDP level 0 for users not knowing the key.
- If RDP level is 2, either provision a OEM1 key (refer to first bullet) or do not provision a OEM2 key (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, and the SRAMs.

Readout protection levels when Trustzone is disabled

There are three levels of readout protection from no protection (level 0) to maximum protection or no debug (level 2). The flash memory is protected according to the RDP option byte value shown in *Table 51*.

Table 51. 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, 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.
- 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 definitely disabled.
    - if OEM2 key has been provided, JTAG and SWD remain enabled under reset only to interface with DBGMCU_SR, DBGMCU_DBG_AUTH_HOST and

DBG_MCU_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, any attempt to modify option bytes set the OPTWERR flag. Thus, the non-volatile option areas can no longer be modified, and 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 52. 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)	Yes
	2	Yes	No	N/A	N/A	N/A	N/A
OTP	1	Yes	Yes(5)	N/A	Yes	Yes(5)	N/A
	2	Yes	Yes(5)	N/A	N/A	N/A	N/A
Backup registers	1	Yes	Yes	N/A	No	No	N/A(6)
	2	Yes	Yes	N/A	N/A	N/A	N/A
SRAM2	1	Yes	Yes	N/A	No	No	N/A(6)
	2	Yes	Yes	N/A	N/A	N/A	N/A

1. When the protection level 2 is active, the debug port, the boot from RAM and the boot from system memory are disabled.
2. The system memory is only read-accessible, whatever the protection level (0, 1 or 2) and execution mode.
3. 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. OTP can only be written once.
6. All SRAMs and backup registers are erased when regressing RDP 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 *Table 51*.

Table 53. Flash memory readout protection status (TZEN = 1)

RDP byte value	Readout protection level
0xAA	Level 0
0x55	Level 0.5
Any value except 0xAA or 0x55 or 0xCC	Level 1
0xCC	Level 2

- Level 0: no protection

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

  – **RSS mode**: when booting from RSS, the debug access is disabled while executing RSS code.
- Level 0.5: nonsecure debug only

  All read and write operations (if no write protection is set) from/to the nonsecure flash memory are possible. The debug access to secure area is prohibited. Debug access to nonsecure 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).

  – **Nonsecure debug mode**: nonsecure debug is possible when the CPU is in nonsecure state. The secure flash memory, the secure backup registers and SRAMs area are inaccessible; the nonsecure flash memory, the nonsecure backup registers and the nonsecure 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).

  – **Nonsecure debug mode**: nonsecure debug is possible when the CPU is in nonsecure state. However, an intrusion is detected in case of debug access: the flash main memory, the backup registers and the SRAM2 are totally inaccessible;

any read or write access to the flash main memory generates a bus error and a hard fault interrupt.

- **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 definitely disabled.
    - if OEM2 key has been provided, JTAG and SWD remain enabled under reset only to interface with DBGMCU_SR, DBGMCU_DBG_AUTH_HOST and DBG_MCU_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. Thus, the non-volatile security areas SECWM, HDP and WRP areas can no longer be modified, 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 54. 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)	Yes
	1	Yes	Yes(5)	N/A	Yes	Yes(5)	Yes
	2	Yes	No	N/A	N/A	N/A	N/A

Table 54. 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
OTP	0.5	Yes	Yes(6)	N/A	Yes	Yes(6)	N/A
	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	0.5	Yes	Yes	N/A	Yes(2)	Yes(2)	N/A(7)
	1	Yes	Yes	N/A	No	No	N/A(7)
	2	Yes	Yes	N/A	N/A	N/A	N/A
SRAM2	0.5	Yes	Yes	N/A	Yes(2)	Yes(2)	N/A(7)
	1	Yes	Yes	N/A	No	No	N/A(7)
	2	Yes	Yes	N/A	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. OTP can only be written once.
7. All SRAMs and TAMP backup registers are erased when regressing RDP to level 0.5 and level 0.

Device life cycle managed by readout protection (RDP) transitions

It is easy to move from level 0 or level 0.5 to level 1 by changing the value of the RDP byte to any value (except 0xCC). By programming the 0xCC value in the RDP byte, it is possible to go to level 2 either directly from level 0 or from level 0.5 or 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 SRAM1, SRAM2, PKA SRAM and TAMP backup registers is performed. The OTP area is not erased.

Note: Before launching a RDP regression, the software must invalidate the ICACHE and wait for the BUSYF bit to get cleared.

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 nonsecure watermark-based areas are erased (regardless if it is defined as secure by block-based). The SRAM1, SRAM2, PKA SRAM and TAMP backup registers are mass erased. The OTP area is not erased. The RDP level 0.5 and partial nonsecure erase are only available when TrustZone is active.

Note: Before launching a RDP regression, the software must invalidate the ICACHE and wait for the BUSYF bit to get cleared.

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 control register (FLASH_NSCR1).

Before launching a RDP regression, the software must invalidate the ICACHE and wait for the BUSYF bit to get cleared.

Figure 16. RDP level transition scheme when TrustZone is disabled (TZEN = 0)

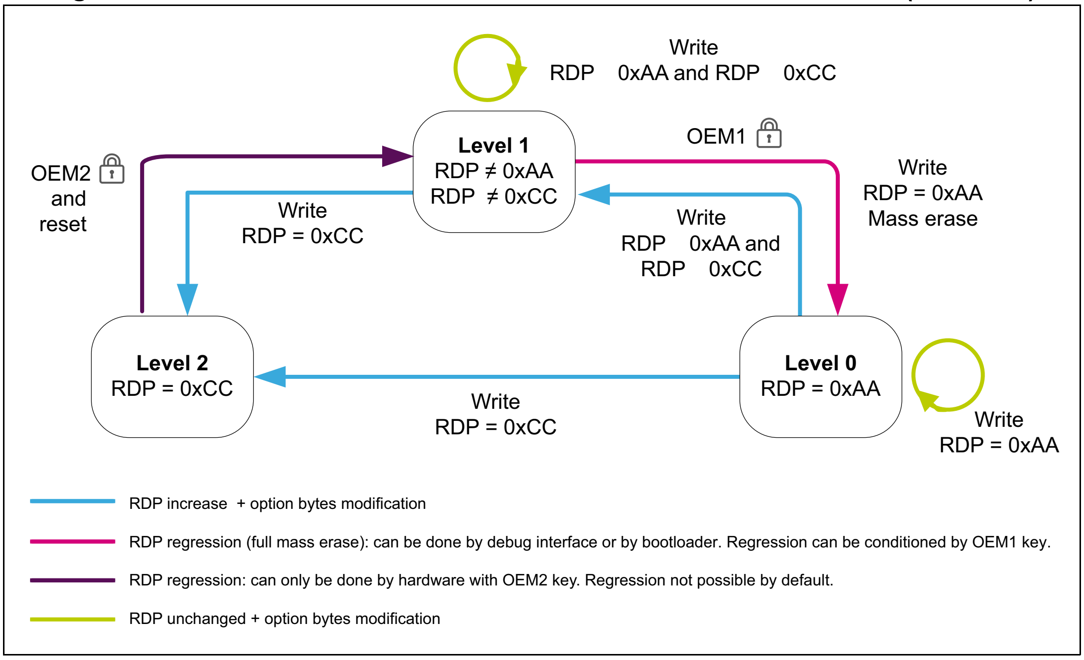
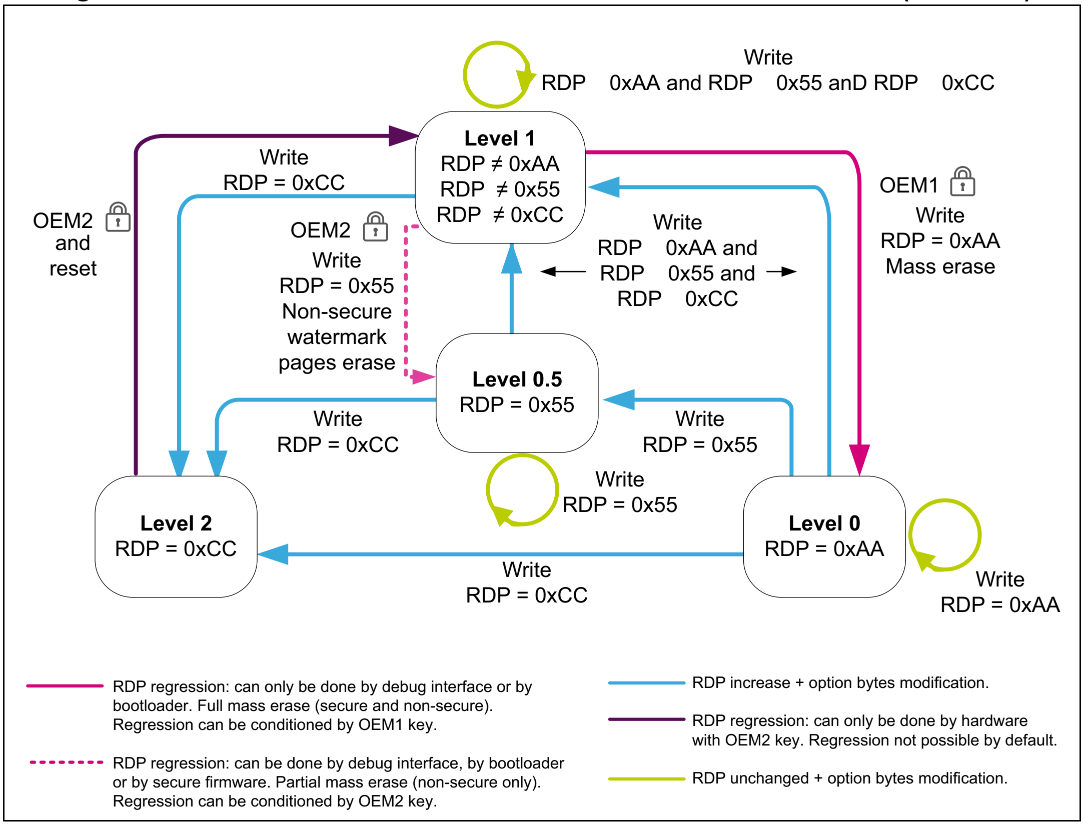


Figure 17. RDP level transition scheme when TrustZone is enabled (TZEN = 1)



OEM1/OEM2 lock activation

Two 64-bit keys (OEM1 key and OEM2 key) 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)*. OEM1 key and OEM2 key cannot be read through these registers. They are read as zero.

OEM1 key can be modified:

- in readout protection RDP level 0
- in readout protection RDP level 0.5 or 1 if OEM1LOCK = 0 in *FLASH status register (FLASH_NSSR)*

OEM2 key can be modified:

- in readout protection level 0 or 0.5
- in readout protection level 1 if OEM2LOCK = 0 in *FLASH 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 key 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 OEM1 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 control register (FLASH_NSCR1)*.
- Wait for the BSY bit to be cleared and check that OPTWERR is not set.
- Disable LSE oscillator by setting LSEON to 0 and waiting for LSERDY to be 0.
- 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 key 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 OEM 2 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 control register (FLASH_NSCR1)*.
- Wait for the BSY bit to be cleared and check that OPTWERR is not set.
- Disable LSE oscillator by setting LSEON to 0 and waiting for LSERDY to be 0.
- Set the OBL_LAUNCH option bit to start option bytes loading or perform a power-on reset.
- Check that OEM2LOCK is set.

Note: The OEM1 key and OEM2 key must not contain only 1 or only 0.

OEM1 RDP lock mechanism

The OEM1 key RDP lock mechanism is active when the OEM1LOCK bit is set. It conditions the RDP level 1 to RDP level 0 regression.

In order to regress from RDP level 1 to RDP level 0, the following sequence must be applied:

- Shift OEM1 key bit[31:0] followed by OEM1 key bit[63:32] through the JTAG or SWD under reset in the DBGMCU_DBG_AUTH_HOST register.
  - If this key is matching the OEM1KEY register value, the RDP regression can be launched by setting the OPTSTRT bit.
  - If the key is not matching the OEM1KEY register value, RDP regression and any access to the flash are blocked until a next power on rest.

Attempting to regress from RDP level 1 to RDP level 0 without following this sequence sets the OPTWERR flag 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 key RDP lock mechanism is active when the OEM2LOCK bit is set. It allows the following actions:

- Condition 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 OEM2 key bits[31:0] followed by OEM2 key bits[63:32] through the JTAG or SWD under reset in the DBGMCU_DBG_AUTH_HOST register.
  - If this key is matching the OEM2KEY register value, the RDP regression can be launched by setting the OPTSTRT bit.
  - If the key is not matching the OEM2KEY register value, RDP regression and any access to the flash are blocked until a next power on rest.

Attempting to regress from RDP level 1 to RDP level 0.5 without following this sequence, sets the OPTWERR flag and the option bytes remain unchanged.

To regress from RDP level 2 to RDP level 1, apply the following unlock sequence:

- Shift OEM2 key bits[31:0] followed by OEM2 key bits[63:32] through the JTAG or SWD under reset in the DBGMCU_DBG_AUTH_HOST register.
  - If this key is matching the OEM2KEY register value, the RDP regression is launched by hardware (it is not possible to execute instructions when the key is matching). Subsequently apply a power-on reset (cycle V_{DD} power supply OFF and ON).
  - If the key is not matching the OEM2KEY register value, RDP regression and any access to the flash are blocked until a next power on rest.

Attempting to regress from RDP level 2 to RDP level 1 without following this sequence, leaves option bytes 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 OPTWERR flag is set and the option bytes remain unchanged.

7.7 Summary of flash memory and registers access control

The following tables summarize the flash memory and registers accesses status versus RDP level, WRP and HDP protections.

Table 55. Flash memory access vs. 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">Nonsecure page</th><th colspan="2">Secure page</th><th rowspan="2">Nonsecure or secure page</th></tr>
<tr><th>HDP area (HDPEN = 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="3">Bus error</td></tr>
<tr><td>Read</td><td colspan="2">RAZ, flash illegal access event</td></tr>
<tr><td>Write</td><td rowspan="2">No WRP: OK
WRP pages: WI and non secure WRPERR flag set</td><td colspan="2" rowspan="2">WI, non secure WRPERR flag set, flash illegal access event</td></tr>
<tr><td>Page erase</td><td>WI, non secure WRPERR flag set</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. Refers to flash memory secure configurations different from the one described for HDP protections. Example: flash memory secure and HDP area enabled but ACCDIS = 0.

Table 56. Flash memory access vs. RDP level when TrustZone is disabled (TZEN = 0)

<table>
<tr><th>Access type</th><th>RDP level 0, RDP level1 no intrusion(1) or RDP level 2</th><th>RDP level 1 with intrusion(2)</th></tr>
<tr><td>Fetch</td><td rowspan="2">OK</td><td rowspan="3">Bus error</td></tr>
<tr><td>Read</td></tr>
<tr><td>Write</td><td rowspan="2">No WRP: OK
WRP pages: WI and non secure WRPERR flag set</td></tr>
<tr><td>Erase</td><td>WI and non secure WRPERR flag set</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 57. 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)
		Nonsecure flash	Secure flash: HDP area (HDPEN=1 and ACCDIS = 1)	Secure flash: Others(3)	Mix nonsecure and secure flash memory	Nonsecure or secure flash memory
Secure	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	Mass erase	No WRP: OK WRP pages: WI and nonsecure WRPERR flag set	WI, nonsecure 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 flash memory secure configurations different from the one described for HDP protections. Example: flash memory secure and HDP area enabled but ACCDIS = 0.

Table 58. Flash system memory, OTP and RSS accesses(1)

Access type		System memory (bootloader)	OTP(2)	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
Nonsecure (TZEN = 0 or TZEN = 1)	Fetch	OK	Bus error	Bus error
	Read		OK	RAZ(3)
	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. Write to a non-virgin OTP generate a PGSERR.
3. Flash memory illegal access event is generated when TZEN = 1.

Table 59. Flash registers access[(1)]

<table>
<tr><th colspan="3" rowspan="2">Access type</th><th colspan="2">Nonsecure register</th><th colspan="2">Secure register</th></tr>
<tr><th>NSPRIV = 1</th><th>NSPRIV = 0</th><th>SPRIV = 1</th><th>SPRIV = 0</th></tr>
<tr><td>Fetch</td><td>Secure/ nonsecure</td><td>Privileged/ unprivileged</td><td colspan="4">Bus error</td></tr>
<tr><td rowspan="4">Read/ Write</td><td rowspan="2">Secure[(2)]</td><td>Privileged</td><td colspan="4">OK</td></tr>
<tr><td>Unprivileged</td><td>RAZ/WI</td><td>OK</td><td>RAZ/WI</td><td>OK</td></tr>
<tr><td rowspan="2">Nonsecure[(3)]</td><td>Privileged</td><td colspan="2">OK</td><td colspan="2" rowspan="2">RAZ/WI and a flash memory register illegal access event[(4)]</td></tr>
<tr><td>Unprivileged</td><td>RAZ/WI</td><td>OK</td></tr>
</table>

1. Except SECBBRx, PRIVBBRx and PRIVCFGR registers.
2. Secure access is valid only when TrustZone is active (TZEN = 1).
3. Nonsecure access are valid when TrustZone is active or disabled.
4. Flash register illegal access event is generated only when TZEN = 1.

Table 60. Flash page access versus privilege mode[(1)]

<table>
<tr><th colspan="2">Access type</th><th>Unprivileged page</th><th>Privileged page</th></tr>
<tr><td>Fetch, Read/Write, Page erase</td><td>Privileged</td><td colspan="2">OK</td></tr>
<tr><td>Fetch, Read</td><td>Unprivileged</td><td rowspan="2">OK</td><td>RAZ</td></tr>
<tr><td>Write, Page erase</td><td>Unprivileged</td><td>WI, secure or nonsecure WRPERR flag set</td></tr>
</table>

1. When TZEN = 1, access must be granted by security firewall before privilege is considered.

Table 61. Flash mass erase versus privilege mode[(1)]

<table>
<tr><th colspan="2">Access type</th><th>Unprivileged flash memory</th><th>Privileged flash memory</th><th>Mix unprivileged and privileged flash memory</th></tr>
<tr><td>Mass erase</td><td>Privileged</td><td colspan="3">OK</td></tr>
<tr><td>Mass erase</td><td>Unprivileged</td><td>OK</td><td colspan="2">WI, secure or nonsecure WRPERR flag set</td></tr>
</table>

1. When TZEN = 1, access must be granted by security firewall before privilege is considered.

Table 62. SECBBRx registers access when TrustZone is active (TZEN = 1)

<table>
<tr><th colspan="3">Access type</th><th>Bit y in PRIVBBRx</th><th>Bit y in SECBBRx</th></tr>
<tr><td>Fetch</td><td>Secure/nonsecure</td><td>Privileged/unprivileged</td><td>-</td><td>Bus error</td></tr>
<tr><td>Read</td><td>Secure/nonsecure</td><td>Privileged/unprivileged</td><td>-</td><td>OK</td></tr>
</table>

Table 62. SECBBRx registers access when TrustZone is active (TZEN = 1) (continued)

Access type			Bit y in PRIVBBRx	Bit y in SECBBRx
Write	Secure	Privileged	x	OK
		Unprivileged	0	OK for bit y
		Unprivileged	1	WI for bit y
	Nonsecure	Privileged/unprivileged	x	WI and a flash memory register illegal access event

Table 63. PRIVBBRx registers access when TrustZone is active (TZEN = 1)

Access type			Page secure state (watermark or blocked based)	Bit y in PRIVBBRx
Fetch	Privileged/unprivileged	Secure/nonsecure	-	Bus error
Read	Privileged/unprivileged	Secure/nonsecure	-	OK for all bits
Write	Privileged	Secured	-	OK for all bits
		Nonsecure	Nonsecure	OK for bit y
		Nonsecure	Secure	WI for bit y
	Unprivileged	Secured/nonsecure	-	WI for all bits

Table 64. PRIVBBRx registers access when TrustZone is disabled (TZEN = 0)

Access type		PRIVBBRx
Fetch	Privileged/unprivileged	Bus error
Read	Privileged/unprivileged	OK
Write	Privileged	OK
	Unprivileged	WI

7.8 FLASH interrupts

Table 65. 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 65. 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	Nonsecure end of operation	Nonsecure EOP[1]	Write nonsecure EOP = 1	Nonsecure EOPIE	Yes	No
	Nonsecure operation error	Nonsecure OPERR[2]	Write nonsecure OPERR = 1	Nonsecure ERRIE	Yes	No
	ECC correction	ECCC	Write ECCC=1	ECCCIE	Yes	No

1. Secure EOP (resp. nonsecure EOP) is set only if secure EOPIE (resp. nonsecure EOPIE) is set.
2. Secure OPERR (resp. nonsecure OPERR) is set only if secure ERRIE (resp. nonsecure ERRIE) is set.

7.9 FLASH registers

7.9.1 FLASH access control register (FLASH_ACR)

Address offset: 0x000

Reset value: 0x0000 0001

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	Res.	PDREQ	LPM	Res.	Res.	PRFTEN	Res.	Res.	Res.	Res.	LATENCY[3:0]			
	rw		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.

Access to the bit can be secured by PWR LPMSEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with FLASH SPRIV or when nonsecure with FLASH NSPRIV.

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 Reserved, must be kept at reset value.

Bit 12 **PDREQ**: Flash power-down mode request

This bit requests flash to enter power-down mode. When flash enters power-down mode, this bit is cleared by hardware and the PDKEYR is locked.

This bit is write-protected with FLASH_PDKEYR.

Access to the bit can be secured by PWR LPMSEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with FLASH SPRIV or when nonsecure with FLASH NSPRIV.

0: No request for flash to enter power-down mode

1: Flash requested to enter power-down mode

Bit 11 **LPM**: Low-power read mode

This bit puts the flash memory in low-power read mode.

Access to the bit can be secured by PWR LPMSEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with FLASH SPRIV or when nonsecure with FLASH NSPRIV.

This bit cannot be written when a flash program or erase operation is busy (BSY = 1), or when the write buffer is not empty (WDW = 1). Changing this bit while a flash program or erase operation is busy (BSY = 1) is rejected.

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.

This bit can be protected against unprivileged access by FLASH NSPRIV.

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 AHB hclk1 clock period and the flash memory access time.

Access to the bit can be secured by RCC SYSCLKSEC. When secure, a nonsecure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with FLASH SPRIV or when nonsecure with FLASH NSPRIV.

0000: Zero wait state
0001: One wait state
0010: Two wait states
...
1111: Fifteen wait states

Note: Before entering Stop 1 mode software must set FLASH wait state latency to at least 1.

7.9.2 FLASH key register (FLASH_NSKEYR)

Address offset: 0x008

Reset value: 0x0000 0000

Access: one wait state; word access

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV =1 in the FLASH_PRIVCFGR register.

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 nonsecure key
The following values must be written consecutively to unlock the FLASH_NSCR1 register, allowing the flash memory nonsecure programming/erasing operations:
KEY1: 0x4567 0123
KEY2: 0xCDEF 89AB

7.9.3 FLASH secure key register (FLASH_SECKEYR)

Address offset: 0x00C

Reset value: 0x0000 0000

Access: one wait state; word access

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

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_SECCR1 register, allowing the flash memory secure programming/erasing operations:
KEY1: 0x4567 0123
KEY2: 0xCDEF 89AB

7.9.4 FLASH option key register (FLASH_OPTKEYR)

Address offset: 0x010

Reset value: 0x0000 0000

Access: one wait state; word access

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

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 LOCK bit in the FLASH_NSCR1 must be cleared before doing the unlock sequence for OPTLOCK bit. The following values must be written consecutively to unlock the FLASH_NSCR1.OPTSTRT and OBL_LAUNCH register bits concerning user option operations:
KEY1: 0x0819 2A3B
KEY2: 0x4C5D 6E7F

7.9.5 FLASH power-down key register (FLASH_PDKEYR)

Address offset: 0x018

Reset value: 0x0000 0000

Access: no wait state; word access

Access to this register can be protected by PWR LPMSEC. When secure it can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when secure by SPRIV = 1 and when nonsecure by NSPRIV in the FLASH_PRIVCFGR register.

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]**: Flash power-down key
The following values must be written consecutively to unlock the PDREQ bit in FLASH_ACR:
PDKEY_1: 0x0415 2637
PDKEY_2: 0xFAFB FCFD

7.9.6 FLASH status register (FLASH_NSSR)

Address offset: 0x020

Reset value: 0x000X 0000

Access: no wait state; word, half-word and byte access

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR 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.	PD	OEM2 LOCK	OEM1 LOCK	WDW	BSY
											r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OPTW ERR	Res.	Res.	Res.	Res.	Res.	PGS ERR	SIZ ERR	PGA ERR	WRP ERR	PROG ERR	Res.	OP ERR	EOP
		rc_w1						rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **PD**: Flash in power-down mode
This bit indicates that the flash memory is in power-down state. It is reset when flash is in normal mode or being awaken.

Bit 19 **OEM2LOCK**: OEM2 key RDP lock
This bit indicates that the OEM2 key read during the OBL is not virgin. When set, the OEM2 key RDP lock mechanism is active.

Bit 18 **OEM1LOCK**: OEM1 key RDP lock
This bit indicates that the OEM1 key read during the OBL is not virgin. When set, the OEM1 key RDP lock mechanism is active.

Bit 17 **WDW**: Nonsecure wait data to write
This bit indicates that the flash memory write buffer has been written by a secure or nonsecure 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**: Nonsecure busy
This indicates that a flash memory secure or nonsecure 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 or when modifying options in RDP level 2.. It is cleared by writing 1.
Refer to *Section 7.3.10* for full conditions of error flag setting.

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **PGSERR**: Nonsecure programming sequence error
This bit is set by hardware when programming sequence is not correct. It is cleared by writing 1.
Refer to *Section 7.3.10* for full conditions of error flag setting.

Bit 6 **SIZERR**: Nonsecure size error
This bit is set by hardware when the size of the access is a byte or half-word during a nonsecure program sequence. Only quad-word programming is allowed by means of successive word accesses. This bit is cleared by writing 1.

Bit 5 **PGAERR**: Nonsecure 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**: Nonsecure write protection error
This bit is set by hardware when a nonsecure address to be erased/programmed belongs to a write-protected part (by WRP or HDP) of the flash memory. This bit is cleared by writing 1.
Refer to *Section 7.3.10* for full conditions of error flag setting.

Bit 3 **PROGERR**: Nonsecure programming error
This bit is set by hardware when a nonsecure 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**: Nonsecure operation error

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

Bit 0 **EOP**: Nonsecure end of operation

This bit is set by hardware when one or more flash memory nonsecure operation (program/erase) has been completed successfully. This bit is set only if the nonsecure end of operation interrupts are enabled (EOPIE = 1 in FLASH_NSCR1). This bit is cleared by writing 1.

7.9.7 FLASH secure status register (FLASH_SECSR)

Address offset: 0x024

Reset value: 0x0000 0000

Access: no wait state; word, half-word and byte access

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR 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.	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.	PGS ERR	SIZ ERR	PGA ERR	WRP ERR	PROG ERR	Res.	OP ERR	EOP
								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

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 nonsecure 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 nonsecure 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 7.3.10* 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 an secure address to be erased/programmed belongs to a write-protected part (by WRP or HDP) of the flash memory. This bit is cleared by writing 1. Refer to *Section 7.3.10* 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_SECCR1). This bit is cleared by writing 1.

7.9.8 FLASH control register (FLASH_NSCR1)

Address offset: 0x028

Reset value: 0xC000 0000

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

This register is write protected with LOCK and must be unlocked by NSKEY unlock sequence.

This register can only be written when the BSY or OBL_LAUNCH are reset. Otherwise, the write access is ignored and generate a bus error and hard fault interrupt.

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	OPT LOCK	Res.	Res.	OBL_ LAUNCH	Res.	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	Res.	OPT STRT	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**
Res.	BWR	Res.	Res.	Res.	Res.	PNB[6:0]							MER	PER	PG
	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LOCK:** Nonsecure lock

This bit is set only.

When set, the FLASH_NSCR1 register write access is locked. This bit is cleared by hardware after detecting the unlock sequence in FLASH_NSKEYR.

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, the FLASH_NSCR1.OPTSRT and OBL_LAUNCH bits concerning user options write access is locked. This bit is cleared by hardware after detecting the unlock sequence in FLASH_OPTKEYR. The FLASH_NSCR1.LOCK bit must be cleared before doing the FLASH_OPTKEYR unlock sequence.

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. This bit is write-protected with OPTLOCK.

0: Option byte loading complete

1: Option byte loading requested

Note: The LSE oscillator must be disabled, LSEON = 0 and LSERDY = 0, before starting OBL_LAUNCH.

Bit 26 Reserved, must be kept at reset value.

Bit 25 **ERRIE**: Nonsecure error interrupt enable

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

0: Nonsecure OPERR error interrupt disabled

1: Nonsecure OPERR error interrupt enabled

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

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

0: Nonsecure EOP Interrupt disabled

1: Nonsecure EOP Interrupt enabled

Bits 23:18 Reserved, must be kept at reset value.

Bit 17 **OPTSTRT**: Options modification start

This bit triggers an option bytes erase and program operation when set. This bit is write-protected with OPTLOCK.. This bit is set only by software, and is cleared when the BSY bit is cleared in FLASH_NSSR.

Bit 16 **STRT**: Nonsecure operation start

This bit triggers a nonsecure erase operation when set. If MER 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 Reserved, must be kept at reset value.

Bit 14 **BWR**: Nonsecure burst write programming mode

When set, this bit selects the burst write programming mode.

Bits 13:10 Reserved, must be kept at reset value.

Bits 9:3 **PNB[6:0]**: Nonsecure page number selection
These bits select the page to erase.
0000000: page 0
0000001: page 1
...
1111111: page 127
Note that bit 9 is reserved on STM32WBA5xxE devices.

Bit 2 **MER**: Nonsecure flash mass erase
This bit triggers the flash nonsecure mass erase (all user pages) when set.

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

Bit 0 **PG**: Nonsecure programming
0: Nonsecure flash programming disabled
1: Nonsecure flash programming enabled

7.9.9 FLASH secure control register (FLASH_SECCR1)

Address offset: 0x02C

Reset value: 0x8000 0000

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

Access to this register is locked and must be unlocked by SECKEY unlock sequence.

This register can only be written when the BSY or OBL_LAUNCH are reset. Otherwise, the write access is ignored and generate a bus error and hard fault interrupt.

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

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
Res.	BWR	Res.	Res.	Res.	Res.	PNB[6:0]							MER	PER	PG
	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_SECCR1 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 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 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 MER 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 Reserved, must be kept at reset value.

Bit 14 **BWR**: Secure burst write programming mode

When set, this bit selects the burst write programming mode.

Bits 13:10 Reserved, must be kept at reset value.

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

These bits select the page to erase:
0000000: page 0
0000001: page 1
...
1111111: page 127
Note that bit 9 is reserved on STM32WBA5xxE devices.

Bit 2 **MER**: Secure flash mass erase

This bit triggers the flash secure mass erase (all 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

7.9.10 FLASH ECC register (FLASH_ECCR)

Address offset: 0x030

Reset value: 0x0000 0000

Access: no wait state; word, half-word and byte access

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

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	Res.	Res.	ADDR_ECC[19:16]			
rc_w1	rc_w1						rw		r			r	r	r	r

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

Bit 31 **ECCD**: ECC detection

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.

Bits 21:20 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 relative to flash base address, from offset 0x0 0000 to 0xF FFF0.

Note that bit 19 is reserved on STM32WBA5xxE devices.

7.9.11 FLASH operation status register (FLASH_OPSR)

Address offset: 0x034

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

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

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	Res.	Res.	ADDR_OP[19:16]			
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: Reserved
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.

Bits 21:20 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 relative to the flash base address, from offset 0x0 0000 to 0xF FFF0. Note that bit 19 is reserved on STM32WBA5xxE devices.

7.9.12 FLASH control 2 register (FLASH_NSCR2)

Address offset: 0x038

Reset value: 0x0000 0000

Access: no wait state; word, half-word and byte access

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR 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**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ES	PS
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **ES**: Erase suspend request

0: erase suspend disabled
1: erase suspend requested (enabled)

Bit 0 **PS**: Program suspend request
0: program suspend disabled
1: program suspend requested (enabled)

7.9.13 FLASH secure control 2 register (FLASH_SECCR2)

Address offset: 0x03C

Reset value: 0x0000 0000

Access: no wait state; word, half-word and byte access

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR 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**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ES	PS
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **ES**: Erase suspend request
0: erase suspend disabled
1: erase suspend requested (enabled)

Bit 0 **PS**: Program suspend request
0: program suspend disabled
1: program suspend requested (enabled)

7.9.14 FLASH option register (FLASH_OPTR)

Address offset: 0x040

Reset value: 0xXXXX XXXX

Register bits 0 to 31 are loaded with values from the flash memory at OBL.

Access: no wait state when no option bytes modification is ongoing; word, half-word and byte access

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TZEN	Res.	Res.	Res.	NBOOT0	NSWBOOT0	SRAM2_RST	SRAM2_PE	Res.	Res.	Res.	Res.	WWDG_SW	IWDG_STDBY	IWDG_STOP	IWDG_SW
rw				rw	rw	rw	rw					rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRAM1_RST	Res.	NRST_STDBY	NRST_STOP	Res.	BOR_LEV[2:0]			RDP[7:0]							
rw		rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TZEN:** Global TrustZone security enable
0: Global TrustZone security disabled
1: Global TrustZone security enabled

Bits 30:28 Reserved, must be kept at reset value.

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_PE:** SRAM2 parity check enable
0: SRAM2 parity check enabled
1: SRAM2 parity check disabled

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

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 enable selection
0: Hardware mode, independent watchdog started automatically be hardware on reset selected
1: Software mode, independent watchdog started by software command selected

Bit 15 **SRAM1_RST:** SRAM1 erase upon system reset
0: SRAM1erased when a system reset occurs
1: SRAM1 not erased when a system reset occurs

Bit 14 Reserved, must be kept at reset value.

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

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

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 nonsecure 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 for more details.

7.9.15 FLASH boot address 0 register (FLASH_NSBOOTADD0R)

Address offset: 0x044

Reset value: 0xXXXX XXXF

The option bytes are loaded with values from the flash memory at reset release.

Access: no wait state when no option bytes modification is ongoing; word, half-word and byte access

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

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]**: Nonsecure boot base address 0
The nonsecure boot memory address can be programmed to any address in the valid address range (see *Table 24: Boot space versus RDP protection*) 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 flash memory (0x0800 0000)
NSBOOTADD0[24:0] = 0x017F100: Boot from system memory bootloader (0x0BF8 8000)
NSBOOTADD0[24:0] = 0x0400200: Boot from SRAM2 on S-Bus (0x2001 0000)

Bits 6:0 Reserved, must be kept at reset value.

7.9.16 FLASH boot address 1 register (FLASH_NSBOOTADD1R)

Address offset: 0x048

Reset value: 0xXXXX XXXF

The option bytes are loaded with values from the flash memory at reset release.

Access: no wait state when no option bytes modification is ongoing; word, half-word and byte access

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

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]**: Nonsecure boot address 1
The nonsecure boot memory address can be programmed to any address in the valid address range (see *Table 24: Boot space versus RDP protection*) 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 flash memory (0x0800 0000)
NSBOOTADD1[24:0] = 0x017F100: boot from system memory bootloader (0x0BF8 8000)
NSBOOTADD1[24:0] = 0x0400200: boot from SRAM2 (0x2001 0000)

Bits 6:0 Reserved, must be kept at reset value.

7.9.17 FLASH secure boot address 0 register (FLASH_SECBOOTADD0R)

Address offset: 0x04C

Reset value: 0xXXXX XXXX

The option bytes are loaded with values from the flash memory at reset release.

Access: no wait state when no option bytes modification is ongoing; word, half-word and byte access

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

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							rs

Bits 31:7 **SECBOOTADD0[24:0]**: Secure boot base address 0

This address is only used when TZEN = 1.

The secure boot memory address can be programmed to any address in the valid address range (see *Table 24: Boot space versus RDP protection*) 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 user flash memory (0x0C00 0000)

SECBOOTADD0[24:0] = 0x01F F000: Boot from RSS system flash memory (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

This lock is only used when TZEN = 1.

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 by an RDP regression level 1 to level 0.

7.9.18 FLASH secure watermark register 1 (FLASH_SECWMR1)

Address offset: 0x050

Reset value: 0xFFXX FFXX

Register bits are loaded with values from the flash memory at OBL. Reserved bits are read as 1.

Access: no wait state when no option bytes modification is ongoing; word, half-word and byte access

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR 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.	SECWM_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.	SECWM_PSTRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

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

Bits 22:16 **SECWM_PEND[6:0]:** End page of secure area
This field contains the last page of the secure area.

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

Bits 6:0 **SECWM_PSTRT[6:0]:** Start page of secure area
This field contains the first page of the secure area.

7.9.19 FLASH secure watermark register 2 (FLASH_SECWMR2)

Address offset: 0x054

Reset value: 0xXFXX XFXX

Register bits are loaded with values from the flash memory at OBL.

Access: no wait state when no option bytes modification is ongoing; word, half-word and byte access

This register is secure, it can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HDPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP_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 **HDPEN**: Secure Hide protection area enable
0: No secure HDP area
1: Secure HDP area enabled

Bits 30:23 Reserved, must be kept at reset value.

Bits 22:16 **HDP_PEND[6:0]**: End page of secure hide protection area
This field contains the last page of the secure HDP area.

Bits 15:0 Reserved, must be kept at reset value.

7.9.20 FLASH WRP area A address register (FLASH_WRPAR)

Address offset: 0x058

Reset value: 0xXFXX FFXX

Register bits are loaded with values from the flash memory at OBL. Reserved bits are read as 1.

Access: no wait state when no option bytes modification is ongoing; word, half-word and byte access

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

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.	WRPA_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.	WRPA_PSTRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bit 31 **UNLOCK**: WPR area A unlock
0: WRPA start and end pages locked
1: WRPA start and end pages unlocked

Bits 30:23 Reserved, must be kept at reset value.

Bits 22:16 **WRPA_PEND[6:0]**: WPR area A end page
This field contains the last page of the WPR area A.
Note that bit 22 is reserved on STM32WBA5xxE devices.

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

Bits 6:0 **WRPA_PSTRT[6:0]**: WPR area A start page
This field contains the first page of the WPR area A.
Note that bit 6 is reserved on STM32WBA5xxE devices.

7.9.21 FLASH WRP area B address register (FLASH_WRPBR)

Address offset: 0x05C

Reset value: 0xXFXX FFXX

Register bits are loaded with values from the flash memory at OBL.

Access: no wait state when no option bytes modification is ongoing; word, half-word and byte access

This register is nonsecure. It can be read and written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

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.	WRPB_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.	WRPB_PSTRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bit 31 **UNLOCK**: WPR area B unlock
0: WRPB start and end pages locked
1: WRPB start and end pages unlocked

Bits 30:23 Reserved, must be kept at reset value.

Bits 22:16 **WRPB_PEND[6:0]**: WRP area B end page
This field contains the last page of the WRP area B.
Note that bit 22 is reserved on STM32WBA5xxE devices.

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

Bits 6:0 **WRPB_PSTRT[6:0]**: WRP area B start page
This field contains the first page of the WRP area B.
Note that bit 6 is reserved on STM32WBA5xxE devices.

7.9.22 FLASH OEM1 key register 1 (FLASH_OEM1KEYR1)

Address offset: 0x070

Reset value: 0x0000 0000

Access: no wait state when no option bytes modification is ongoing; word, half-word and byte access

This register is nonsecure. It can be written by both secure and nonsecure access. This register is read as zero. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

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 key least significant bytes

7.9.23 FLASH OEM1 key register 2 (FLASH_OEM1KEYR2)

Address offset: 0x074

Reset value: 0x0000 0000

Access: no wait state when no option bytes modification is ongoing; word, half-word and byte access

This register is nonsecure. It can be written by both secure and nonsecure access. This register is read as zero. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

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 key most significant bytes

7.9.24 FLASH OEM2 key register 1 (FLASH_OEM2KEYR1)

Address offset: 0x078

Reset value: 0x0000 0000

Access: no wait state when no option bytes modification is ongoing; word, half-word and byte access

This register is nonsecure. It can be written by both secure and nonsecure access. This register is read as zero. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

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 key least significant bytes

7.9.25 FLASH OEM2 key register 2 (FLASH_OEM2KEYR2)

Address offset: 0x07C

Reset value: 0x0000 0000

Access: no wait state when no option bytes modification is ongoing; word, half-word and byte access

This register is nonsecure. It can be written by both secure and nonsecure access. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

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 key most significant bytes

7.9.26 FLASH secure block based register x (FLASH_SECBBRx)

Address offset: 0x080 + 0x4 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

STM32WBA5xxE devices contain only FLASH_SECBBR1 and FLASH_SECBBR2.

Access: no wait state; word, half-word and byte access

This register is secure. It can be written only by secure access. Individual register bits can be protected against unprivileged access (refer to *Table 62*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SECBB31	SECBB30	SECBB29	SECBB28	SECBB27	SECBB26	SECBB25	SECBB24	SECBB23	SECBB22	SECBB21	SECBB20	SECBB19	SECBB18	SECBB17	SECBB16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SECBB15	SECBB14	SECBB13	SECBB12	SECBB11	SECBB10	SECBB9	SECBB8	SECBB7	SECBB6	SECBB5	SECBB4	SECBB3	SECBB2	SECBB1	SECBB0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SECBB[31:0]:** page secure/nonsecure attribution (y = 0 to 31)
Each bit is used to set one page security attribution.
0: Page (32 x (x - 1) + y) in flash not block-based secure
1: Page (32 x (x - 1) + y) in flash block-based secure

7.9.27 FLASH secure HDP control register (FLASH_SECHDPCR)

Address offset: 0x0C0

Reset value: 0x0000 0000

Access: no wait state; word, half-word and byte access

This register is secure. It can be read and written only by secure access. A nonsecure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP_ACCDIS
															rs

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

Bit 0 **HDP_ACCDIS:** Secure HDP area access disable
When set, this bit is cleared only by a system reset.
0: Access to secure HDP area granted
1: Access to secure HDP area denied (SECWMRx option bytes modification blocked, refer to *Rules for modifying specific option bytes*)

7.9.28 FLASH privilege configuration register (FLASH_PRIFCFGR)

Address offset: 0x0C4

Reset value: 0x0000 0000

Access: no wait state; word, half-word and byte access

This register is privileged write protected. It can be written only by a privileged access. All bits in this register can be read by both privileged and unprivileged, secure and nonsecure 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 nonsecure registers
0: Nonsecure flash registers can be read and written by privileged or unprivileged access.
1: Nonsecure flash registers can be read and written by privileged access only.

Bit 0 **SPRIV**: Privileged protection for secure registers
This bit is secure write protected. It can only be written by a secure privileged access when TrustZone is enabled (TZEN = 1).
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.

7.9.29 FLASH privilege block based register x (FLASH_PRIVBBRx)

Address offset: 0x0D0 + 0x4 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

STM32WBA5xxE devices contain only FLASH_PRIVBBR1 and FLASH_PRIVBBR2.

Access: no wait state; word, half-word and byte access

This register is privileged. It can be read written only by a privileged access. Individual register bits can be protected against nonsecure access (refer to *Table 63*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIVBB31	PRIVBB30	PRIVBB29	PRIVBB28	PRIVBB27	PRIVBB26	PRIVBB25	PRIVBB24	PRIVBB23	PRIVBB22	PRIVBB21	PRIVBB20	PRIVBB19	PRIVBB18	PRIVBB17	PRIVBB16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PRIVBB15	PRIVBB14	PRIVBB13	PRIVBB12	PRIVBB11	PRIVBB10	PRIVBB9	PRIVBB8	PRIVBB7	PRIVBB6	PRIVBB5	PRIVBB4	PRIVBB3	PRIVBB2	PRIVBB1	PRIVBB0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIVBB[31:0]:** page privileged/unprivileged attribution (y = 0 to 31)
Each bit is used to set one page privilege attribution in flash.
0: Page (32 x (x - 1) + y) in flash accessible by unprivileged access
1: Page (32 x (x - 1) + y) in flash only accessible by privileged access

7.9.30 FLASH register map

Table 66. 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
0x000	FLASH_ACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEP_PD	Res.	PDREQ	LPM	Res.	Res.	PRFTEN	Res.	Res.	Res.	Res.	LATENCY [3:0]			
	Reset value																		0		0	0			0					0	0	0	1
0x004	Reserved	Reserved																															
0x008	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
0x00C	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
0x010	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
0x014	Reserved	Reserved																															
0x018	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
0x01C	Reserved	Reserved																															
0x020	FLASH_NSSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PD	OEM2LOCK	OEM1LOCK	WDW	BSY	Res.	Res.	OPTWERR	Res.	Res.	Res.	Res.	Res.	PGSERR	SIZERR	PGAERR	WRPERR	PROGERR	Res.	OPERR	EOP
	Reset value												0	X	X	X	X			0						0	0	0	0	0		0	0
0x024	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
0x028	FLASH_NSCR1	LOCK	OPTLOCK	Res.	Res.	OBL_LAUNCH	Res.	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	Res.	OPTSTRT	STRT	Res.	BWR	Res.	Res.	Res.	Res.	PNB[6:0](1)							MER	PER	PG
	Reset value	1	1			0		0	0							0	0		0					0	0	0	0	0	0	0	0	0	0
0x02C	FLASH_SECCR1	LOCK	Res.	INV	Res.	Res.	Res.	ERRIE	EOPIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRT	Res.	BWR	Res.	Res.	Res.	Res.	PNB[6:0](1)							MER	PER	PG
	Reset value	1		0				0	0								0		0					0	0	0	0	0	0	0	0	0	0
0x030	FLASH_ECCR	ECCD	ECCC	Res.	Res.	Res.	Res.	Res.	ECCIE	Res.	SYSF_ECC	Res.	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
0x034	FLASH_OPSR	CODE_OP[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	SYSF_OP	Res.	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

Table 66. 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
0x038	FLASH_NSCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ES	PS
	Reset value																															0	0
0x03C	FLASH_SECCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ES	PS
	Reset value																															0	0
0x040	FLASH_OPTR	TZEN	Res.	Res.	Res.	NBOOT0	NSWBOOT0	SRAM2_RST	SRAM2_PE	Res.	Res.	Res.	Res.	WWDG_SW	IWDG_STDBY	IWDG_STOP	IWDG_SW	SRAM1_RST	Res.	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
0x044	FLASH_ NSBOOTADD0R	NSBOOTADD0[24:0]																									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							
0x048	FLASH_ NSBOOTADD1R	NSBOOTADD1[24:0]																									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							
0x04C	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
0x050	FLASH_ SECWMR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM_PSTRT[6:0]						
	Reset value										X	X	X	X	X	X	X										X	X	X	X	X	X	X
0x054	FLASH_ SECWMR2	HDPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.						
	Reset value	X									X	X	X	X	X	X	X																
0x058	FLASH_WRPAR	UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPA_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPA_PSTRT[6:0][1]						
	Reset value	X									X	X	X	X	X	X	X										X	X	X	X	X	X	X
0x05C	FLASH_WRPBR	UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPB_PEND[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRPB_PSTRT[6:0][1]						
	Reset value	X									X	X	X	X	X	X	X										X	X	X	X	X	X	X
0x070	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
0x074	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
0x078	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
0x07C	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 66. 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
0x080	FLASH_ SECBBR1	SECBB31	SECBB30	SECBB29	SECBB28	SECBB27	SECBB26	SECBB25	SECBB24	SECBB23	SECBB22	SECBB21	SECBB20	SECBB19	SECBB18	SECBB17	SECBB16	SECBB15	SECBB14	SECBB13	SECBB12	SECBB11	SECBB10	SECBB9	SECBB8	SECBB7	SECBB6	SECBB5	SECBB4	SECBB3	SECBB2	SECBB1	SECBB0
	Reset value	0	0	0	0	0	0	0	0	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	FLASH_ SECBBR2	SECBB31	SECBB30	SECBB29	SECBB28	SECBB27	SECBB26	SECBB25	SECBB24	SECBB23	SECBB22	SECBB21	SECBB20	SECBB19	SECBB18	SECBB17	SECBB16	SECBB15	SECBB14	SECBB13	SECBB12	SECBB11	SECBB10	SECBB9	SECBB8	SECBB7	SECBB6	SECBB5	SECBB4	SECBB3	SECBB2	SECBB1	SECBB0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088	FLASH_ SECBBR3(2)	SECBB31	SECBB30	SECBB29	SECBB28	SECBB27	SECBB26	SECBB25	SECBB24	SECBB23	SECBB22	SECBB21	SECBB20	SECBB19	SECBB18	SECBB17	SECBB16	SECBB15	SECBB14	SECBB13	SECBB12	SECBB11	SECBB10	SECBB9	SECBB8	SECBB7	SECBB6	SECBB5	SECBB4	SECBB3	SECBB2	SECBB1	SECBB0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	FLASH_ SECBBR4(2)	SECBB31	SECBB30	SECBB29	SECBB28	SECBB27	SECBB26	SECBB25	SECBB24	SECBB23	SECBB22	SECBB21	SECBB20	SECBB19	SECBB18	SECBB17	SECBB16	SECBB15	SECBB14	SECBB13	SECBB12	SECBB11	SECBB10	SECBB9	SECBB8	SECBB7	SECBB6	SECBB5	SECBB4	SECBB3	SECBB2	SECBB1	SECBB0
	Reset value	0	0	0	0	0	0	0	0	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 to 0x0BC	Reserved	Reserved																															
0x0C0	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.	Res.	HDP1_ACCDIS
	Reset value																																0
0x0C4	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
0x0C8	Reserved	Reserved																															
0x0D0	FLASH_ PRIVBBR1	PRIVBB31	PRIVBB30	PRIVBB29	PRIVBB28	PRIVBB27	PRIVBB26	PRIVBB25	PRIVBB24	PRIVBB23	PRIVBB22	PRIVBB21	PRIVBB20	PRIVBB19	PRIVBB18	PRIVBB17	PRIVBB16	PRIVBB15	PRIVBB14	PRIVBB13	PRIVBB12	PRIVBB11	PRIVBB10	PRIVBB9	PRIVBB8	PRIVBB7	PRIVBB6	PRIVBB5	PRIVBB4	PRIVBB3	PRIVBB2	PRIVBB1	PRIVBB0
	Reset value	0	0	0	0	0	0	0	0	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	FLASH_ PRIVBBR2	PRIVBB31	PRIVBB30	PRIVBB29	PRIVBB28	PRIVBB27	PRIVBB26	PRIVBB25	PRIVBB24	PRIVBB23	PRIVBB22	PRIVBB21	PRIVBB20	PRIVBB19	PRIVBB18	PRIVBB17	PRIVBB16	PRIVBB15	PRIVBB14	PRIVBB13	PRIVBB12	PRIVBB11	PRIVBB10	PRIVBB9	PRIVBB8	PRIVBB7	PRIVBB6	PRIVBB5	PRIVBB4	PRIVBB3	PRIVBB2	PRIVBB1	PRIVBB0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D8	FLASH_ PRIVBBR3(2)	PRIVBB31	PRIVBB30	PRIVBB29	PRIVBB28	PRIVBB27	PRIVBB26	PRIVBB25	PRIVBB24	PRIVBB23	PRIVBB22	PRIVBB21	PRIVBB20	PRIVBB19	PRIVBB18	PRIVBB17	PRIVBB16	PRIVBB15	PRIVBB14	PRIVBB13	PRIVBB12	PRIVBB11	PRIVBB10	PRIVBB9	PRIVBB8	PRIVBB7	PRIVBB6	PRIVBB5	PRIVBB4	PRIVBB3	PRIVBB2	PRIVBB1	PRIVBB0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0DC	FLASH_ PRIVBBR4(2)	PRIVBB31	PRIVBB30	PRIVBB29	PRIVBB28	PRIVBB27	PRIVBB26	PRIVBB25	PRIVBB24	PRIVBB23	PRIVBB22	PRIVBB21	PRIVBB20	PRIVBB19	PRIVBB18	PRIVBB17	PRIVBB16	PRIVBB15	PRIVBB14	PRIVBB13	PRIVBB12	PRIVBB11	PRIVBB10	PRIVBB9	PRIVBB8	PRIVBB7	PRIVBB6	PRIVBB5	PRIVBB4	PRIVBB3	PRIVBB2	PRIVBB1	PRIVBB0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0E0 to 0x3FC	Reserved	Reserved																															

1. Bit [6] is reserved on STM32WBA5xxE devices.

2. Register reserved on STM32WBA5xxE devices.

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

8 Instruction cache (ICACHE)

8.1 ICACHE introduction

The instruction cache (ICACHE) is introduced on the C-AHB code bus of the Cortex®-M33 processor, to improve performance when fetching instructions and data from internal memories.

Some specific features, like dual master ports, hit-under-miss, and critical-word-first refill policy, result in close to zero-wait-state performance in most use cases.

8.2 ICACHE main features

The main features of ICACHE are listed 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, with 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, the ICACHE assigns fast traffic (addressing flash memories) to the AHB master1 port, and slow traffic (addressing SRAMs) to the AHB master2 port, thus preventing processor stalls on line refills from SRAMs. This allows ISR (interrupt service routine) fetching on internal flash memory to take place in parallel with a cache line refill from SRAMs.
  - 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 SRAM memories to the code region, for execution from the C-AHB code interface)
- Replacement and refill
  - pLRU-t (pseudo-least-recently-used, based on binary tree) replacement policy, 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
  The 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, noninterruptible

8.3 ICACHE implementation

Table 67. 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

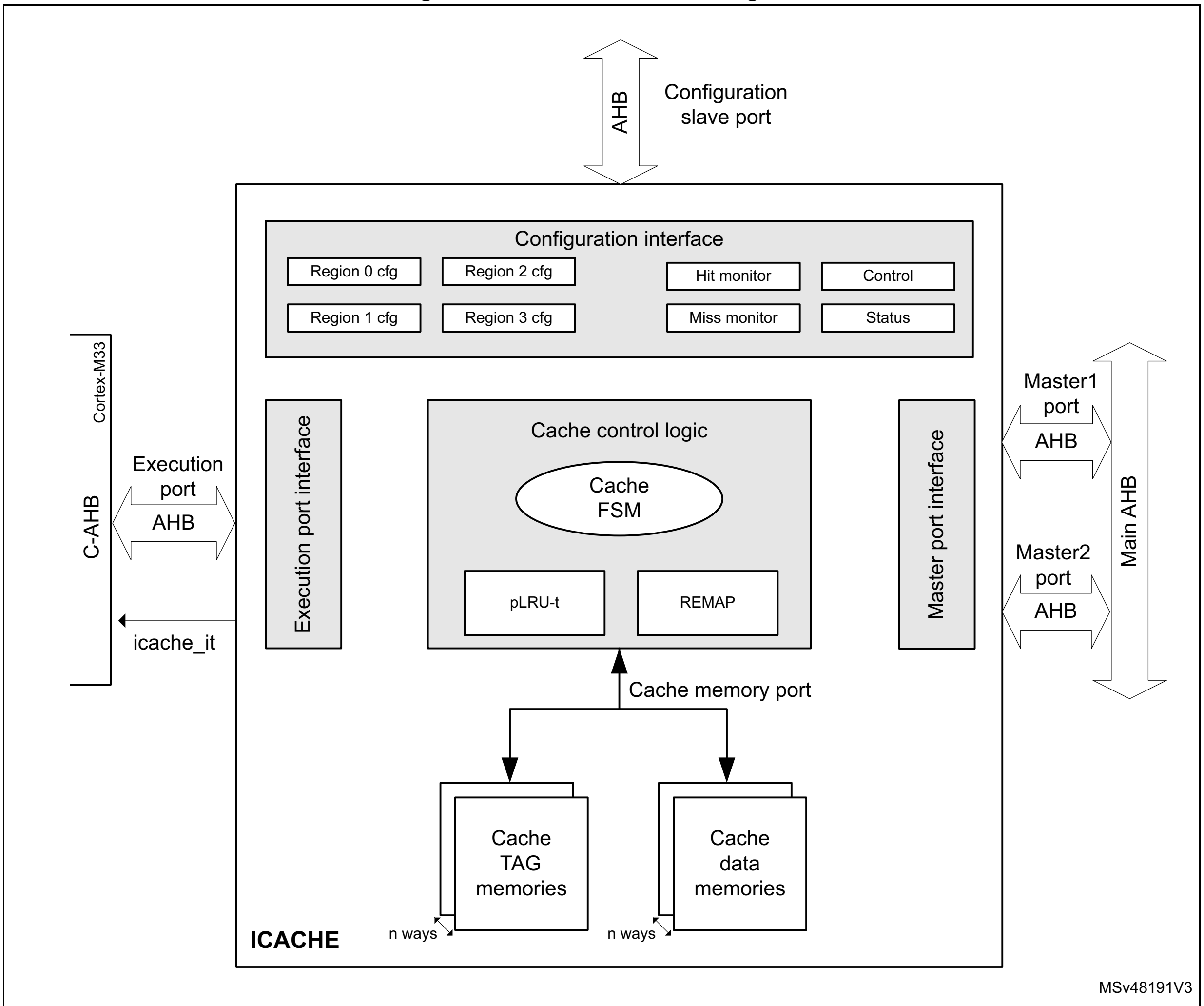
The purpose of this block is to cache instruction fetches or instruction memories loads, coming from the processor. As such, the ICACHE manages only cacheable read transactions, and does not manage cacheable write transactions.

The noncacheable transactions (both read and write) bypass the ICACHE.

For the error management purpose, if a write cacheable transaction is presented (this happens only in case of bad software programming), the ICACHE sets an error flag and, if enabled, raises an interrupt to the processor.

8.4.1 ICACHE block diagram

Figure 18. ICACHE block diagram



8.4.2 ICACHE reset and clocks

The ICACHE is clocked on the Cortex®-M33 C-AHB bus clock.

When the ICACHE reset signal is released, a cache invalidate procedure is automatically launched, making the ICACHE busy (ICACHE_SR = 0x0000 0001).

When this procedure is finished:

- the 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 longer busy)
- the ICACHE is disabled: the EN bit in ICACHE_CR holds its reset state (= 0).

Note: When disabled, the ICACHE is bypassed, except the remapping mechanism that is still functional: slave input requests (remapped or not) are forwarded to the master port(s).

8.4.3 ICACHE TAG memory

The ICACHE TAG memory contains:

- address tags indicating 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 the ICACHE reset is released
- when the cache is disabled, by setting EN = 0 in ICACHE_CR (by software)
- when executing an ICACHE invalidate command, by setting CACHEINV = 1 in ICACHE_CR (by software)

When a cacheable transaction is received at the execution input port, its AHB address (HADDR_in) is split into the following fields (see *Table 68* for B and W definitions):

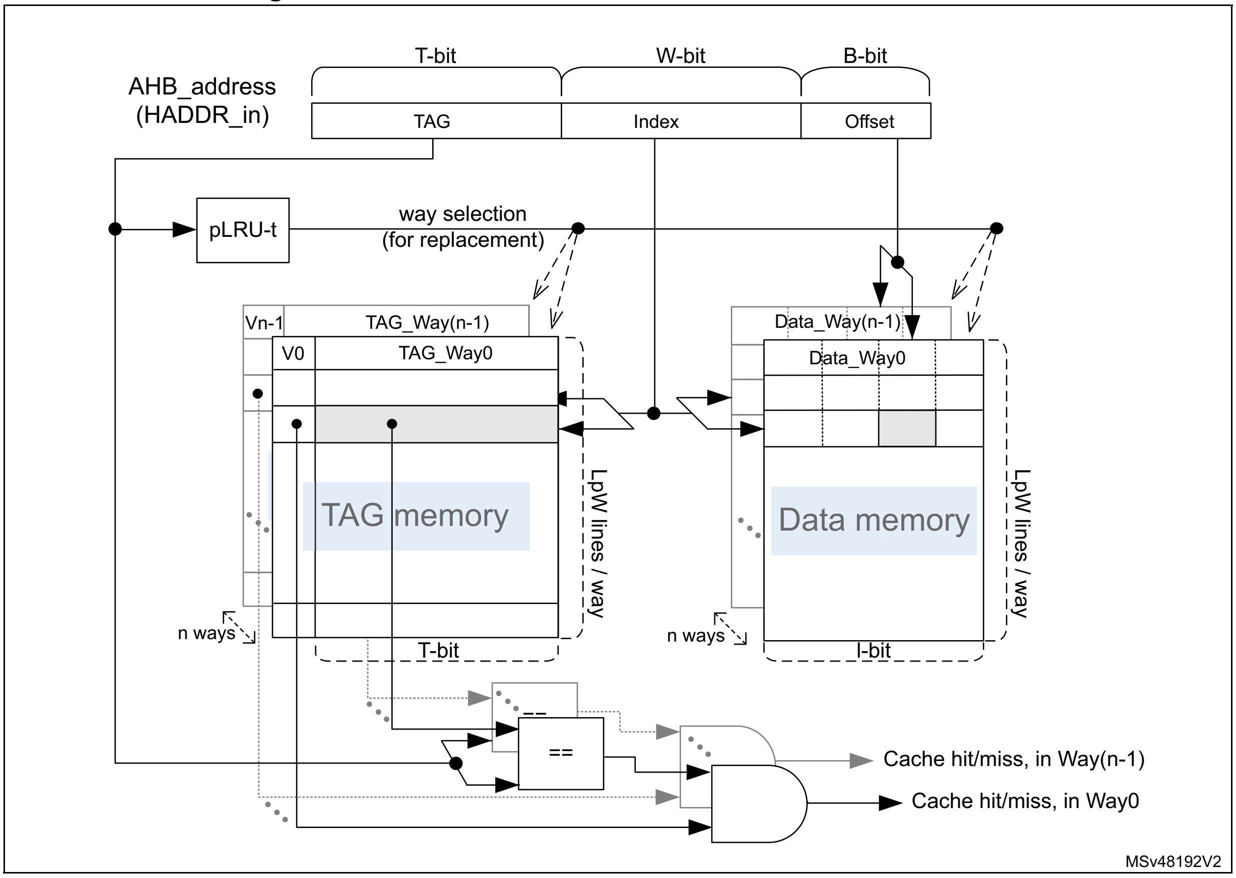
- HADDR_in[B-1:0]: address byte offset, indicates which byte to select inside a cache line.
- HADDR_in[B+W-1:B]: address way index, indicates which cache line to select inside each way.
- HADDR_in[31:B+W]: tag address, to be compared to the TAG memory address to check if the requested data is already available (meaning valid) inside the ICACHE.

The following table gives a summary of the ICACHE main parameters for TAG memory dimensioning. *Figure 19* shows the functional view of TAG and data memories, for an n-way set associative ICACHE.

Table 68. 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	LpW = s / (n x L) lines / way	256 lines / way
Address byte offset size	B = $\log_2$(L) bit	4-bit
Address way index size	W = $\log_2$(LpW) bit	8-bit
TAG address size	T = (32 - W - B) bit	20-bit

Figure 19. ICACHE TAG and data memories functional view



8.4.4 Direct-mapped ICACHE (1-way cache)

The default configuration (at reset) is an n-way set associative cache (WAYSEL = 1 in ICACHE_CR), but the user can configure the ICACHE as direct mapped by writing WAYSEL = 0 (only possible when the cache is disabled, EN = 0 in ICACHE_CR).

The following table gives a summary of ICACHE main parameters for TAG memory when the direct-mapped cache operating mode is selected.

Table 69. 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 = $\log_2$(L) bit	4-bit
Address way index size	W = $\log_2$(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 the 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*): only one way (the unique one) is possible for any data refill.

8.4.5 ICACHE enable

To activate the ICACHE, the EN bit in ICACHE_CR must be set to 1.

When the ICACHE is disabled, it is bypassed and all transactions are copied from the slave to the master ports in the same clock cycle.

It is recommended to initialize or modify the main memory content (region to be later cached) with the ICACHE disabled, and to enable the ICACHE only when this region remains unchanged (an enabled ICACHE detects cacheable write transactions as errors).

To ensure performance determinism, it is recommended to wait for the end of a potential cache invalidate procedure before enabling the ICACHE. This procedure occurs when the hardware reset signal is released, when CACHEINV is set, or when EN is cleared in ICACHE_CR. During the procedure, BUSYF is set in ICACHE_SR, and once finished, BUSYF is cleared and BSYENDF is set in the same register (raising the ICACHE interrupt if enabled on such a busy end condition).

The software must test BUSYF and/or BSYENDF values before enabling the ICACHE. Else, if the ICACHE is enabled before the end of an invalidate procedure, any cache access (while BUSYF = 1) is treated as noncacheable, and its performance depends on the main memory access time.

The address remapping is performed, whether the ICACHE is enabled or not, if the input transaction address falls into the memory regions defined and enabled in ICACHE_CRRx (see *Figure 20*).

The ICACHE is, by default, disabled at boot.

8.4.6 Cacheable and noncacheable traffic

The ICACHE is developed for the Cortex®-M33 core. It is placed on the C-AHB bus, and thus caches the code memory region, ranging from address 0x0000 0000 to 0x1FFF FFFF of the memory map.

To make other memory regions cacheable, the ICACHE supports a memory region remapping feature, used to define up to four SRAM regions, whose addresses have an alias in the code region. Addressing these memory regions through their code alias address allows the memory request to be routed to the C-AHB bus, and to be managed by the ICACHE.

Any SRAM space physically mapped at an address in range [0x2000 0000:0x3FFF FFFF] can be aliased with an address in range [0x0A00 0000:0x0AFF FFFF] or [0x0E00 0000:0x0EFF FFFF].

For a given memory request in the code region, the 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 noncacheable traffic, and when the cache is disabled.

Additional details on address remapping are provided in *Section 8.4.7*.

An incoming memory request to the ICACHE is defined as cacheable according to its AHB transaction memory lookup attribute, as shown in *Table 70*. This AHB attribute depends on the MPU (memory protection unit) programming for the addressed region.

Table 70. ICACHE cacheability for AHB transaction

AHB lookup attribute	Cacheability
1	Cacheable
0	Noncacheable

In the case of a noncacheable access (read or write), the ICACHE is bypassed. 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 the case of a cacheable access, the ICACHE behaves as explained in *Section 8.4.8*.

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.

Table 71 summarizes programmable configurations of various memories.

Table 71. Memory configurations

Memory	Cacheable (MPU programming)	Remapped in the ICACHE (ICACHE_CRRx programming)
Flash	Yes or no	Not required
SRAM	Yes (not recommended)	Required
	No	Required if the user wants code in SRAM fetched on C-AHB bus (else on S-AHB bus)

8.4.7 Address remapping

The ICACHE allows an alias address to be defined in the code region for up to four SRAM memory regions.

The address remapping is applied on the code alias address, transforming it into the destination SRAMphysical address.

The remapping operation is fully software configurable by programming ICACHE_CRRx (x = 0 to 3, number of remapped regions). This programming can be done only when the ICACHE is disabled.

Each region x can be individually enabled with REN in ICACHE_CRRx. Once enabled, the remap operation occurs even if the ICACHE is disabled, or if the transaction is noncacheable.

Remap regions can have different size: each can be programmed in RSIZE of its ICACHE_CRRx. The size of each region is a power of two multiple of range granularity (2 Mbytes), with a minimum of 2 Mbytes, and a maximum of 128 Mbytes.

The address remapping mechanism is based on the matching of an incoming AHB address (HADDR_in) with a given code subregion base-address, and the modification of this address into its (remapped) 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 subregion base-address programmed in BASEADDR of ICACHE_CRRx.
  - RI defines the number of significant bits to consider. RI = $\log_2$ (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 the region x is enabled, the master port output AHB address (HADDR_out) is composed by concatenating the two below parts:
  - REMAPADDR[31:RI] field of ICACHE_CRRx as MSBs
  - HADDR_in[RI-1:0] as LSBs

Figure 20 describes the matching and the output address generation.

Figure 20. ICACHE remapping address mechanism

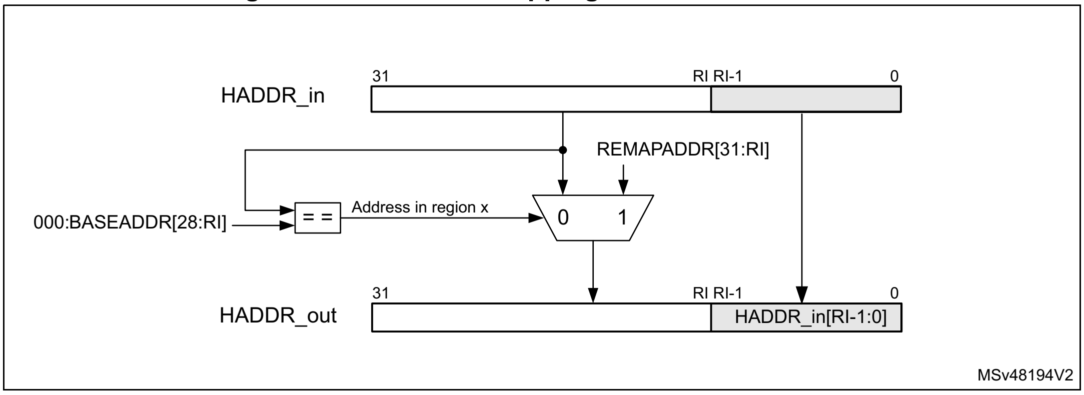


Table 72 summarizes all possible configurations of BASEADDR and REMAPADDR sizes (number of significant MSBs) in ICACHE_CRRx, depending upon RSIZE.

Table 72. ICACHE remap region size, base address, and remap address

Region size (Mbytes)	Base address size (MSBs)	Remap address (MSBs)
2	8	11
4	7	10
8	6	9
16	5	8
32	4	7
64	3	6
128	2	5

Care must be taken while programming BASEADDR and REMAPADDR in ICACHE_CRRx: if the programmed value is bigger than expected (number of MSBs, see *Table 72*), the unneeded extra LSBs are ignored.

and Typical remapping example: a 2-Mbyte SRAM region physically located in the address range [0x2000 0000:0x201F FFFF], remapped in the code section range [0x1000 0000:0x101F FFFF]:

- REMAPADDR[31:21] = 0x100
- BASEADDR[28:21] = 0x80
- HADDR_in[31:21] is compared to 000:BASEADDR[28:21].

If the comparison matches:

- HADDR_out[31:21] gets REMAPADDR[31:21] (in place of HADDR_in[31:21])
- HADDR_out[20:0] gets HADDR_in[20:0]

The software can program the kind of AHB burst generated by the ICACHE master ports on the bus matrix (for cache line refill), by setting HBURST in ICACHE_CRRx with:

- WRAP for remapped SRAMs 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

Note: Coherency is needed when programming the SAU (secure attribution unit) and the MPU (memory protection unit) attributes for both the SRAMs regions and their aliased code subregions.

8.4.8 Cacheable accesses

When the ICACHE receives a cacheable transaction from the Cortex®-M33, it checks if the address requested is present in its TAG memory, and if the corresponding cache line is valid.

There are then three alternatives:

- The address is present inside the TAG memory, the cache line is valid: **cache hit**, the data is read from the cache and provided to the processor in the same cycle.
- The address is not present in the TAG memory: **cache miss**, the data is read from the main memory and provided to the processor, and a cache line refill is performed.

  The critical-word-first policy ensures minimum wait cycles for the processor, since read data can be provided while the cache still performs a cache line refill (associated latency is the latency of fetching one word from the main memory).

  The burst generated on the ICACHE master bus is WRAPw (w being the cache line width, in words) if no address remap occurs. If an address remap occurs, the kind of burst depends upon the HBURST programmed in corresponding ICACHE_CRRx.

  The AHB transaction attributes are also propagated to the main AHB bus matrix on the master port selected for the line refill.
- The address is not present in TAG memory, but belongs to the refill burst from the main memory currently ongoing: **cache hit** (hit-under-miss feature).

  This happens during cache-line refill. The ICACHE can provide the requested data as soon as data are available at its master interface, thus avoiding a miss (fetching data from the main memory).

In the case of cache refill (due to cache miss), the 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 that points, for each index, on the way candidate for the next refill.

If 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

The ICACHE implements a dual-port AHB master on the main AHB bus matrix: master1 and master2 ports. This is used to split the traffic going to different destination memories.

The nonremapped traffic goes systematically to master1 port. The remapped traffic can be routed on the master2 port by programming MSTSEL in ICACHE_CRRx (on a region basis).

The code can typically be fetched as follows:

- internal flash memory on master1 port (Fast bus)
- SRAM on master2 port (Slow bus)

For systems not implementing external memories, the traffic to the internal flash memory can be decoupled 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

The ICACHE implements an Armv8-M TrustZone.

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 CACHEINV in the ICACHE_CR register.

When CACHEINV = 1, the ICACHE control logic sets the BUSYF flag in ICACHE_SR and launches the invalidate cache operation, resetting each TAG valid bit to 0 (one valid bit per cache line). CACHEINV is automatically cleared.

Once the invalidate operation is finished (all valid bits reset to 0), the ICACHE automatically clears BUSYF, and sets BSYENDF 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

The ICACHE provides the following monitors for performance analysis:

- The 32-bit hit monitor counts the cacheable AHB-transactions on the slave cache port that hits the ICACHE content.

  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 the cache master port) (see *Section 8.4.8*).
- The 16-bit miss monitor counts the cacheable AHB-transactions on the slave cache port that misses the ICACHE content.

  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, these 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 HITMEN in ICACHE_CR.
- Reset the hit monitor by setting HITMRST in ICACHE_CR.
- Enable/stop the miss monitor through MISSMEN in ICACHE_CR.
- Reset the miss monitor by setting MISSMRST in ICACHE_CR.

To reduce power consumption, these monitors are disabled (stopped) by default.

8.4.13 ICACHE boot

The ICACHE is disabled (EN = 0 in ICACHE_CR) at boot.

The code remapping at boot is not needed for a Cortex®-M33 as it implements the VTOR (vector tables) that allows a boot start address definition different than 0x0.

Once the boot is finished, the ICACHE can be enabled (software setting EN = 1 in CACHE_CR).

8.5 ICACHE low-power modes

At device level, using the 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.

Applications with lower performance profile (in terms of hit ratio) and stringent power consumption constraints may benefit from the lower power consumption of an ICACHE configured as direct mapped. This single-way cache configuration is obtained by programming WAYSEL = 0 in ICACHE_CR (see *Figure 19*). The power consumption is reduced by accessing, for each request, only the necessary cut of TAG and data memories. The cache effect still improves fetch performance, even if for most code execution, it is less efficient than with an n-way set associative cache mode.

8.6 ICACHE error management and interrupts

If an unsupported cacheable write request is detected (functional error), the ICACHE generates an error by setting the ERRF flag in ICACHE_SR. 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, the ICACHE sets the BSYENDF flag in ICACHE_SR. An interrupt is generated if the corresponding interrupt enable bit is set (BSYENDIE = 1 in ICACHE_IER).

All ICACHE interrupt sources raise the same and unique interrupt signal, icache_it, and then use the same interrupt vector.

Table 73. ICACHE interrupts

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method
ICACHE	Functional error	ERRF in ICACHE_SR	ERRIE in ICACHE_IER	Set CERRF to 1 in ICACHE_FCR
	End of busy state (invalidate finished)	BSYENDF in ICACHE_SR	BSYENDIE in ICACHE_IER	Set CBSYENDF to 1 in ICACHE_FCR

The ICACHE also propagates all AHB bus errors (such as security issues, address decoding issues) from the 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: release the cache miss monitor reset (needed to enable the counting)
1: reset cache miss monitor

Bit 18 **HITMRST**: hit monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
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 + 0x4 * x, (x = 0 to 3)

Reset value: 0x0000 0200

Define an alias address in the 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 74. ICACHE 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	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
0x018-0x01C	Reserved	Reserved																															
0x020	ICACHE_CRR0	HBURST	Res.	Res.	MSTSEL	Res.	Res.	REMAPADDR[31:21]										REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	Res.	BASEADDR[28:21]						
	Reset value	0			0			0	0	0	0	0	0	0	0	0	0	0				0	0	1			0	0	0	0	0	0	0
0x024	ICACHE_CRR1	HBURST	Res.	Res.	MSTSEL	Res.	Res.	REMAPADDR[31:21]										REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	Res.	BASEADDR[28:21]						
	Reset value	0			0			0	0	0	0	0	0	0	0	0	0	0				0	0	1			0	0	0	0	0	0	0
0x028	ICACHE_CRR2	HBURST	Res.	Res.	MSTSEL	Res.	Res.	REMAPADDR[31:21]										REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	Res.	BASEADDR[28:21]						
	Reset value	0			0			0	0	0	0	0	0	0	0	0	0	0				0	0	1			0	0	0	0	0	0	0
0x02C	ICACHE_CRR3	HBURST	Res.	Res.	MSTSEL	Res.	Res.	REMAPADDR[31:21]										REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	Res.	BASEADDR[28:21]						
	Reset value	0			0			0	0	0	0	0	0	0	0	0	0	0				0	0	1			0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

9 Radio system

9.1 Introduction

The 2.4 GHz RADIO is ultra-low power, operating in the 2.4 GHz ISM band. It provides Bluetooth® LE 1 Mbps coded, 1 Mbps, and 2 Mbps non-coded GFSK, and IEEE802.15.4 chip rate 2 Mchip/s, spreading mode DSSS, data rate 125 and 250 kbps O-QPSK-C modulation.

The 2.4 GHz RADIO is compliant with the Bluetooth® LE, Thread, and Zigbee® specifications, and radio regulations including ETSI EN 300 328, EN 300 440, EN 301 489-17,
ARIB STD-T66, FCC CFR47 part 15 section 15.205, 15.209, 15.247, and 15.249, IC RSS-139, and RSS-210.

9.2 Main features

- Radio protocol:
  - Bluetooth® LE
  - IEEE802.15.4
  - Proprietary protocols
  - Concurrent mode
- Bluetooth® LE features:
  - PHY support: 1 Mbps uncoded, 500 kbps and 125 kbps coded, 2 Mbps
  - Device privacy and network privacy modes
  - Anonymous device address types
  - Advertising extension PDUs
  - Advertising channel index
  - Periodic advertising synchronous transfer
  - High duty cycle, nonconnectable advertising
  - Channel selection algorithm #2
  - Angle of arrival (AoA), angle of departure (AoD)
  - Up to 20 connections in any role in addition to advertiser and scanner roles
  - Audio connected isochronous streams
  - Audio broadcast isochronous streams
- IEEE 802.15.4 features:
  - Beacon management
  - 16-bit short and 64-bit IEEE addressing modes
  - PAN formation along with association and disassociation
  - Full handshake protocol for transfer reliability, frame validation, and acknowledgment frame delivery
  - IEEE802.15.4 2020 MAC for non-beaconed PANs
- External PA
- Packet traffic arbitration

9.3 2.4 GHz RADIO implementation

Table 75. 2.4 GHz RADIO implementation

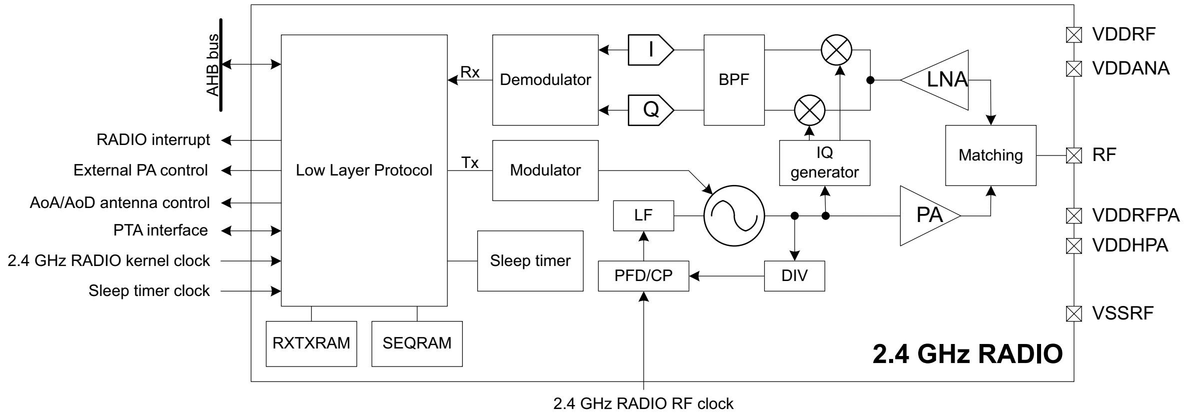
Feature(1)	STM32WBA55xx	STM32WBA54xx	STM32WBA52xx	STM32WBA50xx
Bluetooth® LE AoA/AoD	X	X	-	-
External PA	X	X	-	-
Packet traffic arbitration	X	X	-	-
IEEE802.15.4	X	X	-	-

1. X = supported

9.4 Functional description

9.4.1 Block diagram

Figure 21. Radio system block diagram



VDDANA is only available from packages with SMPS, on LDO packages VDDANA is double bonded with VDDRF.
In QFN packages VSSRF is connected to exposed pad.

Note: Bluetooth® LE AoA/AoD, external PA control, and PTA interface are available only on STM32WBA54/55xx devices. VDDANA and VSSRF are available only on STM32WBA55xx devices.

9.4.2 Pins and internal signals

Table 76. Input / output pins

Pin name	Signal type	Description
RF	RF	2.4 GHz RF input output
RADIO interrupt	Output	2.4 GHz RADIO interrupt to the CPU
External PA control	Output	External PA control

Table 76. Input / output pins (continued)

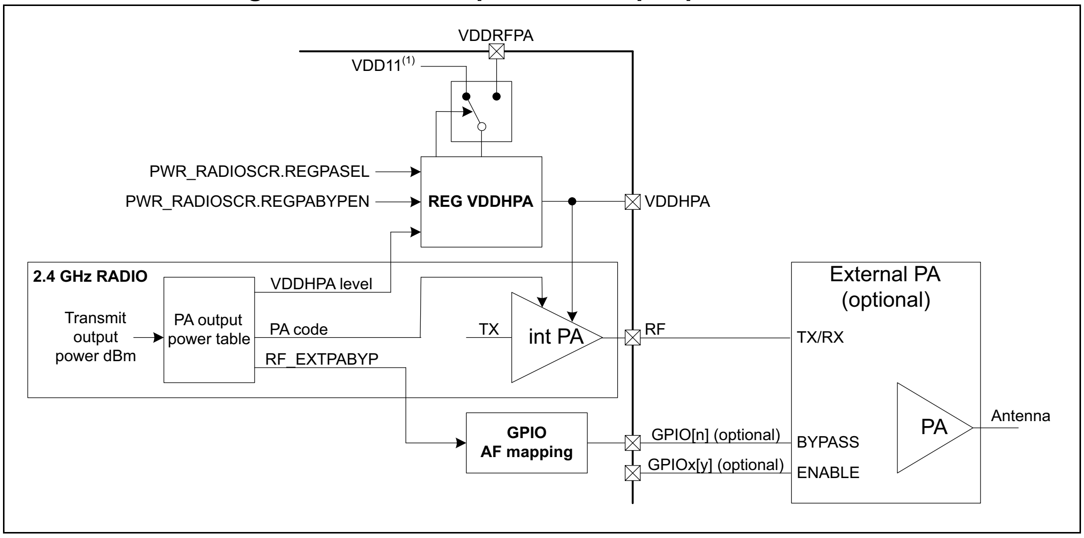
Pin name	Signal type	Description
AoA/AoD antenna control	Output	Angle of arrival/departure antenna area control
PTA interface	Input/Output	Packet traffic arbitration control

9.4.3 Transmit output power

The 2.4 GHz RADIO includes an internal PA. The transmit output power can be increased using an additional external PA. External PA control signals BYPASS can be selected on GPIOs (see the datasheet), and the signal ENABLE can be mapped on any free GPIO by application software.

Figure 22 shows the transmit path and output power control.

Figure 22. Transmit path and output power control



1. VDD11 must be selected only when supplied from device SMPS.

The transmit output power (in dBm) is dependent upon the V_{DDHPA} voltage level and the PA code, and, when the application supports an external PA, an external PA bypass indication. The values for these parameters are defined in a PA output power table available to the link layer software.

Table 77. PA output power table format

V_{DDHPA} level	PA code	BYPASS
0x0 = 0.90 V 0x1 = 0.95 V 0x2 = 1.0 V 0x3 = 1.1 V ... 0xE = 2.2 V 0xF = 2.3 V	0 to 25 dec.	0 or 1

The internal PA maximum transmit output power is depended upon the V_{DDHPA} voltage level and the PA code. The maximum V_{DDHPA} voltage level is dependent upon the application (VDDRFPA pin) supply scheme, see Table 78 for details.

The REGPASEL and REGPABYPEN allow the REG VDDHPA regulator input voltage to be controlled.

Table 78. 2.4 GHz RADIO supply configuration

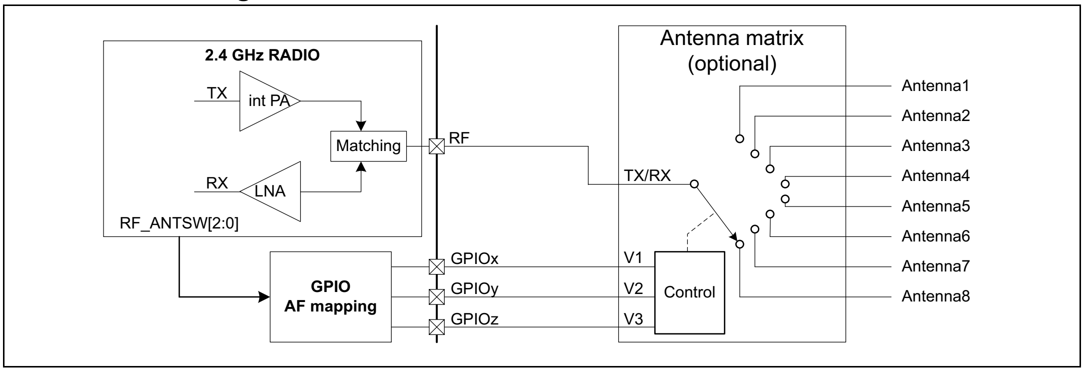
V_{DDHPA} level	Minimum V_{DDRFPA}	Internal PA max transmit output power
1.0 V	1.2 V(1)	+ 3 dBm
1.2 V	1.71 V	+ 6 dBm
1.5 V	1.8 V	+ 8 dBm
2.3 V	2.6 V	+ 10 dBm

1. Only on STM32WBA55xx devices when SMPS is used.

9.4.4 Bluetooth® LE AoA and AoD

Up to eight antennas are supported by means of the coded RF_ANTSW[2:0] signal bus available through GPIO alternate function for Bluetooth® LE AoA when receiving the constant tone extension of a packet, or for AoD while transmitting the constant tone extension of a packet.

Figure 23. Bluetooth® LE AoA/AoD antennas control



9.4.5 RXTX data SRAM access

The Link Layer software must put the 2.4 GHz RADIO in Sleep mode to let the host application access the 2.4 GHz RADIO RXTXRAM. When the access is no longer needed, the host application must request the Link Layer software to put the 2.4 GHz RADIO in DeepSleep mode.

To access the RXTX data SRAM, both the 2.4 GHz RADIO bus clock and the kernel clock must be active and ready.

The 2.4 GHz RADIO modes depend upon the host application request, and upon the Link Layer scheduled radio activity.

9.5 Low-power modes

The 2.4 GHz RADIO supports autonomous operation down to Stop 0 mode voltage scaling range 1, and the 2.4 GHz RADIO sleep timer remains operational down to Standby retention mode.

The 2.4 GHz RADIO in autonomous mode is able to transmit and receive packets over the air for the radio connection.

The 2.4 GHz RADIO sleep timer allows to maintain radio connections when the radio is in low power mode,for example between Bluetooth® LE connection events.

Table 79. Effect of low-power modes on the 2.4 GHz RADIO

Mode	Description
Sleep	No effect. 2.4 GHz RADIO and 2.4 GHz RADIO sleep timer interrupts cause the device to exit Sleep mode. Radio connection are maintained.
Stop 0 voltage range 1	The 2.4 GHz RADIO will autonomously transmit and receive packets, 2.4 GHz RADIO and 2.4 GHz RADIO sleep timer interrupts cause the device to exit Stop 0 mode. Radio connections are maintained.
Stop 0 voltage range 2 and Stop 1	The 2.4 GHz RADIO full configuration is retained but is no longer capable to transmit and receive packets, The 2.4 GHz RADIO sleep timer interrupt cause the device to exit Stop mode. Radio connections are maintained during Bluetooth® LE connection intervals.
Standby retention	The 2.4 GHz RADIO is powered down. The minimum configuration information is retained even if it is no longer capable to transmit and receive packets, The 2.4 GHz RADIO sleep timer interrupt cause the device to exit Stop mode. Radio connections are maintained during Bluetooth® LE connection intervals.
Standby	The 2.4 GHz RADIO and 2.4 GHz RADIO sleep timer are powered down. Radio connections are lost.

10 PTA converter (PTACONV)

10.1 PTACONV introduction

The PTA converter is used to convert the PTA information coming from the 2.4 GHz RADIO into the PTA protocol used on the PTA controller GPIOs.

This IP is available only on STM32WBA55xx and STM32WBA54xx devices.

10.2 PTACONV main features

- Based on IEEE802.15.2 standard
- Supports both grant and deny signaling
- Supports from 1- to 4-wire protocols
- Programmable transmit receive PTA_STATUS polarity
- Programmable priority polarity
- Programmable grant polarity
- Programmable active polarity
- Programmable PTA_ACTIVE timing
- Programmable PTA_STATUS time-multiplexed priority timing
- Programmable transmit packet abort

10.3 PTACONV functional description

The PTA converter adapts the PTA protocol to the various external PTA controllers. It supports a grant or deny signaling on the PTA_GRANT signal, and the following protocols:

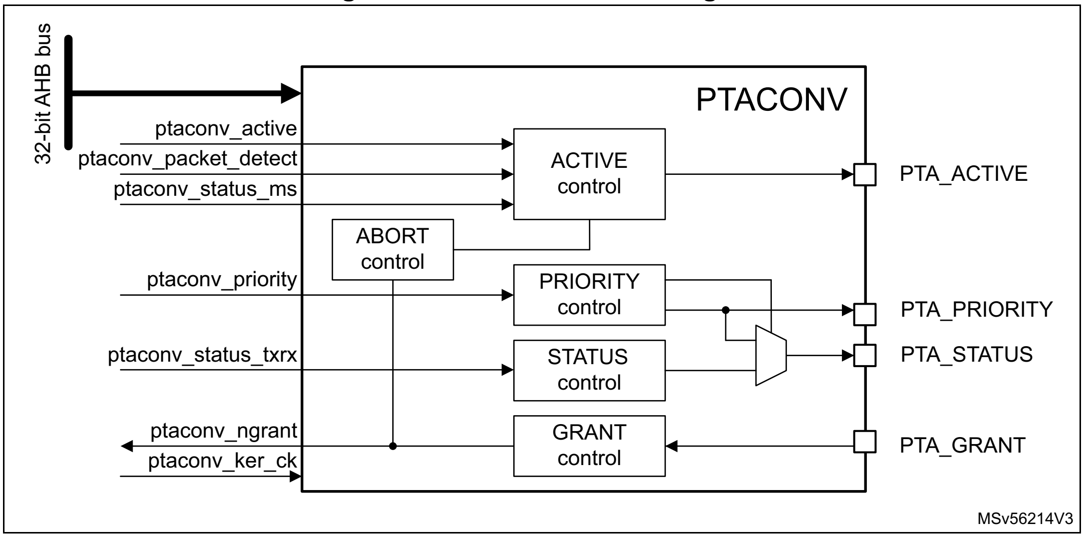
- 1-wire PTA_ACTIVE
- 2-wire PT_ACTIVE and PTA_GRANT
- 3-wire PTA_ACTIVE, PTA_GRANT and time-multiplexed PTA_STATUS
- 4-wire PTA_ACTIVE, PTA_GRANT, PTA_PRIORITY and PTA_STATUS

When the request to cease an ongoing transmission occurs near the end of the 2.4 GHz RADIO packet, PTA_ACTIVE is deactivated at the end of the packet, before the expiration of T4.

10.3.1 PTACONV block diagram

Figure 24 shows the PTA converter block diagram and the interface to the 2.4 GHz RADIO.

Figure 24. PTACONV block diagram



The PTACONV is clocked by the 2.4 GHz RADIO kernel clock, independently from the hclk for the AHB interface used to access the PTACONV configuration registers.

10.3.2 PTACONV pins and internal signals

Table 80. 2.4 PTACONV input/output pins

Pin name	Signal type	Description
PTA_ACTIVE	Output	PTA 2.4 GHz RADIO packet activation request
PTA_PRIORITY	Output	PTA 2.4 GHz RADIO packet priority
PTA_STATUS	Output	PTA 2.4 GHz RADIO packet type
PTA_GRANT	Input	PTA grant medium to 2.4 GHz RADIO

Table 81. PTACONV internal input/output signals

Pin name	Signal type	Description
ptaconv_active	Input	2.4 GHz RADIO packet activation request
ptaconv_priority	Input	2.4 GHz RADIO packet priority
ptaconv_status_txrx	Input	2.4 GHz RADIO packet type Tx or Rx
ptaconv_status_ms	Input	2.4 GHz RADIO packet type maser or slave
ptaconv_packet_detect	Input	2.4 GHz RADIO packet detect
ptaconv_ngrant	Output	PTA active low medium granted

Table 81. PTACONV internal input/output signals (continued)

Pin name	Signal type	Description
ptaconv_ker_ck	Input	Kernel clock
AHB	Input/output	AHB bus interface

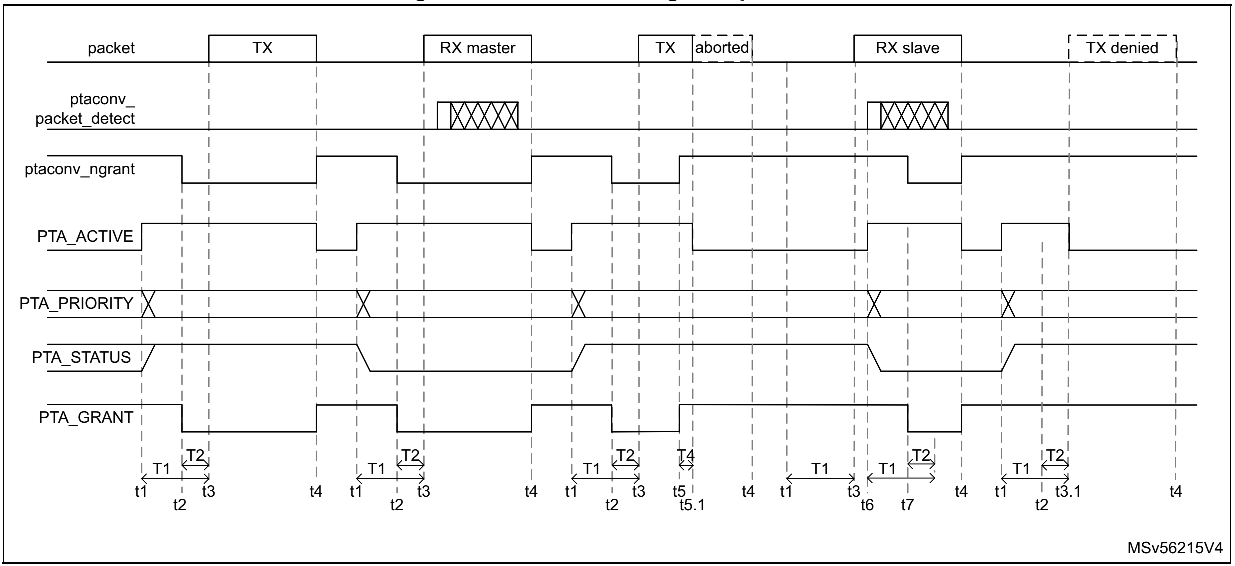
10.3.3 PTACONV protocols

The PTA grant protocol, the PTA deny protocol, and the 3-wire PTA_STATUS time-multiplexed priority and status transmit receive information are detailed in this section.

PTA grant protocol

The 4-wire PTA grant protocol uses PTA_ACTIVE, PTA_STATUS, PTA_PRIORITY, and PTA_GRANT signals, as shown in *Figure 25*.

Figure 25. 4-wire PTA grant protocol



Timing parameters:

- T1 is the time for the PTA_ACTIVE signal to request the medium before the start of the 2.4 GHz RADIO packet transfer.
- T2 is the time before the start of the 2.4 GHz RADIO packet transfer, during which the PTA_GRANT is stable.
- T4 is the delay time to cease an ongoing 2.4 GHz RADIO packet transmission and to deactivate PTA_ACTIVE.

Time instances:

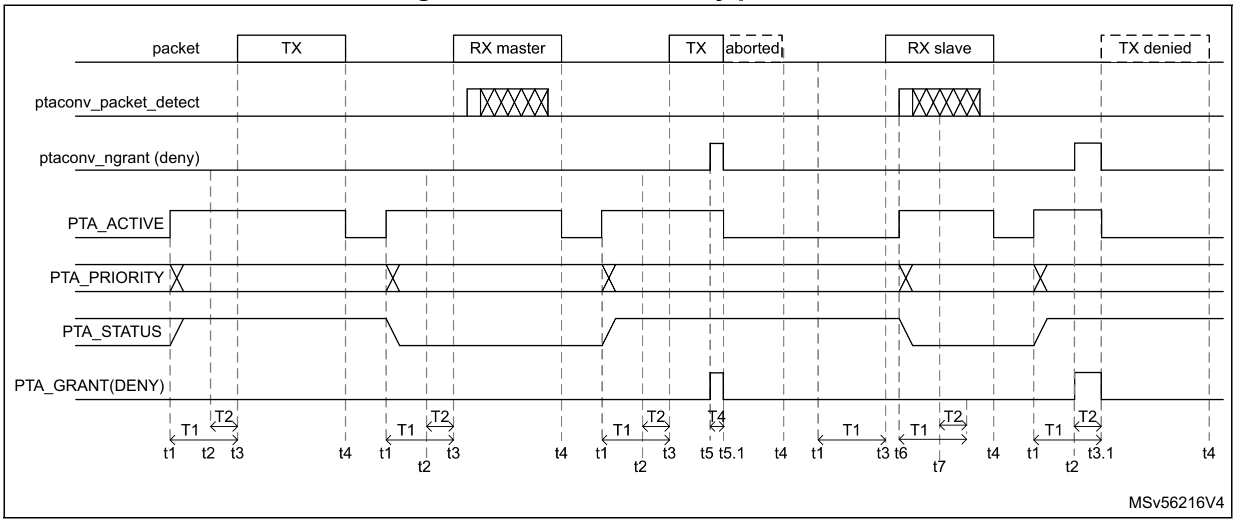
- t1: the request signal is activated before the 2.4 GHz RADIO packet transfer. For transmit and receive master packets t1 is also the time PTA_ACTIVE is activated.
- t2: PTA_GRANT signal is stable.
- t3 start of the 2.4 GHz RADIO packet transfer.
- t3.1: the 2.4 GHz RADIO packet transfer is denied, and PTA_ACTIVE is deactivated. The external PTA controller can remove the PTA_GRANT as well.
- t4: indicates the end of the 2.4 GHz RADIO packet and the deactivation of the PTA_ACTIVE signal when active. The external PTA controller can remove the PTA_GRANT as well.
- t5: PTA_GRANT goes to deny and requests to cease the ongoing packet transmission.
- t5.1: indicates that the ongoing packet is aborted and PTA_ACTIVE signal is deactivated. Abortion of the transmit packet can be disabled with register bit. ABORTDIS, in this case the ngrant signal stays low until the end of the transmit packet.
- t6: the receive packet has been detected and used for slave receive packets to activate the PTA_ACTIVE signal.
- t7: PTA_GRANT is stable.

Receive packets are always granted and reception always proceeds. Receive packets use the PTA protocol to request the reservation of the medium for high priority receive packets.

PTA deny protocol

The PTA deny protocol uses PTA_ACTIVE, PTA_STATUS, PTA_PRIORITY and PTA_DENY signals, as shown in *Figure 26*.

Figure 26. 4-wire PTA deny protocol



Timing parameters:

- T1 is the time for the PTA_ACTIVE signal to request the medium before the start of the 2.4 GHz RADIO packet transfer.
- T2 is the time before the start of the 2.4 GHz RADIO packet transfer, during which the PTA_GRANT is stable.

Time instances:

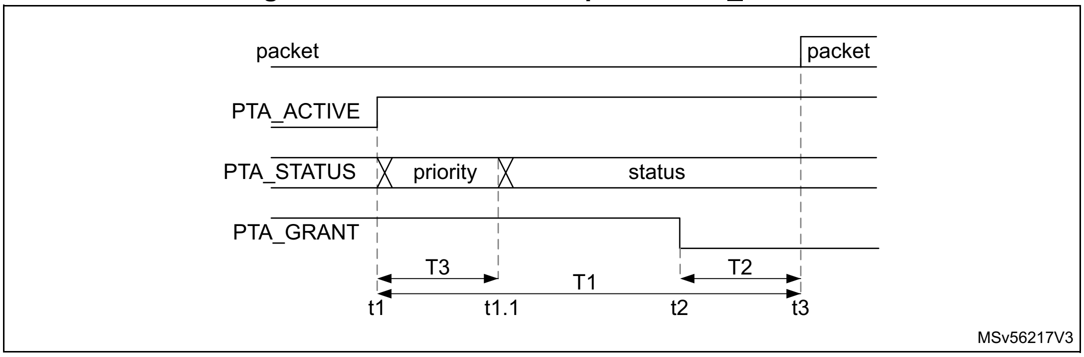
- t1: the request signal is activated before the 2.4 GHz RADIO packet transfer. For transmit and receive master packets t1 is also the time PTA_ACTIVE is activated.
- t2: PTA_GRANT signal is stable.
- t3: start of the 2.4 GHz RADIO packet transfer.
- t3.1: a 2.4 GHz RADIO packet transfer is denied and the PTA_ACTIVE is deactivated. The external PTA controller too can remove the PTA_GRANT.
- t4: indicates the end of the 2.4 GHz RADIO packet and the deactivation of the PTA_ACTIVE signal when active. The external PTA controller too can remove the PTA_GRANT.
- t5: PTA_GRANT goes to deny and requests to cease the ongoing packet transmission.
- t5.1: indicates that the ongoing packet is aborted and the PTA_ACTIVE signal is deactivated. Abortion of the transmit packet can be disabled with register bit ABORTDIS, in this case the deny signal stays high until the end of the transmit packet.
- t6: the receive packet has been detected and used for slave receive packets to activate the PTA_ACTIVE signal.
- t7: PTA_GRANT is stable.

Receive packets are always granted and reception always proceeds. Receive packets use the PTA protocol to request the reservation of the medium for high priority receive packets.

3-wire time shared PTA_STATUS

The 3-wire PTA protocol does not use the PTA_PRIORITY signal. The priority and transmit receive packet status information is time-multiplexed on the PTA_STATUS signal, as shown in *Figure 27*.

Figure 27. 3-wire time-multiplexed PTA_STATUS



Timing parameters:

- T1 is the time for the PTA_ATCIVE to request the medium before the start of the 2.4 GHz RADIO packet transfer.
- T2 is the time before the start of the 2.4 GHz RADIO packet transfer during which the PTA_GRANT is stable.
- T3 is the time the priority is time-multiplexed on the PTA_STATUS signal.

Time instances:

- t1: the request signal is activated before the 2.4 GHz RADIO packet transfer. For transmit and receive master packets it is also the time PTA_ACTIVE is activated. Priority information is time-multiplexed on the PTA_STATUS signal.
- t1.1: the transmit or receive status is time-multiplexed on the PTA_STATUS signal.
- t2: PTA_GRANT signal is stable.
- t3: start of the 2.4 GHz RADIO packet transfer.

PTA timing parameters

Table 82. 2.4 PTACONV timing parameters

Symbol	Parameter	Min	Typ	Max	Unit
T1, T_{1_PTA}	PTA_ACTIVE setup time	20	-	150	µs
$T_{1\_PTA\_jitter}$	PTA_ACTIVE setup time jitter	-2	-	2	
T2, T_{2_PTA}	PTA_GRANT setup time	5	-	-	
T3, T_{3_PTA}	PTA_STATUS priority valid time	8	-	20	
T4, T_{4_PTA}	Transmit packet abort delay	5	-	10	

10.3.4 PTACONV interface with the 2.4 GHz RADIO

The 2.4 GHz RADIO provides the following signals:

- ptaconv_active:
  - activated with a T1 setup time before the start of the packet. This is before:
    - the PA ramping for transmit packet
    - the start of the receive window for receive packets
  - deactivated at the end of the scheduled packet. This is after:
    - the PA has ramped down for transmit packets
    - the last receive packet bit has been received, or the at the end of the review window when no packet detected
- ptaconv_priority: signaling 0 for low priority packets, 1 for high priority packets
- ptaconv_status_txrx: signaling 0 for receive packets, 1 for transmit packets
- ptaconv_status_ms: signaling 0 for master packets, 1 for slave packets
- ptaconv_packet_detect: receive packet (access code) detect indication
- ptaconv_ngrant: active low grant signal
  - to be sampled during the valid window T2 to determine if a packet transmission is granted or denied
  - monitored during the transmit packet to cease transmission upon the reception of a deny

10.4 Low-power modes

The PTACONV supports operation in low power modes, as detailed in the following table.

Table 83. Effect of low-power modes on the PTACONV

Mode	Description
Sleep	No effect, PTACONV remains operational.
Stop 0 voltage range 1	
Stop 0 voltage range 2 and Stop 1	The PTACONV register context is kept, PTA signals keep their values.
Standby retention and Standby	The PTACONV is powered down, PTA signals keep their values when enabled in register PWR_IORETENRx.

10.5 PTACONV registers

10.5.1 PTACONV active control register (PTACONV_ACTCR)

Address offset: 0x000

Reset value: 0x0005 0014

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.	ABORT DIS	TABORT[3:0]			
											rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ACTPOL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TACTIVE[7:0]							
rw								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **ABORTDIS**: Disable PTA_ACTIVE deny to abort an ongoing transmission

Set and cleared by software to define if PTA_ACTIVE will abort an ongoing transmission.
0: PTA_ACTIVE deny aborts an ongoing transmission
1: PTA_ACTIVE deny does not abort an ongoing transmission

Bits 19:16 **TABORT[3:0]**: PTA_ACTIVE delay to cease an ongoing transmission in µs

Set and cleared by software to define transmit packet abort delay and signaling on PTA_ACTIVE.
0x5 to 0xA: transmit packet PTA_ACTIVE abort delay time: T_{4_PTA} = TABORT x 1 µs
Others: reserved

Bit 15 **ACTPOL**: PTA_ACTIVE polarity

Set and cleared by software to define PTA_ACTIVE signal polarity.
0: PTA_ACTIVE active high
1: PTA_ACTIVE active low

Bits 14:8 Reserved, must be kept at reset value.

Bits 7:0 **TACTIVE[7:0]**: PTA_ACTIVE setup time in µs

Set and cleared by software to define PTA_ACTIVE setup time.
0x14 to 0x96: PTA_ACTIVE setup time: T_{1_PTA} = TACTIVE x 1 µs
Others: reserved

10.5.2 PTACONV priority control register (PTACONV_PRICR)

Address offset: 0x004

Reset value: 0x0000 000A

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**
PRIPOL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TPRIORITY[4:0]				
rw											rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **PRIPOL**: Priority polarity

Set and cleared by software to define PTA_PRIOROITY and time-multiplexed priority on PTA_STATUS signal polarity.

0: priority on PTA_PRIORITY or PTA_STATUS not inverted

1: inverted priority on PTA_PRIORITY or PTA_STATUS

Bits 14:5 Reserved, must be kept at reset value.

Bits 4:0 **TPRIORITY[4:0]**: Priority valid time in µs

Set and cleared by software to define PTA_STATUS signal priority valid time.

0x00: no time-multiplexed priority information on PTA_STATUS

0x08 to 0x14: priority information multiplexed on PTA_STATUS with valid time:

T_{3_PTA} = TPRIORITY x 1 µs

Others: reserved

10.5.3 PTACONV control register (PTACONV_CR)

Address offset: 0x008

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
GRANT POL	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**
TXRX POL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

Bit 31 **GRANTPOL**: PTA_GRANT polarity

Set and cleared by software to define PTA_GRANT signal polarity.

0: PTA_GRANT active low

1: PTA_GRANT active high

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

Bit 15 **TXRXPOL**: PTA_STATUS transmit and receive polarity

Set and cleared by software to define PTA_STATUS signal polarity for transmit and receive information.

0: PTA_STATUS receive = 0, transmit = 1

1: PTA_STATUS receive = 1, transmit = 0

Bits 14:0 Reserved, must be kept at reset value.

10.5.4 PTACONV register map

Table 84. PTACONV 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	PTACONV_ACTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ABORTDIS	TABORT[3:0]				ACTPOL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TACTIVE[7:0]							
	Reset value												0	0	1	0	1	0								0	0	0	1	0	1	0	0
0x004	PTACONV_PRICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIPOL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TPRIORITY[4:0]				
	Reset value																	0											0	1	0	1	0
0x008	PTACONV_CR	GRANTPOL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXRXPOL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0																0															
0x00C to 0x3FC	Reserved	Reserved																															

Refer to *Section 2.3* for the register boundary addresses.

11 Power control (PWR)

11.1 Introduction

The power controller manages the device power supplies and power modes transitions.

11.2 PWR main features

The power controller (PWR) main features are:

- Power supplies and supply domains
  - Core domain (V_{CORE})
  - V_{DD} domain
  - Analog domain (V_{DDA})
  - Supply for the SMPS power stage (available only on STM32WBA55xx devices)
  - Supply for the 2.4 GHz RADIO
- System supply voltage regulation
  - SMPS step-down converter
  - Linear voltage regulator (LDO)
- Power supply supervision
  - BOR monitor (including Power-on reset)
  - PVD monitor
- Power management
  - Operating modes
  - Voltage scaling control
  - Low-power modes
- GPIO retention in Standby
- TrustZone® security and privileged protection

11.3 PWR pins and internal signals

Table 85. PWR input/output pins

Pin name	Signal type	Description
VDD	Supply	Main and Backup domain supply
GND	Supply	Main ground
VDDA	Supply	Analog peripherals supply
VDDRF	Supply	2.4 GHz RADIO RF supply
VDDRFPA	Supply	2.4 GHz RADIO PA regulator supply
VDDANA(1)	Supply	2.4 GHz RADIO analog supply
VDDHPA	Output	2.4 GHz RADIO PA regulator output
VSS	Supply	Ground
VSSA(1)	Supply	Analog peripherals ground
VSSRF(1)	Supply	2.4 GHz RADIO ground
VDD11(1)	Input/Output	Logic supply (V_{CORE})
VCAP(2)	Output	Logic supply (V_{CORE})
VDDSMPS(1)	Supply	SMPS supply
VSSSMPS(1)	Supply	SMPS ground
VLXSMPS(1)	Supply	SMPS output

1. Available only on STM32WBA55xx devices.
2. Available only on STM32WBA50/52/54xx devices.

Table 86. PWR internal input/output signals

Internal signal name	Signal type	Description
WKUPx_y (x = 1 to 8, y = 1 to 4)	Input	Wake-up event source selection
WKUP interrupt	Output	Global WKUP pin interrupt
WKUP_S interrupt	Output	Global WKUP_S pin secure interrupt
PWR_CSLEEP	Output	CPU in Sleep mode, Sleep
PWR_CSTOP	Output	MCU in Stop mode, Stop

Each of the wake-up event WKUPx can be generated from device pins or internal events, selected by WUSELx[1:0] in the PWR_WUCR3 register (x = 1 to 8). A WKUP interrupt is generated only when WKUPx is generated from a device pin. WKUPx generated from internal events are associated with an internal event interrupt.

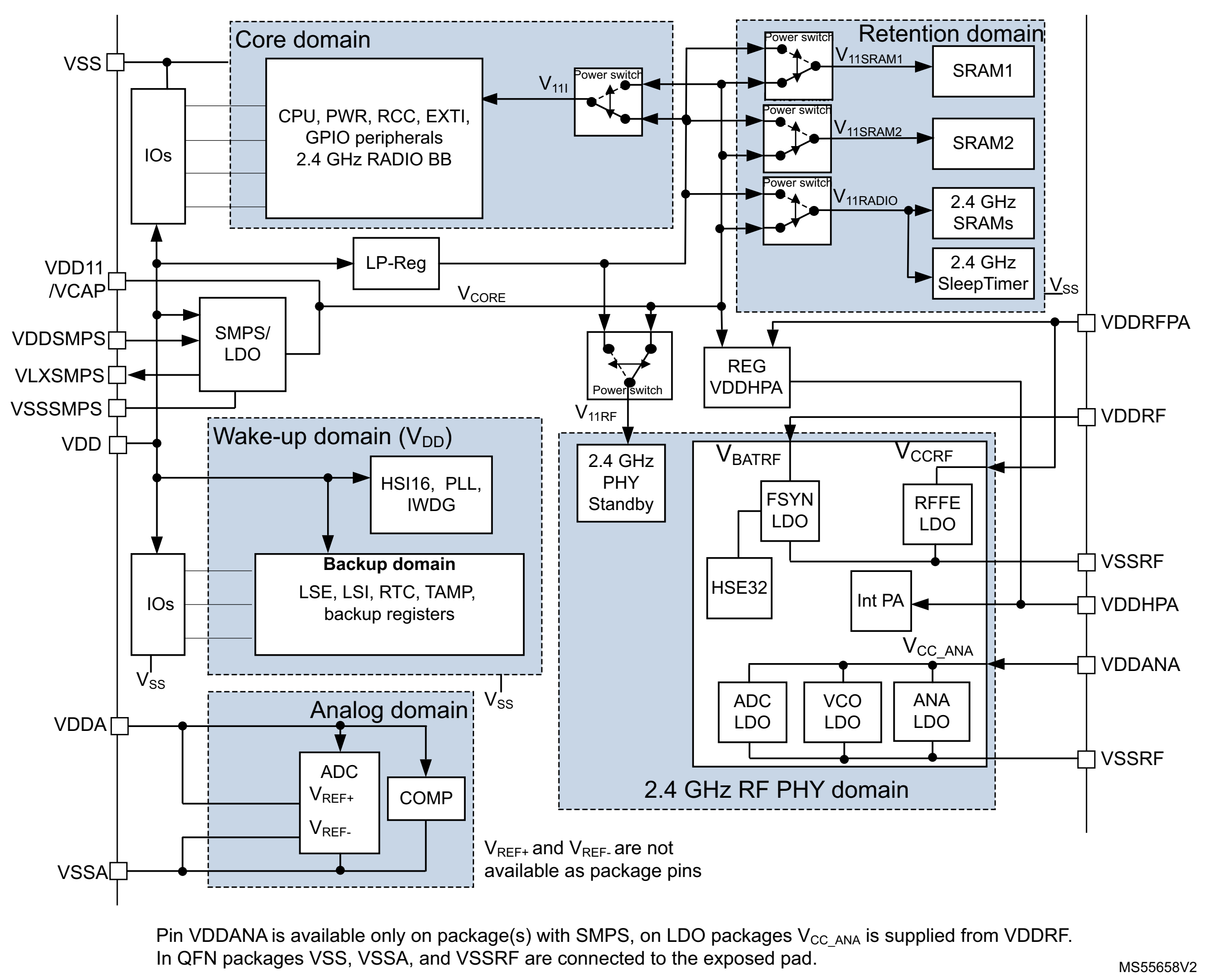
Table 87. PWR wake-up source selection

Wake-up 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	Reserved	Reserved
WKUP2	PA4(1)	PC13	Reserved	Reserved
WKUP3	Reserved	PA1	PB6	Reserved
WKUP4	PA2	PB1	Reserved	Reserved
WKUP5	Reserved	PA3	PB7	Reserved
WKUP6	PA12	PA5	Reserved	rtc_alara_s or rtc_alrb_s or rtc_wut_s or RTC_TS_S(2)
WKUP7	PB14	PA6	Reserved	rtc_alra or rtc_alrb or rtc_wut or RTC_TS(2)
WKUP8	Reserved	PA7	PB9	tamp or tamp_s(3)

1. Available only on STM32WBA55xx devices.
2. As enabled in the RTC interrupt and according the RTC interrupt security setting.
3. As enabled in the TAMP mask.

11.4 PWR power supplies and supply domains

Figure 28. Power supply overview



11.4.1 External power supplies

The devices require a 1.71 to 3.6 V V_{DD} operating voltage supply. Several independent supplies can be provided for specific peripherals. Those supplies must be provided with a valid operating supply on the VDD pin:

- V_{DD} = 1.71 V to 3.6 V

  V_{DD} is the external power supply for the I/Os, the internal regulator, and the system analog such as reset, power management, and internal clocks. It is provided externally through the VDD pins.
- V_{DDA} = 1.62 to 3.6 V

  V_{DDA} is the external analog power supply for ADC and comparators. The V_{DDA} voltage level is independent from the V_{DD} voltage and must preferably be connected to VDD when these peripherals are not used.
- V_{DDSMPS} = 1.71 V to 3.6 V (available only on STM32WBA55xx devices)

  V_{DDSMPS} is the external power supply for the SMPS step-down converter. It is provided externally through V_{DDSMPS} supply pin, and must be connected to the same supply as VDD pin.
- V_{LXSMPS} is the switched SMPS step-down converter output. (available only on STM32WBA55xx devices)

Note: The SMPS power supply pins are available only on specific package with SMPS step-down converter option.

- V_{DD11}/V_{CAP} = 0.9 V to 1.2 V
  V_{CORE} is the internal power supply for the digital logic.
- V_{DDRFPA} = 0 V to 3.6 V (must be above 1.2 V for 2.4 GHz RADIO operation).
  V_{DDRFPA} must be equal to or lower than V_{DDRF}.

  V_{DDRFPA} is the external power supply for the 2.4 GHz RADIO front-end part and PA.
  - V_{DDRFPA} when connected to VDD11 supports a transmit maximum low output power
  - V_{DDRFPA} when connected to VDD supports a transmit maximum high output power
- V_{DDANA} = 0 V to 3.6 V (must be above 1.2 V for 2.4 GHz RADIO operation), available only on STM32WBA55xx devices).
  V_{DDANA} must be equal to or lower than V_{DDRF}.

  V_{DDANA} is the external power supply for the 2.4 GHz RADIO part, can be connected to VDD11.

Note: The VDDANA supply pin is available only on specific packages with SMPS step-down converter option. In packages with only LDO support, VDDANA is double bonded with VDDRF.

- V_{DDRF} = 1.71 V to 3.6 V

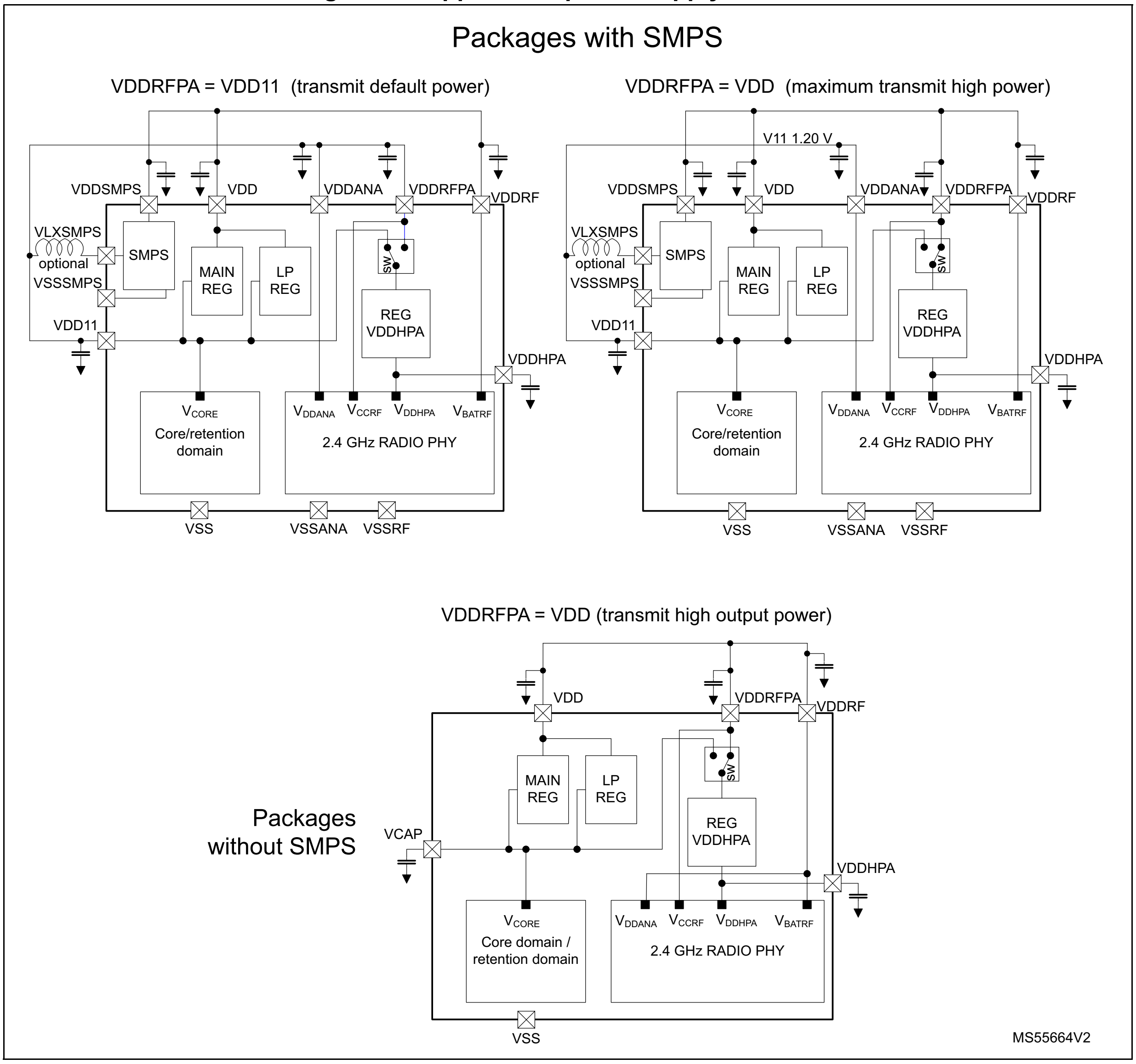
  V_{DDRF} is the external power supply for the 2.4 GHz RADIO IF part. VDDRF must always be connected to the supply used for VDD.
- V_{DDHPA} = 0.9 V to 2.3 V
  V_{DDHPA} is the internal power supply for the 2.4 GHz RADIO power amplifier.

11.4.2 Application RADIO power supply schemes

The device supports different supply schemes, as shown in *Figure 29*. The maximum achievable transmit output power from the 2.4 GHz RADIO internal PA is linked to the supply scheme.

- In packages supporting an SMPS, a maximum transmit low output power can be reached with lowest power consumption. In this power scheme V_{DDRFPA} is connected to V_{DD11}, and REGVDDHPA is connected to V11 internal selected by REGPASEL register bit.
- In all other power schemes V_{DDRFPA} is connected to V_{DD}, allowing maximum transmit high output power.

Figure 29. Application power supply schemes



11.4.3 Power-up and power-down power sequences

During power-up and power-down phases, respect the following requirements:

- When V_{DD} is below 1 V, other power supplies (V_{DDA}, V_{DDRF}) 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 discharge with different time constants during the power-down transient phase.

11.4.4 Independent analog peripherals supply

To improve A/D conversion accuracy and 2.4 GHz RADIO performance, and to extend the supply flexibility, the analog and 2.4 GHz RADIO peripherals have independent power supplies 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
- the 2.4 GHz RADIO peripheral voltage supply input is available on separate VDDRF, VDDANA, and VDDRFPA pins.

The V_{DDA} supply voltage can be different from V_{DD}. The V_{DDA} must be present before enabling any of the analog peripherals supplied by V_{DDA} (ADC4 and COMP).

When a single supply is used, VDDA can be externally connected to VDD through the external filtering circuit to ensure a noise-free V_{DDA} voltage.

11.4.5 Radio peripherals supply

The 2.4 GHz RADIO V_{DDRF}, V_{DDANA}, and V_{DDRFPA} supply voltages can be different from V_{DD}. The supplies must be present before enabling the 2.4 GHz RADIO.

When a single supply is used, VDDRF, VDDANA, and VDDRFPA supplies can be externally connected to VDD through external filtering circuitry to ensure noise-free supply voltage.

In devices with SMPS, VDDANA can be externally connected to VDD11 through external filtering circuitry to ensure noise-free supply voltage.

In devices with SMPS, for low transmit output power applications VDDRFPA can be externally connected to VDD11 through external filtering circuitry to ensure noise-free supply voltage.

11.4.6 Backup domain

The backup domain contains the RTC, TAMP, backup registers, LSE and LSI oscillators, and is directly supplied from VDD pin.

Backup domain access

After a system reset, the Backup domain (RCC Backup domain control register RCC_BDCR1, RTC registers, TAMP registers, backup registers) is protected against possible unwanted write accesses. To enable access to the Backup domain, proceed as follows:

1. Enable the power interface clock by setting the PWREN bits in the *Section 12.8.22: RCC AHB4 peripheral clock enable register (RCC_AHB4ENR)*.
2. Set the DBP bit in the *PWR disable Backup domain register (PWR_DBPR)* to enable access to the Backup domain.

11.4.7 Internal regulators

The devices embed two regulators: one LDO and one SMPS in parallel to provide the V_{CORE} (for digital peripherals, SRAM and embedded flash memory). Both regulators generate this voltage on VDD11/VCAP pins. The SMPS is available only on STM32WBA55xx devices.

Both regulators can provide two different voltage ranges (voltage scaling) and can operate in Run and Stop modes.

It is possible to switch from SMPS to LDO and from LDO to SMPS and change range on the fly (when the 2.4 GHz RADIO is not active).

Other internal supplies for the 2.4 GHz RADIO are generated for the dedicated regulator.

11.5 PWR system supply voltage regulation

11.5.1 SMPS and LDO embedded regulators

All devices embed an internal linear voltage regulator (LDO). STM32WBA55xx devices embed also an internal SMPS step-down converter, which can be selected when the application runs, depending upon application requirements.

The SMPS allows the power consumption to be reduced. Some peripherals can be perturbed by the noise generated by the SMPS, requiring the application to switch to LDO when running this peripheral, to reach the best performances.

The LDO and the SMPS regulators have two modes, namely 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 and Standby with retention mode.

11.5.2 LDO and SMPS versus reset, voltage scaling, and low-power modes

After BOR0 power-on reset and system reset, the LDO regulator is enabled, in range 2. Switching to the SMPS regulator provides lower consumption in particular for high V_{DD} voltages. It is possible to switch from LDO to SMPS, or from SMPS to LDO in any range, by configuring the REGSEL register bit.

When exiting from Stop or Standby retention modes, the regulator is the same used to enter the low-power mode. When exiting from Standby modes, the LDO regulator is always used to startup. When Standby has been entered from the SMPS regulator, after exiting Standby with the LDO, the regulator is switched automatically to SMPS regulator.

When exiting from Stop 0 modes the voltage range is the same as on entering Stop 0 mode. When exiting from Stop 1 and Standby modes the voltage range 2 is used.

When the 2.4 GHz RADIO is active, the regulator and range cannot be changed. Any requested regulator or range change while the 2.4 GHz RADIO is active is suspended and takes effect only after the 2.4 GHz RADIO and PHY have entered Sleep or Deepsleep mode.

11.5.3 LDO and SMPS step-down converter fast startup

After BOR0 power-on reset, the LDO regulator starts in high-power mode and in slow-startup mode. The slow-startup feature is selected to limit the inrush current after power-on reset. This increases the wake-up time also when exiting Standby modes.

It is possible to configure fast-startup on the fly and it is applied for next startup either after a system reset or wake-up from Standby mode. The fast-startup is selected by setting the FSTEN bit in the PWR_CR3 register. Fast-startup selection applies to both LDO and SMPS regulators.

11.5.4 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. This is used to improve device performance.

Dynamic voltage scaling to decrease V_{CORE} is known as undervolting. It is performed to save power, particularly in devices where the energy comes from a battery and is thus limited.

The regulator operates in the following ranges:

- Range 1: high performance

  It allows a system clock frequency up to 100 MHz, and is required for any 2.4 GHz RADIO transmit and receive operation.

  When the 2.4 GHz RADIO is active the range cannot be changed. Any requested range change while the 2.4 GHz RADIO is active is suspended and only takes effect after the 2.4 GHz RADIO and PHY have entered Sleep or Deepsleep mode.
- Range 2: low-power range

  The system clock frequency can be up to 16 MHz. The 2.4 GHz RADIO cannot transmit nor receive.

Voltage scaling is selected through the VOS bit in the PWR_VOSR register.

The sequence to switch the voltage scaling from range 2 to range 1 is the following:

1. Program the VOS to range 1 in the PWR_VOSR
2. Wait until the VOSRDY flag is set in the PWR_VOSR
3. If target SYSCLK > 50 MHz
   a) Switch on the PLL1 oscillator source
   b) Select the PLL1 clock source in PLL1SRC in the RCC_PLL1CFGR
4. Adjust number of wait states according to the new target SYSCLK frequency. Flash LATENCY in the FLASH_ACR, and SRAM WSC in the RAMCFG_MxCR.
5. Configure and enable the PLL1 is needed
6. Switch to the new SYSCLK frequency.

The sequence to switch the voltage scaling from range 1 to range 2 is the following:

1. Switch to the SYSCLK frequency ≤ 16 MHz
2. Adjust number of wait states according to the new target SYSCLK frequency. Flash LATENCY in the FLASH_ACR, and SRAM WSC in the RAMCFG_MxCR.
3. Disable the PLL1
4. Program the VOS to range 2 in the PWR_VOSR
5. Optionally wait until the ACTVOS in the PWR_SVMSR = VOS in the PWR_VOSR and ACTVOSRDY flag is set in PWR_SVMSR

Note: When switching the voltage scaling, the sequence must be completed (VOS = ACTVOS and ACTVOSRDY = 1) before entering Stop or Standby modes. When the 2.4 GHz RADIO is active (PWR_RADIOSCR.MODE = active) the system does not enter low-power mode and the CPU can enter Deepsleep mode independently from VOS, ACTVOS and ACTVOSRDY.

11.5.5 2.4 GHz RADIO PA regulator

The PA regulator REG VDDHPA is used to supply the 2.4 GHz RADIO internal PA with the correct voltage level V_{DDHPA} for a given transmit output power. This regulator output voltage V_{DDHPA} is controlled from the 2.4 GHz RADIO link layer software. V_{DDHPA} maximum supply level and associated maximum transmit output power is depended on the application supply scheme. See *Section 11.4.2: Application RADIO power supply schemes*.

The application software can control the REG VDDHPA regulator input voltage.

- Force the input voltage selection of the REG VDDHPA regulator to be from VDDRFPA pin by setting the REGPASEL register bit.
  - When the REG VDDHPA regulator input voltage is forced to VDDRFPA, power consumption at low transmit output power is higher. Lowering noise in the RADIO which may improve the transmit spectrum and receiver sensitivity.
- Allow the 1.2 V V_{DDHPA} voltage level to be generated directly from the VDD11 supply via the REGPABYPEN register bit. This can be selected only when VDD11 is supplied from the device SMPS. This is available only when the input voltage of the REG VDDHPA regulator is not forced to VDDRFPA.
  - When the SMPS is used, allowing the 1.2 V V_{DDHPA} voltage level to be generated directly from the VDD11 lowers power consumption at the transmit output levels using this supply level at the cost of introducing more noise in the RADIO which may impact the transmit spectrum and receiver sensitivity.

The required REG VDDHPA regulator supply source configuration in REGPASEL and REGPABYPEN must be set before using the regulator. When the REG VDDHPA regulator is used, REGPASEL and REGPABYPEN must not be changed.

The REG VDDHPA is forced off, discharging the external capacitor, in all Standby modes.

For more information on the 2.4 GHz RADIO transmit output power and REG VDDHPA regulator control see *Section 9.4.3: Transmit output power*.

11.6 PWR power supply supervision

11.6.1 Brownout reset (BOR)

The device has an integrated BOR (brownout reset) circuitry. The BOR is active in all power modes, and cannot be disabled. The BOR reset also generates a system reset on pin NRST.

Five BOR thresholds can be selected through option bytes. A power-on reset is always generated at the V_{BOR0} thresholds.

The Backup domain is supplied by V_{DD} and is reset by the V_{BOR0} thresholds.

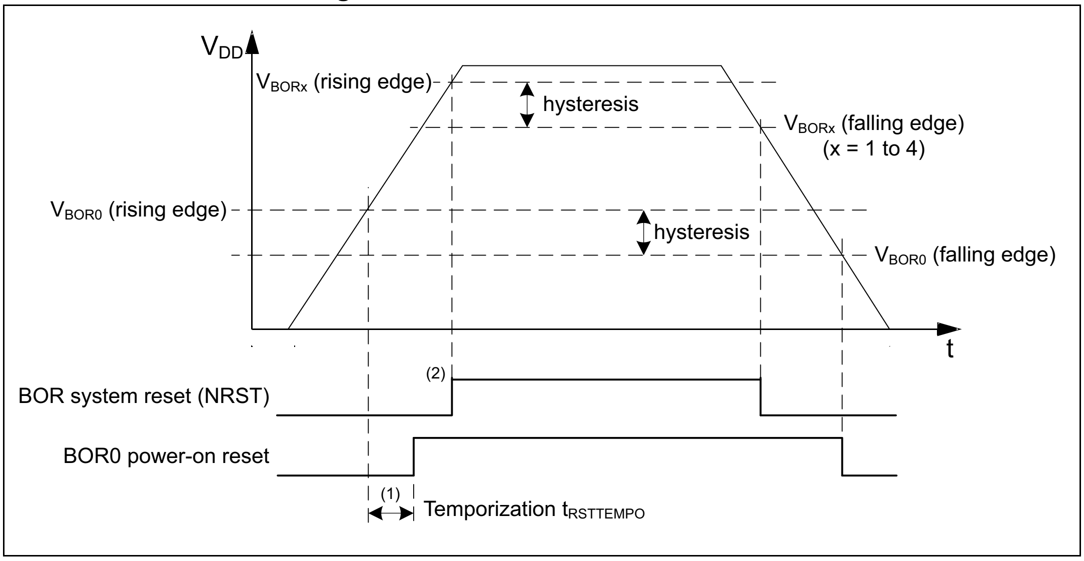
The other V_{BORx} (x = 1 to 4) thresholds keep most of the device under system reset until the supply voltage V_{DD} reaches the specified threshold. When V_{DD} drops below the selected threshold, a system reset on pin NRST is generated. When V_{DD} is above the V_{BORx} upper limit, the system reset on pin NRST 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 Stop 1 and Standby modes, it is possible to set the BOR0 in Ultra-low power (discontinuous) mode to further reduce the current consumption by setting the ULPMEN bit in *PWR control register 1 (PWR_CR1)*.

Warning: In Ultra-low power mode, the supply monitoring is discontinuous and may not detect fast drops in supply, hence it must not be used together with autonomous peripherals using HSI16 as kernel clock.

Figure 30. Brownout reset waveform



1. The reset temporization $t_{RSTTEMPO}$ is present only for the BOR0 lowest threshold (V_{BOR0}).
2. The rising edge of the system reset NRST can be driven from the V_{BOR0} power-on reset or V_{BORx} (x = 1 to 4) rising threshold depend on the V_{DD} rising slope.

11.6.2 Programmable voltage detector (PVD)

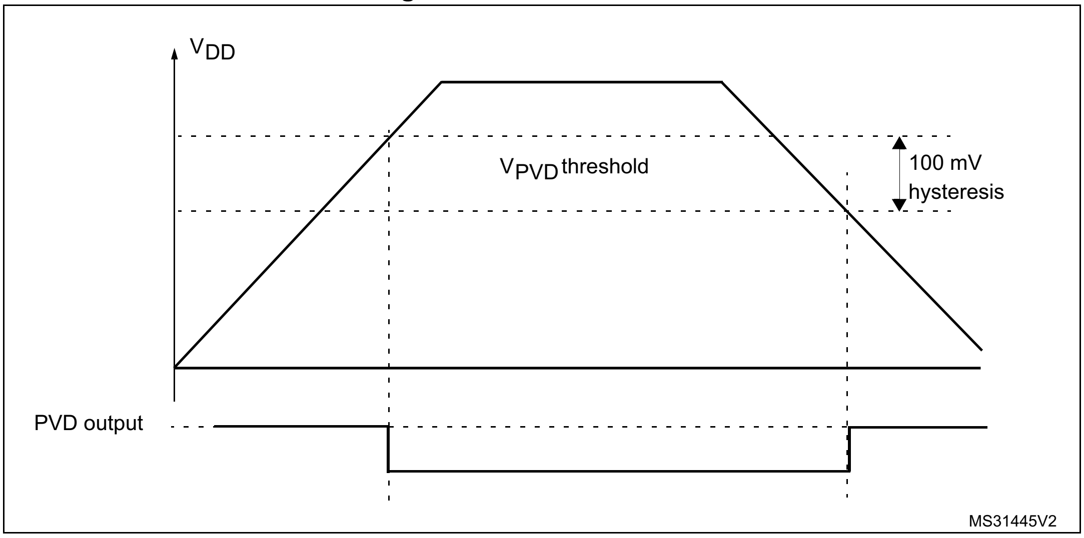
The PVD can be used to monitor the V_{DD} power supply by comparing it to a threshold selected by the PVDLS[2:0] bits in the *PWR supply voltage monitoring control register (PWR_SVMCR)*. Also the PVD can be used to monitor an analog signal on the PVD_IN I/O by comparing it to the V_{REFINT} threshold. The PVD is enabled by setting the PVDE bit.

A PVDO flag is available in the *PWR supply voltage monitoring status register (PWR_SVMSR)* 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 97*).

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 modes, and the PVD interrupt can wake up the system from the Stop modes. The PVD is not functional in Standby mode.

Figure 31. PVD thresholds



11.7 PWR power management

11.7.1 PWR power modes

By default, the microcontroller is in Run mode range 2 and the 2.4 GHz RADIO in Deepsleep after a system or a power reset. Several low-power modes are available to save power when the CPU or peripherals do 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 wake-up sources.

The device features these low-power modes:

- Run mode:
  The CPU is running. The power consumption can be reduced by slowing down the system clocks and configuring voltage scaling to lower-power range, and/or gating the clocks to the APB and AHB peripherals when they are not used.
- 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 11.7.5*.
- Stop 0, Stop 1 modes:

  Stop modes achieve the lowest power consumption while retaining the registers content. The SRAM content can be selected to be retained or not. All clocks (except some autonomous peripherals bus and kernel clocks) in the Core domain are stopped. The PLL and HSE32 crystal oscillator are disabled. The HSI16 RC oscillator is disabled when not activated as autonomous peripheral bus or kernel clock. The LSE or LSI can still run.

  Some peripherals are autonomous and can operate in Stop modes by requesting their kernel clock, and in Stop 0 mode also when requesting their bus clock when needed. When a peripheral request its (APB or AHB) bus clock,for example to transfer data with GPDMA1, the device transitions from Stop 1 to Stop 0 mode.

  The I2C, USART, LPUART, SPI, LPTIM, ADC, RTC can remain active in Stop modes when kernel clock is LSE, LSI, or HSI16. The 2.4 GHz RADIO can remain active in Stop 0 mode on its kernel clock HSE32. In Stop 0 mode the autonomous peripherals bus clock can remain active using HSI16.

  The brownout reset (BOR) remains always active, and the PVD can be kept active in Stop modes. In Stop 1 mode the BOR0 can be configured in ultra-low power mode to further reduce power consumption.

Warning: Ultra-low power mode must not be used together with autonomous peripherals using HSI16 as kernel clock.

In Stop 0 mode, the regulator remains in main regulator mode, allowing a very fast wake-up time but with higher consumption compared with Stop 1.

The system clock when exiting from Stop mode is HSI16 at 16 MHz.

Refer to *Section 11.7.6*, and *Section 11.7.7*.

- Standby mode:

  The Standby mode is used to achieve the lowest power consumption. The internal regulator is switched off so that the Core domain is powered off. The PLL, the HSI16 RC, and the HSE32 crystal oscillators are also switched off. The LSE or LSI can still run. The RTC and TAMPER can remain active in Standby modes when kernel clock is LSE or LSI.

  The BOR always remains active in Standby mode. The BOR can be configured in ultra-low power mode to further reduce power consumption during 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.

  When entering Standby mode register contents are lost except for registers in the Backup domain and Standby circuitry.

  Optionally, the full SRAM1 and/or SRAM2 can be retained in Standby mode, supplied by the low-power regulator (Standby with retention mode).

  Optionally, the 2.4 GHz RADIO sleep Timer, RXTXRAM, and sequence SRAM can be retained in Standby mode, supplied by the low-power regulator (Standby with retention mode).

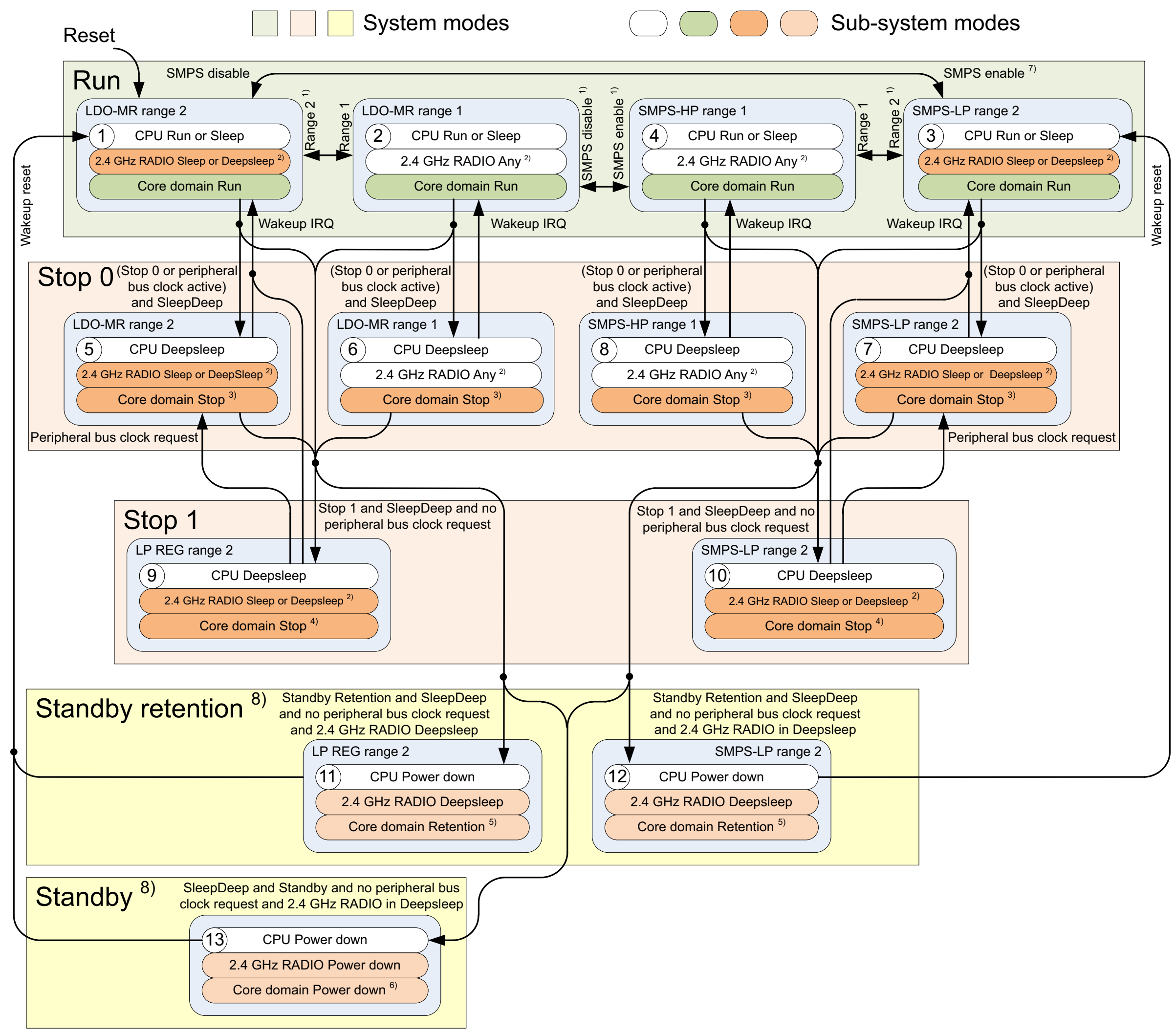
  The device exits Standby mode when an NRST pin external reset, an IWDG early interrupt or reset, WKUP pin event (configurable rising or falling edge), an RTC event occurs (alarm, periodic wake-up, timestamp), a TAMP tamper detection, or a 2.4 GHz RADIO sleep timer event (available only in Standby with retention mode). The tamper detection can be raised either due to external pins or due to an internal failure detection.

  The system clock after wake-up is HSI16 16 MHz.

  Refer to *Section 11.7.8*.

The operating modes and transitions are shown in *Figure 32*.

Figure 32. Operating modes



1) When the 2.4 GHz RADIO is active the range and SMPS/LDO operating mode cannot be changed. Any request to change power operating mode is delayed and takes effect only after the 2.4 GHz RADIO has entered Sleep mode.

2) The 2.4 GHz RADIO can only change states Deepsleep, Sleep and Active by software, when the device is in Run mode. When the device moves to a low power mode the 2.4 GHz RADIO remains in its same mode. 2.4 GHz RADIO Deepsleep state is independent from the presence of power supply.

3) BAM autonomous peripherals can operate on bus clock HSI16 and/or kernel clock LSE, LSI or HSI16.

4) Autonomous peripherals can operate on kernel clock LSE, LSI or HSI16.

5) Autonomous peripherals RTC, TAMP, 2.4 GHz RADIO sleep timer can operate on kernel clock LSE or LSI.

6) Autonomous peripherals RTC, TAMP can operate on kernel clock LSE.

7) For Standby entered from SMPS modes, wakeup is done to Run LDO-MR range 2 mode and a subsequent move to Run SMPS-LP range 2 mode is performed in hardware.

8) Standby modes are entered regardless of autonomous peripherals kernel clock activity. It is up to software to select the correct low power mode with autonomous peripherals kernel clock activity.

MS55660V4

Table 88 shows the power modes overview.

Table 88. Low-power mode summary

<table>
<tr><th>Name</th><th>Entry</th><th>Wake-up source(1)</th><th>Wake-up system clock</th><th>Effect on clocks</th><th>Voltage regulator(2)</th></tr>
<tr><td rowspan="3">Sleep</td><td>WFI or Return from ISR</td><td>Any interrupt</td><td rowspan="3">Same as before entering Sleep mode</td><td rowspan="3">CPU clock OFF
No effect on other clocks or analog clock sources</td><td rowspan="4">Main regulator range 1, 2 (SMPS or LDO)</td></tr>
<tr><td>WFE SEVONPEDND = 0</td><td>Wake-up event</td></tr>
<tr><td>WFE SEVONPEDND = 1</td><td>Any interrupt, wake-up event</td></tr>
<tr><td>Stop 0</td><td>(LPMS = Stop 0 or autonomous peripheral bus clock request active) + SLEEPDEEP bit + WFI or Return from ISR or WFE</td><td rowspan="2">Any EXTI line, autonomous peripherals, 2.4 GHz RADIO sleep timer event, IWDG event, RTC event, TAMP event, WKUP event, NRST external reset, BOR reset (only in Stop 0 range 1. HSECSS event)</td><td rowspan="4">HSI16 at 16 MHz</td><td>All clocks OFF except LSE, LSI and HSI16 or HSE32 when requested as autonomous peripheral kernel or bus clock</td></tr>
<tr><td>Stop 1</td><td>(LPMS = Stop 1 and no autonomous peripheral bus clock request) + SLEEPDEEP bit + WFI or Return from ISR or WFE</td><td>All clocks OFF except LSE, LSI and HSI16 when requested as autonomous peripheral kernel clock.</td><td rowspan="2">Low-power regulator (SMPS or LDO)</td></tr>
<tr><td>Standby with retention</td><td>(LPMS = Standby + (R[2:1]RSB or RADIORSB != 0) and no autonomous peripheral bus clock request and 2.4 GHz RADIO mode = Deepsleep) + SLEEPDEEP bit + WFI or Return from ISR or WFE</td><td>2.4 GHz RADIO sleep timer event, IWDG event, RTC event, TAMP event, WKUP event, NRST external reset, BOR reset</td><td rowspan="2">All clocks OFF except LSI and LSE</td></tr>
<tr><td>Standby</td><td>(LPMS = Standby + (R[2:1]RSB and RADIORSB = 0) and no autonomous peripheral bus clock request and 2.4 GHz RADIO mode = Deepsleep) + SLEEPDEEP bit + WFI or Return from ISR or WFE</td><td>IWDG event, RTC event, TAMP event, WKUP event, NRST external reset, BOR reset</td><td>OFF</td></tr>
</table>

1. Refer to *Table 89*.
2. SMPS is available only on STM32WBA55xx devices.

Table 89. Functionalities depending on the working mode[(1)]

Peripheral	Run/Sleep		Stop 0			Stop 1		Standby retention		Standby	
	Range 1	Range 2	Range 1	Range 2	Wake-up capability	-	Wake-up capability	-	Wake-up capability	-	Wake-up capability
CPU	A		R		-	R	-	-	-	-	-
ICACHE	O		R		-	R	-	-	-	-	-
Flash memory	O[(2)]		R		-	R	-	R	-	R	-
SRAM1	A		O		-	O	-	O	-	-	-
SRAM2	A		O		Y	O	-	O	-	-	-
Backup registers	A		R		-	R	-	R	-	R	-
2.4 GHz RADIO	O	R	O	R	Y	R	-	-	-	-	-
2.4 GHz RADIO SRAM	O	R	O	R	-	R	-	O	-	-	-
2.4 GHz RADIO Sleep timer	O	O	O	O	Y	O	Y	O	Y	-	-
BOR	A		A		Y	A	Y	A	Y	A	Y
PVD	O		O		O	O	O	-	-	-	-
HSI16 clock	O		O[(3)]		-	O[(3)]	-	-	-	-	-
HSE32 clock	O		O[(4)]	-	-	-	-	-	-	-	-
LSI clock	O		O		-	O	-	O	-	O	-
LSE clock	O		O		-	O	-	O	-	O	-
CSSHSE clock security	O		O	-	Y	-	-	-	-	-	-
CSSLSE clock security	O		O		Y	O	Y	O	Y	O	Y
RTC	O		O		Y	O	Y	O	Y	O	Y
TAMP	O		O		Y	O	Y	O	Y	O	Y
GPIO	O		R[(5)]		Y[(6)]	R[(5)]	Y[(6)]	R[(7)]	Y[(8)]	R[(7)]	Y[(8)]
IWDG	O		O		Y	O	Y	O	Y	O	Y
GPDMA1	O		O		Y	R[(9)]	-	-	-	-	-
USARTx (x = 1, 2[(10)])	O		O		Y	O	Y	-	-	-	-
LPUART1	O		O		Y	O	Y	-	-	-	-
I2Cx (x = 1[(10)], 3)	O		O		Y	O	Y	-	-	-	-
SPIx (x= 1[(10)], 3)	O		O		Y	O	Y	-	-	-	-
ADC4	O		O		Y	O	Y	-	-	-	-
COMPx[(11)] (x = 1, 2)	O		O		Y	O	Y	-	-	-	-
Temperature sensor	O		O[(12)]		Y[(12)]	O[(12)]	Y[(12)]	-	-	-	-

Table 89. Functionalities depending on the working mode[1] (continued)

Peripheral	Run/Sleep		Stop 0			Stop 1		Standby retention		Standby	
	Range 1	Range 2	Range 1	Range 2	Wake-up capability	-	Wake-up capability	-	Wake-up capability	-	Wake-up capability
LPTIMx (x = 1, 2[10])	O		O		Y	O	Y	-	-	-	-
TIMx (x = 1, 2, 3[10], 16, 17[10])	O		R		-	R	-	-	-	-	-
SAI1[11]	O		R		-	R	-	-	-	-	-
TSC	O		R		-	R	-	-	-	-	-
RNG	O		R		-	R	-	-	-	-	-
AES	O		R		-	R	-	-	-	-	-
SAES[10]	O		R		-	R	-	-	-	-	-
HSEM	O		R		-	R	-	-	-	-	-
PKA	O		R		-	R	-	-	-	-	-
HASH	O		R		-	R	-	-	-	-	-
CRC	O		R		-	R	-	-	-	-	-
WWDG	O		R		-	R	-	-	-	-	-
GTZC_TZSC[10]	O		R		-	R	-	-	-	-	-
GTZC_TZIC[10]	O		R		-	R	-	-	-	-	-
GTZC_MPCBB1[10]	O		R		-	R	-	-	-	-	-
GTZC_MPCBB2[10]	O		R		-	R	-	-	-	-	-
GTZC_MPCBB6[10]	O		R		-	R	-	-	-	-	-
SysTick timer	O		R		-	R	-	-	-	-	-
Debug	O		O[13]		-	O[13]	-	O[14]	-	O[14]	-

1. A = Active, Y = yes. O = optional (can be enabled/disabled by software).R = Retained, - = not available. Gray cells highlight the wake-up capability in each mode.
2. The flash memory can be configured in power-down mode.
3. Some peripherals with autonomous mode and wake-up from Stop capability can request HSI16 to be enabled. In this case, the oscillator is woken up by the peripheral, and is automatically put off when no peripheral needs it.
4. For the autonomous 2.4 GHz RADIO the HSE32 can be kept running.
5. GPIO pins from peripherals supporting autonomous mode are still operational.
6. Only GPIOs with enabled wake-up functionality in the EXTI or WKUP are able to wake up the system.
7. GPIO level retention in Standby modes can be enabled.
8. Only GPIOs with WKUP functionality are able to wake up the system.
9. BAM autonomous peripherals can wake up the GPDMA to perform data transfers.
10. Available only on STM32WBA52/4/5xx devices.
11. Available only on STM32WBA54/5xx devices.

12. Functional through ADC4 in autonomous mode.
13. DBGMCU remains accessible trough AP0.
14. DBGMCU remains accessible through AP0 when CDBGPWRUPREQ is set.

Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop and Standby modes while the debug features are used. This is because the Cortex-M33 core is no longer clocked or powered.

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 45.2.5: DBG low-power modes.*

11.7.2 PWR background autonomous mode (BAM)

The devices support a background autonomous mode (BAM). This allows autonomous peripherals to be functional on its bus and kernel clock in Stop 0 mode and on its kernel clock in Stop 0 and Stop 1 modes (when the CPU is in Deepsleep, not running any software). In Stop 1 mode, whenever an autonomous peripheral request its bus clock, Stop 0 mode is entered while the CPU remains in Deepsleep.

Stop 0 and Stop 1 modes

In Stop 0 and Stop 1 modes, the autonomous peripherals are ADC, LPTIM, USART, LPUART, SPI, I2C:

- Peripherals are autonomous only with LSE, LSI or HSI16 used as kernel clock, or HSI16 used as bus clock
- When an autonomous peripheral request its bus clock Stop 0 mode is entered while keeping the CPU in Sleepdeep.

In Stop 0 mode the 2.4 GHz RADIO can operate autonomously on HSI16 used as bus clock and HSE32 used as kernel clock. The 2.4 GHz RADIO cannot be autonomous in Stop 1 mode.

When entering low-power modes Stop and Standby from Run and an autonomous peripheral bus clock request is active, Stop 0 mode is entered, regardless of the low-power mode selection in LPMS bits. Entering in the LPMS selected low-power mode (Stop 1 and Standby) is delayed until the autonomous peripheral bus clock request is released.

When in Stop 1 mode and an autonomous peripheral request its bus clock, Stop 0 mode is entered.

When in Stop 0 mode and all autonomous peripheral bus clock request are deactivated, the low-power mode selected in the LPMS bits is entered. Note that Standby modes can be entered only when the 2.4 GHz RADIO is in Deepsleep.

Note: As soon as the CPU enters Sleepdeep, the system enters Stop mode and the BAM operation autonomous peripheral bus clock and SYSCLK is switched to HSI16 at 16 MHz. If autonomous peripheral operation with higher bus clock frequencies is needed, the CPU must enter Sleep and keep the system in Run with the configured Run mode SYSCLK clock frequency.

BAM in Stop mode

BAM is supported by the autonomous peripherals with the following features:

- Functionality in Stop mode thanks to the peripheral 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. For the peripheral kernel clock to be switched on, both the peripheralEN and peripheralSMEN bits in the RCC registers must be set.
- DMA transfers supported in Stop 0 mode, thanks to the peripheral bus clock request capability: the system clock (HSI16) is 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 and both peripheralEN and peripheralSMEN bits in the RCC registers are set, the system clock is woken up and distributed to all peripherals for which the RCC peripheralEN and peripheralSMEN bits are set. This makes possible DMA transfers between peripherals and SRAMs. The 2.4 GHz RADIO bus clock is requested independently from the RADIOEN and RADIOSMEN.
- Automatic start of the peripheral thanks to the hardware synchronous or asynchronous triggers (such as I/Os edge detection and low-power timer events)
- Wake up the CPU from Sleepdeep mode through a peripheral interrupt.

The GPDMA1 is fully functional and the linked-list is updated in Stop 0 mode, allowing the different DMA transfers to be linked without any CPU intervention. This can be used to chain transfers between different peripherals, or to write peripheral registers, to change their configuration while in Stop 0 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. In BAM in Stop mode only SRAM transfers are supported (flash memory transfers are not).

Here below some use-cases that can be done while remaining in Stop mode:

- A/D conversion triggered by a low-power timer (or any other trigger)
  – Wake-up from Stop mode on analog watchdog if the A/D conversion result is out of the programmed thresholds
  – Wake-up from Stop mode on DMA buffer event
- I^2C slave reception or transmission, SPI reception, UART/LPUART reception
  – Wake-up at the end of peripheral transfer or on DMA buffer event
- I^2C master transfer, SPI transmission, UART/LPUART transmission, triggered by a low-power timer (or any other trigger)
  – Example: Sensor periodic read
  – Wake-up at the end of peripheral transfer or on DMA buffer event
- Bridges between peripherals
  – Example: A/D converted data transferred by communication peripherals
- Data transfer from/to GPIO to/from SRAM for:
  – Controlling external components
  – Implementing data transmission and reception protocols
- Data transfer from one SRAM to another one.

Here below some 2.4 GHz RADIO use-cases that can be done while remaining in Stop 0 mode range 1:

- Transmit and receive data packets.
  - Wake-up at the end of the scheduled event.

11.7.3 PWR 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 12: Reset and clock control (RCC)*.

In addition to slowing down the system clocks, the voltage range can be changed. When the system clocks frequency is low enough range 2 may be entered.

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. The peripheral clock gating is controlled by the RCC_AHBxENR and RCC_APBxENR registers.

In Sleep and Stop modes, the hclk and pclk for individual peripherals and memories can be stopped automatically. The Sleep and Stop mode peripheral clock gating is controlled by the RCC_AHBxSMENR and RCC_APBxSMENR registers. For autonomous peripherals to be able to request their clocks the peripheral bus clock in Sleep and Stop mode must be enabled. Except for the 2.4 GHz RADIO, which requests its bus clock independently from the setting in the RADIOEN and RADIOSMEN register bits.

For the 2.4 GHz RADIO the bus clock is running only in Sleep and Stop modes when the STRADIOCLKON is set, or when the 2.4 GHz RADIO is active and RADIOEN and RADIOSMEN are set.

11.7.4 PWR low-power modes

Entering into a low-power mode

The MCU enters in low-power modes by

- The CPU 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.

Software must enter Low-power Stop and Standby modes only when voltage scaling is ready (ACTVOSRDY = 1).

When an autonomous peripheral bus clock request is active only Stop 0 is entered, regardless of the low-power mode set in LPMS register bits.

Standby modes are entered only when the 2.4 GHz RADIO is in Deepsleep mode.

Exiting a low-power mode

The way the CPU 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 CPU exits the low-power mode as soon as an event occurs. The wake-up event can be generated 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 CPU resumes from WFE, the peripheral interrupt pending bit and the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) must be cleared. Only NVIC interrupts with high enough priority wake up and interrupt the CPU.
    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 CPU resumes from WFE, the peripheral interrupt pending bit and when enabled the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) must be cleared. All NVIC interrupts wake up the MCU, even the disabled ones. Only enabled NVIC interrupts with high enough priority wake up and interrupt the CPU.
  - an event:
    Configuring an EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the EXTI peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bits corresponding to the event line is not set. It may be necessary to clear the interrupt flag in the peripheral.

The CPU exits Standby mode through a reset. After waking up from Standby 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 wake up the CPU 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 on the same AHB or APB bus. Consequently, enabled peripherals, even without autonomous mode capability, are temporarily clocked and can also generate an interrupt during this time. These interrupts also wake up the device from Stop mode.

11.7.5 PWR Sleep mode

I/O states in Sleep mode

In Sleep mode, all I/O pins keep the same state as in Run mode. In addition I/O pins can be toggled through DMA or other active communication peripherals.

To further reduce the power consumption in Sleep mode, disabling the peripherals clocks in Sleep mode can be performed automatically by resetting the corresponding bit in the RCC_AHBxSMENR and RCC_APBxSMENR registers.

Entering the 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 the 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 bit is cleared in Cortex-M33 system control register – No interrupt (for WFI) or event (for WFE) pending
	On return from ISR while: – SLEEPDEEP bit is cleared in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending
Mode exit	If WFI or Return from ISR was used for entry Interrupt (see *Table 135: Vector table*) If WFE was used for entry and SEVONPEND = 0: Wake-up event (see *Section 19.3: EXTI functional description*) If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC (see *Table 135: Vector table*) or wake-up event (see *Section 19.3: EXTI functional description*)
Wake-up latency	None

11.7.6 PWR Stop 0 mode

The Stop mode is based on the CPU1 Cortex®-M33 Sleepdeep mode combined with peripheral clock gating. The voltage regulator configuration as used in Run mode remains the same in Stop 0 mode. In Stop mode, all clocks in the Core domain are stopped. The PLL, HSI16 and HSE32 oscillators are disabled.

Some peripherals with autonomous capability can switch on HSI16 or HSE32 for transferring data (see *Section 11.7.2* for details). Autonomous capable peripherals using HSE32 require the use of Stop 0 mode range 1.

To reduce the power consumption in Stop mode, disabling the peripherals clocks can be performed automatically by resetting the corresponding bit in the RCC_AHBxSMENR and RCC_APBxSMENR registers. This bit must be set for the autonomous peripherals requesting clocks in Stop mode.

All register contents are preserved, SRAM1 and/or SRAM2 can totally or not be retained to further reduce consumption.

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. In addition, I/O pins can be toggled through DMA or other autonomous communication peripherals.

Entering the 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. The regulator is kept in the same regulation mode and voltage scaling range as used in Run mode. (see the table below 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 34.4: IWDG functional description*).
- The real-time clock (RTC) is configured by the RTCSEL bit in the *RCC Backup domain control register (RCC_BDCR1)*.
- The internal RC oscillator LSI or LSI1 clock divided by 128, is configured by the LSION and LSI1PREDIV bits in the RCC_BCDR1 register.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in RCC_BCDR1.
- The 2.4 GHz RADIO sleep timer configured through the link layer software.
- SRAM1 retention configured by the SRAM1PDS1 bit in *PWR control register 2 (PWR_CR2)*.
- SRAM2 retention configured by the SRAM2PDS1 bit in *PWR control register 2 (PWR_CR2)*.

Several peripherals can be autonomous in Stop 0 mode, increasing power consumption if enabled (see *Section 11.7.2* for more details).

The COMPs and the PVD can be used in Stop mode. If not needed, they must be disabled by software to reduce power consumption.

The ADC4 and the temperature sensor can consume power during the Stop 0 mode, unless they are disabled before entering the mode.

Entering Stop 1and Standby modes from Stop 0

When entering low-power with Stop 1 or Standby selected in LPMS and an autonomous peripheral bus clock request is active, Stop 0 mode is entered instead. Only when all autonomous peripheral bus clock requests are de-asserted the LPMS selected low-power mode is subsequently entered.

Exiting the Stop 0 mode

The MCU exits the Stop 0 mode as described in *Exiting a low-power mode* (see *Table 91*).

When exiting Stop 0 mode by issuing an interrupt or a wake-up event, HSI16 is selected as system clock.

Several peripherals are autonomous in Stop mode, and can generate interrupts with wake-up from Stop capability. All peripheral clocks must be enabled to allow a wake-up from Stop interrupt (see *Peripheral clock gating*).

When exiting the Stop 0 mode, the MCU is in Run mode same range as before entering Stop 0 mode.

The MCU can enter another low-power mode, as selected in LPMS, from the Stop 0 mode when all autonomous peripheral bus clocks requests are de-asserted.

Table 91. Stop 0 mode

Stop 0 mode	Description
Mode entry from Run	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
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending
	– LPMS = Stop 0 in PWR_CR1 or autonomous peripheral bus clock request is active
	Note: To enter Stop 0 mode, all EXTI line pending bits (in the EXTI rising edge pending register (EXTI_RPR1) and EXTI falling edge pending register (EXTI_FPR1)), and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Stop 0 mode entry procedure is ignored and the program execution continues.
Mode entry from Stop 1	Autonomous peripheral bus clock request active

Table 91. Stop 0 mode (continued)

Stop 0 mode	Description
Mode exit to Run	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 wake-up capability (see *Table 135: Vector table*). - 2.4 GHz RADIO sleep timer event, WKUPx event, RTC event, TAMP event, IWDG event, NRST external reset, BOR reset - 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) - In addition only from Stop 0 range 1, 2.4 GHz RADIO interrupt and HSECSS interrupt If WFE was used for entry and SEVONPEND = 0: - any EXTI line configured in event mode (see *Section 19.3: 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 wake-up capability (see *Table 135: Vector table*). - any EXTI line is configured in event mode (see *Section 19.3: EXTI functional description*) - 2.4 GHz RADIO sleep timer event, WKUPx event, RTC event, TAMP event, IWDG event, NRST external reset, BOR reset - 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) - In addition only from Stop 0 range 1, 2.4 GHz RADIO interrupt and HSECSS interrupt
	Note: All peripheral clocks must be enabled to allow this peripheral to generate a wake-up from Stop interrupt (peripheralEN, and peripheralSMEN bits must be set in the RCC, and a functional independent clock must be selected).
Mode exit to low-power modes	All autonomous peripheral bus clock requests de-asserted.
Wake-up latency	HSI16 wake-up time when applicable and flash wake-up time from Stop 0 mode.

11.7.7 PWR Stop 1 mode

The Stop 1 mode is the same as Stop 0 mode except that the regulator is in low-power mode and voltage scaling is set to range 2. (see the table below for details on how to enter and exit Stop 1 mode).

The BOR is always available in Stop 1 mode. When not using autonomous peripherals operation with HSI16 as kernel clock, the BOR0 can be forced in ultra-low power mode by the ULPMEN bit in the PWR_CR1 to reach the lowest power consumption.

Entering the Stop 1 mode

The MCU enters the Stop 1 mode in the same way as entering Stop 0. Whenever an autonomous peripheral bus clock request is active, Stop 1 mode is not entered and Stop 0 mode is entered instead.

Entering the Stop 0 mode from Stop 1

When in low-power Stop 1 mode an autonomous peripheral bus clock request is activated, Stop 0 mode range 2 is entered, with HSI16 as system clock and HDIV5 is set to divide by 2 by hardware.

Exiting the Stop 1 mode

The MCU exits the Stop 1 mode as described in *Exiting a low-power mode* (see *Table 92*).

When exiting Stop 1 mode by issuing an interrupt or a wake-up event, Run range 2 with HSI16 selected as system clock and HDIV5 is set to divide by 2 by hardware. Before entering Stop 1 mode software must configure adequate FLASH wait states latency to at least 1 in FLASH_ACR register and SRAM1 and SRAM2 wait states to at least 1 in RAMCFG_MxCR.

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
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending
	– LPMS = Stop 1 in PWR_CR1 and all autonomous peripheral bus clock requests de-asserted. – LPMS = Standby in PWR_CR1 and all autonomous peripheral bus clock requests de-asserted and 2.4 GHz RADIO not in Deepsleep.
	Note: To enter Stop 1 mode, all EXTI line pending bits (in the EXTI rising edge pending register (EXTI_RPR1) and EXTI falling edge pending register (EXTI_FPR1)), and the peripheral flags generating wake-up 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 to Run	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 wake-up capability (see *Table 135: Vector table*). - 2.4 GHz RADIO sleep timer event, WKUPx event, RTC event, TAMP event, IWDG event, NRST external reset, BOR reset - 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 19.3: 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 wake-up capability (see *Table 135: Vector table*). - any EXTI line is configured in event mode (see *Section 19.3: EXTI functional description*) - 2.4 GHz RADIO sleep timer event, WKUPx event, RTC event, TAMP event, IWDG event, NRST external reset, BOR reset - 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)
	Note: All peripheral clocks must be enabled to allow this peripheral to generate a wake-up from Stop interrupt (peripheralEN and peripheralSMEN bits must be set in the RCC, and a functional independent clock must be selected).
Mode exit to Stop 0	Autonomous peripheral bus clock request active.
Wake-up latency	HSI16 wake-up time and regulator wake-up time from low-power mode + Flash wake-up time from Stop 1 mode.

11.7.8 PWR Standby mode

It is based on the Cortex-M33 Sleepdeep mode, with the voltage regulators disabled (except when SRAM1, SRAM2, 2.4 GHz RADIO RAMs or sleep timer are retained). The PLL, HSI16 and HSE32 oscillators are also switched off.

The register contents are lost except for SRAMs and registers in the retention domain when enabled and the Backup domain and Standby circuitry (see *Figure 28*). SRAM1, SRAM2 content can fully be preserved depending on R1RSB1 and R2RSB1 bit configuration in PWR_CR1. In this case, the low-power regulator is ON and provides the supply to SRAM1 and/or SRAM2. Also the 2.4 GHz RADIO SRAMs can be retained, and the sleep timer kept operational depending on RADIORSB bit configuration in PWR_CR1.

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 BOR0 in ultra-low power mode.

I/O states in Standby mode

In the Standby mode, the GPIOs are by default in floating state. If Standby GPIO retention is enabled in the PWR_IORETENRx register, the GPIO retains the pull or output level. When entering Standby mode, GPIOs that are enabled for Standby mode retention keep their pull or level during and after exiting from Standby mode until the PWR_IORETRx bit is cleared by software. When entering Standby mode the PWR_IORETRx bit is set by hardware for the GPIOs with Standby retention enabled. The PWR_IORETRx only controls the pulls, it does not affect the GPIO analog, input and output states. When exiting from Standby and before re-enabling a GPIO as output, the output level must be restored to the one retained. To do this, first enable the GPIO as input and copy the input data from GPIOx_IDR to the GPIOx_ODR, before setting the GPIO as output. Once the GPIO is reconfigured, disable the retained pulls in PWR_IORETRx.

The GPIO standby retention enable information in PWR_IORETENRx and PWR_IORETRx are retained in Standby mode.

Note: The Standby GPIO retention level cannot be guaranteed when the GPIO port pin is connected to a low impedance destination.

Table 93. GPIO retention pin with pull-up and pull-down

PWR_IORETEN	GPIO port pin configuration before entering Standby	GPIO port pin configuration in Standby
0	Any	High-Z
1	Input no pulls	High-Z
	Input or output with pull-up	Pull-up
	Input or output with pull-down	Pull-down
	Output no pulls driving level high	Pull-up
	Output no pulls driving level low	Pull-down

For GPIO port pins enabled to be functional in Standby modes, the Standby GPIO retention can also be controlled in the PWR_IORETENRx register. The following GPIO functions are available in Standby modes:

- RTC outputs on PC13 and/or PB2 (standby GPIO retention must be disabled)
- TAMP tamper pins
- WKUPx_y wake-up pins
- LSE pins on PC14 and/or PC15 (Standby GPIO retention must be disabled)
- JTDO/TRACESWO on PB3 when debugger is connected and CDBGPWRUPREQ is set (Standby GPIO retention must be disabled)

When waking up from Standby with a system reset (NRST) GPIO retention is removed before software can reconfigure the GPIO port pins in the GPIO peripheral.

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 94*).

Whenever an autonomous peripheral bus clock request is active, Standby mode is not entered and Stop 0 mode is entered instead.

Standby mode is entered only when the 2.4 GHz RADIO is in Deepsleep mode. Whenever the 2.4 GHz RADIO is in Active or Sleep modes, Stop 0 mode is entered instead.

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 34.4: IWDG functional description*).
- The real-time clock (RTC) is configured by the RTCSEL bit in *RCC Backup domain control register (RCC_BDCR1)RCC Backup domain control register (RCC_BDCR1)*.
- The internal RC oscillator LSI or LSI1 clock divided by 128 is configured by the LSION and LSI1PREDIV bits in RCC_BDCR1.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in RCC_BDCR1.
- The 2.4 GHz RADIO SRAMs and sleep timer configured by the RADIORSB bit in *PWR control register 1 (PWR_CR1)*. Available in Standby retention mode.
- SRAM1 retention configured by the R1RSB1 bit in *PWR control register 1 (PWR_CR1)*. Available in Standby retention mode.
- SRAM2 retention configured by the R2RSB1 bit in *PWR control register 1 (PWR_CR1)*. Available in Standby retention mode.

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 *(see* *Table 94* for more details on how to exit Standby mode).

Table 94. 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
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending
	– LPMS = Standby in PWR_CR1 and all autonomous peripheral bus clock requests de-asserted and the 2.4 GHz RADIO in Deepsleep. – WUFx bits cleared in PWR_WUSR – RTC and TAMP flags corresponding to the chosen wake-up source cleared – 2.4 GHz RADIO sleep timer wake-up source cleared.
	Note: To enter Standby mode, all EXTI line pending bits (in the EXTI rising edge pending register (EXTI_RPR1) and EXTI falling edge pending register (EXTI_FPR1)), and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Standby mode entry procedure is ignored and the program execution continues.

Table 94. Standby mode

Standby mode	Description
Mode exit	WKUPx event, RTC event, TAMP event, IWDG event, NRST external reset, BOR reset In addition only from Standby retention 2.4 GHz RADIO sleep timer event.
Wake-up latency	Reset phase

11.7.9 Power modes output pins

To help the debug, two signals are available as device pins alternate functions:

- **PWR_CSLEEP**
  When set, indicates that the CPU is in Sleep mode:
  - WFI or WFE has been executed, and CPU stops execution CPU hclk1 stopped

  When cleared, indicates that the CPU is in Run mode.
- **PWR_CSTOP**
  When set, indicates that the device is in Stop mode, meaning that the following conditions are true:
  - WFI or WFE has been executed with CPU SLEEPDEEP = 1
  - No AHB/APB clock is running, except for BAM autonomous peripherals

  When cleared, indicates that the device is in Run or Sleep mode with AHB/APB clocked.

Note: After WFI or WFE has been executed the SYSCLK clock is kept running in Stop 0 mode when an autonomous peripheral requests its bus clock. The peripherals bus clock request can prevent the device to enter the selected low-power mode, which enters Stop 0 mode instead. (refer to Section 11.7.2 and Section 11.7.4).

Table 95 explains the MCU power mode depending on these signals states.

Table 95. Power modes output states versus MCU power modes

PWR_CSLEEP	PWR_CSTOP	MCU power modes(1)
0	0	Run mode (CPU executing)
1	0	Sleep mode (CPU Sleep)
X	1	Stop mode (MCU in Stop mode)

1. PWR_CSLEEP and PWR_CSTOP are generated in the Core domain, consequently they are not driven in Standby mode.

11.8 PWR security and privileged protection

11.8.1 PWR security protection

TrustZone security is activated by the TZEN user option bit in the FLASH_OPTR. 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
- Backup domain control

Other PWR configuration bits are secure when:

- The system clock selection is secure in RCC: the voltage scaling (VOS) configuration is secure.
- The I/O Standby mode retention configuration is secure when the corresponding GPIO is secure.

If SPRIV is set in the *PWR privilege control register (PWR_PRIVCFGR)*, the PWR_SECCFGR register can be written only by secure and privileged access. If SPRIV is cleared, PWR_SECCFGR can be written only by secure access, privileged, or unprivileged.

PWR_SECCFGR can be read by secure, non-secure, privileged, and unprivileged 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.

When the TrustZone security is disabled (TZEN = 0), PWR_SECCFGR is RAZ/WI and all other registers are non-secure.

When a peripheral is configured as secure, its related PWR feature control bits, are also secure in the associated registers. PWR_PUCRx, PWR_PDCRx, PWR_RADIOSCR, and PWR_VOSR.

A peripheral is secure when:

- For securable peripherals by GTZC-TZSC (TrustZone security controller), by the SEC security bit in the secure configuration registers corresponding to this peripheral.
- For TrustZone-aware peripherals, a security feature of this peripheral is enabled through its dedicated bits.

Table 96 gives a summary of the PWR secured bits following the security configuration bit in PWR_SECCFGR.

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.

Table 96. 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	Not applicable(2)	PWR_PRIVCFGR	SPRIV	Read OK. WI
PWR_SECCFGR	LPMSEC	PWR_CR1	All bits	RAZ/WI
		PWR_CR2	All bits	
		PWR_SR	CSSF	WI

Table 96. 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_DBPR	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	RADIOSEC	PWR_RADIOSCR	REGPABYPEN	RAZ/WI
			REGPASEL	RAZ/WI
RCC_SECCFGR	SYSCLKSEC	PWR_VOSR	VOS	RAZ/WI
GPIOx_SECCFGR (x = A to C and H)	SECy (y = 0 to 15)	PWR_IORETENRx	ENy	RAZ/WI
		PWR_IORETRx	RETy	RAZ/WI

1. PWR_SECCFGR is always secure.
2. PWR_PRIVCFGR.SPRIV is always secure.

11.8.2 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 96*).

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 96*).

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 bit in PWR_VOSR and the registers 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).

11.9 PWR interrupts

Table 97 gives a summary of the interrupt sources, and how to control them.

Table 97. PWR interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep, Stop 0, 1 modes	Exit Standby retention mode	Exit Standby modes
WKUP	External WKUP	WUFx	WUPENx	CWUFx	Yes	Yes	Yes
WKUP_S(1)	External secure WKUP	WUFx	WUPENx	CWUFx			
PVD	Programmable voltage detector through EXTI line 16	PVDO	EXTI line 16 enabled	EXTI PIF16	Yes	No	No

1. The WKUP_S secure interrupt is used only when trustZone is enabled.

11.10 PWR registers

11.10.1 PWR control register 1 (PWR_CR1)

Address offset: 0x000

Reset value: 0x0000 0000 (reset value not affected by exit Standby mode)

Access: 14 AHB clock cycles added compared to a standard AHB access

Access to this register can be protected by bits LPMSEC and SPRIV or NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	R1RSB1	Res.	Res.	RADIORSB	Res.	ULPMEN	Res.	R2RSB1	Res.	Res.	LPMS[2:0]		
			rw			rw		rw		rw			rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **R1RSB1**: SRAM1 retention in Standby mode
This bit is used to keep the SRAM1 content in Standby retention mode.
0: SRAM1 content not retained in Standby mode
1: SRAM1 content retained in Standby mode

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **RADIORSB**: 2.4 GHz RADIO SRAMs (RXTXRAM and Sequence RAM) and Sleep clock retention in Standby mode.
This bit is used to keep the 2.4 GHz RADIO SRAMs content in Standby retention mode and the 2.4 GHz RADIO sleep timer counter operational.
0: 2.4 GHz RADIO SRAMs and sleep timer content not retained in Standby mode
1: 2.4 GHz RADIO SRAMs and sleep timer content retained in Standby mode

Bit 8 Reserved, must be kept at reset value.

Bit 7 **ULPMEN**: BOR0 ultra-low power mode.
This bit is used to reduce the consumption by configuring the BOR0 in discontinuous mode for Stop 1 and Standby modes. Discontinuous mode is available only when BOR levels 1 to 4 and PVD are disabled.
0: BOR0 operating in continuous (normal) mode in all operating modes
1: BOR0 operating in discontinuous (ultra-low power) mode in Stop 1 and Standby modes.

Note: This bit must be set to reach the lowest power consumption in the low-power modes, and not set together with autonomous peripherals using HSI16 as kernel clock.

Note: When BOR level 1 to 4 or PVD is enabled continuous mode applies independently from ULPMEN.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **R2RSB1**: SRAM2 retention in Standby mode
This bit is used to keep the SRAM2 content in Standby retention mode.
0: SRAM2 content not retained in Standby mode
1: SRAM2 content retained in Standby mode

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 SleepDeep mode.
000: Stop 0 mode
001: Stop 1 mode
10x: Standby mode
Others: reserved

11.10.2 PWR control register 2 (PWR_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

Access to this register can be protected by bits LPMSEC and SPRIV or NSPRIV.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FLASHFWU	Res.	Res.	Res.	Res.	Res.	ICRAMPDS	Res.	Res.	Res.	SRAM2PDS1	Res.	Res.	Res.	SRAM1PDS1
	rw						rw				rw				rw

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

Bit 14 **FLASHFWU**: Flash memory fast wake-up from Stop modes (Stop 0, 1)
This bit is used to obtain the best trade-off between low-power consumption and wake-up 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 wake-up time).

Bits 13:9 Reserved, must be kept at reset value.

Bit 8 **ICRAMPDS**: ICACHE SRAM power-down in Stop modes (Stop 0, 1)
0: ICACHE SRAM content retained in Stop modes
1: ICACHE SRAM content lost in Stop modes

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SRAM2PDS1**: SRAM2 power-down in Stop modes (Stop 0, 1)
0: SRAM2 content retained in Stop modes
1: SRAM2 content lost in Stop modes
Note: The SRAM2 retention in Standby mode is controlled by R2RSB1 bit in PWR_CR1.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **SRAM1PDS1**: SRAM1 power-down in Stop modes (Stop 0, 1)
0: SRAM1 content retained in Stop modes
1: SRAM1 content lost in Stop modes
Note: The SRAM1 retention in Standby mode is controlled by R1RSB1 bit in PWR_CR1.

11.10.3 PWR control register 3 (PWR_CR3)

Address offset: 0x008

Power-on reset value: 0x0000 0000 (reset value not affected by exit Standby mode, not affected by system reset, except REGSEL)

Access: 14 AHB clock cycles added compared to a standard AHB access

Access to this register can be protected by bits VDMSEC and SPRIV or NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**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 after system reset and wake-up from Standby)

1: LDO/SMPS fast startup enabled

Bit 1 **REGSEL**: Regulator selection

0: LDO selected

1: SMPS selected

Note that this bit is reserved on STM32WBA50/52/54xx devices.

Bit 0 Reserved, must be kept at reset value.

11.10.4 PWR voltage scaling register (PWR_VOSR)

Address offset: 0x00C

Reset value: 0x0000 8000

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

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

Bit 16 **VOS**: Voltage scaling range selection
Set a and cleared by software.
Cleared by hardware when entering Stop 1 mode.
Access can be secured by RCC SYSCLKSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with bit SPRIV or when non-secure with bit NSPRIV.
0: Range 2 (lowest power)
1: Range 1 (highest frequency).

Bit 15 **VOSRDY:** Ready bit for V_{CORE} voltage scaling output selection
Set and cleared by hardware. When decreasing the voltage scaling range, VOSRDY must be one before increasing the SYSCLK frequency.
0: Not ready, voltage level < VOS selected level
1: Ready, voltage level ≥ VOS selected level

Bits 14:0 Reserved, must be kept at reset value.

11.10.5 PWR supply voltage monitoring control register (PWR_SVMCR)

Address offset: 0x010

Reset value: 0x0000 0000

Access to this register can be protected by bits VDMSEC and SPRIV or NSPRIV.

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

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				

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

Bits 7:5 **PVDLS[2:0]**: Programmable voltage detector level selection
These bits select the threshold detected by the programmable voltage detector:
000: V_{PVD0} ~ 2.0 V
001: V_{PVD1} ~ 2.2 V
010: V_{PVD2} ~ 2.4 V
011: V_{PVD3} ~ 2.5 V
100: V_{PVD4} ~ 2.6 V
101: V_{PVD5} ~ 2.8 V
110: V_{PVD6} ~ 2.9 V
111: External input analog voltage PVD_IN (compared internally to VREFINT). The I/O used as PVD_IN input must be configured in analog mode in the GPIO register.

Bit 4 **PVDE**: Programmable voltage detector enable
0: Programmable voltage detector disabled
1: Programmable voltage detector enabled

Bits 3:0 Reserved, must be kept at reset value.

11.10.6 PWR wake-up control register 1 (PWR_WUCR1)

Address offset: 0x014

Reset value: 0x0000 0000 (reset value not affected by exit Standby mode)

Access: 14 AHB clock cycles added compared to a standard AHB 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.	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**: Wake-up and interrupt pin WKUP8 enable

Access can be secured by WUP8SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

0: Wake-up and interrupt pin WKUP8 disabled

1: Wake-up and interrupt pin WKUP8 enabled

Bit 6 **WUPEN7**: Wake-up and interrupt pin WKUP7 enable

Access can be secured by WUP7SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

0: Wake-up and interrupt pin WKUP7 disabled

1: Wake-up and interrupt pin WKUP7 enabled

Bit 5 **WUPEN6**: Wake-up and interrupt pin WKUP6 enable

Access can be secured by WUP6SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

0: Wake-up and interrupt pin WKUP6 disabled

1: Wake-up and interrupt pin WKUP6 enabled

Bit 4 **WUPEN5**: Wake-up and interrupt pin WKUP5 enable

Access can be secured by WUP5SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

0: Wake-up and interrupt pin WKUP5 disabled

1: Wake-up and interrupt pin WKUP5 enabled

Bit 3 **WUPEN4**: Wake-up and interrupt pin WKUP4 enable

Access can be secured by WUP4SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

0: Wake-up and interrupt pin WKUP4 disabled

1: Wake-up and interrupt pin WKUP4 enabled

Bit 2 **WUPEN3**: Wake-up and interrupt pin WKUP3 enable

Access can be secured by WUP3SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

0: Wake-up and interrupt pin WKUP3 disabled

1: Wake-up and interrupt pin WKUP3 enabled

Bit 1 **WUPEN2**: Wake-up and interrupt pin WKUP2 enable

Access can be secured by WUP2SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

0: Wake-up and interrupt pin WKUP2 disabled

1: Wake-up and interrupt pin WKUP2 enabled

Bit 0 **WUPEN1**: Wake-up and interrupt pin WKUP1 enable

Access can be secured by WUP1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

0: Wake-up and interrupt pin WKUP1 disabled

1: Wake-up and interrupt pin WKUP1 enabled

11.10.7 PWR wake-up control register 2 (PWR_WUCR2)

Address offset: 0x018

Reset value: 0x0000 0000 (reset value not affected by exit Standby mode)

Access: 14 AHB clock cycles added compared to a standard AHB 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.	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**: Wake-up pin WKUP8 polarity

This bit must be configured when WUPEN8 = 0.

Access can be secured by WUP8SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

0: Detection on high level (rising edge)

1: Detection on low level (falling edge)

Bit 6 **WUPP7**: Wake-up pin WKUP7 polarity

This bit must be configured when WUPEN7 = 0.

Access can be secured by WUP7SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

0: Detection on high level (rising edge)

1: Detection on low level (falling edge)

Bit 5 **WUPP6**: Wake-up pin WKUP6 polarity

This bit must be configured when WUPEN6 = 0.

Access can be secured by WUP6SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

0: Detection on high level (rising edge)

1: Detection on low level (falling edge)

Bit 4 **WUPP5**: Wake-up pin WKUP5 polarity

This bit must be configured when WUPEN5 = 0.

Access can be secured by WUP5SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

0: Detection on high level (rising edge)

1: Detection on low level (falling edge)

Bit 3 **WUPP4**: Wake-up pin WKUP4 polarity

This bit must be configured when WUPEN4 = 0.

Access can be secured by WUP4SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

0: Detection on high level (rising edge)

1: Detection on low level (falling edge)

Bit 2 **WUPP3**: Wake-up pin WKUP3 polarity

This bit must be configured when WUPEN3 = 0.

Access can be secured by WUP3SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

0: Detection on high level (rising edge)

1: Detection on low level (falling edge)

Bit 1 **WUPP2**: Wake-up pin WKUP2 polarity

This bit must be configured when WUPEN2 = 0.\

Access can be secured by WUP2SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

0: Detection on high level (rising edge)

1: Detection on low level (falling edge)

Bit 0 **WUPP1**: Wake-up pin WKUP1 polarity.

This bit must be configured when WUPEN1 = 0.

Access can be secured by WUP1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

0: Detection on high level (rising edge)

1: Detection on low level (falling edge)

11.10.8 PWR wake-up control register 3 (PWR_WUCR3)

Address offset: 0x01C

Reset value: 0x0000 0000 (reset value not affected by exit from Standby mode)

Access: 14 AHB clock cycles added compared to a standard AHB 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**
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]**: Wake-up and interrupt pin WKUP8 selection

This field must be configured when WUPEN8 = 0.

Access can be secured by WUP8SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

00: reserved

01: WKUP8_1

10: WKUP8_2

11: WKUP8_3 (internal source, does not generate a WKUP interrupt)

Bits 13:12 **WUSEL7[1:0]**: Wake-up and interrupt pin WKUP7 selection

This field must be configured when WUPEN7 = 0.

Access can be secured by WUP7SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

00: WKUP7_0

01: WKUP7_1

10: reserved

11: WKUP7_3 (internal source, does not generate a WKUP interrupt)

Bits 11:10 **WUSEL6[1:0]**: Wake-up and interrupt pin WKUP6 selection

This field must be configured when WUPEN6 = 0.

Access can be secured by WUP6SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

00: WKUP6_0

01: WKUP6_1

10: reserved

11: WKUP6_3 (internal source, does not generate a WKUP interrupt)

Bits 9:8 **WUSEL5[1:0]**: Wake-up and interrupt pin WKUP5 selection

This field must be configured when WUPEN5 = 0.

Access can be secured by WUP5SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

00: reserved

01: WKUP5_1

10: WKUP5_2

11: reserved

Bits 7:6 **WUSEL4[1:0]**: Wake-up and interrupt pin WKUP4 selection
This field must be configured when WUPEN4 = 0.
Access can be secured by WUP4SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.
00: WKUP4_0
01: WKUP4_1
10: reserved
11: reserved

Bits 5:4 **WUSEL3[1:0]**: Wake-up and interrupt pin WKUP3 selection
This field must be configured when WUPEN3 = 0.
Access can be secured by WUP3SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.
00: reserved
01: WKUP3_1
10: WKUP3_2
11: reserved

Bits 3:2 **WUSEL2[1:0]**: Wake-up and interrupt pin WKUP2 selection
This field must be configured when WUPEN2 = 0.
Access can be secured by WUP2SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.
00: WKUP2_0, reserved on STM32WBA50/52/54xx devices
01: WKUP2_1
10: reserved
11: reserved

Bits 1:0 **WUSEL1[1:0]**: Wake-up and interrupt pin WKUP1 selection
This field must be configured when WUPEN1 = 0.
Access can be secured by WUP1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.
00: WKUP1_0
01: WKUP1_1
10: reserved
11: reserved

11.10.9 PWR disable Backup domain register (PWR_DBPR)

Address offset: 0x028

Reset value: 0x0000 0000

Access to this register can be protected by PWR VBSEC and SPRIV or NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**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 in the Backup domain are protected against parasitic write access. This bit must be set to enable the write access to these registers. Before disabling Backup domain access, make sure any write access to the domain has finished.

0: Write access to Backup domain disabled

1: Write access to Backup domain enabled

11.10.10 PWR security configuration register (PWR_SECCFGR)

Address offset: 0x030

Reset value: 0x0000 0000

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 are not written. This register can be read by secure or non-secure, privileged, or unprivileged access.

When the system is not secure (TZEN = 0), this register is read as 0 and register writes are 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.	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

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

Bit 14 **VBSEC**: Backup domain secure protection

0: PWR_DBPR can be read and written with secure or non-secure access.

1: PWR_DBPR can be read and written only with secure access.

Bit 13 **VDMSEC**: Voltage detection 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.

11.10.11 PWR privilege control register (PWR_PRIVCFGR)

Address offset: 0x034

Reset value: 0x0000 0000

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**: 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.

11.10.12 PWR status register (PWR_SR)

Address offset: 0x038

Power on reset value: 0x0000 0000

System reset value: 0b0000 0000 0000 0000 0000 0000 0000 0X00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**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 the CPU restarts from its reset vector. It 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 or Standby mode at the same time as the sysclk has been set by hardware to select HSI16. It is cleared by software by writing 1 to the CSSF bit and by hardware when SBF is set.
0: The device did not enter any Stop mode.
1: The device entered a Stop mode.

Bit 0 **CSSF**: Clear Stop and Standby flags

Access can be secured by PWR LPMSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

Writing 1 to this bit clears the STOPF, and SBF flags.

11.10.13 PWR supply voltage monitoring status register (PWR_SVMSR)

Address offset: 0x03C

Reset value: 0x0000 8000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACTVOS
															r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ACT VOSRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDO	Res.	Res.	REGS	Res.
r											r			r	

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

Bit 16 **ACTVOS**: VOS currently applied to V_{CORE}

This field provides the last VOS value.

0: Range 2 (lowest power)

1: 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: V_{CORE} is equal to the current voltage scaling provided by ACTVOS

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

Note that this bit is reserved on STM32WBA50/52/54xx devices.

Bit 0 Reserved, must be kept at reset value.

11.10.14 PWR wake-up status register (PWR_WUSR)

Address offset: 0x044

Reset value: 0x0000 0000 (reset value not affected by exit from Standby modes)

Access: 14 AHB clock cycles added compared to a standard AHB 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.	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**: Wake-up and interrupt pending flag 8

This bit is set when a wake-up event is detected on WKUP8 pin. This bit is cleared by writing 1 in the CWUF8 bit of PWR_WUSCR when WUSEL8 ≠ 11, or by hardware when WUPEN8 = 0.
When WUSEL8 = 11, this bit is cleared by hardware when all associated internal wake-up sources are cleared.
When WUSEL8 = 11, no WKUP interrupt is generated

Bit 6 **WUF7**: Wake-up and interrupt pending flag 7

This bit is set when a wake-up event is detected on WKUP7 pin. This bit is cleared by writing 1 in the CWUF7 bit of PWR_WUSCR when WUSEL7 ≠ 11, or by hardware when WUPEN7 = 0.
When WUSEL7 = 11, this bit is cleared by hardware when all associated internal wake-up sources are cleared. When WUSEL7 = 11, no WKUP interrupt is generated.

Bit 5 **WUF6**: Wake-up and interrupt pending flag 6

This bit is set when a wake-up event is detected on WKUP6 pin. This bit is cleared by writing 1 in the CWUF6 bit of PWR_WUSCR when WUSEL6 ≠ 11, or by hardware when WUPEN6 = 0.
When WUSEL6 = 11, this bit is cleared by hardware when all associated internal wake-up sources are cleared. When WUSEL6 = 11, no WKUP interrupt is generated

Bit 4 **WUF5**: Wake-up and interrupt pending flag 5

This bit is set when a wake-up event is detected on WKUP5 pin. This bit is cleared by writing 1 in the CWUF5 bit of PWR_WUSCR or by hardware when WUPEN5 = 0.

Bit 3 **WUF4**: Wake-up and interrupt pending flag 4

This bit is set when a wake-up event is detected on WKUP4 pin. This bit is cleared by writing 1 in the CWUF4 bit of PWR_WUSCR or by hardware when WUPEN4 = 0.

Bit 2 **WUF3**: Wake-up and interrupt pending flag 3

This bit is set when a wake-up event is detected on WKUP3 pin. This bit is cleared by writing 1 in the CWUF3 bit of PWR_WUSCR or by hardware when WUPEN3 = 0.

Bit 1 **WUF2**: Wake-up and interrupt pending flag 2

This bit is set when a wake-up event is detected on WKUP2 pin. This bit is cleared by writing 1 in the CWUF2 bit of PWR_WUSCR or by hardware when WUPEN2 = 0.

Bit 0 **WUF1**: Wake-up and interrupt pending flag 1

This bit is set when a wake-up event is detected on WKUP1 pin. This bit is cleared by writing 1 in the CWUF1 bit of PWR_WUSCR or by hardware when WUPEN1 = 0.

11.10.15 PWR wake-up status clear register (PWR_WUSCR)

Address offset: 0x048

Reset value: 0x0000 0000

Access: 14 AHB clock cycles added compared to a standard AHB 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.	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**: Clear wake-up flag 8

Access can be secured by WUP8SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

Writing 1 to this bit clears the WUF8 flag in PWR_WUSR.

Bit 6 **CWUF7**: Clear wake-up flag 7

Access can be secured by WUP7SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

Writing 1 to this bit clears the WUF7 flag in PWR_WUSR.

Bit 5 **CWUF6**: Clear wake-up flag 6

Access can be secured by WUP6SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

Writing 1 to this bit clears the WUF6 flag in PWR_WUSR.

Bit 4 **CWUF5**: Clear wake-up flag 5

Access can be secured by WUP5SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

Writing 1 to this bit clears the WUF5 flag in PWR_WUSR.

Bit 3 **CWUF4**: Clear wake-up flag 4

Access can be secured by WUP4SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

Writing 1 to this bit clears the WUF4 flag in PWR_WUSR.

Bit 2 **CWUF3**: Clear wake-up flag 3

Access can be secured by WUP3SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

Writing 1 to this bit clears the WUF3 flag in PWR_WUSR.

Bit 1 **CWUF2**: Clear wake-up flag 2

Access can be secured by WUP2SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

Writing 1 to this bit clears the WUF2 flag in PWR_WUSR.

Bit 0 **CWUF1**: Clear wake-up flag 1

Access can be secured by WUP1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

Writing 1 to this bit clears the WUF1 flag in PWR_WUSR.

11.10.16 PWR port A Standby IO retention enable register (PWR_IORETENRA)

Address offset: 0x050

Reset value: 0x0000 0000 (reset value not affected by exit Standby mode)

Access: 14 AHB clock cycles added compared to a standard AHB 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**
EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
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 **EN[15:0]**: Port A Standby GPIO retention enable

Access can be secured by GPIOA SECy. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

0: PAy Standby GPIO retention feature disabled.

1: PAy Standby GPIO retention feature enabled.

Note: Bit 4 is reserved on STM32WBA50/52/54xx devices.

11.10.17 PWR port A Standby IO retention status register (PWR_IORETRA)

Address offset: 0x054

Reset value: 0x0000 0000 (reset value not affected by exit Standby mode)

Access: 14 AHB clock cycles added compared to a standard AHB 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**
RET15	RET14	RET13	RET12	RET11	RET10	RET9	RET8	RET7	RET6	RET5	RET4	RET3	RET2	RET1	RET0
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	rc_w0	rc_w0

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RET[15:0]**: Port A Standby GPIO retention active

Access can be secured by GPIOA SECy. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

1: Set by hardware when Standby GPIO PAy is enabled in PWR_IORETENRA and Standby mode is entered. Standby GPIO retention PAy active.

0: Cleared by software, writing 0. Standby GPIO retention PAy disabled.

Note: Bit 4 is reserved on STM32WBA50/52/54xx devices.

11.10.18 PWR port B Standby IO retention enable register (PWR_IORETENRB)

Address offset: 0x058

Reset value: 0x0000 0000 (reset value not affected by exit Standby mode)

Access: 14 AHB clock cycles added compared to a standard AHB 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**
EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
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 **EN[15:0]**: Port B Standby GPIO retention enable

Access can be secured by GPIOB SECy. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

0: PBy Standby GPIO retention feature disabled.

1: PBy Standby GPIO retention feature enabled.

Note: Bit 10 is reserved on STM32WBA55xx devices.

11.10.19 PWR port B Standby IO retention status register (PWR_IORETRB)

Address offset: 0x05C

Reset value: 0x0000 0000 (reset value not affected by exit Standby mode)

Access: 14 AHB clock cycles added compared to a standard AHB 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**
RET15	RET14	RET13	RET12	RET11	RET10	RET9	RET8	RET7	RET6	RET5	RET4	RET3	RET2	RET1	RET0
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	rc_w0	rc_w0

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RET[15:0]**: Port B Standby GPIO retention active

Access can be secured by GPIOB SECy. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

1: Set by hardware when Standby GPIO PBy is enabled in PWR_IORETENRB and Standby mode is entered. Standby GPIO retention PBy active.

0: Cleared by software, writing 0. Standby GPIO retention PBy disabled.

Note: Bit 10 is reserved on STM32WBA55xx devices.

11.10.20 PWR port C Standby IO retention enable register (PWR_IORETENRC)

Address offset: 0x060

Reset value: 0x0000 0000 (reset value not affected by exit Standby mode)

Access: 14 AHB clock cycles added compared to a standard AHB 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**
EN15	EN14	EN13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **EN[15:13]**: Port C Standby GPIO retention enable

Access can be secured by GPIOC SECy. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

0: PCy Standby GPIO retention feature disabled.

1: PCy Standby GPIO retention feature enabled.

Bits 12:0 Reserved, must be kept at reset value.

11.10.21 PWR port C Standby IO retention status register (PWR_IORETRC)

Address offset: 0x064

Reset value: 0x0000 0000 (reset value not affected by exit Standby mode)

Access: 14 AHB clock cycles added compared to a standard AHB 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**
RET15	RET14	RET13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rc_w0	rc_w0	rc_w0													

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **RET[15:13]**: Port C Standby GPIO retention active

Access can be secured by GPIOC SECy. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

1: Set by hardware when Standby GPIO PCy is enabled in PWR_IORETENRC and Standby mode is entered. Standby GPIO retention PCy active.

0: Cleared by software, writing 0. Standby GPIO retention PCy disabled.

Bits 12:0 Reserved, must be kept at reset value.

11.10.22 PWR port H Standby IO retention enable register (PWR_IORETENRH)

Address offset: 0x088

Reset value: 0x0000 0000 (reset value not affected by exit Standby mode)

Access: 14 AHB clock cycles added compared to a standard AHB 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.	EN3	Res.	Res.	Res.
												rw			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **EN3**: Port H Standby GPIO retention enable
Access can be secured by GPIOH SEC3. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.
0: PHy Standby GPIO retention feature disabled.
1: PHy Standby GPIO retention feature enabled.

Bits 2:0 Reserved, must be kept at reset value.

11.10.23 PWR port H Standby IO retention status register (PWR_IORETRH)

Address offset: 0x08C

Reset value: 0x0000 0000 (reset value not affected by exit Standby mode)

Access: 14 AHB clock cycles added compared to a standard AHB 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.	RET3	Res.	Res.	Res.
												rc_w0			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RET3**: Port H Standby GPIO retention active
Access can be secured by GPIOH SEC3. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.
1: Set by hardware when Standby GPIO PHy is enabled in PWR_IORETENRH and Standby mode is entered. Standby GPIO retention PHy active.
0: Cleared by software, writing 0. Standby GPIO retention PHy disabled.

Bits 2:0 Reserved, must be kept at reset value.

11.10.24 PWR 2.4 GHz RADIO status and control register (PWR_RADIOSCR)

Address offset: 0x100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REGPA BYPEN	REGPA SEL	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
REGPARDY VDDRFPA	REGPARDY V11	Res.	RFVDDHPA[4:0]					Res.	Res.	Res.	Res.	ENCMODE	PHYMODE	MODE[1:0]	
r	r		r	r	r	r	r					r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **REGPABYPEN**: regulator REG_VDDHPA bypass enable.

This bit must be written only when the VDDHPA regulator is not used.

When REGPASEL = 0 this bit has no meaning.

When REGPASEL = 1, allow bypassing the REG_VDDHPA regulator when V_{DDHPA} 1.2 V is requested and input voltage is V_{DD11}.

Access can be secured by GTZC_TZSC RADIOSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

0: 2.4 GHz RADIO PA supplied by REG_VDDHPA regulator output voltage.

1: 2.4 GHz RADIO PA 1.2 V supplied directly from internal V_{DD11} (available only when REGPASEL = 1)

Note that this bit is reserved on STM32WBA50/52/54xx devices.

Bit 23 **REGPASEL**: regulator REG_VDDHPA input supply selection.

This bit must be written only when the VDDHPA regulator is not used.

Access can be secured by GTZC_TZSC RADIOSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with SPRIV or when non-secure with NSPRIV.

0: VDDRFPA pin selected as regulator REG_VDDHPA input supply.

1: regulator REG_VDDHPA input supply selection between V_{DDRFPA} and V_{DD11}, dependent on requested regulated output voltage. see *Table 78: 2.4 GHz RADIO supply configuration*.

Must be set only when device SMPS is used to generate V_{DD11}.

Note that this bit is reserved on STM32WBA50/52/54xx devices.

Bits 22:16 Reserved, must be kept at reset value.

Bit 15 **REGPARDYVDDRFPA:** Ready bit for V_{DDHPA} voltage level when selecting VDDRFPA input.

0: Not ready, V_{DDHPA} voltage level < REGPAVOS selected supply level

1: Ready, V_{DDHPA} voltage level ≥ REGPAVOS selected supply level

Note: REGPARDYVDDRFPA does not allow to detect correct V_{DDHPA} voltage level when requested to lower the level.

Bit 14 **REGPARDYV11:** Ready bit for V_{DDHPA} voltage level when selecting V_{DD11} input.
0: Not ready, V_{DDHPA} voltage level < REGPAVOS selected supply level
1: Ready, V_{DDHPA} voltage level ≥ REGPAVOS selected supply level
Note that this bit is reserved on STM32WBA50/52/54xx devices.

Note: REGPARDYV11 does not allow to detect correct V_{DDHPA} voltage level when requested to lower the level.

Bit 13 Reserved, must be kept at reset value.

Bits 12:8 **RFVDDHPA[4:0]**: 2.4 GHz RADIO VDDHPA control word.
Bits [3:0] see *Table 77: PA output power table format* for definition.
Bit [4] rf_event.

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

Bit 3 **ENCMODE**: 2.4 GHz RADIO encryption function operating mode
0: 2.4 GHz RADIO encryption function disabled
1: 2.4 GHz RADIO encryption function enabled

Bit 2 **PHYMODE**: 2.4 GHz RADIO PHY operating mode
0: 2.4 GHz RADIO Sleep mode
1: 2.4 GHz RADIO Standby mode

Bits 1:0 **MODE[1:0]**: 2.4 GHz RADIO operating mode.
00: 2.4 GHz RADIO deep sleep mode
01: 2.4 GHz RADIO sleep mode
1x: 2.4 GHz RADIO active mode

11.10.25 PWR register map

Table 98. PWR register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PWR_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	R1RSB1	Res.		RADIORSB	Res.	ULPMEN	Res.	R2RSB1	Res.	Res.	LPMS [2:0]		
	Reset value																				0			0		0		0			0	0	0
0x004	PWR_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLASHFWU	Res.	Res.	Res.	Res.	Res.	ICRAMPDS	Res.	Res.	Res.	SRAM2PDS1	Res.	Res.	Res.	SRAM1PDS1
	Reset value																		0						0				0				0
0x008	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(1)	Res.
	Reset value																														0	0	
0x00C	PWR_VOSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VOS	VOSRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																0	1															
0x010	PWR_SVMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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				
0x014	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
0x018	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
0x01C	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
0x020 to 0x024	Reserved	Reserved																															
0x028	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
0x02C	Reserved	Reserved																															
0x030	PWR_SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x034	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

Table 98. PWR register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x038	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
0x03C	PWR_SVMSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACTVOS	ACTVOSRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDO	Res.	Res.	REGS	Res.
	Reset value																0	1											0			0	
0x040	Reserved	Reserved																															
0x044	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
0x048	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
0x04C	Reserved	Reserved																															
0x050	PWR_IORETENRA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8	EN7	EN6	EN5	EN4(1)	EN3	EN2	EN1	EN0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	PWR_IORETRA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RET15	RET14	RET13	RET12	RET11	RET10	RET9	RET8	RET7	RET6	RET5	RET4(1)	RET3	RET2	RET1	RET0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x058	PWR_IORETENRB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN15	EN14	EN13	EN12	EN11	EN10(2)	EN9	EN8	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	PWR_IORETRB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RET15	RET14	RET13	RET12	RET11	RET10(2)	RET9	RET8	RET7	RET6	RET5	RET4	RET3	RET2	RET1	RET0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060	PWR_IORETENRC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN15	EN14	EN13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0	0	0													
0x064	PWR_IORETRC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RET15	RET14	RET13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0	0	0													
0x068 to 0x084	Reserved	Reserved																															
0x088	PWR_IORETENRH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN3	Res.	Res.	Res.
	Reset value																													0			
0x08C	PWR_IORETRH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RET3	Res.	Res.	Res.
	Reset value																													0			
0x090 to 0x0FC	Reserved																																

Table 98. PWR register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x100	PWR_RADIOSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REGPABYPEN	REGPASEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REGPARDYVDDRFPA	REGPARDYV11	Res.	RFVDDHPA[4:0]					Res.	Res.	Res.	Res.	ENCMODE	PHYMODE	MODE[1:0]	
	Reset value								0	0								0	0		0	0	0	0	0					0	0	0	0
0x104 to 0x3FC	Reserved																																

1. Bit reserved on STM32WBA50/52/54xx devices.
2. Bit reserved on STM32WBA55xx devices.

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

12 Reset and clock control (RCC)

12.1 Introduction

The reset and clock control (RCC) manages the different resets, and generates all clocks for the bus and peripherals.

12.2 RCC pins and internal signals

Table 99 lists the RCC inputs and output signals connected to package pins or balls.

Table 99. 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

12.3 RCC reset functional description

The following types of reset exist:

- power reset
- system reset
- Backup domain reset
- individual peripheral reset

12.3.1 Power reset

A power reset is generated when one of the following events occurs:

- a brownout reset (BOR)
- when exiting Standby modes

A BOR sets all registers to their reset values. Five BOR threshold levels can be selected by user option bytes. The Backup domain is always reset on the V_{BOR0} threshold, as is the power-on reset.

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 modes control) are not impacted.

12.3.2 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:

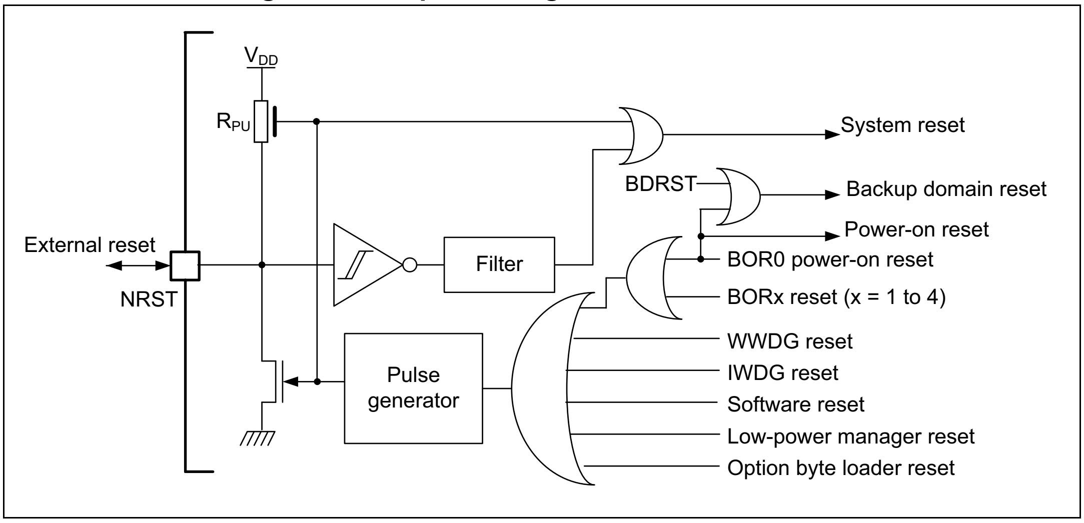
- a low level on the NRST pin (external reset)
- a window watchdog event (WWDG reset)
- an independent watchdog event (IWDG reset)
- a software reset (SW reset), see *Software reset*
- a low-power mode security reset, see *Low-power mode security reset*
- an option byte loader reset, see *Option byte loader reset*
- a brownout reset

The reset source can be identified by checking the reset flags in *RCC control/status register (RCC_CSR)*.

The device internal reset sources (BOR, WWDG, IWGD, ...) act on the NRST pin and this pin is always kept low during the delay phase. The internal reset signal is output on the NRST pin. The pulse generator guarantees a minimum reset pulse duration of $t_{O(NRST)}$, see datasheet, 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 of an internal reset, the internal pull-up R_{PU} is deactivated to save the power consumption through the pull-up resistor.

Figure 33. Simplified diagram of the reset circuit



Software reset

The SYSRESETREQ bit in CPU1 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, low-power mode security resets are available. If enabled in option bytes, a reset is generated in any of the following conditions:

- 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 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.

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_CR register. This bit is used to launch by software the option byte loading.

12.3.3 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_BDCR1)*
- a V_{DD} supply BOR0 power-on reset

A Backup domain reset affects only the LSE and LSI oscillators, the RTC, TAMP and the backup registers, and the RCC_BDCRx registers.

12.3.4 Individual peripheral reset

Individual peripherals can be reset by software with their reset register bit in the RCC.

12.3.5 CPU reset

The CPU reset vector is selected via the boot option bytes.

12.4 RCC clocks functional description

The following clock sources can be used to drive the system clock (SYSCLK):

- HSI16: high-speed internal 16 MHz RC oscillator clock
- HSE32: high-speed external crystal or clock, 32 MHz
- PLL1 clocks

The HSI16 is used as system clock source after startup from reset, configured at 16 MHz.

There are additional clock sources:

- LSI:
  - LSI1: 32 kHz low-speed low-power internal RC that drives the independent watchdog and optionally the RTC used for auto wake-up from Stop and Standby modes
  - LSI2: 32 kHz low-speed low drift internal RC that drives optionally the RTC or 2.4 GHz RADIO sleep timer used for auto wake-up from Stop and Standby modes[a].
- LSE: 32.768 kHz low-speed external crystal or clock that optionally drives the real-time clock (rtc_ck)

Each clock source can be switched on or off independently when it is not used, to optimize power consumption.

The AHB1, AHB2, the APB1, APB2, and APB7 frequencies are derived from the SYSCLK and are provided with several prescalers to configure them. The maximum frequency of these AHB and APB domains is 100 MHz.

The AHB5 frequency is also derived from the SYSCLK and is provided with a prescaler, which enables to configure it. The maximum frequency of this AHB domain is 32 MHz.

Most peripheral kernel clocks are derived from their bus clock (hclk1, hclk5, pclk1, pclk2 or pclk7). In addition, some peripherals receive an independent kernel clock. For these peripherals the kernel clock can be selected by software between several sources, thanks to RCC_CCIPRx registers (x = 1, 2, 3):

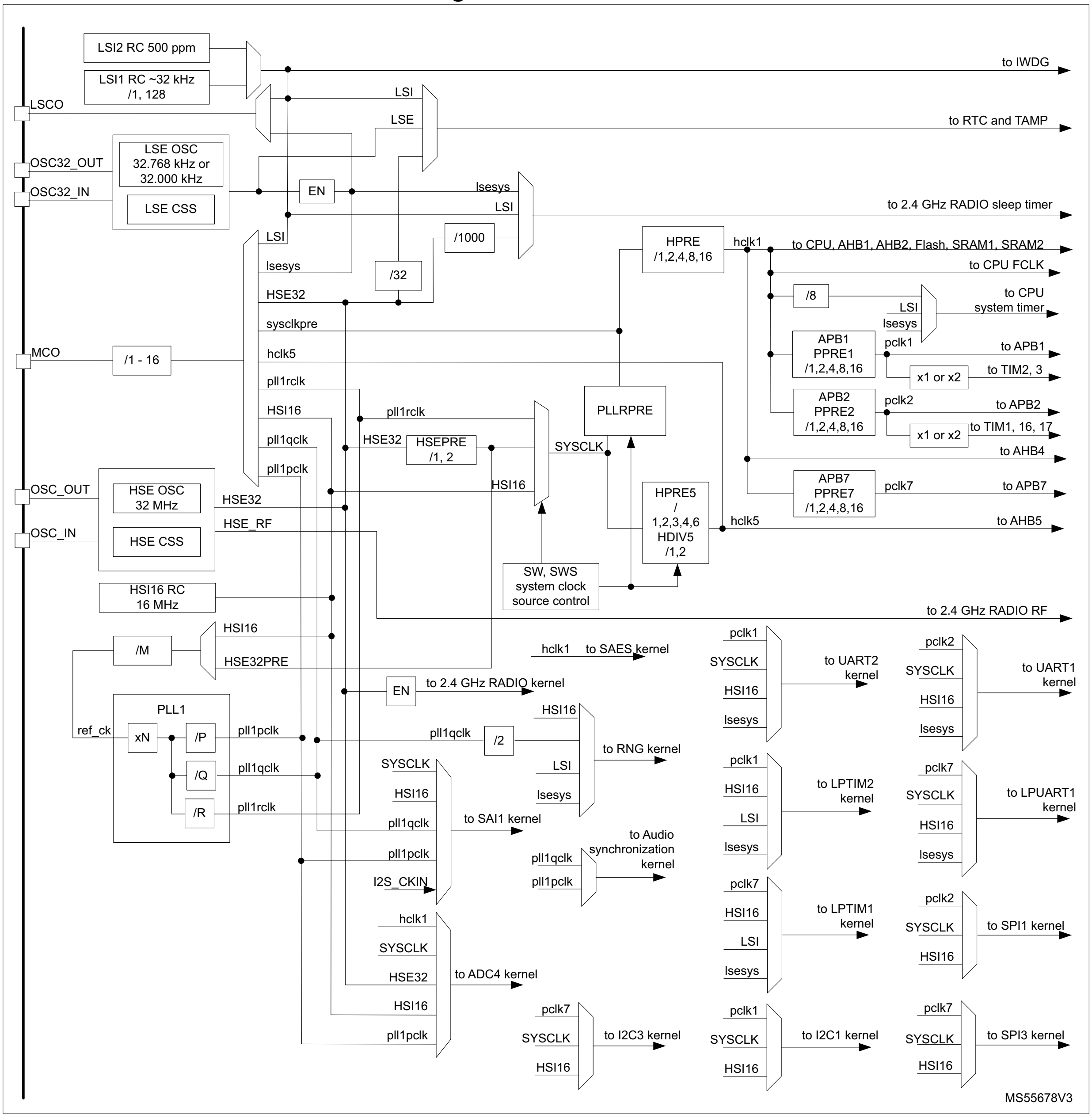
- RNG
- ADC4
- U(S)ARTx (x = 1, 2)
- LPUART1
- I2Cx (x = 1, 3)
- SPIx (x = 1, 3)
- SAI1
- LPTIMx (x = 1, 2)
- RTC and TAMP (selected in RCC_BDCR1)
- IWDG (always LSI)
- 2.4 GHz RADIO (always HSE32)
- 2.4 GHz RADIO sleep timer (selected in RCC_BDCR1)

The RCC feeds the CPU system timer (SysTick) clock with the AHB clock (hclk1) divided by 8, or LSE or LSI. The SysTick can work either with this clock or directly with the CPU bus clock (hclk1), configurable in the CPU SysTick control and status register.

FCLK acts as CPU free-running clock.

a. Available only on STM32WBA54/55xx devices.

Figure 34. Clock tree



12.4.1 HSE32 clock with trimming

The HSE32 32 MHz external oscillator has the advantage of producing a very accurate rate on the main clock. The HSE32 furthermore provides on-chip trimming capability in RCC_ECSCR1. The HSE32 must be used for any 2.4 GHz RADIO transmission and reception.

The high-speed external clock signal (HSE32) can be generated from the following clock sources:

- HSE32 external crystal
- HSE32 external clock

The clock source must be placed as close as possible to the oscillator pins to minimize output distortion and startup stabilization time.

HSE32 is controlled from the CPU and from the 2.4 GHz RADIO. It can be switched on and off using the HSEON bit in the *RCC clock control register (RCC_CR)*. HSE32 must be enabled with the HSEON bit when used for the CPU.

The HSERDY flag in the *RCC clock control register (RCC_CR)* indicates if the HSE32 oscillator is stable and forwarded or not for use by the CPU. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *RCC PLL1 dividers register (RCC_PLL1DIVR)*. See data sheet for HSE32 stabilization ready time.

When waking up from a sleep timer event the 2.4 GHz RADIO enables HSE32 autonomously for its own purpose with STRADIOCLKON, independently of the HSEON bit. To use the 2.4 GHz RADIO outside a sleep timer event, software must enable HSE32 with HSEON.

External crystal (HSE32 crystal)

The associated hardware configuration is shown in *Figure 35*. Refer to the electrical characteristics section of the datasheet for more details.

Frequency trimming

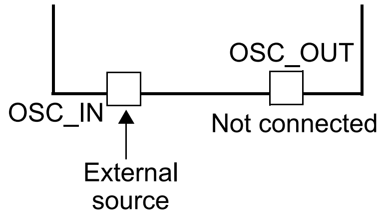
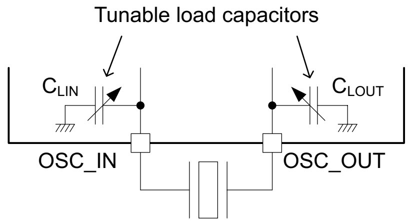
When using HSE32 with external crystal, the load capacitors are provided by the integrated capacitor banks, these can be trimmed. The HSE32 load capacitor trimming allows a compensation of device manufacturing process variations, used crystal, and PCB design. The HSE32 frequency can be tuned in the application via the RCC_ECSCR1 register.

The HSE32 frequency can be measured by outputting the HSE32 clock on the MCO when in Run or Sleep mode.

External clock (HSE32 external)

The associated hardware configuration is shown in *Figure 35*. Refer to the electrical characteristics section of the datasheet for more details.

Figure 35. HSE 32 clock sources

Clock source	Hardware configuration
External clock	
Crystal	

12.4.2 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. It also has a faster startup time than the HSE32 crystal oscillator. However, even with calibration, the frequency is less accurate than an external crystal oscillator.

The HSI16 clock is used as system clock after reset and wake-up from Stop and Standby modes. It can also be used as a backup clock source (auxiliary clock) for the CPU if the HSE32 crystal oscillator fails. Refer to *Section 12.4.8*.

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)*.

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 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 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 12.4.18*.

12.4.3 PLL1

The RCC features one PLL1 that is generally used to provide clocks to the CPU and to some peripherals. The PLL1 must be enabled only in range 1. Before entering range 2 the PLL must be disabled.

The PLL1 integrated into the RCC offers the following features:

- input frequency range: 4 to 16 MHz
- VCO frequency range: 128 to 544 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 PLL1 outputs
- three outputs with postdividers

The PLL1 is controlled via the registers RCC_PLL1DIVR, RCC_PLL1FRACR, RCC_PLL1CFGR and RCC_CR.

The frequency of the reference clock provided to the PLL1 (ref_ck) must range between 4 and 16 MHz. The user application must program properly the PLL1M dividers in the *RCC PLL1 configuration register (RCC_PLL1CFGR)* to match this condition. In addition, PLL1RGE must be set according to the reference input frequency to guarantee an optimal performance of the PLL1.

To reduce the power consumption, it is recommended to configure the VCO output to the lowest possible frequency.

PLL1N loop divider must be programmed to achieve the expected frequency at VCO output. In addition, the VCO output range (128 to 544 MHz) must be respected.

The PLL1 operates in integer mode when PLL1FRACEN is 0 and the PLL is enabled with PLL1ON. At any time fractional mode can be enabled by setting PLL1FRACN to the required value and subsequently setting PLLFRACEN from 0 to 1. The ΣΔ modulator is designed to minimize the jitter impact while allowing very small step frequency adjustments. To update the fractional value, first set PLL1FRACEN to 0 before updating the PLL1FRACN value, and subsequently set PLL1FRACEN to 1. The old PLL1FRACN value is used until the new value is activated by setting PLL1FRACEN from 0 to 1. PLL1FRACN can be updated by software only when PLL1FRACEN is 0.

The PLL1 can be enabled by setting PLL1ON to 1 in the *RCC clock control register (RCC_CR)*. The PLL1RDY bit indicates that the PLL1 is ready (meaning locked).

Note: Before enabling the PLL1, make sure that the reference frequency (ref_ck) provided to the PLL1 is stable.

The hardware prevents writing PLL1ON to 0 if the PLL1 is currently used to deliver the system clock.

The following PLL1 parameters cannot be changed once the PLL1 is enabled: PLL1M, PLL1SRC, PLL1N, PLL1RGE, PLL1P, PLL1Q, and PLL1R.

To ensure an optimal behavior of the PLL1 when one of the postdividers (PLL1P, PLL1Q, or PLL1R) is not used, the application must set the enable bit (PLL1PEN, PLL1QEN, or PLL1REN), and, preferably, also the corresponding postdivider field (PLL1P, PLL1Q, or PLL1R) to 0.

If the above rules are not respected, the PLL1 output frequency is not guaranteed.

Output frequency computation

When the PLL1 is operated in integer mode, the VCO frequency (F_{VCO}) is given by

$$F_{VCO} = F_{ref\_ck} \times PLL1N$$

When the PLL1 is operated in fractional mode, it is possible to change the value of the PLL1FRACN on-the-fly without disturbing the PLL1 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 PLL1, the VCO frequency is given by the following formula:

$$F_{VCO} = F_{ref\_ck} \times (PLL1N + PLL1FRACN / 2^{13})$$

For both integer and fractional mode, the PLL1 output frequency is given by

$$F_{pll(y)clk} = F_{VCO} / (PLL1(y) + 1), \text{ with } y = P, Q, \text{ or } R.$$

The PLL1 is disabled by hardware when:

- the system enters Stop or Standby mode
- an HSE32 failure occurs, and PLL1 clocked by HSE32 is used as system clock.

The fractional information used by the PLL is reset when disabling the PLL.

PLL1 initialization phase

Here below the recommended PLL1 initialization sequence in integer and fractional mode (PLL1 is supposed to be disabled at the start of the sequence):

1. Initialize the PLL1 registers according to the required frequency.
   - For integer mode, set PLL1FRACEN to 0.
   - For fractional mode, set PLL1FRACN to the required initial value and then set PLL1FRACEN to 1.
2. Once the PLL1ON bit is set to 1, the application must wait till PLL1RDY bit goes to 1. As long as PLL1RDY = 0, the PLL1FRACEN bit must not be altered.
3. When the PLL1RDY bit goes to 1, the PLL1 is ready to be used.
4. If the application intends to tune the PLL1 frequency on-the-fly.
   a) PLL1FRACEN must be set to 0. This allows to update the PLL1FRACN value while keeping the PLL running.
   b) A new value can be uploaded into PLL1FRACN.
   c) PLL1FRACEN must be set to 1 to activate the new programed value in PLL1FRACN and to have it taken into account by the PLL.

Note: When the PLL1RDY goes to 1, it means that the difference between the PLL1 output frequency and the target value is lower than ± 2 %.

12.4.4 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, the 2.4 GHz RADIO sleep timer or other timing functions.

The LSE crystal is switched on and off using the LSEON bit in *RCC Backup domain control register (RCC_BDCR1)*. If the LSE is used by other peripherals of functions than RTC and TAMP, the LSESYSEN bit must be also be set in the *RCC Backup domain control register (RCC_BDCR1)* (refer to *LSE when used by peripherals other than RTC/TAMP and RCC functions*).

The crystal oscillator driving strength can be changed at runtime using the LSEDRV[1:0] bits to obtain the best compromise between robustness and short start-up time on one side and low-power-consumption on the other side. The LSE drive can be decreased to a lower drive capability when the LSE is ON. However, once LSEDRV is selected, the drive capability cannot be increased if LSEON = 1.

The LSERDY flag in the *RCC Backup domain control register (RCC_BDCR1)* 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)*.

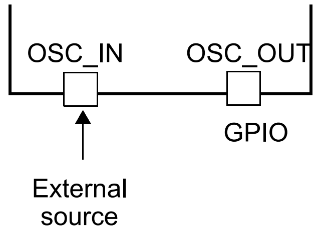
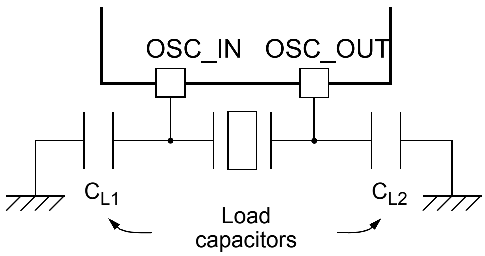
In addition, glitches on LSE can be filtered by setting LSEGFON. LSEGFON must be written when the LSE is disabled (LSEON = 0 and LSERDY = 0).

The LSE oscillator can be trimmed using the LSETRIM trimming bits in *RCC Backup domain control register (RCC_BDCR1)*. After BOR0 reset and OBL_LAUNCH when SBF is cleared the factory trimmed values are loaded in the LSETRIM bits, which can subsequently be modified by the application software.

External source (LSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 1 MHz. This mode is selected by setting the LSEBYP and LSEON bits in the *RCC Backup domain control register (RCC_BDCR1)*. 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 36*).

Figure 36. LSE 32 clock sources

<table>
<tr><th>Clock source</th><th>Hardware configuration</th></tr>
<tr><td>External clock</td><td>
</td></tr>
<tr><td>Crystal/ceramic resonators</td><td>
</td></tr>
</table>

LSE when used by peripherals other than RTC/TAMP and RCC functions

By default, when enabled by LSEON, the LSE is sent only to RTC and TAMP (assuming that RTCSEL = 01).

If the LSE is needed for other peripherals (such as peripheral clock or trigger source), or if the LSE is used by an RCC function (such as LSCO or MCO), the lsesys clock must be enabled with LSESYSEN according to the sequence below:

1. Wait till LSE clock is ready and LSEON bit is set and LSERDY bit goes to 1 in *RCC Backup domain control register (RCC_BDCR1)*.
2. Set the LSESYSEN bit in RCC_BDCR1.
3. Wait till LSESYS clock is ready (LSESYSRDY = 1 in RCC_BDCR1).

The LSE power consumption is increased when LSESYSEN = 1.

12.4.5 LSI clock

The low-power clock LSI can be kept running in Stop and Standby modes for the IWDG, RTC, TAMP, and 2.4 GHz RADIO sleep timer. The LSI clock can be generated from two sources:

- LSI1 RC low-power oscillator
- LSI2 RC low-drift oscillator[a]

a. Available only on STM32WBA54/55xx devices, Selection between LSI1 or LSI2 is done by the LSI2ON bit. Whenever LSI2 is enabled (LSI2ON =1) and LSI2 is ready (LSY2RDY = 1) the LSI clock is generated by LSI2. Else LSI1 is selected as LSI clock source.

Table 100. LSI clock selection

LSI1ON / LSI1RDY	LSI2ON / LSI2RDY	LSI clock
0 / 0	0 / 0	No clock
1 / 1	0 / 0	LSI1 RC source
0 / 0	1 / 1	LSI2 RC source
1 / 1	1 / 1	

When the IWDG is started the LSI clock is forced on and cannot be disabled. When both the LSI1 and LSI2 are disabled LSI1 is forced on. When LSI selects LSI2 RC source, the LSI1 RC source can be disabled.

LSI1 low-power

Caution: The LSI1 must not be used for the 2.4 GHz RADIO sleep timer.

The LSI1 RC can be switched on and off using the LSI1ON bit in the *RCC Backup domain control register (RCC_BDCR1)*.

The LSI1RDY flag in the *RCC Backup domain control register (RCC_BDCR1)* indicates if the LSI1 oscillator is stable or not. At startup, the clock is not released until this bit goes to 1 by hardware. An interrupt can be generated if enabled in the *RCC clock interrupt enable register (RCC_CIER)*. After LSI1 ready, there is an additional delay of up to seven clock cycles before clocking the peripheral.

The clock frequency is either ~32 kHz or ~250 Hz, depending on the LSI1PREDIV bit in *RCC Backup domain control register (RCC_BDCR1)*. Setting this bit results in a lower consumption (refer to the electrical characteristics section of the datasheet for more details).

Note: When the IWDG is enabled or when the RTC or TAMP is clocked by the LSI, the LSI1PREDIV cannot be changed anymore.

LSI2 low-drift[a]

The LSI2 RC can be switched on and off using the LSI2ON bit in the *RCC Backup domain control register (RCC_BDCR1)*.

The LSI2RDY flag in the *RCC Backup domain control register (RCC_BDCR1)* indicates if the LSI2 oscillator is stable or not. At startup, the clock is not released until this bit goes to 1 by hardware. An interrupt can be generated if enabled in the *RCC clock interrupt enable register (RCC_CIER)*. After LSI2 ready, there is an additional delay of up to six clock cycles before clocking the peripheral.

a. Available only on STM32WBA54/55xx devices, The LSI2 makes possible trimming using the LSI2CFG and LSI2MODE in *RCC Backup domain control register (RCC_BDCR2)*:

- use LSI2CFG to select the temperature at which the frequency temperature sensitivity is the lowest
- use LSI2MODE to configure the power consumption versus the accuracy. For best performance nominal power consumption and highest accuracy must be selected.

12.4.6 System clock (SYSCLK) selection

Different clock sources can be used to drive the system clock (SYSCLK):

- HSI16 oscillator
- HSE32 oscillator
- PLL1

The system clock maximum frequency is 100 MHz. After a system reset, the HSI16 oscillator at 16 MHz, is selected as system clock. When a clock source is used directly or through the PLL1 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 (for example, clock stable after startup delay or PLL1 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 configuration register 1 (RCC_CFGR1)* indicate which clocks are ready, and which one is currently used as a system clock.

Table 101 gives the different bus frequencies depending on the product voltage range.

Table 101. SYSCLK and bus maximum frequency

Product voltage	SYSCLK / AHB1 / AHB2 / AHB4 / APB1 / APB2 / APB7	AHB5
Range 1	100 MHz (maximum incremental frequency step 43 MHz)	32 MHz
Range 2	16 MHz	12 MHz

Note: After reset voltage scaling range 2 is used with SYSCLK at 16 MHz, hclk5 at 8 MHz, and one wait state on FLASH, SRAM1 and SRAM2.

System clock frequency change and hclk5

When increasing the SYSCLK frequency above 32 MHz, the hclk5 division ratio in HPRE5 must be adapted to keep the hclk5 frequency ≤ 32 MHz.

Warning: The AHB5 clock frequency must never exceed 32 MHz. When this is not respected, device operation cannot be guaranteed.

For this purpose HPRE5 must be written by software with the divider value corresponding to the $f_{PLL1RCLK}$ frequency, before switching the system clock to the PLL1 source in SW. The written HPRE5 value is used by the hardware to divide the SYSCLK, at the same time as the SYSCLK clock switch to the PLL1 source.

1. Lock PLL1 with pll1rclk at required frequency
2. Optional: select PLL1RCLKPRE to divide
3. Set HPRE5 value to be used with $f_{PLL1RCLK}$ frequency
4. Switch SYSCLK source to pll1rclk in SW
5. Wait for SYSCLK switch to be completed in SWS (hclk5 = $f_{PLL1RCLK}$ / HPRE5)

When decreasing the SYSCLK frequency from a frequency above 32 MHz from PLL1 source to the HSE32 or HSE16 source, the hclk5 is set to not divided by hardware.

6. Switch SYSCLK source away from PLL1
7. Wait for SYSCLK switch to be completed in SWS (SYSCLK = HSI16 or HSE32, hclk5 = SYSCLK / HDIV5)

The HPRE5 value written by software takes effect only when the SYSCLK has been switched to PLL1 in SWS.

When SW = PLL1 and SWS = not PLL1, writes to HPRE5 are ignored.

Note: The HPRE5 divider is not used when SYSCLK source is HSE32, HSE32 divided by 2 or HSI16 (SWS = not PLL1).

In range 2 the hclk5 frequency must be kept ≤ 8 MHz. Before entering in range 2 HDIV5 must select divide by 2.

When going from range 1 to range 2:

1. Set HDIV5 to divide by 2
2. Switch SYSCLK source to HSE32 divided by 2 or HSI16
3. Wait for SYSCLK switch to be completed in SWS
4. Select range 2 in VOS in PWR_VOSR.
5. Optionally, wait until the ACTVOS in PWR_SVMSR = VOS in PWR_VOSR and ACTVOSRDY flag is set in PWR_SVMSR.

When going from range 2 to range 1:

1. Select range 1 in VOS in PWR_VOSR
2. Wait until VOSRDY flag is set in PWR_VOSR
3. Optionally clear HDIV5 to no longer divide by 2

Note: When entering in Stop 1 mode hardware set hclk5 frequency to divider by 2 in HDIV5.

Note: The HDIV5 divider is not used when SYSCLK source is PLL1 (SWS = PLL1).

Note: When 2.4 GHz RADIO is active, the device must be in range 1 and hclk5 must be ≥ 16 MHz and HDIV5 must be cleared.

12.4.7 Clock source frequency versus voltage scaling

Table 102. Clock source maximum frequency

Product voltage	HSI16	HSE32	PLL1 outputs (VCO min to max)
range 1	Allowed	Allowed	100 MHz (128 to 544 MHz)
range 2	Allowed	Allowed (divided by 2)	Not allowed

12.4.8 HSE32 clock security system (HSECSS)

The HSECSS can be activated by software with the HSECSSON. In this case, the clock detector is enabled after the HSE32 oscillator wake-up time and disabled when this oscillator is stopped.

Thanks to the HSECSS it is possible to detect the absence of a clock. See device datasheet for more information.

If a failure is detected on the HSE32 clock, the HSE32 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 HSECSSI). This allows the MCU to perform rescue operations. The HSECSSI is linked to the CPU1 Cortex®-M33 NMI (non-maskable interrupt) exception vector.

Once the HSECSS is enabled and if the HSE32 clock fails, the HSECSSI occurs and an NMI is automatically generated. The NMI is executed indefinitely unless the HSECSSI pending bit is cleared. As a consequence, in the NMI ISR, the user must clear the HSECSSI by setting the HSECSSC bit in the *RCC clock interrupt clear register (RCC_CICR)*.

If the HSE32 oscillator is used directly or indirectly as the system clock (indirectly means: it is used as PLL1 input clock and the PLL1 clock is used as system clock), a detected failure causes a switch of the system clock to the HSI16 oscillator and the disabling of the HSE32 oscillator. If the HSE32 clock (divided or not) is the clock entry of the PLL1 used as system clock when the failure occurs, the PLL1 is disabled too.

12.4.9 LSE clock security system on (LSECSS)

A clock security system on LSE can be activated by software writing the LSECSSON bit in the *RCC Backup domain control register (RCC_BDCR1)*. 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 LSECSS is working in all modes, also under system reset (excluding power-on reset and BDRST).

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 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 and no hardware action is made to register settings.

In case of an LSECSS detection event (LSECSSD = 1 in the RCC_BDCR1), the software must disable the LSECSSON bit, stop the defective 32 kHz oscillator (disabling LSEON), change the low-power clock source (no clock or LSI or HSE32), or take any required action to secure the application.

The LSECSS detection event is connected to the internal tamper 3 of the TAMP peripheral. The internal tamper 3 must be enabled (ITAMP3E = 1 in TAMP_CR1 register) and the associated interrupt enabled (ITAMP3IE in TAMP_IER) to enable wake up from the low-power modes.

An LSECSS detection event erases also the TAMP backup registers and backup SRAM unless ITAMP3POM = 1 in TAMP_CR3, see *Section 37: Tamper and backup registers (TAMP)* for more details.

Refer to the datasheet for LSECSS electrical characteristics.

12.4.10 ADC kernel clock

The ADC kernel clock source is selected thanks to ADCSEL in the *RCC peripherals independent clock configuration register 3 (RCC_CCIPR3)*.

If the application requires that the ADC is precisely triggered by a (LP)TIM without any uncertainty, the hclk must be selected as ADC kernel clock source. The other clock sources are asynchronous to (LP)TIM therefore an uncertainty of the trigger instant is added by the resynchronization between the two clock domains.

12.4.11 RTC and TAMP kernel clock

The RTC kernel clock source is used by RTC and TAMP and can be either the HSE32 / 32, LSE or LSI clock. It is selected by programming the RTCSEL bits in the *RCC Backup domain control register (RCC_BDCR1)*. This selection cannot be modified without resetting the Backup domain. For a proper RTC operation, the RTC bus clock pclk must always be configured so as to get a greater or equal frequency compared to the RTC kernel clock.

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 37: Tamper and backup registers (TAMP)* for more details).

The LSE and the LSI clocks are in the Backup domain, whereas the HSE32 clock is not. Consequently:

- If LSE or LSI is selected as RTC or TAMP clock, these peripherals continue to work in Stop and Standby modes, provided the V_{DD} supply is maintained.
- If the HSE32 clock is used as the RTC or TAMP clock, these peripherals work only in Run and Sleep modes. They stop working in Stop and Standby modes. Depending on the TAMP configuration, they can remain functional if used in a mode that does not need a kernel clock.

When the RTC and TAMP clock is LSE or LSI, the RTC and TAMP remain 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.

12.4.12 2.4 GHz RADIO bus clocks

The 2.4 GHz RADIO bus clock can be enabled by software with RADIOEN and RADIOSMEN bits, and by hardware on a sleep timer wake-up event by STRADIOCLKON. Before accessing the 2.4 GHz RADIO sleep timer registers the bus clock must be ready, indicated by RADIOCLKRDY register bit.

Table 103. 2.4 GHz RADIO bus clock control

Device state	CPU state	RADIOEN	RADIOSMEN	STRADIOCLKON	2.4 GHz RADIO state	2.4 GHz RADIO bus clock
X	X	0	X	0	X	Off
Run	RUN	1	X	X	X	On
		X	X	1	X	On
Sleep	SLEEP	1	0	0	X	Off
		1	1	0	ACTIVE	On
					SLEEP/DEEPSLEEP	Off
		X	X	1	X	On
Stop 0	SLEEPDEEP	1	0	0	X	Off
		1	1	0	ACTIVE	On
					SLEEP/DEEPSLEEP	Off
		X	X	1	X	On
Stop 1(1)		X	X	0	SLEEP/DEEPSLEEP	Off
Standby(2)	RESET	X	X	0	DEEPSLEEP	Off

1. When 2.4 GHz RADIO state is ACTIVE or STRADIOCLKON is 1, the device does not enter Stop 1 or Standby modes.
2. When 2.4 GHz RADIO state is SLEEP, the device does not enter Standby mode.

When exiting from low-power mode and the 2.4 GHz RADIO bus clock has been stopped, the RADIOCLKRDY must be rechecked before accessing the 2.4 GHz RADIO registers.

The 2.4 GHz RADIO bus clock is kept active only in low-power modes, when STRADIOCLKON is set and/or RADIOEN and RADIOSMEN are set and the 2.4 GHz RADIO is active.

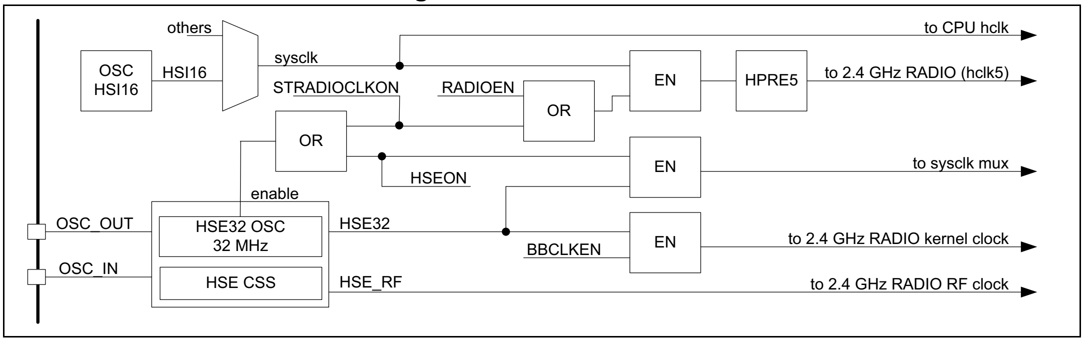
12.4.13 2.4 GHz RADIO kernel clocks

The 2.4 GHz RADIO has different kernel clocks

- The baseband kernel clock
- The sleep timer low power clock

The 2.4 GHz RADIO baseband kernel clock is enabled by register bit BBCLKEN. This clock has the HSE32 as clock source. For this purpose, the HSE32 oscillator is enabled by hardware on a 2.4 GHz RADIO sleep timer wake-up event setting the STRADIOCLKON register bit or by software setting HSEON register bit. When the 2.4 GHz RADIO no longer needs the HSE32 and bus clocks, software must clear the BBCLKEN, STRADIOCLKON and if HSE32 is not used by any other function the HSEON bits.

Figure 37. Radio control



When in range 2 the 2.4 GHz RADIO baseband kernel clock is divided by 4 by hardware. The 2.4 GHz RADIO baseband kernel clock need to be enabled to access the 2.4 GHz RADIO RXTX RAM or sequence RAM.

Also the 2.4 GHz RADIO bus clock (hclk5) is enabled by hardware via the STRADIOCLKON, independent from RADIOEN and RADIOSMEN bits. When the 2.4 GHz RADIO no longer needs its bus clock, software must clear the STRADIOCLKON bit. The STRADIOCLKON bit also keeps the 2.4 GHz RADIO bus clock and baseband kernel clock active when entering low power modes.

Outside any scheduled radio activity, when in Run mode, software can enable the 2.4 GHz RADIO bus clock by the RADIOEN and RADIOSMEN register bits. In this case, the RADIOSMEN allows to keep the 2.4 GHz RADIO bus clock and baseband kernel clock active when entering low power modes.

The 2.4 GHz RADIO sleep timer kernel clock source can be either the HSE32 / 1000, LSE or LSI clock. LSI must be used only when its source is LSI2. It is selected by programming the RADIOSTSEL bits in the *RCC Backup domain control register (RCC_BDCR1)*.

12.4.14 Timer kernel clock

The timer (TIM) kernel clock frequency is derived from the bus clock pclk. The frequency is automatically defined by hardware:

- if the APB prescaler equals 1, the timer clock frequencies are set to the APB domain frequency (kernel clock frequency = pclk frequency)
- otherwise, they are set to twice (×2) the APB domain frequency (kernel clock frequency = 2 x pclk frequency)

12.4.15 Independent watchdog kernel clock

The independent watchdog uses the LSI as kernel clock.

If the independent watchdog (IWDG) is started by either user option or software and the LSI clock is disabled (LSI1ON and LSI2ON are cleared to 0), the LSI1 oscillator is forced on. After the LSI oscillator ready delay, the LSI clock is provided to the IWDG.

12.4.16 SysTick calibration value register

The Cortex-M33 with TrustZone 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 Cortex-M33 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. 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 Cortex-M33 SYST_RVR register:

- When SysTick clock source is CPU clock hclk1
  reload value = (f_{HCLK1} x STCALIB) - 1
- When SysTick clock source is external clock (hclk1 divided by8)
  reload value = ((f_{HCLK1}/8) x STCALIB) - 1

Example: SysTick clock source is CPU clock hclk1 of 100 MHz, to match a 1 ms time base:

SysTick reload value = (100 x STCALIB) - 1 = 0x1869F

Note: When using debug Stop mode (DBG_STOP), before the CPU enters SleepDeep, it is good practice to disable the SysTick by software.

12.4.17 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.
  - active in Run, Sleep and Stop modes
    - LSI
    - LSE
    - HSI16 (in Stop modes only when an autonomous peripheral kernel clock request is active)
    - HSE32 (in Stop modes only when the 2.4 GHz RADIO kernel clock request is active)
    - SYSCLKpre (in Stop modes when an autonomous peripheral, other than 2.4 GHz RADIO, kernel clock request is active)
    - hclk5 (when enabled by RCC_AHB5CR.RADIOEN, and in Stop modes when the 2.4 GHz RADIO kernel clock request is active)
  - active only in Run and Sleep modes
    - pll1pclk
    - pll1qclk
    - pll1rclk

  The selection is controlled by the MCOSEL bits in the *RCC clock configuration register 1 (RCC_CFGR1)*. The selected clock can be divided with the MCOPRE field in the *RCC clock configuration register 1 (RCC_CFGR1)*. The MCO clock output requires the corresponding GPIO pin alternate function to select MCO.

- **LSCO**

  Slow clock output (LSCO) allows one of the low-speed clocks to be output onto the external LSCO pin:
  - LSI
  - LSE

  This output remains available in all Run, Sleep, Stop and Standby modes. The selection is controlled by the LSCOSEL bit and enabled with the LSCOEN in the *RCC Backup domain control register (RCC_BDCR1)*.

12.4.18 Internal/external clock measurement

The HSI16 and LSI frequency can indirectly be measured by mean of the TIM16 or TIM17 channel 1 input capture and LPTIM1 or LPTIM2 channel 2 input capture.

HSI16 calibration using LSE

The primary purpose of connecting the LSE to the channel 1 input capture of TIM16 and TIM17 and to the channel 2 input capture of LPTIM1, is to be able to precisely measure the HSI16 frequency. When using TIM16 or TIM17 for this purpose the HSI16 must be used as system clock source.

The number of HSI16 clock counts between two edges of the LSE signal provides a measure of the internal clock period. Taking advantage of the high precision of LSE crystal (typically a few hundred ppm), the internal clock frequency can be determined with the similar resolution depending on the measurement time. The HSI16 can be trimmed to compensate the process, temperature and/or voltage related frequency deviations.

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, the better the measurement.

The HSI16 oscillator has dedicated user-accessible calibration bits (HSITRIM) for this purpose.

HSI16 calibration using HSE32

The HSE32 must be used as system clock and the timer input capture must be connected to HSI16/256. TIM16 and 17 channel 1 input capture, as well and the LPTIM2 input capture 2, are connected to the divided oscillator only when TIMICSEL is different from 0b0xx in the *RCC peripherals independent clock configuration register 1 (RCC_CCIPR1)*.

LSI calibration using HSE32

The calibration of the LSI follows the same principle as the HSI16 calibration, 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 HSE32 as system clock source. The number of HSE32 clock counts between 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 HSE32/LSI ratio). The precision is therefore closely related to the ratio between the two clock sources, the higher the ratio, the better the measurement.

12.4.19 Audio synchronization

The audio synchronization system provides capture compare information between the Bluetooth® LE radio packet timing and audio clock.

- 20-bit programmable free-running up-counter
- Auto-reload
- Clock prescaler
- Compare
- Input capture
- Capture period
- Interrupt:
  - Input capture
  - Compare
  - Compare error

Before enabling the audio synchronization counter the auto-reload, clock prescaler, and capture prescaler must be provided.

- The clock prescaler is used to provide a lower speed clock to the counter.
- The auto-reload is used to define the audio synchronization counter period.
- The capture prescaler is used to define the capture period. The capture period is a multiple of the audio synchronization counter period.

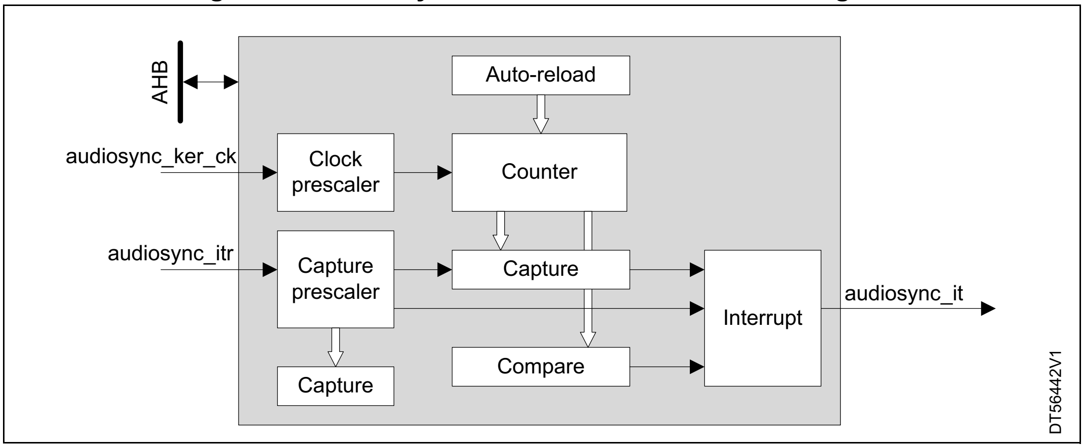
A capture value of the counter and the capture prescaler value are updated on the first audio synchronization trigger event in the capture period. Any subsequent audio trigger synchronization event during this capture period is discarded. When enabled, an associated capture event interrupt can be generated.

After enabling the audio synchronization counter in CEN, the capture prescaler starts counting the capture period only after having received a first synchronization from a Bluetooth® LE radio packet.

A capture error flag is set when no audio trigger synchronization event occurs during the capture period. When enabled, an associated capture error interrupt event can be generated.

The compare can be used to generate an interrupt when the counter reaches the compare value.

Figure 38. Audio synchronization counter block diagram

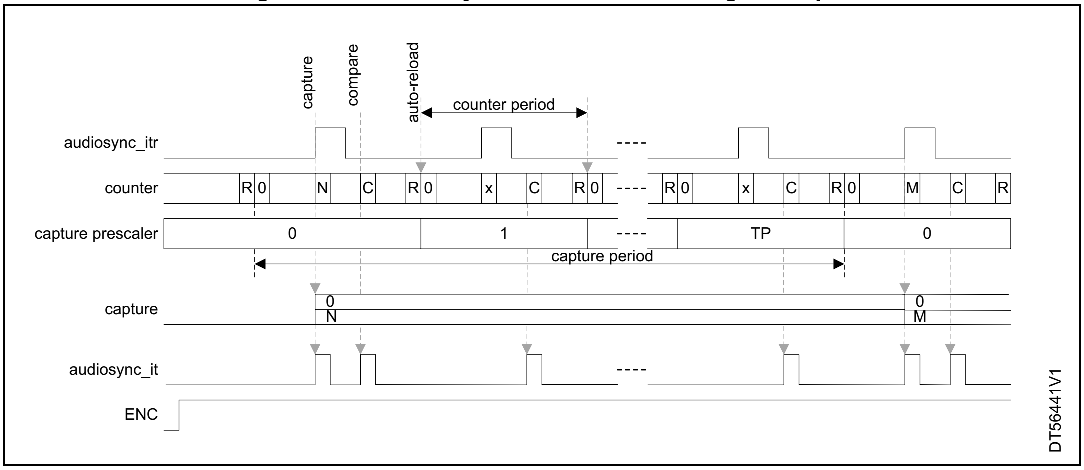


How to use the audio synchronization counter

In the example below the following parameters have been used:

- auto-reload = R (counter period)
- capture prescaler = TP (capture period)
- compare = C

Figure 39. Audio synchronization timing example



The capture value is updated only the first audiosync_itr event in the capture period. In the first capture period counter value N and capture prescaler value 0 are captured. The capture period counting is only started after the very first audiosync_itr event after enabling the audio synchronization counter with ENC. The other audiosync_itr events during this capture period are discarded. In the next capture period counter value M and capture prescaler value 0 are captured.

The counter drift (in ppm) is calculated by software: Drift = 10^6 x (N - M) / (R x TP)

The trigger event may be delayed due to missing receive packets at the 2.4 GHz radio. This delay has to be compensated for in the calculation.

Error handling

When no audiosync_itr event has been received during the capture period, a capture error interrupt is generated when enabled.

Audio synchronization associated functionality

The audiosync_ker_ck is gated with the audio synchronization counter enable bit CEN.

The audiosync_itr must be connected to NVIC71.

12.4.20 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, the clock is active after two cycles of the peripheral bus clock.

Caution: Just after enabling the peripheral clock, the software must wait for these two clock cycles before accessing the peripheral registers.

Peripherals clock gating in Sleep and Stop modes

When a peripheral is enabled in RCC_AHBxENR or RCC_APBxENR registers, its bus and kernel clocks can be automatically gated off when the device is in Sleep and Stop modes, by clearing the peripheral SMEN bit in the RCC_AHBxSMENR or RCC_APBxSMENR registers. Both peripheralEN and peripheralSMEN bit of the peripheral must be set to keep the peripheral bus and kernel clocks on in Sleep and Stop modes. In Stop modes, the peripheral bus and kernel clocks are further more active only upon the peripheral clock requests. Except for the 2.4 GHz RADIO, which requests its bus clock independently from the setting in the RADIOEN and RADIOSMEN register bits.

For the 2.4 GHz RADIO the bus clock runs only in Sleep and Stop modes when the STRADIOCLKON is set, or when the 2.4 GHz RADIO is active and RADIOEN and RADIOSMEN are set.

Caution: All peripherals on the same bus, with the SMEN bit set, get a clock when an autonomous peripheral on the same bus requests its clock. Peripherals that are not supposed to be clocked in Stop mode must have their SMEN bit cleared.

Caution: The SMEN bit must be set to allow the generation of an interrupt capable to wake up the device from Sleep and Stop mode. This is not necessary when the peripheral wake-up interrupt is generated though the EXTI (GPIO, COMP and PVD).

Peripherals clock gating and autonomous mode in Stop 0/1 modes

Some peripherals support autonomous mode (refer to *Table 104: Autonomous peripherals*). They 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 upon the configuration, either a DMA request or an interrupt can be associated to the peripheral event.

When the system enters low-power mode (Stop and Standby) and an autonomous peripheral bus clock request is active or upon an autonomous peripheral bus clock request during Stop mode, Stop 0 mode is entered and the HSI16 oscillator is kept active or woken up and selected as SYSCLK and the bus clocks for all peripherals, with their clock enabled in peripheralEN and peripheralSMEN, are activated.

Note: As soon as the CPU enters Sleepdeep, the system enter Stop mode and the autonomous mode operation peripheral bus clock and SYSCLK is switched to HSI16 at 16 MHz. If autonomous peripheral operation with higher bus clock frequencies is needed, the CPU must enter Sleep and keep the system in Run with the configured Run mode SYSCLK clock frequency.

If the autonomous peripheral is configured with DMA requests enabled, a data transfer is performed thanks to the peripheral bus clock. The bus clocks as well as the oscillator (HSI16) are automatically switched off as soon as the transfer is finished, and no other peripheral requests its bus clock.

If the autonomous peripheral is configured with interrupt enabled, the interrupt wakes up the device into Run mode.

The autonomous peripherals are autonomous in Stop 0 with the GPDMA1 and SRAM1, SRAM2 and are autonomous in Stop 1 mode on their kernel clock.

Table 104 shows the list of peripherals with autonomous mode capability.

Table 104. Autonomous peripherals

Domain	Peripheral	Autonomous in Stop 0 mode	Associated DMA	Associated SRAM
AHB1, APB1, APB2	U(S)ARTx (x = 1 to 2)	Yes(1)	GPDMA1	SRAM1, SRAM2
	SPI1			
	I2C1			
	LPTIM2			
AHB4, APB7	LPUART1			
	SPI3			
	I2C3			
	LPTIM1			
	ADC4			
AHB5	2.4 GHz RADIO + RXTXRAM	Yes(2)	-	-

1. Enabled when both peripheralEN and peripheralSMEN bits of the peripheral are set.
2. Enabled when the 2.4 GHz RADIO bit STRADIOCLKON and BBCLKEN bits are set and the 2.4 GHz RADIO is active. Available only in Stop 0 range 1.

For peripherals the autonomous mode is enabled in Stop 0 and Stop 1 modes if both peripheralEN and peripheralSMEN bits of the peripheral are set.

For the 2.4 GHz RADIO sleep timer it is operational down to standby with retention mode. Waking up from the sleep timer puts the system in Stop 0 mode and enables the 2.4 GHz RADIO bus clock. The 2.4 GHz RADIO active mode is enabled by software when in addition BBCLKEN bit is set, and allows autonomous operation in Stop 0 range 1 mode.

If an autonomous peripheral requests its kernel clock in Stop mode, the internal oscillator (HSI16) is woken up if it was off and the kernel clock is propagated only to the peripheral requesting it and the peripheralEN and peripheralSMEN bits are set. When the peripheral releases its kernel clock request, the HSI16 is switched off if no other peripheral requests it. Only the 2.4 GHz RADIO uses HSE32 as kernel clock which is woken-up if it was off by a 2.4 GHZ RADIO each time it is woke-up by the sleep timer.

If an autonomous peripheral requests its bus clock in Stop mode and the peripheral peripheralEN and peripheralSMEN bits ate set, the internal oscillator (HSI16) is woken up if it was off and the system clock is propagated to all peripherals on the associated AHB bus configured with both peripheralEN and peripheralSMEN bits set.

Caution: The bus clock propagates to all peripherals (autonomous and non-autonomous) on the same AHB bus with both peripheralEN and peripheralSMEN bits set.

HSI16 can be forced to remain ON in Stop mode, by configuring HSIKERON in the RCC_CR. In this case, the oscillator is propagated only to the peripheral kernel clocks of the enabled autonomous peripherals which select this oscillator as kernel clock. This allows the peripheral baudrates or conversion rates increase, as there is no need to wait for the oscillator wake-up time when the peripheral requests its kernel clock.

The LSE or LSI selected as peripheral kernel clock remains always ON in Stop modes.

12.5 RCC security and privilege functional description

12.5.1 RCC TrustZone® security protection modes

TrustZone security is activated by the TZEN user option bit in the FLASH_OPTR. The RCC is able to secure RCC configuration and status bits from being modified by non-secure accesses.

This is configured through the *RCC Backup domain control register (RCC_BDCR1)* to prevent non-secure access to read or modify the following features:

- HSE32, HSECSS, HSI16, LSI, LSE, LSECSS, LSCO, configuration and status bits
- PLL1, AHB and APB prescalers configuration and status bits
- system clock (SYSCLK) source clock selection and status bits
- MCO clock output configuration bits
- 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, see *Table 105*.

A peripheral is secure when:

- For securable peripherals by GTZC-TZSC (TrustZone security controller), by the SEC security bit in the secure configuration registers corresponding to this peripheral.
- For TrustZone-aware peripherals, a security feature of this peripheral is enabled through its dedicated bits.

Table 105 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 allowed only 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), all registers are non-secure. RCC_SECCFGR write accesses are ignored.

Table 105. RCC security configuration summary

Configuration bit in RCC_SECCFGR	Secured bits	Corresponding register
HSISEC	HSION, HSIKERON, HSIRDY	RCC_CR
	HSICAL, HSITRIM	RCC_ICSCR3
	HSIRDYIE	RCC_CIER
	HSIRDYIF	RCC_CIFR
	HSIRDYC	RCC_CICR
HSESEC	HSEON, HSERDY, HSECSSON, HSEPRE	RCC_CR
	HSERDYIE	RCC_CIER
	HSERDYIF, HSECSSF	RCC_CIFR
	HSERDYC, HSECSSC	RCC_CICR
	HSETRIM	RCC_ECSCR1
LSISEC	LSI1ON, LSI1RDY, LSI1PREDIV, LSI2ON, LSI2RDY, LSCOSEL, LSCOEN	RCC_BDCR1
	LSI2MODE, LSI2CFG	RCC_BDCR2
	LSI1RDYIE, LSI2RDYIE	RCC_CIER
	LSI1RDYIF, LSI2RDYIF	RCC_CIFR
	LSI1RDYC, LSI2RDYC	RCC_CICR

Table 105. RCC security configuration summary (continued)

Configuration bit in RCC_SECCFGR	Secured bits	Corresponding register
LSESEC	LSECSSON, LSECSSD, LSEDRV, LSEBYP, LSERDY, LSEON, LSEGFON, LSESYSRDY, LSESYSEN, LSCOSEL, LSCOEN	RCC_BDCR1
	LSERDYIE	RCC_CIER
	LSERDYF	RCC_CIFR
	LSERDYC	RCC_CICR
SYSCLKSEC	SW, SWS, MCOSEL, MCOPRE	RCC_CFGR1
	SYSTICKSEL	RCC_CCIPR1
	VOS	PWR_VOSR
PRESCSEC	HPRE, PPRE1, PPRE2	RCC_CFGR2
	PPRE7	RCC_CFGR3
	HPRE5, HDIV5	RCC_CFGR4
PLL1SEC	PLL1SRC, PLL1RGE, PLL1FRACEN, PLL1M, PLL1PEN, PLL1QEN, PLL1REN, PLL1RCLKPRE, PLL1RCLKSTEP, PLL1RCLKPRERDY	RCC_PLL1CFGR
	PLL1N, PLL1P, PLL1Q, PLL1R	RCC_PLL1DIVR
	PLL1FRACN	RCC_PLL1FRACR
	PLL1RDY, PLL1ON	RCC_CR
	PLL1RDYIE	RCC_CIER
	PLL1RDYF	RCC_CIFR
	PLL1RDYC	RCC_CICR
RMVFSEC	RMVF	RCC_CSR

12.5.2 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 only with privileged access. RCC_PRIVCFGR can be read by secure and non secure, privileged and unprivileged access.

The SPRIV bit in RCC_PRIVCFGR can be written only with secure privileged access. This bit configures the privileged access of all RCC secure functions (as defined by *RCC Backup domain control register (RCC_BDCR1)* or by the GTZC-TZSC 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-TZSC 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).

12.6 RCC low-power modes

- AHB and APB peripheral clocks, including DMA clock, can be disabled by software.
- Sleep mode stops the CPU hclk1 clock. The memory interface clocks (flash memory, cache 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 all the clocks in the Core domain and disable the PLL1, HSI16 and HSE32 oscillators. However, HSI16 can be switched on if the peripheral requests it for kernel clock operation purpose, or to generate a wake-up interrupt (see *Section 12.4.20* for more details). LSI and LSE can also remain active in Stop modes.
- Standby mode stop all the clocks in the Core domain and disable the PLL1, HSI16, HSE32 oscillators. LSI and LSE can remain active in Standby modes.

Stopping the system clock in Stop and Standby modes can be overridden for debugging by setting the DBG_STOP, and/or DBG_STANDBY. For more details, refer to *Section 43.12.4: Low-power mode emulation.*

Note: When using debug Stop mode (DBG_STOP) a SysTick event wakes up the device. It is good practice to disable the SysTick, before the CPU enters SleepDeep.

When entering and exiting Stop modes, the system clock is HSI16, whose user trim is kept.

When leaving the Standby modes, the system clock is HSI16. The user trim is lost.

If a flash memory programming operation is ongoing, Stop and Standby mode entry is delayed until the flash memory interface access is finished. If an access to the APB domain is ongoing, Stop and Standby modes entry is delayed until the APB access is finished. If an autonomous peripheral bus clock request is active, Stop 0 mode is entered. If an other low-power mode (Stop 1 and Standby) is selected in LPMS, entry to the selected low-power mode is delayed until the autonomous peripheral bus clock request is released.

12.7 RCC interrupts

Table 106 summarizes the interrupt sources and the way to control them.

Table 106. Interrupt sources and control

Interrupt vector	Interrupt event flag	Description	Enable control bit(s)	Interrupt clear method	Exit from Sleep mode	Exit from Stop, Standby modes
RCC	LSI1RDYF	LSI1 ready	LSI1RDYIE and LSISEC = 0	Set LSI1RDYC to 1	Yes	No
	LSI2RDYF	LSI2 ready	LSI2RDYIE and LSISEC = 0	Set LSI2RDYC to 1		
	LSERDYF	LSE ready	LSERDYIE and LSESEC = 0	Set LSERDYC to 1		
	HSIDRYF	HSI16 ready	HSIDRYIE and HSISEC = 0	Set HSIRDYC to 1		
	HSERDYF	HSE32 ready	HSERDYIE and HSESEC = 0	Set HSERDYC to 1		
	PLL1RDYF	PLL1 ready	PLL1RDYIE and PLL1SEC = 0	Set PLL1RDYC to 1		
RCC_S(1)	LSI1RDYF	LSI1 ready	LSI1RDYIE and LSISEC = 1	Set LSI1RDYC to 1	Yes	No
	LSI2RDYF	LSI2 ready	LSI2RDYIE and LSISEC = 1	Set LSI2RDYC to 1		
	LSERDYF	LSE ready	LSERDYIE and LSESEC = 1	Set LSERDYC to 1		
	HSIDRYF	HSI16 ready	HSIDRYIE and HSISEC = 1	Set HSIRDYC to 1		
	HSERDYF	HSE32 ready	HSERDYIE and HSESEC = 1	Set HSERDYC to 1		
	PLL1RDYF	PLL1 ready	PLL1RDYIE and PLL1SEC = 1	Set PLL1RDYC to 1		
TAMP	ITAMP3F(2)	LSECSS failure	LSECSSON and ITAMP3E(2) and ITAMP3IE(2)	Set CITAMP3F(2) to 1	Yes	Yes
NMI	HSECSSF	HSECSS failure	HSECSSON(3)	Set HSECSSC to 1	Yes	No
AUDIO SYNC	CAF	Capture event	CAIE	Write CAF to 0	Yes	No
	COF	Compare event	COIE	Write COF to 0		
	CAFE	Capture error event	CAEIE	Write CAEF to 0		

1. The RCC secure interrupt vector is used only when TrustZone is enabled.
2. The LSECSS failure event (LSECSSD) is connected to TAMP internal tamper 3. 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.

3. It is not possible to mask this interrupt when the security system feature is enabled (HSECSSON = 1).

12.8 RCC registers

12.8.1 RCC clock control register (RCC_CR)

Address offset: 0x000

Reset value: 0x0000 0500

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.	PLL1RDY	PLL1ON	Res.	Res.	Res.	HSEPRE	HSECSSON	Res.	HSERDY	HSEON
						r	rw				rw	rs		r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HSIRDY	HSIKERON	HSION	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
					r	rw	rw								

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **PLL1RDY**: PLL1 clock ready flag

Set by hardware to indicate that the PLL1 is locked.

Access can be secured by RCC PLL1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: PLL1 unlocked

1: PLL1 locked (PLL1RDY remains set when PLL1 is selected as sysclk and PLL1 is disabled by HSECSS failure).

Bit 24 **PLL1ON**: PLL1 enable

Set and cleared by software to enable the main PLL.

Cleared by hardware when entering Stop or Standby modes and when PLL1 on HSE32 is selected as sysclk, on a HSECSS failure.

This bit cannot be reset if the PLL1 clock is used as the system clock.

Access can be secured by RCC PLL1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: PLL1 off

1: PLL1 on

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

Bit 20 **HSEPRE**: HSE32 clock for SYSCLK prescaler

Set and cleared by software to control the division factor of the HSE32 clock for SYSCLK.

Access can be secured by RCC HSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: HSE32 not divided, SYSCLK = HSE32

1: HSE32 divided, SYSCLK = HSE32/2

Bit 19 **HSECSSON**: HSE32 clock security system enable

Set by software to enable the HSE32 clock security system. When HSECSSON is set, the clock detector is enabled by hardware when the HSE32 oscillator is ready and disabled by hardware if a HSE32 clock failure is detected. This bit is set only and is cleared by reset.

Access can be secured by RCC HSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: HSE32 clock security system off (clock detector off)

1: HSE32 clock security system on (clock detector on if the HSE32 oscillator is stable, off if not).

Bit 18 Reserved, must be kept at reset value.

Bit 17 **HSERDY**: HSE32 clock ready flag

Set by hardware to indicate that the HSE32 oscillator is stable. This bit is set both when HSE32 is enabled by software by setting HSEON and when requested as kernel clock by the 2.4 GHz RADIO.

Access can be secured by RCC HSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: HSE32 oscillator not ready

1: HSE32 oscillator ready to be used by the CPU (HSERDY remains set when HSE32 is disabled by HSECSS failure).

Bit 16 **HSEON**: HSE32 clock enable

Set and cleared by software.

Cleared by hardware to stop the HSE32 clock for the CPU when entering Stop and Standby modes and on a HSECSS failure.

When the HSE32 is used as 2.4 GHz RADIO kernel clock, enabled by RADIOEN and RADIOSMEN and the 2.4 GHz RADIO is active, HSEON is not cleared when entering low power mode. In this case, only Stop 0 mode is entered as low power mode.

This bit cannot be reset if the HSE32 oscillator is used directly or indirectly as the system clock.

Access can be secured by RCC HSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: HSE32 oscillator not requested by the CPU.

1: HSE32 oscillator ON

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

Bit 10 **HSIRDY**: HSI16 clock ready flag

Set by hardware to indicate that HSI16 oscillator is stable. This bit is set only when HSI16 is enabled by software by setting HSION.

Access can be secured by RCC HSISEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

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 oscillator on even in Stop modes.

Keeping the HSI16 oscillator on in Stop modes allows the communication speed not to be reduced by the HSI16 oscillator startup time. This bit has no effect on register bit HSION value.

Cleared by hardware when entering Standby modes.

Refer to *Peripherals clock gating and autonomous mode* for more details.

Access can be secured by RCC HSISEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

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 when entering Stop and Standby modes.

Set by hardware to force the HSI16 oscillator on when exiting Stop and Standby modes.

Set by hardware to force the HSI16 oscillator on in case of clock security failure of the HSE32 crystal oscillator.

This bit is set by hardware if the HSI16 is used directly or indirectly as system clock.

Access can be secured by RCC HSISEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: HSI16 oscillator off

1: HSI16 oscillator on

Bits 7:0 Reserved, must be kept at reset value.

12.8.2 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

Access to this register can be protected by RCC HSISEC and RCC SPRIV or RCC NSPRIV.

31	30	29	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]**: HSI16 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 HSI16.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSICAL[11:0]**: HSI16 clock calibration

These bits are initialized at startup with the factory-programmed HSI16 calibration value. When HSITRIM[4:0] is written, HSICAL[11:0] is updated with the sum of HSITRIM[4:0] and the initial factory trim value.

12.8.3 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

One or two wait states are inserted only if the access occurs during clock source switch.

Access to this register can be protected by RCC SYSCLKSEC and RCC SPRIV or RCC NSPRIV.

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.	Res.	Res.	SWS[1:0]		SW[1:0]	
												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: sysclkpre system clock after PLL1RCLKPRE division selected
0010: reserved
0011: HSI16 clock selected
0100: HSE32 clock selected
0101: pll1rclk clock selected
0110: LSI clock selected
0111: LSE clock selected
1000: pll1pclk clock selected
1001: pll1qclk clock selected
1010: hclk5 clock selected
others: reserved

Note: This clock output may have some truncated cycles at startup or during MCO clock source switching.

Bits 23:4 Reserved, must be kept at reset value.

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: HSI16 oscillator used as system clock
01: reserved
10: HSE32 or HSE32/2, as defined by HSEPRE, used as system clock
11: pll1rclk used as system clock

Bits 1:0 **SW[1:0]**: system clock switch

Set and cleared by software to select system clock source (SYSCLK).
Cleared by hardware when entering Stop and Standby modes
When selecting HSE32 directly or indirectly as system clock and HSE32 oscillator clock security fails, cleared by hardware.
00: HSI16 selected as system clock
01: reserved
10: HSE32 or HSE32/2, as defined by HSEPRE, selected as system clock
11: pll1rclk selected as system clock

12.8.4 RCC clock configuration register 2 (RCC_CFGR2)

Address offset: 0x020

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.

Access to this register can be protected by RCC PRESCSEC and RCC SPRIV or RCC NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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]			Res.	HPRE[2:0]		
					rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:11 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 = hclk1 not divided
100: pclk2 = hclk1 divided by 2
101: pclk2 = hclk1 divided by 4
110: pclk2 = hclk1 divided by 8
111: pclk2 = hclk1 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 = hclk1 not divided
100: pclk1 = hclk1 divided by 2
101: pclk1 = hclk1 divided by 4
110: pclk1 = hclk1 divided by 8
111: pclk1 = hclk1 divided by 16

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **HPRE[2:0]**: AHB1, AHB2 and AHB4 prescaler

Set and cleared by software to control the division factor of the AHB1, AHB2 and AHB4 clock (hclk1).

Caution: The software must limit the incremental frequency step by setting these bits correctly to ensure that the hclk1 maximum incremental frequency step does not exceed the maximum allowed incremental frequency step (for more details, refer to *Table 101*). 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.

0xx: hclk1 = SYSCLK not divided
100: hclk1 = SYSCLK divided by 2
101: hclk1 = SYSCLK divided by 4
110: hclk1 = SYSCLK divided by 8
111: hclk1 = SYSCLK divided by 16

12.8.5 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.

Access to this register can be protected by RCC PRESCSEC and RCC SPRIV or RCC NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PPRE7[2:0]			Res.	Res.	Res.	Res.
									rw	rw	rw				

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **PPRE7[2:0]**: APB7 prescaler
Set and cleared by software to control the division factor of the APB7 clock (pclk7).
0xx: hclk1 not divided
100: hclk1 divided by 2
101: hclk1 divided by 4
110: hclk1 divided by 8
111: hclk1 divided by 16

Bits 3:0 Reserved, must be kept at reset value.

12.8.6 RCC PLL1 configuration register (RCC_PLL1CFGR)

Address offset: 0x028

Reset value: 0x0000 0000

Access: no wait state; word, half-word and byte access.

Access to this register can be protected by RCC PLL1SEC and RCC SPRIV or RCC NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL1RCLKPRERDY	PLL1RCLKPRESTEP	PLL1RCLKPRE	Res.	PLL1REN	PLL1QEN	PLL1PEN
									r	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.	PLL1M[2:0]			Res.	Res.	Res.	PLL1FRACEN	PLL1RGE[1:0]		PLL1SRC[1:0]	
					rw	rw	rw				rw	rw	rw	rw	rw

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

Bit 22 **PLL1RCLKPRERDY**: pll1rclkpre not divided ready.
Set by hardware after PLL1RCLKPRE has been set from divided to not divide, to indicate that the pll1rclk not divided is available on sysclkpre.
0: pll1rclk divided
1: pll1rclk not divided ready

Bit 21 **PLL1RCLKPRESTEP**: pll1rclk clock for SYSCLK prescaler division step selection
Set and cleared by software to control the division step of the pll1rclk clock for SYSCLK.
0: pll1rclk 2-step division
1: pll1rclk 3-step division

Bit 20 **PLL1RCLKPRE**: pll1rclk clock for SYSCLK prescaler division enable
Set and cleared by software to control the division of the pll1rclk clock for SYSCLK.
0: pll1rclk not divided, sysclkpre = pll1rclk
1: pll1rclk divided, sysclkpre = pll1rclk divided

Bit 19 Reserved, must be kept at reset value.

Bit 18 **PLL1REN**: PLL1 DIVR divider output enable
Set and cleared by software to enable the pll1rclk output of the PLL1. This bit cannot be cleared when pll1rclk is used as system clock as indicated in SWS.
To save power, PLL1REN and PLL1R bits must be set to 0 when the pll1rclk is not used.
0: pll1rclk output disabled
1: pll1rclk output enabled

Bit 17 **PLL1QEN**: PLL1 DIVQ divider output enable
Set and reset by software to enable the pll1qclk output of the PLL1.
To save power, PLL1QEN and PLL1Q bits must be set to 0 when the pll1qclk is not used.
0: pll1qclk output disabled
1: pll1qclk output enabled

Bit 16 **PLL1PEN**: PLL1 DIVP divider output enable
Set and reset by software to enable the pll1pclk output of the PLL1.
To save power, PLL1PEN and PLL1P bits must be set to 0 when the pll1pclk is not used.
0: pll1pclk output disabled
1: pll1pclk output enabled

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

Bits 10:8 **PLL1M[2:0]**: 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 field can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
000: division by 1 (bypass)
001: division by 2
010: division by 3
...
111: division by 8

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 *PLL1 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 field can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
00-01-10: PLL1 input (ref_ck) clock range frequency between 4 and 8 MHz
11: PLL1 input (ref_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. This field can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
Cleared by hardware when entering Stop or Standby modes.
00: no clock sent to PLL1
01: reserved
10: HSI16 clock selected as PLL1 clock entry
11: HSE32 clock after HSEPRE divider selected as PLL1 clock entry

Note: In order to save power, when no PLL1 clock is used, the value of PLL1SRC must be 0.

12.8.7 RCC PLL1 dividers register (RCC_PLL1DIVR)

Address offset: 0x034

Reset value: 0x0101 0280

Access: no wait state; word, half-word and byte access.

Access to this register can be protected by RCC PLL1SEC and RCC SPRIV or RCC NSPRIV.

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 pll1rclk clock. Division factors are forbidden if VCO frequency / (2 x (TRUNC(division factor / 2)) > pll1rclk maximum frequency.
These bits can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
0000000: pll1rclk = VCO output frequency
0000001: pll1rclk = VCO output frequency / 2 (default after reset)
0000010: pll1rclk = VCO output frequency / 3
0000011: pll1rclk = VCO output frequency / 4
...
1111111: pll1rclk = VCO output frequency / 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 pll1qclk clock. Division factors are forbidden if VCO frequency / (2 x (TRUNC(division factor / 2)) > pll1qclk maximum frequency.
These bits can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
0000000: pll1qclk = VCO output frequency
0000001: pll1qclk = VCO output frequency / 2 (default after reset)
0000010: pll1qclk = VCO output frequency / 3
0000011: pll1qclk = VCO output frequency / 4
...
1111111: pll1qclk = VCO output frequency / 128

Bits 15:9 **PLL1P[6:0]**: PLL1 DIVP division factor

Set and reset by software to control the frequency of the pll1pclk clock. Division factors are forbidden if VCO frequency / (2 x (TRUNC(division factor / 2)) > pll1pclk maximum frequency.

These bits can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).

0000000: pll1pclk = VCO output frequency
0000001: pll1pclk = VCO output frequency / 2 (default after reset)
0000010: pll1pclk = VCO output frequency
0000011: pll1pclk = VCO output frequency / 4
...
1111111: pll1pclk = VCO output frequency / 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 PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).

0x003: multiplication factor for PLL1 VCO = 4
0x004: multiplication factor for PLL1 VCO = 5
0x005: multiplication factor for PLL1 VCO = 6
...
0x080: multiplication factor for PLL1 VCO = 129 (default after reset)
...
0x1FF: multiplication factor for PLL1 VCO = 512
others: reserved

VCO output frequency = F_{ref_ck} x multiplication factor for PLL1 VCO, when fractional value 0 has been loaded into PLL1FRACN, with:

- multiplication factor for PLL1 VCO between 4 and 512
- input frequency F_{ref_ck} between 4 and 16 MHz

12.8.8 RCC PLL1 fractional divider register (RCC_PLL1FRACR)

Address offset: 0x038

Reset value: 0x0000 0000

Access: no wait state; word, half-word and byte access.

Access to this register can be protected by RCC PLL1SEC and RCC SPRIV or RCC NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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_{ref_ck} x [multiplication factor for PLL1 VCO + (PLL1FRACN / 2^{13})], with:

- Multiplication factor for PLL1 VCO must be between 4 and 512
- PLL1FRACN can be between 0 and 2^{13}- 1
- The input frequency F_{ref_ck} must be between 4 and 16 MHz

To change the used fractional value on-the-fly even if the PLL1 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.

12.8.9 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.	LSI2RDYIE
															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.	PLL1RDYIE	Res.	HSERDYIE	HSIRDYIE	Res.	LSERDYIE	LSI1RDYIE
									rw		rw	rw		rw	rw

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

Bit 16 **LSI2RDYIE**: LSI2 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the LSI2 oscillator stabilization.

Access can be secured by RCC LSISEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: LSI2 ready interrupt disabled

1: LSI2 ready interrupt enabled

If the LSI2 is ready (LSI2RDY in RCC_BDCR1 is 1) before enabling LSI2RDYIE, the LSI2RDYF in RCC_CIFR is not set, and no interrupt is generated.

Note that this bit is reserved on STM32WBA50/52xx devices.

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

Bit 6 **PLL1RDYIE**: PLL1 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by PLL1 lock.

Access can be secured by RCC PLL1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: PLL1 lock interrupt disabled

1: PLL1 lock interrupt enabled

If the PLL1 is ready (PLL1RDY in RCC_CR is 1) before enabling PLL1RDYIE, the PLL1RDYF in RCC_CIFR is not set, and no interrupt is generated.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **HSERDYIE**: HSE32 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the HSE32 oscillator stabilization.

Access can be secured by RCC HSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: HSE32 ready interrupt disabled

1: HSE32 ready interrupt enabled

If the HSE is ready (HSERDY in RCC_CR is 1) before enabling HSERDYIE, the HSERDYF in RCC_CIFR is not set and no interrupt is generated.

Bit 3 **HSIRDYIE**: HSI16 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the HSI16 oscillator stabilization.

Access can be secured by RCC HSISEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: HSI16 ready interrupt disabled

1: HSI16 ready interrupt enabled

If the HSI16 is ready (HSIRDY in RCC_CR is 1) before enabling HSIRDYIE, the HSIRDYF in RCC_CIFR is not set, and no interrupt is generated.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LSERDYIE**: LSE ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the LSE oscillator stabilization.

Access can be secured by RCC LSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: LSE ready interrupt disabled

1: LSE ready interrupt enabled

If the LSE is ready (LSERDY in RCC_BDCR1 is 1) before enabling LSERDYIE, the LSERDYF in RCC_CIFR is not set, and no interrupt is generated.

Bit 0 **LSI1RDYIE**: LSI1 ready interrupt enable

Set and cleared by software to enable/disable interrupt caused by the LSI1 oscillator stabilization.

Access can be secured by RCC LSISEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: LSI1 ready interrupt disabled

1: LSI1 ready interrupt enabled

If the LSI1 is ready (LSI1RDY in RCC_BDCR1 is 1) before enabling LSI1RDYIE, the LSI1RDYF in RCC_CIFR is not set, and no interrupt is generated.

12.8.10 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.	LSI2RDYF
															r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HSECSSF	Res.	Res.	Res.	PLL1RDYF	Res.	HSERDYF	HSIRDYF	Res.	LSERDYF	LSI1RDYF
					r				r		r	r		r	r

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

Bit 16 **LSI2RDYF**: LSI2 ready interrupt flag

Set by hardware when the LSI2 clock becomes stable and LSI2RDYIE is set.

Cleared by software setting the LSI2RDYC bit.

Access can be secured by RCC LSISEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: no clock ready interrupt caused by the LSI2 oscillator

1: clock ready interrupt caused by the LSI2 oscillator

Note that this bit is reserved on STM32WBA50/52xx devices.

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

Bit 10 **HSECSSF**: HSE32 clock security system interrupt flag

Set by hardware when a clock security failure is detected in the HSE32 oscillator.

Cleared by software setting the HSECSSC bit.

Access can be secured by RCC HSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: no clock security interrupt caused by HSE32 clock failure

1: clock security interrupt caused by HSE32 clock failure

Bits 9:7 Reserved, must be kept at reset value.

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.
Access can be secured by RCC PLL1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: no clock ready interrupt caused by PLL1 lock
1: clock ready interrupt caused by PLL1 lock

Bit 5 Reserved, must be kept at reset value.

Bit 4 **HSERDYF**: HSE32 ready interrupt flag

Set by hardware when the HSE32 clock becomes stable and HSERDYIE is set.
Cleared by software setting the HSERDYC bit.
Access can be secured by RCC HSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: no clock ready interrupt caused by the HSE32 oscillator
1: clock ready interrupt caused by the HSE32 oscillator

Bit 3 **HSIRDYF**: HSI16 ready interrupt flag

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.
Access can be secured by RCC HSISEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
Cleared by software setting the HSIRDYC bit.
0: no clock ready interrupt caused by the HSI16 oscillator
1: clock ready interrupt caused by the HSI16 oscillator

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LSERDYF**: LSE ready interrupt flag

Set by hardware when the LSE clock becomes stable and LSERDYIE is set.
Cleared by software setting the LSERDYC bit.
Access can be secured by RCC LSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: no clock ready interrupt caused by the LSE oscillator
1: clock ready interrupt caused by the LSE oscillator

Bit 0 **LSI1RDYF**: LSI1 ready interrupt flag

Set by hardware when the LSI1 clock becomes stable and LSI1RDYIE is set.
Cleared by software setting the LSI1RDYC bit.
Access can be secured by RCC LSISEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: no clock ready interrupt caused by the LSI1 oscillator
1: clock ready interrupt caused by the LSI1 oscillator

12.8.11 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.	LSI2RDYC
															w
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	HSECSSC	Res.	Res.	Res.	PLL1RDYC	Res.	HSERDYC	HSIRDYC	Res.	LSERDYC	LSI1RDYC
					w				w		w	w		w	w

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

Bit 16 **LSI2RDYC**: LSI2 ready interrupt clear

Writing this bit to 1 clears the LSI2RDYF flag. Writing 0 has no effect.

Access can be secured by RCC LSISEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

Note that this bit is reserved on STM32WBA50/52xx devices.

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

Bit 10 **HSECSSC**: High speed external clock security system interrupt clear

Writing this bit to 1 clears the HSECSSF flag. Writing 0 has no effect.

Access can be secured by RCC HSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

Bits 9:7 Reserved, must be kept at reset value.

Bit 6 **PLL1RDYC**: PLL1 ready interrupt clear

Writing this bit to 1 clears the PLL1RDYF flag. Writing 0 has no effect.

Access can be secured by RCC PLL1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **HSERDYC**: HSE32 ready interrupt clear

Writing this bit to 1 clears the HSERDYF flag. Writing 0 has no effect.

Access can be secured by RCC HSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

Bit 3 **HSIRDYC**: HSI16 ready interrupt clear

Writing this bit to 1 clears the HSIRDYF flag. Writing 0 has no effect.\

Access can be secured by RCC HSISEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LSERDYC**: LSE ready interrupt clear
Writing this bit to 1 clears the LSERDYF flag. Writing 0 has no effect.
Access can be secured by RCC LSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

Bit 0 **LSI1RDYC**: LSI1 ready interrupt clear
Writing this bit to 1 clears the LSI1RDYF flag. Writing 0 has no effect.
Access can be secured by RCC LSISEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

12.8.12 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.	Res.	Res.	TSC RST
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	CRC RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA1 RST
			rw												rw

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

Bit 16 **TSCRST**: TSC reset
Set and cleared by software.
Access can be secured by GTZC_TZSC TSCSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
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.
Access can be secured by GTZC_TZSC CRCSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: No effect
1: Reset CRC

Bits 11:1 Reserved, must be kept at reset value.

Bit 0 **GPDMA1RST**: GPDMA1 reset
Set and cleared by software.
Access can be secured by GPDMA1 SECx. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: No effect
1: Reset GPDMA1

12.8.13 RCC AHB2 peripheral reset register (RCC_AHB2RSTR)

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKARST	HSEMRST	SAESRST	RNGRST	HASHRST	AESRST
										rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOHRST	Res.	Res.	Res.	Res.	GPIOCRST	GPIOBRST	GPIOARST
								rw					rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **PKARST**: PKA reset
Set and cleared by software.
Access can be secured by GTZC_TZSC PKASEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: No effect
1: Reset PKA

Bit 20 **HSEMRST**: HSEM reset
Set and cleared by software.
Can only be accessed secure when one or more features in the HSEM is secure. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: No effect
1: Reset HSEM

Bit 19 **SAESRST**: SAES hardware accelerator reset

Set and cleared by software.

Access can be secured by GTZC_TZSC SAESSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: No effect

1: Reset SAES

Note that this bit is reserved on STM32WBA50xx devices.

Bit 18 **RNGRST**: Random number generator reset

Set and cleared by software.

Access can be secured by GTZC_TZSC RNGSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: No effect

1: Reset RNG

Bit 17 **HASHRST:** Hash reset

Set and cleared by software.

Access can be secured by GTZC_TZSC HASHSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: No effect

1: Reset HASH

Bit 16 **AESRST**: AES hardware accelerator reset

Set and cleared by software.

Access can be secured by GTZC_TZSC AESSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: No effect

1: Reset AES

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

Bit 7 **GPIOHRST**: IO port H reset

Set and cleared by software.

Access can be secured by GPIOH SECx. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: No effect

1: Reset IO port H

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **GPIOCRST**: IO port C reset

Set and cleared by software.

Access can be secured by GPIOC SECx. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: No effect

1: Reset IO port C

Bit 1 **GPIOBRST**: IO port B reset
Set and cleared by software.
Access can be secured by GPIOB SECx. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: No effect
1: Reset IO port B

Bit 0 **GPIOARST**: IO port A reset
Set and cleared by software.
Access can be secured by GPIOA SECx. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: No effect
1: Reset IO port A

12.8.14 RCC AHB4 peripheral reset register (RCC_AHB4RSTR)

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.	Res.	Res.	Res.	Res.	Res.	ADC4 RST	Res.	Res.	Res.	Res.	Res.
										rw					

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **ADC4RST**: ADC4 reset
Set and cleared by software.
Access can be secured by GTZC_TZSC ADC4SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: No effect
1: Reset ADC4 interface

Bits 4:0 Reserved, must be kept at reset value.

12.8.15 RCC AHB5 peripheral reset register (RCC_AHB5RSTR)

Address offset: 0x070

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.	PTACONV RST	RADIO RST
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PTACONVRST**: PTACONV reset

Set and cleared by software.

Access can be secured by GTZC_TZSC PTACONVSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: No effect

1: Reset PTACONV

Bit 1 is reserved on STM32WBA50/52xx devices.

Bit 0 **RADIORST**: 2.4 GHz RADIO reset

Set and cleared by software.

Access can be secured by GTZC_TZSC RADIOSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: No effect

1: Reset 2.4 GHz RADIO

12.8.16 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.	Res.	Res.	Res.	I2C1RST	Res.	Res.	Res.	USART2RST	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.	TIM3RST	TIM2RST
														rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **I2C1RST**: I2C1 reset

Set and cleared by software.

Access can be secured by GTZC_TZSC I2C1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: No effect

1: Reset I2C1

Bit 21 is reserved on STM32WBA50xx devices.

Bits 20:18 Reserved, must be kept at reset value.

Bit 17 **USART2RST**: USART2 reset

Set and cleared by software.

Access can be secured by GTZC_TZSC UART2SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: No effect

1: Reset USART2

Bit 17 is reserved on STM32WBA50xx devices.

Bits 16:2 Reserved, must be kept at reset value.

Bit 1 **TIM3RST**: TIM3 reset

Set and cleared by software.

Access can be secured by GTZC_TZSC TIM3SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: No effect

1: Reset TIM3

Bit 1 is reserved on STM32WBA50xx devices.

Bit 0 **TIM2RST**: TIM2 reset

Set and cleared by software.

Access can be secured by GTZC_TZSC TIM2SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: No effect

1: Reset TIM2

12.8.17 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

This register is reserved on STM32WBA50xx devices.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2 RST	Res.	Res.	Res.	Res.	Res.
										rw					

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2RST**: LPTIM2 reset

Set and cleared by software.

Access can be secured by GTZC_TZSC LPTIM2SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: No effect

1: Reset LPTIM2

Bits 4:0 Reserved, must be kept at reset value.

12.8.18 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.	Res.	SAI1 RST	Res.	Res.	TIM17 RST	TIM16 RST	Res.
										rw			rw	rw	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	USART1 RST	Res.	SPI1 RST	TIM1 RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw	rw											

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **SAI1RST**: SAI1 reset

Set and cleared by software.

Access can be secured by GTZC_TZSC SAI1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: No effect

1: Reset SAI1

Note that this bit is reserved on STM32WBA50/52xx devices.

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

Bit 18 **TIM17RST**: TIM17 reset

Set and cleared by software.

Access can be secured by GTZC_TZSC TIM17SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: No effect

1: Reset TIM17

Note that this bit is reserved on STM32WBA50xx devices.

Bit 17 **TIM16RST**: TIM16 reset

Set and cleared by software.

Access can be secured by GTZC_TZSC TIM16SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: No effect

1: Reset TIM16

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **USART1RST**: USART1 reset

Set and cleared by software.

Access can be secured by GTZC_TZSC USART1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: No effect

1: Reset USART1

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1RST**: SPI1 reset

Set and cleared by software.

Access can be secured by GTZC_TZSC SPI1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: No effect

1: Reset SPI1

Note that this bit is reserved on STM32WBA50xx devices.

Bit 11 **TIM1RST**: TIM1 reset

Set and cleared by software.

Access can be secured by GTZC_TZSC TIM1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: No effect

1: Reset TIM1

Bits 10:0 Reserved, must be kept at reset value.

12.8.19 RCC APB7 peripheral reset register (RCC_APB7RSTR)

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.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
COMP RST	Res.	Res.	Res.	LPTIM1 RST	Res.	Res.	Res.	I2C3 RST	LPUART1 RST	SPI3 RST	Res.	Res.	Res.	SYSCFG RST	Res.
rw				rw				rw	rw	rw				rw	

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **COMPRST**: COMP reset

Set and cleared by software.

Access can be secured by GTZC_TZSC COMPSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: No effect

1: Reset COMP

Note that this bit is reserved on STM32WBA50/52xx devices.

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

Bit 11 **LPTIM1RST**: LPTIM1 reset

Set and cleared by software.

Access can be secured by GTZC_TZSC LPTIM1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

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.

Access can be secured by GTZC_TZSC I2C3SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: No effect

1: Reset I2C3

Bit 6 **LPUART1RST**: LPUART1 reset

Set and cleared by software.

Access can be secured by GTZC_TZSC LPUART1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: No effect

1: Reset LPUART1

Bit 5 **SPI3RST**: SPI3 reset
Set and cleared by software.
Access can be secured by GTZC_TZSC SPI3SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
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.
Access can be secured by SYSCFG SYSCFGSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: No effect
1: Reset SYSCFG

Bit 0 Reserved, must be kept at reset value.

12.8.20 RCC AHB1 peripheral clock enable register (RCC_AHB1ENR)

Address offset: 0x088

Reset value: 0x8000 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
SRAM1 EN	Res.	Res.	Res.	Res.	Res.	Res.	GTZC1 EN	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFG EN	TSC EN
rw							rw							rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	CRC EN	Res.	Res.	Res.	FLASH EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA1 EN
			rw				rw								rw

Bit 31 **SRAM1EN**: SRAM1 bus clock enable
Set and reset by software.
Access can be secured by GTZC_MPCBB1 SECx, INVSECSTATE. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: SRAM1 bus clock disabled
1: SRAM1 bus clock enabled

Bits 30:25 Reserved, must be kept at reset value.

Bit 24 **GTZC1EN**: GTZC1 bus clock enable
Set and reset by software.
Can only be accessed secure when device is secure (TZEN = 1). When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: GTZC1 bus clock disabled
1: GTZC1 bus clock enabled
Note that this bit is reserved on STM32WBA50xx devices.

Bits 23:18 Reserved, must be kept at reset value.

Bit 17 **RAMCFGEN**: RAMCFG bus clock enable
Set and cleared by software.
Access can be secured by GTZC_TZSC RAMCFGSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: RAMCFG bus clock disabled
1: RAMCFG bus clock enabled

Bit 16 **TSCEN**: Touch sensing controller bus clock enable
Set and cleared by software.
Access can be secured by GTZC_TZSC TSCSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: TSC bus clock disabled
1: TSC bus clock enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CRCEN**: CRC bus clock enable
Set and cleared by software.
Access can be secured by GTZC_TZSC CRCSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: CRC bus clock disabled
1: CRC bus clock enabled

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **FLASHEN**: FLASH bus clock enable
Set and cleared by software. This bit can be disabled only when the flash memory is in power down mode.
Can only be accessed secured when the flash security state is secure. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: FLASH bus clock disabled
1: FLASH bus clock enabled

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

Bit 0 **GPDMA1EN**: GPDMA1 bus clock enable

Set and cleared by software.

Access can be secured by GPDMA1 SECx. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: GPDMA1 bus clock disabled

1: GPDMA1 bus clock enabled

12.8.21 RCC AHB2 peripheral clock enable register (RCC_AHB2ENR)

Address offset: 0x08C

Reset value: 0x4000 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.	SRAM2EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKAEN	HSEMEN	SAESEN	RNGEN	HASHEN	AESEN
	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.	GPIOHEN	Res.	Res.	Res.	Res.	GPIOCEN	GPIOBEN	GPIOAEN
								rw					rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SRAM2EN:** SRAM2 bus clock enable

Set and cleared by software.

Access can be secured by GTZC_MPCBB2 SECx, INVSECSTATE. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: SRAM2 bus clock disabled

1: SRAM2 bus clock enabled

Bits 29:22 Reserved, must be kept at reset value.

Bit 21 **PKAEN:** PKA bus clock enable

Set and cleared by software.

Access can be secured by GTZC_TZSC PKASEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: PKA bus clock disabled

1: PKA bus clock enabled

Bit 20 **HSEMEN:** HSEM bus clock enable

Set and cleared by software.

Can only be accessed secure when one or more features in the HSEM is secure. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: HSEM bus clock disabled

1: HSEM bus clock enabled

Bit 19 **SAESEN:** SAES bus clock enable

Set and cleared by software.

Access can be secured by GTZC_TZSC SAESSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: SAES bus clock disabled

1: SAES bus clock enabled

Note that this bit is reserved on STM32WBA50xx devices.

Bit 18 **RNGEN:** RNG bus and kernel clocks enable

Set and cleared by software.

Access can be secured by GTZC_TZSC RNGSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: RNG bus and kernel clocks disabled

1: RNG bus and kernel clocks enabled

Bit 17 **HASHEN**: HASH bus clock enable

Set and cleared by software.

Access can be secured by GTZC_TZSC HASHSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: HASH bus clock disabled

1: HASH bus clock enabled

Bit 16 **AESEN:** AES bus clock enable

Set and cleared by software.

Access can be secured by GTZC_TZSC AESSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: AES bus clock disabled

1: AES bus clock enabled

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

Bit 7 **GPIOHEN:** IO port H bus clock enable

Set and cleared by software.

Access can be secured by GPIOH SECx. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: IO port H bus clock disabled

1: IO port H bus clock enabled

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **GPIOCEN:** IO port C bus clock enable
Set and cleared by software.
Access can be secured by GPIOC SECx. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: IO port C bus clock disabled
1: IO port C bus clock enabled

Bit 1 **GPIOBEN:** IO port B bus clock enable
Set and cleared by software.
Access can be secured by GPIOB SECx. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: IO port B bus clock disabled
1: IO port B bus clock enabled

Bit 0 **GPIOAEN:** IO port A bus clock enable
Set and cleared by software.
Access can be secured by GPIOA SECx. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: IO port A bus clock disabled
1: IO port A bus clock enabled

12.8.22 RCC AHB4 peripheral clock enable register (RCC_AHB4ENR)

Address offset: 0x094

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC4 EN	Res.	Res.	PWR EN	Res.	Res.
										rw			rw		

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **ADC4EN**: ADC4 bus and kernel clocks enable
Set and cleared by software.
Access can be secured by GTZC_TZSC ADC4SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: ADC4 bus and kernel clocks disabled
1: ADC4 bus and kernel clocks enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **PWREN**: PWR bus clock enable
Set and cleared by software.
Can only be accessed secure when one or more features in the PWR is/are secure. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: PWR bus clock disabled
1: PWR bus clock enabled

Bits 1:0 Reserved, must be kept at reset value.

12.8.23 RCC AHB5 peripheral clock enable register (RCC_AHB5ENR)

Address offset: 0x098

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

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

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

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PTACONVEN**: PTACONV bus clock enable
Set and cleared by software.
Access can be secured by GTZC_TZSC PTACONVSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: PTACONV bus clock disabled
1: PTACONV bus clock enabled
Note that this bit is reserved on STM32WBA50/52xx devices.

Bit 0 **RADIOEN**: 2.4 GHz RADIO bus clock enable

Set and cleared by software.

Access can be secured by GTZC_TZSC RADIOSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: 2.4 GHz RADIO bus clock disabled (The 2.4 GHz RADIO bus clock may still be enabled by STRADIOCLKON)

1: 2.4 GHz RADIO bus clock enabled

Note: Before accessing the 2.4 GHz RADIO sleep timer registers, check the RADIOCLKRDY bit. When RADIOSMEN and STRADIOCLKON are both cleared, RADIOCLKRDY bit must be re-checked when exiting low-power modes (Sleep and Stop).

12.8.24 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.	Res.	Res.	Res.	I2C1 EN	Res.	Res.	Res.	USART2 EN	Res.
										rw				rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WWDG EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM3 EN	TIM2 EN
				rs										rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **I2C1EN**: I2C1 bus and kernel clocks enable

Set and cleared by software.

Access can be secured by GTZC_TZSC I2C1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: I2C1 bus and kernel clocks disabled

1: I2C1 bus and kernel clocks enabled

Note that this bit is reserved on STM32WBA50xx devices.

Bits 20:18 Reserved, must be kept at reset value.

Bit 17 **USART2EN**: USART2 bus and kernel clocks enable

Set and cleared by software.

Access can be secured by GTZC_TZSC USART2SEC When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV..

0: USART2 bus and kernel clocks disabled

1: USART2 bus and kernel clocks enabled

Note that this bit is reserved on STM32WBA50xx devices.

Bits 16:12 Reserved, must be kept at reset value.

Bit 11 **WWDGEN**: WWDG bus clock enable

Set by software to enable the window watchdog bus clock. Reset by hardware system reset.

This bit can also be set by hardware if the WWDG_SW option bit is reset.

Access can be secured by GTZC_TZSC WWDGSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: WWDG bus clock disabled

1: WWDG bus clock enabled

Bits 10:2 Reserved, must be kept at reset value.

Bit 1 **TIM3EN**: TIM3 bus and kernel clocks enable

Set and cleared by software.

Access can be secured by GTZC_TZSC TIM3SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: TIM3 bus and kernel clocks disabled

1: TIM3 bus and kernel clocks enabled

Note that this bit is reserved on STM32WBA50xx devices.

Bit 0 **TIM2EN**: TIM2 bus and kernel clocks enable

Set and cleared by software.

Access can be secured by GTZC_TZSC TIM2SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: TIM2 bus and kernel clocks disabled

1: TIM2 bus and kernel clocks enabled

12.8.25 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

This register is reserved on STM32WBA50xx devices.

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2 EN	Res.	Res.	Res.	Res.	Res.
										rw					

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2EN:** LPTIM2 bus and kernel clocks enable

Set and cleared by software.

Access can be secured by GTZC_TZSC LPTIM2SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: LPTIM2 bus and kernel clocks disabled

1: LPTIM2 bus and kernel clocks enabled

Bits 4:0 Reserved, must be kept at reset value.

12.8.26 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.	Res.	SAI1 EN	Res.	Res.	TIM17 EN	TIM16 EN	Res.
										rw			rw	rw	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	USART1 EN	Res.	SPI1 EN	TIM1 EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw	rw											

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **SAI1EN**: SAI1 bus and kernel clocks enable

Set and cleared by software.

Access can be secured by GTZC_TZSC SAI1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: SAI1 bus and kernel clocks disabled

1: SAI1 bus and kernel clocks enabled

Note that this bit is reserved on STM32WBA50/52xx devices.

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

Bit 18 **TIM17EN**: TIM17 bus and kernel clocks enable

Set and cleared by software.

Access can be secured by GTZC_TZSC TIM17SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: TIM17 bus and kernel clocks disabled

1: TIM17 bus and kernel clocks enabled

Note that this bit is reserved on STM32WBA50xx devices.

Bit 17 **TIM16EN**: TIM16 bus and kernel clocks enable

Set and cleared by software.

Access can be secured by GTZC_TZSC TIM16SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: TIM16 bus and kernel clocks disabled

1: TIM16 bus and kernel clocks enabled

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **USART1EN**: USART1bus and kernel clocks enable

Set and cleared by software.

Access can be secured by GTZC_TZSC USART1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: USART1 bus and kernel clocks disabled

1: USART1 bus and kernel clocks enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1EN**: SPI1 bus and kernel clocks enable

Set and cleared by software.

Access can be secured by GTZC_TZSC SPI1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: SPI1 bus and kernel clocks disabled

1: SPI1 bus and kernel clocks enabled

Note that this bit is reserved on STM32WBA50xx devices.

Bit 11 **TIM1EN**: TIM1 bus and kernel clocks enable

Set and cleared by software.

Access can be secured by GTZC_TZSC TIM1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: TIM1 bus and kernel clocks disabled

1: TIM1 bus and kernel clocks enabled

Bits 10:0 Reserved, must be kept at reset value.

12.8.27 RCC APB7 peripheral clock enable register (RCC_APB7ENR)

Address offset: 0x0A8

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.	RTCAPBEN	Res.	Res.	Res.	Res.	Res.
										rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMPEN	Res.	Res.	Res.	LPTIM1EN	Res.	Res.	Res.	I2C3EN	LPUART1EN	SPI3EN	Res.	Res.	Res.	SYSCFGEN	Res.
rw				rw				rw	rw	rw				rw	

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **RTCAPBEN**: RTC and TAMP bus clock enable

Set and cleared by software.

Can only be accessed secure when one or more features in the RTC or TAMP is/are secure. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: RTC bus clock disabled

1: RTC bus clock enabled

Bits 20:16 Reserved, must be kept at reset value.

Bit 15 **COMPEN**: COMP bus clock enable

Set and cleared by software.

Access can be secured by GTZC_TZSC COMPSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: COMP bus clock disabled

1: COMP bus clock enabled

Note that this bit is reserved on STM32WBA50/52xx devices.

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

Bit 11 **LPTIM1EN**: LPTIM1 bus and kernel clocks enable

Set and cleared by software.

Access can be secured by GTZC_TZSC LPTIM1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: LPTIM1 bus and kernel clocks disabled

1: LPTIM1 bus and kernel clocks enabled

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **I2C3EN**: I2C3 bus and kernel clocks enable
Set and cleared by software.
Access can be secured by GTZC_TZSC I2C3SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: I2C3 bus and kernel clocks disabled
1: I2C3 bus and kernel clocks enabled

Bit 6 **LPUART1EN**: LPUART1 bus and kernel clocks enable
Set and cleared by software.
Access can be secured by GTZC_TZSC LPUART1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: LPUART1 bus and kernel clocks disabled
1: LPUART1 bus and kernel clocks enabled

Bit 5 **SPI3EN**: SPI3 bus and kernel clocks enable
Set and cleared by software.
Access can be secured by GTZC_TZSC SPI3SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: SPI3 bus and kernel clocks disabled
1: SPI3 bus and kernel clocks enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **SYSCFGEN**: SYSCFG bus clock enable
Set and cleared by software.
Access can be secured by SYSCFG SYSCFGSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: SYSCFG bus clock disabled
1: SYSCFG bus clock enabled

Bit 0 Reserved, must be kept at reset value.

12.8.28 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 configures the clock gating only when the corresponding RCC_AHB1ENR.peripheralEN bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAM1 SMEN	Res.	ICACHE SMEN	Res.	Res.	Res.	Res.	GTZC1 SMEN	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFG SMEN	TSC SMEN
rw		rw					rw							rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	CRC SMEN	Res.	Res.	Res.	FLASH SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA1 SMEN
			rw				rw								rw

Bit 31 **SRAM1SMEN**: SRAM1 bus clock enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC_MPCBB1 SECx, INVSECSTATE. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: SRAM1 bus clock disabled by the clock gating during Sleep and Stop modes

1: SRAM1 bus clock enabled by the clock gating during Sleep and Stop modes

Bit 30 Reserved, must be kept at reset value.

Bit 29 **ICACHESMEN**: ICACHE bus clock enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC_TZSC ICACHE_REGSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV..

0: ICACHE bus clock disabled by the clock gating during Sleep and Stop modes

1: ICACHE bus clock enabled by the clock gating during Sleep and Stop modes

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **GTZC1SMEN**: GTZC1 bus clock enable during Sleep and Stop modes

Set and cleared by software.

Can only be accessed secure when one device is secure (TZEN = 1). When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: GTZC1 bus clock disabled by the clock gating during Sleep and Stop modes

1: GTZC1 bus clock enabled by the clock gating during Sleep and Stop modes

Note that this bit is reserved on STM32WBA50xx devices.

Bits 23:18 Reserved, must be kept at reset value.

Bit 17 **RAMCFGSMEN**: RAMCFG bus clock enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC_TZSC RAMCFGSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: RAMCFG bus clock disabled by the clock gating during Sleep and Stop modes

1: RAMCFG bus clock enabled by the clock gating during Sleep and Stop modes

Bit 16 **TSCSMEN**: TSC bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GTZC_TZSC TSCSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV..
0: TSC bus clock disabled by the clock gating during Sleep and Stop modes
1: TSC bus clock enabled by the clock gating during Sleep and Stop modes

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CRCSMEN**: CRC bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GTZC_TZSC CRCSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: CRC bus clock disabled by the clock gating during Sleep and Stop modes
1: CRC bus clock enabled by the clock gating during Sleep and Stop modes

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **FLASHSMEN**: FLASH bus clock enable during Sleep and Stop modes
Set and cleared by software.
Can only be accessed secured when the flash security state is secure. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: FLASH bus clock disabled by the clock gating during Sleep and Stop modes
1: FLASH bus clock enabled by the clock gating during Sleep and Stop modes

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

Bit 0 **GPDMA1SMEN**: GPDMA1 bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GPDMA1 SECx. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: GPDMA1 bus clock disabled by the clock gating during Sleep and Stop modes
1: GPDMA1 bus 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.

12.8.29 RCC AHB2 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB2SMENR)

Address offset: 0x0B4

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word and byte access

This register configures the clock gating only when the corresponding RCC_AHB2ENR.peripheralEN bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SRAM2 SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKA SMEN	Res.	SAES SMEN	RNG SMEN	HASH SMEN	AES SMEN
	rw									rw		rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOH SMEN	Res.	Res.	Res.	Res.	GPIOC SMEN	GPIOB SMEN	GPIOA SMEN
								rw					rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SRAM2SMEN**: SRAM2 bus clock enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC_MPCBB2 SECx, INVSECSTATE. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: SRAM2 bus clock disabled by the clock gating during Sleep and Stop modes

1: SRAM2 bus clock enabled by the clock gating during Sleep and Stop modes

Bits 29:22 Reserved, must be kept at reset value.

Bit 21 **PKASMEN**: PKA bus clock enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC_TZSC PKASEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: PKA bus clock disabled by the clock gating during Sleep and Stop modes

1: PKA bus clock enabled by the clock gating during Sleep and Stop modes

Bit 20 Reserved, must be kept at reset value.

Bit 19 **SAESSMEN:** SAES accelerator bus clock enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC_TZSC SAESSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: SAES bus clock disabled by the clock gating during Sleep and Stop modes

1: SAES bus clock enabled by the clock gating during Sleep and Stop modes

Note that this bit is reserved on STM32WBA50xx devices.

Bit 18 **RNGSMEN:** Random number generator (RNG) bus and kernel clocks enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC_TZSC RNGSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: RNG bus and kernel clocks disabled by the clock gating during Sleep and Stop modes

1: RNG bus and kernel clocks enabled by the clock gating during Sleep and Stop modes

Bit 17 **HASHSMEN**: HASH bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GTZC_TZSC HASHSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: HASH bus clock disabled by the clock gating during Sleep and Stop modes
1: HASH bus clock enabled by the clock gating during Sleep and Stop modes

Bit 16 **AESSMEN**: AES bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GTZC_TZSC AESSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: AES bus clock disabled by the clock gating during Sleep and Stop modes
1: AES bus clock enabled by the clock gating during Sleep and Stop modes

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

Bit 7 **GPIOHSMEN:** IO port H bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GPIOH SECx. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: IO port H bus clock disabled by the clock gating during Sleep and Stop modes
1: IO port H bus clock enabled by the clock gating during Sleep and Stop modes

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **GPIOCSMEN:** IO port C bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GPIOC SECx. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: IO port C bus clock disabled by the clock gating during Sleep and Stop modes
1: IO port C bus clock enabled by the clock gating during Sleep and Stop modes

Bit 1 **GPIOBSMEN:** IO port B bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GPIOB SECx. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: IO port B bus clock disabled by the clock gating during Sleep and Stop modes
1: IO port B bus clock enabled by the clock gating during Sleep and Stop modes

Bit 0 **GPIOASMEN:** IO port A bus clock enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GPIOA SECx. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: IO port A bus clock disabled by the clock gating during Sleep and Stop modes
1: IO port A bus clock enabled by the clock gating during Sleep and Stop modes

12.8.30 RCC AHB4 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB4SMENR)

Address offset: 0x0BC

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word and byte access

This register configures the clock gating only when the corresponding RCC_AHB4ENR.peripheralEN bit 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC4 SMEN	Res.	Res.	PWR SMEN	Res.	Res.
										rw			rw		

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **ADC4SMEN**: ADC4 bus and kernel clocks enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GTZC_TZSC ADC4SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: ADC4 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes
1: ADC4 bus and kernel 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 4:3 Reserved, must be kept at reset value.

Bit 2 **PWRSMEN**: PWR bus clock enable during Sleep and Stop modes
Set and cleared by software.
Can only be accessed secure when one or more features in the PWR is/are secure. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: PWR bus clock disabled by the clock gating during Sleep and Stop modes
1: PWR bus clock enabled by the clock gating during Sleep and Stop modes

Bits 1:0 Reserved, must be kept at reset value.

12.8.31 RCC AHB5 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB5SMENR)

Address offset: 0x0C0

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word and byte access

This register configures the clock gating only when the corresponding RCC_AHB5ENR.peripheralEN bit 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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**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.	PTACONV SMEN	RADIO SMEN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PTACONVSMEN**: PTACONV bus clock enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC_TZSC PTACONVSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: PTACONV bus clock disabled by the clock gating during Sleep and Stop modes

1: PTACONV bus clock enabled by the clock gating during Sleep and Stop modes

Note that this bit is reserved on STM32WBA50/52xx devices.

Bit 0 **RADIOSMEN**: 2.4 GHz RADIO bus clock enable during Sleep and Stop modes when the 2.4 GHz RADIO is active.

Set and cleared by software.

Access can be secured by GTZC_TZSC RADIOSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: 2.4 GHz RADIO bus clock disabled by the clock gating during Sleep and Stop modes (The 2.4 GHz RADIO bus clock may still be enabled by STRADIOCLKON)

1: 2.4 GHz RADIO bus clock enabled by the clock gating during Sleep and Stop modes when the 2.4 GHz RADIO is active

12.8.32 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 configures the clock gating only when the corresponding RCC_APB1ENR1.peripheralEN bit 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.	Res.	Res.	Res.	I2C1 SMEN	Res.	Res.	Res.	USART2 SMEN	Res.
										rw				rw	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	WWDG SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM3 SMEN	TIM2 SMEN
				rw										rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **I2C1SMEN**: I2C1 bus and kernel clocks enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC_TZSC I2C1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: I2C1 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes

1: I2C1 bus and kernel clocks enabled by the clock gating during Sleep and Stop modes

Note that this bit is reserved on STM32WBA50xx devices.

Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bits 20:18 Reserved, must be kept at reset value.

Bit 17 **USART2SMEN**: USART2 bus and kernel clocks enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC_TZSC USART2SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: USART2 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes

1: USART2 bus and kernel clocks enabled by the clock gating during Sleep and Stop modes

Note that this bit is reserved on STM32WBA50xx devices.

Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bits 16:12 Reserved, must be kept at reset value.

Bit 11 **WWDGSMEN**: Window watchdog bus clock 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.

Access can be secured by GTZC_TZSC WWDGSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: WWDG bus clock disabled by the clock gating during Sleep mode

1: WWDG bus clock enabled by the clock gating during Sleep mode

Bits 10:2 Reserved, must be kept at reset value.

Bit 1 **TIM3SMEN**: TIM3 bus and kernel clocks enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC_TZSC TIM3SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: TIM3 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes

1: TIM3 bus and kernel clocks enabled by the clock gating during Sleep and Stop modes

Note that this bit is reserved on STM32WBA50xx devices.

Bit 0 **TIM2SMEN**: TIM2 bus and kernel clocks enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GTZC_TZSC TIM2SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: TIM2 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes
1: TIM2 bus and kernel clocks enabled by the clock gating during Sleep and Stop modes

12.8.33 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 configures the clock gating only when the corresponding RCC_APB1ENR2.peripheralEN bit is set.

This register is reserved on STM32WBA50xx devices.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2 SMEN	Res.	Res.	Res.	Res.	Res.
										rw					

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2SMEN:** LPTIM2 bus and kernel clocks enable during Sleep and Stop modes
Set and cleared by software.
Access can be secured by GTZC_TZSC LPTIM2SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0: LPTIM2 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes
1: LPTIM2 bus and kernel 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 4:0 Reserved, must be kept at reset value.

12.8.34 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 configures the clock gating only when the corresponding RCC_APB2ENR.peripheralEN bit 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.	Res.	Res.	Res.	SAI1 SMEN	Res.	Res.	TIM17 SMEN	TIM16 SMEN	Res.
										rw			rw	rw	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	USART1 SMEN	Res.	SPI1 SMEN	TIM1 SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw	rw											

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **SAI1SMEN**: SAI1 bus and kernel clocks enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC_TZSC SAI1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: SAI1 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes

1: SAI1 bus and kernel clocks enabled by the clock gating during Sleep and Stop modes

Note that this bit is reserved on STM32WBA50/52xx devices.

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

Bit 18 **TIM17SMEN**: TIM17 bus and kernel clocks enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC_TZSC TIM17SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: TIM17 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes

1: TIM17 bus and kernel clocks enabled by the clock gating during Sleep and Stop modes

Note that this bit is reserved on STM32WBA50xx devices.

Bit 17 **TIM16SMEN**: TIM16 bus and kernel clocks enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC_TZSC TIM16SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: TIM16 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes

1: TIM16 bus and kernel clocks enabled by the clock gating during Sleep and Stop modes

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **USART1SMEN**: USART1 bus and kernel clocks enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC_TZSC USART1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: USART1 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes

1: USART1 bus and kernel 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 13 Reserved, must be kept at reset value.

Bit 12 **SPI1SMEN**: SPI1 bus and kernel clocks enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC_TZSC SPI1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: SPI1 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes

1: SPI1 bus and kernel clocks enabled by the clock gating during Sleep and Stop modes

Note that this bit is reserved on STM32WBA50xx devices.

Note: This bit must be set to allow the peripheral to wake up from Stop modes.

Bit 11 **TIM1SMEN**: TIM1 bus and kernel clocks enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC_TZSC TIM1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: TIM1 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes

1: TIM1 bus and kernel clocks enabled by the clock gating during Sleep and Stop modes

Bits 10:0 Reserved, must be kept at reset value.

12.8.35 RCC APB7 peripheral clock enable in Sleep and Stop modes register (RCC_APB7SMENR)

Address offset: 0x0D0

Reset value: 0xFFFF FFFF

Access: no wait state; word, half-word and byte access

This register configures the clock gating only when the corresponding RCC_APB7ENR.peripheralEN bit 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.	Res.	Res.	Res.	RTCAPBSMEN	Res.	Res.	Res.	Res.	Res.
										rw					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
COMPSMEN	Res.	Res.	Res.	LPTIM1SMEN	Res.	Res.	Res.	I2C3SMEN	LPUART1SMEN	SPI3SMEN	Res.	Res.	Res.	SYSCFGSMEN	Res.
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.

Can only be accessed secure when one or more features in the RTC or TAMP is/are secure. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

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.

Bits 20:16 Reserved, must be kept at reset value.

Bit 15 **COMPSMEN**: COMP bus clock enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC_TZSC COMPSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: COMP bus clock disabled by the clock gating during Sleep and Stop modes

1: COMP bus clock enabled by the clock gating during Sleep and Stop modes

Note that this bit is reserved on STM32WBA50/52xx devices.

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

Bit 11 **LPTIM1SMEN**: LPTIM1 bus and kernel clocks enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC_TZSC LPTIM1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: LPTIM1 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes

1: LPTIM1 bus and kernel 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 bus and kernel clocks enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC_TZSC I2C3SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: I2C3 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes

1: I2C3 bus and kernel 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 bus and kernel clocks enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC_TZSC LPUART1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: LPUART1 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes

1: LPUART1 bus and kernel 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 bus and kernel clocks enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by GTZC_TZSC SPI3SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: SPI3 bus and kernel clocks disabled by the clock gating during Sleep and Stop modes

1: SPI3 bus and kernel 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 bus clock enable during Sleep and Stop modes

Set and cleared by software.

Access can be secured by SYSCFG SYSCFGSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: SYSCFG bus clock disabled by the clock gating during Sleep and Stop modes

1: SYSCFG bus clock enabled by the clock gating during Sleep and Stop modes

Bit 0 Reserved, must be kept at reset value.

12.8.36 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
TIMIC SEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSTICKSEL [1:0]		SPI1SEL [1:0]		LPTIM2SEL [1:0]		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.	I2C1SEL [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	USART2SEL [1:0]		USART1SEL [1:0]	
				rw	rw							rw	rw	rw	rw

Bit 31 **TIMICSEL**: Clocks sources for TIM16,TIM17 and LPTIM2 internal input capture

When the TIMICSEL bit is set, the TIM16, TIM17 and LPTIM2 internal input capture can be connected to HSI16/256.

When TIMICSEL is cleared, the HSI16, clock sources cannot be selected as TIM16, TIM17 or LPTIM2 internal input capture.

Access can be secured by GTZC_TZSC TIM16SEC, TIM17SEC, or LPTIM2SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: HSI16 divider disabled

1: HSI16/256 generated and can be selected by TIM16, TIM17 and LPTIM2 as internal input capture

Note: The clock division must be disabled (TIMICSEL configured to 0) before selecting or changing a clock sources division.

Bits 30:24 Reserved, must be kept at reset value.

Bits 23:22 **SYSTICKSEL[1:0]**: SysTick clock source selection

These bits are used to select the SysTick clock source.

Access can be secured by RCC SYSCLKSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

00: hclk1 divided by 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 hclk1 cycle is introduced, due to the LSE or LSI sampling with hclk1 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.

Access can be secured by GTZC_TZSC SPI1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

00: pclk2 selected

01: SYSCLK selected

10: HSI16 selected

11: reserved

Note that these bits are reserved on STM32WBA50xx devices.

Note: The SPI1 is functional in Stop 0 and Stop 1 mode only when the kernel clock is HSI16.

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.

Access can be secured by GTZC_TZSC LPTIM2SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

00: pclk1 selected
01: LSI selected
10: HSI16 selected
11: LSE selected

Note that these bits are reserved on STM32WBA50xx devices.

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 17:12 Reserved, must be kept at reset value.

Bits 11:10 **I2C1SEL[1:0]**: I2C1 kernel clock source selection

These bits are used to select the I2C1 kernel clock source.

Access can be secured by GTZC_TZSC I2C1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

00: pclk1 selected
01: SYSCLK selected
10: HSI16 selected
11: reserved

Note that these bits are reserved on STM32WBA50xx devices.

Note: The I2C1 is functional in Stop 0 and Stop 1 mode only when the kernel clock is HSI16.

Bits 9:4 Reserved, must be kept at reset value.

Bits 3:2 **USART2SEL[1:0]**: USART2 kernel clock source selection

This bits are used to select the USART2 kernel clock source.

Access can be secured by GTZC_TZSC USART2SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

00: pclk1 selected
01: SYSCLK selected
10: HSI16 selected
11: LSE selected

Note that these bits are reserved on STM32WBA50xx devices.

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.

Access can be secured by GTZC_TZSC USART1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

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.

12.8.37 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.	ASSEL	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.	RNGSEL[1:0]		Res.	Res.	Res.	Res.	SAI1SEL[2:0]			Res.	Res.	Res.	Res.	Res.
		rw	rw					rw	rw	rw					

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ASSEL**: Audio synchronization kernel clock source selection

This bit allows to select the audio synchronization kernel clock source.

Access can be secured by GTZC_TZSC SAI1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: pll1pclk selected
1: pll1qclk selected

Note: This bit is reserved on STM32WBA50/52xx devices.

Bits 29:14 Reserved, must be kept at reset value.

Bits 13:12 **RNGSEL[1:0]**: RNGSEL kernel clock source selection

These bits allow to select the RNG kernel clock source.

Access can be secured by GTZC_TZSC RNGSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

00: LSE selected
01: LSI selected
10: HSI16 selected
11: pll1qclk divide by 2 selected

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:5 **SAI1SEL[2:0]**: SAI1 kernel clock source selection

These bits are used to select the SAI1 kernel clock source.

Access can be secured by GTZC_TZSC SAI1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

000: pll1pclk selected
001: pll1qclk selected
010: SYSCLK selected
011: input pin AUDIOCLK selected
100: HSI16 clock selected
others: reserved

Note that these bits are reserved on STM32WBA50/52xx devices.

Note: If the selected clock is the external clock and this clock is stopped, a switch to another clock is impossible.

Bits 4:0 Reserved, must be kept at reset value.

12.8.38 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.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ADCSEL [2:0]			LPTIM1SEL [1:0]		Res.	Res.	I2C3SEL [1:0]		Res.	SPI3SEL [1:0]		Res.	LPUART1SEL [1:0]	
	rw	rw	rw	rw	rw			rw	rw		rw	rw		rw	rw

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

Bits 14:12 **ADCSEL[2:0]**: ADC4 kernel clock source selection

These bits are used to select the ADC4 kernel clock source.

Access can be secured by GTZC_TZSC ADC4SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

000: hclk1 clock selected
001: SYSCLK selected
010: pll1pclk selected
011: HSE32 clock selected
100: HSI16 clock selected
others: reserved

Note: The ADC4 is functional in Stop modes only when the kernel clock is HSI16.

Bits 11:10 **LPTIM1SEL[1:0]**: LPTIM1 kernel clock source selection

These bits are used to select the LPTIM1 kernel clock source.

Access can be secured by GTZC_TZSC LPTIM1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

00: pclk7 selected.
01: LSI selected
10: HSI16 selected
11: LSE selected

Note: The LPTIM1 is functional in Stop modes only when the kernel clock is LSI, LSE, HSI16 with HSIKERON = 1.

Bits 9:8 Reserved, must be kept at reset value.

Bits 7:6 **I2C3SEL[1:0]**: I2C3 kernel clock source selection

These bits are used to select the I2C3 kernel clock source.

Access can be secured by GTZC_TZSC I2C3SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

00: pclk7 selected
01: SYSCLK selected
10: HSI16 selected
11: reserved

Note: The I2C3 is functional in Stop modes only when the kernel clock is HSI16

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.

Access can be secured by GTZC_TZSC SPI3SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

00: pclk7 selected
01: SYSCLK selected
10: HSI16 selected
11: reserved

Note: The SPI3 is functional in Stop modes only when the kernel clock is HSI16.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **LPUART1SEL[1:0]**: LPUART1 kernel clock source selection

These bits are used to select the LPUART1 kernel clock source.

Access can be secured by GTZC_TZSC LPUART1SEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

00: pclk7 selected
01: SYSCLK selected
10: HSI16 selected
11: LSE selected

Note: The LPUART1 is functional in Stop modes only when the kernel clock is HSI16 or LSE.

12.8.39 RCC Backup domain control register (RCC_BDCR1)

Address offset: 0x0F0

Backup domain reset value: 0b0000 0000 0000 0000 0XX0 0000 0000 1000

Where X (LSETRIM) is loaded with factory-programmed value at BOR0 reset and OBL_LAUNCH when SBF is cleared.

Fields LSCOSEL, LSCOEN, LSETRIM, RADIOSTSEL, LSI1ON, LSI1PREDIV, and BDRST are reset only by Backup domain power-on reset (BOR0), and not by a BDRST reset.

Reset value not affected by exit Standby mode, nor by system reset or BORx (x = 1 to 4).

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 LSETRIM bits are outside the Core domain: as a result, after reset, they are write-protected and the DBP bit in the PWR disable Backup domain register (PWR_DBPR) must be set before they can be modified (see Section 11: Power control (PWR) for further information).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	LSI2 RDY	LSI2 ON	LSI1 PREDIV	LSI1 RDY	LSI1 ON	LSCO SEL	LSCO EN	Res.	Res.	Res.	Res.	RADIOSTSEL [1:0]		Res.	BDRST
	r	rw	rw	r	rw	rw	rw					rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LSETRIM[1:0]		LSE GFON	LSE SYSRDY	Res.	RTCSEL[1:0]		LSE SYSEN	LSE CSSD	LSE CSSON	LSEDRV[1:0]		LSE BYP	LSE RDY	LSE ON
	rw	rw	rw	r		rw	rw	rw	r	rw	rw	rw	rw	r	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **LSI2RDY**: LSI2 oscillator ready

Set and cleared by hardware to indicate when the LSI2 oscillator is stable. After the LSI2ON bit is cleared, LSI2RDY goes low after three internal low-speed oscillator clock cycles.

Access can be secured by RCC LSISEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: LSI2 oscillator not ready

1: LSI2 oscillator ready

Note that this bit is reserved on STM32WBA50/52xx devices.

Bit 29 **LSI2ON**: LSI2 oscillator enable

Set and cleared by software.

Access can be secured by RCC LSISEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: LSI2 oscillator off

1: LSI2 oscillator on

Note that this bit is reserved on STM32WBA50/52xx devices.

Bit 28 **LSI1PREDIV**:LSI1 low-speed clock divider configuration

Set and cleared by software to enable the LSI1 division. This bit can be written only when the LSI1 is disabled (LSI1ON = 0 and LSI1RDY = 0). The LSI1PREDIV cannot be changed if the LSI1 is used by the IWDG or by the RTC.

Access can be secured by RCC LSISEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: LSI1 not divided

1: LSI1 divided by 128

Bit 27 **LSI1RDY**: LSI1 oscillator ready

Set and cleared by hardware to indicate when the LSI1 oscillator is stable. After the LSI1ON bit is cleared, LSI1RDY goes low after three internal low-speed oscillator clock cycles. This bit is set when the LSI1 is used by IWDG or RTC, even if LSI1ON = 0.

Access can be secured by RCC LSISEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: LSI1 oscillator not ready

1: LSI1 oscillator ready

Bit 26 **LSI1ON**: LSI1 oscillator enable

Set and cleared by software.

Access can be secured by RCC LSISEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: LSI1 oscillator off

1: LSI1 oscillator on

Bit 25 **LSCOSEL**: Low-speed clock output selection

Set and cleared by software.

Access can be secured by RCC LSISEC and/or RCC LSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: LSI clock selected

1: LSE clock selected

Bit 24 **LSCOEN**: Low-speed clock output (LSCO) enable

Set and cleared by software.

Access can be secured by RCC LSISEC and/or RCC LSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: LSCO disabled

1: LSCO enabled

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

Bits 19:18 **RADIOSTSEL[1:0]**: 2.4 GHz RADIO sleep timer kernel clock enable and selection

Set and cleared by software.

Access can be secured by GTZC_TZSC RADIOSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

00: no clock selected, 2.4 GHz RADIO sleep timer kernel clock disabled

01: LSE oscillator clock selected

10: LSI oscillator clock selected

11: HSE32 oscillator clock divided by 1000 selected

Bit 17 Reserved, must be kept at reset value.

Bit 16 **BDRST**: Backup domain software reset

Set and cleared by software.

A Backup domain reset is generated only when the domain protection is disabled.

Can only be accessed secure when one or more features in the RTC or TAMP are secure. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: Reset not activated

1: Reset the entire Backup domain when the protection is disabled

Bit 15 Reserved, must be kept at reset value.

Bits 14:13 **LSETRIM[1:0]**: LSE trimming

These bits are initialized at startup and after OBL_LAUNCH with SBF cleared with the factory-programmed LSE calibration value.

Set and cleared by software. These bits must be modified only once after a BOR reset or an OBL_LAUNCH and before enabling LSE with LSEON (when both LSEON = 0 and LSERDY = 0).

Access can be secured by RCC LSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

00: current source resistance 5/4 x R

01: current source resistance R

10: current source resistance 3/4 x R

11: current source resistance 2/3 x R

Note: OBL_LAUNCH of this field occurs only when SBF is cleared and must then only be started by software when LSE oscillator is disabled, LSEON = 0 and LSERDY = 0.

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).

Access can be secured by RCC LSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

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.

Access can be secured by RCC LSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

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 kernel clock source enable and selection

Set by software to enable and select the clock source for the RTC.

Can only be accessed secure when one or more features in the RTC or TAMP is/are secure. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

00: no clock selected, RTC and TAMP kernel clock disabled

01: LSE oscillator clock selected, and enabled

10: LSI oscillator clock selected, and enabled

11: HSE32 oscillator clock divided by 32 selected, and enabled

Bit 7 **LSESYSEN**: LSE system clock (LSESYS) enable

Set by software to enable the LSE system clock generated by RCC. The lsesys clock is used for peripherals (USART, LPUART, LPTIM, RNG, 2.4 GHz RADIO) and functions (LSCO, MCO, TIM triggers, LPTIM trigger) excluding the RTC, TAMP and LSECSS.

Access can be secured by RCC LSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: LSESYS clock disabled

1: LSESYS clock enabled

Bit 6 **LSECSSD**: Low speed external clock security, LSE failure Detection

Set by hardware to indicate when a failure is detected by the LSECCS on the external 32 kHz oscillator.

Reset when LSCSSON bit is cleared.

Access can be secured by RCC LSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: no failure detected on LSE

1: failure detected on LSE

Bit 5 **LSECSSON**: Low speed external clock security enable

Set by software to enable the LSECSS. 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.

Access can be secured by RCC LSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: LSECSS disabled off

1: LSECSS enabled on

Bits 4:3 **LSEDRV[1:0]**: LSE oscillator drive capability

Set by software to modulate the drive capability of the LSE oscillator. LSEDRV must be programmed to a different value than 0 before enabling the LSE oscillator in ‘Xtal’ mode.

Access can be secured by RCC LSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

00: reserved

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).

Access can be secured by RCC LSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: LSE oscillator ‘Xtal’ mode

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.

Access can be secured by RCC LSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: LSE oscillator not ready

1: LSE oscillator ready

Bit 0 **LSEON**: LSE oscillator enable

Set and cleared by software.

Access can be secured by RCC LSESEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.

0: LSE oscillator off

1: LSE oscillator on

12.8.40 RCC control/status register (RCC_CSR)

Address offset: 0x0F4

Reset value: 0x0C00 0000

Reset flags are only reset by BOR0 power reset.

Access: 0 ≤ wait state ≤ 3; word, half-word and byte access

Access can be protected by RCC SPRIV or RCC NSPRIV.

Wait states are inserted in case of successive accesses to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWR RSTF	WWDG RSTF	IWDG RSTF	SFT RSTF	BOR RSTF	PIN RSTF	OBL RSTF	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r		rw							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 **LPWRRSTF**: Low-power reset flag

Set by hardware when a reset occurs due to illegal Stop and Standby modes entry.
Cleared by writing to the RMVF bit.
0: no illegal mode reset occurred
1: illegal mode reset occurred

Bit 30 **WWDGRSTF**: Window watchdog reset flag

Set by hardware when a window watchdog reset occurs.
Cleared by writing to the RMVF bit.
0: no window watchdog reset occurred
1: window watchdog reset occurred

Bit 29 **IWDGRSTF**: Independent 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.
Access can be secured by RCC RMVFSEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt.
0: No effect
1: Clear the reset flags (once set by software bit is cleared automatically by hardware)

Bits 22:0 Reserved, must be kept at reset value.

12.8.41 RCC Backup domain control register (RCC_BDCR2)

Address offset: 0x0F8

Backup domain reset value: 0x0000 0000

Reset value is not effected by exit Standby mode, nor by system reset or BORx (x = 1 to 4).

Access: 0 ≤ wait state ≤ 3; word, half-word and byte access

Wait states are inserted in case of successive accesses to this register.

This register is reserved on STM32WBA50/52xx devices.

Note: The bits of this register are outside the Core domain. As a result, after reset, they are write-protected and the DBP bit in the PWR disable Backup domain register (PWR_DBPR) must be set before they can be modified (see Section 11: Power control (PWR) for further information).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSI2CFG[3:0]				Res.	LSI2MODE[2:0]		
								rw	rw	rw	rw		rw	rw	rw

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

Bits 7:4 **LSI2CFG[3:0]**: LSI2 oscillator configuration
Set and cleared by software to control the temperature at which the frequency temperature sensitivity is close to zero.
Access can be secured by RCC LSISEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0x0: LSI2 frequency temperature sensitivity is close to zero at +80 ^{o}C
0x1: LSI2 frequency temperature sensitivity is close to zero at +50 ^{o}C
0x2: LSI2 frequency temperature sensitivity is close to zero at +20 ^{o}C
others reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **LSI2MODE[2:0]**: LSI2 oscillator operating mode configuration
Set and cleared by software to select operating mode of power consumption versus accuracy.
Access can be secured by RCC LSISEC. When secure, a non-secure read/write access is RAZ/WI. It does not generate an illegal access interrupt. This bit can be protected against unprivileged access when secure with RCC SPRIV or when non-secure with RCC NSPRIV.
0b000: nominal-power, high accuracy
0b001: low-power, medium accuracy
0b010: ultra-low-power, low accuracy
others reserved

12.8.42 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 RCC SPRIV = 1 and by a secure privileged or unprivileged access if RCC SPRIV = 0. A non-secure write access generates an illegal access event and data are not written. This register can be read by secure or non-secure, privileged or unprivileged access.

When the system is not secure (TZEN = 0), this register is read as 0 writes are ignored.

This register is reserved on STM32WBA50xx devices.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RMVF SEC	Res.	Res.	Res.	Res.	PLL1 SEC	PRESC SEC	SYSCLK SEC	LSE SEC	LSI SEC	Res.	HSE SEC	HSI SEC
			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

Bits 11:8 Reserved, must be kept at reset value.

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 selection, 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 Reserved, must be kept at reset value.

Bit 1 **HSESEC**: HSE32 clock configuration bits, status bits and HSECSS security
Set and reset by software.
0: Non secure
1: Secure

Bit 0 **HSISEC**: HSI16 clock configuration and status bits security
Set and reset by software.
0: Non secure
1: Secure

12.8.43 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.

This register is reserved on STM32WBA50xx devices.

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

15	14	13	12	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.

12.8.44 RCC audio synchronization control register (RCC_ASCR)

Address offset: 0x1C0

Reset value: 0x0000 4000

Access to this register can be protected by GTZC_TZSC SAI1SEC and RCC SPRIV or RCC NSPRIV.

This register is reserved on STM32WBA50/52xx devices.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPS[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.	PSC[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEN
	rw	rw	rw	rw	rw	rw	rw								rw

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

Bits 22:16 **CPS[6:0]:** Capture prescaler
This field is set and cleared by software.
Capture period in number of counter periods. Capture period = counter period * (TPS + 1).

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **PSC[6:0]:** Clock prescaler
This field is set and cleared by software.
Counter clock frequency = $f_{audiosync\_ker\_ck}$ / (PSC + 1).

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

Bit 0 **CEN:** Counter enable
This bit is set and cleared by software.
Clearing this bit resets the audio synchronization counter and capture prescaler and all associated registers ASCR, ASIER, ASSR, ASCNTR, ASARR, ASCAR, and ASCOR.
0: Audio synchronization counter and kernel clock disabled
1: Audio synchronization counter and kernel clock enabled

12.8.45 RCC audio synchronization interrupt enable register (RCC_ASIER)

Address offset: 0x1C4

Reset value: 0x0000 0000

Access to this register can be protected by GTZC_TZSC SAI1SEC and RCC SPRIV or RCC NSPRIV.

This register is reserved on STM32WBA50/52xx devices.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**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.	CAEIE	COIE	CAIE
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CAEIE:** Capture error interrupt enable
This bit is set and cleared by software.
0: Capture error interrupt disabled and flag masked
1: Capture error interrupt and flag enabled

Bit 1 **COIE:** Comparer interrupt enable
This bit is set and cleared by software.
0: Compare interrupt disabled and flag masked
1: Compare interrupt and flag enabled

Bit 0 **CAIE:** Capture trigger interrupt enable
This bit is set and cleared by software.
0: Capture trigger interrupt disabled and flag masked
1: Capture trigger interrupt and flag enabled

12.8.46 RCC audio synchronization status register (RCC_ASSR)

Address offset: 0x1C8

Reset value: 0x0000 0000

Access to this register can be protected by GTZC_TZSC SAI1SEC and RCC SPRIV or RCC NSPRIV.

This register is reserved on STM32WBA50/52xx devices.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**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.	CAEF	COF	CAF
													rc_w0	rc_w0	rc_w0

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CAEF:** Capture error interrupt flag
This bit is set by hardware, only when CAEIE is enabled. This bit is cleared by software by writing it to 0 or masked when CAEIE is 0.
0: No capture error has been detected
1: A capture error has been detected

Bit 1 **COF:** Comparer interrupt flag

This field is set by hardware, only when COIE is enabled. This bit is cleared by software by writing it to 0 or masked when COIE is 0.

0: No counter compare occurred

1: A counter compare has occurred

Bit 0 **CAF:** Capture trigger interrupt flag

This field is set by hardware, only when CAIE is enabled. This bit is cleared by software by writing it to 0 or masked when CAIE is 0.

0: No capture update occurred

1: A capture update has occurred

12.8.47 RCC auto-reload register (RCC_ASCNTR)

Address offset: 0x1CC

Reset value: 0x0000 0000

Access to this register can be protected by GTZC_TZSC SAI1SEC and RCC SPRIV or RCC NSPRIV.

This register is reserved on STM32WBA50/52xx devices.

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

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

Bits 19:0 **CNT[19:0]:** Counter value

This field is set by hardware.

CNT[19:0] is the counter value at the time this register is read.

12.8.48 RCC auto-reload register (RCC_ASARR)

Address offset: 0x1D0

Reset value: 0x0008 0000

Access to this register can be protected by GTZC_TZSC SAI1SEC and RCC SPRIV or RCC NSPRIV.

This register is reserved on STM32WBA50/52xx devices.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AR[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 **AR[19:0]:** Auto-reload value
This field is set by software. This is the counter auto-reload value at which to restart the audio synchronization counter from value 0. It defines the counter period.

12.8.49 RCC capture register (RCC_ASCAR)

Address offset: 0x1D4

Reset value: 0x0000 0000

Access to this register can be protected by GTZC_TZSC SAI1SEC and RCC SPRIV or RCC NSPRIV.

This register is reserved on STM32WBA50/52xx devices.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	CA[26:16]										
					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
CA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:0 **CA[26:0]:** Capture value
This field is set by hardware.
CA[26:20] is the capture period counter value loaded on the trigger event.
CA[19:0] is the audio synchronization counter value loaded on the trigger event.

12.8.50 RCC compare register (RCC_ASCOR)

Address offset: 0x1D8

Reset value: 0x0000 0000

Access to this register can be protected by GTZC_TZSC SAI1SEC and RCC SPRIV or RCC NSPRIV.

This register is reserved on STM32WBA50/52xx devices.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CO[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 **CO[19:0]:** Compare value
This field is set by software. This is the value to compare with the audio synchronization counter to generate a compare interrupt event.

12.8.51 RCC clock configuration register 2 (RCC_CFGR4)

Address offset: 0x200

Reset value: 0x0000 0010

Access: word, half-word and byte access

1 or 2 wait states are inserted only if the access occurs during clock source switch.

Access to this register can be protected by RCC PRESCSEC and RCC SPRIV or RCC NSPRIV.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDIV5	Res.	HPRE5[2:0]		
											rw		rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **HDIV5**: AHB5 divider when SWS select HSI16 or HSE32

Set and reset by software.

Set to 1 by hardware when entering Stop 1 mode.

– When SYSCLK source indicated by SWS is HSI16 or HSE32: HDIV5 is taken into account

– When SYSCLK source indicated by SWS is PLL1: HDIV5 is not taken into account

Caution: Depending on the device voltage range, the software must set this bit correctly to ensure that the AHB5 frequency does not exceed the maximum allowed frequency (for more details refer to *Table 101*). After a write operation to this bit and before decreasing the voltage range, this register must be read to be sure that the new value is taken into account.

0: hclk5 = SYSCLK not divided

1: hclk5 = SYSCLK divided by 2

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **HPRE5[2:0]**: AHB5 prescaler when SWS select PLL1

Set and cleared by software to control the division factor of the AHB5 clock (hclk5).

Must not be changed when SYSCLK source indicated by SWS is PLL1.

– When SYSCLK source indicated by SWS is not PLL1: HPRE5 is not taken into account.

– When SYSCLK source indicated by SWS is PLL1: HPRE5 is taken into account, from the moment the system clock switch occurs

Caution: Depending on the device voltage range, the software must set these bits correctly to ensure that the AHB5 frequency does not exceed the maximum allowed frequency (for more details refer to *Table 101*). 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.

0xx: hclk5 = SYSCLK not divided

100: hclk5 = SYSCLK divided by 2

101: hclk5 = SYSCLK divided by 3

110: hclk5 = SYSCLK divided by 4

111: hclk5 = SYSCLK divided by 6

12.8.52 RCC RADIO peripheral clock enable register (RCC_RADIOENR)

Address offset: 0x208

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

Access to this register can be protected by GTZC_TZSC RADIOSEC and RCC SPRIV or NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RADIO CLKRDY	STRADIO CLKON
														r	rc_w0

15	14	13	12	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.	BBCLKEN	Res.
														rw	

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **RADIOCLKRDY**: 2.4 GHz RADIO bus clock ready.

Set and cleared by hardware to indicate that the 2.4 GHz RADIO bus clock is ready and the 2.4 GHz RADIO registers can be accessed.

0: 2.4 GHz RADIO bus clock not ready

1: 2.4 GHz RADIO bus clock ready

Note: Once both RADIOEN and STRADIOCLKON are cleared, RADIOCLKRDY goes low after three hclk5 clock cycles.

Bit 16 **STRADIOCLKON**: 2.4 GHz RADIO bus clock enable and HSE32 oscillator enable by 2.4 GHz RADIO sleep timer wake-up event

Set by hardware on a 2.4 GHz RADIO sleep timer wake-up event.

Cleared by software writing zero to this bit.

0: 2.4 GHz RADIO bus clock and HSE32 oscillator not requested by 2.4 GHz RADIO sleep timer wake-up event.

1: 2.4 GHz RADIO bus clock and HSE32 oscillator enabled by 2.4 GHz RADIO sleep timer wake-up event

Note: Before accessing the 2.4 GHz RADIO registers the RADIOCLKRDY bit must be checked.

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **BBCLKEN**: 2.4 GHz RADIO baseband kernel clock (aclk) enable

Set and cleared by software.

0: 2.4 GHz RADIO baseband kernel clock disabled

1: 2.4 GHz RADIO baseband kernel clock enabled

Note: The HSE32 oscillator needs to be enabled by either HSEON or STRADIOCLKON.

Bit 0 Reserved, must be kept at reset value.

12.8.53 RCC external clock sources calibration register 1(RCC_ECSCR1)

Address offset: 0x210

Power-on reset value: 0x0020 0000

Reset value not effected by exit Standby mode, nor reset from system reset and BORx (x = 1 to 4).

Access: no wait state; word, half-word and byte access

Access to this register can be protected by RCC HSESEC and RCC SPRIV or RCC NSPRIV.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSETRIM[5:0]					
										rw	rw	rw	rw	rw	rw

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

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **HSETRIM[5:0]**: HSE32 clock trimming

These bits provide user-programmable capacitor trimming value. It can be programmed to adjust the HSE32 oscillator frequency.

Bits 15:0 Reserved, must be kept at reset value.

12.8.54 RCC register map

Table 107. 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.	Res.	Res.	Res.	Res.	PLL1RDY	PLL1ON	Res.	Res.	Res.	HSEPRE	HSECSSON	Res.	HSERDY	HSEON	Res.	Res.	Res.	Res.	Res.	HSIRDY	HSIKERON	HSION	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value							0	0				0	0		0	0						0	0	0								
0x004 to 0x00C	Reserved	Reserved																															
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 to 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.	Res.	Res.	SWS [1:0]		SW [1:0]	
	Reset value		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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PPRE2 [2:0]			Res.	PPRE1 [2:0]			Res.	HPRE [2:0]		
	Reset value																						0	0	0		0	0	0		0	0	0
0x024	RCC_CFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PPRE7 [2:0]			Res.	Res.	Res.	Res.
	Reset value																										0	0	0				
0x028	RCC_PLL1CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL1RCLKPRERDY	PLL1RCLKPRESTEP	PLL1RCLKPRE	Res.	PLL1REN	PLL1QEN	PLL1PEN	Res.	Res.	Res.	Res.	Res.	PLL1M[2:0]			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
0x02C to 0x030	Reserved	Reserved																															
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 to 0x04C	Reserved	Reserved																															
0x050	RCC_CIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSI2RDYIE(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL1RDYIE	Res.	HSERDYIE	HSIRDYIE	Res.	LSERDYIE	LSI1RDYIE
	Reset value																0										0		0	0		0	0

Table 107. 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.	LSI2RDYF(1)	Res.	Res.	Res.	Res.	Res.	HSECSSF	Res.	Res.	Res.	PLL1RDYF	Res.	HSERDYF	HSIRDYF	Res.	LSERDYF	LSI1RDYF
	Reset value																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.	LSI2RDYC(1)	Res.	Res.	Res.	Res.	Res.	HSECSSC	Res.	Res.	Res.	PLL1RDYC	Res.	HSERDYC	HSIRDYC	Res.	LSERDYC	LSI1RDYC
	Reset value																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.	Res.	Res.	TSCRST	Res.	Res.	Res.	CRCRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA1RST
	Reset value																0				0												0
0x064	RCC_ AHB2RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKARST	HSEMRST	SAESRST(2)	RNGRST	HASHRST	AESRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOHRST	Res.	Res.	Res.	Res.	GPIOCRST	GPIOBRST	GPIOARST
	Reset value											0	0	0	0	0	0									0					0	0	0
0x068	Reserved	Reserved																															
0x06C	RCC_ AHB4RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC4RST	Res.	Res.	Res.	Res.	Res.
	Reset value																											0					
0x070	RCC_ AHB5RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTACONVRST(1)	RADIORST
	Reset value																															0	0
0x074	RCC_ APB1RSTR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C1RST(2)	Res.	Res.	Res.	USART2RST(2)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM3RST(2)	TIM2RST
	Reset value											0				0																0	0
0x078	RCC_ APB1RSTR2(3)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2RST	Res.	Res.	Res.	Res.	Res.
	Reset value																											0					
0x07C	RCC_ APB2RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI1RST(1)	Res.	Res.	TIM17RST(2)	TIM16RST	Res.	Res.	USART1RST	Res.	SPI1RST(2)	TIM1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0			0	0			0		0	0											

Table 107. 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_APB7RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPRST(1)	Res.	Res.	Res.	LPTIM1RST	Res.	Res.	Res.	I2C3RST	LPUART1RST	SPI3RST	Res.	Res.	Res.	SYSCFGRST	Res.
	Reset value																	0				0				0	0	0				0	
0x084	Reserved	Reserved																															
0x088	RCC_AHB1ENR	SRAM1EN	Res.	Res.	Res.	Res.	Res.	Res.	GTZC1EN(2)	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGEN	TSCEN	Res.	Res.	Res.	CRCEN	Res.	Res.	Res.	FLASHEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA1EN
	Reset value	1							0							0	0				0				1								0
0x08C	RCC_AHB2ENR	Res.	SRAM2EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKAEN	HSEMEN	SAESEN(2)	RNGEN	HASHEN	AESEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOHEN	Res.	Res.	Res.	Res.	GPIOCEN	GPIOBEN	GPIOAEN
	Reset value		1									0	0	0	0	0	0									0					0	0	0
0x090	Reserved	Reserved																															
0x094	RCC_AHB4ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC4EN	Res.	Res.	PWREN	Res.	Res.
	Reset value																											0			0		
0x098	RCC_AHB5ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTACONVEN(1)	RADIOEN
	Reset value																															0	0
0x09C	RCC_APB1ENR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C1EN(2)	Res.	Res.	Res.	USART2EN(2)	Res.	Res.	Res.	Res.	Res.	WWDGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM3EN(2)	TIM2EN
	Reset value											0				0						0										0	0
0x0A0	RCC_APB1ENR2(3)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2EN	Res.	Res.	Res.	Res.	Res.
	Reset value																											0					
0x0A4	RCC_APB2ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI1EN(1)	Res.	Res.	TIM17EN(2)	TIM16EN	Res.	Res.	USART1EN	Res.	SPI1EN(2)	TIM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0			0	0			0		0	0											
0x0A8	RCC_APB7ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBEN	Res.	Res.	Res.	Res.	Res.	COMPEN(1)	Res.	Res.	Res.	LPTIM1EN	Res.	Res.	Res.	I2C3EN	LPUART1EN	SPI3EN	Res.	Res.	Res.	SYSCFGEN	Res.
	Reset value											0						0				0				0	0	0				0	
0x0AC	Reserved	Reserved																															

Table 107. 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	Res.	ICACHESMEN	Res.	Res.	Res.	Res.	GTZC1SMEN(2)	Res.	Res.	Res.	Res.	Res.	Res.	RAMCFGSMEN	TSCSMEN	Res.	Res.	Res.	CRCSMEN	Res.	Res.	Res.	FLASHSMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA1SMEN
	Reset value	1		1					1							1	1				1				1								1
0x0B4	RCC_ AHB2SMENR	Res.	SRAM2SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKASMEN	Res.	SAESSMEN(2)	RNGSMEN	HASHSMEN	AESSMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOHSMEN	Res.	Res.	Res.	Res.	GPIOCSMEN	GPIOBSMEN	GPIOASMEN
	Reset value		1									1		1	1	1	1									1					1	1	1
0x0B8	Reserved	Reserved																															
0x0BC	RCC_ AHB4SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC4SMEN	Res.	Res.	PWRSMEN	Res.	Res.
	Reset value																											1			1		
0x0C0	RCC_ AHB5SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTACONVSMEN(1)	RADIOSMEN
	Reset value																															1	1
0x0C4	RCC_ APB1SMENR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C1SMEN(2)	Res.	Res.	Res.	USART2SMEN(2)	Res.	Res.	Res.	Res.	Res.	WWDGSMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIM3SMEN(2)	TIM2SMEN
	Reset value											1				1						1										1	1
0x0C8	RCC_ APB1SMENR2(3)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM2SMEN	Res.	Res.	Res.	Res.	Res.
	Reset value																											1					
0x0CC	RCC_ APB2SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI1SMEN(1)	Res.	Res.	TIM17SMEN(2)	TIM16SMEN	Res.	Res.	USART1SMEN	Res.	SPI1SMEN(2)	TIM1SMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											1			1	1			1		1	1											
0x0D0	RCC_ APB7SMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBSMEN	Res.	Res.	Res.	Res.	Res.	COMPSMEN(1)	Res.	Res.	Res.	LPTIM1SMEN	Res.	Res.	Res.	I2C3SMEN	LPUART1SMEN	SPI3SMEN	Res.	Res.	Res.	SYSCFGSMEN	Res.
	Reset value											1						1				1				1	1	1				1	
0x0D4 to 0x0DC	Reserved	Reserved																															

Table 107. 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
0x0E0	RCC_ CCIPR1	TIMICSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSTICKSEL [1:0]		SPI1SEL [1:0](2)		LPTIM2SEL [1:0](2)		Res.	Res.	Res.	Res.	Res.	Res.	I2C1SEL [1:0](2)		Res.	Res.	Res.	Res.	Res.	Res.	USART2SEL [1:0](2)		USART1SEL [1:0]	
	Reset value	0								0	0	0	0	0	0							0	0							0	0	0	0
0x0E4	RCC_ CCIPR2	Res.	ASSEL(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGSEL [1:0]		Res.	Res.	Res.	Res.	SAI1SEL [2:0](1)			Res.	Res.	Res.	Res.	Res.
	Reset value		0																	0	0					0	0	0					
0x0E8	RCC_ CCIPR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADCSEL [2:0]			LPTIM1SEL [1:0]		Res.	Res.	I2C3SEL [1:0]		SPI3SEL [2:0]			Res.	LPUART1 SEL[1:0]	
	Reset value																		0	0	0	0	0			0	0	0	0	0		0	0
0x0F0	RCC_BDCR1	Res.	LSI2RDY(1)	LSI2ON(1)	LSI1PREDIV	LSI1RDY	LSI1ON	LSCOSEL	LSCOEN	Res.	Res.	Res.	Res.	RADIOSTSEL [1:0]		Res.	BDRST	Res.	LSETRIM [1:0]		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	0	0	0	0	0
0x0F4	RCC_CSR	LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	BORRSTF	PINRSTF	OBLRSTF	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	1	1	0		0																							
0x0F8	RCC_BDCR2(4)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSI2CFG[3:0]				Res.	LSI2 MODE [2:0]		
	Reset value																									0	0	0	0		0	0	0
0x0FC to 0x10C	Reserved	Reserved																															
0x110	RCC_ SECCFGR(3)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RMVFSEC	Res.	Res.	Res.	Res.	PLL1SEC	PRESCSEC	SYSCLKSEC	LSESEC	LSISEC	Res.	HSESEC	HSISEC
	Reset value																				0					0	0	0	0	0		0	0
0x114	RCC_ PRIVCFGR(3)	Res.	Res.	Res.	Res.	Res.	Res.	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
0x118 to 0x1BC	Reserved	Reserved																															
0x1C0	RCC_ASCR(4)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TPS[6:0]							Res.	PSC[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEN
	Reset value										0	0	0	0	0	0	0		1	0	0	0	0	0	0								0
0x1C4	RCC_ASIER(4)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CAEIE	COIE	CAIE
	Reset value																														0	0	0
0x1C8	RCC_ASSR(4)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CAEF	COF	CAF
	Reset value																														0	0	0

Table 107. 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
0x1CC	RCC_ASCNTR[4]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1D0	RCC_ASARR[4]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR[19:0]																			
	Reset value													1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1D4	RCC_ASCAR[4]	Res.	Res.	Res.	Res.	Res.	CA[26:0]																										
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1D8	RCC_ASCOR[4]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CO[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1E0 to 0x1FC	Reserved	Reserved																															
0x200	RCC_CFGR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDIV5	Res.	HPRE5 [2:0]		
	Reset value																												1		0	0	0
0x204	Reserved	Reserved																															
0x208	RCC_ RADIOENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRADIOCLKON	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BBCLKEN	Res.
	Reset value																0															0	
0x20C	Reserved	Reserved																															
0x210	RCC_ECSCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSETRIM[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											1	0	0	0	0	0																
0x214 to 0x3FC	Reserved	Reserved																															

1. Bit(s) reserved on STM32WBA50/52xx devices.
2. Bit(s) reserved on STM32WBA50xx devices.
3. Register reserved on STM32WBA50xx devices.
4. Register reserved on STM32WBA50/52xx devices.

Refer to *Section 2.3* for the register boundary addresses.

13 Hardware semaphore (HSEM)

13.1 Introduction

The hardware semaphore block provides 16 (32-bit) register based semaphores.

The semaphores can be used to ensure synchronization between different processes running on the core. The HSEM provides a non-blocking mechanism to lock semaphores in an atomic way. The following functions are provided:

- Semaphore lock, in two ways:
  - 2-step lock: by writing LOCKID and PROCID, SEC and PRIV to the semaphore, followed by a read check
  - 1-step lock: by reading the LOCKID, SEC and PRIV from the semaphore
- Interrupt generation when a semaphore is unlocked
  - Each semaphore may generate an interrupt (secure and non-secure) on one of the interrupt lines
- Semaphore clear protection
  - A semaphore is only unlocked when LOCKID and PROCID, SEC, and PRIV match
- Global semaphore clear per LOCKID
- Secure semaphore lock support
- Privileged / unprivileged semaphore lock support
- Semaphore lock protection via semaphore attribute

13.2 Main features

The HSEM includes the following features:

- 16 (32-bit) semaphores with security, privilege attributes
- 8-bit PROCID
- 4-bit LOCKID
- 1-bit secure domain indication
- 1-bit privileged domain indication
- One non-secure and one secure interrupt lines
- Lock indication

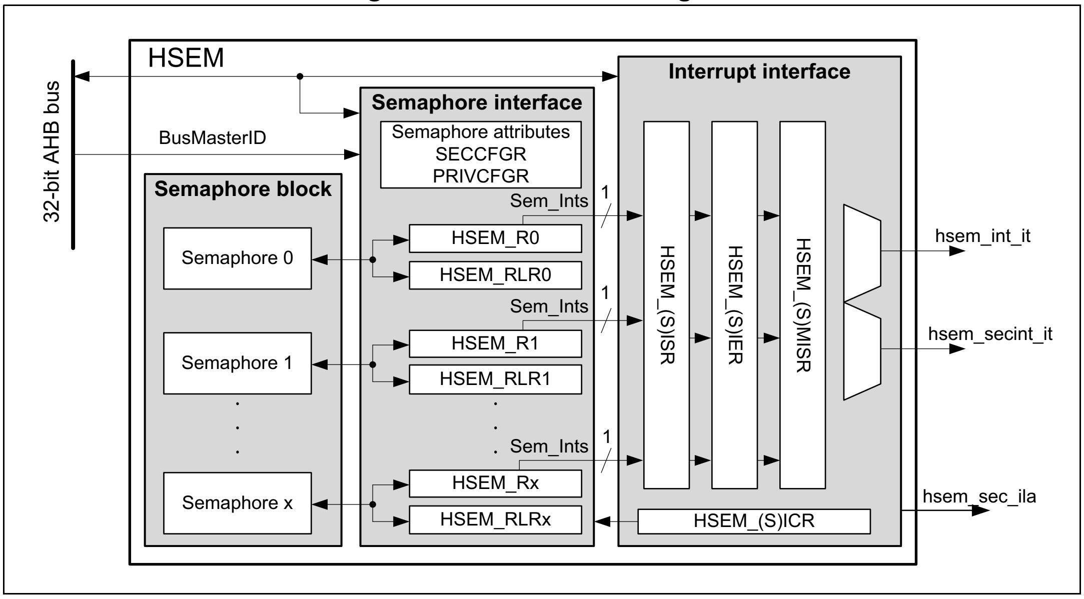
13.3 Functional description

13.3.1 HSEM block diagram

As shown in *Figure 40*, the HSEM is based on three sub-blocks:

- the semaphore block containing the semaphore status and IDs
- the semaphore interface block providing AHB access to the semaphore via the HSEM_Rx and HSEM_RLRx registers, and the semaphore security, privilege
- the interrupt interface block providing control for the interrupts via HSEM_ISR, HSEM_(S)IER, HSEM_(S)MISR, and HSEM_(S)ICR registers.

Figure 40. HSEM block diagram



13.3.2 HSEM internal signals

Table 108. HSEM internal input/output signals

Signal name	Signal type	Description
AHB bus	Digital input/output	AHB register access bus
BusMasterID	Digital input	AHB bus master ID
hsem_int_it	Digital output	Non-secure Interrupt line
hsem_secint_it	Digital output	Secure interrupt line
hsem_sec_ila	Digital output	illegal secure access event

13.3.3 HSEM lock procedures

There are two lock procedures, namely 2-step (write) lock and 1-step (read) lock. The two procedures can be used concurrently.

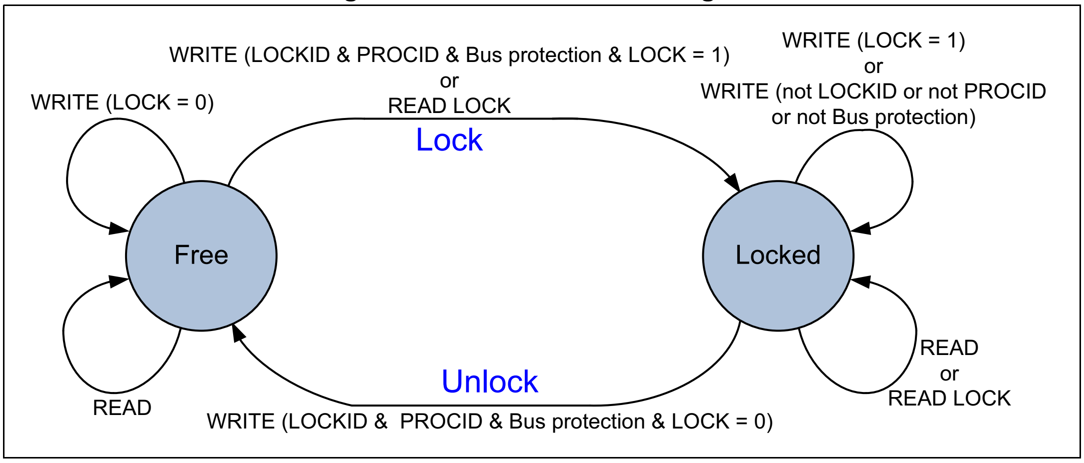
The semaphore is free when its LOCK bit is 0. In this case, the LOCKID, SEC, PRIV, and PROCID are also 0. When the LOCK bit is 1, the semaphore is locked and the LOCKID indicates which AHB bus master ID has locked it. The PROCID indicates which process of that AHB bus master ID has locked the semaphore. The SEC and PRIV indicated the lock protection, if the AHB bus master that locked the semaphore was running in secure and or privileged mode.

When write locking a semaphore, the written LOCKID, SEC and PRIV must match the AHB bus master ID , SEC and PRIV, and the PROCID is written by the AHB bus master software process taking the lock.

When read locking the semaphore, the LOCKID is taken from the AHB bus master ID, SEC and PRIV are taken from the bus protection, and the PROCID is forced to 0 by hardware. There is no PROCID available with read lock.

Semaphores can be locked only by a processor complying with the attribute settings.

Figure 41. Procedure state diagram



2-step (write) lock procedure

The 2-step lock procedure consists in a write to lock the semaphore, followed by a read to check if the lock has been successful, carried out from the HSEM_Rx register.

- Write semaphore with PROCID, SEC, PRIV and LOCKID, and LOCK = 1. The SEC, PRIV and LOCKID data written by software must match the AHB bus master information, that is, a secure, privileged, AHB bus master ID = 1 writes data SEC = 1, PRIV = 1 and LOCKID = 1.

  Lock is put in place when the semaphore is free at write time.
- Read-back the semaphore

  The software checks the lock status, if PROCID, SEC, PRIV and LOCKID match the written data, then the lock is confirmed.
- Else retry (the semaphore has been locked by another process or protection level).

A semaphore can only be locked when it is free.

A semaphore can be locked when the PROCID = 0.

Consecutive write attempts with LOCK = 1 to a locked semaphore are ignored.

1-step (read) lock procedure

The 1-step procedure consists in a read to lock and check the semaphore in a single step, carried out from the HSEM_RLRx register.

- Read lock semaphore with the AHB bus master LOCKID, SEC, PRIV.
- If read LOCKID, SEC, PRIV matches and PROCID = 0, then lock is put in place. If LOCKID, SEC, PRIV matches and PROCID is not 0, this means that another process from the same LOCKID and protection level, has locked the semaphore with a 2-step (write) procedure.
- Else retry (the semaphore has been locked by another process or protection level).

A semaphore can only be locked when it is free. When read locking a free semaphore, PROCID is 0. Read locking a locked semaphore returns the LOCKID, SEC, PRIV and PROCID that locked it. All read locks, including the first one that locks the semaphore, return the LOCKID, SEC, PRIV that locks or locked the semaphore.

Note: The 1-step procedure must not be used when running multiple processes of the same AHB bus master ID and protection level. All processes with the same bus protection level using the same semaphore read the same status. When only one process locks the semaphore, each process of that AHB bus master ID and protection level reads the semaphore as locked by itself with the LOCKID, SEC, PRIV.

13.3.4 HSEM write/read/read lock register address

For each semaphore, two AHB register addresses are provided, separated in two banks of 32-bit semaphore registers, spaced by a 0x80 address offset.

In the first register address bank the semaphore can be written (locked/unlocked) and read through the HSEM_Rx registers.

In the second register address bank the semaphore can be read (locked) through the HSEM_RLRx registers.

13.3.5 HSEM unlock procedures

Unlocking a semaphore is a protected process, to prevent accidental clearing by a AHB bus master ID, SEC, PRIV or by a process not having the semaphore lock right. The procedure consists in writing to the semaphore HSEM_Rx register with the corresponding LOCKID, SEC, PRIV and PROCID and LOCK = 0. A semaphore can always be unlocked by the LOCKID, SEC, PRIV, and PROCID that locked it. The semaphore is unlocked regardless of its attribute settings. When unlocked, the LOCKID, SEC, PRIV, and the PROCID are all 0.

Note: When the attributes of a locked semaphore are changed, it can always be unlocked by the AHB bus master ID, SEC, PRIV and process that locked it. To unlock the semaphore the PROCID must match.

When unlocked, an interrupt may be generated to signal the event. To this end, the semaphore non-secure interrupt and or secure interrupt must be enabled.

The unlock procedure consists in a write to the semaphore HSEM_Rx register with matching LOCKID, SEC, and PRIV, regardless on how the semaphore has been locked (1- or 2-step) and how it is protected by the semaphore attributes.

- Write semaphore with PROCID, SEC, PRIV, LOCKID, and LOCK = 0
- If the written data matches the semaphore PROCID, SEC, PRIV and LOCKID and the AHB bus master ID, SEC, PRIV, the semaphore is unlocked and an interrupt may be

generated when enabled, else write is ignored, semaphore remains locked and no interrupt is generated (the semaphore is locked by another process, and bus protection level or the written data does not match the AHB bus master signaling).

Note: Different processes of the AHB bus master ID and bus protection can write any PROCID value. Preventing other processes of the AHB bus master ID and bus protection from unlocking a semaphore must be ensured by software, handling the PROCID correctly.

13.3.6 HSEM LOCKID semaphore clear

All semaphores locked by a LOCKID and bus protection can be unlocked at once by using the HSEM_CR register. Write LOCKID, SEC, PRIV and correct KEY value in HSEM_CR. All locked semaphores with a matching LOCKID, SEC, PRIV are unlocked, and may generate an interrupt when enabled.

An interrupt may be generated for the unlocked semaphore(s). To this end, the semaphore interrupt must be enabled in the HSEM_(S)IER register.

13.3.7 HSEM interrupts

An interrupt line hsem_int_it (non-secure) and hsem_secint_it (secure) allows each semaphore to generate an interrupt.

An interrupt line provides the following features per semaphore:

- interrupt enable
- interrupt clear
- interrupt status
- masked interrupt status
- interrupt generation based on semaphore security attribute settings

With the interrupt enable (HSEM_(S)IER) the semaphores affecting the interrupt line can be enabled. Disabled (masked) semaphore interrupts do not set the masked interrupt status MISF for that semaphore, and do not generate an interrupt on the interrupt line.

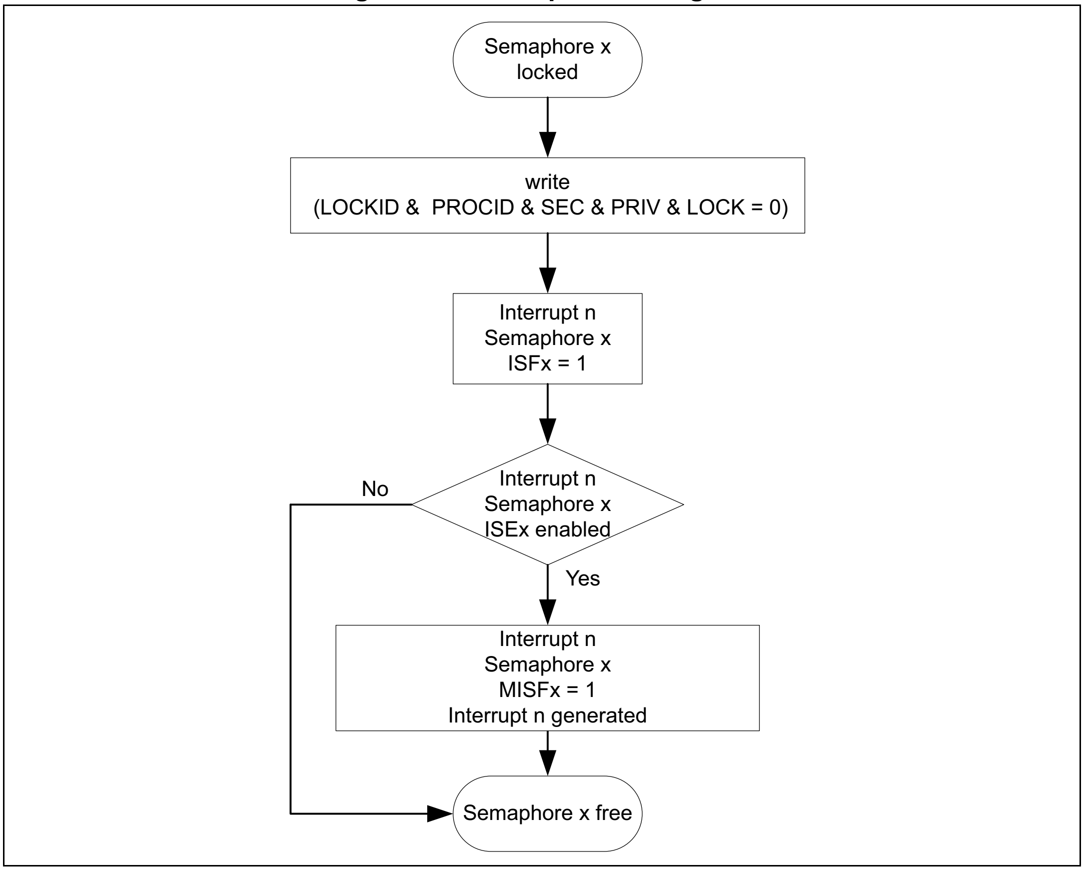
The interrupt clear (HSEM_(S)ICR) clears the interrupt status ISF and masked interrupt status MISF of the associated semaphore for the interrupt line.

The interrupt status (HSEM_(S)ISR) mirrors the semaphore interrupt status ISF before the enable.

The masked interrupt status (HSEM_(S)MISR) only mirrors the semaphore enabled interrupt status MISF on the interrupt line. All masked interrupt status MISF of the enabled semaphores need to be cleared to clear the interrupt line.

A semaphore with its security attribute set to secure generates only a secure interrupt, non-secure interrupts are omitted. On the other hand, a non-secure semaphore may generate non-secure and secure interrupts.

Figure 42. Interrupt state diagram



The procedure to get an interrupt when a semaphore becomes free is described hereafter.

Try to lock semaphore x

- If the semaphore lock is obtained, no interrupt is needed.
- If the semaphore lock fails:
- Clear pending semaphore x interrupt status for the (non-secure) interrupt line or secure interrupt line in HSEM_(S)ICR.
  Re-try to lock the semaphore x again:
  - If the semaphore lock is obtained, no interrupt is needed (semaphore has been freed between first try to lock and clear semaphore interrupt status).
  - If the semaphore lock fails, enable the semaphore x (non-secure) interrupt or secure interrupt line in HSEM_(S)IER.

On semaphore x free interrupt, try to lock semaphore x

- If the semaphore lock is obtained:
  Disable the semaphore x (non-secure) interrupt or secure interrupt in HSEM_(S)IER.
  Clear pending semaphore x interrupt status in HSEM_(S)ICR.

- If the semaphore x lock fails:
  Clear pending semaphore x interrupt status in HSEM_(S)ICR.
  Try again to lock the semaphore x:
  - If the semaphore lock is obtained (semaphore has been freed between first try to lock and semaphore interrupt status clear), disable the semaphore (non-secure) interrupt or secure interrupt in HSEM_(S)IER.
  - If the semaphore lock fails, wait for semaphore free interrupt.

Note: An interrupt does not lock the semaphore. After an interrupt, either the AHB bus master or the process must still perform the lock procedure to lock the semaphore.

It is possible to have multiple AHB bus masters informed by the semaphore free interrupts. Each AHB bus master gets its interrupt, and the first one to react locks the semaphore.

13.3.8 Semaphore attributes

The HSEM allows semaphores to be protected by the following attributes:

- security as defined in HSEM_SECCFGR
- privilege as defined in HSEM_PRIVCFGR

Semaphores with security attribute set to secure can be locked only by a secure access, and generate only a secure interrupt.

Semaphores with privilege attribute set to privilege can be locked only by a privileged access.

Table 109. Semaphore attributes

Attributes		Interrupt		Description
Security	Privilege	Secure	Non secure	
0	0	Yes	Yes	No protection. All locks allowed, any interrupt generated.
0	1			Privileged protection. Only privileged locks allowed, any interrupt generated.
1	0		No	Security protection. Only secure locks allowed and secure interrupt generated.
1	1			Security and privileged protection. Only secure privileged locks allowed and secure interrupt generated.

The HSEM allows only authorized AHB bus master ID to lock and unlock semaphores.

- The AHB bus master 2-step lock write access to the semaphore HSEM_Rx register is checked against the valid bus master ID.
  - Accesses from unauthorized AHB bus master IDs are discarded and do not lock the semaphore.
- The AHB bus master 1-step lock read access from the semaphore HSEM_RLRx register is checked against the valid bus master ID.
  - An unauthorized AHB bus master ID read from HSEM_RLRx returns all 0.
- The semaphore unlock write access to the HSEM_CR register is checked against the valid bus master ID. Only the valid bus master ID can write to the HSEM_CR register and unlock any of the LOCKID semaphores.
  - Accesses from unauthorized AHB bus master IDs are discarded and do not clear the LOCKID semaphores.

Table 110 details the relation between bus master/processor and LOCKID.

Table 110. Authorized AHB bus master ID

Bus master 0 (CPU)
LOCKID = 2

Note: Accesses from unauthorized AHB bus master IDs to other registers are granted.

13.4 HSEM registers

Registers must be accessed using word format. Byte and half-word accesses are ignored and have no effect on the semaphores, they generate a bus error.

13.4.1 HSEM register semaphore x (HSEM_Rx)

Address offset: 0x000 + 0x4 * x (x = 0 to 15)

Reset value: 0x0000 0000

The HSEM_Rx must be used to perform a 2-step write lock, read back, and for unlocking a semaphore. Only write accesses with authorized AHB bus master ID is granted. Write accesses with unauthorized AHB bus master IDs are discarded.

When the semaphore x security attribute SECx is enabled the semaphore can only be locked by a secure access. A non-secure write access generates a secure illegal access event. When the semaphore x security attribute SECx is disabled both secure and non-secure accesses may lock the semaphore.

When the semaphore x privileged attribute PRIVx is enabled the semaphore can only be locked by a privileged access. An unprivileged write access generates a privileged illegal access event. When the semaphore x privileged attribute PRIVx is disabled both privileged and unprivileged accesses may lock the semaphore.

A locked semaphore can always be unlocked with the semaphore PRIV, SEC and LOCKID that locked the semaphore, regardless of the semaphore attribute settings. To unlock the semaphore the PROCID must match.

When reading the semaphore x with a non-secure access, the semaphore data are returned only if:

- the semaphore security attribute SECx is disabled (non-secure semaphore)
- the semaphore has been locked with semaphore SEC = 0 (before security attribute SECx has been enabled) and the semaphore PRIV and LOCKID matches.

Else a 0 is returned, and a secure illegal access event is generated.

When reading the semaphore x with an unprivileged access, the semaphore data are returned only when:

- The semaphore privileged attribute PRIVx is disabled (unprivileged semaphore)
- The semaphore has been locked with semaphore PRIV = 0 (before security attribute PRIVx has been enabled) and the semaphore SEC and LOCKID matches.

Else a 0 is returned, and an privileged illegal access event is generated.

Semaphore data can only be read when the read access protection (SEC, PRIV and LOCKID) complies with the semaphore attribute protection rules or by a read access from the AHB bus master with the SEC, PRIV and LOCKID that locked the semaphore.

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

Bit 31 **LOCK:** Lock indication
This bit can be written and read by software.
0: On write free semaphore (only when LOCKID and PROCID match), on read semaphore is free.
1: On write try to lock semaphore, on read semaphore is locked.

Bits 30:14 Reserved, must be kept at reset value.

Bit 13 **PRIV:** Semaphore privilege
Written by software
- When the semaphore is free and the LOCK bit is at the same time written to 1, and the LOCKID matches the AHB bus master ID, SEC and PRIV matches the AHB bus master definition.
- When the semaphore is unlocked (LOCK written to 0 and AHB bus master ID matched LOCKID, SEC and PRIV matches the AHB bus master definition, the PRIV is cleared to 0.
- When the semaphore is unlocked (LOCK bit written to 0 and AHB bus master ID does not match LOCKID and/or SEC or PRIV do not match AHB bus master definition, the PRIV is not affected.
- Write when LOCK bit is already 1 (semaphore locked), the PRIV is not affected.
- An authorized read returns the stored PRIV value.
0: Semaphore free or locked by unprivileged compartment.
1: Semaphore locked by privilege compartment.

Bit 12 **SEC:** Semaphore secure
Written by software
- When the semaphore is free and the LOCK bit is at the same time written to 1, and the LOCKID matches the AHB bus master ID, SEC and PRIV matches the AHB bus master definition.
- When the semaphore is unlocked (LOCK written to 0 and AHB bus master ID matched LOCKID, SEC and PRIV matches the AHB bus master definition, the SEC is cleared to 0.
- When the semaphore is unlocked (LOCK bit written to 0 and AHB bus master ID does not match LOCKID and/or SEC or PRIV do not match AHB bus master definition, the SEC is not affected.
- Write when LOCK bit is already 1 (semaphore locked), the SEC is not affected.
- An authorized read returns the stored SEC value.
0: Semaphore free or locked by non-secure compartment.
1: Semaphore locked by secure compartment.

Bits 11:8 **LOCKID[3:0]:** Semaphore LOCKID
Written by software
- When the semaphore is free and the LOCK bit is at the same time written to 1 and the LOCKID matches the AHB bus master ID, SEC and PRIV matches the AHB bus master protection.
- When the semaphore is unlocked (LOCK written to 0 and AHB bus master ID matched LOCKID, SEC and PRIV matches the AHB bus master protection, the LOCKID is cleared to 0.
- When the semaphore is unlocked (LOCK bit written to 0 or AHB bus master ID does not match LOCKID and/or SEC or PRIV do not match AHB bus master protection, the LOCKID is not affected.
- Write when LOCK bit is already 1 (semaphore locked), the LOCKID is not affected.
- An authorized read returns the stored LOCKID value.

Bits 7:0 **PROCID[7:0]:** Semaphore PROCID

Written by software

-When the semaphore is free and the LOCK is written to 1, and the LOCKID matches the AHB bus master ID, SEC and PRIV matches the AHB bus master definition, PROCID is set to the written data.

- When the semaphore is unlocked, LOCK written to 0 and AHB bus master ID matched LOCKID, SEC and PRIV matches the AHB bus master protection, the PROCID is cleared to 0.

- When the semaphore is unlocked, LOCK bit written to 0 and AHB bus master ID does not match LOCKID and/or SEC or PRIV do not match the AHB bus master definition, the PROCID is not affected.

- Write when LOCK bit is already 1 (semaphore locked), the PROCID is not affected.

- An authorized read returns the stored PROCID value.

13.4.2 HSEM read lock register semaphore x (HSEM_RLRx)

Address offset: 0x080 + 0x4 * x (x = 0 to 15)

Reset value: 0x0000 0000

Accesses the same physical bits as HSEM_Rx. The HSEM_RLRx must be used to perform a 1-step read lock. Only read accesses with authorized AHB bus master ID is granted. Read accesses with unauthorized AHB bus master IDs are discarded and return 0.

When the semaphore x security attribute SECx is enabled the semaphore can only be locked by a secure access. A non-secure read access returns 0 and generates a secure illegal access event. When the semaphore x security attribute SECx is disabled both secure and non-secure accesses may lock the semaphore.

When the semaphore x privileged attribute PRIVx is enabled the semaphore can be locked only by a privileged access. An unprivileged read access reads 0 and generates a privileged illegal access event. When the semaphore x privileged attribute PRIVx is disabled both privileged and unprivileged accesses may lock the semaphore.

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

Bit 31 **LOCK:** Lock indication

This bit is read only by software at this address.

- When the semaphore is free:

A read with a valid AHB bus master ID and SEC and PRIV locks the semaphore and returns 1.

- When the semaphore is locked:

A read with a valid AHB bus master ID and SEC and PRIV returns 1 (the LOCKID and SEC and PRIV and PROCID reflect the already locked semaphore information).

Bits 30:14 Reserved, must be kept at reset value.

Bit 13 **PRIV:** Semaphore privilege

This field is read only by software at this address.

- When the semaphore is free:

A read with a valid AHB bus master ID and SEC and PRIV locks the semaphore and hardware sets the PRIV to the valid AHB bus master privileged definition.

- When the semaphore is locked:

A read with a valid AHB bus master ID and SEC and PRIV returns the PRIV of the AHB bus master that has locked the semaphore.

0: Semaphore free or locked by unprivileged compartment.

1: Semaphore locked by privileged compartment.

Bit 12 **SEC:** Semaphore secure.

This field is read only by software at this address.

- When the semaphore is free:

A read with a valid AHB bus master ID and SEC and PRIV locks the semaphore and hardware sets the SEC to the valid AHB bus master security definition.

- When the semaphore is locked:

A read with a valid AHB bus master ID and SEC and PRIV returns the SEC of the AHB bus master that has locked the semaphore.

0: Semaphore free or locked by non-secure compartment.

1: Semaphore locked by secure compartment.

Bits 11:8 **LOCKID[3:0]:** Semaphore LOCKID

This field is read only by software at this address.

On a read, when the semaphore is free, the hardware sets the LOCKID to the AHB bus master ID reading the semaphore. The LOCKID of the AHB bus master locking the semaphore is read.

On a read when the semaphore is locked, this field returns the LOCKID of the AHB bus master that has locked the semaphore.

Bits 7:0 **PROCID[7:0]:** Semaphore processor ID

This field is read only by software at this address.

- On a read when the semaphore is free:

A read with a valid AHB bus master ID and SEC and PRIV locks the semaphore and hardware sets the PROCID to 0.

- When the semaphore is locked:

A read with a valid AHB bus master ID and SEC and PRIV returns the PROCID of the AHB bus master that has locked the semaphore.

13.4.3 HSEM non-secure interrupt enable register (HSEM_IER)

Address offset: 0x100

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ISE[15:0]:** Non-secure Interrupt semaphore x enable bit (x = 0 to 15)
This bit is read and written by software.
When semaphore x SECx is disabled, bit x can be accessed with secure and non-secure access.
When semaphore x SECx is enabled, bit x is forced to 0 and cannot be accessed, write to this bit is discarded and a read returns 0.
When semaphore x PRIVx is disabled, bit x can be accessed with privilege and unprivileged access.
When semaphore x PRIVx is enabled, bit x can be accessed only with privileged access. Unprivileged write to this bit is discarded, unprivileged read returns 0.
0: Non-secure Interrupt generation for semaphore x disabled (masked)
1: Non-secure Interrupt generation for semaphore x enabled (not masked)

13.4.4 HSEM non-secure interrupt clear register (HSEM_ICR)

Address offset: 0x104

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ISC[15:0]:** Non-secure Interrupt semaphore x clear bit (x = 0 to 15)
This bit is written by software, and is always read 0.
When semaphore x SECx is disabled, bit x can be accessed with secure and non-secure access.
When semaphore x SECx is enabled, bit x cannot be accessed, write to this bit is discarded.
When semaphore x PRIVx is disabled, bit x can be accessed with privileged and unprivileged access.
When semaphore x PRIVx is enabled, bit x can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0: non-secure Interrupt semaphore x status ISFx and masked status MISFx not affected.
1: non-secure Interrupt semaphore x status ISFx and masked status MISFx cleared.

13.4.5 HSEM non-secure interrupt status register (HSEM_ISR)

Address offset: 0x108

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ISF[15:0]:** Interrupt semaphore x status bit before enable (mask) (x = 0 to 15)

This bit is set by hardware, and reset only by software. This bit is cleared by software writing the corresponding HSEM_ICR bit.

0: Interrupt semaphore x status, no interrupt pending

1: Interrupt semaphore x status, interrupt pending

13.4.6 HSEM non-secure interrupt status register (HSEM_MISR)

Address offset: 0x10C

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MISF[15:0]:** Masked non-secure interrupt semaphore x status bit after enable (mask) (x = 0 to 15)

This bit is set by hardware and read only by software. This bit is cleared by software writing the corresponding HSEM_ICR bit. This bit is read as 0 when semaphore x status is masked in HSEM_IER bit x.

When semaphore x SECx is disabled, bit x can be accessed with secure and non-secure access.

When semaphore x SECx is enabled, bit x cannot be accessed, read returns 0.

When semaphore x PRIVx is disabled, bit x can be accessed with privileged and unprivileged access.

When semaphore x PRIVx is enabled, bit x can be accessed only with privileged access. Unprivileged read returns 0.

0: non-secure interrupt semaphore x status after masking not pending

1: non-secure interrupt semaphore x status after masking pending

13.4.7 HSEM secure interrupt enable register (HSEM_SIER)

Address offset: 0x180

Reset value: 0x0000 0000

This register provides access security, a non-secure write access is discarded and a non-secure read returns all 0. Both non-secure read and write generate a secure illegal access event (when security is disabled by user option TZEN this register cannot be accessed).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SISE[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 **SISE[15:0]:** Secure interrupt semaphore x enable bit (x = 0 to 15)
This bit is read and written by software.
When semaphore x PRIVx is disabled, bit x can be accessed with secure privilege and secure unprivileged access.
When semaphore x PRIVx is enabled, bit x can be accessed only with secure privilege access. secure unprivileged write to this bit is discarded, secure unprivileged read return 0 value.
0: Secure interrupt generation for semaphore x is disabled (masked).
1: Secure interrupt generation for semaphore x is enabled (not masked).

13.4.8 HSEM secure interrupt clear register (HSEM_SICR)

Address offset: 0x184

Reset value: 0x0000 0000

This register provides access security, a non-secure write access is discarded and a non-secure read returns all 0. Both non-secure read and write generate a secure illegal access event (when security is disabled by user option TZEN this register cannot be accessed).

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SISC[15:0]:** Secure interrupt semaphore x clear bit (x = 0 to 15)
This bit is written by software, and is always read 0.
When semaphore x PRIVx is disabled, bit x can be accessed with secure privilege and secure unprivileged access.
When semaphore x PRIVx is enabled, bit x can be accessed only with secure privilege access. Secure unprivileged write to this bit is discarded.
0: Secure interrupt semaphore x status ISFx and masked status MISFx not affected.
1: Secure interrupt semaphore x status ISFx and masked status MISFx cleared.

13.4.9 HSEM secure interrupt status register (HSEM_SISR)

Address offset: 0x188

Reset value: 0x0000 0000

This register provides access security, a non-secure read returns all 0. Both non-secure read and write generate a secure illegal access event (when security is disabled by user option TZEN, then this register cannot be accessed any longer).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SISF[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 **SISF[15:0]:** Secure interrupt semaphore x status bit before enable (mask) (x = 0 to 15)
This bit is set by hardware and read only by software.
Bit is cleared by software writing the corresponding HSEM_SCnICR bit x.
When semaphore x PRIVx is disabled, bit x can be accessed with secure privilege and secure unprivileged access.
When semaphore x PRIVx is enabled, bit x can be accessed only with secure privilege access. Secure unprivileged read return 0 value.
0: Secure interrupt semaphore x status, no interrupt pending.
1: Secure interrupt semaphore x status, interrupt pending.

13.4.10 HSEM secure masked interrupt status register (HSEM_MSISR)

Address offset: 0x18C

Reset value: 0x0000 0000

This register provides access security, a non-secure read returns all 0. Both non-secure read and write generate a secure illegal access event (when security is disabled by user option TZEN, then this register cannot be accessed).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMISF[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 **SMISF[15:0]:** Secure masked interrupt semaphore x status bit after enable (mask) (x = 0 to 15)

This bit is set by hardware and read only by software.

Bit is cleared by software writing the corresponding HSEM_SCnICR bit x.

Bit is read as 0 when semaphore x status is masked in HSEM_SCnIER bit x.

When semaphore x PRIVx is disabled, bit x can be accessed with secure privilege and secure unprivileged access.

When semaphore x PRIVx is enabled, bit x can be accessed only with secure privilege access. Secure unprivileged read return 0 value.

0: Secure interrupt semaphore x status after masking not pending.

1: Secure interrupt semaphore x status after masking pending.

13.4.11 HSEM security configuration register (HSEM_SECCFGR)

Address offset: 0x200

Reset value: 0x0000 0000

This register provides write access security and privileged, a non-secure or unprivileged write access is discarded and generates an illegal access event. There is no read protection and the register data can be read by any accesses (when security is disabled by user option TZEN this register can no longer 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
SEC[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 **SEC[15:0]:** Semaphore x security attribute (x = 0 to 15)
This bit is set and cleared by software.
0: Semaphore x non-security, can be accessed by both secure and non-secure processors. When unlocking semaphore x both a secure and non-secure interrupt can be generated.
1: Semaphore x security, can be accessed only by secure processors. When unlocking semaphore x only a secure interrupt can be generated.

13.4.12 HSEM privilege configuration register (HSEM_PRIVCFGR)

Address offset: 0x210

Reset value: 0x0000 0000

This register provides write access privilege protection, an unprivileged write access is discarded. There is no read protection and the register data can be read by both privilege and unprivileged accesses.

There is no read protection and the register data can be read by both secure and non-secure accesses.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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[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 **PRIV[15:0]:** Semaphore x privilege attribute (x = 0 to 15)
This bit is set and cleared by software.
When semaphore x SECx is disabled, bit x can be write accessed with secure privileged and non-secure privileged access.
When semaphore x SECx is enabled, bit x can only be write accessed with secure privilege access. Non-secure privileged write access is discarded. Both secure and non-secure read return the register bit x value
0: Semaphore x unprivileged, can be accessed by both privileged and unprivileged processors.
1: Semaphore x privileged, can be accessed only by privileged processors.

13.4.13 HSEM clear register (HSEM_CR)

Address offset: 0x230

Reset value: 0x0000 0000

Only write accesses with authorized AHB bus master ID are granted. Write accesses with unauthorized AHB bus master ID are discarded.

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

Bits 31:16 **KEY[15:0]:** Semaphore clear key
This field can be written by software and is always read 0.
If this key value does not match HSEM_KEYR.KEY, semaphores are not affected.
If this key value matches HSEM_KEYR.KEY, all semaphores matching the LOCKID are cleared to the free state.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **PRIV:** PRIV value of semaphores to be cleared.
This field can be written by software, is always read 0.
Indicates the PRIV for which the semaphores are cleared when writing the HSEM_CR.

Bit 12 **SEC:** SEC value of semaphores to be cleared.
This field can be written by software, is always read 0.
Indicates the SEC for which the semaphores are cleared when writing the HSEM_CR

Bits 11:8 **LOCKID[3:0]:** LOCKID of semaphores to be cleared
This field can be written by software and is always read 0.
This field indicates the LOCKID for which the semaphores are cleared when writing the HSEM_CR.

Bits 7:0 Reserved, must be kept at reset value.

13.4.14 HSEM clear semaphore key register (HSEM_KEYR)

Address offset: 0x234

Reset value: 0x0000 0000

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

Bits 31:16	**KEY[15:0]:** Semaphore clear key This field can be written and read by software. Key value to match when clearing semaphores.
Bits 15:0	Reserved, must be kept at reset value.

13.4.15 HSEM register map

Table 111. HSEM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	HSEM_R0	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	LOCKID [3:0]				PROCID[7:0]							
	Reset value	0																		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	HSEM_R1	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	LOCKID [3:0]				PROCID[7:0]							
	Reset value	0																		0	0	0	0	0	0	0	0	0	0	0	0	0	0
⋮																																	
0x03C	HSEM_R15	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	LOCKID [3:0]				PROCID[7:0]							
	Reset value	0																		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x080	HSEM_RLR0	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	LOCKID [3:0]				PROCID[7:0]							
	Reset value	0																		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	HSEM_RLR1	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	LOCKID [3:0]				PROCID[7:0]							
	Reset value	0																		0	0	0	0	0	0	0	0	0	0	0	0	0	0
⋮																																	
0x0BC	HSEM_RLR15	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	LOCKID [3:0]				PROCID[7:0]							
	Reset value	0																		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100	HSEM_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ISE[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x104	HSEM_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ISC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x108	HSEM_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ISF[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10C	HSEM_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISF[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x180	HSEM_SIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SISE[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x184	HSEM_SICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SISC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x188	HSEM_SISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SISF[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 111. HSEM 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
0x18C	HSEM_SMISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMISF[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x200	HSEM_SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x210	HSEM_PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x230	HSEM_CR	KEY[15:0]																Res.	Res.	PRIV	SEC	LOCKID[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0								
0x234	HSEM_KEYR	KEY[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

14 General-purpose I/Os (GPIO)

14.1 GPIO introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR), a 16-bit 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), and a secure configuration register (GPIOx_SECCFGR).

14.2 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

14.3 GPIO implementation

Table 112. GPIO implementation

Device	GPIOA	GPIOB	GPIOC	GPIOH
STM32WBA55xx GPIO port pins	[15:0]	[15:11, 9:0]	[15:13]	[3]
STM32WBA54xx GPIO port pins	[15:5, 3:0]	[15:0]	[15:13]	[3]
STM32WBA52xx GPIO port pins	[15:5, 3:0]	[15:0]	[15:13]	[3]
STM32WBA50xx GPIO port pins	[15:12, 8:5, 2:0]	[15, 12, 9:8, 4:3]	[15:14]	[3]

14.4 GPIO functional description

Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the GPIO ports can be individually configured by software in several modes:

- Input floating
- Input pull-up
- Input pull-down
- Analog
- Analog pull-down
- 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.

GPIO configuration is available only in Run and Stop modes, it keeps the I/O port in its defined state. In Standby modes the GPIO configuration is lost. To keep I/O port input definition and output levels, a GPIO standby retention can be enabled in the PWR (see *Section 11.7.8: PWR Standby mode*).

Figure 43 shows the basic structures of a 3 V- or 5 V-tolerant GPIO (TT or FT). *Table 113* gives the possible port bit configurations.

Figure 43. Structure of 3 V- or 5 V-tolerant GPIO (TT or FT)

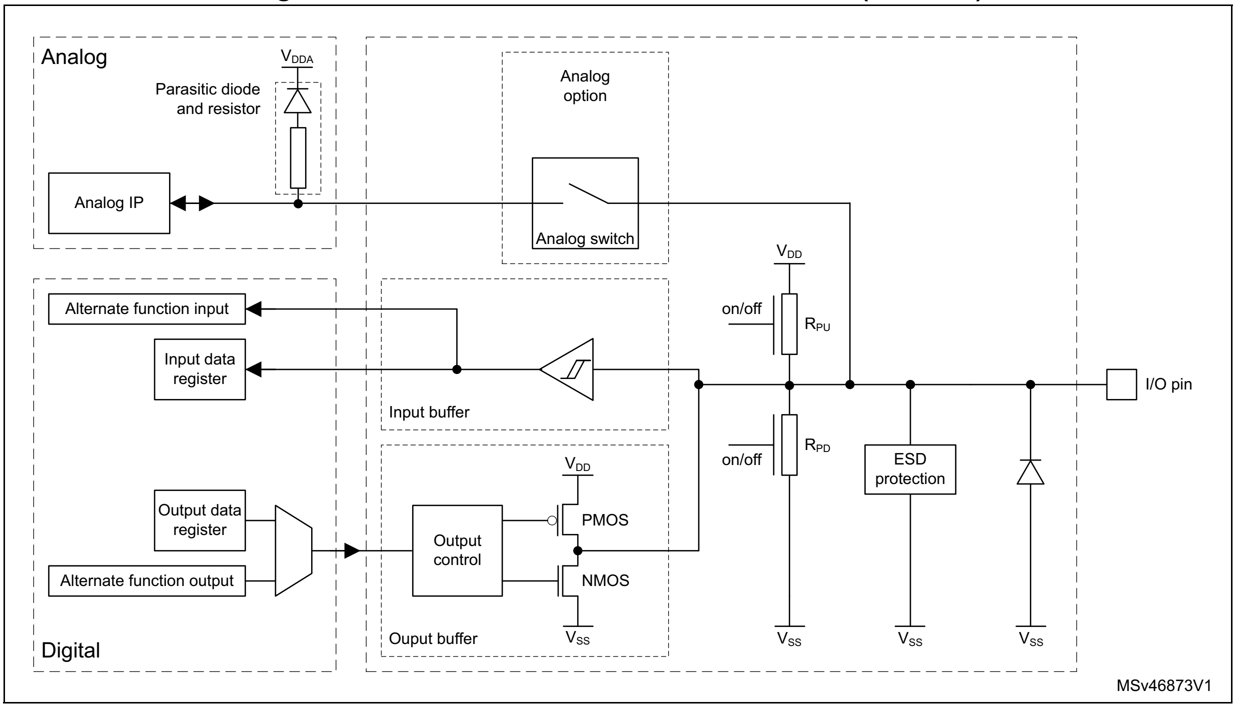


Table 113. Port bit configuration(1)

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)	
10	0	SPEED [1:0]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0	Input/output	Analog PD
	x	x	x	1	1	Reserved	

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

14.4.1 GPIO general-purpose I/O

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 14.4.16*).

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.

14.4.2 GPIO 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.

For secure peripherals the GPIO pin must be secure to use the secure peripheral alternate function.

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 alternate function in one of the GPIOx_AFRL or GPIOx_AFRH register.
  - Select the type, pull-up/down, and output speed via the GPIOx_OTYPER, GPIOx_PUPDR and GPIOx_OSPEEDER registers respectively.
  - Configure the desired I/O as an alternate function in the GPIOx_MODER register.
- **Additional functions:**
  - For the ADC, COMP and PVD_IN, configure the desired I/O in analog mode in the GPIOx_MODER register and configure the required function in the ADC, COMP, and PVD registers.

Refer to the “Alternate function mapping” table in the device datasheet for the detailed mapping of the alternate function I/O pins.

14.4.3 GPIO port additional function multiplexer

For the additional functions like RTC, TAMPx, WKUPx and LSE oscillator, configure the required I/O function in the related RTC, TAMP, PWR, and RCC registers. These functions have priority over the configuration in the standard GPIO registers.

14.4.4 GPIO 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.

14.4.5 GPIO port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (*GPIO port A input data register (GPIOA_IDR)* and *GPIO port A output data register (GPIOA_ODR)*).

GPIOx_ODR stores the data to be output, it is read/write accessible. The data input through the I/Os are stored into the input data register (GPIOx_IDR), a read-only register.

When changing MODER to select input or ODR level, up to three HCLK cycles are needed to reflect the GPIO level in the IDR register.

14.4.6 GPIO 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). This register has twice the size of GPIOx_ODR.

Two control bits in GPIOx_BSRR, namely BS(i) and BR(i) correspond to each bit in GPIOx_ODR. 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 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.

Individual bits in GPIOx_ODR can also be reset in a single atomic AHB write to GPIOx_BRR.

14.4.7 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.

To write the GPIOx_LCKR register, a specific write/read sequence must be applied. When the right LOCK sequence is applied to bit 16, 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 because GPIOx_LCKR bit 16 must be set at the same time as the [15:0] bits.

14.4.8 GPIO 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 's 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.

14.4.9 GPIO 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 19: Extended interrupts and event controller (EXTI)*.

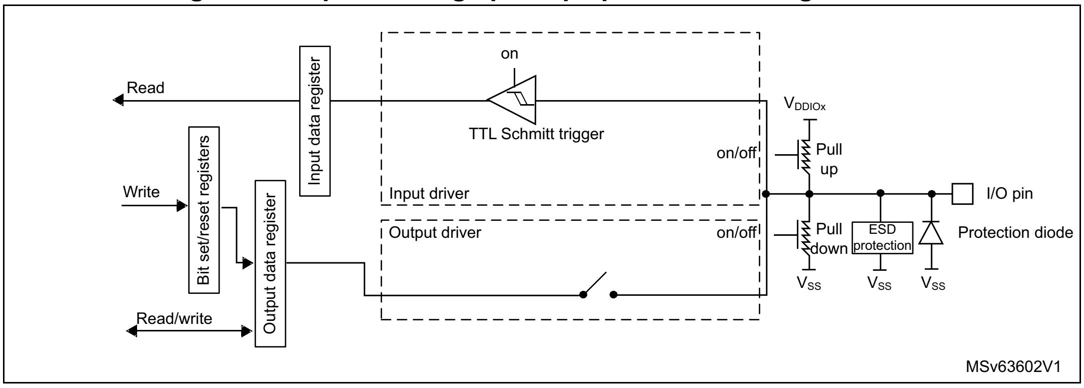
14.4.10 GPIO input configuration

When the I/O port is programmed as input:

- The output buffer is disabled.
- The Schmitt trigger input is activated.
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register provides the I/O state.

Figure 44 shows the input configuration of the I/O port bit.

Figure 44. Input floating / pull-up / pull-down configurations



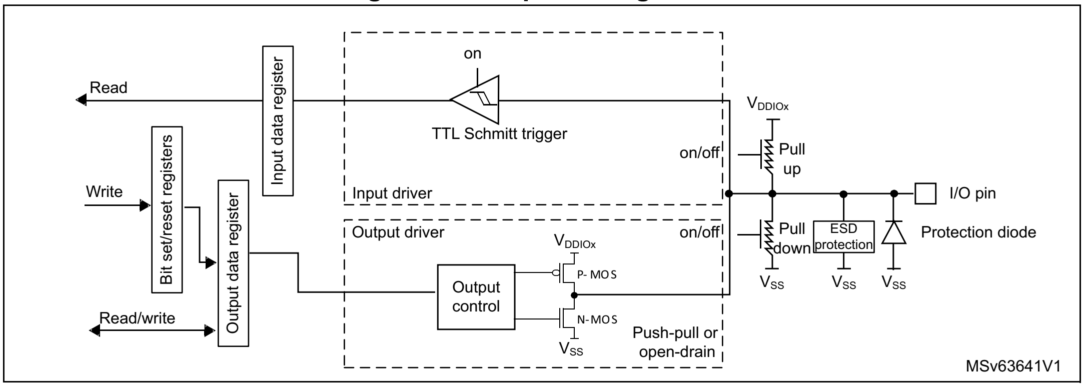
14.4.11 GPIO output configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
  - Open-drain mode: a 0 in the output register activates the N-MOS whereas a 1 in the output register leaves the port in Hi-Z (the P-MOS is never activated).
  - Push-pull mode: a 0 in the output register activates the N-MOS whereas a 1 in the output register activates the P-MOS.
- The Schmitt trigger input is activated.
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register gets the I/O state.
- A read access to the output data register gets the last written value.

Figure 45 shows the output configuration of the I/O port bit.

Figure 45. Output configuration



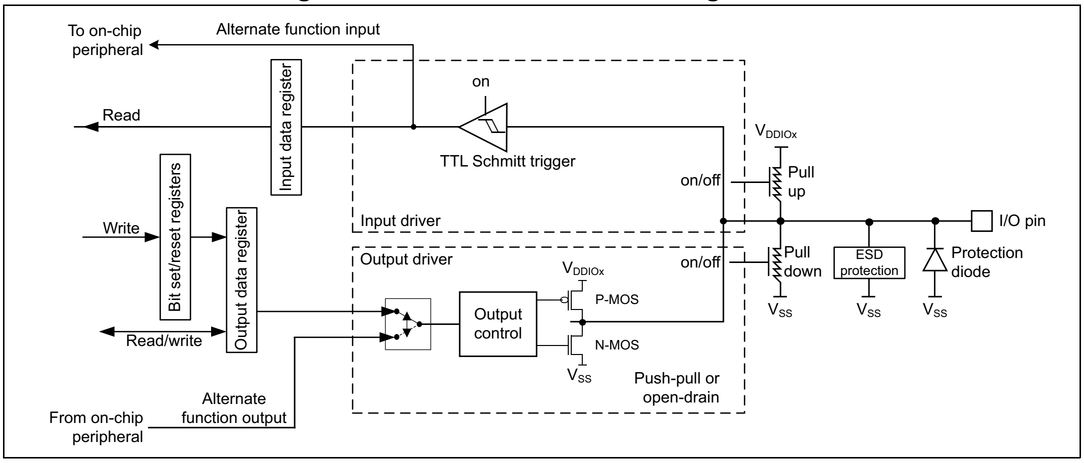
14.4.12 GPIO alternate function configuration

When the I/O port is programmed as alternate function:

- The output buffer can be configured in open-drain or push-pull mode.
- The output buffer is driven by the signals coming from the peripheral (transmitter enable and data).
- The Schmitt trigger input is activated.
- The weak pull-up and pull-down resistors are activated or not depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register gets the I/O state.

Figure 46 shows the alternate function configuration of the I/O port bit.

Figure 46. Alternate function configuration



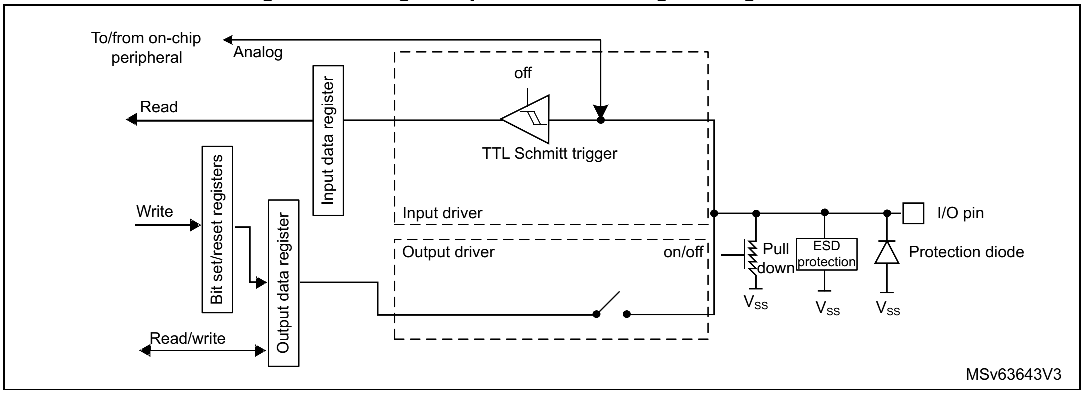
14.4.13 GPIO analog configuration

When the I/O port is programmed in 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-down resistor is activated depending on the value in the GPIOx_PUPDR register. The weak pull-up resistor is disabled by hardware.
- Read access to the input data register gets the value 0.

Figure 47 shows the high-impedance, analog-input configuration of the I/O port bits.

Figure 47. High-impedance analog configuration



14.4.14 GPIO using the LSE oscillator pins as GPIOs

When the LSE oscillator is switched off (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the LSE oscillator is switched on (by setting the LSEON bit in the RCC_BDCR1 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 OSC32_OUT pin can still be used as normal GPIO.

14.4.15 GPIO using GPIO pins with RTC

The PC13/PC14/PC15 GPIO functionality is lost when the Core domain is powered off (when the device enters Standby modes). 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 36.3: RTC functional description*

14.4.16 GPIO using PH3 as GPIO

PH3 may be used as boot pin (BOOT0) or as a GPIO. Depending on the nSWBOOT0 user option bit in the FLASH_OPTR, 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

14.4.17 GPIO TrustZone security

The TrustZone security is activated by the TZEN user option bit in the FLASH_OPTR. When the TrustZone is active (TZEN = 1), each I/O pin of GPIO port can be individually configured as secure through the GPIOx_SECCFGR register.

When the selected I/O pin is configured as secure, its corresponding configuration bits for alternate function, mode selection, I/O data are secure against a non-secure access. In case of non-secure access, these bits are RAZ/WI.

The I/Os with peripherals functions are also conditioned by the peripheral security configuration (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): 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 prevents 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 114 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), all registers bits are non-secure. The GPIOx_SECCFGR register is RAZ/WI.

Table 114. 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	

1. x = port index, y = port pin index, for values see *Table 112: GPIO implementation*.

As soon as at least one function is configured to be secure, the GPIO reset and clock control bits in the RCC are also secured.

14.4.18 GPIO 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).

14.4.19 GPIO 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.

14.4.20 GPIO standby retention

The I/O state can be retained in Standby mode. This is configured in the PWR.

14.5 GPIO port A registers

This section gives a detailed description of the GPIO port A registers.

The peripheral registers can be written in word, half word or byte mode.

14.5.1 GPIO port A mode register (GPIOA_MODER)

Address offset: 0x000

Reset value: 0xABFF FFFF

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 31:30 **MODE15[1:0]:** Port configuration I/O pin 15
These bits are written by software to configure the I/O mode.
Access can be protected by GPIOA SEC15.
00: Input mode
01: General purpose output mode
10: Alternate function mode
11: Analog mode (reset state)

Bits 29:28 **MODE14[1:0]:** Port configuration I/O pin 14

Bits 27:26 **MODE13[1:0]:** Port configuration I/O pin 13

Bits 25:24 **MODE12[1:0]:** Port configuration I/O pin 12

Bits 23:22 **MODE11[1:0]:** Port configuration I/O pin 11

Bits 21:20 **MODE10[1:0]:** Port configuration I/O pin 10

Bits 19:18 **MODE9[1:0]:** Port configuration I/O pin 9

Bits 17:16 **MODE8[1:0]:** Port configuration I/O pin 8

Bits 15:14 **MODE7[1:0]:** Port configuration I/O pin 7

Bits 13:12 **MODE6[1:0]:** Port configuration I/O pin 6

Bits 11:10 **MODE5[1:0]:** Port configuration I/O pin 5

Bits 9:8 **MODE4[1:0]:** Port configuration I/O pin 4
Note that bits 9:8 are reserved on STM32WBA54xx and STM32WBA52xx devices.

Bits 7:6 **MODE3[1:0]:** Port configuration I/O pin 3

Bits 5:4 **MODE2[1:0]:** Port configuration I/O pin 2

Bits 3:2 **MODE1[1:0]:** Port configuration I/O pin 1

Bits 1:0 **MODE0[1:0]:** Port configuration I/O pin 0

14.5.2 GPIO port A output type register (GPIOA_OTYPER)

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**
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OT[15:0]:** Port configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O output type.
Access can be protected by GPIOA SECy.
0: Output push-pull (reset state)
1: Output open-drain
Note that bit 4 is reserved on STM32WBA54xx and STM32WBA52xx devices.

14.5.3 GPIO port A output speed register (GPIOA_OSPEEDR)

Address offset: 0x008

Reset value: 0x0800 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		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 31:30 **OSPEED15[1:0]**: Port configuration I/O pin 15
These bits are written by software to configure the I/O output speed.
Access can be protected by GPIOA SEC15.
00: Low speed
01: Medium speed
10: High speed
11: Reserved

Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed.

Bits 29:28 **OSPEED14[1:0]:** Port configuration I/O pin 14

Bits 27:26 **OSPEED13[1:0]:** Port configuration I/O pin 13

Bits 25:24 **OSPEED12[1:0]:** Port configuration I/O pin 12

Bits 23:22 **OSPEED11[1:0]:** Port configuration I/O pin 11

Bits 21:20 **OSPEED10[1:0]:** Port configuration I/O pin 10

Bits 19:18 **OSPEED9[1:0]:** Port configuration I/O pin 9

Bits 17:16 **OSPEED8[1:0]:** Port configuration I/O pin 8

Bits 15:14 **OSPEED7[1:0]:** Port configuration I/O pin 7

Bits 13:12 **OSPEED6[1:0]:** Port configuration I/O pin 6

Bits 11:10 **OSPEED5[1:0]:** Port configuration I/O pin 5

Bits 9:8 **OSPEED4[1:0]:** Port configuration I/O pin 4
Note that bit 9:8 are reserved on STM32WBA54xx and STM32WBA52xx devices.

Bits 7:6 **OSPEED3[1:0]:** Port configuration I/O pin 3

Bits 5:4 **OSPEED2[1:0]:** Port configuration I/O pin 2

Bits 3:2 **OSPEED1[1:0]:** Port configuration I/O pin 1

Bits 1:0 **OSPEED0[1:0]:** Port configuration I/O pin 0

14.5.4 GPIO port A pull-up/pull-down register (GPIOA_PUPDR)

Address offset: 0x00C

Reset value: 0x6400 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		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 31:30 **PUPD15[1:0]:** Port configuration I/O pin 15
These bits are written by software to configure the I/O pull-up or pull-down
Access can be protected by GPIOA SEC15.
00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

Bits 29:28 **PUPD14[1:0]:** Port configuration I/O pin 14

Bits 27:26 **PUPD13[1:0]:** Port configuration I/O pin 13

Bits 25:24 **PUPD12[1:0]:** Port configuration I/O pin 12

Bits 23:22 **PUPD11[1:0]:** Port configuration I/O pin 11

Bits 21:20 **PUPD10[1:0]:** Port configuration I/O pin 10

Bits 19:18 **PUPD9[1:0]:** Port configuration I/O pin 9

Bits 17:16 **PUPD8[1:0]:** Port configuration I/O pin 8

Bits 15:14 **PUPD7[1:0]:** Port configuration I/O pin 7

Bits 13:12 **PUPD6[1:0]:** Port configuration I/O pin 6

Bits 11:10 **PUPD5[1:0]:** Port configuration I/O pin 5

Bits 9:8 **PUPD4[1:0]:** Port configuration I/O pin 4
Note that bit 9:8 are reserved on STM32WBA54xx and STM32WBA52xx devices.

Bits 7:6 **PUPD3[1:0]:** Port configuration I/O pin 3

Bits 5:4 **PUPD2[1:0]:** Port configuration I/O pin 2

Bits 3:2 **PUPD1[1:0]:** Port configuration I/O pin 1

Bits 1:0 **PUPD0[1:0]:** Port configuration I/O pin 0

14.5.5 GPIO port A input data register (GPIOA_IDR)

Address offset: 0x010

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ID[15:0]:** Port 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.
Access can be protected by GPIOA SECy.
Note that bit 4 is reserved on STM32WBA54xx and STM32WBA52xx devices.

14.5.6 GPIO port A output data register (GPIOA_ODR)

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**
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OD[15:0]:** Port output data I/O pin y (y = 15 to 0)
These bits can be read and written by software.
Access can be protected by GPIOA SECy.
Note that bit 4 is reserved on STM32WBA54xx and STM32WBA52xx devices.

Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOA_BSRR or GPIOA_BRR registers.

14.5.7 GPIO port A bit set/reset register (GPIOA_BSRR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BR[15:0]:** Port reset I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0.
Access can be protected by GPIOA SECy.
0: No action on the corresponding ODy bit
1: Resets the corresponding ODy bit
Note that bit 20 is reserved on STM32WBA54xx and STM32WBA52xx devices.

Note: If both BSy and BRy are set, BSy has priority.

Bits 15:0 **BS[15:0]:** Port set I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0.
Access can be protected by GPIOA SECy.
0: No action on the corresponding ODy bit
1: Sets the corresponding ODy bit
Note that bit 4 is reserved on STM32WBA54xx and STM32WBA52xx devices.

14.5.8 GPIO port A configuration lock register (GPIOA_LCKR)

Address offset: 0x01C

Reset value: 0x0000 0000

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOA_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	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.
Access can be protected by any GPIOA SECy.
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 LCKR[15:0] must not change.
Any error in the lock sequence aborts the LOCK.
After the first LOCK sequence on any bit of the port, any read access on the LCKK bit returns 1 until the next MCU reset or peripheral reset.

Bits 15:0 **LCK[15:0]:** Port 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
Access can be protected by GPIOA SECy.
0: Port configuration not locked
1: Port configuration locked
Note that bit 4 is reserved on STM32WBA54xx and STM32WBA52xx devices.

14.5.9 GPIO port A alternate function low register (GPIOA_AFRL)

Address offset: 0x020

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 31:28 **AFSEL7[3:0]:** Alternate function selection for port I/O pin 7
These bits are written by software to configure alternate function I/Os.
Access can be protected by GPIOA SECy.
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

Bits 27:24 **AFSEL6[3:0]:** Alternate function selection for port I/O pin 6

Bits 23:20 **AFSEL5[3:0]:** Alternate function selection for port I/O pin 5

Bits 19:16 **AFSEL4[3:0]:** Alternate function selection for port I/O pin 4
Note that bit 19:16 are reserved on STM32WBA54xx and STM32WBA52xx devices.

Bits 15:12 **AFSEL3[3:0]:** Alternate function selection for port I/O pin 3

Bits 11:8 **AFSEL2[3:0]:** Alternate function selection for port I/O pin 2

Bits 7:4 **AFSEL1[3:0]:** Alternate function selection for port I/O pin 1

Bits 3:0 **AFSEL0[3:0]:** Alternate function selection for port I/O pin 0

14.5.10 GPIO port A alternate function high register (GPIOA_AFRH)

Address offset: 0x024

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 31:28 **AFSEL15[3:0]:** Alternate function selection for port I/O pin 15
These bits are written by software to configure alternate function I/Os.
Access can be protected by GPIOA SEC15.
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

Bits 27:24 **AFSEL14[3:0]:** Alternate function selection for port I/O pin 14

Bits 23:20 **AFSEL13[3:0]:** Alternate function selection for port I/O pin 13

Bits 19:16 **AFSEL12[3:0]:** Alternate function selection for port I/O pin 12

Bits 15:12 **AFSEL11[3:0]:** Alternate function selection for port I/O pin 11

Bits 11:8 **AFSEL10[3:0]:** Alternate function selection for port I/O pin 10

Bits 7:4 **AFSEL9[3:0]:** Alternate function selection for port I/O pin 9

Bits 3:0 **AFSEL8[3:0]:** Alternate function selection for port I/O pin 8

14.5.11 GPIO port A bit reset register (GPIOA_BRR)

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**
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BR[15:0]:** Port reset I/O pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0.
Access can be protected by GPIOA SECy.
0: No action on the corresponding ODy bit
1: Reset the corresponding ODy bit
Note that bit 4 is reserved on STM32WBA54xx and STM32WBA52xx devices.

14.5.12 GPIO port A secure configuration register (GPIOA_SECCFGR)

Address offset: 0x030

Reset value: 0x0000 FFFF

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.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
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 **SEC[15:0]**: I/O pin of port secure bit enable y (y = 15 to 0)
These bits are written by software to enabled the security I/O port pin.
0: The I/O pin is non-secure
1: The I/O pin is secure. Refer to *Table 114* for all corresponding secured bits.
Note that bit 4 is reserved on STM32WBA54xx and STM32WBA52xx devices.

14.6 GPIO port B registers

This section gives a detailed description of the GPIO port B registers.

The peripheral registers can be written in word, half word or byte mode.

14.6.1 GPIO port B mode register (GPIOB_MODER)

Address offset: 0x000

Reset value: 0xFFFF FEBF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **MODE15[1:0]:** Port configuration I/O pin 15
These bits are written by software to configure the I/O mode.
Access can be protected by GPIOB SEC15.
00: Input mode
01: General purpose output mode
10: Alternate function mode
11: Analog mode (reset state)

Bits 29:28 **MODE14[1:0]:** Port configuration I/O pin 14

Bits 27:26 **MODE13[1:0]:** Port configuration I/O pin 13

Bits 25:24 **MODE12[1:0]:** Port configuration I/O pin 12

Bits 23:22 **MODE11[1:0]:** Port configuration I/O pin 11

Bits 21:20 **MODE10[1:0]:** Port configuration I/O pin 10
Note that bits 21:20 are reserved on STM32WBA55xx devices.

Bits 19:18 **MODE9[1:0]:** Port configuration I/O pin 9

Bits 17:16 **MODE8[1:0]:** Port configuration I/O pin 8

Bits 15:14 **MODE7[1:0]:** Port configuration I/O pin 7

Bits 13:12 **MODE6[1:0]:** Port configuration I/O pin 6

Bits 11:10 **MODE5[1:0]:** Port configuration I/O pin 5

Bits 9:8 **MODE4[1:0]:** Port configuration I/O pin 4

Bits 7:6 **MODE3[1:0]:** Port configuration I/O pin 3

Bits 5:4 **MODE2[1:0]:** Port configuration I/O pin 2

Bits 3:2 **MODE1[1:0]:** Port configuration I/O pin 1

Bits 1:0 **MODE0[1:0]:** Port configuration I/O pin 0

14.6.2 GPIO port B output type register (GPIOB_OTYPER)

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**
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OT[15:0]:** Port configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O output type.
Access can be protected by GPIOB SECy.
0: Output push-pull (reset state)
1: Output open-drain
Note that bit 10 is reserved on STM32WBA55xx devices.

14.6.3 GPIO port B output speed register (GPIOB_OSPEEDR)

Address offset: 0x008

Reset value: 0x0000 0080

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 31:30 **OSPEED15[1:0]**: Port configuration I/O pin 15
These bits are written by software to configure the I/O output speed.
Access can be protected by GPIOB SEC15.
00: Low speed
01: Medium speed
10: High speed
11: Reserved

Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed.

Bits 29:28 **OSPEED14[1:0]:** Port configuration I/O pin 14

Bits 27:26 **OSPEED13[1:0]:** Port configuration I/O pin 13

Bits 25:24 **OSPEED12[1:0]:** Port configuration I/O pin 12

Bits 23:22 **OSPEED11[1:0]:** Port configuration I/O pin 11

Bits 21:20 **OSPEED10[1:0]:** Port configuration I/O pin 10
Note that bits 21:20 are reserved on STM32WBA55xx devices.

Bits 19:18 **OSPEED9[1:0]:** Port configuration I/O pin 9

Bits 17:16 **OSPEED8[1:0]:** Port configuration I/O pin 8

Bits 15:14 **OSPEED7[1:0]:** Port configuration I/O pin 7

Bits 13:12 **OSPEED6[1:0]:** Port configuration I/O pin 6

Bits 11:10 **OSPEED5[1:0]:** Port configuration I/O pin 5

Bits 9:8 **OSPEED4[1:0]:** Port configuration I/O pin 4

Bits 7:6 **OSPEED3[1:0]:** Port configuration I/O pin 3

Bits 5:4 **OSPEED2[1:0]:** Port configuration I/O pin 2

Bits 3:2 **OSPEED1[1:0]:** Port configuration I/O pin 1

Bits 1:0 **OSPEED0[1:0]:** Port configuration I/O pin 0

14.6.4 GPIO port B pull-up/pull-down register (GPIOB_PUPDR)

Address offset: 0x00C

Reset value: 0x0000 0100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **PUPD15[1:0]:** Port configuration I/O pin 15
These bits are written by software to configure the I/O pull-up or pull-down
Access can be protected by GPIOB SEC15.
00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

Bits 29:28 **PUPD14[1:0]:** Port configuration I/O pin 14

Bits 27:26 **PUPD13[1:0]:** Port configuration I/O pin 13

Bits 25:24 **PUPD12[1:0]:** Port configuration I/O pin 12

Bits 23:22 **PUPD11[1:0]:** Port configuration I/O pin 11

Bits 21:20 **PUPD10[1:0]:** Port configuration I/O pin 10
Note that bits 21:20 are reserved on STM32WBA55xx devices.

Bits 19:18 **PUPD9[1:0]:** Port configuration I/O pin 9

Bits 17:16 **PUPD8[1:0]:** Port configuration I/O pin 8

Bits 15:14 **PUPD7[1:0]:** Port configuration I/O pin 7

Bits 13:12 **PUPD6[1:0]:** Port configuration I/O pin 6

Bits 11:10 **PUPD5[1:0]:** Port configuration I/O pin 5

Bits 9:8 **PUPD4[1:0]:** Port configuration I/O pin 4

Bits 7:6 **PUPD3[1:0]:** Port configuration I/O pin 3

Bits 5:4 **PUPD2[1:0]:** Port configuration I/O pin 2

Bits 3:2 **PUPD1[1:0]:** Port configuration I/O pin 1

Bits 1:0 **PUPD0[1:0]:** Port configuration I/O pin 0

14.6.5 GPIO port B input data register (GPIOB_IDR)

Address offset: 0x010

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ID[15:0]:** Port 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.
Access can be protected by GPIOB SECy.
Note that bit 10 is reserved on STM32WBA55xx devices.

14.6.6 GPIO port B output data register (GPIOB_ODR)

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**
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OD[15:0]:** Port output data I/O pin y (y = 15 to 0)
These bits can be read and written by software.
Access can be protected by GPIOB SECy.
Note that bit 10 is reserved on STM32WBA55xx devices.

Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOB_BSRR or GPIOB_BRR registers.

14.6.7 GPIO port B bit set/reset register (GPIOB_BSRR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BR[15:0]:** Port reset I/O pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0.
Access can be protected by GPIOB SECy.
0: No action on the corresponding ODy bit
1: Resets the corresponding ODy bit
Note that bit 26 is reserved on STM32WBA55xx devices.

Note: If both BSy and BRy are set, BSy has priority.

Bits 15:0 **BS[15:0]:** Port set I/O pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0.
Access can be protected by GPIOB SECy.
0: No action on the corresponding ODy bit
1: Sets the corresponding ODy bit
Note that bit 10 is reserved on STM32WBA55xx devices.

14.6.8 GPIO port B configuration lock register (GPIOB_LCKR)

Address offset: 0x01C

Reset value: 0x0000 0000

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOB_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	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.
Access can be protected by any GPIOB SECy.
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 LCKR[15:0] must not change.
Any error in the lock sequence aborts the LOCK.
After the first LOCK sequence on any bit of the port, any read access on the LCKK bit returns 1 until the next MCU reset or peripheral reset.

Bits 15:0 **LCK[15:0]:** Port 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
Access can be protected by GPIOB SECy.
0: Port configuration not locked
1: Port configuration locked
Note that bit 10 is reserved on STM32WBA55xx devices.

14.6.9 GPIO port B alternate function low register (GPIOB_AFRL)

Address offset: 0x020

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 31:28 **AFSEL7[3:0]:** Alternate function selection for port I/O pin 7

These bits are written by software to configure alternate function I/Os.
Access can be protected by GPIOB SEC7.
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

Bits 27:24 **AFSEL6[3:0]:** Alternate function selection for port I/O pin 6

Bits 23:20 **AFSEL5[3:0]:** Alternate function selection for port I/O pin 5

Bits 19:16 **AFSEL4[3:0]:** Alternate function selection for port I/O pin 4

Bits 15:12 **AFSEL3[3:0]:** Alternate function selection for port I/O pin 3

Bits 11:8 **AFSEL2[3:0]:** Alternate function selection for port I/O pin 2

Bits 7:4 **AFSEL1[3:0]:** Alternate function selection for port I/O pin 1

Bits 3:0 **AFSEL0[3:0]:** Alternate function selection for port I/O pin 0

14.6.10 GPIO port B alternate function high register (GPIOB_AFRH)

Address offset: 0x024

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 31:28 **AFSEL15[3:0]:** Alternate function selection for port I/O pin 15
These bits are written by software to configure alternate function I/Os.
Access can be protected by GPIOB SEC15.
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

Bits 27:24 **AFSEL14[3:0]:** Alternate function selection for port I/O pin 14

Bits 23:20 **AFSEL13[3:0]:** Alternate function selection for port I/O pin 13

Bits 19:16 **AFSEL12[3:0]:** Alternate function selection for port I/O pin 12

Bits 15:12 **AFSEL11[3:0]:** Alternate function selection for port I/O pin 11

Bits 11:8 **AFSEL10[3:0]:** Alternate function selection for port I/O pin 10
Note that bits11:8 are reserved on STM32WBA55xx devices.

Bits 7:4 **AFSEL9[3:0]:** Alternate function selection for port I/O pin 9

Bits 3:0 **AFSEL8[3:0]:** Alternate function selection for port I/O pin 8

14.6.11 GPIO port B bit reset register (GPIOB_BRR)

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
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BR[15:0]:** Port reset I/O pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0.
Access can be protected by GPIOB SECy.
0: No action on the corresponding ODy bit
1: Reset the corresponding ODy bit
Note that bit 10 is reserved on STM32WBA55xx devices.

14.6.12 GPIO port B secure configuration register (GPIOB_SECCFGR)

Address offset: 0x030

Reset value: 0x0000 FFFF

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.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
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 **SEC[15:0]**: I/O pin of port secure bit enable y (y = 15 to 0)
These bits are written by software to enabled the security I/O port pin.
0: The I/O pin is non-secure
1: The I/O pin is secure. Refer to *Table 114* for all corresponding secured bits.
Note that bit 10 is reserved on STM32WBA55xx devices.

14.7 GPIO port C registers

This section gives a detailed description of the GPIO port C registers.

The peripheral registers can be written in word, half word or byte mode.

14.7.1 GPIO port C mode register (GPIOC_MODER)

Address offset: 0x000

Reset value: 0xFC00 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 **MODE15[1:0]:** Port C configuration I/O pin 15
These bits are written by software to configure the I/O mode.
Access can be protected by GPIOC SEC15.
00: Input mode
01: General purpose output mode
10: Alternate function mode
11: Analog mode (reset state)

Bits 29:28 **MODE14[1:0]:** Port C configuration I/O pin 14

Bits 27:26 **MODE13[1:0]:** Port C configuration I/O pin 13

Bits 25:0 Reserved, must be kept at reset value.

14.7.2 GPIO port C output type register (GPIOC_OTYPER)

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**
OT15	OT14	OT13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **OT[15:13]:** Port C configuration I/O pin y (y = 15 to 13)
These bits are written by software to configure the I/O output type.
Access can be protected by GPIOC SECy.
0: Output push-pull (reset state)
1: Output open-drain

Bits 12:0 Reserved, must be kept at reset value.

14.7.3 GPIOC port output speed register (GPIOC_OSPEEDR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw										
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 **OSPEED15[1:0]**: Port C configuration I/O pin 15
These bits are written by software to configure the I/O output speed.
Access can be protected by GPIOC SEC15.
00: Low speed
01: Medium speed
10: High speed
11: Reserved

Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed.

Bits 29:28 **OSPEED14[1:0]:** Port C configuration I/O pin 14

Bits 27:26 **OSPEED13[1:0]:** Port C configuration I/O pin 13

Bits 25:0 Reserved, must be kept at reset value.

14.7.4 GPIO port C pull-up/pull-down register (GPIOC_PUPDR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw										
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 **PUPD15[1:0]:** Port C configuration I/O pin 15
These bits are written by software to configure the I/O pull-up or pull-down
Access can be protected by GPIOC SEC15.
00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

Bits 29:28 **PUPD14[1:0]:** Port C configuration I/O pin 14

Bits 27:26 **PUPD13[1:0]:** Port C configuration I/O pin 13

Bits 25:0 Reserved, must be kept at reset value.

14.7.5 GPIO port C input data register (GPIOC_IDR)

Address offset: 0x010

Reset value: 0x0000 X000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID15	ID14	ID13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r													

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **ID[15:13]:** Port C input data I/O pin y (y = 15 to 13)
These bits are read-only. They contain the input value of the corresponding I/O port.
Access can be protected by GPIOC SECy.

Bits 12:0 Reserved, must be kept at reset value.

14.7.6 GPIO port C output data register (GPIOC_ODR)

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**
OD15	OD14	OD13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **OD[15:13]:** Port C output data I/O pin y (y = 15 to 13)

These bits can be read and written by software.

Access can be protected by GPIOC SECy.

Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOC_BSRR or GPIOC_BRR registers.

Bits 12:0 Reserved, must be kept at reset value.

14.7.7 GPIO port C bit set/reset register (GPIOC_BSRR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w	w	w													
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BS15	BS14	BS13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w	w	w													

Bits 31:29 **BR[15:13]:** Port C reset I/O pin y (y = 15 to 13)

These bits are write-only. A read to these bits returns the value 0.

Access can be protected by GPIOC SECy.

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 28:16 Reserved, must be kept at reset value.

Bits 15:13 **BS[15:13]:** Port C set I/O pin y (y = 15 to 13)

These bits are write-only. A read to these bits returns the value 0.

Access can be protected by GPIOC SECy.

0: No action on the corresponding ODy bit

1: Sets the corresponding ODy bit

Bits 12:0 Reserved, must be kept at reset value.

14.7.8 GPIO port C configuration lock register (GPIOC_LCKR)

Address offset: 0x01C

Reset value: 0x0000 0000

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:13] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:13] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOC_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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.
Access is protected by any GPIOC SECy.
0: Port configuration lock key not active
1: Port configuration lock key active. The GPIOC_LCKR register is locked until the next MCU reset or peripheral reset.
- LOCK key write sequence:
WR LCKR[16] = 1 + LCKR[15:13]
WR LCKR[16] = 0 + LCKR[15:13]
WR LCKR[16] = 1 + LCKR[15:13]
- 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[15:13] must not change.
Any error in the lock sequence aborts the LOCK.
After the first LOCK sequence on any bit of the port, any read access on the LCKK bit returns 1 until the next MCU reset or peripheral reset.

Bits 15:13 **LCK[15:13]:** Port C lock I/O pin y (y = 15 to 13)

These bits are read/write but can only be written when the LCKK bit is 0
Access can be protected by GPIOC SECy.
0: Port configuration not locked
1: Port configuration locked

Bits 12:0 Reserved, must be kept at reset value.

14.7.9 GPIO port C alternate function high register (GPIOC_AFRH)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:28 **AFSEL15[3:0]:** Alternate function selection for port C I/O pin 15
These bits are written by software to configure alternate function I/Os.
Access can be protected by GPIOC SEC15.
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

Bits 27:24 **AFSEL14[3:0]:** Alternate function selection for port C I/O pin 14

Bits 23:20 **AFSEL13[3:0]:** Alternate function selection for port C I/O pin 13

Bits 19:0 Reserved, must be kept at reset value.

14.7.10 GPIO port C bit reset register (GPIOC_BRR)

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**
BR15	BR14	BR13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w	w	w													

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **BR[15:13]:** Port reset I/O pin y (y = 15 to 13)
These bits are write-only. A read to these bits returns the value 0.
Access can be protected by GPIOC SECy.
0: No action on the corresponding ODy bit
1: Reset the corresponding ODy bit

Bits 12:0 Reserved, must be kept at reset value.

14.7.11 GPIO port C secure configuration register (GPIOC_SECCFGR)

Address offset: 0x030

Reset value: 0x0000 FFFF

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.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **SEC[15:13]**: I/O pin of port C secure bit enable y (y = 15 to 13)
These bits are written by software to enabled the security I/O port pin.
0: The I/O pin is non-secure
1: The I/O pin is secure. Refer to *Table 114* for all corresponding secured bits.

Bits 12:0 Reserved, must be kept at reset value.

14.8 GPIO port H registers

This section gives a detailed description of the GPIO port H registers.

The peripheral registers can be written in word, half word or byte mode.

14.8.1 GPIO port H mode register (GPIOH_MODER)

Address offset: 0x000

Reset value: 0x0000 00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
								rw	rw						

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

Bits 7:6 **MODE3[1:0]:** Port H configuration I/O pin 3
These bits are written by software to configure the I/O mode.
Access can be protected by GPIOH SEC3.
00: Input mode
01: General purpose output mode
10: Alternate function mode
11: Analog mode (reset state)

Bits 5:0 Reserved, must be kept at reset value.

14.8.2 GPIO port H output type register (GPIOH_OTYPER)

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.	OT3	Res.	Res.	Res.
												rw			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **OT3:** Port H configuration I/O pin 3
This bit is written by software to configure the I/O output type.
Access can be protected by GPIOH SEC3.
0: Output push-pull (reset state)
1: Output open-drain

Bits 2:0 Reserved, must be kept at reset value.

14.8.3 GPIO port H output speed register (GPIOH_OSPEEDR)

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.	OSPEED3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
								rw	rw						

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

Bits 7:6 **OSPEED3[1:0]**: Port H configuration I/O pin 3

These bits are written by software to configure the I/O output speed.
Access can be protected by GPIOH SEC3.
00: Low speed
01: Medium speed
10: High speed
11: Reserved

Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed.

Bits 5:0 Reserved, must be kept at reset value.

14.8.4 GPIO port H pull-up/pull-down register (GPIOH_PUPDR)

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.	PUPD3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
								rw	rw						

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

Bits 7:6 **PUPD3[1:0]:** Port H configuration I/O pin 3

These bits are written by software to configure the I/O pull-up or pull-down
Access can be protected by GPIOH SEC3.
00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

Bits 5:0 Reserved, must be kept at reset value.

14.8.5 GPIO port H input data register (GPIOH_IDR)

Address offset: 0x010

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**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID3	Res.	Res.	Res.
												r			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ID3:** Port H input data I/O pin 3
This bit is read-only. It contain the input value of the corresponding I/O port.
Access can be protected by GPIOH SEC3.

Bits 2:0 Reserved, must be kept at reset value.

14.8.6 GPIO port H output data register (GPIOH_ODR)

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.	OD3	Res.	Res.	Res.
												rw			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **OD3:** Port H output data I/O pin 3
This bits can be read and written by software.
Access can be protected by GPIOH SEC3.
Note: For atomic bit set/reset, the OD bit can be individually set and/or reset by writing to the GPIOH_BSRR or GPIOH_BRR registers.

Bits 2:0 Reserved, must be kept at reset value.

14.8.7 GPIO port H bit set/reset register (GPIOH_BSRR)

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.	BR3	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.	Res.	Res.	Res.	Res.	Res.	BS3	Res.	Res.	Res.
												w			

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

Bit 19 **BR3:** Port H reset I/O pin 3
This bit is write-only. A read to this bit returns the value 0.
Access can be protected by GPIOH SEC3.
0: No action on the corresponding OD3 bit
1: Resets the corresponding OD3 bit
Note: If both BS3 and BR3 are set, BS3 has priority.

Bits 18:4 Reserved, must be kept at reset value.

Bit 3 **BS3:** Port H set I/O pin 3
This bit is write-only. A read to this bit returns the value 0.
Access can be protected by GPIOH SEC3.
0: No action on the corresponding OD3 bit
1: Sets the corresponding OD3 bit

Bits 2:0 Reserved, must be kept at reset value.

14.8.8 GPIO port H configuration lock register (GPIOH_LCKR)

Address offset: 0x01C

Reset value: 0x0000 0000

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bit [3] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[3] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOH_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCK3	Res.	Res.	Res.
												rw			

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

Bit 16 **LCKK:** Lock key
This bit can be read any time. It can only be modified using the lock key write sequence.
Access is protected by GPIOH SEC3.
0: Port configuration lock key not active
1: Port configuration lock key active. The GPIOH_LCKR register is locked until the next MCU reset or peripheral reset.
- LOCK key write sequence:
WR LCKR[16] = 1 + LCKR[3]
WR LCKR[16] = 0 + LCKR[3]
WR LCKR[16] = 1 + LCKR[3]
- 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 LCK3 must not change.
Any error in the lock sequence aborts the LOCK.
After the first LOCK sequence on any bit of the port, any read access on the LCKK bit returns 1 until the next MCU reset or peripheral reset.

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **LCK3:** Port H lock I/O pin 3

This bit is read/write but can only be written when the LCKK bit is 0
Access can be protected by GPIOH SEC3.
0: Port configuration not locked
1: Port configuration locked

Bits 2:0 Reserved, must be kept at reset value.

14.8.9 GPIO port H alternate function low register (GPIOH_AFRL)

Address offset: 0x020

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw												

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **AFSEL3[3:0]:** Alternate function selection for port H I/O pin 3

These bits are written by software to configure alternate function I/Os.
Access can be protected by GPIOH SEC3.
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

Bits 11:0 Reserved, must be kept at reset value.

14.8.10 GPIO port H bit reset register (GPIOH_BRR)

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.	Res.	Res.	Res.	BR3	Res.	Res.	Res.
												w			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **BR3:** Port H reset I/O pin 3
This bit is write-only. A read to this bit returns the value 0.
Access can be protected by GPIOH SEC3.
0: No action on the corresponding OD3 bit
1: Reset the corresponding OD3 bit

Bits 2:0 Reserved, must be kept at reset value.

14.8.11 GPIO port H secure configuration register (GPIOH_SECCFGR)

Address offset: 0x030

Reset value: 0x0000 FFFF

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.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**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	Res.	Res.	Res.
												w			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **SEC3**: I/O pin of port H secure bit enable 3
This bit is written by software to enabled the security I/O port pin.
0: The I/O pin is non-secure
1: The I/O pin is secure. Refer to *Table 114* for all corresponding secured bits.

Bits 2:0 Reserved, must be kept at reset value.

14.8.12 GPIOA register map

Table 115. GPIOA 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	GPIOA_MODER	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0](1)		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x004	GPIOA_OTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4(1)	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	GPIOA_OSPEEDR	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]		OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0](1)		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	GPIOA_PUPDR	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]		PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0](1)		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
0x010	GPIOA_IDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4(1)	ID3	ID2	ID1	ID0
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x014	GPIOA_ODR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4(1)	OD3	OD2	OD1	OD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	GPIOA_BSRR	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4(1)	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4(1)	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
0x01C	GPIOA_LCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4(1)	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	GPIOA_AFRL	AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0](1)				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
0x024	GPIOA_AFRH	AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]				AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	GPIOA_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4(1)	BR3	BR2	BR1	BR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	Reserved	Reserved																															

Table 115. GPIOA 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	GPIOA_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(1)	SEC3	SEC2	SEC1	SEC0
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x034 to 0x3FC	Reserved	Reserved																															

1. Bit(s) reserved on STM32WBA54xx and STM32WBA52xx devices.

14.8.13 GPIOB register map

Table 116. GPIOB 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	GPIOB_MODER	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0][1]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1
0x004	GPIOB_OTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT15	OT14	OT13	OT12	OT11	OT10[1]	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
0x008	GPIOB_OSPEEDR	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0][1]		OSPEED9[1:0]		OSPEED8[1:0]		OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x00C	GPIOB_PUPDR	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0][1]		PUPD9[1:0]		PUPD8[1:0]		PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0x010	GPIOB_IDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID15	ID14	ID13	ID12	ID11	ID10[1]	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
0x014	GPIOB_ODR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD15	OD14	OD13	OD12	OD11	OD10[1]	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
0x018	GPIOB_BSRR	BR15	BR14	BR13	BR12	BR11	BR10[1]	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10[1]	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
0x01C	GPIOB_LCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10[1]	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
0x020	GPIOB_AFRL	AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]				AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	GPIOB_AFRH	AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]				AFSEL11[3:0]				AFSEL10[3:0][1]				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
0x028	GPIOB_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	BR14	BR13	BR12	BR11	BR10[1]	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
0x02C	Reserved	Reserved																															

Table 116. GPIOB 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	**GPIOB_SECCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10(1)	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
0x034 to 0x3FC	Reserved	Reserved																															

1. Bit(s) reserved on STM32WBA55xx devices.

14.8.14 GPIOC register map

Table 117. GPIOC 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	GPIOC_MODER	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	1	1	1	1	1	1																										
0x004	GPIOC_OTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT15	OT14	OT13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0	0	0													
0x008	GPIOC_OSPEEDR	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0																										
0x00C	GPIOC_PUPDR	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0																										
0x010	GPIOC_IDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID15	ID14	ID13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	X	X	X													
0x014	GPIOC_ODR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD15	OD14	OD13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0	0	0													
0x018	GPIOC_BSRR	BR15	BR14	BR13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BS15	BS14	BS13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0														0	0	0													
0x01C	GPIOC_LCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																0	0	0	0													
0x020	Reserved	Reserved																															
0x024	GPIOC_AFRH	AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0																				
0x028	GPIOC_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	BR14	BR13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0	0	0													
0x02C	Reserved	Reserved																															
0x030	GPIOC_SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC15	SEC14	SEC13	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	1	1	1													
0x034 to 0x3FC	Reserved	Reserved																															

14.8.15 GPIOH register map

Table 118. GPIOH 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	GPIOH_MODER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																									1	1						
0x004	GPIOH_OTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT3	Res.	Res.	Res.
	Reset value																													0			
0x008	GPIOH_OSPEEDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSPEED3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																									0	0						
0x00C	GPIOH_PUPDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PUPD3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																									0	0						
0x010	GPIOH_IDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID3	Res.	Res.	Res.
	Reset value																													X			
0x014	GPIOH_ODR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD3	Res.	Res.	Res.
	Reset value																													0			
0x018	GPIOH_BSRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BS3	Res.	Res.	Res.
	Reset value													0																0			
0x01C	GPIOH_LCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCK3	Res.	Res.	Res.
	Reset value																0													0			
0x020	GPIOH_AFRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFSEL3[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0	0	0	0												
0x024	Reserved	Reserved																															
0x028	GPIOH_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR3	Res.	Res.	Res.
	Reset value																													0			
0x02C	Reserved	Reserved																															
0x030	GPIOH_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	Res.	Res.	Res.
	Reset value																													1			
0x034 to 0x3FC	Reserved	Reserved																															

Refer to *Section 2.3.2: Memory map and register boundary addresses* for the register boundary addresses.

15 System configuration controller (SYSCFG)

15.1 SYSCFG main features

The 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 I^2C fast-mode plus high-drive mode on some I/Os and voltage booster for I/Os analog switches
- Managing the I/O compensation cell
- Provides memory erase status
- Communication channel with the RSS

15.2 SYSCFG functional description

15.2.1 I/O compensation cell management

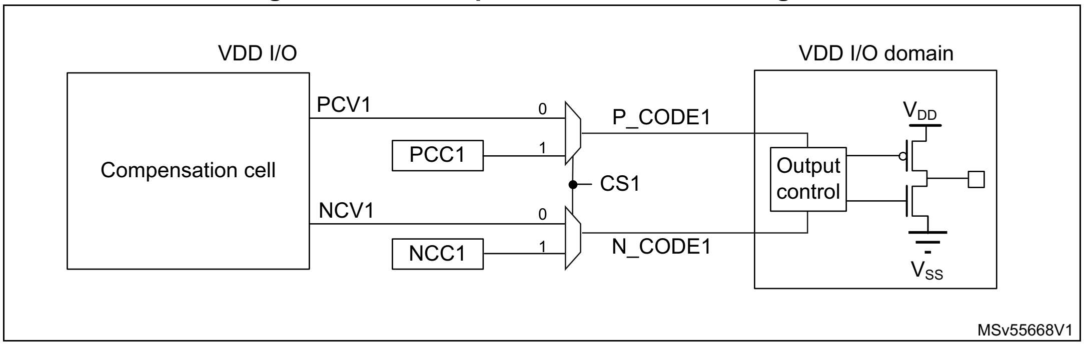
The I/O compensation cell generates an 8-bit value for the I/O buffer (4 bits for N-MOS and 4 bits for P-MOS), which depends on PVT operating conditions such as process, voltage, and temperature. These bits are used to control the current slew-rate and output impedance in the I/O buffer. A compensation cell is embedded for the I/Os (supplied by V_{DD}).

By default the compensation cells are disabled, and a fixed code is applied to all the I/Os.

When enabled, the compensation cell tracks the PVT, and the 8-bit code PCV1 and NCV1 for I/Os supplied by V_{DD} is available in SYSCFG_CCVR once the RDY1 is set. If the CS1 bit is cleared, the I/O receives the code from SYSCFG_CCVR, resulting from the compensation cell.

To optimize the trimming, the code can be adjusted through SYSCFG_CCCR. A set of bits is available, namely PCC1/NCC1 for the V_{DD} power rail. It can be selected through CS1 bit in SYSCFG_CCCSR (see *Figure 48*).

Figure 48. I/O compensation cell block diagram



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 CS bit and disable the compensation cell.

Table 119. Effect of low-power modes on I/O compensation

<table>
<tr><th>Mode</th><th>Description</th></tr>
<tr><td>Sleep</td><td rowspan="2">Compensation value applied on I/Os.
Compensation cell controlled by register SYSCFG_CCCSR bit EN.</td></tr>
<tr><td>Stop</td></tr>
<tr><td>Standby</td><td>Default compensation value applied on I/Os. Compensation cell disabled.</td></tr>
</table>

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 user option bit in the FLASH_OPTR.

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.

As soon as at least one function is configured to be secured, the SYSCFG reset and clock control bits in the RCC are also secured.

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.

Table 120 shows the register security overview.

Table 120. TrustZone security and privilege register accesses

<table>
<tr><th>SYSCFG register name</th><th colspan="2">Read/write access</th><th>Privileged /unprivileged access</th></tr>
<tr><th>TrustZone configuration(1)</th><th>TZEN = 1</th><th>TZEN = 0</th><th>Not applicable</th></tr>
<tr><td>SYSCFG_SECCFGR</td><td>Read: no restriction
Write: secure access only
Non-secure write is WI and generates an illegal access event.</td><td rowspan="2">RAZ/WI</td><td>Read: no restriction
FPUSEC privileged write only
Other bits write: no restriction</td></tr>
<tr><td>SYSCFG_CSLCKR</td><td>Read/Write: secure access only
Non-secure access is RAZ/WI and generates and an illegal access event.</td><td>Privileged only
Unprivileged: RAZ/WI</td></tr>
</table>

Table 120. TrustZone security and privilege register accesses (continued)

<table>
<tr><th>SYSCFG register name</th><th colspan="2">Read/write access</th><th>Privileged /unprivileged access</th></tr>
<tr><th>TrustZone configuration[1]</th><th>TZEN = 1</th><th>TZEN = 0</th><th>Not applicable</th></tr>
<tr><td>SYSCFG_FPUIMR</td><td>– FPUSEC = 1:
Read/Write: secure access only
Non-secure access is RAZ/WI and generates an illegal access event.
– FPUSEC = 0:
Read/Write: no restriction</td><td rowspan="6">No restriction</td><td rowspan="2">Privileged only
Unprivileged: RAZ/WI</td></tr>
<tr><td>SYSCFG_CNSLCKR</td><td>Read/write: no restriction</td></tr>
<tr><td>SYSCFG_CFGR1</td><td>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</td><td rowspan="5">No restriction</td></tr>
<tr><td>SYSCFG_CFGR2</td><td>– CLASSBSEC = 1:
Read/Write: secure access only
Non-secure access is RAZ/WI and generates an illegal access event.
– CLASSBSEC = 0:
Read/Write: no restriction</td></tr>
<tr><td>SYSCFG_MESR</td><td>– SYSCFGSEC = 1:
Read/Write: secure access only
Non-secure access is RAZ/WI and generates an illegal access event.
– SYSCFGSEC = 0:
Read/Write: no restriction</td></tr>
<tr><td>SYSCFG_CCCSR
SYSCFG_CCVR
SYSCFG_CCCR</td><td>– SYSCFGSEC = 1:
Read/Write: secure access only
Non-secure access is RAZ/WI and generates an illegal access event
– SYSCFGSEC = 0:
Read/Write: no restriction</td></tr>
<tr><td>SYSCFG_RSSCMDR</td><td>RAZ/WI if register access is not allowed[2]</td><td>RAZ/WI</td></tr>
</table>

1. TrustZone security is activated by the TZEN user option bit in the FLASH_OPTR.
2. Refer to register description for register access.

15.3 SYSCFG registers

15.3.1 SYSCFG secure configuration register (SYSCFG_SECCFGR)

Address offset: 0x000

Reset value: 0x0000 0000

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.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**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)

Address offset: 0x004

Reset value: 0x0000 0000

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 the GPIO port pin security configuration in the GPIO peripheral. 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.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PB3_FMP	PA15_FMP	PA7_FMP	PA6_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 **PB3_FMP**: Fast-mode Plus drive capability activation on PB3
This bit can be read and written only with secure access if PB3 is secure in GPIOB. This bit enables the Fast-mode Plus drive mode for PB3 when PB3 is not used by I2C peripheral. This can be used to dive a LED for instance.
Access can be protected by GPIOB SEC3.
0: PB3 pin operates in standard mode when not used by I2C peripheral
1: Fast-mode Plus mode is enabled on PAB3 pin and the GPIO speed control is bypassed.

Bit 18 **PA15_FMP**: Fast-mode Plus drive capability activation on PA15
This bit can be read and written only with secure access if PA15 is secure in GPIOA. This bit enables the Fast-mode Plus drive mode for PA15 when PA15 is not used by I2C peripheral. This can be used to dive a LED for instance.
Access can be protected by GPIOA SEC15.
0: PA15 pin operates in standard mode when not used by I2C peripheral
1: Fast-mode Plus mode is enabled on PA15 pin and the GPIO speed control is bypassed.

Bit 17 **PA7_FMP**: Fast-mode Plus drive capability activation on PA7
This bit can be read and written only with secure access if PA7 is secure in GPIOA. This bit enables the Fast-mode Plus drive mode for PA7 when PA7 is not used by I2C peripheral. This can be used to dive a LED for instance.
Access can be protected by GPIOA SEC7.
0: PA7 pin operates in standard mode when not used by I2C peripheral
1: Fast-mode Plus mode is enabled on PA7 pin and the GPIO speed control is bypassed.

Bit 16 **PA6_FMP**: Fast-mode Plus drive capability activation on PA6
This bit can be read and written only with secure access if PA6 is secure in GPIOA. This bit enables the Fast-mode Plus drive mode for PA6 when PA6 is not used by I2C peripheral. This can be used to dive a LED for instance.
Access can be protected by GPIOA SEC6.
0: PA6 pin operates in standard mode when not used by I2C peripheral
1: Fast-mode Plus mode is enabled on PA6 pin and the GPIO speed control is bypassed.

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

Bit 9 **ANASWVDD**: GPIO analog switch control voltage selection
Access can be protected by GTZC_TZSC ADC4SEC.
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 121 for setting.

Bit 8 **BOOSTEN**: I/O analog switch voltage booster enable
Access can be protected by GTZC_TZSC ADC4SEC.
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 121 for setting.

Bits 7:0 Reserved, must be kept at reset value.

Table 121 describes when the BOOSTEN and the ANASWVDD must be set or reset, depending on the voltage settings.

Table 121. 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)

Address offset: 0x008

Reset value: 0x0000 001F

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.

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

15	14	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)

Address offset: 0x00C

Reset value: 0x0000 0000

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.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**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_CSLCKR)

Address offset: 0x010

Reset value: 0x0000 0000

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.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**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	LOCK SVTAIRCR
													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, it 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, it 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, it disables write access to VTOR_S register, PRIS and BFHFNMINS bits in the AIRCR register.

0: VTOR_S register PRIS and BFHFNMINS bits in the AIRCR register write enabled

1: VTOR_S register PRIS and BFHFNMINS bits in the AIRCR register write disabled

15.3.6 SYSCFG configuration register 2 (SYSCFG_CFGR2)

Address offset: 0x014

Reset value: 0x0000 0000

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.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**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 TIM1/16/17 break input.

0: ECC double error disconnected from TIM1/16/17 break input

1: ECC double error connected to TIM1/16/17 break input

Bit 2 **PVDL**: PVD lock enable bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the PVD connection to TIM1/16/17 break input, as well as the PVDE and PVDLS[2:0] in the PWR register.

0: PVD interrupt disconnected from TIM1/16/17 break input. PVDE and PVDLS[2:0] bits can be programmed by the application.

1: PVD interrupt connected to TIM1/16/17 break input. PVDE and PVDLS[2:0] bits are read only.

Bit 1 **SPL**: SRAM2 parity lock bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the SRAM2 parity error signal connection to TIM1/16/17 break inputs.

0: SRAM2 parity error disconnected from TIM1/16/17 break inputs

1: SRAM2 parity error connected to TIM1/16/17 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 TIM1/16/17 break input.

0: Cortex-M33 LOCKUP output disconnected from TIM1/16/17 break inputs

1: Cortex-M33 LOCKUP output connected to TIM1/16/17 break inputs

15.3.7 SYSCFG memory erase status register (SYSCFG_MESR)

Address offset: 0x018

Power-on reset value: 0x0000 0000

System reset value: 0x0000 000X (bit MCLKR not affected by system reset)

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.

31	30	29	28	27	26	25	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 37: 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 ongoing 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, ICACHE, PKA SRAM erase is completed after power-on reset or tamper detection (refer to *Section 37: 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 ongoing if not yet cleared by software

1: Memory erase done

15.3.8 SYSCFG compensation cell control/status register (SYSCFG_CCCSR)

Address offset: 0x01C

Reset value: 0x0000 0002

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.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDY1	Res.	Res.	Res.	Res.	Res.	Res.	CS1	EN1
							r							rw	rw

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

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/Os compensation cell not ready

1: VDD I/Os compensation cell ready

*Note: The HSI16 clock is required for the compensation cell to work properly. The compensation cell ready bit (*RDY1*) is not set if the HSI16 clock is not enabled (HSION).*

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

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)

Address offset: 0x020

Reset value: 0x0000 0000

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.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCV1[3:0]				NCV1[3:0]			
								r	r	r	r	r	r	r	r

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

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/Os compensation cell code for PMOS transistors. This code is applied to the I/Os 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/Os compensation cell code for NMOS transistors. This code is applied to the I/Os compensation cell when the CS1 bit of the SYSCFG_CCCSR is reset.

15.3.10 SYSCFG compensation cell code register (SYSCFG_CCCR)

Address offset: 0x024

Reset value: 0x0000 0078

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.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCC1[3:0]				NCC1[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

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

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/Os compensation cell code for PMOS transistors. This code is applied to the I/Os 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/Os compensation cell code for NMOS transistors. This code is applied to the I/Os 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 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: 0x02C

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 122. 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
0x000	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
0x004	SYSCFG_CFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PB3_FMP	PA15_FMP	PA7_FMP	PA6_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								
0x008	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
0x00C	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
0x010	SYSCFG_CSLCKR	Res.	Res.	Res.	Res.	Res.	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
0x014	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
0x018	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
0x01C	SYSCFG_CCCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDY1	Res.	Res.	Res.	Res.	Res.	Res.	CS1	EN1
	Reset value																								0							1	0
0x020	SYSCFG_CCVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCV1[3:0]				NCV1[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0x024	SYSCFG_CCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCC1[3:0]				NCC1[3:0]			
	Reset value																									0	1	1	1	1	0	0	0
0x02C	SYSCFG_RSSCMDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSSCMD[15:0]															
	Power-on reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	System reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x030 to 0x3FC	Reserved	Reserved.																															

Refer to *Section 2.3* for the register boundary addresses.

16 Peripherals interconnect matrix

16.1 Introduction

Several peripherals have direct connections between them, enabling autonomous communication and/or synchronization between them. This saves CPU resources and, consequently power consumption. In addition, these hardware connections remove software latency and result in more predictable system design.

Depending on peripherals, these interconnections can operate in Run, Sleep, Stop 0, and Stop 1 modes.

16.2 Connection summary

Table 123. Peripherals interconnect matrix(1) (2)

Source	Destination																				
	TIM1	TIM2	TIM3(3)	TIM16	TIM17(3)	LPTIM1	LPTIM2(3)	ADC4	COMP1/2(3)	GPDMA1	IRTIM(3)	USART1	USART2(3)	LPUART1	I2C1(3)	I2C3	SPI1(3)	SPI3	TAMP	RTC	AES/SAES(3)
TIM1	-	1	1	-	-	-	-	2	14	-	-	-	-	-	-	-	-	-	-	-	-
TIM2	1	-	1	-	-	-	-	2	14	8	-	-	-	-	-	-	-	-	-	-	-
TIM3(3)	1	1	-	-	-	-	-	-	14	-	-	-	-	-	-	-	-	-	-	-	-
TIM16	1	1	1	-	-	-	-	-	-	-	12	-	-	-	-	-	-	-	-	-	-
TIM17(3)	1	1	1	-	-	-	-	-	-	-	12	-	-	-	-	-	-	-	-	-	-
LPTIM1	-	-	-	-	-	-	-	2	-	8	-	10	10	10	10	10	10	10	-	-	-
LPTIM2(3)	-	-	-	-	-	-	-	-	-	8	-	10	10	-	10	-	10	-	-	-	-
ADC4	3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	9	-	-
Temperature sensor	-	-	-	-	-	-	-	6	-	-	-	-	-	-	-	-	-	-	-	-	-
V_{CORE}	-	-	-	-	-	-	-	6	-	-	-	-	-	-	-	-	-	-	-	-	-
VREFINT	-	-	-	-	-	-	-	6	-	-	-	-	-	-	-	-	-	-	-	-	-
HSE32	-	-	-	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
HSI16	4	4	4	4	4	-	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-
LSE	-	4	-	4	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
LSI	-	-	-	4	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MCO	-	-	-	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
GPIO EXTI	-	-	-	-	-	5	5	2	-	8	-	10	10	10	10	10	10	10	-	-	-
RTC	-	-	-	-	-	5	5	-	-	8	-	10	10	10	10	10	10	10	9	-	-
TAMP	-	-	-	-	-	5	5	-	-	8	-	-	-	-	-	-	-	-	-	11	-
COMP1(3)	15	15	15	15	15	15	15	-	-	8	-	10	10	10	10	10	10	10	-	-	-

Table 123. Peripherals interconnect matrix(1) (2) (continued)

Source	Destination																				
	TIM1	TIM2	TIM3(3)	TIM16	TIM17(3)	LPTIM1	LPTIM2(3)	ADC4	COMP1/2(3)	GPDMA1	IRTIM(3)	USART1	USART2(3)	LPUART1	I2C1(3)	I2C3	SPI1(3)	SPI3	TAMP	RTC	AES/SAES(3)
COMP2(3)	15	15	15	15	15	15	15	-	-	8	-	10	10	10	10	10	10	-	-	-	-
GPDMA1	-	-	-	-	-	-	-	-	-	8	-	10	10	10	10	10	10	10	-	-	-
SYST ERR	7	-	-	7	7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Backup registers	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	13
FLASH	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	13
AES/SAES(3)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	9	-	13
PKA	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	9	-	-
TRNG	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	9	-	-
IWDG	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	9	-	-
DEBUG	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	9	-	-

1. Numbers in this table are links to corresponding subsections of *Section 16.3*.
2. The “-” symbol in gray cells means no interconnect.
3. Available only on STM32WBA52/54/55xx devices.

16.3 Interconnection details

16.3.1 Master to slave interconnection for timers

From timer (TIM1/TIM2/TIM3/TIM16/TIM17) to timer (TIM1/TIM2/TIM3).

Purpose

Some 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.

The synchronization modes are detailed in:

- *Section 29.3.30: Timer synchronization* for advanced-control timers (TIM1)
- *Section 30.4.23: Timer synchronization* and *Section 30.4.22: Timers and external trigger synchronization* for general-purpose timers (TIM2/TIM3)

Triggering signals

The output from master timer is on signal tim_trgo for TIM1/TIM2/TIM3, and tim_oc for TIM16/TIM17, following a configurable timer event. The input to slave timer is on signals tim_itr.

The possible master/slave connections are given in:

- *Table 259: Internal trigger connection* for TIM1
- *Table 283: TIMx internal trigger connection* for TIM2/TIM3

Active power mode(s)

Run, Sleep.

16.3.2 Triggers to ADC4

From EXTI and timers (TIM1/TIM2) and (LPTIM1) to ADC4.

Purpose

The timers TIM1/TIM2 can be used to generate the ADC4 trigger event through the timer outputs tim_oc or tim_trgo. Low-power timer LPTIM1 can be used to generate the ADC4 trigger event through output lptim1_ch1. In addition GPIO pin 15 via EXTI channel can be used to generate an ADC4 trigger event.

Triggering signals

The input trigger signal list and the description of the interconnection between ADC4 and timers and EXTI is given in:

- *Table 150: ADC interconnection*
- *Section 21.4.16: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN)*.
- *Section 21.4.21: Example timing diagrams (single/continuous modes hardware/software triggers)*

Active power mode(s)

Run, Sleep, and for LPTIM and EXTI also in Stop 0 and Stop 1.

16.3.3 ADC4 analog watchdog as trigger to timer

From ADC4 to TIM1/TIM3.

Purpose

The internal analog watchdog output signals from ADC4 are connected to timers. ADC4 can provide trigger event through watchdog signals to timer (TIM1/TIM3) in order to reset, start, stop or enable the counting.

Settings description of the ADC analog watchdog and timer trigger are provided in:

- *Section 29.3.6: External trigger input* for TIM1/TIM3
- *Table 260: Interconnect to the tim_etr input multiplexer* for TIM1
- *Table 284: Interconnect to the tim_etr input multiplexer* for TIM3

Triggering signals

The output from ADC4 is on signals adc_awd (three watchdogs on ADC4) and the input to timer on signal tim_etr.

Active power mode(s)

Run, Sleep, and an ADC4 conversion in autonomous mode in Stop 0 and Stop 1 can generate a wakeup interrupt and desired trigger action to timers.

16.3.4 Internal clock source to timer

From HSE32, HSI16, LSE, LSI and MCO to timer (TIM1/TIM2/TIM3/TIM16/TIM17) and low power timer (LPTIM1/LPTIM2).

Purpose

A timer input or clock can receive different clock sources and can be used, for example, to calibrate internal oscillators and a reference clock.

External clocks (HSE32, LSE), internal clocks (HSI16, LSI), microcontroller output clock (MCO) can be used as input to timer.

- HSI16 is assigned to timer (TIM1) as external trigger input signal (tim_etr4). HSI16 can be selected as counter clock provided by an external clock source in mode2: external trigger input. Inputs assignment and clock selection description are detailed in *Section 29.3.7: Clock selection*.
- HSI16 and LSE are assigned to timer TIM2/TIM3 as external input signals (tim_etr4/tim_etr11). HSI/LSE can be selected as counter clock provided by an external clock source in mode2: external trigger input. Inputs assignment and clock selection description are detailed in *Section 30.4.5: Clock selection*.
- HSE32, HSI16, LSE, and LSI are assigned to general purpose timers TIM16/TIM17 as input multiplexer input signal (tim_ti1_in3/tim_ti1_in5/tim_ti1_in6/tim_ti1_in9). HSI16/LSE/LSI can be selected as counter clock provided by an external clock source in mode1: input multiplexer input. Inputs assignment and clock selection description are detailed in *Section 31.3.6: Clock selection*.
- MCO is connected as external input to general-purpose timers TIM16/TIM17, making possible the calibration of the HSI16 system clock with LSE or LSI with HSE system clock. This feature is detailed in *Section 31.3.6: Clock selection*.
- LSE and LSI can be selected as input capture 2 (lptim_ic2_mux1/lptim_ic2_mux2) to LPTIM1. This feature is detailed in *Section 32.4.18: Input capture mode*.
- HSI16/256 can be selected as input capture 2 (lptim_ic2_mux1) to LPTIM2. This feature is detailed in *Section 32.4.18: Input capture mode*.

Triggering signals

The input to timer is on signals tim_etr or tim_ti1_in and for low power timer on signals lptim_ic2_mux.

The possible connections are given in:

- *Table 260: Interconnect to the tim_etr input multiplexer* for TIM1
- *Table 284: Interconnect to the tim_etr input multiplexer* for TIM2/TIM3
- *Table 279: Interconnect to the tim_ti1 input multiplexer* for TIM16/TIM17
- *Table 313: LPTIM1/2 input 2 connections* for LPTIM1/LPTIM2

Active power mode(s)

Run, Sleep.

16.3.5 Triggers to low-power timer

From RTC wake-up, RTC alarm, TAMP, GPDMA1 and LPTIM_ETR to low power timer (LPTIM1/LPTIM2).

Purpose

Low-power timer counters may be started after the detection of an active edge on a trigger input (lptim_ext_trig0 to 5). This feature is detailed in *Section 32.4.7: Trigger multiplexer*.

Triggering signals

The input to low power timer on signals lptim_ext_trig.

The possible connections are given in *Table 311: LPTIM1/2 external trigger connections*.

Active power mode(s)

Run, Sleep, Stop 0, Stop 1.

16.3.6 Internal analog signals to analog peripheral

From internal analog source to analog peripheral (ADC4).

Purpose

The internal reference voltage (V_{REFINT}), the internal temperature sensor (V_{SENSE}) and digital core voltage (V_{CORE}) monitoring signals are connected to analog peripheral (ADC4), as described in:

- *Section 21.4.9: Channel selection (CHSEL, SCANDIR, CHSELRMOD)*
- *Section 21.4.27: Temperature sensor and internal reference voltage*

Input signals

The input to analog peripheral on signals Vin.

The possible connections are given in *Table 150: ADC interconnection*.

Active power mode(s)

Run, Sleep, Stop 0, Stop 1.

16.3.7 System errors as break signals to timers

From system errors to timers (TIM1/TIM16/TIM17).

Purpose

HSE32 clock security, CPU lockup, SRAM2 parity error, FLASH ECC double error detection and PVD can generate system errors in the form of timer break toward timers (TIM1/TIM16/TIM17).

The purpose of the break function is to protect power switches driven by PWM signals generated by the timers. This feature is detailed in:

- *Section 29.3.18: Using the break function* for TIM1
- *Section 31.3.13: Using the break function* for TIM16/TIM17

Break signals

The input to timer on signals tim_sys_brk.

The possible connections are given in:

- *Table 263: System break interconnect* for TIM1
- *Table 299: System break interconnect* for TIM16/TIM17

Active power mode(s)

Run, Sleep.

16.3.8 Triggers to GPDMA1

From GPIO pin EXTI, RTC, TAMP, timers (TIM2), low-power timer (LPTIM1/LPTIM2), COMP1/COMP2, GPDMA1, ADC4 to GPDMA1.

Purpose

A GPDMA trigger can be assigned to a GPDMA channel x. A programmed 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 linked-list item transfer.

More details are given in:

- *Section 17.3.5: GPDMA triggers*
- *Section 17.4.12: GPDMA triggered transfer*

Triggering signals

Trigger mapping is specified in *Table 129: Programmed GPDMA1 trigger*.

Active power mode(s)

Run, Sleep, and for peripherals supporting autonomous mode also in Stop 0 and Stop 1.

16.3.9 Internal tamper sources

From LSE clock security, RTC, Debug, ADC4, AES/SAES, PKA, TRNG and IWDG to TAMP

Purpose

In order to detect any abnormal activity or tentative to corrupt the device, tampers are introduced and alert the system of such undesired events. Different actions can be taken in consequence. More details are given in *Section 37: Tamper and backup registers (TAMP)*.

Resources

List of tamper sources can be found in *Table 344: TAMP interconnection*.

Active power mode(s)

These interconnections are active in all power modes if the tamper source is active.

16.3.10 Triggers to communication peripherals

From LP timers (LPTIM1/LPTIM2), comparators (COMP1/COMP2), GPDMA1 transfer complete, EXTI GPIOs, RTC alarm and RTC wakeup to I2C1, I2C3, USART1, USART2, LPUART1, SPI1 and SPI3.

Purpose

LP timer (LPTIM1) output channels (lptim1_ch1 and lptim2_ch1), comparator (COMP1, COMP2) output channels (comp1_out and comp2_out), EXTI GPIOs, RTC alarm and RTC wakeup, can be used as trigger to start a communication on the selected I2C, USART, LPUART and SPI peripheral.

A GPDMA1 transfer complete can trigger both the GPDMA1 regular or linked-list new transfers and communication on selected communication peripheral.

These features are detailed in:

- *Section 38.4.16: Autonomous mode* I2C
- *Section 39.5.22: USART autonomous mode* USART
- *Section 40.4.15: LPUART autonomous mode* LPUART
- *Section 41.4.15: Autonomous mode* SPI

Triggering signals

The outputs from triggers are directly connected to peripheral trigger inputs.

The selection of input triggers is detailed in:

- *Table 357: I2C1 interconnection* and *Table 358: I2C3 interconnection*
- *Table 377: USART interconnection (USART1/2)*
- *Table 389: LPUART interconnections (LPUART1)*
- *Table 399: SPI interconnection (SPI1)* and *Table 400: 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 39.6: USART in low-power modes*
- *Section 38.5: I2C in low-power modes*
- *Section 41.6: SPI in low-power modes*

16.3.11 Output from tamper

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 RTC_OUT and send tamp status tamp_evt outside the MCU. More details are given in *Section 36.3.3: GPIOs controlled by the RTC and TAMP*.

Active power mode

This interconnection remain active in all power modes.

16.3.12 Timers generating IRTIM signal

From timer (TIM16/TIM17) to IRTIM.

Purpose

Timers (TIM16/TIM17) output channels timx_oc1 are used to generate the waveform of infrared signal output. The functionality is detailed in *Section 33: Infrared interface (IRTIM)*.

Active power mode(s)

Run, Sleep.

16.3.13 From encryption keys to AES/SAES

From TAMP backup registers, system Flash memory to and in between SAES and AES.

Purpose

The encryption mechanism requires a 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 26.4.14: 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 26.4.15: SAES operation with shared keys* for more details.

Active power mode

AES and SAES are operational in Run and Sleep modes.

16.3.14 From timer (TIM1/TIM2/TIM3) to comparators (COMP1/COMP2)

Purpose

Advanced-control timer (TIM1) and general-purpose timer (TIM2/TIM3) can be used as blanking window input to COMP1/COMP2.

The blanking function is described in *Section 22.3.6: Comparator output-blanking function*.

The blanking sources are given in the following registers:

- *COMP1 control and status register (COMP1_CSR)* BLANKING
- *COMP2 control and status register (COMP2_CSR)* BLANKING

Triggering signals

Timer output signals TIMx_OCx are the inputs to blanking source of COMP1/COMP2.

Active power mode(s)

Run, Sleep.

16.3.15 From comparators (COMP1/COMP2) to timers

From RTC wakeup, RTC alarm and TAMP to timers (TIM1/TIM2/TIM3/TIM16/TIM17) and low power timers (LPTIM1/LPTIM2).

Purpose

Comparators (COMP1/COMP2) output values can be connected to timers TIM1/TIM2/TIM3/TIM16/TIM17 input captures or TIMx_ETR signals.

Comparators (COMP1/COMP2) output values can also generate break input signals for timer TIM1 on input pins TIMx_BKIN or TIMx_BKIN2 through GPIO alternate function selection using open drain connection of I/Os.

Comparators (COMP1/COMP2) output values can be connected to low-power timers LPTIM1/LPTIM2 input, input capture and external trigger signals.

The possible connections are given in:

- *Section 29.3.2: TIM1 pins and internal signals*
- *Section 30.4.2: TIM2/TIM3 pins and internal signals*
- *Section 31.3.2: TIM16/TIM17 pins and internal signals*
- *Section 32.4.2: LPTIM pins and internal signals*

Active power mode(s)

Run, Sleep.

17 General purpose direct memory access controller (GPDMA)

17.1 GPDMA 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 Stop mode (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
- 8 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
- 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
- Channel abort and restart

- Debug:
  - Channel suspend and resume support
  - Channel status reporting, including FIFO level, and event flags
- TrustZone support:
  - Support for secure and nonsecure GPDMA transfers, independently at a first channel level, and independently at a source/destination and link sublevels
  - Secure and nonsecure interrupts reporting, resulting from any of the respectively secure and nonsecure channels
  - TrustZone-aware AHB slave port, protecting any GPDMA secure resource (register, register field) from a nonsecure 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

Table 124. GPDMA1 implementation

Feature	GPDMA1
Number of channels	8
Number of requests	52
Number of triggers	30
Block request	X
TrustZone security	X

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 125. GPDMA1 channel implementation

Channel x	Hardware parameters		Features
	dma_fifo_ size[x]	dma_ addressing[x]	
x = 0 to 5	2	0	Channel x (x = 0 to 3) 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 = 6 to 7	4	0	Channel x (x = 4 to 7) is implemented with: – a FIFO of 32 bytes, 8 words – fixed/contiguously incremented addressing These channels may be 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 wake-up features are implemented in the device low-power modes as per the table below.

Table 126. GPDMA1 autonomous mode and wake-up in low-power modes

Feature	Low-power modes
Autonomous mode and wake-up	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 REQSEL[5:0] in GPDMA_CxTR2, provided that SWREQ = 0.

The GPDMA requests mapping is specified in the table below.

Table 127. Programmed GPDMA1 request

GPDMA_CxTR2.REQSEL[5:0]	Selected GPDMA request
0	adc4_dma
1	spi1_rx_dma(1)
2	spi1_tx_dma(1)
3	spi3_rx_dma
4	spi3_tx_dma
5	i2c1_rx_dma(1)
6	i2c1_tx_dma(1)
7	i2c1_evc_dma(1)
8	i2c3_rx_dma

Table 127. Programmed GPDMA1 request (continued)

GPDMA_CxTR2.REQSEL[5:0]	Selected GPDMA request
9	i2c3_tx_dma
10	i2c3_evc_dma
11	usart1_rx_dma
12	usart1_tx_dma
13	usart2_rx_dma(1)
14	usart2_tx_dma(1)
15	lpuart1_rx_dma
16	lpuart1_tx_dma
17	sai_a_dma(2)
18	sai_b_dma(2)
19	tim1_cc1_dma
20	tim1_cc2_dma
21	tim1_cc3_dma
22	tim1_cc4_dma
23	tim1_upd_dma
24	tim1_trg_dma
25	tim1_com_dma
26	tim2_cc1_dma
27	tim2_cc2_dma
28	tim2_cc3_dma
29	tim2_cc4_dma
30	tim2_upd_dma
31	tim3_cc1_dma(1)
32	tim3_cc2_dma(1)
33	tim3_cc3_dma(1)
34	tim3_cc4_dma(1)
35	tim3_upd_dma(1)
36	tim3_trg_dma(1)
37	tim16_cc1_dma
38	tim16_upd_dma
39	tim17_cc1_dma(1)
40	tim17_upd_dma(1)
41	aes_in_dma
42	aes_out_dma
43	hash_in_dma

Table 127. Programmed GPDMA1 request (continued)

GPDMA_CxTR2.REQSEL[5:0]	Selected GPDMA request
44	saes_in_dma(1)
45	saes_out_dma(1)
46	lptim1_ic1_dma
47	lptim1_ic2_dma
48	lptim1_ue_dma
49	lptim2_ic1_dma(1)
50	lptim2_ic2_dma(1)
51	lptim2_ue_dma(1)

1. Only available on STM32WBA52/54/55xx devices.
2. Only available on STM32WBA54/55xx devices.

17.3.4 GPDMA block requests

Some GPDMA requests must be programmed as a block request, and not as a burst request. Then BREQ in GPDMA_CxTR2 must be set for a correct GPDMA execution of the requested peripheral transfer at the hardware level.

Table 128. Programmed GPDMA1 request as a block request

GPDMA block requests
lptim1_ue_dma
lptim2_ue_dma(1)

1. Only available on STM32WBA52/54/55xx devices.

17.3.5 GPDMA triggers

A GPDMA trigger can be assigned to a GPDMA channel x, via TRIGSEL[4:0] in 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 129.Programmed GPDMA1 trigger

GPDMA_CxTR2.TRIGSEL[4:0]	Selected GPDMA trigger
0	exti0
1	exti1
2	exti2
3	exti3
4	exti4
5	exti5
6	exti6

Table 129.Programmed GPDMA1 trigger (continued)

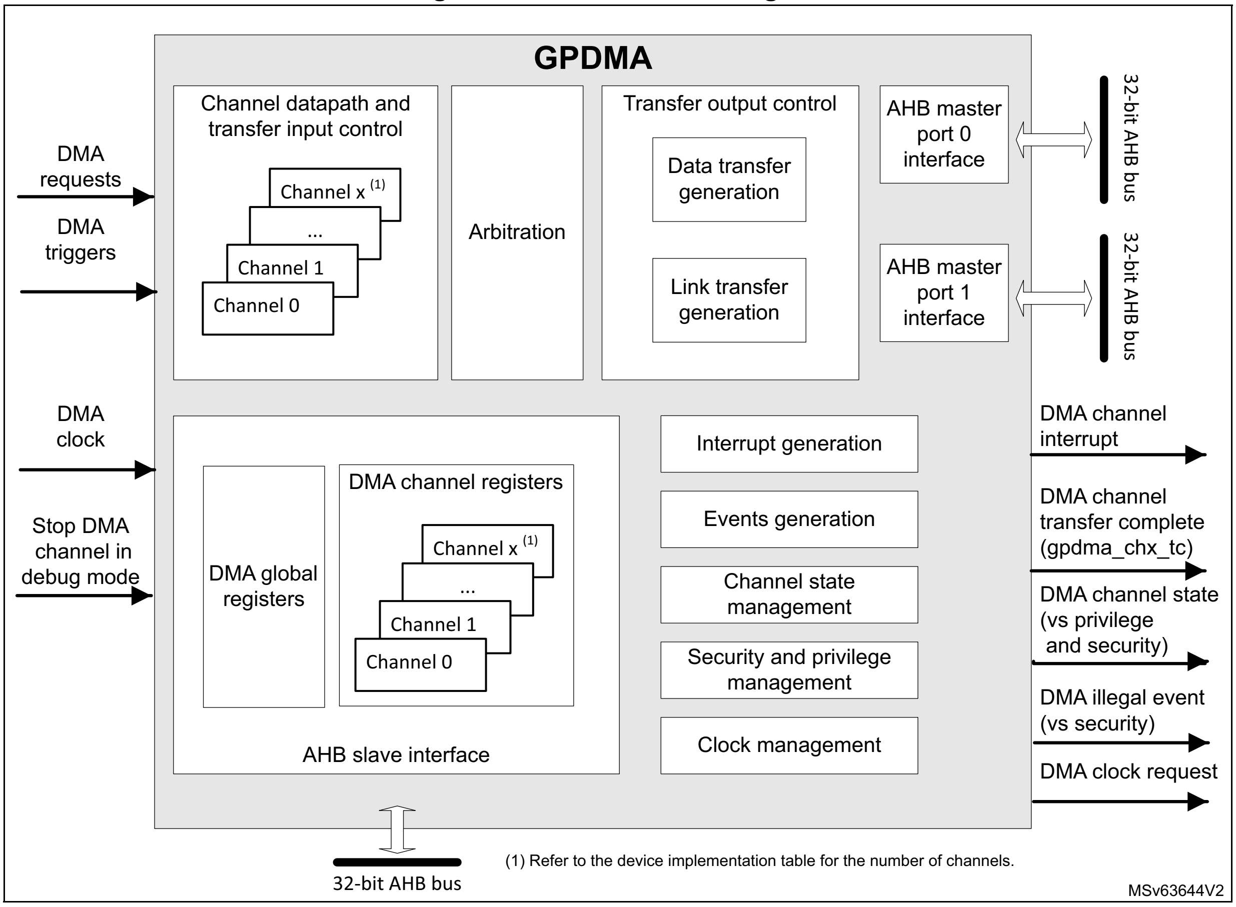
GPDMA_CxTR2.TRIGSEL[4:0]	Selected GPDMA trigger
7	exti7
8	tamp_trg1
9	tamp_trg2
10	tamp_trg3
11	lptim1_ch1
12	lptim1_ch2
13	lptim2_ch1(1)
14	lptim2_ch2(1)
15	comp1_out(2)
16	comp2_out(2)
17	rtc_alra_trg
18	rtc_alrb_trg
19	rtc_wut_trg
20	gpdma1_ch0_tc
21	gpdma1_ch1_tc
22	gpdma1_ch2_tc
23	gpdma1_ch3_tc
24	gpdma1_ch4_tc
25	gpdma1_ch5_tc
26	gpdma1_ch6_tc
27	gpdma1_ch7_tc
28	tim2_trgo
29	adc4_awd1

1. Only available on STM32WBA52/54/55xx devices.
2. Only available on STM32WBA54xx and STM32WBA55xx devices.

17.4 GPDMA functional description

17.4.1 GPDMA block diagram

Figure 49. GPDMA block diagram



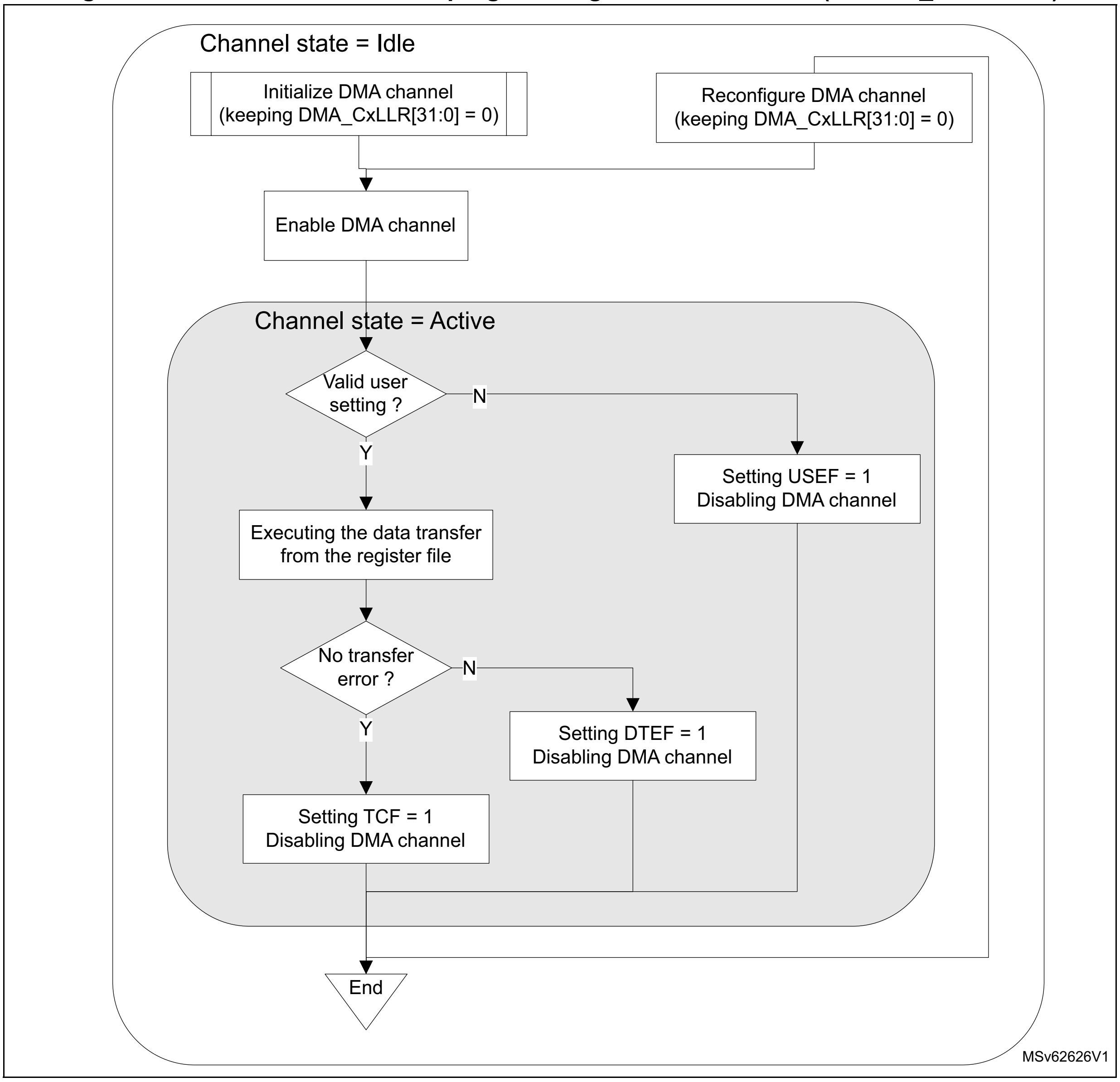
17.4.2 GPDMA channel state and direct programming without any linked-list

After a GPDMA reset, a GPDMA channel x is in idle state. When the software writes 1 in GPDMA_CxCR.EN, 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 in GPDMA_CxCR.EN, 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.

Figure 50 illustrates this GPDMA direct programming without any linked-list (GPDMA_CxLLR = 0).

Figure 50. GPDMA channel direct programming without linked-list (GPDMA_CxLLR = 0)



17.4.3 GPDMA channel suspend and resume

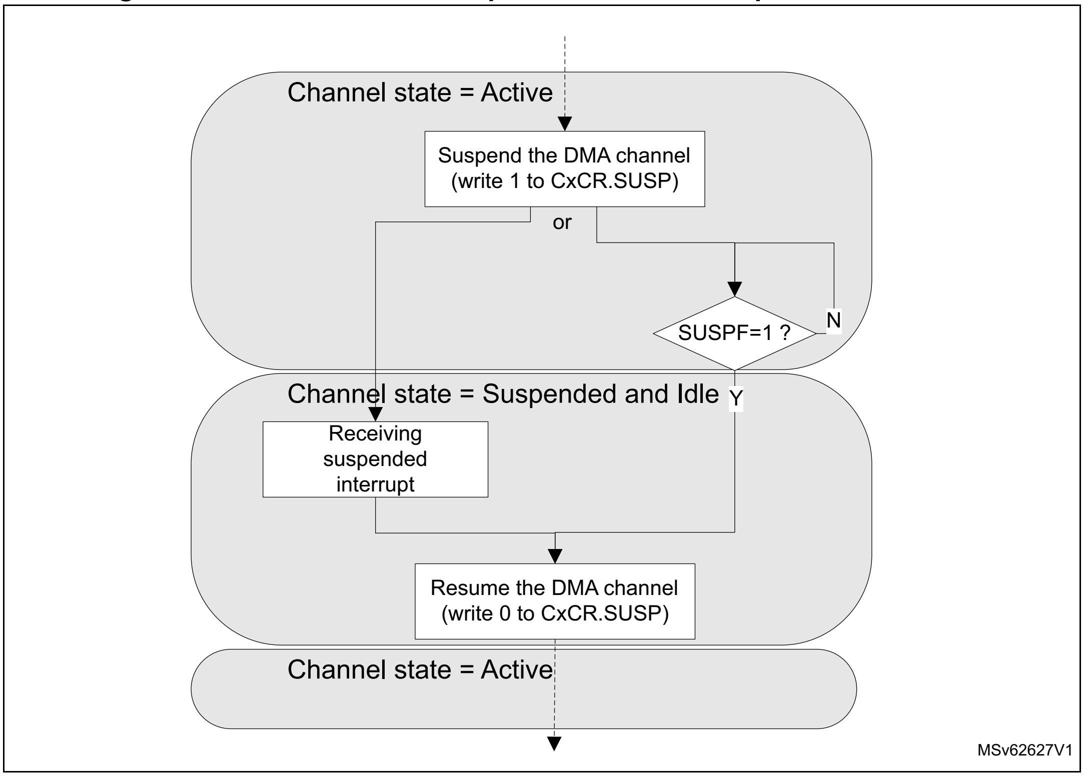
The software can suspend on its own a channel still active, with the following sequence:

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: 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 51. GPDMA channel suspend and resume sequence



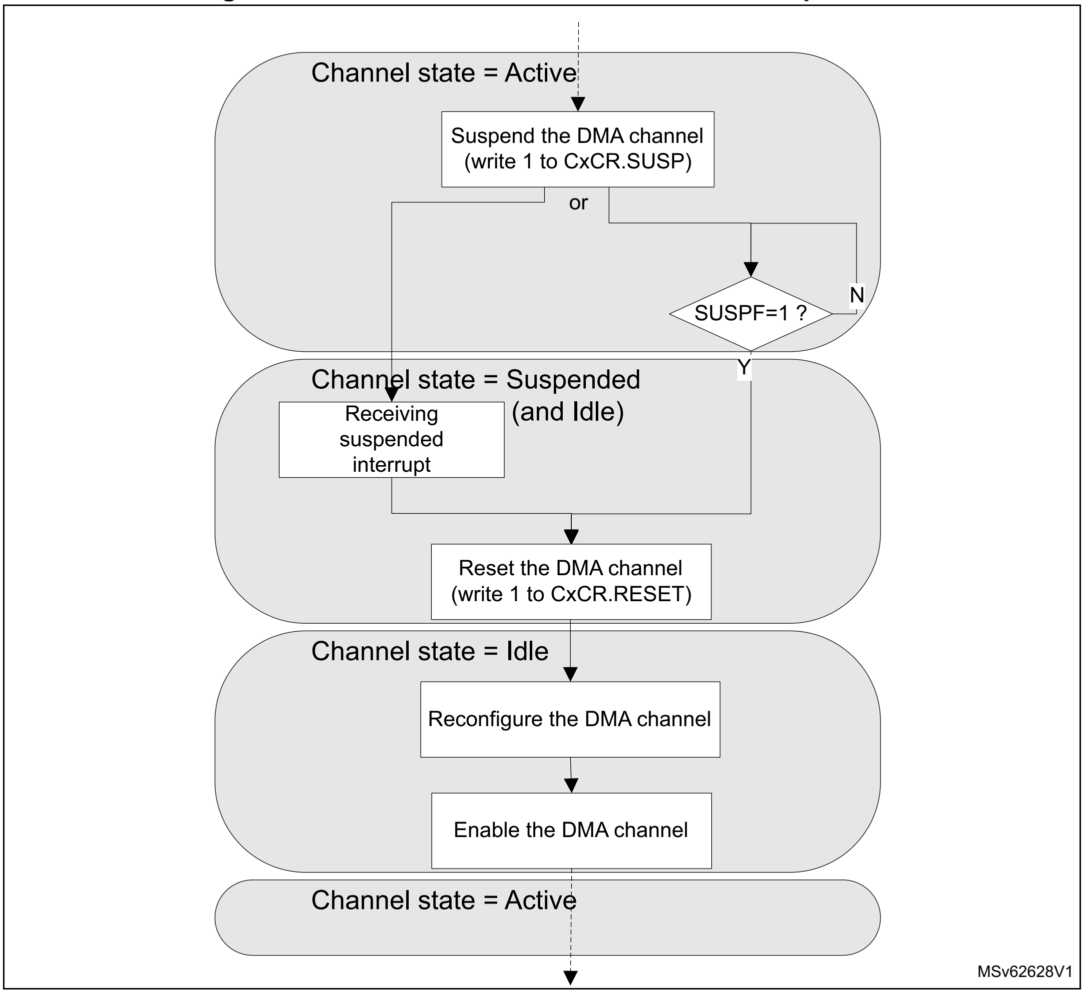
Note: A suspend and resume sequence does not impact the GPDMA_CxCR.EN bit. Suspending a channel (transfer) does not suspend a started trigger detection.

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 52. 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 in GPDMA_CxLLR.

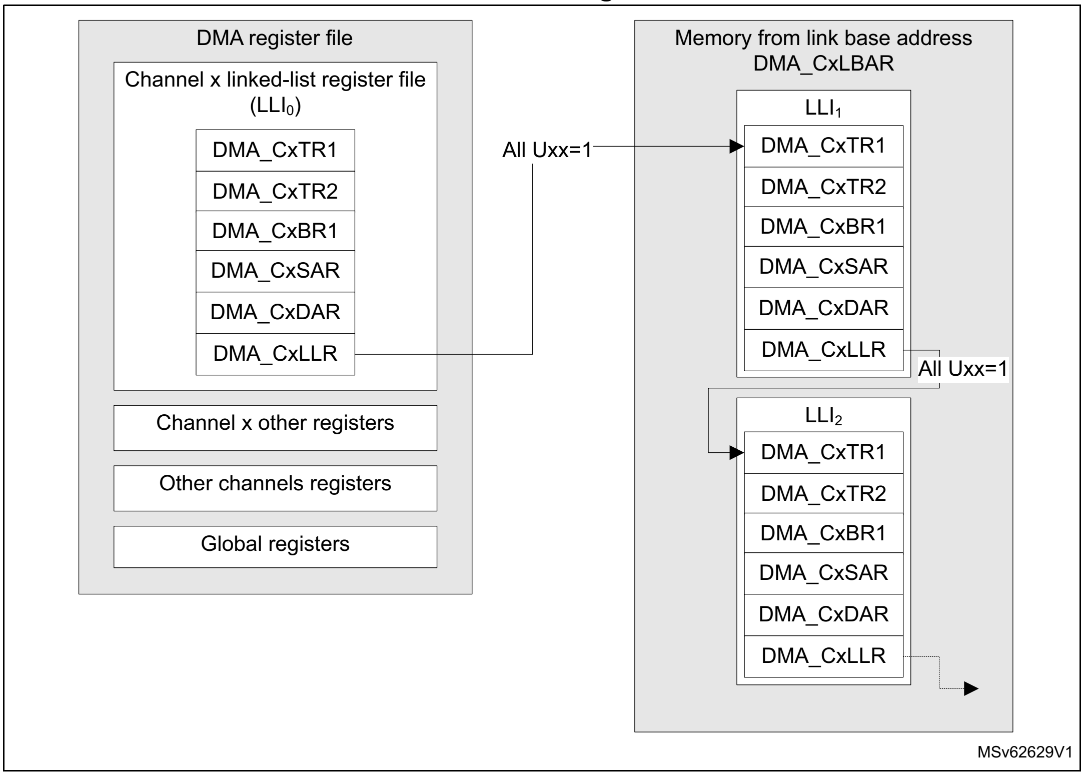
In linked-list mode, each GPDMA linked-list register (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR or GPDMA_CxLLR) is conditionally and automatically updated from the next linked-list data structure in the memory, following the current value of the GPDMA_CxLLR register that was conditionally 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) in GPDMA_CxLLR are all asserted, the linked-list data structure in the memory is maximal with:

- channel x (x = 0 to 7) contiguous 32-bit locations, including GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR and GPDMA_CxLLR (see *Figure 53*) and including the first linked-list register file (LLI_0) and the next LLIs (such as LLI_1, LLI_2) in the memory

Figure 53. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x

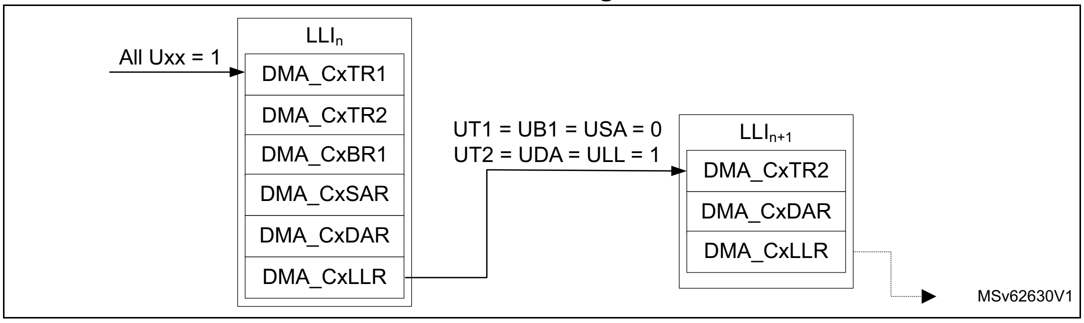


Dynamic linked-list data structure

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 54*).

Figure 54. GPDMA dynamic linked-list data structure of a linear 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
  - 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 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).

The channel may never be completed when GPDMA_CxLLR.LSM = 0:

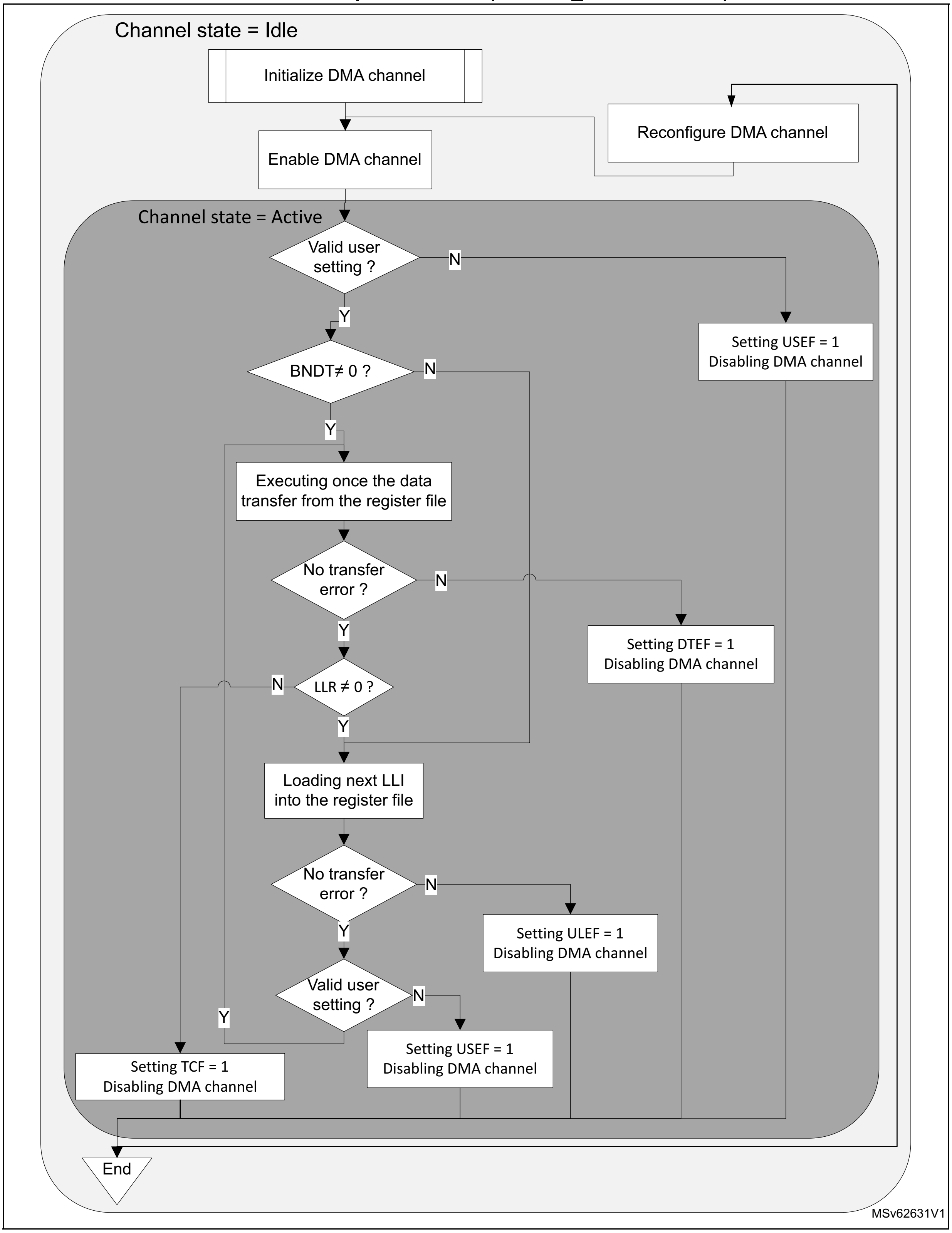
- If the last LLI_N is recursive, pointing to itself as a next LLI:
  - either GPDMA_CxLLR.ULL = 1 and GPDMA_CxLLR.LA[15:2] is updated by the same value
  - or GPDMA_CxLLR.ULL = 0
- If LLI_N is pointing to a previous LLI

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.

Figure 55 depicts the GPDMA channel execution and its registers programming in run-to-completion mode.

Note: Figure 55 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at channel completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion, the hardware always set TCEF = 1 and disables the channel.

Figure 55. 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}:

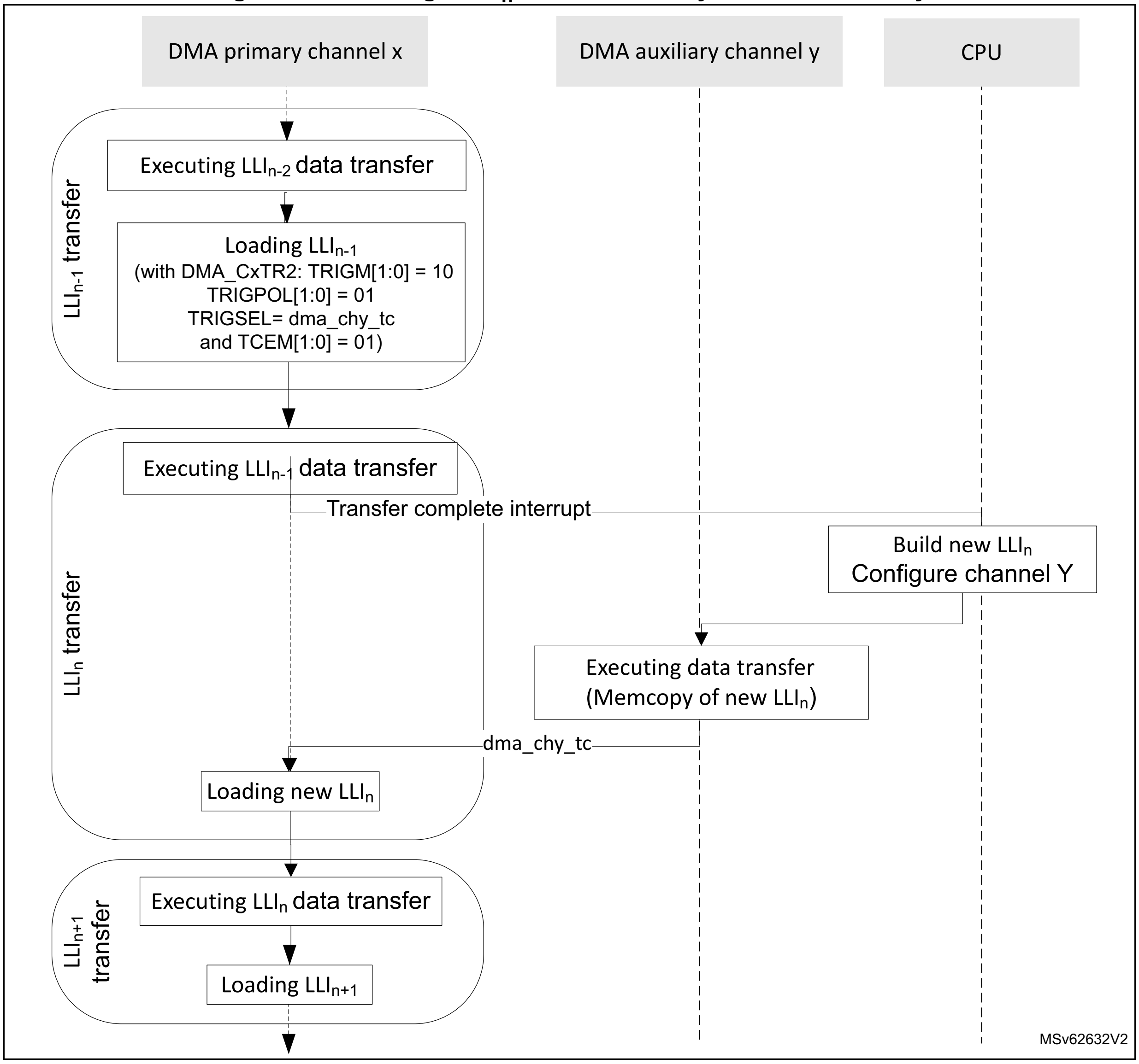
- TRIGM[1:0] = 10 (link transfer triggering mode)
- TRIGPOL[1:0] = 01 or 10 (rising or falling edge)
- TRIGSEL[4:0] (see *Section 17.3.5* for the trigger selection details)

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.

Figure 56 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 56. 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]).
  - 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_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 (whenGPDMA_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

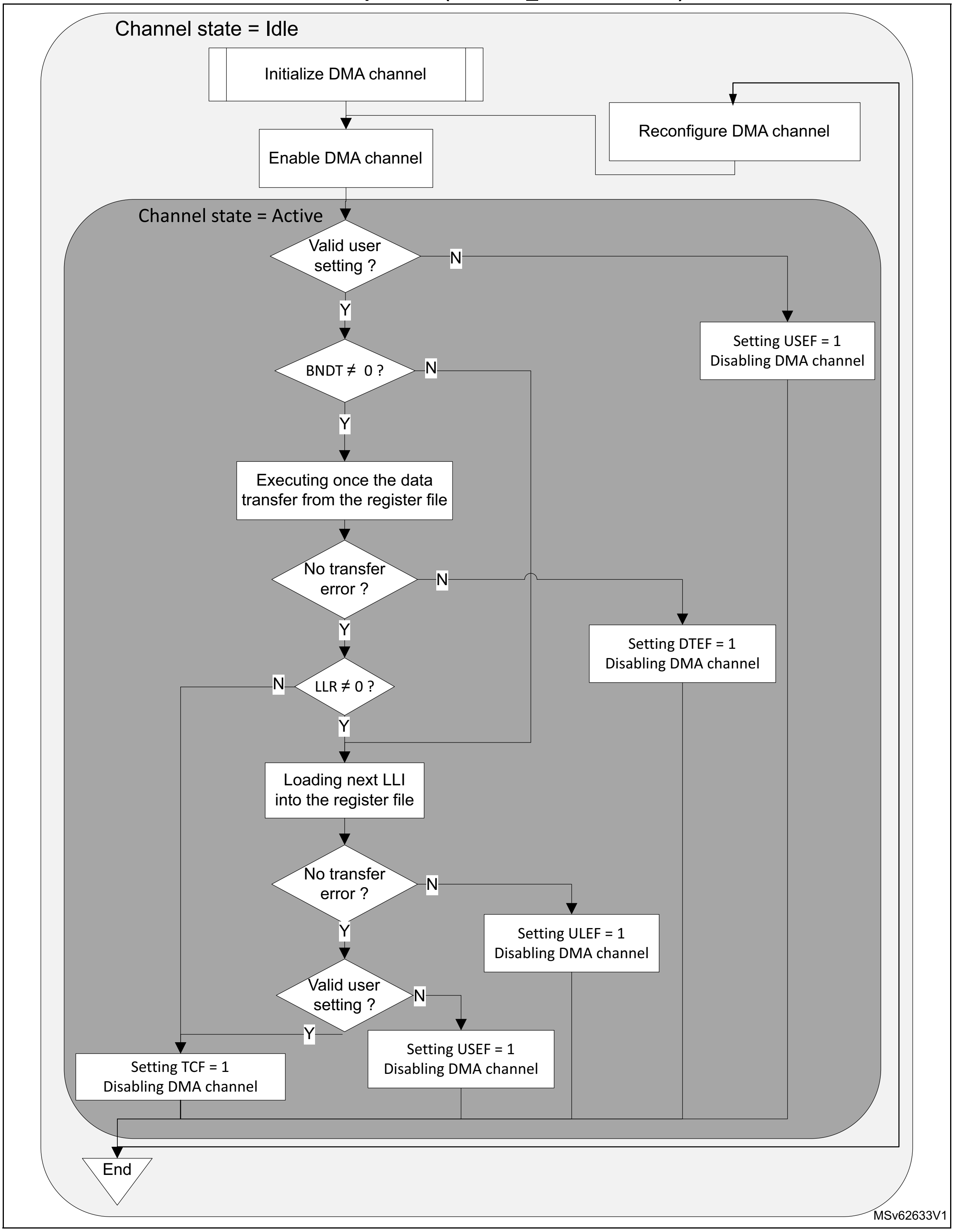
Note: If a LLI is recursive (pointing to itself as a next LLI, either GPDMA_CxLLR.ULL = 1 and GPDMA_CxLLR.LA[15:2] is updated by the same value, or GPDMA_CxLLR.ULL = 0), a channel in link step mode is completed after each repeated single execution of this LLI.

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).

Figure 57 depicts the GPDMA channel execution mode, and its programming in link step mode.

Note: Figure 57 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

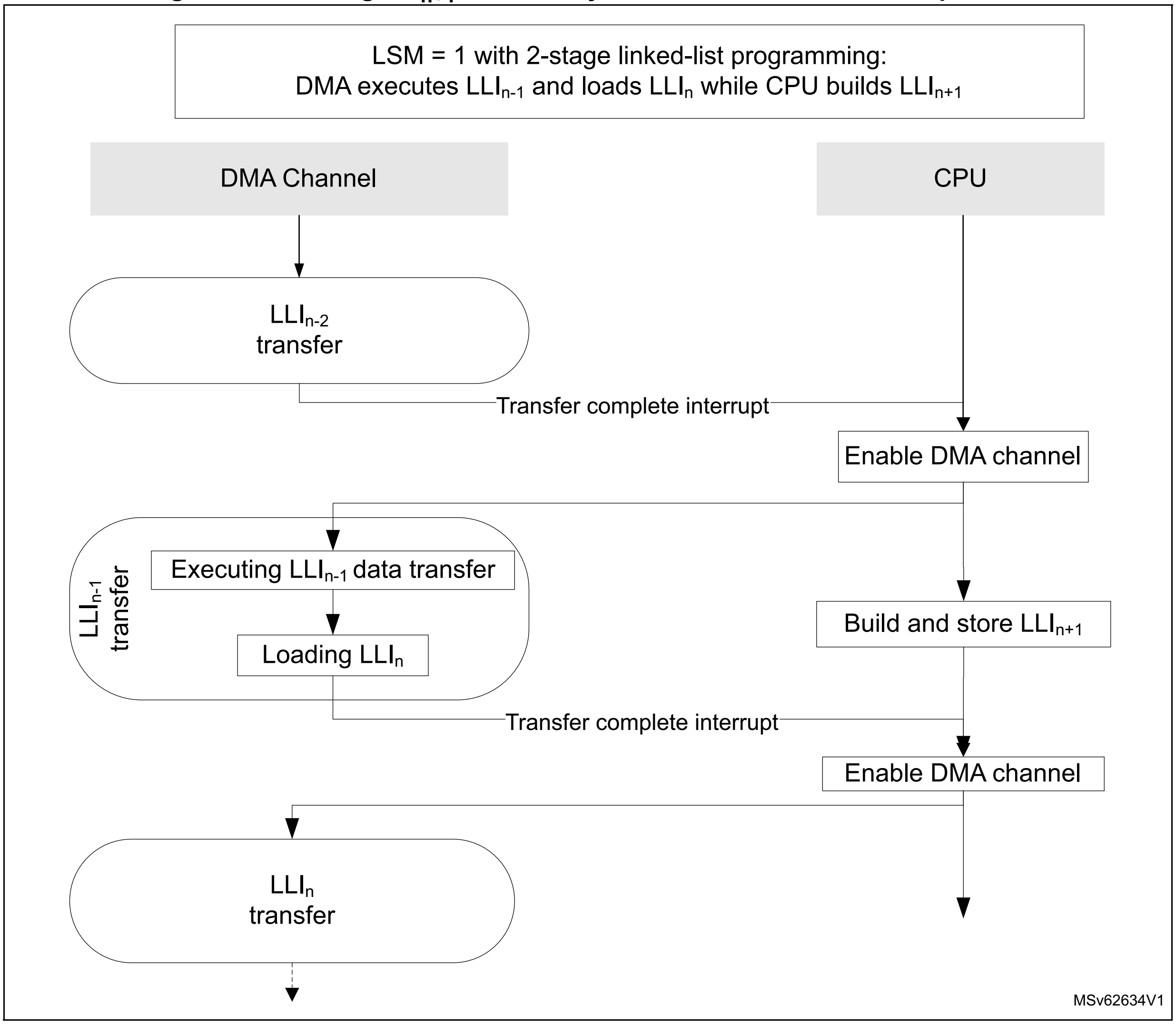
Figure 57. GPDMA channel execution and linked-list programming in link step mode (GPDMA_CxCR.LSM = 1)



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 *Figure 58*.

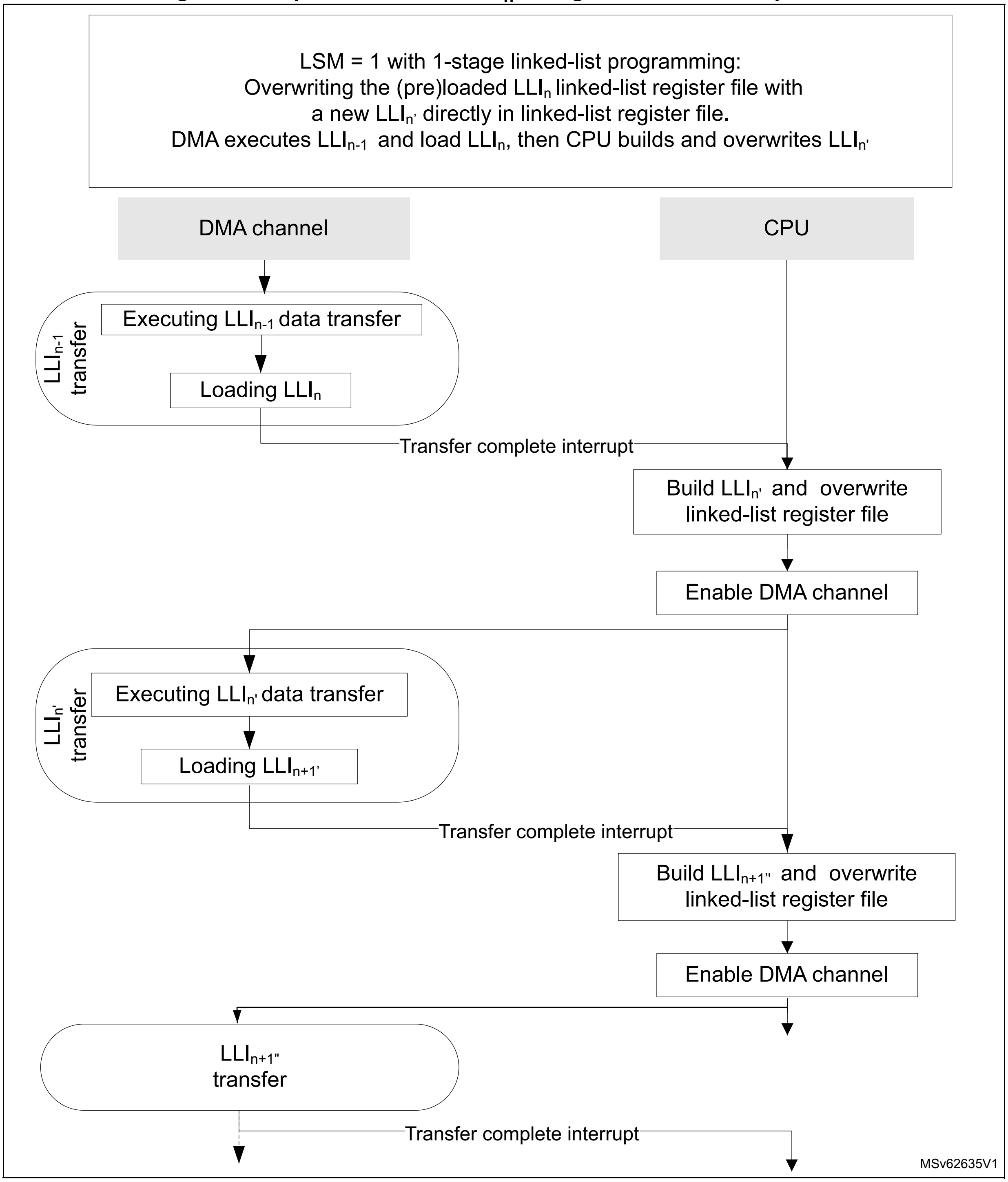
Figure 58. Building LLI_{n+1}: GPDMA dynamic linked-lists in link step mode



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 *Figure 59*.

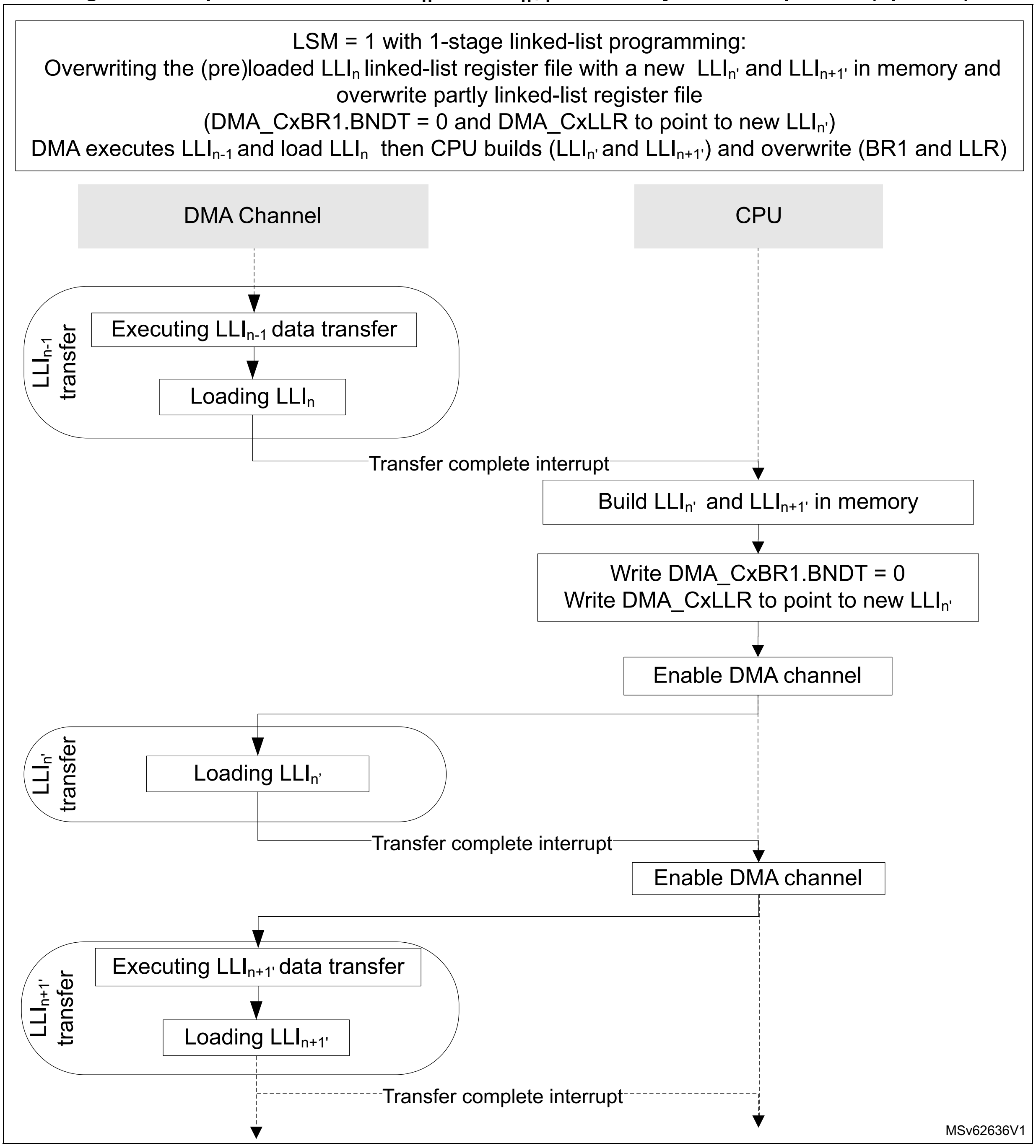
Figure 59. Replace with a new $LLI_{n'}$ in register file in link step mode



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 *Figure 60*.

Figure 60. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 1)



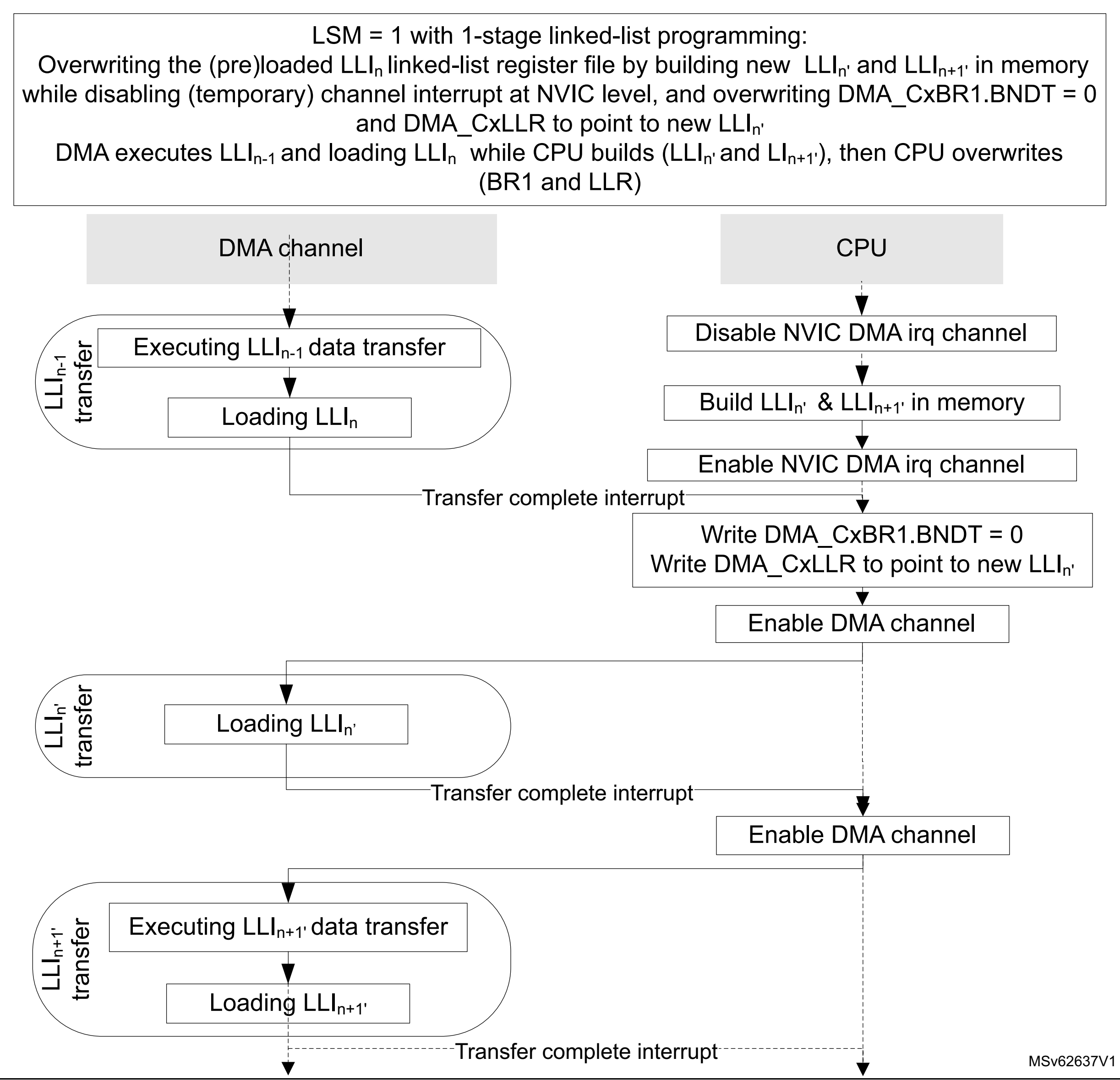
Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode

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 *Figure 61*.

Figure 61. 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:

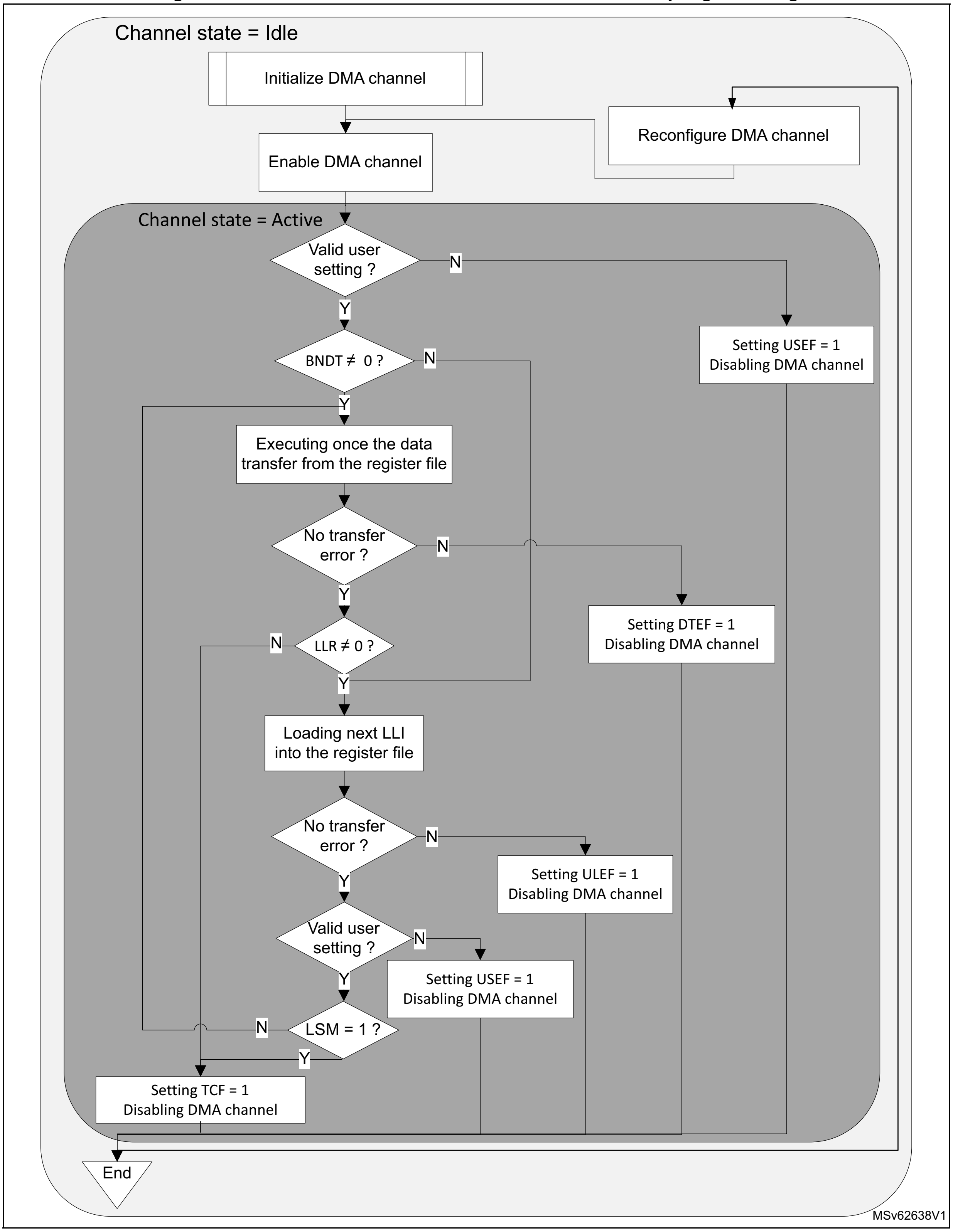
- reprogram LLI_{n+1} in the memory to finally complete the channel by this LLI_{n+1} (clear the GPDMA_CxLLR of this LLI_{n+1}), before that this LLI_{n+1} is loaded/used by the GPDMA channel
- abort and reconfigure the channel with a LSM update (see *Section 17.4.4*.)

In link step mode, the software can clear LSM after each a single execution of any LLI, during LLI_{n-1}.

Figure 62 shows the overall and unified GPDMA linked-list programming, whatever is the execution mode.

Note: Figure 62 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion the hardware always set TCEF = 1 and disables the channel. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

Figure 62. 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 130. 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 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[1:0] and DDW_LOG2[1:0] fields (see *Table 131*).

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 131. 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 131. 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 131. 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[5:0] 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[5:0] 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). 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 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:

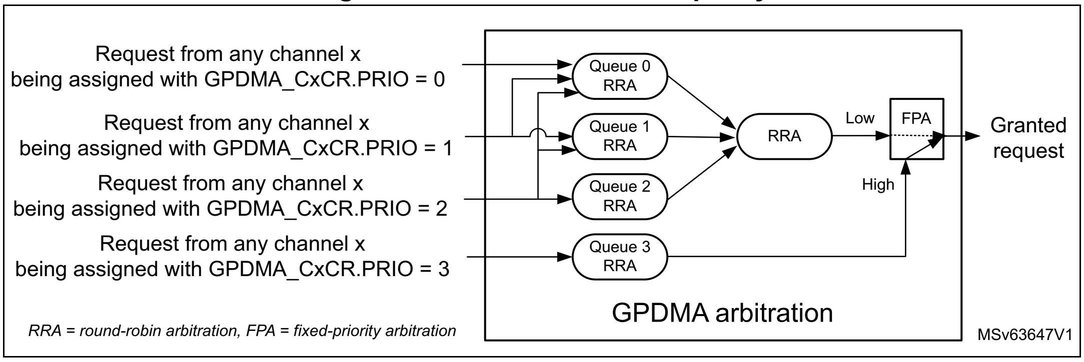
- one high-priority traffic class (queue 3), dedicated to the assigned channels with priority 3, for time-sensitive channels

  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}).
  $B_{req0} = B_{P0} = 1/2 * 1/2 * (1 - B_{Q3}) = 1/4 * (1 - B_{Q3})$
  $B_{req1} = B_{P1} = (1/2 + 1) * 1/2 * (1 - B_{Q3}) = 3/4 * (1 - B_{Q3})$
  – 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[4:0] (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 or each programmed single read. The trigger mode can also be programmed to condition the LLI link transfer (see TRIGM[1:0] in 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).

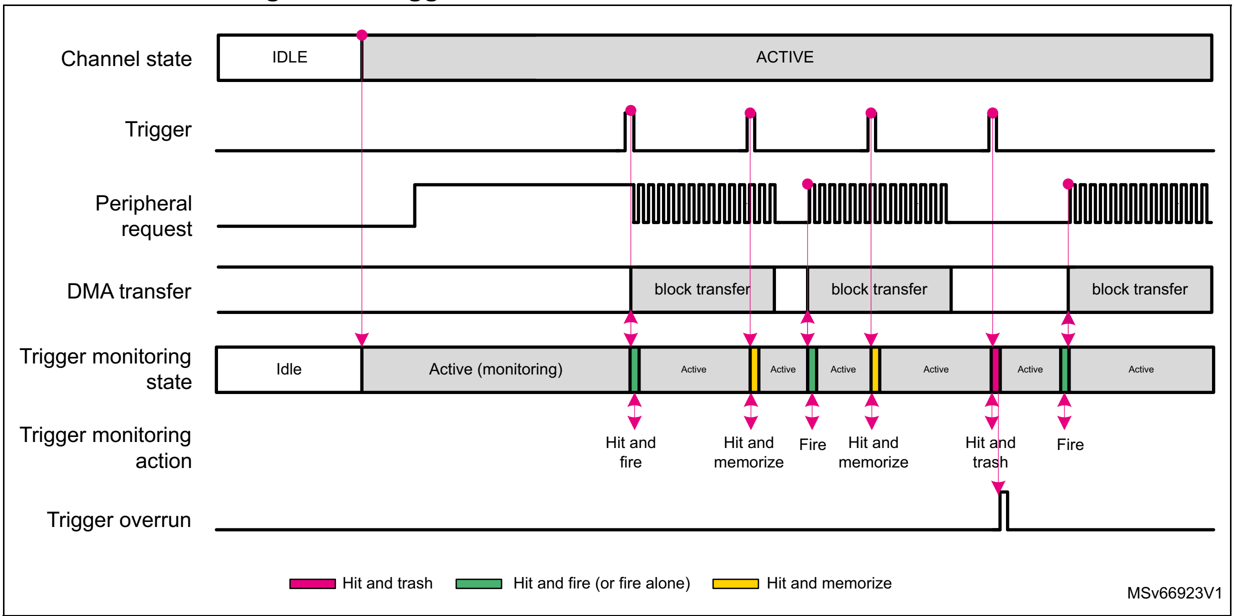
The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer. If a new trigger is detected, this hit is internally memorized to grant the next transfer, as long as the defined rising/falling edge and TRIGSEL[4:0] are not modified, and the channel is enabled.

Transferring a next $LLI_{n+1,}$ that updates the GPDMA_CxTR2 with a new value for any of TRIGSEL[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 (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.

Figure 64 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

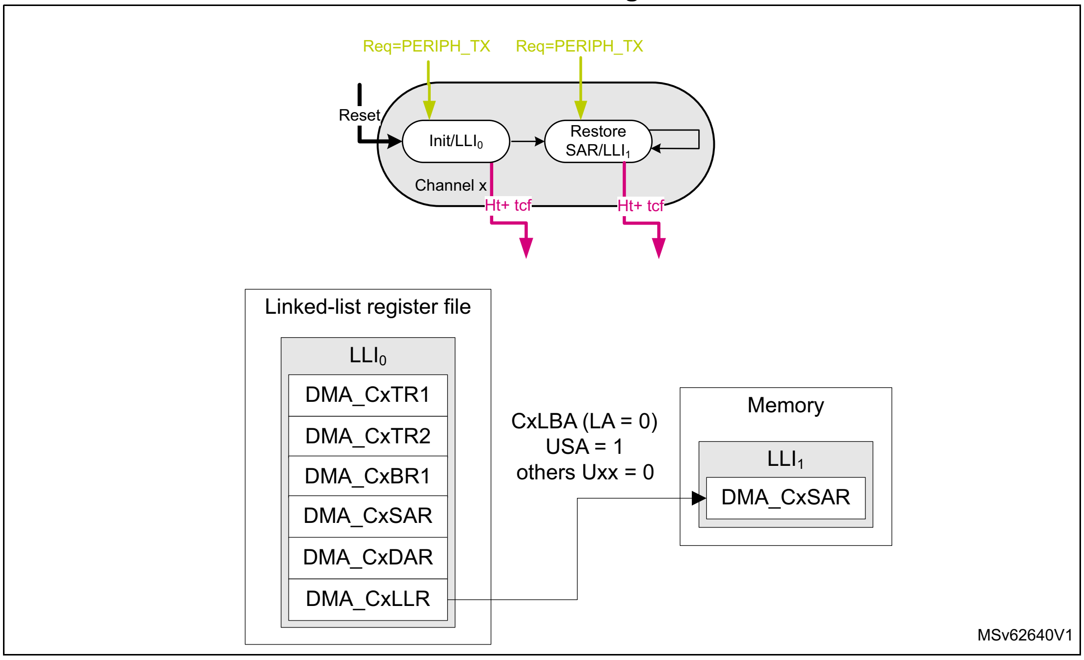
GPDMA circular buffering for memory-to-peripheral and peripheral-to-memory transfers, with a linear addressing channel

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.

Figure 65 illustrates this programming with a linear addressing GPDMA channel and a source circular buffer.

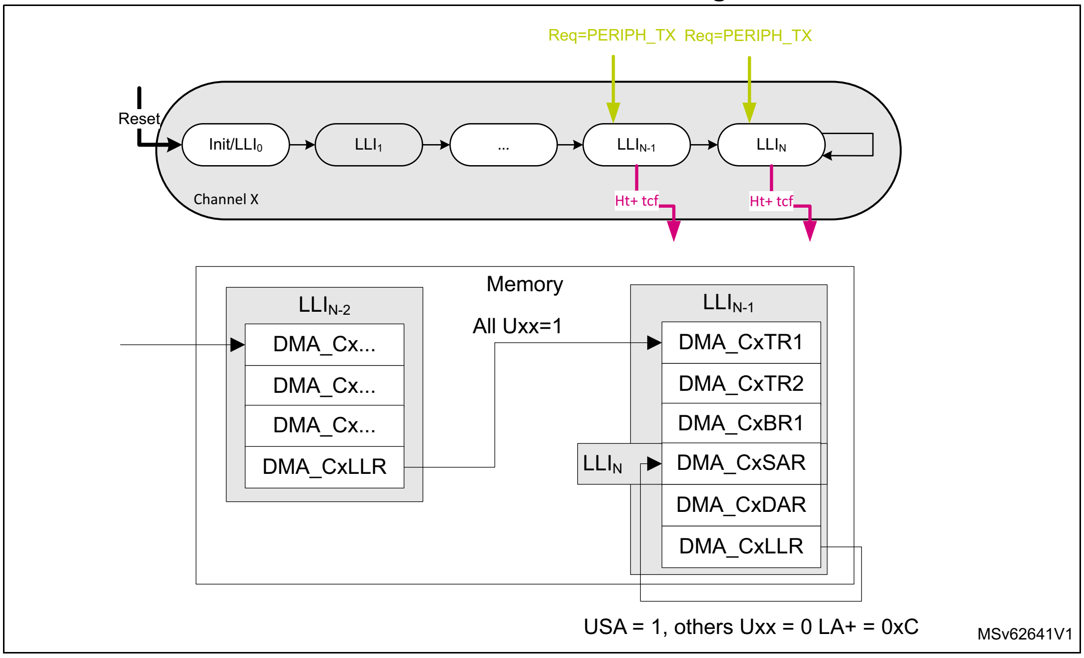
Figure 65. GPDMA circular buffer programming: update of the memory start address with a linear addressing channel



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).

Figure 66 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/nonsecure 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 nonsecure worlds at any given time.

Any channel x is a secure or a nonsecure hardware resource, as 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 nonsecure 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 nonsecure agent.
- A nonsecure 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 nonsecure 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/nonsecure state of each GPDMA channel x.

The GPDMA also generates a security illegal access pulse signal on an illegal nonsecure 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 nonsecure 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 nonsecure transfer as a nonsecure software is not allowed to do so and must have GPDMA_CxCR.EN = 0 before the nonsecure 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), 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])
  - 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])

– 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*.

Note: Design/firmware: In low-power modes, DMA clock request is asserted upon a DMA request/trigger from an autonomous peripheral or upon a DMA trigger from an EXTI line.

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[y[a]:0], SWREQ, and REQSEL[z[a]: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 wake-up 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.

a. Refer to *GPDMA channel x transfer register 2 (GPDMA_CxTR2)* for y and z values.

- 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 132. 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 0 and 1 modes(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 133. 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, 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.

  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.

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 and privileged. A read access is secure or nonsecure, privileged or unprivileged.

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 (when GPDMA_CxCR.EN = 0), to securely allocate individually any channel x to the secure or nonsecure 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.	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **SECx**: secure state of channel x (x = 7 to 0)
0: Nonsecure
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 nonsecure.

This register can mix secure and nonsecure 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**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **PRIVx**: privileged state of channel x (x = 7 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, for example 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, for example at boot time).

A read access may be privileged or unprivileged, secure or nonsecure.

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.	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
								rs	rs	rs	rs	rs	rs	rs	rs

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

Bits 7:0 **LOCKx**: lock the configuration of GPDMA_SECCFGR.SECx and GPDMA_PRIVCFGR.PRIVx, until a global GPDMA reset (x = 7 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 nonsecure masked interrupt status register (GPDMA_MISR)

Address offset: 0x0C

Reset value: 0x0000 0000

This register is a read register.

This is a nonsecure register, containing the masked interrupt status bit MISx for each nonsecure 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 nonsecure interrupt status. An unprivileged software is restricted to read the status of unprivileged (and nonsecure) 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.	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
								r	r	r	r	r	r	r	r

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

Bits 7:0 **MISx**: masked interrupt status of channel x (x = 7 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 nonsecure 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
								r	r	r	r	r	r	r	r

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

Bits 7:0 **MISx**: masked interrupt status of the secure channel x (x = 7 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 7)

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 nonsecure 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 7)

Reset value: 0x0000 0000

This is a write register, secure or nonsecure 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 7)

Reset value: 0x0000 0001

This is a read register, reporting the channel status.

This register is secure or nonsecure, 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 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.

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, 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 7)

Reset value: 0x0000 0000

This register is secure or nonsecure 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). Then GPDMA_CxBR1.BNDT[15:0] = 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. 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 51*.

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 52*).

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 7)

Reset value: 0x0000 0000

This register is secure or nonsecure 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: 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 nonsecure

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 nonsecure), and the GPDMA transfer to the destination is nonsecure.

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: A destination burst transfer must have an aligned address with its data width (start address GPDMA_CxDAR[2:0] and if present address offset GPDMA_CxTR3.DAO[2:0], versus DDW_LOG2[1:0]). Otherwise a user setting error is reported and no transfer is issued.

When configured in packing mode (PAM[1] = 1 and destination data width different from source data width), a source block size must be a multiple of the destination data width (see GPDMA_CxBR1.BNDT[2:0] versus DDW_LOG2[1:0]). Else a user setting error is reported and none transfer is issued.

Setting a 8-byte data width causes a user setting error to be reported and none 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 nonsecure

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 nonsecure), and the GPDMA transfer from the source is nonsecure.

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 than source data width, and the condition PAM_2 is when source data width is higher than destination 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: 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.

Setting a 8-byte data width causes a user setting error to be reported and no transfer is issued.

17.8.11 GPDMA channel x transfer register 2 (GPDMA_CxTR2)

Address offset: 0x94 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure 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 (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.	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	DREQ	SWREQ	Res.	Res.	Res.	REQSEL[5:0]					
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 (x = 0 to 7), same as 00

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.

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 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). 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 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 .

01: channel x (x = 0 to 7), same as 0010: 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 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[4:0] 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[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. 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[5:0] is taken into account.

1: software request for a memory-to-memory transfer. The default selected hardware request as per REQSEL[5:0] is ignored.

Bits 8:6 Reserved, must be kept at reset value.

Bits 5:0 **REQSEL[5:0]**: 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[5:0] 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 7)

Reset value: 0x0000 0000

This register is secure or nonsecure 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 source address register (GPDMA_CxSAR)

Address offset: 0x9C + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure 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: 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[1:0]

– the additional source incremented/decremented offset value as programmed by GPDMA_CxBR1.SDEC

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.

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.

17.8.14 GPDMA channel x destination address register (GPDMA_CxDAR)

Address offset: 0xA0 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure 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: 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[1:0]) 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.

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.

17.8.15 GPDMA channel x linked-list address register (GPDMA_CxLLR)

Address offset: 0xCC + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure 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[15:2] = 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.

17.8.16 GPDMA register map

Table 134. GPDMA 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	GPDMA_ 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.	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value																									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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value																									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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
	Reset value																									0	0	0	0	0	0	0	0
0x0C	GPDMA_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.	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																									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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																									0	0	0	0	0	0	0	0
0x14 - 0x4C	Reserved	Reserved																															
0x50+0x80 * x (x=0 to 7)	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																
0x54 to 0x58+0x80*x (x=0 to 7)	Reserved	Reserved																															

Table 134. GPDMA 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+0x80 * x (x=0 to 7)	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 7)	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 7)	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.	Res.	SUSP	RESET	EN
	Reset value									0	0					0	0		0	0	0	0	0	0	0						0	0	0
0x68 to 0x8C+0x80 * x (x=0 to 7)	Reserved	Reserved																															
0x90+0x80 * x (x=0 to 7)	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 7)	GPDMA_CxTR2	TCEM[1:0]		Res.	Res.	Res.	Res.	TRIGPOL[1:0]		Res.	Res.	Res.	TRIGSEL [4:0]					TRIGM[1:0]		Res.	Res.	BREQ	DREQ	SWREQ	Res.	Res.	Res.	REQSEL[5:0]					
	Reset value	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 7)	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
0x9C+0x80 * x (x=0 to 7)	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 7)	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
0xA4 to 0xC8+0x80 * x (x=0 to 7)	Reserved	Reserved																															
0xCC+0x80 * x (x=0 to 7)	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		

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

18 Nested vectored interrupt controller (NVIC)

The NVIC is an Arm Cortex embedded interrupt controller that supports low latency interrupt processing.

18.1 NVIC main features

- 70 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.

18.2 Interrupt and exception vectors

The grey rows in *Table 135* indicate vectors without specific position.

Table 135. Vector table

Position	Priority	Type of priority	Acronym	Description	Address
-	-	-	-	Reserved	0x0000 0000
-	-4	Fixed	Reset	Reset	0x0000 0004
-14	-2	Fixed	NMI	Non maskable interrupt. - RCC clock security system (HSECSS) - FLASH ECC double error detection - RAMCFG SRAM2 parity error	0x0000 0008
-13	-3	Fixed	Secure HardFault	Secure hard fault when AIRCR.BFHFNMINS = 1	0x0000 000C
	-1	Fixed		Secure hard fault when AIRCR.BFHFNMINS = 0	
	-1	Fixed	Non-secure HardFault	Non-secure hard fault. All classes of fault	
-12	0	Settable	MemManage	Memory management	0x0000 0010
-11	1	Settable	BusFault	Pre-fetch fault, memory access fault	0x0000 0014
-10	2	Settable	UsageFault	Undefined instruction or illegal state	0x0000 0018
-9	3	Settable	SecureFault	Secure fault	0x0000 001C

Table 135. Vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
-	-	-	-	Reserved	0x0000 0020 - 0x0000 0028
-5	4	Settable	SVCall	System service call via SWI instruction	0x0000 002C
-4	5	Settable	Debug monitor	Debug monitor	0x0000 0030
-	-	-	-	Reserved	0x0000 0034
-2	6	Settable	PendSV	Pendable request for system service	0x0000 0038
-1	7	Settable	SysTick	System tick timer	0x0000 003C
0	8	Settable	WWDG	Window watchdog interrupt	0x0000 0040
1	9	Settable	PVD	Power voltage monitor	0x0000 0044
2	10	Settable	RTC	RTC non-secure global interrupts	0x0000 0048
3	11	Settable	RTC_S(1)	RTC secure global 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 interface non-secure global interrupt Flash ECC single error correction interrupt	0x0000 0058
7	15	Settable	FLASH_S(1)	Flash interface secure global interrupt	0x0000 005C
8	16	Settable	GTZC_TZIC(1)	GTZC_TZIC global interrupt	0x0000 0060
9	17	Settable	RCC	RCC non-secure global interrupt	0x0000 0064
10	18	Settable	RCC_S(1)	RCC secure global interrupt	0x0000 0068
11	19	Settable	EXTI0	EXTI line0 interrupt	0x0000 006C
12	20	Settable	EXTI1	EXTI line1 interrupt	0x0000 0070
13	21	Settable	EXTI2	EXTI line2 interrupt	0x0000 0074
14	22	Settable	EXTI3	EXTI line3 interrupt	0x0000 0078
15	23	Settable	EXTI4	EXTI line4 interrupt	0x0000 007C
16	24	Settable	EXTI5	EXTI line5 interrupt	0x0000 0080
17	25	Settable	EXTI6	EXTI line6 interrupt	0x0000 0084
18	26	Settable	EXTI7	EXTI line7 interrupt	0x0000 0088
19	27	Settable	EXTI8	EXTI line8 interrupt	0x0000 008C
20	28	Settable	EXTI9	EXTI line9 interrupt	0x0000 0090
21	29	Settable	EXTI10(1)	EXTI line10 interrupt	0x0000 0094
22	30	Settable	EXTI11(1)	EXTI line11 interrupt	0x0000 0098
23	31	Settable	EXTI12	EXTI line12 interrupt	0x0000 009C
24	32	Settable	EXTI13	EXTI line13 interrupt	0x0000 00A0

Table 135. Vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
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(1)	Secure AES interrupt	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	TIM1_BRK TIM1_TERR TIM1_IERR	TIM1 break TIM1 transition error TIM1 index error	0x0000 00D4
38	46	Settable	TIM1_UP	TIM1 update	0x0000 00D8
39	47	Settable	TIM1_TRG_COM TIM1_DIR TIM1_IDX	TIM1 trigger and commutation TIM1 direction change interrupt TIM1 index	0x0000 00DC
40	48	Settable	TIM1_CC	TIM1 capture compare interrupt	0x0000 00E0
41	49	Settable	TIM2	TIM2 global interrupt	0x0000 00E4
42	50	Settable	TIM3(1)	TIM3 global interrupt	0x0000 00E8
43	51	Settable	I2C1_EV(1)	I2C1 event interrupt	0x0000 00EC
44	52	Settable	I2C1_ER(1)	I2C1 error interrupt	0x0000 00F0
45	53	Settable	SPI1(1)	SPI1 global interrupt	0x0000 00F4
46	54	Settable	USART1	USART1 global interrupt	0x0000 00F8
47	55	Settable	USART2(1)	USART2 global interrupt	0x0000 00FC
48	56	Settable	LPUART1	LPUART1 global interrupt	0x0000 0100
49	57	Settable	LPTIM1	LPTIM1 global interrupt	0x0000 0104
50	58	Settable	LPTIM2(1)	LPTIM2 global interrupt	0x0000 0108
51	59	Settable	TIM16	TIM16 global interrupt	0x0000 010C
52	60	Settable	TIM17(1)	TIM17 global interrupt	0x0000 0110
53	61	Settable	COMP(2)	COMP1/COMP2 interrupt	0x0000 0114
54	62	Settable	I2C3_EV	I2C3 event interrupt	0x0000 0118

Table 135. Vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
55	63	Settable	I2C3_ER	I2C3 error interrupt	0x0000 011C
56	64	Settable	SAI1(2)	SAI1 global interrupt	0x0000 0120
57	65	Settable	TSC	TSC global interrupt	0x0000 0124
58	66	Settable	AES	AES global interrupt	0x0000 0128
59	67	Settable	RNG	RNG global interrupt	0x0000 012C
60	68	Settable	FPU	Floating point interrupt	0x0000 0130
61	69	Settable	HASH	HASH interrupt	0x0000 0134
62	70	Settable	PKA	PKA global interrupt	0x0000 0138
63	71	Settable	SPI3	SPI3 global interrupt	0x0000 013C
64	72	Settable	ICACHE	Instruction cache global interrupt	0x0000 0140
65	73	Settable	ADC4	ADC4 global interrupt	0x0000 0144
66	74	Settable	RADIO	2.4 GHz RADIO global interrupt	0x0000 0148
67	75	Settable	WKUP	PWR non-secure global WKUP pin interrupt	0x0000 014C
68	76	Settable	HSEM	HSEM non-secure interrupt	0x0000 0150
69	77	Settable	HSEM_S(1)	HSEM secure interrupt	0x0000 0154
70	78	Settable	WKUP_S(1)	PWR secure WKUP pin interrupt	0x0000 0158
71	79	Settable	RCC_AUDIO(2)	RCC audio synchronization interrupt	0x0000 015C

1. Available only on STM32WBA52/54/55xx devices.
2. Available only on STM32WBA54/55xx devices.

19 Extended interrupts and event controller (EXTI)

The extended interrupts and event controller (EXTI) manages the individual CPU and system wake-up through configurable event inputs. It provides wake-up 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 wake-up 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.

19.1 EXTI main features

- 19 input events supported
- All event inputs allow the possibility to wake up the system.
- Events that do not have an associated wake-up 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 wake-up 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 wake-up, 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 privileged.
- EXTI IO port selection

19.2 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 67*:

- 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 wake-up, interrupt and event outputs, and their masking
- the EXTI mux provides the IO port selection on to the EXTI event signal.

Figure 67. EXTI block diagram

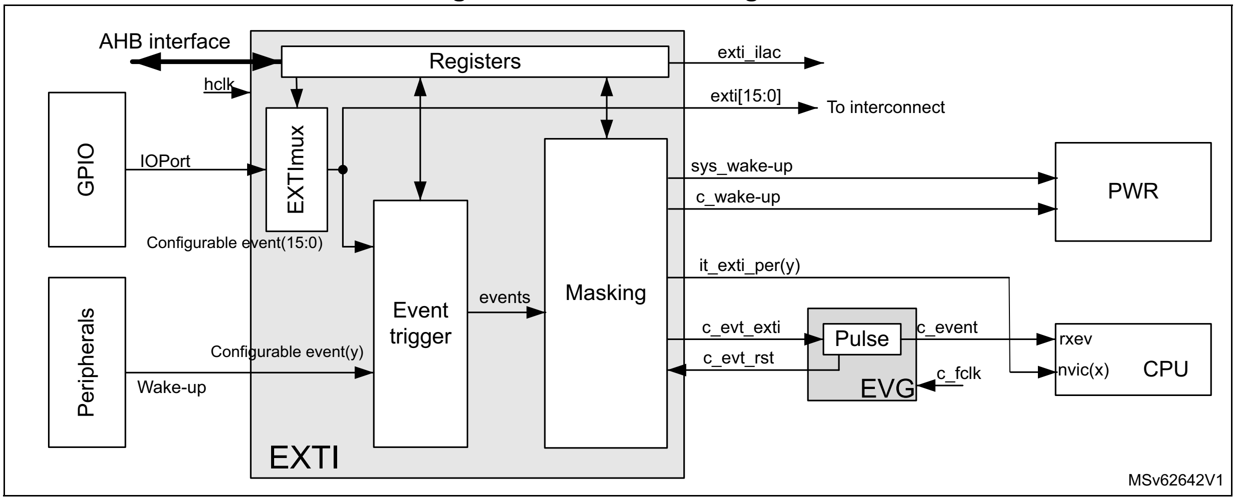


Table 136. EXTI pin overview

Pin name	I/O	Description
AHB interface	I/O	EXTI register bus interface. When one event is configured to enable security, the AHB interface supports secure accesses.
hclk	I	AHB bus clock and EXTI system clock
Configurable event(y)	I	Asynchronous wake-up 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 wake-up request to PWR for ck_sys and hclk
c_wakeup	O	Wake-up request to PWR for CPU, synchronous to hclk

Table 137. EVG pin overview

Pin name	I/O	Description
c_fclk	I	CPU free running clock
c_evt_in	I	High-level sensitive events input from EXTI, asynchronous to CPU clock
c_event	O	Event pulse, synchronous to CPU clock
c_evt_rst	O	Event reset signal, synchronous to CPU clock

19.2.1 EXTI connections between peripherals and CPU

Some peripherals able to generate wake-up or interrupt events when the system is in Stop mode are connected to the EXTI.

- Peripheral wake-up signals that generate a pulse or do not have an interrupt status bit in the peripheral are connected to an EXTI configurable event input. For these events, the EXTI provides a status pending bit that must be cleared. It is the EXTI interrupt, associated with the status bit, that interrupts the CPU.
- All GPIO ports input to the EXTI multiplexer make possible 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 wake-up signals are connected to the PWR and are used to wake up the system and the CPU sub-system bus clocks.

19.2.2 EXTI interrupt/event mapping

The EXTI lines are connected as shown in *Table 138*.

Table 138. EXTI line connections

EXTI line	Line source	Line type
0-15(1)	GPIO	Configurable
16	PVD output	Configurable
17	COMP1 output(2)	Configurable
18	COMP2 output(2)	Configurable

1. EXTI lines 11 and 10 are available only on STM32WBA52/54/55xx devices.
2. Available only on STM32WBA54/55xx devices.

19.3 EXTI functional description

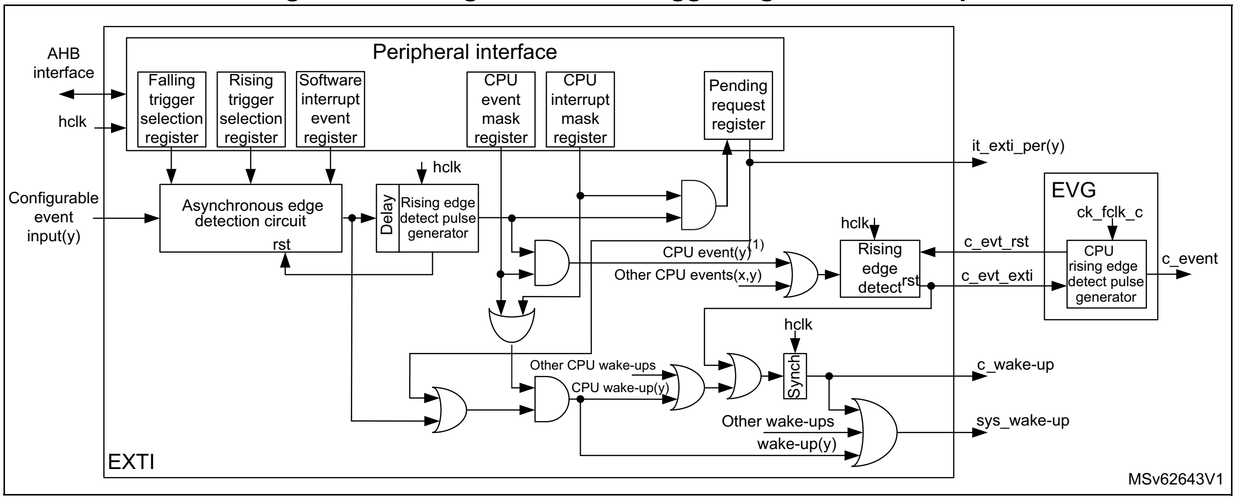
The events features are controlled from register bits as follows:

- 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 wake-up and interrupt enable in the
  EXTI CPU wake-up with interrupt mask register (EXTI_IMR1)
- CPU wake-up and event enable in the
  EXTI CPU wake-up with event mask register (EXTI_EMR1)

19.3.1 EXTI configurable event input wake-up

Figure 68 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 wake-up event.

Figure 68. Configurable event trigger logic CPU wake-up



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 wake-up (interrupt) mask register and 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.

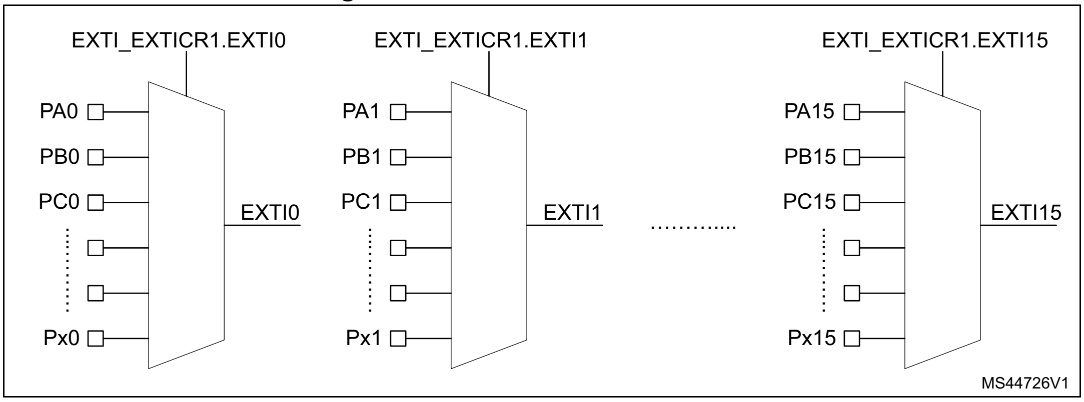
When a CPU wake-up (interrupt) or CPU event is enabled, the asynchronous edge detection circuit is reset by the clocked delay and rising edge detect pulse generator. This guarantees that the EXTI hclk clock is woken up before the asynchronous edge detection circuit is reset.

Note: A detected configurable event interrupt pending request can be cleared by the CPU with the correct access permission. The system is unable to enter into low-power modes as long as an interrupt pending request is active.

19.3.2 EXTI mux selection

The EXTI mux allows the selection of GPIOs as interrupts and wake-up. 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_EXTICR1)*.

Figure 69. 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.

19.4 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 wake-up generation is conditioned by the CPU interrupt mask and CPU event mask.

Table 139. 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	CPU event	CPU wake-up
0	0	No	Masked	Masked	Masked
	1			Yes	Yes
1	0	Status latched	Yes	Masked	Yes
	1			Yes	Yes

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. When the event is unmasked in EXTI_IMR or EXTI_EMR, the rising edge event pending bit is set in EXTI_RPR, whatever the setting in EXTI_RTSR.

19.5 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 140. 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 privileged. 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	
EXTI_HWCFGR	R	Non-secure unprivileged
EXTI_VER	R	
EXTI_ID	R	
EXTI_SID	R	

1. Security is enabled with the individual input event (EXTI_SECCFGR register).
2. Privilege is enabled with the individual Input event (EXTI_PRIVCFGR register).

19.5.1 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.

19.5.2 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.

19.6 EXTI registers

The EXTI register map is divided in sections, as indicated in *Table 141*.

Table 141. EXTI register map sections

Address offset	Description
0x000 - 0x01C	General configurable event [18:0] 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.

19.6.1 EXTI rising trigger selection register (EXTI_RTSR1)

Address offset: 0x000

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT18	RT17	RT16
													rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 18:0 **RT[18:0]:** Rising trigger event configuration bit of configurable event input x[(1)] (x = 0 to 18)

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

Note that bits 18:17 are reserved on STM32WBA50/52xx devices.

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.

19.6.2 EXTI falling trigger selection register (EXTI_FTSR1)

Address offset: 0x004

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT18	FT17	FT16
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 18:0 **FT[18:0]:** Falling trigger event configuration bit of configurable event input x [(1)] (x = 0 to 18)

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.

Note that bits 18:17 are reserved on STM32WBA50/52xx devices.

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.

19.6.3 EXTI software interrupt event register (EXTI_SWIER1)

Address offset: 0x008

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI18	SWI17	SWI16
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 18:0 **SWI[18:0]:** Software interrupt on event x (x = 0 to 18)

When EXTI_SECFGR.SECx is disabled, SWIx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, SWIx can only be accessed with secure access. Non-secure write to this SWI x is discarded, non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, SWIx can be accessed with unprivileged and 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.
When unmasked in EXTI_IMR or EXTI_EMR, a software interrupt is generated and EXTI_RPR is set, a software interrupt is generated independently 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.
Note that bits 18:17 are reserved on STM32WBA50/52xx devices.

19.6.4 EXTI rising edge pending register (EXTI_RPR1)

Address offset: 0x00C

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF18	RPIF17	RPIF16
													rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPIF15	RPIF14	RPIF13	RPIF12	RPIF11	RPIF10	RPIF9	RPIF8	RPIF7	RPIF6	RPIF5	RPIF4	RPIF3	RPIF2	RPIF1	RPIF0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

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

Bits 18:0 **RPIF[18:0]:** configurable event inputs x rising edge pending bit (x = 0 to 18)

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 when unmasked in EXTI_IMR or EXTI_EMR and an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing 1 to it.
Note that bits 18:17 are reserved on STM32WBA50/52xx devices.

19.6.5 EXTI falling edge pending register (EXTI_FPR1)

Address offset: 0x010

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF18	FPIF17	FPIF16
													rc_w1	rc_w1	rc_w1
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FPIF15	FPIF14	FPIF13	FPIF12	FPIF11	FPIF10	FPIF9	FPIF8	FPIF7	FPIF6	FPIF5	FPIF4	FPIF3	FPIF2	FPIF1	FPIF0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

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

Bits 18:0 **FPIF[18:0]:** configurable event inputs x falling edge pending bit (x =0 to 18)

When EXTI_SECCFGR.SECx is disabled, FPIFx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, FPIFx can only be accessed with secure access. Non-secure write to this FPIFx is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, FPIFx can be accessed with unprivileged and 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.

Note that bits 18:17 are reserved on STM32WBA50/52xx devices.

19.6.6 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.	Res.	Res.	Res.	Res.	Res.	SEC18	SEC17	SEC16
													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 31:19 Reserved, must be kept at reset value.

Bits 18:0 **SEC[18:0]:** Security enable on event input x (x = 0 to 18)
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)
Note that bits 18:17 are reserved on STM32WBA50/52xx devices.

19.6.7 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.	Res.	Res.	Res.	Res.	Res.	PRIV18	PRIV17	PRIV16
													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 31:19 Reserved, must be kept at reset value.

Bits 18:0 **PRIV[18:0]:** Security enable on event input x (x = 0 to 18)
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)
Note that bits 18:17 are reserved on STM32WBA50/52xx devices.

19.6.8 EXTI external interrupt selection register (EXTI_EXTICR1)

Address offset: 0x060

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI3[7:0]								EXTI2[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
EXTI1[7:0]								EXTI0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI3[7:0]:** EXTI3 GPIO port selection
These bits are written by software to select the source input for EXTI3 external interrupt.
When EXTI_SECCFGR.SEC3 is disabled, EXTI3 can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SEC3 is enabled, EXTI3 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIV3 is disabled, EXTI3 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV3 is enabled, EXTI3 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA3 pin
0x01: PB3 pin
0x07: PH3 pin
Others: reserved

Bits 23:16 **EXTI2[7:0]:** EXTI2 GPIO port selection
These bits are written by software to select the source input for EXTI2 external interrupt.
When EXTI_SECCFGR.SEC2 is disabled, EXTI2 can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SEC2 is enabled, EXTI2 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIV2 is disabled, EXTI2 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV2 is enabled, EXTI2 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA2 pin
0x01: PB2 pin
Others: reserved

Bits 15:8 **EXTI1[7:0]:** EXTI1 GPIO port selection
These bits are written by software to select the source input for EXTI1 external interrupt.
When EXTI_SECCFGR.SEC1 is disabled, EXTI1 can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SEC1 is enabled, EXTI1 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIV1 is disabled, EXTI1 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV1 is enabled, EXTI1 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA1 pin
0x01: PB1 pin
Others: reserved

Bits 7:0 **EXTI0[7:0]:** EXTI0 GPIO port selection

These bits are written by software to select the source input for EXTI0 external interrupt.
When EXTI_SECCFGR.SEC0 is disabled, EXTI0 can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SEC0 is enabled, EXTI0 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIV0 is disabled, EXTI0 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV0 is enabled, EXTI0 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA0 pin
0x01: PB0 pin
Others: reserved

19.6.9 EXTI external interrupt selection register (EXTI_EXTICR2)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI7[7:0]								EXTI6[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
EXTI5[7:0]								EXTI4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI7[7:0]:** EXTI7 GPIO port selection

These bits are written by software to select the source input for EXTI7 external interrupt.
When EXTI_SECCFGR.SEC7 is disabled, EXTI7 can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SEC7 is enabled, EXTI7 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIV7 is disabled, EXTI7 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV7 is enabled, EXTI7 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA7 pin
0x01: PB7 pin
Others: reserved

Bits 23:16 **EXTI6[7:0]:** EXTI6 GPIO port selection

These bits are written by software to select the source input for EXTI6 external interrupt.
When EXTI_SECCFGR.SEC6 is disabled, EXTI6 can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SEC6 is enabled, EXTI6 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIV6 is disabled, EXTI6 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV6 is enabled, EXTI6 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA6 pin
0x01: PB6 pin
Others: reserved

Bits 15:8 **EXTI5[7:0]:** EXTI5 GPIO port selection

These bits are written by software to select the source input for EXTI5 external interrupt.
When EXTI_SECCFGR.SEC5 is disabled, EXTI5 can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SEC5 is enabled, EXTI5 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIV5 is disabled, EXTI5 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV5 is enabled, EXTI5 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA5 pin
0x01: PB5 pin
Others: reserved

Bits 7:0 **EXTI4[7:0]:** EXTI4 GPIO port selection

These bits are written by software to select the source input for EXTI4 external interrupt.
When EXTI_SECCFGR.SEC4 is disabled, EXTI4 can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SEC4 is enabled, EXTI4 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIV4 is disabled, EXTI4 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV4 is enabled, EXTI4 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA4 pin (no port pin selected on STM32WBA50/52/54xx devices)
0x01: PB4 pin
Others: reserved

19.6.10 EXTI external interrupt selection register (EXTI_EXTICR3)

Address offset: 0x068

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI11[7:0]								EXTI10[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**
EXTI9[7:0]								EXTI8[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI11[7:0]:** EXTI11 GPIO port selection

These bits are written by software to select the source input for EXTI11 external interrupt.
When EXTI_SECCFGR.SEC11 is disabled, EXTI11 can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SEC11 is enabled, EXTI11 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIV11 is disabled, EXTI11 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV11 is enabled, EXTI11 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA11 pin
0x01: PB11 pin
Others: reserved

Bits 23:16 **EXTI10[7:0]:** EXTI10 GPIO port selection

These bits are written by software to select the source input for EXTI10 external interrupt.
When EXTI_SECCFGR.SEC10 is disabled, EXTI10 can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SEC10 is enabled, EXTI10 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIV10 is disabled, EXTI10 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV10 is enabled, EXTI10 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA10 pin
0x01: PB10 pin (reserved on STM32WBA55xx devices)
Others: reserved

Bits 15:8 **EXTI9[7:0]:** EXTI9 GPIO port selection

These bits are written by software to select the source input for EXTI9 external interrupt.
When EXTI_SECCFGR.SEC9 is disabled, EXTI9 can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SEC9 is enabled, EXTI9 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIV9 is disabled, EXTI9 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV9 is enabled, EXTI9 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA9 pin
0x01: PB9 pin
Others: reserved

Bits 7:0 **EXTI8[7:0]:** EXTI8 GPIO port selection

These bits are written by software to select the source input for EXTI8 external interrupt.
When EXTI_SECCFGR.SEC8 is disabled, EXTI8 can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SEC8 is enabled, EXTI8 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIV8 is disabled, EXTI8 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV8 is enabled, EXTI8 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA8 pin
0x01: PB8 pin
Others: reserved

19.6.11 EXTI external interrupt selection register (EXTI_EXTICR4)

Address offset: 0x06C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI15[7:0]								EXTI14[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
EXTI13[7:0]								EXTI12[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI15[7:0]:** EXTI15 GPIO port selection

These bits are written by software to select the source input for EXTI15 external interrupt.
When EXTI_SECCFGR.SEC15 is disabled, EXTI15 can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SEC15 is enabled, EXTI15 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIV15 is disabled, EXTI15 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV15 is enabled, EXTI15 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA15 pin
0x01: PB15 pin
0x02: PC15 pin
Others: reserved

Bits 23:16 **EXTI14[7:0]:** EXTI14 GPIO port selection

These bits are written by software to select the source input for EXTI14 external interrupt.
When EXTI_SECCFGR.SEC14 is disabled, EXTI14 can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SEC14 is enabled, EXTI14 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIV14 is disabled, EXTI14 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV14 is enabled, EXTI14 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA14 pin
0x01: PB14 pin
0x02: PC14 pin
Others: reserved

Bits 15:8 **EXTI13[7:0]:** EXTI13 GPIO port selection

These bits are written by software to select the source input for EXTI13 external interrupt.
When EXTI_SECCFGR.SEC13 is disabled, EXTI13 can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SEC13 is enabled, EXTI13 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIV13 is disabled, EXTI13 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV13 is enabled, EXTI13 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA13 pin
0x01: PB13 pin
0x02: PC13 pin
Others: reserved

Bits 7:0 **EXTI12[7:0]:** EXTI0 GPIO port selection

These bits are written by software to select the source input for EXTI12 external interrupt.
When EXTI_SECCFGR.SEC12 is disabled, EXTI12 can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SEC12 is enabled, EXTI12 can only be accessed with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIV12 is disabled, EXTI12 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIV12 is enabled, EXTI12 can only be accessed with privileged access. Unprivileged write to this bit is discarded.
0x00: PA12 pin
0x01: PB12 pin
Others: reserved

19.6.12 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.

19.6.13 EXTI CPU wake-up 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.	Res.	Res.	Res.	Res.	Res.	IM18	IM17	IM16
													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 31:19 Reserved, must be kept at reset value.

Bits 18:0 **IM[18:0]:** CPU wake-up with interrupt mask on event input x [(1)] (x = 0 to 18)

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: Wake-up with interrupt request from input event x is masked.

1: Wake-up with interrupt request from input event x is unmasked.

Note that bits 18:17 are reserved on STM32WBA50/52xx devices.

1. The reset value for configurable event inputs is set to 0 to disable the interrupt by default.

19.6.14 EXTI CPU wake-up 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.	Res.	Res.	Res.	Res.	Res.	EM18	EM17	EM16
													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 31:19 Reserved, must be kept at reset value.

Bits 18:0 **EM[18:0]:** CPU wake-up with event generation mask on event input x (x = 0 to 18)

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: Wake-up with event generation from Line x is masked.

1: Wake-up with event generation from Line x is unmasked.

Note that bits 18:17 are reserved on STM32WBA50/52xx devices.

19.6.15 EXTI register map

Table 142. EXTI register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	EXTI_RTSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT18[1]	RT17[1]	RT16	RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	EXTI_FTSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT18[1]	FT17[1]	FT16	FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	EXTI_SWIER1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI18[1]	SWI17[1]	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
0x00C	EXTI_RPR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF18[1]	RPIF17[1]	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
0x010	EXTI_FPR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF18[1]	FPIF17[1]	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
0x014	EXTI_ SECCFG1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC18[1]	SEC17[1]	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
0x018	EXTI_ PRIVCFG1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV18[1]	PRIV17[1]	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
0x020 to 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
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 to 0x07C	Reserved	Reserved																															

Table 142. EXTI 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	**EXTI_IMR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM18(1)	IM17(1)	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
0x084	**EXTI_EMR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM18(1)	EM17(1)	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
0x088 to 0x3FC	Reserved	Reserved																															

1. Bit reserved on STM32WBA50/52xx devices.

Refer to *Section 2.3* for the register boundary addresses.

20 Cyclic redundancy check calculation unit (CRC)

20.1 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.

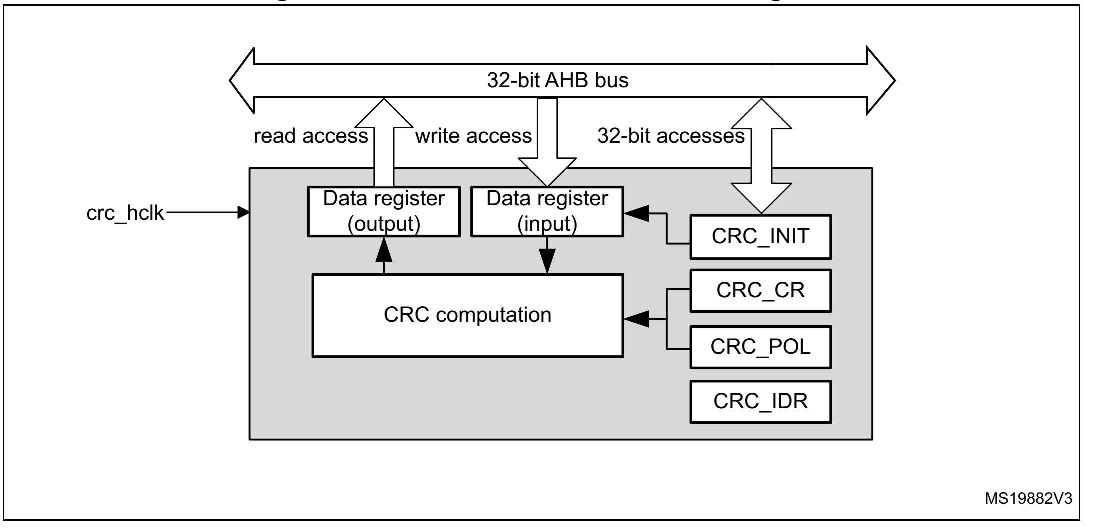
20.2 CRC main features

- 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

20.3 CRC functional description

20.3.1 CRC block diagram

Figure 70. CRC calculation unit block diagram



20.3.2 CRC internal signals

Table 143. CRC internal input/output signals

Signal name	Signal type	Description
crc_hclk	Digital input	AHB clock

20.3.3 CRC operation

The CRC calculation unit has a single 32-bit read/write data register (CRC_DR). It is used to input new data (write access), and holds the result of the previous CRC calculation (read access).

Each write operation to the data register creates a combination of the previous CRC value (stored in CRC_DR) and the new one. CRC computation is done on the whole 32-bit data word or byte by byte depending on the format of the data being written.

The CRC_DR register can be accessed by word, right-aligned half-word and right-aligned byte. For the other registers only 32-bit accesses are allowed.

The duration of the computation depends on data width:

- 4 AHB clock cycles for 32 bits
- 2 AHB clock cycles for 16 bits
- 1 AHB clock cycles for 8 bits

An input buffer allows a second data to be immediately written without waiting for any wait-states due to the previous CRC calculation.

The data size can be dynamically adjusted to minimize the number of write accesses for a given number of bytes. For instance, a CRC for 5 bytes can be computed with a word write followed by a byte write.

The input data can be reversed to manage the various endianness schemes. The reversing operation can be performed on 8 bits, 16 bits and 32 bits depending on the REV_IN[1:0] bits in the CRC_CR register.

For example, 0x1A2B3C4D input data are used for CRC calculation as:

- 0x58D43CB2 with bit-reversal done by byte
- 0xD458B23C with bit-reversal done by half-word
- 0xB23CD458 with bit-reversal done on the full word

The output data can also be reversed by setting the REV_OUT bit in the CRC_CR register.

The operation is done at bit level. For example, 0x11223344 output data are converted to 0x22CC4488.

The CRC calculator can be initialized to a programmable value using the RESET control bit in the CRC_CR register (the default value is 0xFFFFFFFF).

The initial CRC value can be programmed with the CRC_INIT register. The CRC_DR register is automatically initialized upon CRC_INIT register write access.

The CRC_IDR register can be used to hold a temporary value related to CRC calculation. It is not affected by the RESET bit in the CRC_CR register.

Polynomial programmability

The polynomial coefficients are fully programmable through the CRC_POL register, and the polynomial size can be configured to be 7, 8, 16 or 32 bits by programming the POLYSIZE[1:0] bits in the CRC_CR register. Even polynomials are not supported.

Note: The type of an even polynomial is $X+X^2+..+X^n$*, while the type of an odd polynomial is* $1+X+X^2+..+X^n$*.*

If the CRC data is less than 32-bit, its value can be read from the least significant bits of the CRC_DR register.

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size can not be performed during a CRC calculation. As a result, if a CRC calculation is ongoing, the application must either reset it or perform a CRC_DR read before changing the polynomial.

The default polynomial value is the CRC-32 (Ethernet) polynomial: 0x4C11DB7.

20.4 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.

20.4.1 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.

20.4.2 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

20.4.3 CRC control register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV_ OUT	REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
								rw	rw	rw	rw	rw			rs

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

Bit 7 **REV_OUT**: Reverse output data

This bit controls the reversal of the bit order of the output data.

0: Bit order not affected

1: Bit-reversed output format

Bits 6:5 **REV_IN[1:0]**: Reverse input data

This bitfield controls the reversal of the bit order of the input data

00: Bit order not affected

01: Bit reversal done by byte

10: Bit reversal done by half-word

11: Bit reversal done by word

Bits 4:3 **POLYSIZE[1:0]**: Polynomial size

These bits control the size of the polynomial.

00: 32 bit polynomial

01: 16 bit polynomial

10: 8 bit polynomial

11: 7 bit polynomial

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **RESET**: RESET bit

This bit is set by software to reset the CRC calculation unit and set the data register to the value stored in the CRC_INIT register. This bit can only be set, it is automatically cleared by hardware

20.4.4 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.

20.4.5 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.

20.4.6 CRC register map

Table 144. 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: Memory organization* for the register boundary addresses.

21 Analog-to-digital converter (ADC4)

21.1 Introduction

The 12-bit ADC is a successive approximation analog-to-digital converter. It has up to 13 multiplexed channels enabling it to measure signals from 10 external sources and 3 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.

21.2 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
  - Wake-up from Stop on ADC interrupts
  - Enter and exit from Deep-power-down mode managed automatically
- Analog input channels
  - 10 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})

- 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: $V_{SSA} \leq V_{IN} \leq V_{DDA}$

21.3 ADC implementation

Table 145. ADC features(1)

ADC modes/features	ADC4		
	STM32WBA52xx	STM32WBA54xx	STM32WBA55xx
ADC4 channels	ADC4_IN[10:6,4:1]	ADC4_IN[10:6,4:1]	ADC4_IN[10:1]
Resolution	12 bits	12 bits	12 bits
Maximum sampling speed for 12-bit resolution	2.5 Msps	2.5 Msps	2.5 Msps
Hardware offset calibration	X	X	X
Hardware linearity calibration	-	-	-
Single-ended inputs	X	X	X
Differential inputs	-	-	-
Injected channel conversion	-	-	-
Oversampling	up to x256	up to x256	up to x256
Data register	16 bits	16 bits	16 bits
DMA support	X	X	X
Parallel data output to MDF	-	-	-
Dual mode	-	-	-
Autonomous mode	X	X	X
Offset compensation	-	-	-

Table 145. ADC features[1] (continued)

ADC modes/features	ADC4		
	STM32WBA52xx	STM32WBA54xx	STM32WBA55xx
Gain compensation	-	-	-
Number of analog watchdogs	3	3	3
Wakeup from Stop mode	X	X	X

1. Note: ‘X’ = supported, ‘-’= not supported.

Table 146. Memory location of the temperature sensor calibration values

Name	Description	Memory address
TS_CAL1	Temperature sensor 12-bit raw data acquired by ADC4 at 30 °C (± 5 °C), $V_{DDA} = V_{REF+}$ = 3.0 V (±10 mV)	0x0BF9 0710 - 0x0BF9 0711
TS_CAL2	Temperature sensor 12-bit raw data acquired by ADC4 at 130 °C (± 5 °C), $V_{DDA} = V_{REF+}$ = 3.0 V (±10 mV)	0x0BF9 0742 - 0x0BF9 0743

Table 147. Memory location of the internal reference voltage sensor calibration value

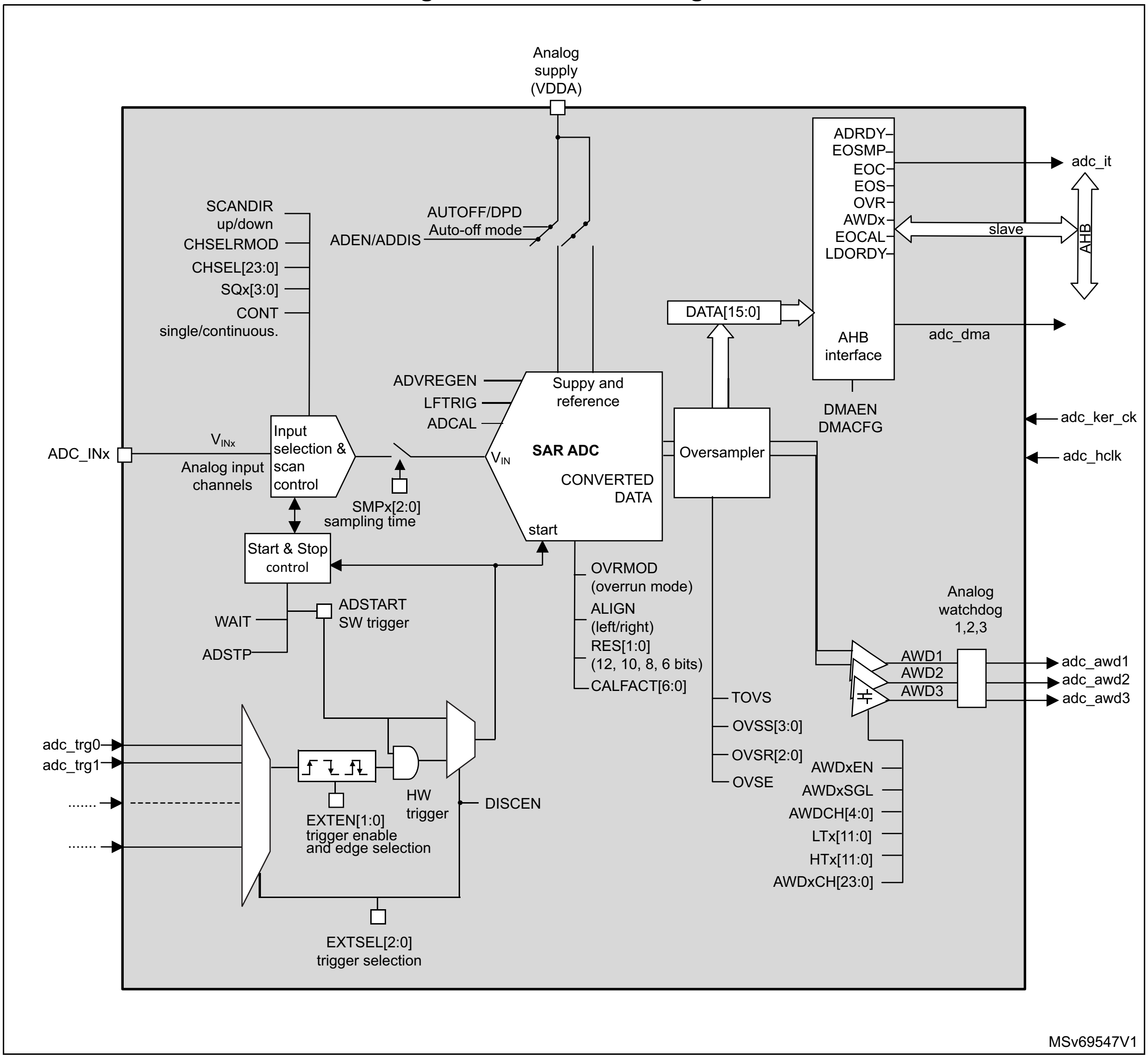
Name	Description	Memory address
VREFINT_CAL	12-bit raw data acquired by ADC4 at 30 °C (± 5 °C), $V_{DDA} = V_{REF+}$ = 3.0 V (±10 mV)	0x0BF9 07A5 - 0x0BF9 07A6

21.4 ADC functional description

21.4.1 ADC block diagram

Figure 71 shows the ADC block diagram and *Table 148* gives the ADC pin description.

Figure 71. ADC block diagram



21.4.2 ADC pins and internal signals

Table 148. 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}.
ADC_INx	Analog input signals	10 external analog input channels.

Table 149. 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 150. 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 $V_{IN}[12]$	V_{CORE} (internal logic supply voltage).
adc_trg0	tim1_trgo2
adc_trg1	tim1_oc4
adc_trg2	tim2_trgo
adc_trg3	Reserved
adc_trg4	Reserved
adc_trg5	lptim1_ch1
adc_trg6	Reserved
adc_trg7	exti15

21.4.3 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),

21.4.4 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_{DDA} 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 mode).
- The ADC peripheral is reset.

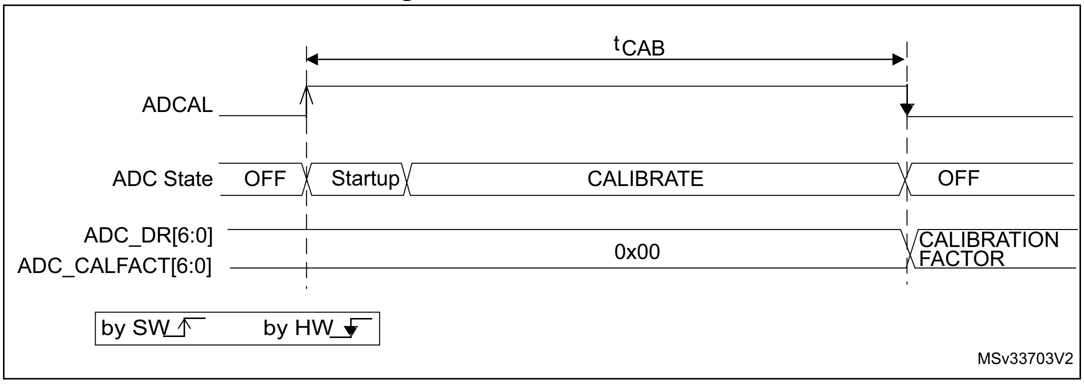
The calibration factor is lost each time power is removed from the ADC (for example when the product enters Standby mode). Still, it is possible to save and restore the calibration factor by software to save time when re-starting the ADC (as long as temperature and voltage are stable during the ADC power-down).

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 72. ADC calibration

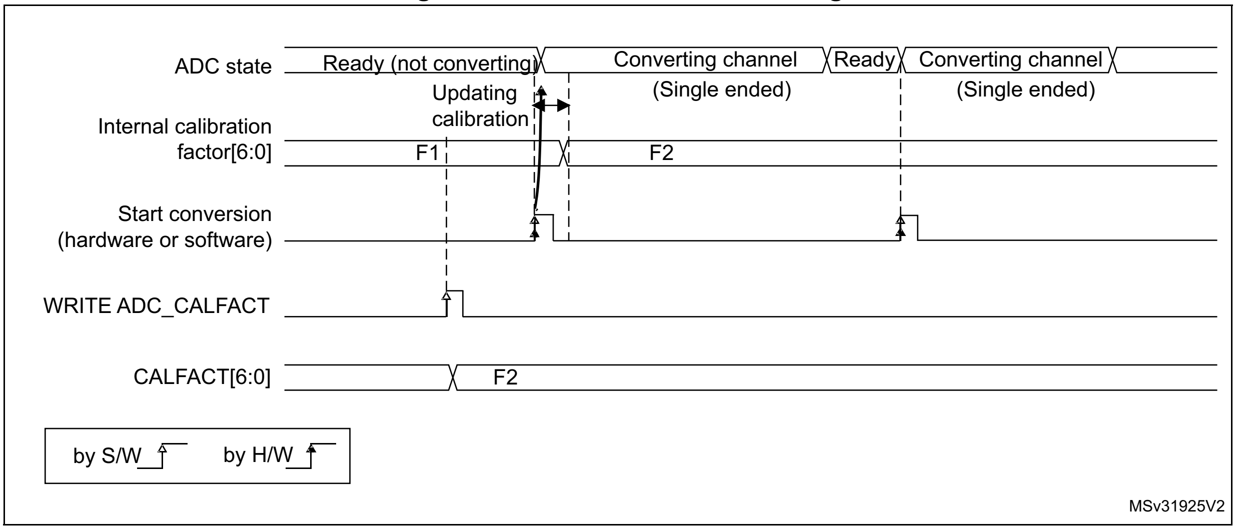


1. Refer to the device datasheet for the value of t_{CAB}.

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 73. Calibration factor forcing



21.4.5 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 74*, 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 21.4.16: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN)*) or when an external trigger event occurs if triggers are enabled.

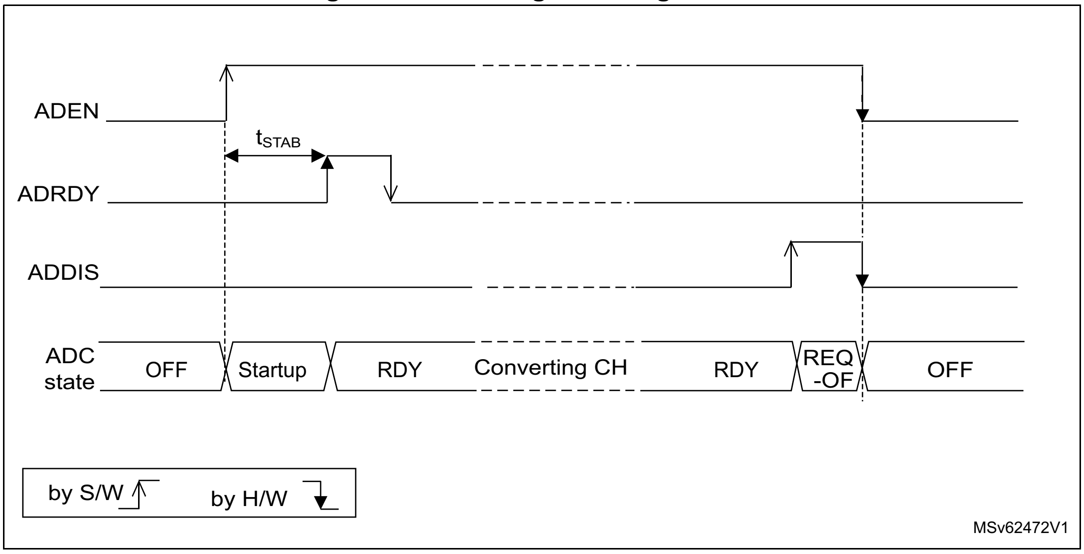
Follow the procedure below to enable the ADC:

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 74. Enabling/disabling the ADC

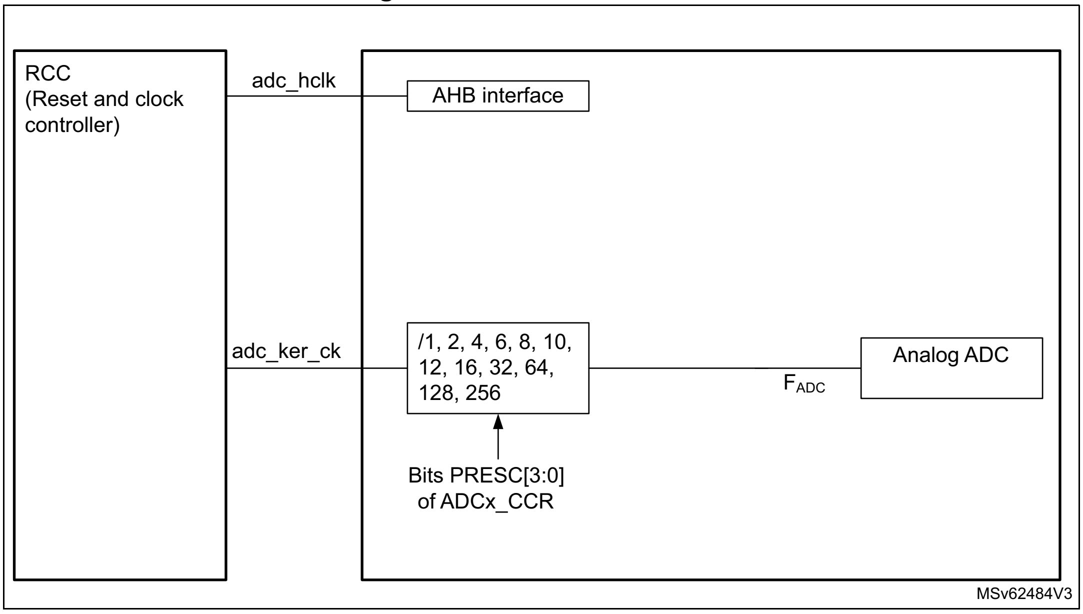


Note: In auto-off mode (AUTOFF = 1) the power-on/off phases are performed automatically, by hardware and the ADRDY flag is not set.

21.4.6 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 75. 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 75: ADC clock scheme*). This selection is done in the RCC (refer to th RCC section for more information):

- The ADC clock can be provided by an internal or external clock source, which is independent and asynchronous with the bus clock.
- The ADC clock can be derived from the bus clock by selecting the adc_ker_ck as 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 151. 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

Table 151. Latency between trigger and start of conversion(1)

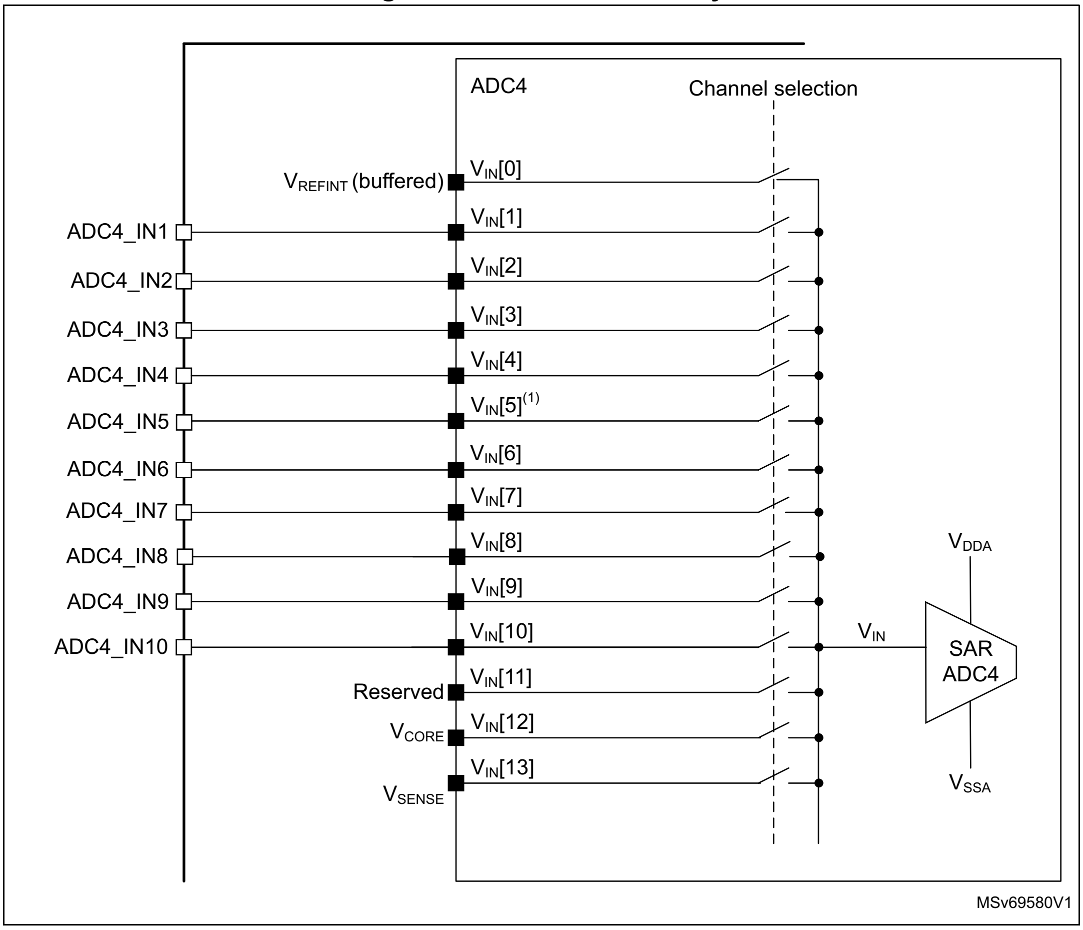
ADC clock source	Latency between the trigger event and the start of conversion
Bus clock divided by 4	Latency is deterministic (no jitter) and equal to 3.75 ADC clock cycles
Bus clock divided by 1	Latency is deterministic (no jitter) and equal to 4 ADC clock cycles

1. Refer to the device datasheet for the maximum F_{ADC} frequency.

21.4.7 ADC connectivity

ADC inputs are connected to the external channels as well as internal sources as described in *Figure 76*.

Figure 76. ADC4 connectivity



1. $V_{IN}[5]$ is connected to ADC4_IN5. It is not available on STM32WBA52/54xx devices.

21.4.8 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 21.7: 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).*

21.4.9 Channel selection (CHSEL, SCANDIR, CHSELRMOD)

There are up to 13 multiplexed channels:

- 10 analog inputs from GPIO pins (ADC_INx)
- 3 internal analog inputs: temperature Sensor, internal reference voltage, V_{CORE}

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.
  - 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, V_{REFINT}, and V_{CORE} internal channels

The temperature sensor, the internal reference voltage (V_{REFINT}), and V_{CORE} are connected to ADC internal channels. Refer to *Table ADC interconnection* in *Section 21.4.2: ADC pins and internal signals* for details.

21.4.10 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.

I/O analog switch voltage booster

The resistance of the I/O analog switch increases when the V_{DDA} voltage is too low. The sampling time must consequently be adapted accordingly (refer to the device datasheet for the corresponding electrical characteristics). This resistance can be minimized at low V_{DDA} voltage by enabling an internal voltage booster through the BOOSTEN bit of the

SYSCFG_CFGR1 register or by selecting a V_{DD} booster voltage through the ANASWVDD bit of the SYSCFG_CFGR1 register.

21.4.11 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.

21.4.12 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.

21.4.13 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 21.4.15: Stopping an ongoing conversion (ADSTP)*).

When the ADC operates in autonomous mode (DPD bit transition from 1 to 0, see *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.

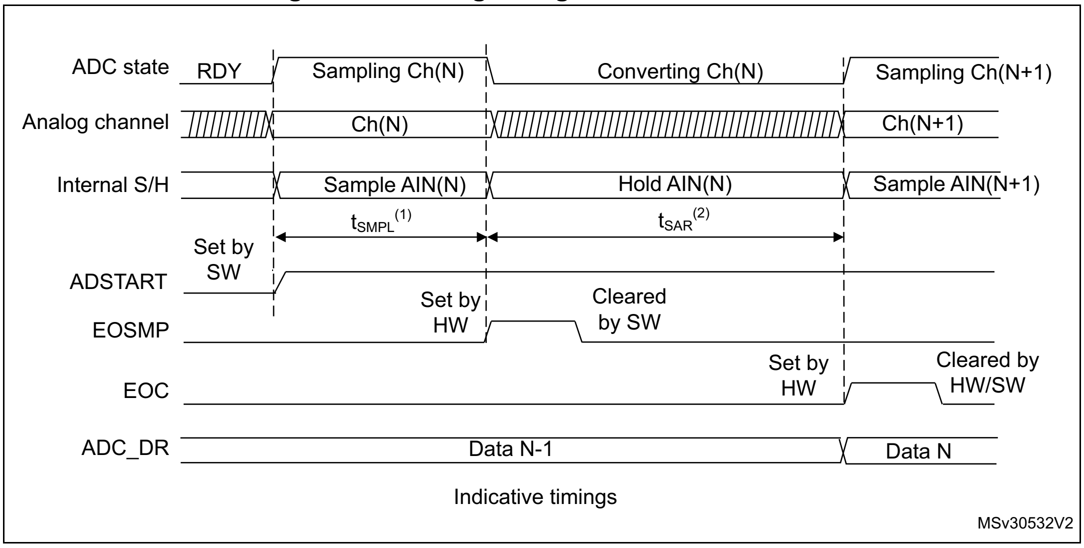
21.4.14 Timings

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:

$$t_{CONV} = t_{SMPL} + t_{SAR} = [1.5_{|min} + 12.5_{|12bit}] \times t_{ADC\_CLK}$$

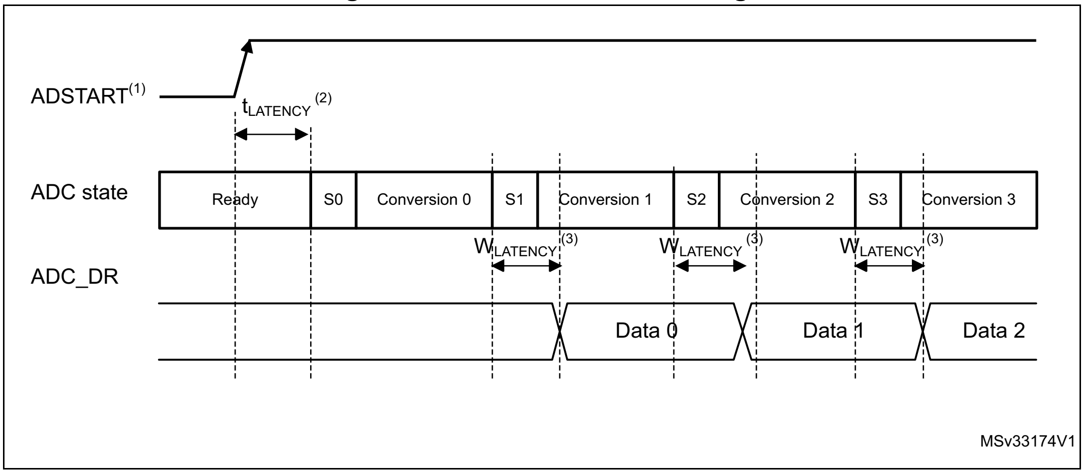
$$t_{CONV} = t_{SMPL} + t_{SAR} = 42.9\ ns_{|min} + 357.1\ ns_{|12bit} = 0.400\ \mu s_{|min}\ (for\ f_{ADC\_CLK} = 35\ MHz)$$

Figure 77. Analog-to-digital conversion time



1. t_{SMPL} depends on SMP[2:0].
2. t_{SAR} depends on RES[2:0].

Figure 78. 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).

21.4.15 Stopping an ongoing conversion (ADSTP)

The software can decide to stop any ongoing conversions by setting ADSTP to 1 in the ADC_CR register.

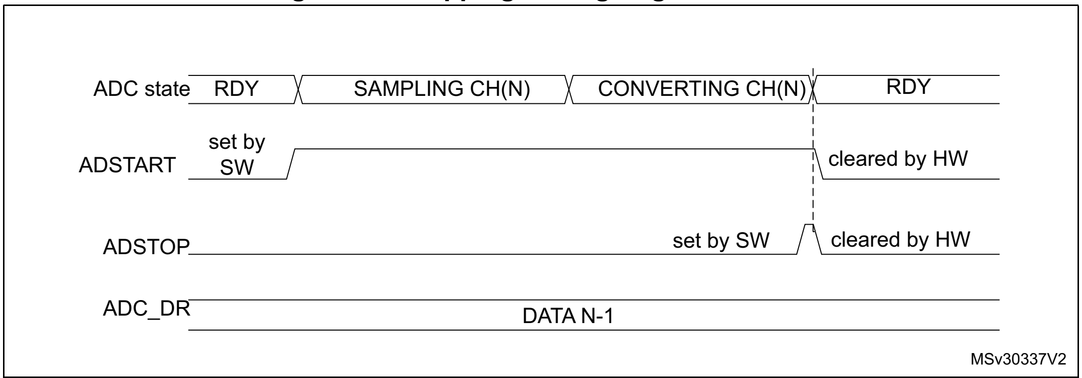
This resets the ADC operation and the ADC is idle, ready for a new operation.

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 79. Stopping an ongoing conversion



21.4.16 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 152 provides the correspondence between the EXTEN[1:0] values and the trigger polarity.

Table 152. 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 21.4.2: 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).

21.4.17 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.

21.4.18 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 153*.

Table 153. 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

21.4.19 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.

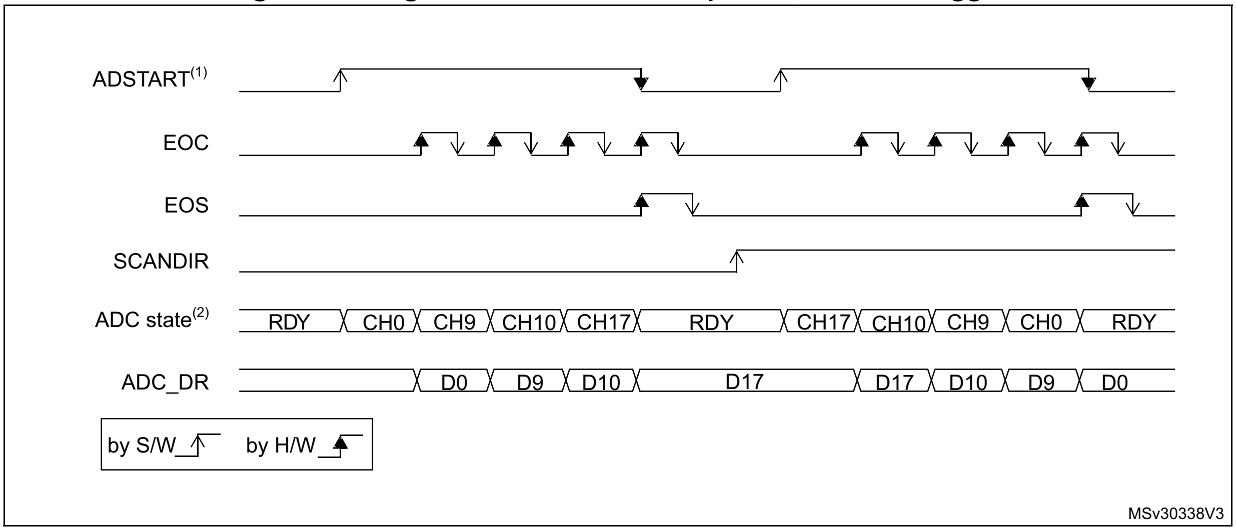
21.4.20 End of conversion sequence (EOS flag)

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.

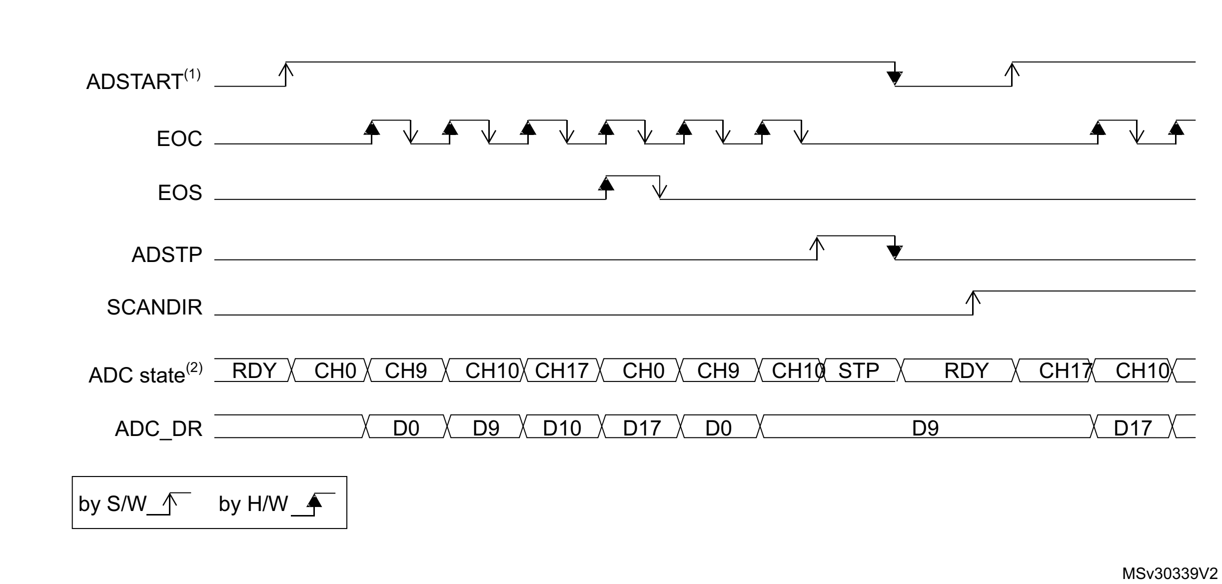
21.4.21 Example timing diagrams (single/continuous modes hardware/software triggers)

Figure 80. Single conversions of a sequence, software trigger



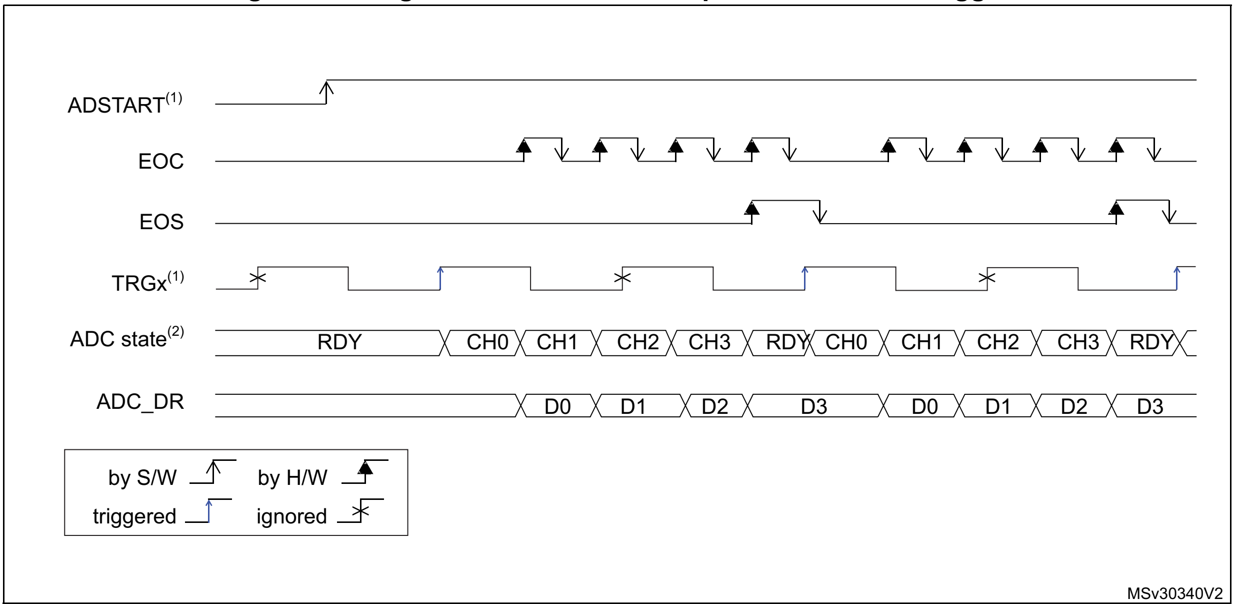
1. EXTEN = 00, CONT = 0.
2. CHSEL = 0x20601, WAIT = 0, AUTOFF = 0.

Figure 81. Continuous conversion of a sequence, software trigger



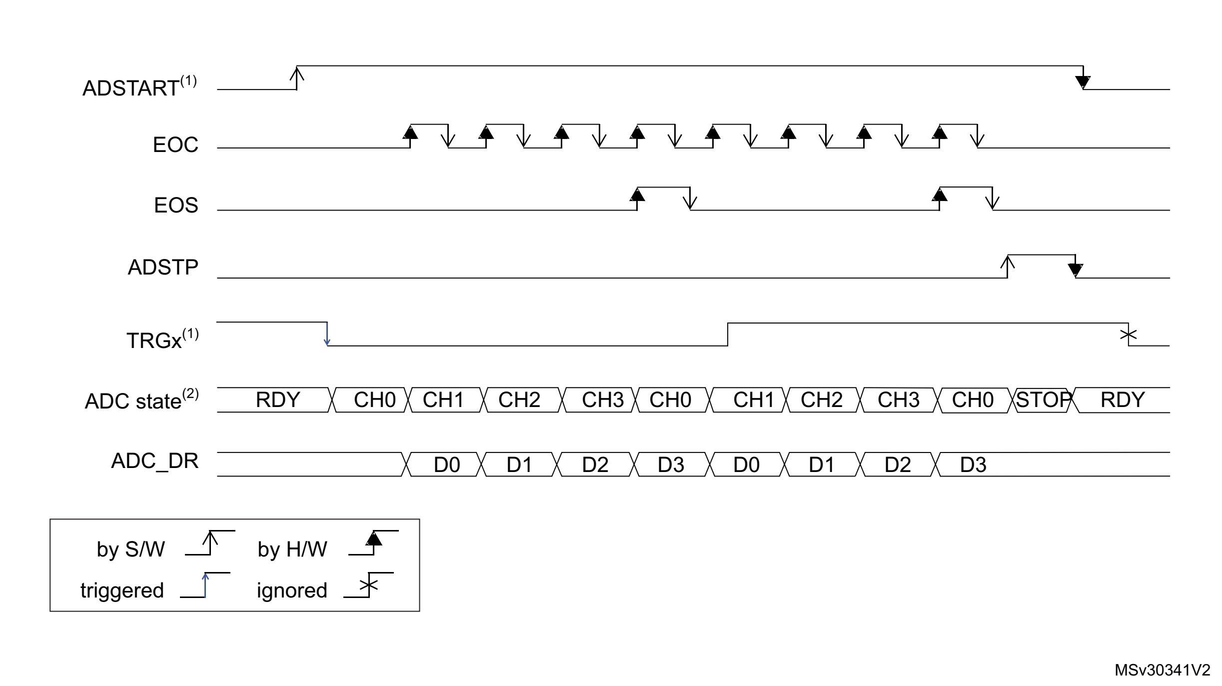
1. EXTEN = 00, CONT = 1.
2. CHSEL = 0x20601, WAIT = 0, AUTOFF = 0.

Figure 82. 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 83. 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.

21.4.22 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.

21.4.23 Data management

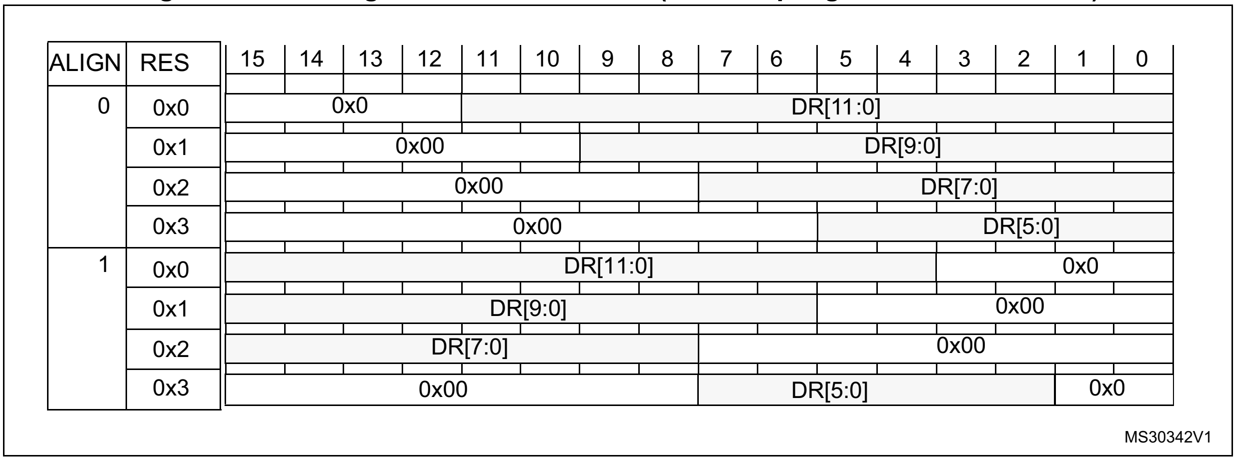
Data register and data alignment (ADC_DR, ALIGN)

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 84*.

Figure 84. 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.

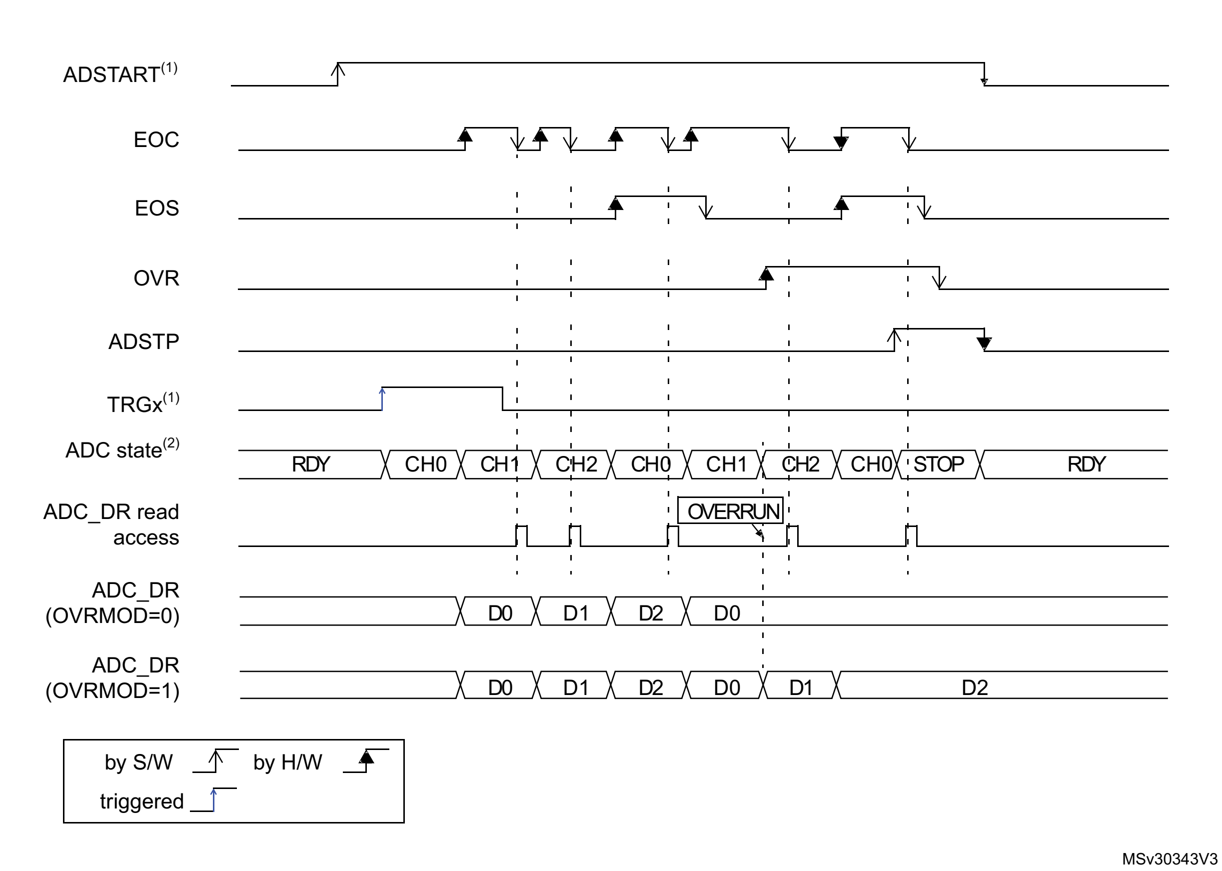
When an overrun condition occurs, the ADC keeps operating and can continue to convert unless the software decides to stop and reset the sequence by setting the ADSTP bit in the ADC_CR register.

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 85. 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 did 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 *ADC overrun (OVR, OVRMOD)*).

The DMA transfer requests are blocked until the software clears the OVR bit.

Two different DMA modes are proposed depending on the application use and are configured with bit DMACFG 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.

21.4.24 Low-power features

Wait conversion mode (WAIT)

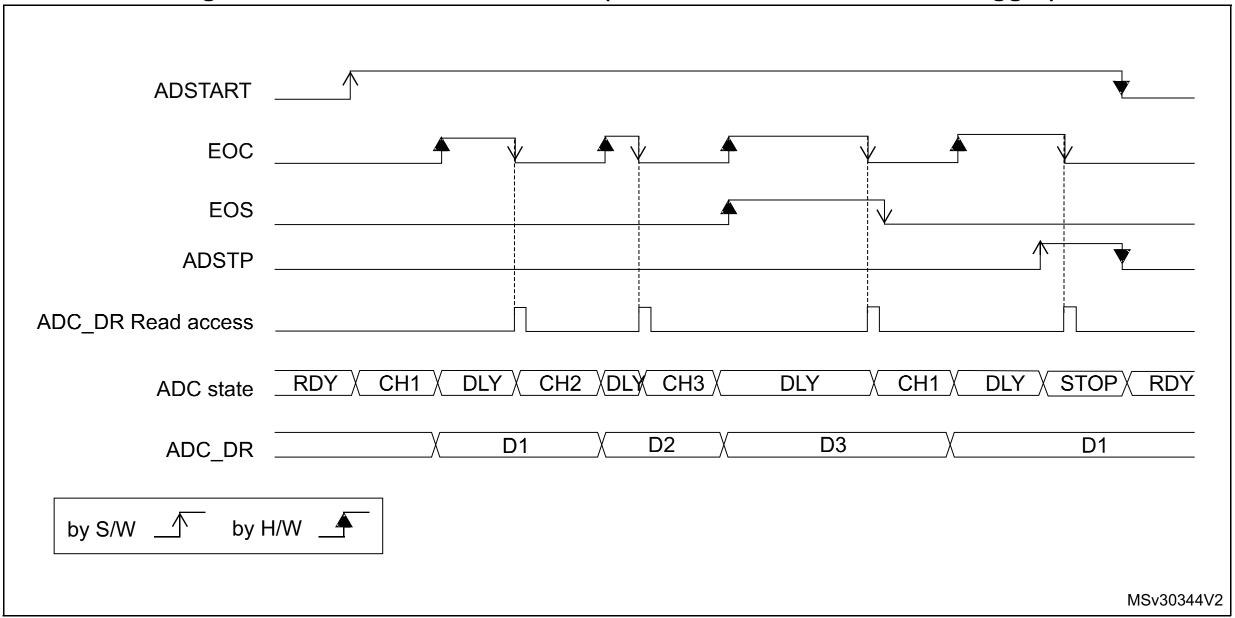
Wait conversion mode can be used to simplify the software as well as optimizing the performance of applications clocked at low frequency where there might be a risk of ADC overrun occurring.

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 86. 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 87* 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 88: ADC behavior with WAIT = 0 and AUTOFF = 1* and *Figure 89: ADC behavior with WAIT = 1 and AUTOFF = 1*).

The auto-off mode is compatible with the autonomous peripheral mode.

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 87. Auto-off mode state diagram

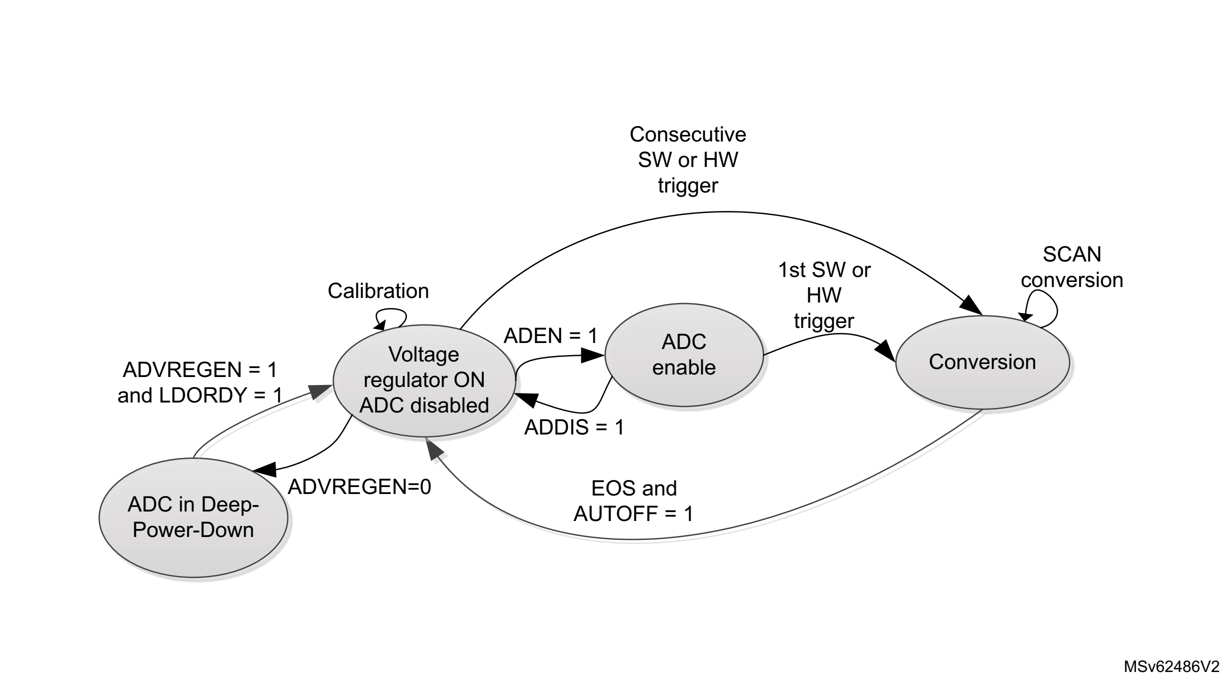
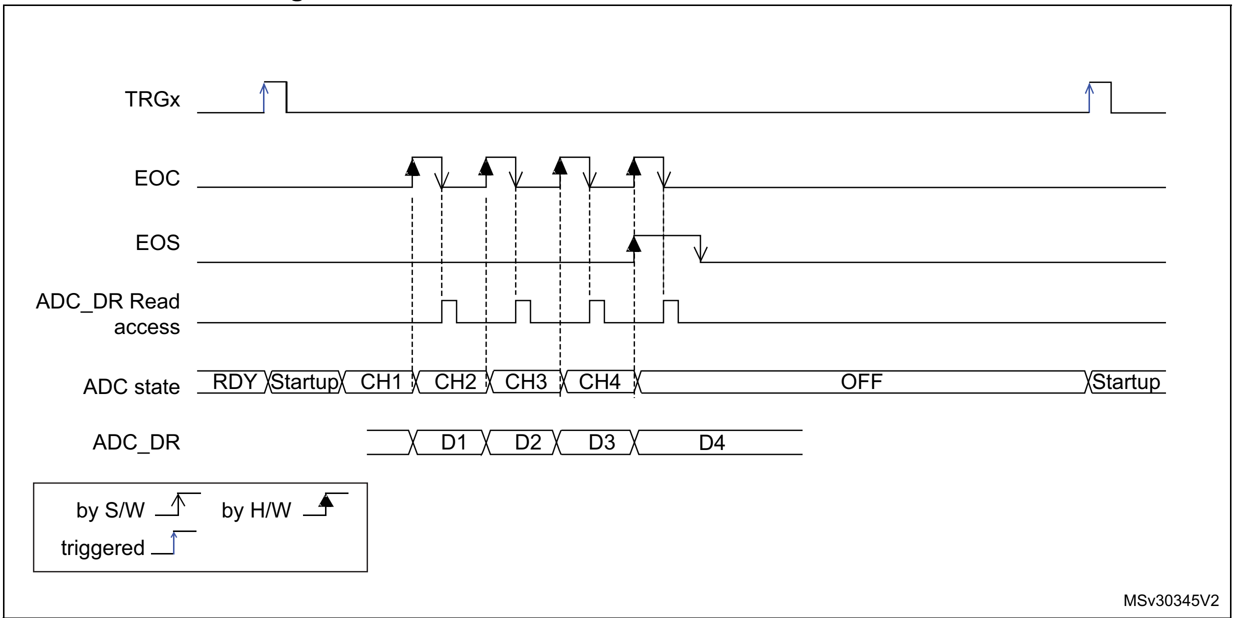
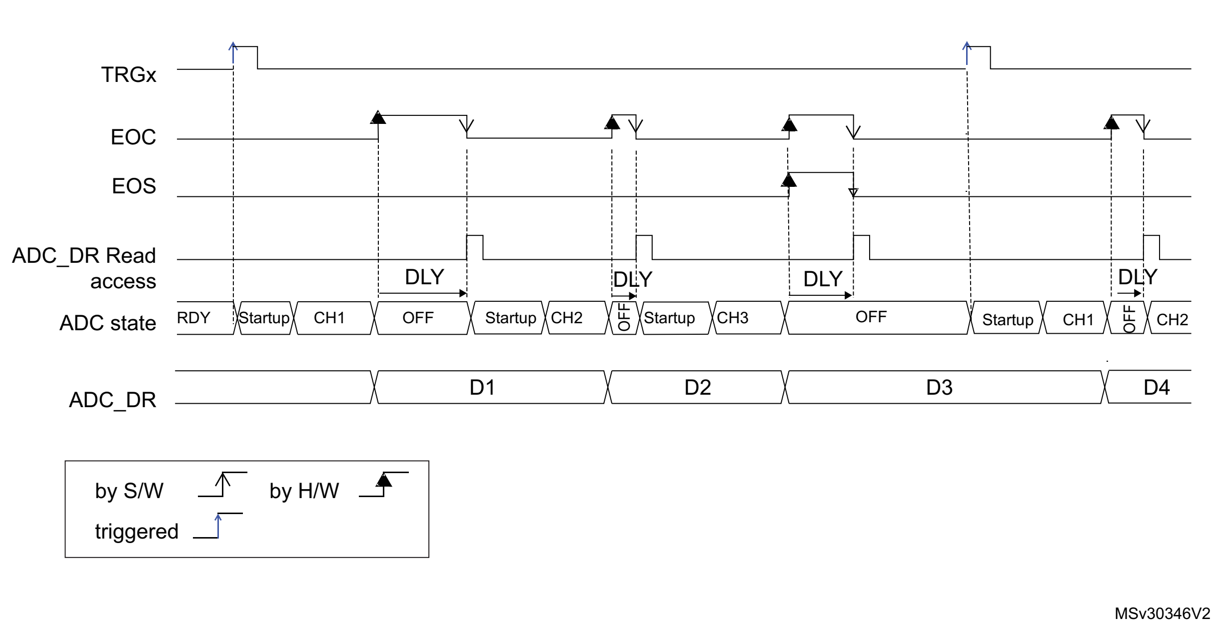


Figure 88. ADC behavior with WAIT = 0 and AUTOFF = 1



1. EXTSEL = TRGx, EXTEN = 01 (rising edge), CONT = x, ADSTART = 1, CHSEL = 0xF, SCANDIR = 0, WAIT = 0, AUTOFF = 1.

Figure 89. 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:

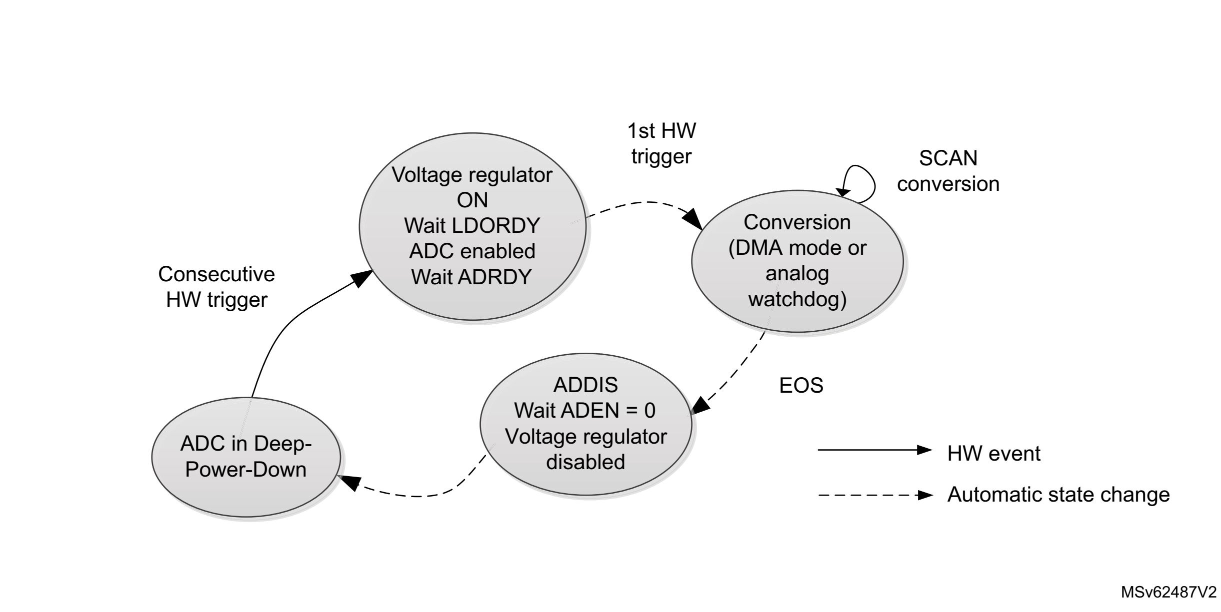
1. When AUTOFF and DPD are both set to 1, the ADC is powered off when no conversion is ongoing.
2. Upon hardware trigger reception, the ADC requests the adc_ker_ck and adc_hclk clocks to the RCC, the ADC voltage regulator is enabled, the calibration factor is loaded, the ADC is enabled and the conversion 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 90* 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 autonomous peripheral mode.

Figure 90. Autonomous mode state diagram



21.4.25 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 155: Analog watchdog 1 channel selection*) remain within a configured voltage range (window) as shown in *Figure 91*.

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 154 describes how the comparison is performed for all the possible resolutions.

Table 154. 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 155 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 91. Analog watchdog guarded area

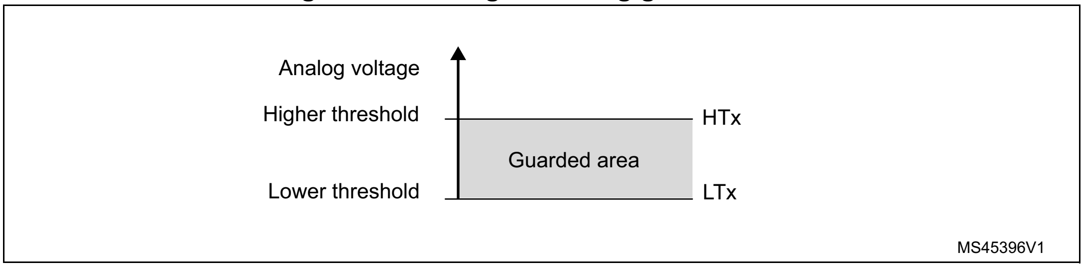


Table 155. Analog watchdog 1 channel selection

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 154 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 94*)

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 92. ADC_AWDx_OUT signal generation

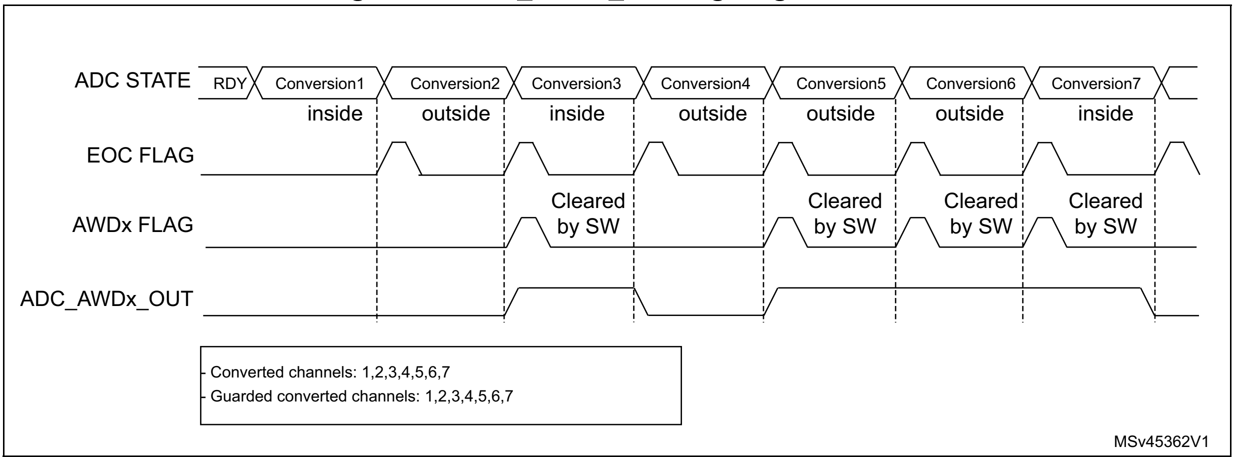


Figure 93. ADC_AWDx_OUT signal generation (AWDx flag not cleared by software)

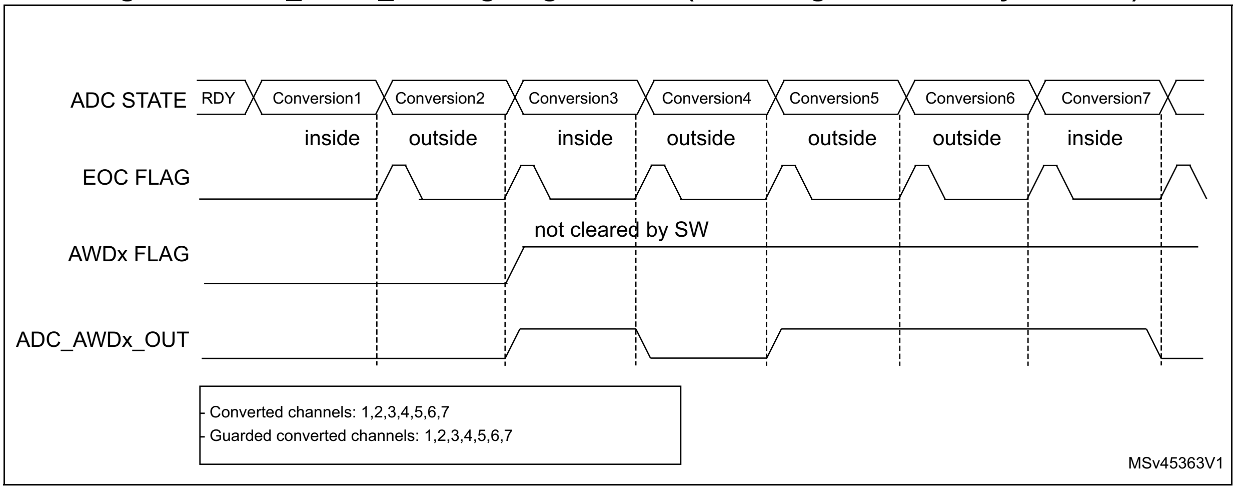
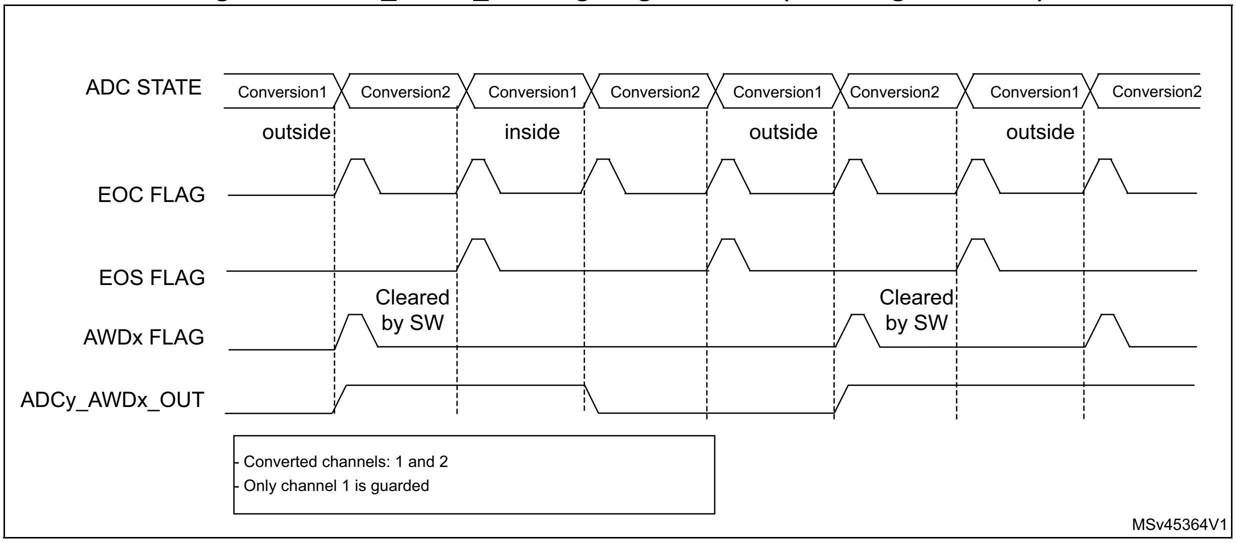


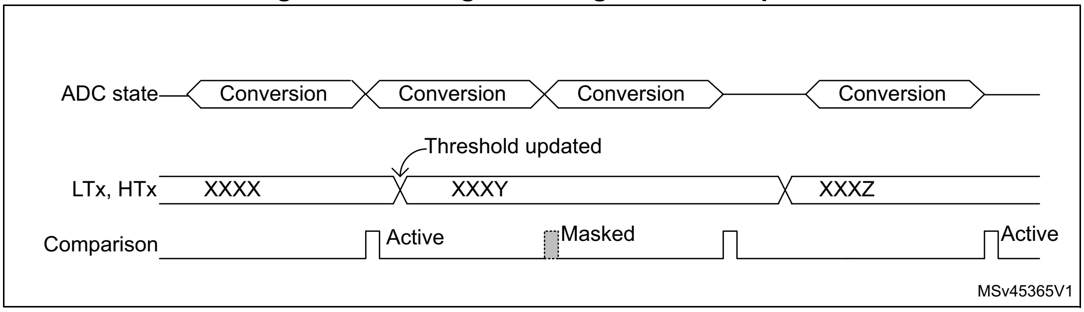
Figure 94. ADC_AWDx_OUT signal generation (on a single channel)



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 95. Analog watchdog threshold update



21.4.26 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:

$$\text{Result} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{Conversion}(t_n)$$

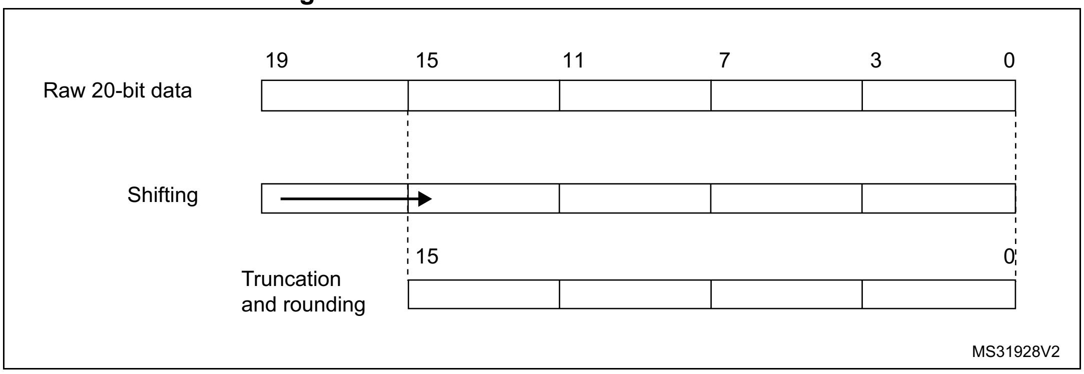
It allows the following functions to be performed by hardware: averaging, data rate reduction, SNR improvement, basic filtering.

The oversampling ratio N is defined using the OVSR[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.

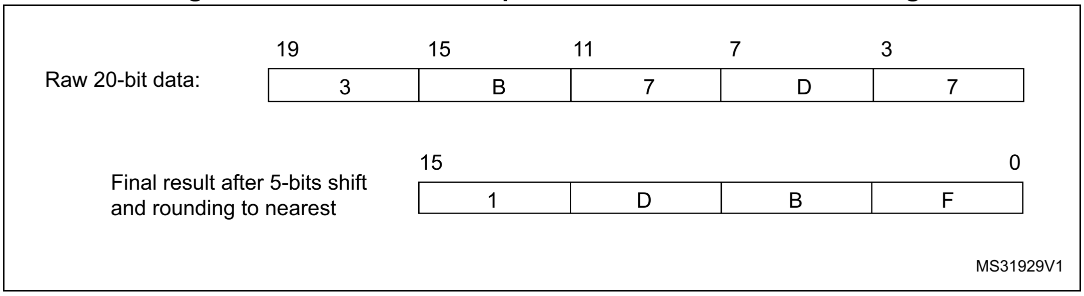
Note: If the intermediate result after the shifting exceeds 16 bits, the upper bits of the result are simply truncated.

Figure 96. 20-bit to 16-bit result truncation



The *Figure 97* gives a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

Figure 97. Numerical example with 5-bits shift and rounding



The *Table 156* below gives the data format for the various N and M combination, for a raw conversion data equal to 0xFFF.

Table 156. 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 oversampler 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, 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.

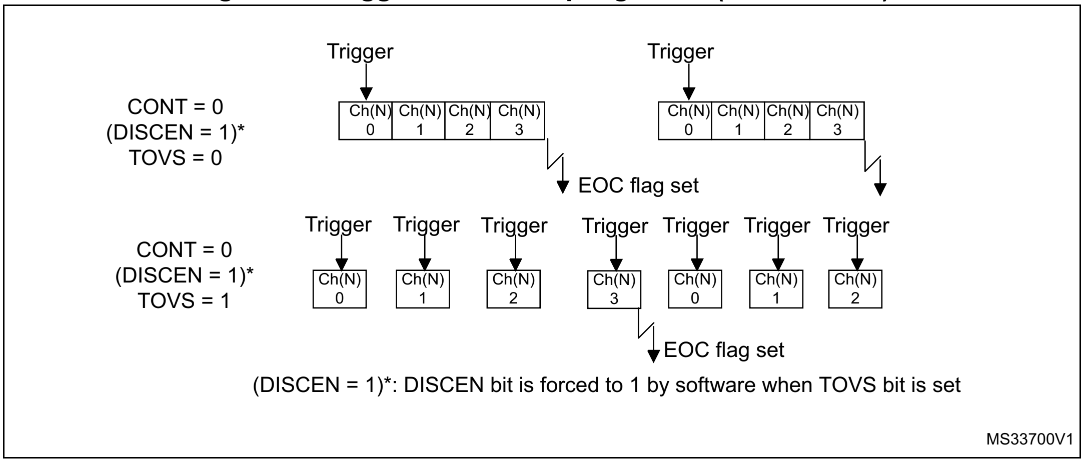
Triggered mode

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 98 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 98. Triggered oversampling mode (TOVS bit = 1)



21.4.27 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 21.4.2: ADC pins and internal signals* for details on the ADC internal input channel to which the above voltages are connected.

Figure 99 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).

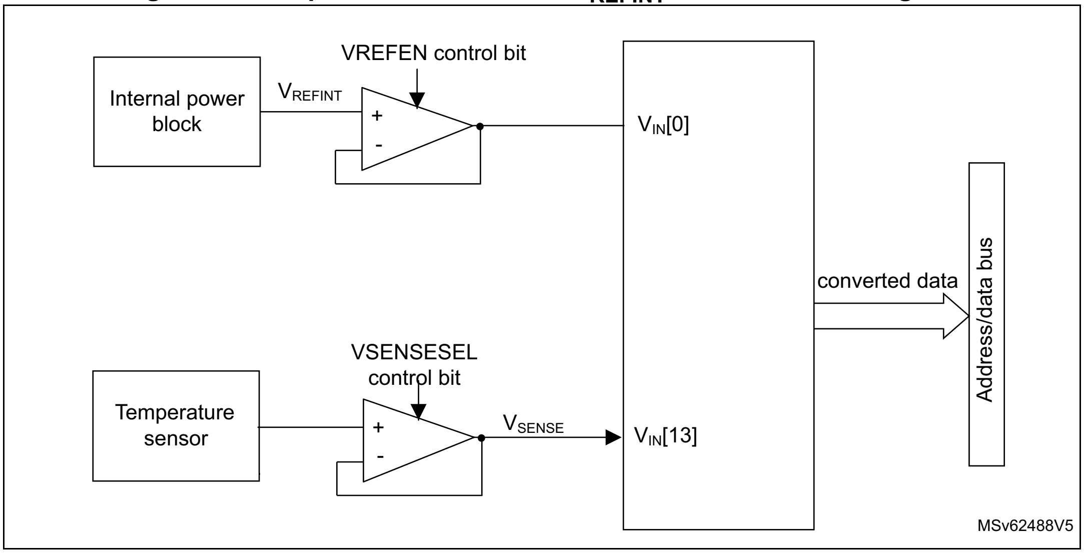
The uncalibrated internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperatures. To improve the accuracy of the temperature sensor measurement, calibration values are stored in system memory for each device by STMicroelectronics during 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.

Main features

- Linearity: ±2 °C max., precision depending on calibration

Figure 99. 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 21.4.2: 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 21.3: 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_{DDA} 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_{DDA} = 3.0 V can be used to evaluate the actual V_{DDA} voltage level.

The following formula gives the actual V_{DDA} voltage supplying the device:

$$V_{DDA} = 3.0\ V \times VREFINT\_CAL / VREFINT\_DATA$$

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_{DDA} 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_{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:

$$V_{CHANNELx} = \frac{V_{DDA}}{FULL\_SCALE} \times ADC\_DATA_x$$

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_{DDA} 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 21.3: 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.

21.5 ADC low-power modes

Table 157. 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.

21.6 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 158. ADC wake-up 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 wake-up from Stop mode feature. Refer to *Section 21.3: ADC implementation* for the list of supported Stop modes.

21.7 ADC registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

21.7.1 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

21.7.2 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).

21.7.3 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.	ADVR EGEN	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.	AD START	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)

21.7.4 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.	AWD1 EN	AWD1 SGL	CHSEL RMOD	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	SCAN DIR	RES[1:0]		DMAC FG	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

.....

01101: ADC analog input Channel 13 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 21.4.2: 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 84: Data alignment and resolution (oversampling disabled: OVSE = 0)*

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 CHSEL13)
1: Backward scan (from CHSEL13 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 *Managing converted data using the DMA*.

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 *Managing converted data using the DMA*.

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).

21.7.5 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).

21.7.6 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMPSEL13	SMPSEL12	SMPSEL11	SMPSEL10	SMPSEL9	SMPSEL8
										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 31:22 Reserved, must be kept at reset value.

Bits 21:8 **SMPSELx:** Channel-x sampling time selection (x = 13 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).

21.7.7 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 21.4.25: Analog window watchdog*.

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 21.4.25: Analog window watchdog*.

21.7.8 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 21.4.25: Analog window watchdog*.

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 21.4.25: Analog window watchdog*.

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

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:0 **CHSELx**: Channel x selection (x = 13 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).

21.7.10 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: Reserved

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).

21.7.11 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 21.4.25: Analog window watchdog*.

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 21.4.25: Analog window watchdog*.

21.7.12 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 84: Data alignment and resolution (oversampling disabled: OVSE = 0)*.

Just after a calibration is complete, DATA[6:0] contains the calibration factor.

21.7.13 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.	Res.	Res.	DPD	AUTOFF
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

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).

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

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:0 **AWD2CHx:** Analog watchdog channel selection (x = 13 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).

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

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:0 **AWD3CHx:** Analog watchdog channel selection (x = 13 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).

21.7.16 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).

21.7.17 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.	Res.	VSENSE SEL	VREF EN	PRESC[3:0]				Res.	Res.
								rw	rw	rw	rw	rw	rw		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 Reserved, must be kept at reset value.

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.

21.7.18 ADC register map

The following table summarizes the ADC registers.

Table 159. 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	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

Table 159. 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	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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
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
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.	Res.	Res.	DPD	AUTOFF
	Reset value																															0	0
0x48 - 0xBF	Reserved	Reserved																															
0xA0	ADC_AWD2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0xA4	ADC_AWD3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
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																															
0x308	ADC_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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																		

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

22 Comparator (COMP)

Available only on STM32WBA54/55xx devices.

22.1 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

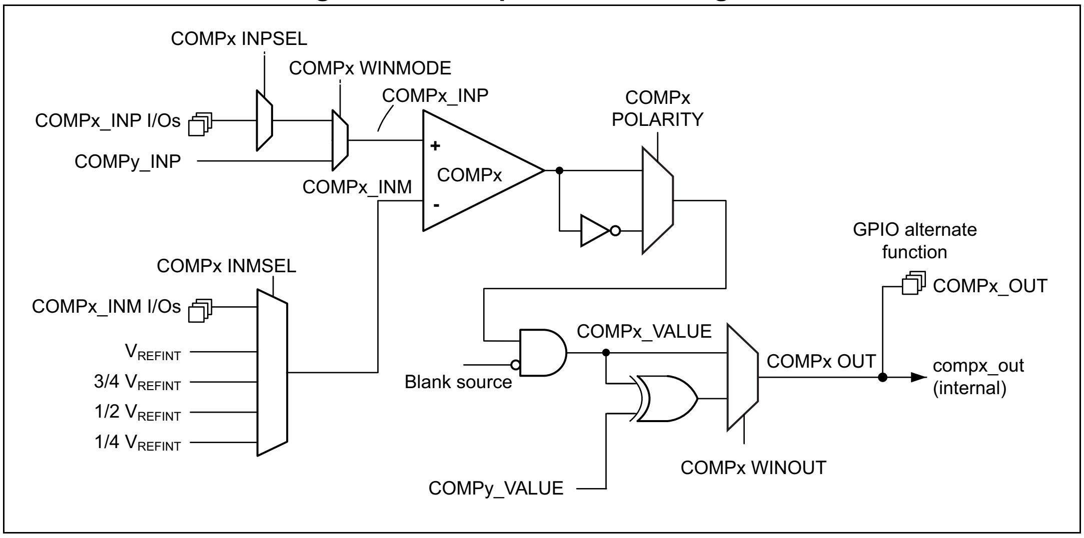
22.2 COMP main features

- Each comparator has configurable plus and minus inputs used for flexible voltage selection:
  - Multiplexed I/O pins
  - 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)

22.3 COMP functional description

22.3.1 COMP block diagram

Figure 100. Comparator block diagrams



22.3.2 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 160. COMP1 noninverting input assignment

COMP1_INP	COMP1_INPSEL[2:0]
COMP1_INP1	00
Open	01
Open	10
Open	11

Table 161. 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
Open	0100
Open	0101
COMP1_INM1	0110
Open	0111
Open	1000
Reserved	> 1000

Table 162. COMP2 noninverting input assignment

COMP2_INP	COMP2_INPSEL[1:0]
COMP2_INP1	00
Open	01
Open	10
Open	11

Table 163. 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
Open	0100
Open	0101
COMP2_INM1	0110
Open	0111
Open	1000
Reserved	> 1000

Table 164. COMP1 output-blanking PWM assignment

PWM output	COMP1_BLANKSEL[4:0]
None (no blanking)	00000
tim1_oc5	xxxx1
tim2_oc3	xxx1x
tim3_oc3	xx1xx
Reserved	Others

Table 165. COMP2 output-blanking PWM assignment

PWM output	COMP2_BLANKSEL[4:0]
None (no blanking)	00000
tim3_oc4	xxxx1
Reserved	Others

22.3.3 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 be reset only by an MCU reset.

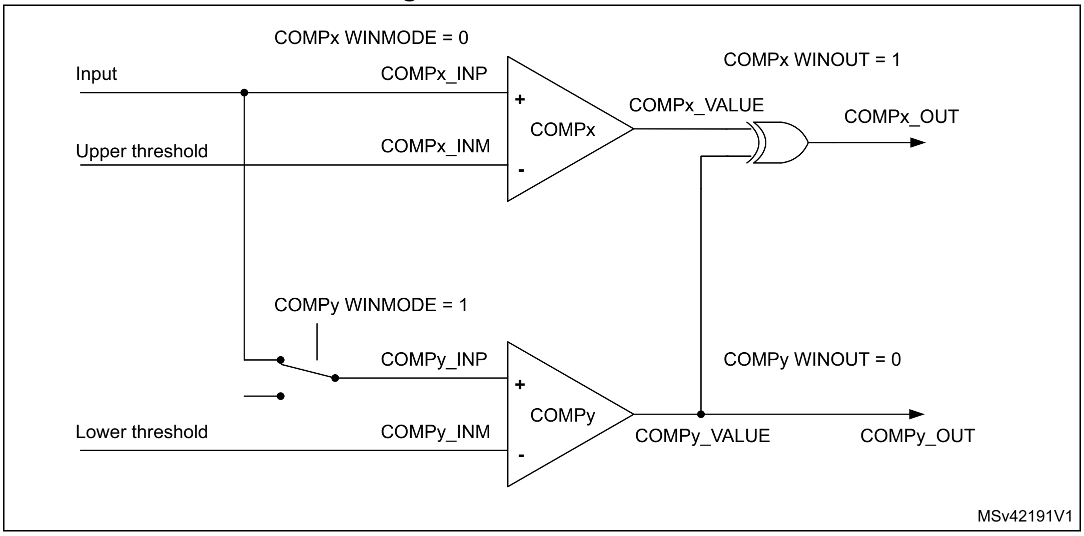
22.3.4 Window comparator

The purpose of the window comparator is to monitor if the analog voltage is within the range defined by the lower and upper thresholds.

The two embedded comparators can be used to create a window comparator. The monitored analog voltage is connected to the noninverting (plus) inputs of the two comparators. The upper and lower threshold voltages are connected to the inverting (minus) inputs of the comparators.

Two noninverting inputs can be connected internally by enabling the WINMODE bit to save one IO for other purposes.

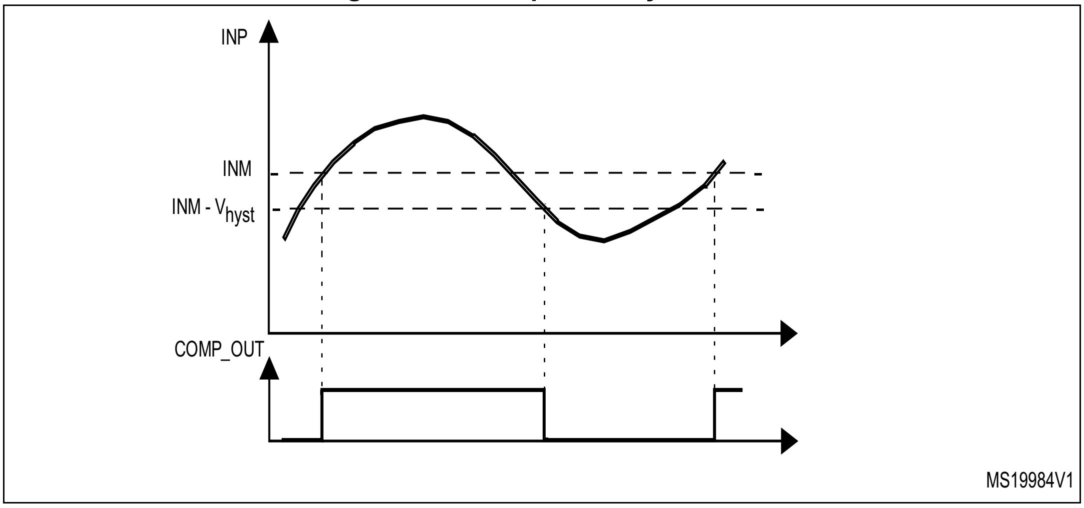
Figure 101. Window mode



22.3.5 Hysteresis

The comparator includes a programmable hysteresis to avoid spurious output transitions in case of noisy signals. The hysteresis can be disabled if not needed (for instance when exiting a low-power mode), to be able to force the hysteresis value using external components.

Figure 102. Comparator hysteresis

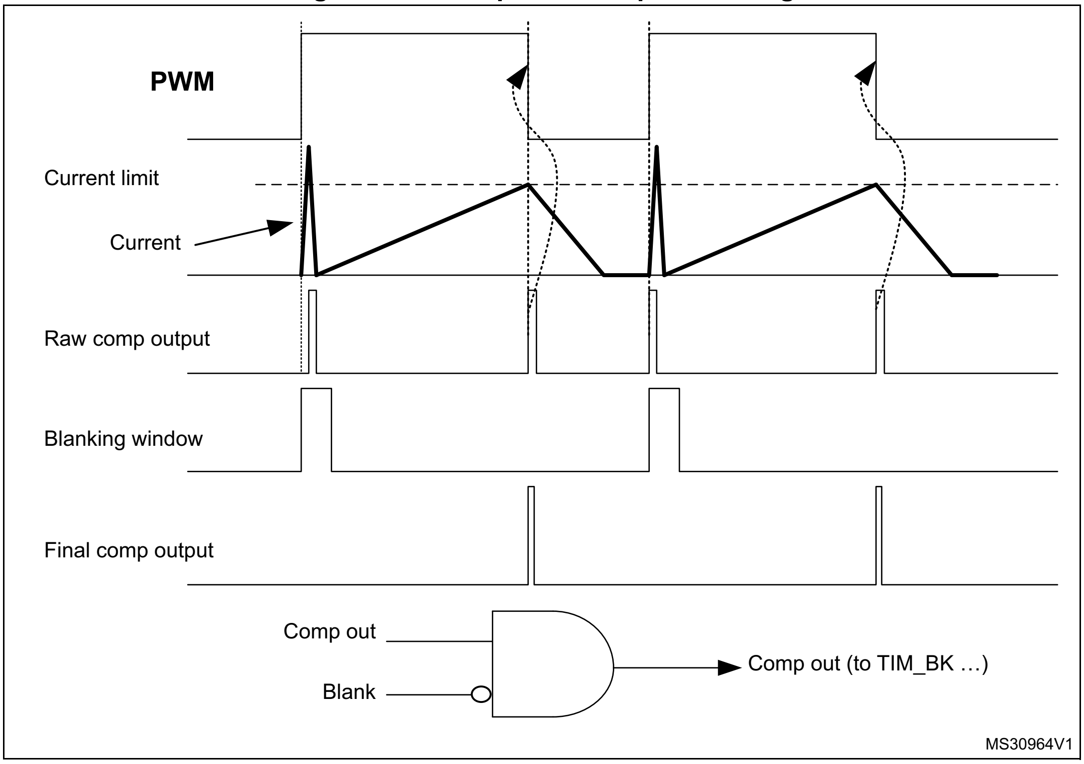


22.3.6 Comparator output-blanking function

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 antiparallel 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 AND-ed with the comparator output to provide the wanted comparator output (see the example in *Figure 103*).

Figure 103. Comparator output blanking



22.3.7 COMP power and speed modes

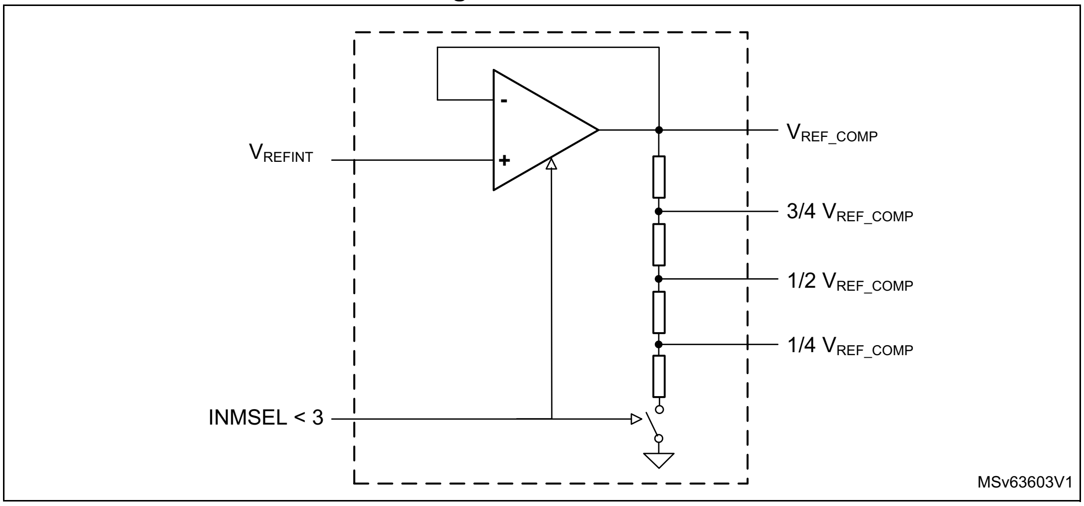
COMP1 and COMP2 power consumption versus propagation delay can be adjusted to have the optimum trade-off for a given application.

22.3.8 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 divided voltage is not used, the resistor bridge and the amplifier are disabled to reduce power 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 104. Scaler



22.4 COMP low-power modes

Table 166. 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	COMP registers are powered down and must be reinitialized after exiting Standby mode.

22.5 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 19: 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 167. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Exit Sleep mode	Exit Stop modes	Exit Standby mode
COMP1 output	In EXTI	Through EXTI	Yes	Yes	No
COMP2 output	In EXTI	Through EXTI	Yes	Yes	No

22.6 COMP registers

22.6.1 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.	WIN MODE	INPSEL[2:0]			INMSEL[3:0]				Res.	Res.	Res.	EN
rw	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 103*).

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 blanking source:
00000: None (no blanking)
xxxx1: tim1_oc5
xxx1x: tim2_oc3
xx1xx: tim3_oc3
Others: Reserved

Bits 19:18 **PWRMODE[1:0]:** COMP1 power mode selector

Controlled by software (if not locked), selects the power consumption and, as a consequence, the speed of the COMP1.
00: High speed
01: Intermediate speed and power
10: Medium speed and power
11: Ultra-low-power

Bits 17:16 **HYST[1:0]:** COMP1 hysteresis selector
Controlled by software (if not locked), selects the COMP1 hysteresis.
00: None
01: Low hysteresis
10: Medium hysteresis
11: High hysteresis

Bit 15 **POLARITY:** COMP1 polarity selector
Controlled by software (if not locked), selects the COMP1 output polarity.
0: Noninverted
1: Inverted

Bit 14 **WINOUT:** COMP1 output selector
Controlled by software (if not locked), selects the COMP1 output.
0: COMP1_VALUE
1: COMP1_VALUE XOR COMP2_VALUE (required for window mode, see *Figure 101*)

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WINMODE**: COMP1 noninverting input selector for window mode
Controlled by software (if not locked), selects the signal for the 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 101*)

Bits 10:8 **INPSEL[2:0]**: COMP1 signal selector for noninverting input
Controlled by software (if not locked), selects the signal for the noninverting input COMP1_INP (see *Table 160* for the assignment).

Bits 7:4 **INMSEL[3:0]**: COMP1 signal selector for inverting input INM
Controlled by software (if not locked), selects the signal for the inverting input COMP1_INM (see *Table 161* for the assignment).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **EN:** COMP1 enable
Controlled by software (if not locked), enables COMP1.
0: COMP1 disabled
1: COMP1 enabled

22.6.2 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	WIN OUT	Res.	Res.	WIN MODE	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 103*).

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

Bits 24:20 **BLANKSEL[4:0]:** COMP2 blanking source selector
Controlled by software (if not locked) and selects the blanking source:
00000: None (no blanking)
xxxx1: tim3_oc4
Others: Reserved

Bits 19:18 **PWRMODE[1:0]:** COMP2 power mode selector
Controlled by software (if not locked), selects the power consumption and, as a consequence, the speed of the COMP2.
00: High speed
01: Intermediate speed and power
10: Medium speed and power
11: Ultra-low-power

Bits 17:16 **HYST[1:0]:** COMP2 hysteresis selector
Controlled by software (if not locked), selects the COMP2 hysteresis.
00: None
01: Low hysteresis
10: Medium hysteresis
11: High hysteresis

Bit 15 **POLARITY:** COMP2 polarity selector
Controlled by software (if not locked), selects the COMP2 output polarity.
0: Noninverted
1: Inverted

Bit 14 **WINOUT:** COMP2 output selector
Controlled by software (if not locked), selects the COMP2 output.
0: COMP2_VALUE
1: COMP1_VALUE XOR COMP2_VALUE (required for window mode, see *Figure 101*)

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WINMODE**: COMP2 noninverting input selector for window mode
Controlled by software (if not locked), selects the signal for the 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 101*)

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **INPSEL[1:0]**: COMP2 signal selector for noninverting input
Controlled by software (if not locked), selects the signal for the noninverting input COMP2_INP (see *Table 162* for the assignment).

Bits 7:4 **INMSEL[3:0]**: COMP2 signal selector for inverting input INM
Controlled by software (if not locked), selects the signal for the inverting input COMP2_INM (see *Table 163* for the assignment).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **EN:** COMP2 enable
Controlled by software (if not locked), enables COMP2.
0: COMP2 disabled
1: COMP2 enabled

22.6.3 COMP register map

Table 168. 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	INPSEL [2:0]			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				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.

23 Touch sensing controller (TSC)

23.1 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.

23.2 TSC main features

The touch sensing controller has the following main features:

- Proven and robust surface charge transfer acquisition principle
- Supports up to 15 capacitive sensing channels without shield
- Supports up to 12 capacitive sensing channels when shield is used in group 2 (recommended mode)
- 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

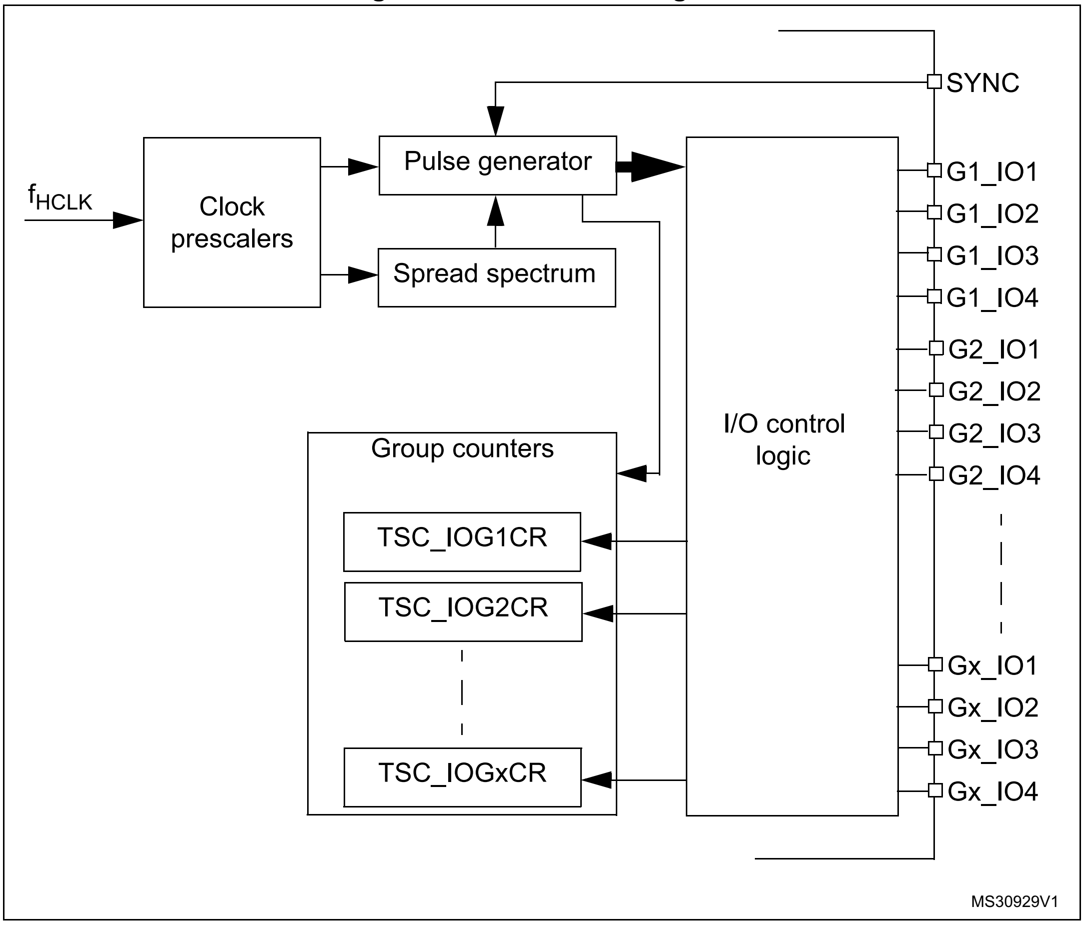
Note: The number of capacitive sensing channels is dependent on the size of the packages and subject to I/O availability.

23.3 TSC functional description

23.3.1 TSC block diagram

The block diagram of the touch sensing controller is shown in *Figure 105*.

Figure 105. TSC block diagram



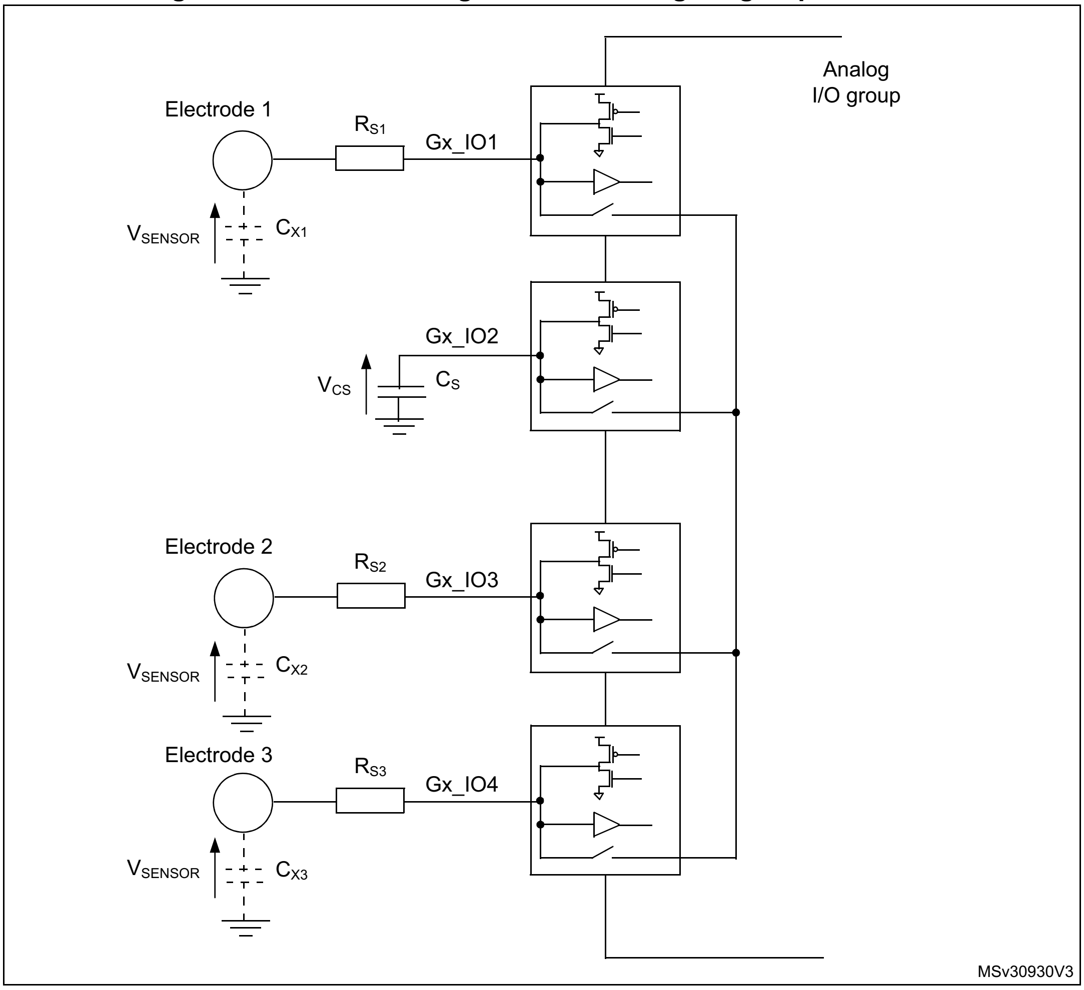
23.3.2 Surface charge transfer acquisition overview

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 106*). 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 106. 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 169 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.

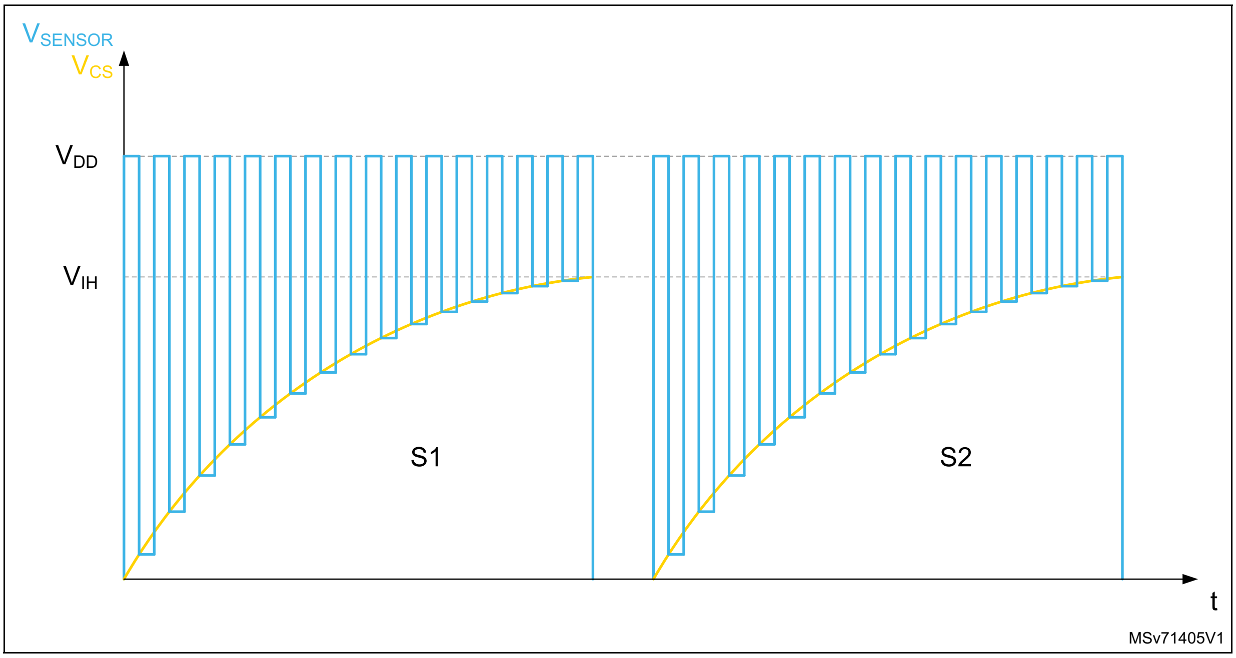
Table 169. Acquisition sequence summary

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 (refer to *Figure 106* for V_{SENSOR} and V_{CS} definition):

Figure 107. Sampling capacitor voltage variation



23.3.3 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:

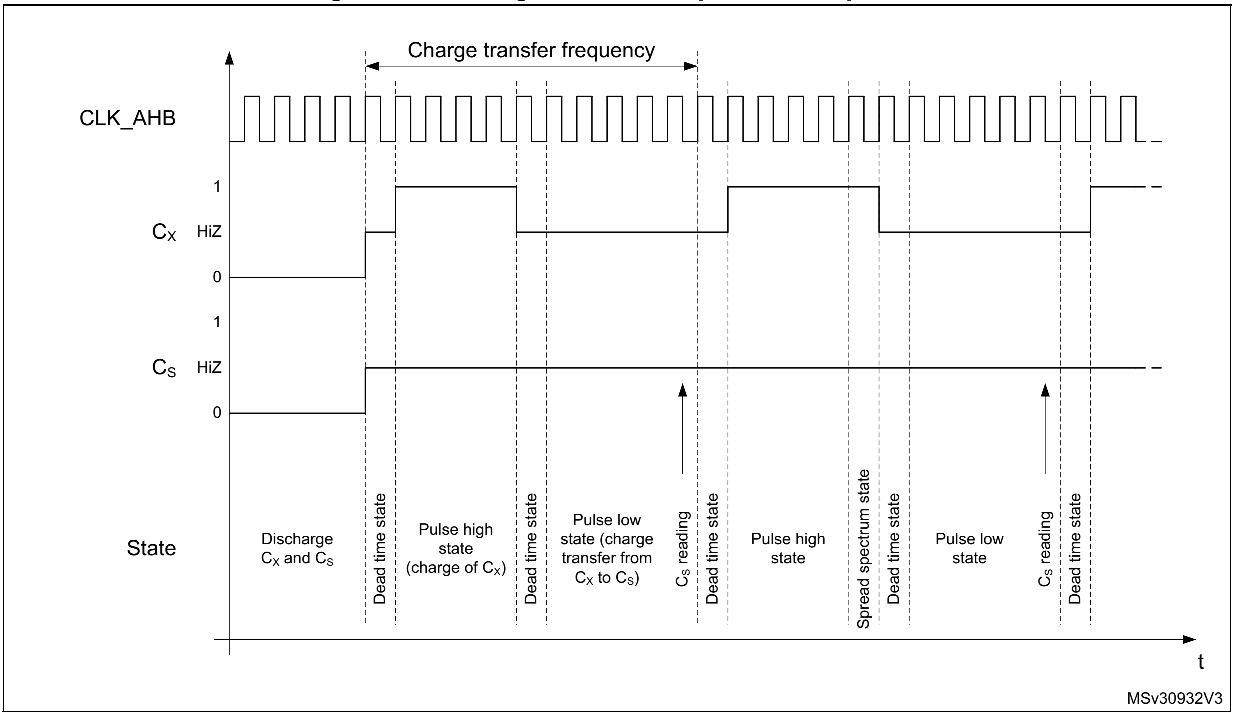
- The pulse generator clock (PGCLK) is defined using the PGPSC[2:0] bits of the TSC_CR register
- The spread spectrum clock (SSCLK) is defined using the SSPSC bit of the TSC_CR register

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 12: Reset and clock control (RCC)*.

23.3.4 Charge transfer acquisition sequence

An example of a charge transfer acquisition sequence is detailed in *Figure 108*.

Figure 108. Charge transfer acquisition sequence



For higher flexibility, the charge transfer frequency is fully configurable. Both the pulse high state (charge of C_X) and the pulse low state (transfer of charge from C_X to C_S) duration can be defined using the CTPH[3:0] and CTPL[3:0] bits in the TSC_CR register. The standard range for the pulse high and low states duration is 500 ns to 2 µs. To ensure a correct measurement of the electrode capacitance, the pulse high state duration must be set to ensure that C_X is always fully charged.

A dead time where both the sampling capacitor I/O and the channel I/O are in input floating state is inserted between the pulse high and low states to ensure an optimum charge transfer acquisition sequence. This state duration is 1 period 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.

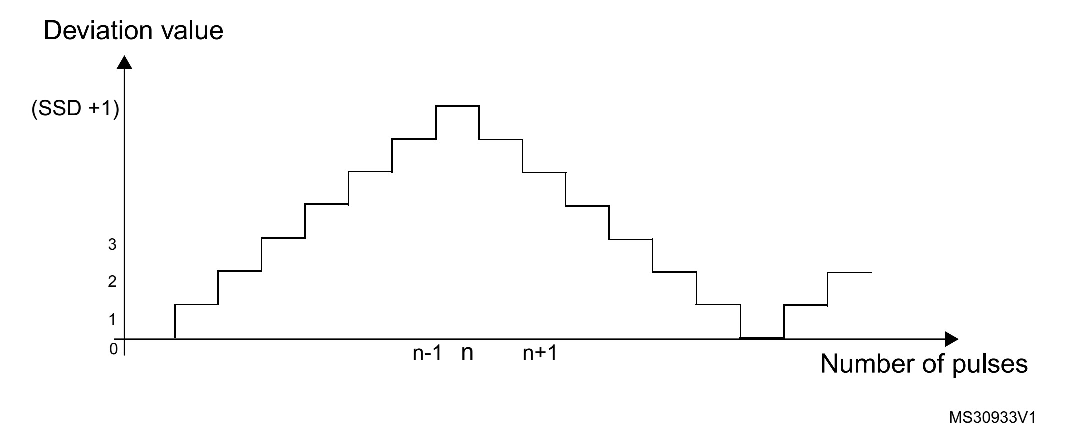
Note: The following TSC control register configurations are forbidden:
- *bits PGPSC are set to 000 and bits CTPL are set to 0001*
- *bits PGPSC are set to 001 and bits CTPL are set to 0000*

23.3.5 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 109. Spread spectrum variation principle



The table below details the maximum frequency deviation with different HCLK settings:

Table 170. 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.

23.3.6 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.

23.3.7 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 171. 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.

23.3.8 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.

23.3.9 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.

23.4 TSC low-power modes

Table 172. 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.

23.5 TSC interrupts

Table 173. 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

23.6 TSC 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).

23.6.1 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 23.3.4: 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 23.3.4: 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.

23.6.2 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

23.6.3 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

23.6.4 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

23.6.5 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
								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 31:24 Reserved, must be kept at reset value.

Bits 23:0 **Gx_IOy**: Gx_IOy Schmitt trigger hysteresis mode

These bits are set and cleared by software to enable/disable the Gx_IOy Schmitt trigger hysteresis.

0: Gx_IOy Schmitt trigger hysteresis disabled

1: Gx_IOy Schmitt trigger hysteresis enabled

Note: These bits control the I/O Schmitt trigger hysteresis whatever the I/O control mode is (even if controlled by standard GPIO registers).

23.6.6 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
								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 31:24 Reserved, must be kept at reset value.

Bits 23:0 **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).

23.6.7 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
								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 31:24 Reserved, must be kept at reset value.

Bits 23:0 **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.

23.6.8 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1
								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 31:24 Reserved, must be kept at reset value.

Bits 23:0 **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.

23.6.9 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.	Res.	Res.	G6S	G5S	G4S	G3S	G2S	G1S
										r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G6E	G5E	G4E	G3E	G2E	G1E
										rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **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 15:6 Reserved, must be kept at reset value.

Bits 5:0 **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

23.6.10 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 6)

Reset value: 0x0000 0000

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

23.6.11 TSC register map

Table 174. TSC 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	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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x0014	Reserved																																
0x0018	TSC_IOASCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x001C	Reserved																																
0x0020	TSC_IOSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x0024	Reserved																																
0x0028	TSC_IOCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x002C	Reserved																																
0x0030	TSC_IOGCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G6S	G5S	G4S	G3S	G2S	G1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G6E	G5E	G4E	G3E	G2E	G1E
	Reset value											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 174. TSC 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
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

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

24 True random number generator (RNG)

24.1 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 nondeterministic random bit generator (NDRBG).

The RNG true random number generator can be certified NIST SP800-90B. It can also be tested using the German BSI statistical tests of AIS-31 (T0 to T8).

24.2 RNG main features

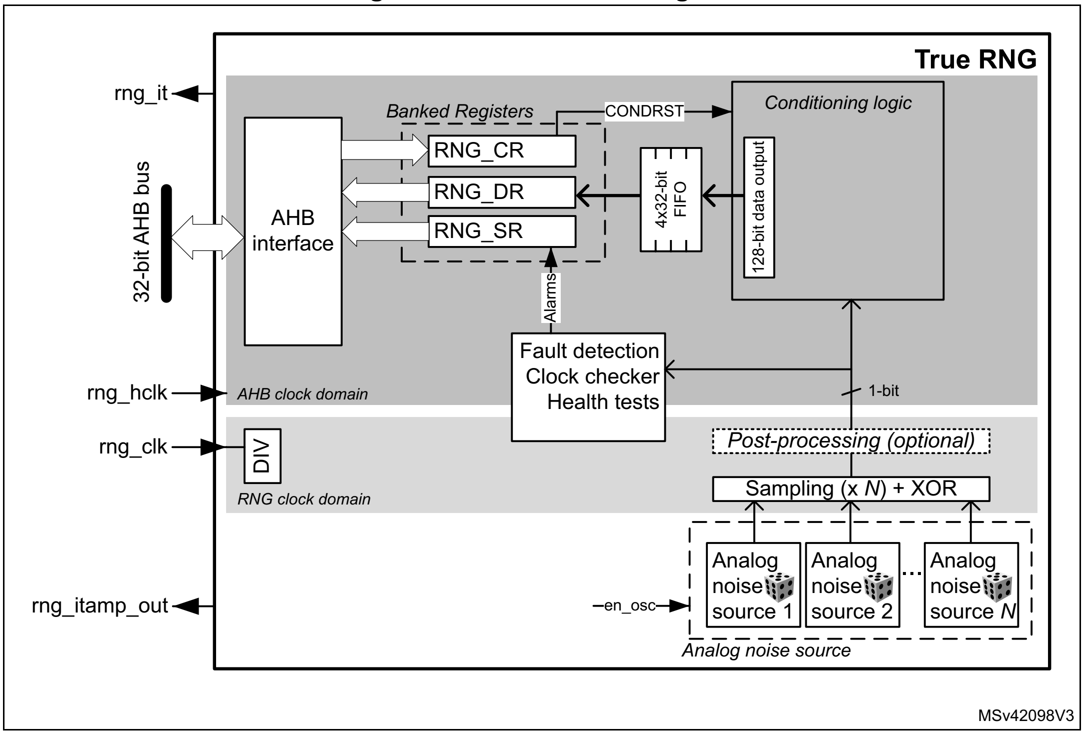
- The RNG delivers 32-bit true random numbers, produced by an analog entropy source conditioned by a NIST SP800-90B approved conditioning stage.
- It can be used as the entropy source to construct a nondeterministic random bit generator (NDRBG).
- In the NIST configuration, it produces four 32-bit random samples every 412 AHB clock cycles if $f_{AHB} < f_{threshold}$ (256 RNG clock cycles otherwise).
- It embeds startup 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).

24.3 RNG functional description

24.3.1 RNG block diagram

Figure 110 shows the RNG block diagram.

Figure 110. RNG block diagram



24.3.2 RNG internal signals

Table 175 describes a list of useful-to-know internal signals available at the RNG level, not at the STM32 product level (on pads).

Table 175. RNG internal input/output signals

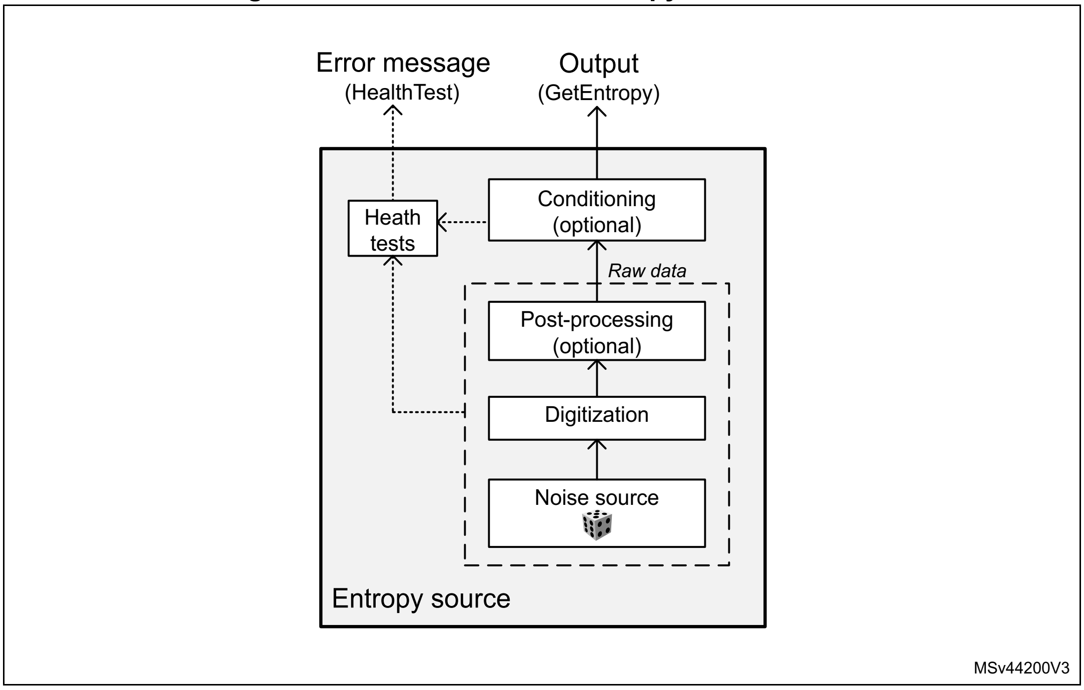
Signal name	Signal type	Description
rng_it	Digital output	RNG global interrupt request
rng_hclk	Digital input	AHB clock
rng_clk	Digital input	RNG dedicated clock, asynchronous to rng_hclk
rng_itamp_out	digital output	RNG internal tamper event signal to TAMP (XORed), triggered when an unexpected hardware fault occurs. When this signal is triggered, RNG stops delivering random samples, requiring a reset and a new initialization to be usable again.

24.3.3 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 111*. 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 111. 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 24.3.8*. Multiple oscillators are also disabled for the configuration A (see *Table 178: RNG configurations*).
- 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 24.6 the recommended RNG clock frequencies and associated divider value are given.

Post processing

In the NIST configuration no post-processing is applied to the sampled noise source. In non-NIST configuration B (as defined in *Section 24.6.2*) 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 the 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 24.5*.

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 24.5*.

Whenever a random number is available through the RNG_DR register, the DRDY flag changes from 0 to 1. This flag remains high until the 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 24.6.2*).

1. Startup health tests, performed after reset and before the first use of the RNG as entropy source:
   - Repetition count test, flagging an error when the noise source has provided more than 42 consecutive bits at a constant value (0 or 1).
   - Adaptive proportion test running on a window of 1024 consecutive bits: the RNG verifies that the first bit on the outputs of the noise source is not repeated more than 711 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 startup tests.
   - Adaptive proportion test, similar to the one running in startup 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 rerunning the startup tests (see software reset sequence in *Section 24.3.4*). 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 24.3.7*.

Note: An interrupt can be generated when an error is detected.

Above the health test thresholds are modified by changing the value in the RNG_HTCR register. See Section 24.6 for details.

24.3.4 RNG initialization

The RNG simplified state machine is pictured on *Figure 112*.

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 the analog output and filling the 128-bit conditioning shift register.
2. The conditioning hardware initializes, automatically triggering startup behavior test on the raw data samples and known-answer tests.
3. When startup 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 24.5*.

Figure 112. RNG initialization overview

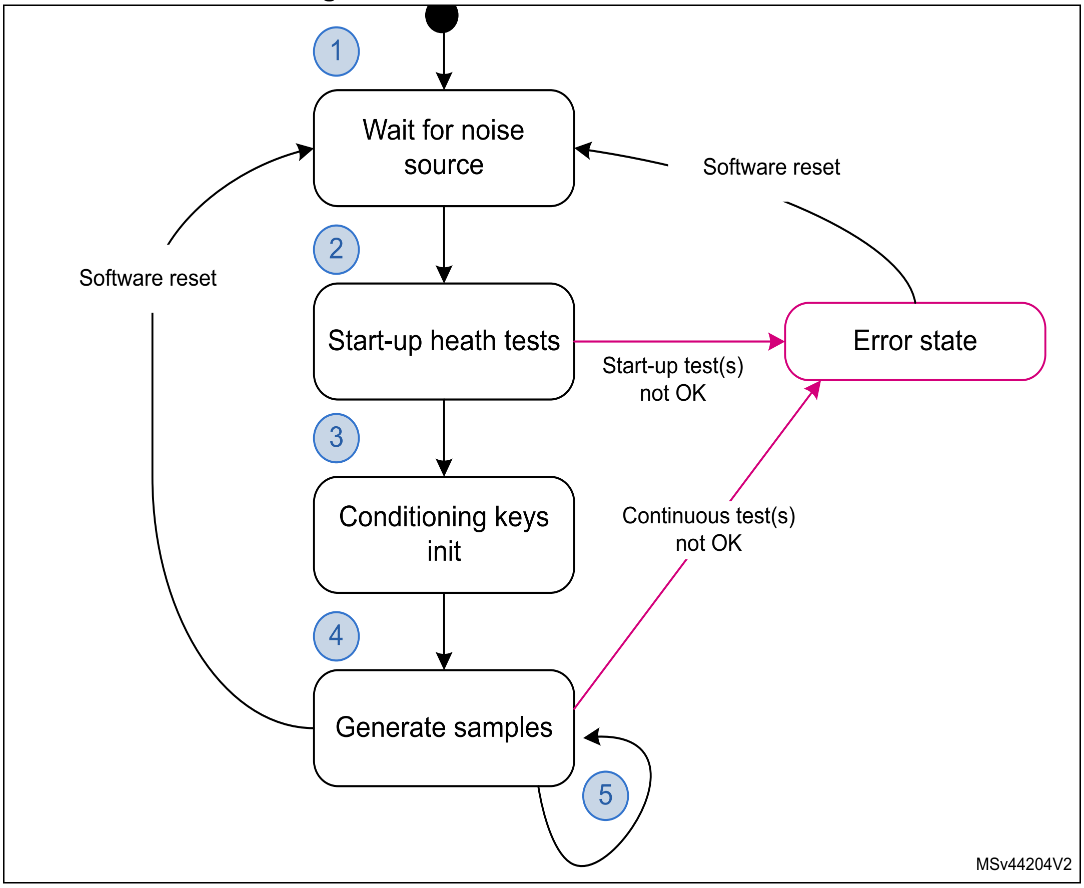


Figure 112 also highlights a possible software reset sequence, implemented by:

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 the RNG peripheral is reset through RCC (hardware reset), the RNG configuration for optimal randomness is lost in the RNG registers. Software reset with CONFIGLOCK set preserves the RNG configuration.

24.3.5 RNG operation

Normal operation

To run the RNG using interrupts, the following steps are recommended:

1. Check *Section 24.6* to verify if a specific RNG configuration is required for the application.
   – In this 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 the 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 the above conditions are false, the RNG_DR register must not be read. If an error occurred, the error recovery sequence described in *Section 24.3.7* must be used.

To run the RNG in polling mode following steps are recommended:
1. Check *Section 24.6* to 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 the above conditions are false, the RNG_DR register must not be read. If an error occurred, the error recovery sequence described in *Section 24.3.7* 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 24.6*. The gain in random number generation speed is summarized in *Section 24.5*.

Low-power operation

If the power consumption is a concern to the application, low-power strategies can be used, as described in *Section 24.3.8*.

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 24.3.3: Random number generation*.

24.3.6 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 24.6: RNG entropy source validation*.

Note: When the CED bit in the RNG_CR register is set to 0, the RNG clock frequency before the internal divider ***must be higher*** *than the AHB clock frequency divided by 32, otherwise the clock checker always flags a clock error (CECS = 1 in the RNG_SR register).*

See *Section 24.3.1: RNG block diagram* for details (AHB and RNG clock domains).

24.3.7 Error management

In parallel to random number generation a health check block verifies the correct noise source behavior and the frequency of the RNG source clock as detailed in this section. Associated error state is also described.

Clock error detection

When the clock error detection is enabled (CED = 0) and if the RNG clock frequency is too low, the RNG sets to 1 both the CEIS and CECS bits to indicate that a clock error occurred. In this case, the application must check that the RNG clock is configured correctly (see *Section 24.3.6: RNG clocking*) and then it must clear the CEIS bit interrupt flag. The CECS bit is automatically cleared when the clocking condition is normal.

Note: The clock error has no impact on generated random numbers that is the application can still read the 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.

When ARDIS is cleared (auto-reset enabled) clear the SEIS interrupt status bit in the RNG_SR register before drawing random numbers again.

When ARDIS is set (auto-reset disabled) the following sequence must be used to fully recover from a seed error:

1. Write CONDRST at 1 and at 0 in the RNG_CR register to reset the conditioning logic.
2. Wait for the CONDRST bit to be cleared by hardware. Then clear the SEIS interrupt status bit in the RNG_SR register.
3. If SECS is still set in RNG_SR it means a new seed error occurred (unlikely). In this case go back to step 1).

Note: After a seed error RNG restarts generating random numbers when SECS is cleared.

When the application sets the ARDIS bit in the RNG_CR register, the 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 the TAMP peripheral, and the RNG stops delivering random data.

When this event occurs, the 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.

24.3.8 RNG low-power use

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 the 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 reenabled 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 24.5: RNG processing time on page 725*.

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.

When the application wants to disable the RNG clock it is recommended to wait two RNG kernel clock cycles between clearing the RNGEN bit and disabling the RNG kernel clock using the RCC.

Also, when application needs to enter a power mode where RNG is de-activated, it is recommended to wait two RNG kernel clock cycles between clearing the RNGEN bit and entering the low power mode using the PWR.

In the two cases above, to avoid unexpected consumption when RNG analog oscillators stay active, application can set the bit 13 in RNG_CR register. Setting this bit adds some marginal power consumption while RNGEN bit is set (RNG activated).

Note: The power modes where RNG is deactivated (that is retained or not available) can be found in the PWR section.

24.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see *Section 24.3.7: Error management*
- Clock error, see *Section 24.3.7: Error management*

Dedicated interrupt enable control bits are available as shown in *Table 176*.

Table 176. 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 CONDRST to 1 then to 0 unless ARDIS is cleared (see *Section 24.3.7*). Write 0 to SEIS.
	Clock error flag	CEIS	IE	Write 0 to CEIS

The user can enable or disable the above interrupt sources individually by changing the mask bits or the general interrupt control bit IE in the RNG_CR register. The status of the individual interrupt sources can be read from the RNG_SR register.

Note: Interrupts are generated only when RNG is enabled.

24.5 RNG processing time

In recommended configuration A or C described in *Table 178*, the time between two sets of four 32-bit data is either:

- 203 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 x f_{RNG}, for instance 77 MHz if f_{RNG}= 48 MHz. Value N is defined in *Section 24.6: RNG entropy source validation*.

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:

- 203 AHB cycles if $f_{AHB} < f_{threshold}$ or
- 32 RNG cycles $f_{AHB} \geq f_{threshold}$

with $f_{threshold}$ = 6.5 x f_{RNG}.

Initialization time

More time is needed for the first set of random numbers after the device exits reset (see Section 1.3.4: RNG initialization). Table below gives details on how to compute the time spent in each initialization step.

Table 177. RNG initialization times

Initialization step	Configuration A or C, reset value	Configuraton B
Wait for noise source, then startup health tests	Max((wait_noise[(1)] + 11 + 1024 x CLKDIV) RNG_cycles, 13 x 203 AHB_cycles)	Max(16 + 1035 RNG_cycles, 13 x 203 AHB_cycles)
Conditioning keys initialization	Max((1+2xN) x 128 x CLKDIV) RNG_cycles + 203 AHB_cycles, (128 x CLKDIV) RNG_cycles + (2xN+2) x 203 AHB_cycles)	Max (3 x 32 RNG_cycles + 203 AHB_cycles, 32 RNG_cycles + 4 x 203 AHB_cycles)

1. 192 RNG_cycles (configuration A or C), 16 RNG_cycles (reset value)

As an example, if AHB clock= 100 MHz and RNG clock= 48 MHz then the initialization time is around 46 µs in the configuration C (CLKDIV=1, N=2) and 36 µs in the configuration B.

24.6 RNG entropy source validation

24.6.1 Introduction

In order to assess the amount of entropy available from the RNG, STMicroelectronics has tested the peripheral using the German BSI AIS-31 statistical tests (T0 to T8), and NIST SP800-90B test suite.

24.6.2 Validation conditions

STMicroelectronics has tested the RNG true random number generator in the following conditions:

- RNG clock rng_clk = 48 MHz
- RNG configurations are described in *Table 178*. Note that only configuration A can be certified NIST SP800-90B. Refer to *Table 179* to select the best configuration for the application.
- Configuration C can be used when configuration A is not flagged as certified.

Table 178. RNG configurations

<table>
<tr><th rowspan="2">Configuration</th><th colspan="6">RNG_CR bits</th><th rowspan="2">Loop number (N)</th><th rowspan="2">RNG_ HTCR register</th><th rowspan="2">RNG_ NSCR register</th></tr>
<tr><th>NISTC bit</th><th>RNG_ CONFIG1 [5:0]</th><th>CLKDIV [3:0]</th><th>RNG_ CONFIG2 [2:0][(1)]</th><th>RNG_ CONFIG3 [3:0][(2)]</th><th>CED bit</th></tr>
<tr><td>A</td><td colspan="9">Refer to NIST compliant RNG configuration table in AN4230 available from www.st.com.
This application note also indicates if this configuration is part of an existing NIST SP800-90B Entropy Certificate listed on https://csrc.nist.gov/projects/cryptographic-module-validation-program.</td></tr>
<tr><td>B</td><td>1</td><td>0x18</td><td rowspan="2">0x0[(3)]</td><td>0x0</td><td>0x0</td><td>0</td><td>1</td><td rowspan="2">0x0000 AAC7[(4)]</td><td>default</td></tr>
<tr><td>C</td><td>0</td><td>0x0F</td><td>0x0</td><td>0xD</td><td>0</td><td>2</td><td>default</td></tr>
</table>

1. 0x1 value is recommended instead of 0x0 for RNG_CONFIG2[2:0], when RNG power consumption is critical. See the end of *Section 24.3.8* for details.
2. RNG_CONFIG3[1:0] defines the loop number N: 0x0 corresponds to N=1, 0x1 to N=2, 0x2 to N=3, 0x3 to N=4

3. The noise source sampling must be 48 MHz or less. Hence, if the RNG clock is different from 48 MHz, this value of CLKDIV must be adapted. See the CLKDIV bitfield description in *Section 24.7.1* for details.
4. This value can be fixed in the RNG driver (it does not depend upon the STM32 product).

Table 179. Configuration selection

Section criteria	Config A	Config B	Config C
Suitable to generate NIST compliant cryptographic keys	Yes	No	
Entropy(1)	Certified	Good	Very good
Speed(2)	Baseline	Faster	Baseline

1. For configurations B and C entropy is verified using AIS-31 test suite (T0 to T8).
2. When speed is not enough for application a NIST compliant DRBG can be used to increase throughput.

For details on data collection and the running of statistical test suites refer to AN4230 "*STM32 microcontrollers random number generation validation using NIST statistical test suite*" available from *www.st.com*.

24.7 RNG registers

The RNG is associated with a control register, a data register and a status register.

24.7.1 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_NSCR, RNG_HTCR and RNG_CR configuration bits [29:4] are allowed.

1: Writes to the RNG_NSCR, RNG_HTCR and RNG_CR configuration bits [29:4] are ignored until the next RNG reset.

Once set, this bit can only be cleared when RNG is reset (set once bit).

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, RNG_NSCR 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 the 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 24.6: RNG entropy source validation*.

Writing any bit of RNG_CONFIG1 is taken into account only if the 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 the 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). Bit 13 can be set when RNG power consumption is critical. See *Section 24.3.8: RNG low-power use*. Refer to the 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 24.6: RNG entropy source validation* for recommended 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 the NISTC bit is cleared in this register RNG_CONFIG3 bitfield values are ignored by RNG.

Bit 7 **ARDIS**: Auto reset disable

Set this bit to deactivate the auto-reset feature.

0: Auto-reset enabled

1: Auto-reset disabled

Keeping the auto-reset enabled (automatic clearance of the SECS bit) simplifies the management of noise source errors, as described in *Section 24.3.7: Error management*.

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 enabled

1: Clock error detection is disabled

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 the 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 the DRDY, SEIS, or CEIS is set 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.

24.7.2 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 ($f_{RNGCLK} > f_{HCLK}/32$)

1: The RNG clock before the internal divider is detected too slow ($f_{RNGCLK} < f_{HCLK}/32$)

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 sequences has been detected:

- Runtime 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)
- Startup or continuous adaptive proportion test on noise source failed.
- Startup post-processing/conditioning sanity check failed.

Bit 1 **CECS:** Clock error current status

0: The RNG clock is correct (f_{RNGCLK}> f_{HCLK}/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 (f_{RNGCLK}< f_{HCLK}/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.

24.7.3 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 the DRDY = 1 and the 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, the RNDATA value is zero.

When DRDY is set, it is recommended to always verify that RNG_DR is different from zero. The zero value means that a seed error occurred between RNG_SR polling and RND_DR output reading (a rare event).

24.7.4 RNG noise source control register (RNG_NSCR)

Address offset: 0x00C

Reset value: 0x0003 FFFF

Writing in RNG_NSCR is taken into account only if the CONDRST bit is set, and the 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_OSC6[2:1]	
														rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EN_OSC6[0]	EN_OSC5[2:0]			EN_OSC4[2:0]			EN_OSC3[2:0]			EN_OSC2[2:0]			EN_OSC1[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:15 **EN_OSC6[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 6. The bitfield has no effect otherwise.

Bits 14:12 **EN_OSC5[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 5. The bitfield has no effect otherwise.

Bits 11:9 **EN_OSC4[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 4. The bitfield has no effect otherwise.

Bits 8:6 **EN_OSC3[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 3. The bitfield has no effect otherwise.

Bits 5:3 **EN_OSC2[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 2. The bitfield has no effect otherwise.

Bits 2:0 **EN_OSC1[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 1. The bitfield has no effect otherwise.

24.7.5 RNG health test control register (RNG_HTCR)

Address offset: 0x010

Reset value: 0x0000 72AC

Writing in RNG_HTCR is taken into account only if the CONDRST bit is set, and the CONFIGLOCK bit is cleared in the 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 24.6: 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.

24.7.6 RNG register map

Table 180. 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
0x00C	RNG_NSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_OSC6[2:0]			EN_OSC5[2:0]			EN_OSC4[2:0]			EN_OSC3[2:0]			EN_OSC2[2:0]			EN_OSC1[2:0]		
	Reset value															1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
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: Memory organization* for the register boundary addresses.

25 AES hardware accelerator (AES)

In this section, any references to SAES only apply to STM32WBA52/54/55xx products. The STM32WBA50xx products do not have the SAES peripheral.

25.1 Introduction

The AES hardware accelerator (AES) encrypts or decrypts data in compliance with the advanced encryption standard (AES) defined by NIST.

AES supports ECB, CBC, CTR, GCM, GMAC, and CCM chaining modes for key sizes of 128 or 256 bits.

AES has the possibility to load by hardware the key stored in SAES peripheral, under SAES control.

The peripheral supports DMA single transfers for incoming and outgoing data (two DMA channels are required).

25.2 AES main features

- Compliant with NIST FIPS publication 197 *“Advanced encryption standard (AES)”* (November 2001)
- 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
- 128-bit data block processing, supporting cipher key lengths of 128-bit and 256-bit
  - 51 or 75 clock cycle latency in ECB mode for processing one 128-bit block with, respectively, 128-bit or 256-bit key
- Using dedicated key bus, optional key sharing with side-channel resistant SAES peripheral (shared-key mode), controlled by SAES
- Integrated key scheduler to compute the last round key for ECB/CBC decryption
- 256-bit of write-only registers for storing cryptographic keys (eight 32-bit registers)
- 128-bit of registers for storing initialization vectors (four 32-bit registers)
- 32-bit buffer for data input and output
- Automatic data flow control supporting two direct memory access (DMA) channels, one for incoming data, one for processed data. Only single transfers are supported.
- Data-swapping logic to support 1-, 8-, 16-, or 32-bit data

- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.
- Possibility for software (in CPU mode only, not in DMA mode) to suspend a message if AES needs to process another message with a higher priority, then resume the original message

25.3 AES implementation

The devices have one AES peripheral, implemented as per the following table. It can use the key generated by the SAES peripheral. For comparison, the SAES peripheral is also included in the table.

Table 181. AES versus SAES features

Modes or features(1)	AES	SAES
ECB, CBC chaining	X	X
CTR, CCM, GCM chaining	X	X
AES 128-bit ECB encryption in cycles	51	480
DHUK and BHK key selection	-	X
Resistance to side-channel attacks	-	X
Shared key between SAES and AES	X	
Key sizes in bits	128, 256	128, 256

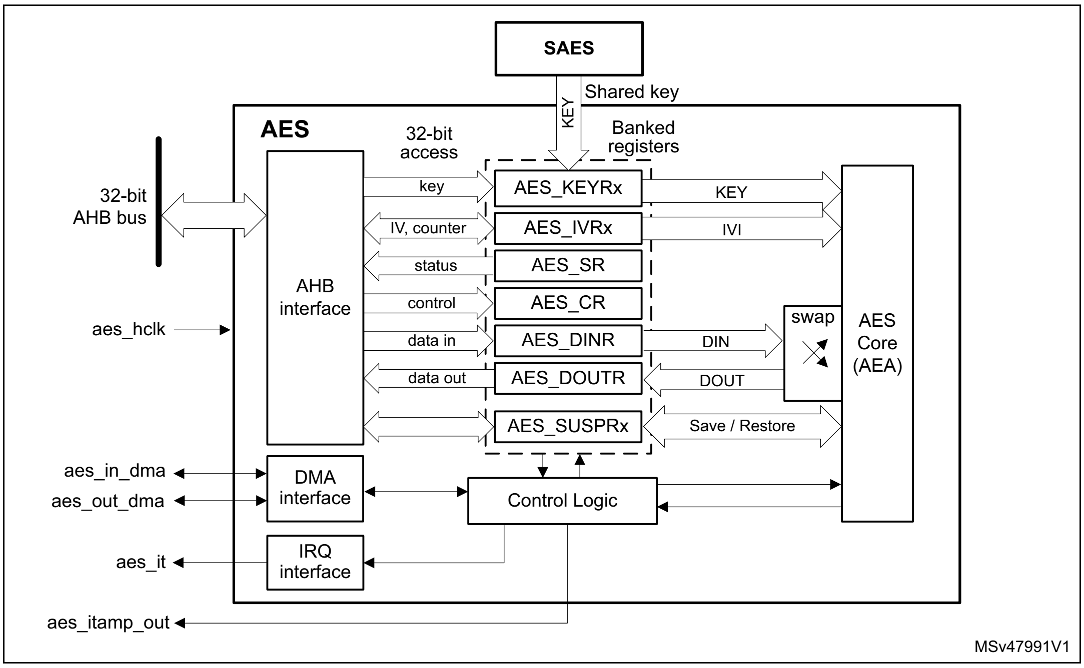
1. X = supported.

25.4 AES functional description

25.4.1 AES block diagram

Figure 113 shows the block diagram of AES.

Figure 113. AES block diagram



25.4.2 AES internal signals

Table 182 describes the user relevant internal signals interfacing the AES peripheral.

Table 182. 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	AES incoming data DMA single request/acknowledge
aes_out_dma	Input/Output	AES processed data 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.

25.4.3 AES reset and clocks

The AES peripheral is clocked by the AHB bus clock. It has a dedicated reset bit controlled through the RCC.

25.4.4 AES symmetric cipher implementation

The AES hardware accelerator (AES) is a 32-bit AHB peripheral that encrypts or decrypts 16-byte blocks of data using the advanced encryption standard (AES). It also implements a set of approved AES symmetric key security functions summarized in *Table 183*. Those functions can be certified NIST PUB 140-3.

Table 183. AES approved symmetric key functions

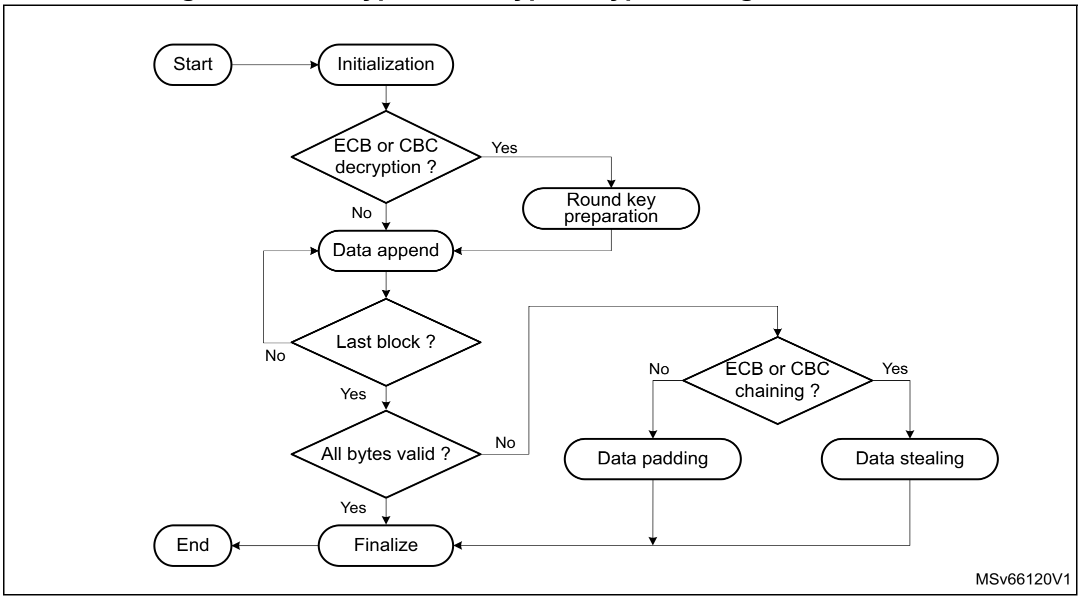
Operations	Algorithm	Specification	Key bit lengths	Chaining modes
Encryption, decryption	AES	FIPS PUB 197 NIST SP800-38A	128, 256	ECB, CBC, CTR
Authenticated encryption or decryption		NIST SP800-38C NIST SP800-38D		GCM, CCM
Cipher-based message authentication code		NIST SP800-38D		GMAC

AES can be used directly by the CPU, or indirectly, using two DMA channels (one for the plaintext, one for the ciphertext).
It is possible to suspend then resume any AES processing, following the sequence described in *Section 25.4.8*.

25.4.5 AES encryption or decryption typical usage

The following figure shows a typical operation for encryption or decryption.

Figure 114. Encryption/ decryption typical usage



Initialization

The AES peripheral is initialized according to the chaining mode. Refer to *Section 25.4.9: AES basic chaining modes (ECB, CBC)* and *Section 25.4.10: AES counter (CTR) mode* for details.

Data append

This section describes different ways of appending data for processing. For ECB or CBC chaining modes, refer to *Section 25.4.7: AES ciphertext stealing and data padding* if the size of data to process is not a multiple of 16 bytes. The last block management in these cases is more complex than what is described in this section.

Appending data using the CPU in polling mode

This method uses flag polling to control the data append through the following sequence:

1. Enable the AES peripheral when KEYVALID is set, by setting the EN bit of the AES_CR register (if not already done).
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_ISR register, 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 next processing block is the last block, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (using NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong).
3. As the data block just processed is the last block of the message, optionally discard the data that is not part of the message/payload, then disable the AES peripheral by clearing EN.

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[3:0] bitfield is not used in header phase of GCM, GMAC and CCM chaining modes.

Appending data using the CPU in interrupt mode

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 when KEYVALID is set, by setting EN (if not already done).
3. Write first four input data words into the AES_DINR register.
4. Handle the data in the AES interrupt service routine. Upon each 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 next processing block is the last block of the message, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (through NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong). Then proceed with point 4e).
   d) If the data block just processed is the last block of the message, optionally discard the data that are not part of the message/payload, then disable the AES peripheral by clearing EN 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.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

Appending data using DMA

With this method, all the transfers and processing are managed by DMA and AES. Proceed as follows:

1. If the last block of the message to process is shorter than 16 bytes, prepare the last four-word data block 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 25.6: AES DMA requests*. Configure the DMA controller so as to generate an interrupt on transfer completion. For GCM payload encryption or CCM payload decryption, the DMA transfer **must not** include the last four-word block if padded with zeros. The sequence described in *Appending data using the CPU in polling mode* must be used instead for this last block, because the NPBLB[3:0] bitfield must be set up before processing the block, for AES to compute a correct tag.
3. Enable the AES peripheral when KEYVALID is set, by setting EN (if not already done).
4. Enable DMA requests, by setting DMAINEN and DMAOUTEN.
5. Upon DMA interrupt indicating the transfer completion, get the AES-processed data from the memory.

When appending data using DMA, the suspend/resume operation as described in *Section 25.4.8* is not supported.

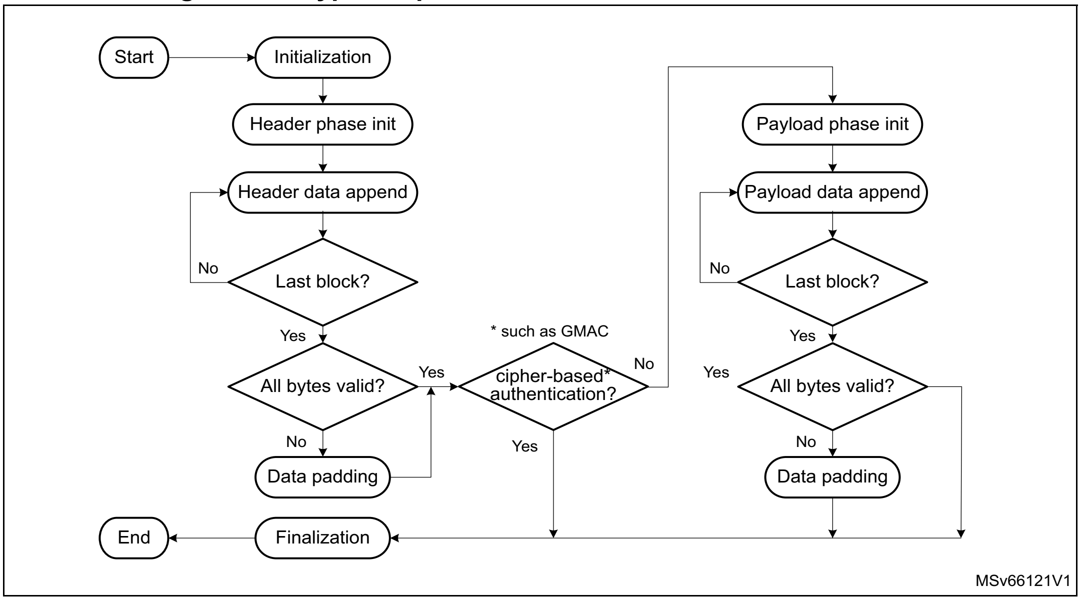
Note: The CCF flag has no use with this method because the reading of the AES_DOUTR register is managed by DMA automatically, without any software action, at the end of the computation phase.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

25.4.6 AES authenticated encryption, decryption, and cipher-based message authentication

The following figure shows a typical operation for authenticated encryption or decryption, and for cipher-based message authentication.

Figure 115. Typical operation with authentication



Section 25.4.11: AES Galois/counter mode (GCM) and *Section 25.4.13: AES counter with CBC-MAC (CCM)* describe detailed sequences supported by AES.

Cipher-based message authentication flow omits the payload phase, as shown in the figure. Detailed sequence supported by AES is described in *Section 25.4.12: AES Galois message authentication code (GMAC)*.

25.4.7 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 (16 bytes), the application must use ciphertext stealing techniques 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 AES does not implement 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, in modes other than ECB or CBC, an incomplete input data block (that is, a block with input data shorter than 16 bytes) 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. Since AES does not implement automatic data padding operation to the *last block*, the application must follow the recommendation given in this document to manage messages the size of which is not a multiple of 16 bytes.

25.4.8 AES suspend and resume operations

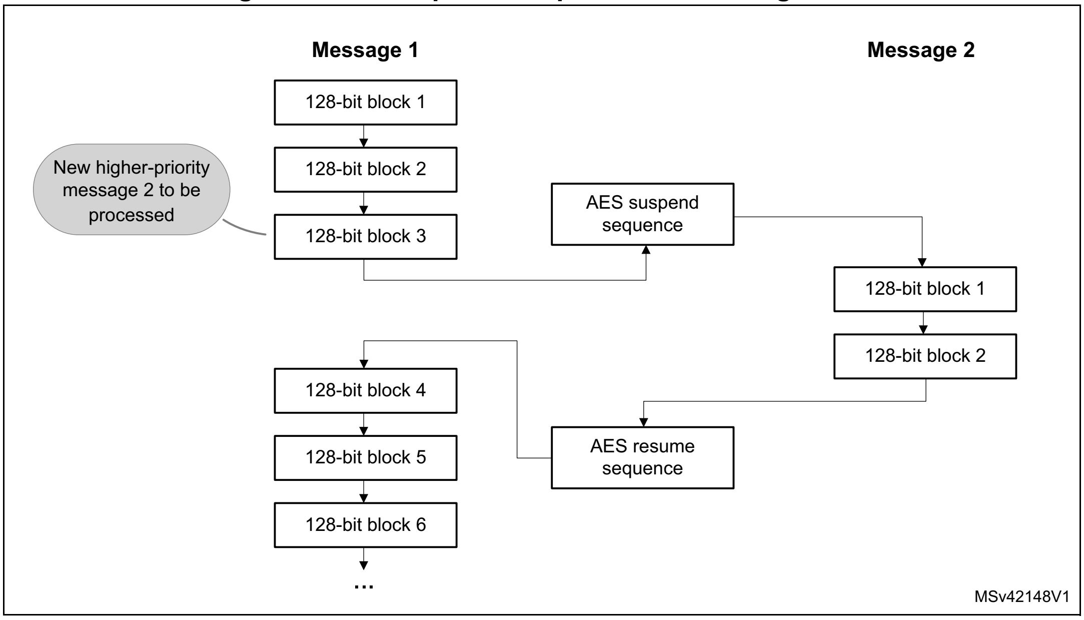
A message can be suspended to process another message with a higher priority. When the higher-priority message is sent, the suspended message can resume. This applies to both encryption and decryption mode.

Suspend and resume operations do not break the chaining operation. The message processing can resume as soon as AES is enabled again, to receive a next data block.

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

Figure 116 gives an example of suspend and resume operations: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

Figure 116. Example of suspend mode management



A detailed description of suspend and resume operations is in the sections dedicated to each chaining mode.

25.4.9 AES basic chaining modes (ECB, CBC)

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. When decrypting in ECB, a special key scheduling is required before processing the first block.

Figure 117 and *Figure 118* describe the electronic codebook (ECB) chaining implementation in encryption and in decryption, respectively. To select ECB chaining mode, write CHMOD[2:0] with 0x0.

Figure 117. ECB encryption

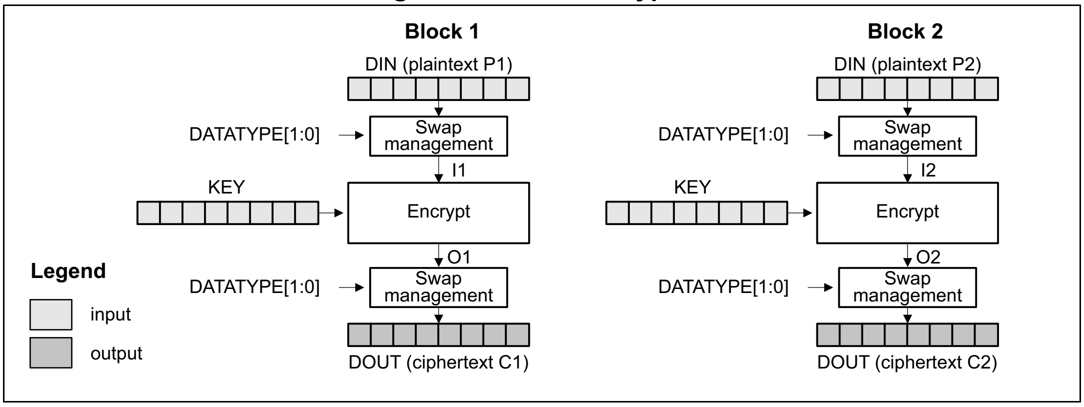
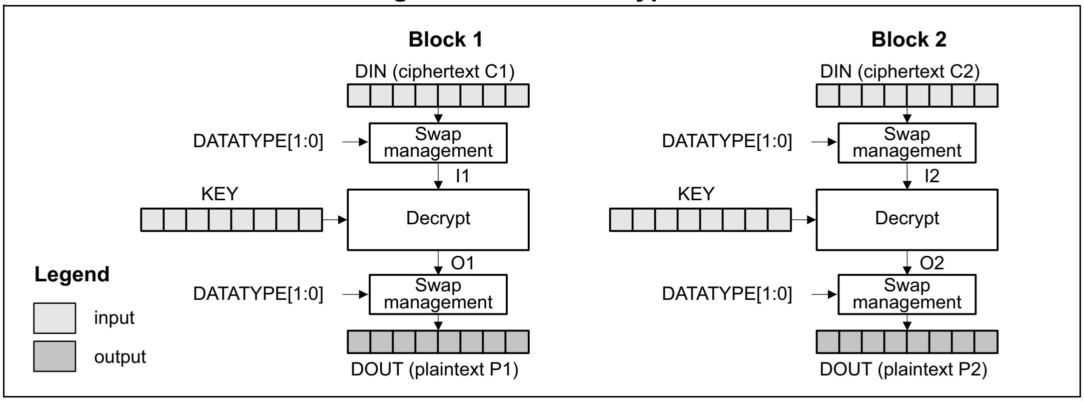


Figure 118. ECB decryption



In CBC encryption 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. When decrypting in CBC, a special key scheduling is required before processing the first block.

Figure 119 and *Figure 120* describe the cipher block chaining (CBC) implementation in encryption and in decryption, respectively. To select this chaining mode, write CHMOD[2:0] with 0x1.

Figure 119. CBC encryption

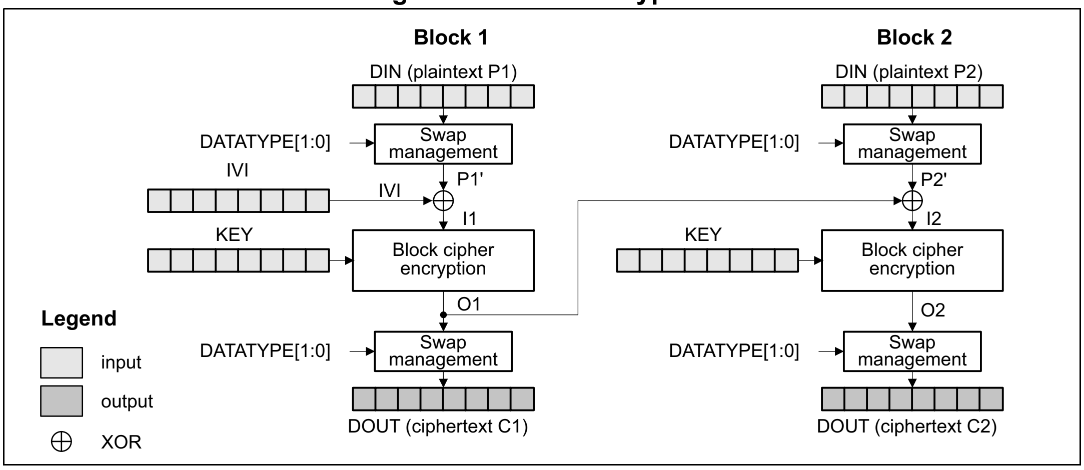
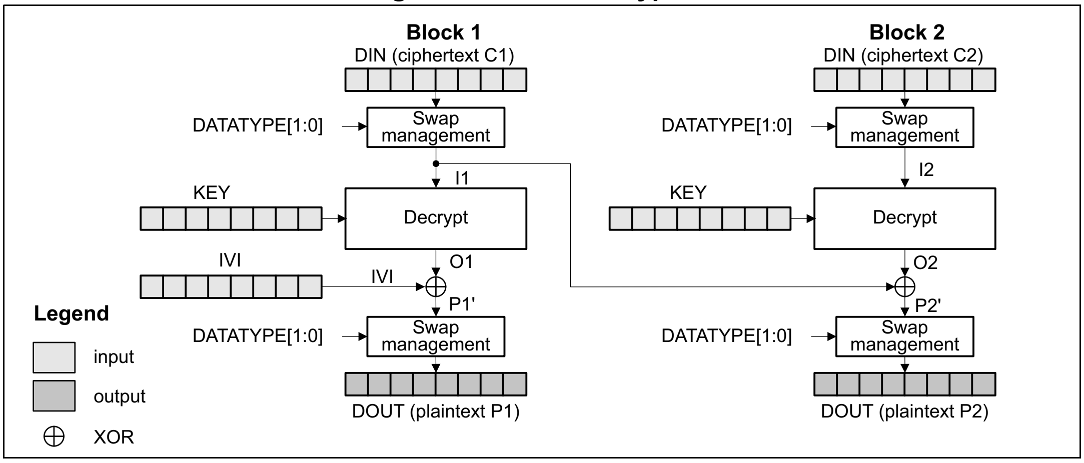


Figure 120. CBC decryption



For more details, refer to NIST Special Publication 800-38A, *Recommendation for Block Cipher Modes of Operation.*

ECB and CBC encryption process

This process is described in *Section 25.4.5*, with the following sequence of events:

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register as follows:
   – Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *encryption* mode (write MODE[1:0] with 0x0).
   – Configure the data type, through DATATYPE[1:0].
   – Configure the key size, through KEYSIZE.
   – Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
3. Write the initialization vector into the AES_IVRx registers if CBC mode is selected in the previous step.
4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 25.4.14*).
5. Wait until KEYVALID is set (the key loading completed).
6. Enable the AES peripheral, by setting EN.
7. Append cleartext data:
   a) If it is the second-last or the last block and the plaintext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 25.4.7*.
   b) Append the cleartext block into AES as described in *Section 25.4.5*, then read the AES_DOUTR register four times to save the ciphertext block.
   c) Repeat the step *b)* until the third-last plaintext block is encrypted. For the last two blocks, follow the steps *a)* and *b)*.
8. Finalize the sequence: disable the AES peripheral, by clearing EN.

ECB/CBC decryption process

This process is described in *Section 25.4.5*, with the following sequence of events:

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register as follows:
   – Select the *key derivation* mode (write MODE[1:0] with 0x1). The CHMOD[2:0] bitfield is not significant during this operation.
   – Configure the data type, through DATATYPE[1:0].
   – Configure the key size, through KEYSIZE.
   – Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
3. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 25.4.14*).
4. Wait until KEYVALID is set (the key loading completed).
5. Enable the AES peripheral, by setting EN. The peripheral immediately starts an AES round for key preparation.
6. Wait until the CCF flag in the AES_ISR register is set.
7. Clear the CCF flag, by setting the CCF bit of the AES_ICR register. The decryption key is available in the AES core and AES is disabled automatically.
8. Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *decryption* mode (write MODE[1:0] with 0x2). Do not change other parameters.
9. Write the initialization vector into the AES_IVRx registers if CBC mode is selected in the previous step.
10. Enable the AES peripheral, by setting EN.
11. Append encrypted data:
    a) If it is the second-last or the last block and the ciphertext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 25.4.7*.
    b) Append the ciphertext block into AES as described in *Section 25.4.5*, then read the AES_DOUTR register four times to save the cleartext block (MSB first).
    c) Repeat the step *b)* until the third-last ciphertext block is decrypted. For the last two blocks, follow the steps *a)* and *b)*.
12. Finalize the sequence: disable the AES peripheral, by clearing EN.

Suspend/resume operations in ECB/CBC modes

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

To suspend the processing of a message, proceed as follows:

1. Wait until the CCF flag in the AES_ISR register is set (computation completed).
2. Read four times the AES_DOUTR register to save the last processed block.
3. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
4. Save initialization vector registers (only required in CBC mode as the AES_IVRx registers are altered during the data processing).

5. Disable the AES peripheral, by clearing EN.
6. Save the AES_CR register and clear the key registers if they are not needed, to process the higher-priority message.

To resume the processing of a message, proceed as follows:

1. Disable the AES peripheral, by clearing EN.
2. Restore the AES_CR register (with correct KEYSIZE) then restore the AES_KEYRx registers. If KMOD[1:0] is at 0x2, the key must be transferred again from the SAES peripheral (see *Section 25.4.14*).
3. Prepare the decryption key, as described in *ECB/CBC decryption process* (only required for ECB or CBC decryption).
4. Restore the AES_IVRx registers, using the saved configuration (only required in CBC mode).
5. Enable the AES peripheral, by setting EN.

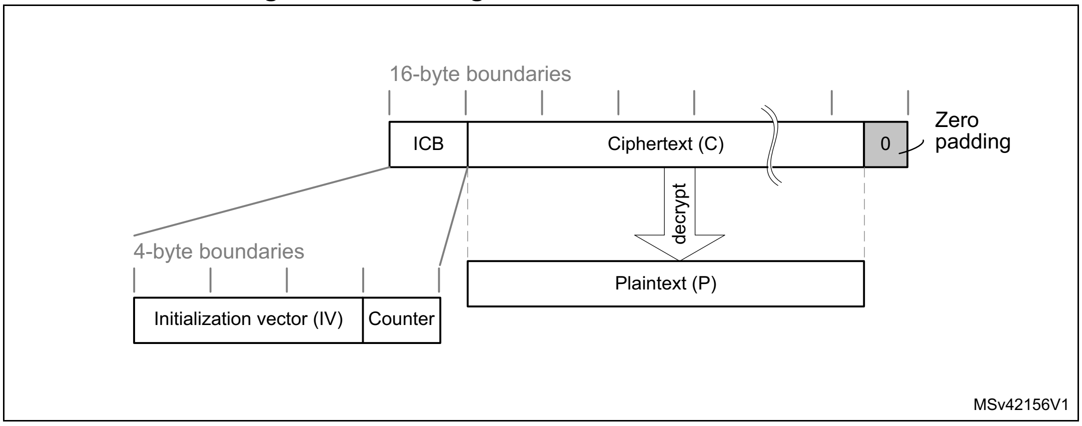
Note: It is not required to save the key registers as the application knows the original key.

25.4.10 AES counter (CTR) mode

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. Unlike with ECB and CBC modes, no key scheduling is required for the CTR decryption since the AES core is always used in encryption mode.

A typical message construction in CTR mode is given in *Figure 121*.

Figure 121. Message construction in CTR mode



The structure of this message is:

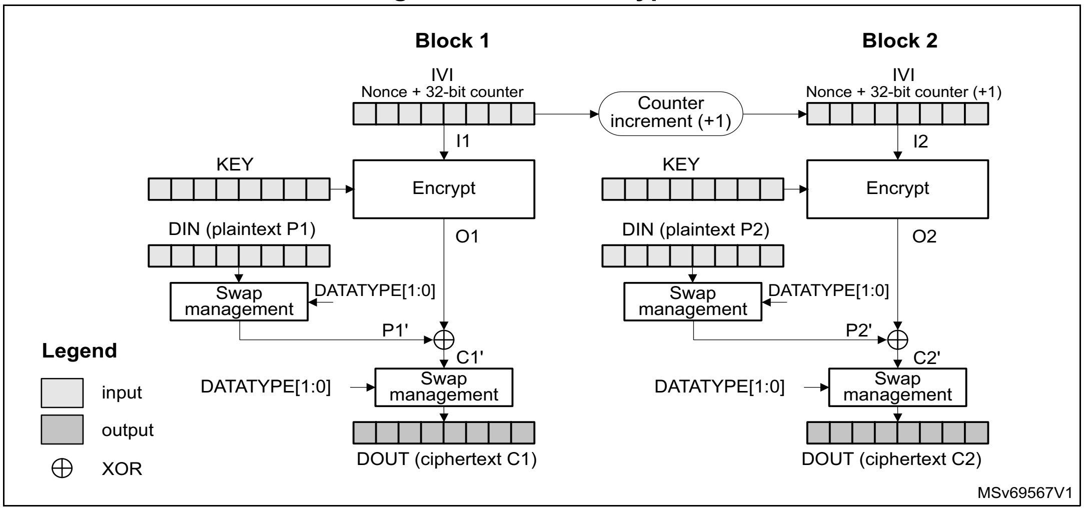
- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - **Initialization vector** (IV): a 96-bit value that must be unique for each encryption cycle with a given key.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. The 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.

For more details, refer to NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.*

CTR encryption and decryption

Figure 122 describes the counter (CTR) chaining implementation in the AES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x2.

Figure 122. CTR encryption



Initialization vectors in AES must be initialized as shown in *Table 184*.

Table 184. Counter 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

CTR encryption and decryption process

This process is described in *Section 25.4.5*, with the following sequence of events:

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register:
   - Select CTR chaining mode (write CHMOD[2:0] with 0x2) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
3. Write the initialization vector into the AES_IVRx registers according to *Table 184*.
4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 25.4.14*).
5. Wait until KEYVALID is set (the key loading completed).
6. Enable the AES peripheral, by setting EN.

7. Append data:
   a) If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in the block is less than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into AES as described in *Section 25.4.5*, then read the AES_DOUTR register four times to save the resulting block (MSB first).
   c) Repeat the step *b)* until the second-last block is processed. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the message when the last block is smaller than 16 bytes.
8. Finalize the sequence: disable the AES peripheral, by clearing EN.

Suspend/resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher-priority message, then resume the interrupted message. Detailed CBC suspend and resume sequence is described in *Section 25.4.9: AES basic chaining modes (ECB, CBC)*.

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

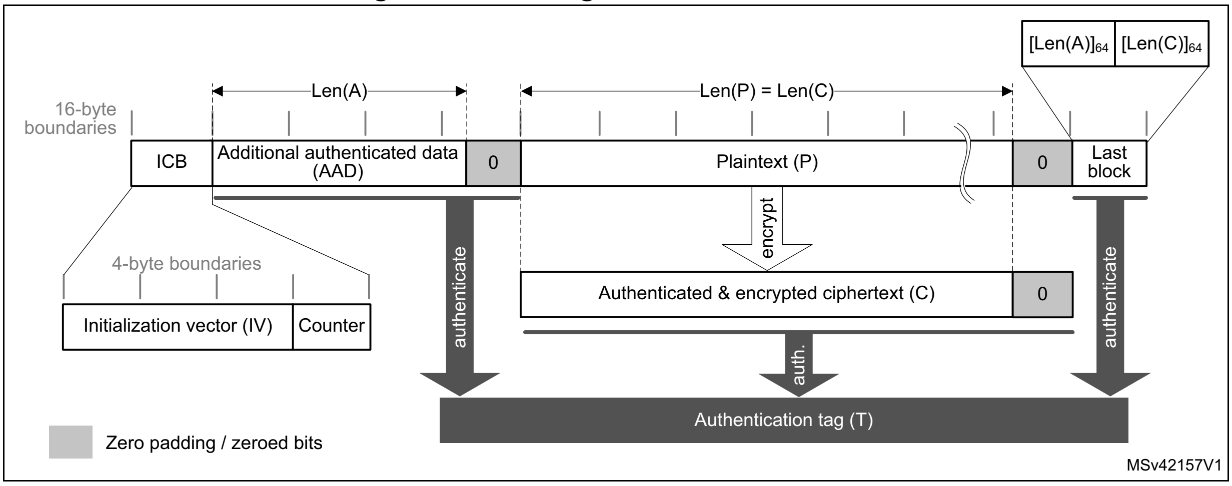
Note: Like for CBC mode, the IV registers must be reloaded during the resume operation.

25.4.11 AES Galois/counter mode (GCM)

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).

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. The following figure shows a typical message construction in GCM mode.

Figure 123. 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. 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 16-byte blocks.
- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in *Table 186*.

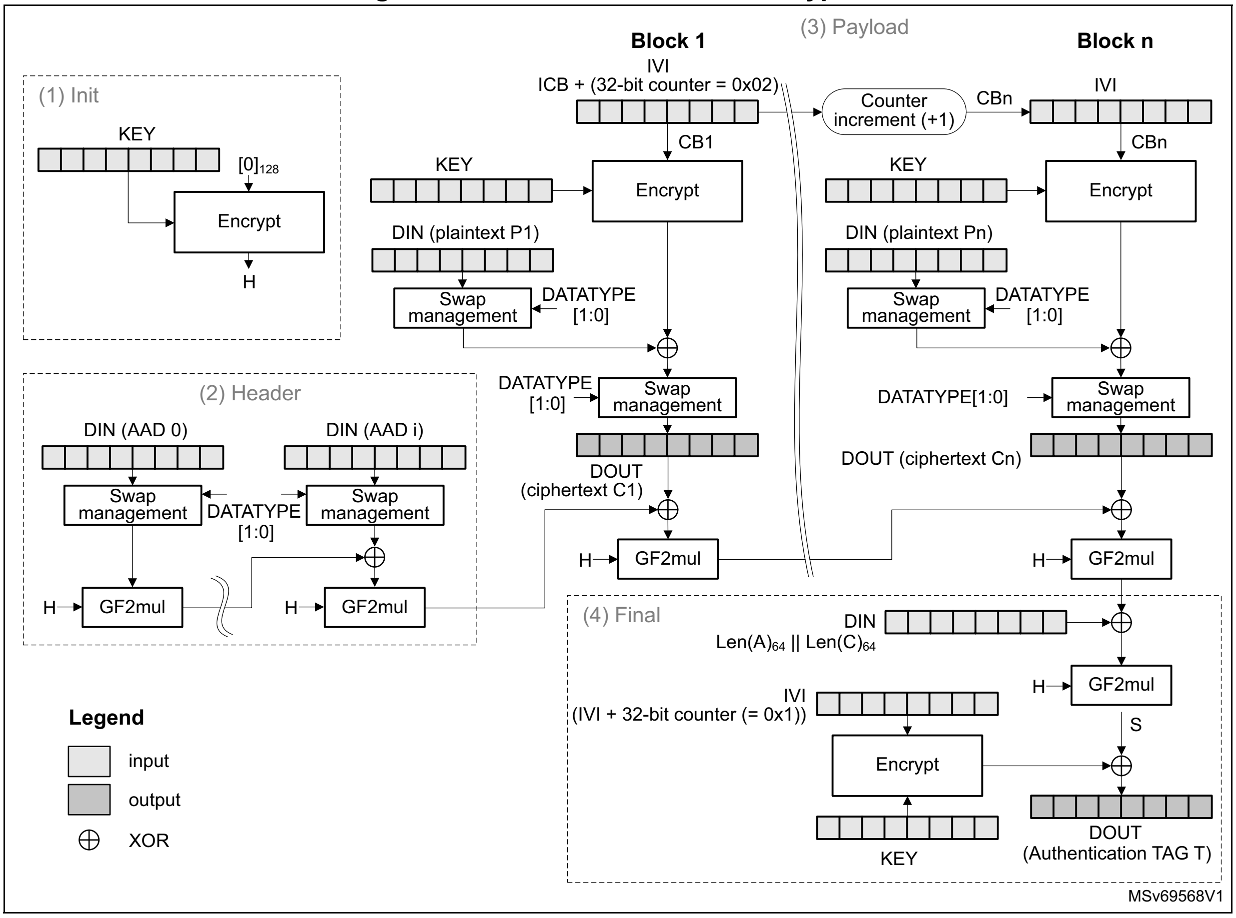
The GCM standard specifies that ciphertext C has the same bit length as the plaintext P.

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.

For more details, refer to NIST Special Publication 800-38D*, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 124 describes the GCM chaining implementation in the AES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 124. GCM authenticated encryption



The first counter block (CB1) is derived from the initial counter block ICB by the application software, as defined in *Table 185*.

Table 185. Initialization of IV 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

The last block of a GCM message contains the AAD header length and the payload length information, as shown in *Table 186*.

Table 186. GCM last block definition

Word order to AES_DINR	First word	Second word	Third word	Fourth word
Input data	AAD length[63:32]	AAD length[31:0]	Payload length[63:32]	Payload length[31:0]

GCM encryption and decryption process

This process is described in *Section 25.4.6*, with the following sequence of events:

GCM initialize

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register:
   - Select GCM chaining mode (write CHMOD[2:0] with 0x3) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
   - Select the GCM initialization phase, by writing GCMPH[1:0] with 0x0.
3. Write the initialization vector in AES_IVRx registers according to *Table 185*.
4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 25.4.14*).
5. Wait until KEYVALID is set (the key loading completed).
6. Set EN to start the calculation of the hash key. EN is automatically cleared when the calculation is completed.
7. Wait until the CCF flag is set in the AES_ISR register, indicating that the GCM hash subkey (H) computation is completed.
8. Clear the CCF flag by setting the CCF bit of the AES_ICR register.

GCM header phase

9. Initialize header phase:
   a) Select the GCM header phase, by writing 0x1 to GCMPH[1:0]. Do not change the other configurations written during GCM initialization.
   b) Enable the AES peripheral, by setting EN.
10. Append header data:
   a) If it is the last block and the AAD in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into AES as described in *Section 25.4.5*.
   c) Repeat the step *b)* until the second-last AAD data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no AAD, that is, Len(A) = 0.

No data are read during header phase.

GCM payload phase

11. Initialize payload phase:
   a) Select the GCM payload phase, by writing GCMPH[1:0] with 0x2. Do not change the other configurations written during GCM initialization.
   b) If the header phase is skipped, enable the AES peripheral by setting EN.

12. Append payload data:
    a) If it is the last block and the message in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into AES as described in *Section 25.4.5*, then read the AES_DOUTR register four times to save the resulting block
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(C)=0 (see GMAC mode).

GCM finalization

13. Select the GCM final phase, by writing GCMPH[1:0] with 0x3. Do not change the other configurations written during GCM initialization.
14. Write the final GCM block into the AES_DINR register. It is the concatenated AAD bit and payload bit lengths, as shown in *Table 186*.
15. Wait until the CCF flag in the AES_ISR register is set.
16. Get the GCM authentication tag, by reading the AES_DOUTR register four times.
17. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
18. Disable the AES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message.

Note: In the final phase, data are 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

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

To suspend the processing of a message, proceed as follows:

1. Wait until the CCF flag in the AES_ISR register is set (computation completed).
2. In the payload phase, read four times the AES_DOUTR register to save the last-processed block.
3. Clear the CCF flag of the AES_ISR register, by setting the CCF bit of the AES_ICR register. When GCM encryption payload phase is selected, verify that BUSY is cleared in AES_SR, to ensure that GF2mul hash function is completed.
4. Save the AES_SUSPRx registers in the memory.
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 EN.
7. Save the current AES_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.

To resume the processing of a message, proceed as follows:

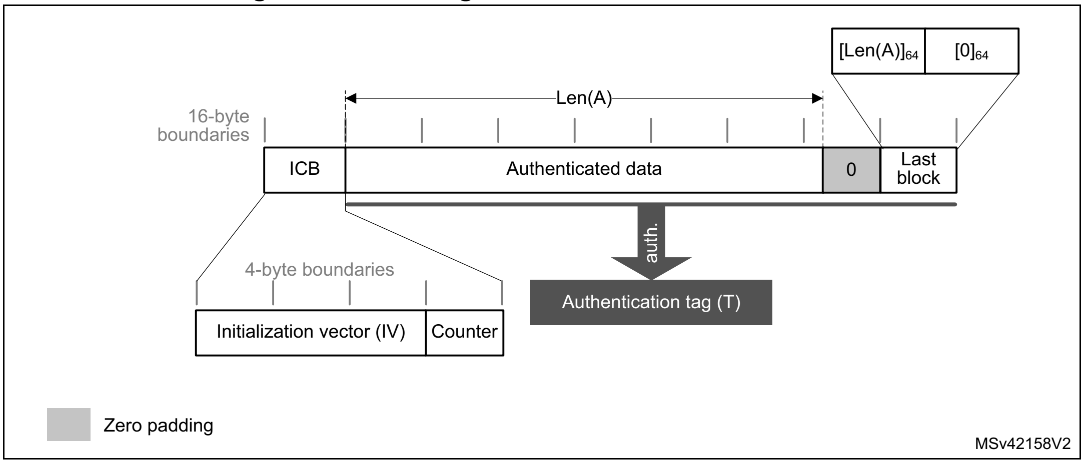
1. Disable the AES peripheral, by clearing EN.
2. Write the suspend register values, previously saved in the memory, back into their corresponding AES_SUSPRx registers.
3. In the payload phase, write the initialization vector register values, previously saved in the memory, back into their corresponding AES_IVRx registers. In the header phase, write initial setting values back into the AES_IVRx registers.
4. Restore the initial setting values in the AES_CR and AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 25.4.14*).
5. Enable the AES peripheral, by setting EN.

25.4.12 AES Galois message authentication code (GMAC)

The Galois message authentication code (GMAC) allows the authentication of a plaintext, generating the corresponding tag information (also known as message authentication code).

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). The following figure shows typical message construction for GMAC.

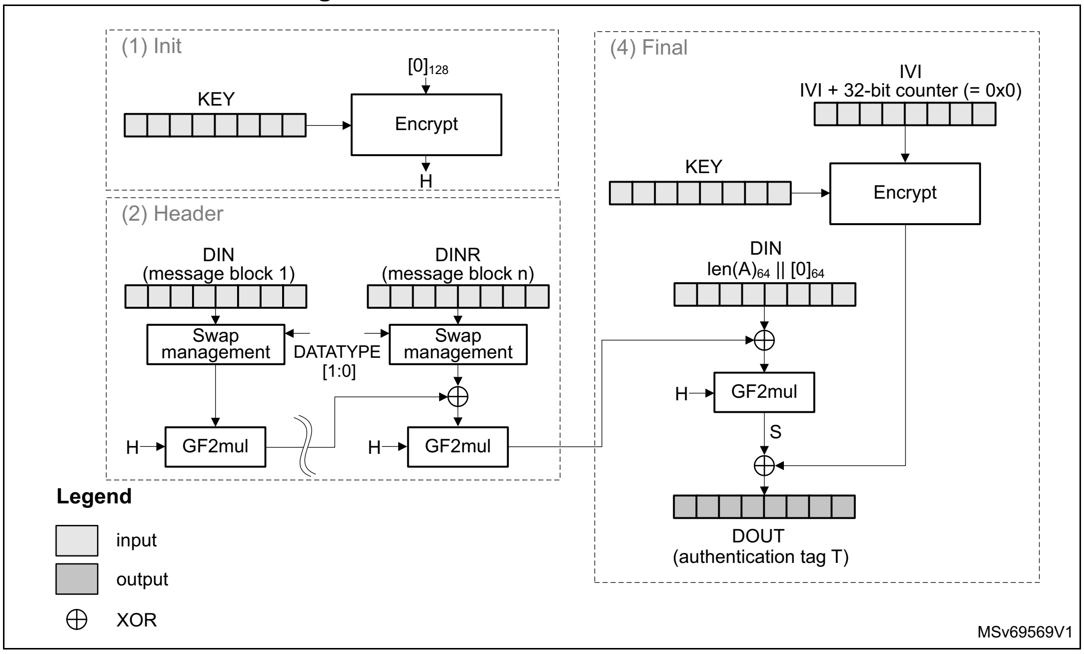
Figure 125. Message construction in GMAC mode



For more details, refer to NIST Special Publication 800-38D*, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 126 describes the GMAC chaining implementation in the AES peripheral. To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 126. GMAC authentication mode



The GMAC algorithm corresponds to the GCM algorithm applied on a message that only contains 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.

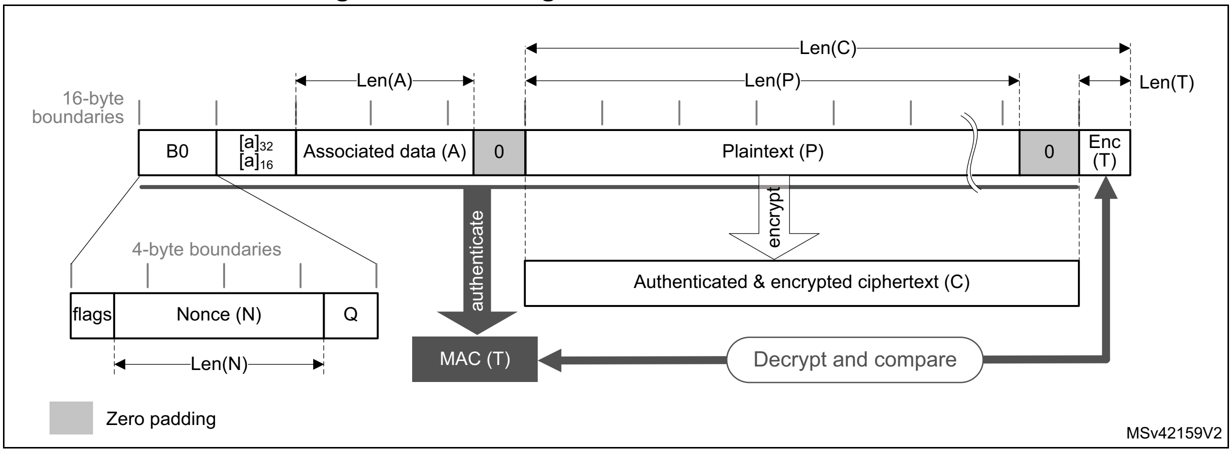
25.4.13 AES counter with CBC-MAC (CCM)

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 counter mode processing. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

Note: NIST does not approve CBC-MAC as an authentication mode outside the context of the CCM specification.

The following figure shows typical message construction for CCM.

Figure 127. 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 127*). 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 127*).
- **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

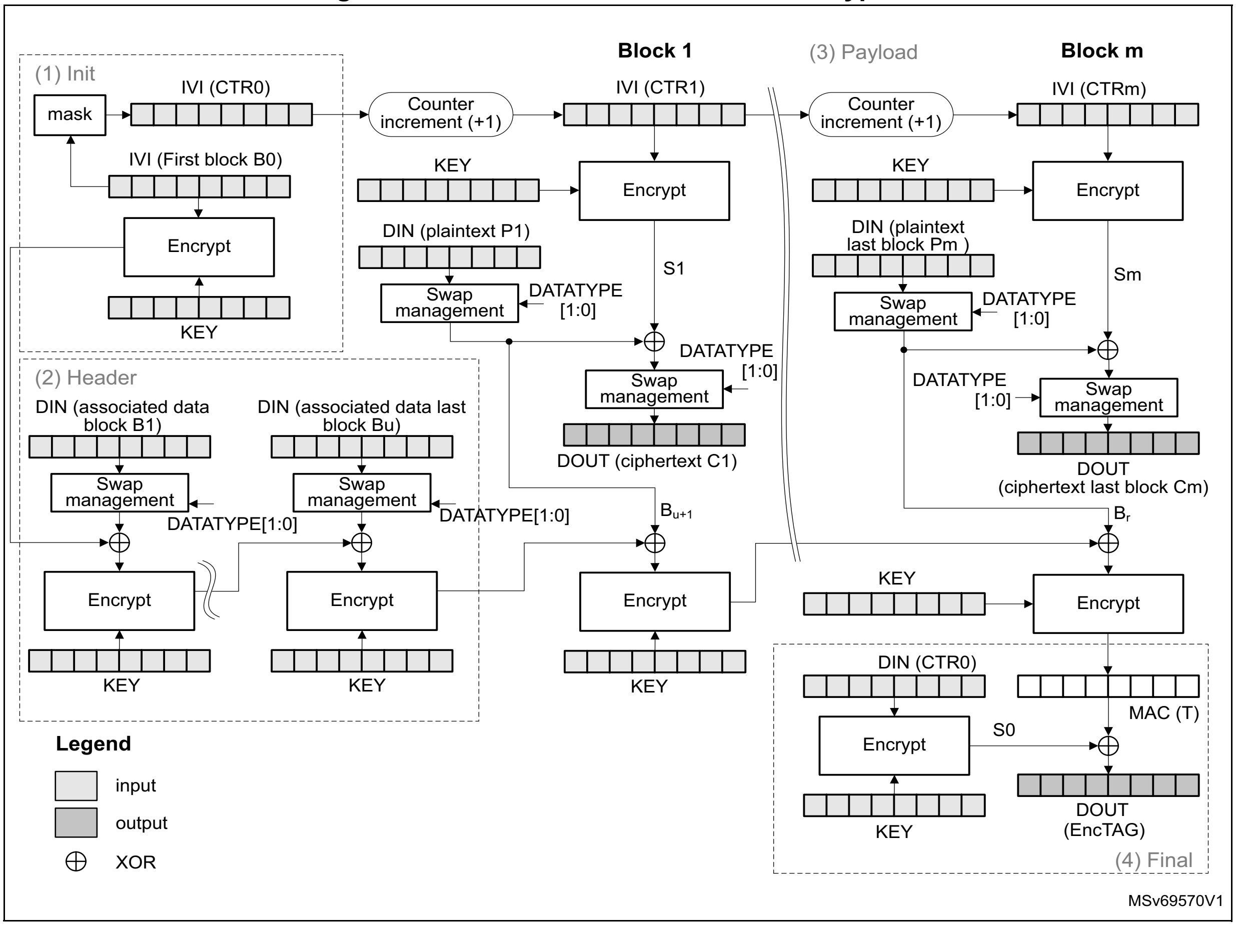
CTR0: 0710111213 141516 00000000 00000000

CTR1: 0710111213 141516 00000000 00000001

For more details, refer to NIST Special Publication 800-38C*, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*.

Figure 128 describes the CCM chaining implementation in the AES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x4.

Figure 128. CCM mode authenticated encryption



The first block of a CCM message (B0) must be prepared by the application as defined in *Table 187*.

Table 187. Initialization of IV registers in CCM mode

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
B0[127:96][(1)]	B0[95:64]	B0[63:32]	B0[31:0][(2)]

1. The five most significant bits are cleared (flag bits).
2. Q length bits are cleared, except for the bit 0 that is set.

AES supports counters up to 64 bits, as specified by NIST.

CCM encryption and decryption process

This process is described in *Section 25.4.6*, with the following sequence of events:

CCM initialize

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register:
   - Select CCM chaining mode (write CHMOD[2:0] with 0x4) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
   - Select the CCM initialization phase, by writing GCMPH[1:0] with 0x0.
3. Write the B0 data in AES_IVRx registers according to *Table 187*.
4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 25.4.14*).
5. Wait until KEYVALID is set (the key loading completed).
6. Set EN to start the first mask calculation. The EN bit is automatically cleared when the calculation is completed.
7. Wait until the CCF flag in the AES_ISR register is set.
8. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.

CCM header phase

9. Initialize header phase:
   a) Prepare the first block of the (B1) data associated with the message, in accordance with CCM chaining rules.
   b) Select the CCM header phase, by writing GCMPH[1:0] with 0x1. Do not change the other configurations written during the CCM initialization.
   c) Enable the AES peripheral, by setting EN.
10. Append header data:
   a) If it is the last block and the associated data in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into AES as described in *Section 25.4.5*.
   c) Repeat the step *b)* until the second-last associated data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no associated data, that is, Len(A) = 0

No data are read during the header phase.

CCM payload phase

11. Initialize payload phase:
    a) Select the CCM payload phase, by writing GCMPH[1:0] with 0x2. Do not change the other configurations written during the CCM initialization.
    b) If the header phase is skipped, enable the AES peripheral, by setting EN.
12. Append payload data:
    a) In encryption only, if it is the last block and the plaintext in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into AES as described in *Section 25.4.5*, then read the AES_DOUTR register four times to save the resulting block.
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(P) = 0 or Len(C) = Len(T).

Remove $LSB_{Len(T)}$(C) encrypted tag information when decrypting ciphertext C.

CCM finalization

13. Select the CCM final phase, by writing GCMPH[1:0] with 0x3. Do not change the other configurations written during the CCM initialization.
14. Wait until CCF flag in the AES_ISR register is set.
15. Get the CCM authentication tag, by reading the AES_DOUTR register four times.
16. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
17. Disable the AES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message. Mask the authentication tag output with tag length to obtain a valid tag.

Note: In the final phase, swapping is applied to tag data read from AES_DOUTR register.

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 and resume operations in CCM mode

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

To suspend the processing of a message in header or payload phase, proceed as follows:

1. Wait until the CCF flag of the AES_ISR register is set (computation completed).
2. In the payload phase, read four times the AES_DOUTR register to save the last-processed block.
3. Clear the CCF flag in the AES_ISR register, by setting the CCF bit of the AES_ICR register.
4. Save the AES_SUSPRx registers in the memory.
5. Save the IV registers as they are altered during the data processing.
6. Disable the AES peripheral, by clearing EN.

7. Save the current AES_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.

To resume the processing of a message, proceed as follows:

1. Disable the AES peripheral, by clearing EN.
2. Write the suspend register values, previously saved in the memory, back into their corresponding AES_SUSPRx registers.
3. Restore AES_IVRx registers using the saved configuration.
4. Restore the initial setting values in the AES_CR and AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key must be transferred again from the SAES peripheral (see *Section 25.4.14*).
5. Enable the AES peripheral, by setting EN.

25.4.14 AES key sharing with secure AES co-processor

The AES peripheral can use the SAES peripheral as security co-processor. The secure application prepares the key in the robust SAES peripheral and when it is ready, the AES application can load this prepared key through a dedicated hardware key bus.

The recommended sequence is described hereafter and in the section *SAES operations with shared keys* in the *SAES* section of this document.

1. In SAES peripheral, the application encrypts (wraps) the key to share in Shared-key mode (KMOD[1:0] at 0x2).
2. Each time the shared key is required in AES peripheral, the application decrypts it in the SAES peripheral in Shared-key mode (KMOD[1:0] at 0x2).
3. Once the shared key is decrypted (unwrapped) and loaded in SAES_KEYRx registers it can be shared with AES. To load the shared key in AES, the application sets KEYSIZE as appropriate and writes KMOD[1:0] with 0x2. When KEYVALID is cleared, the key is automatically transfered by hardware into the AES_KEYRx registers and the BUSY flag in the AES_SR register set.
4. Once the key transfer is completed, the BUSY flag is cleared and the KEYVALID flag set in the AES_SR register. If KEYVALID is not set when BUSY bit is cleared, or if the KEIF flag is set in the AES_ISR register, either the KEYSIZE value is incorrect or an unexpected event occurred during the transfer (such as DPA error, tamper event or KEYVALID cleared before the end of the transfer). When such errors occur, reset both peripherals through their IPRST bits and restart the whole key sharing process.

When the key sharing sequence is completed, the AES is initialized with a valid, shared key. The application can then process data in normal key mode, by writing KMOD[1:0] with 0x0.

Note: This sequence in AES peripheral can be run multiple times (for example, to manage a suspend/resume situation), as long as SAES peripheral is unused and duly remains in key sharing state.

25.4.15 AES data registers and data swapping

Data input and output

A 16-byte data block enters 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 16-byte data block is retrieved from the AES peripheral with four successive 32-bit word reads of 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 four 32-bit words of a 16-byte data block must be stored in the memory consecutively and in big-endian order, that is, with the most significant word on the lowest address. See *Table 188* “no swapping” option for details.

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 DATATYPE[1:0]. The selection applies to both AES input and output.

Note: The data in AES key registers (AES_KEYRx) and initialization vector registers (AES_IVRx) are not sensitive to the swap mode selection.

The AES data swapping feature is summarized in *Table 188* and *Figure 129*.

Table 188. AES data swapping example

DATATYPE[1:0]	Swapping performed	Data block
		System memory data (big-endian)
0x0	No swapping	Block[127..64]: 0x04EEF672 2E04CE96 Block[63..0]: 0x4E6F7720 69732074
		Address @, word[127..96]: 0x04EEF672 Address @ + 0x4, word[95..64]: 0x2E04CE96 Address @ + 0x8, word[63..32]: 0x4E6F7720 Address @ + 0xC, word[31..0]: 0x69732074
0x1	Half-word (16-bit) swapping	Block[63..0]: 0x**4E6F** 7720 **6973** 2074
		Address @, word[63..32]: 0x7720 **4E6F** Address @ + 0x4, word[31..0]: 0x2074 **6973**
0x2	Byte (8-bit) swapping	Block[63..0]: 0x**4E** 6F **77** 20 **69** 73 **20** 74
		Address @, word[63..32]: 0x20**77** 6F**4E** Address @ + 0x4, word[31..0]: 0x74**20** 73**69**

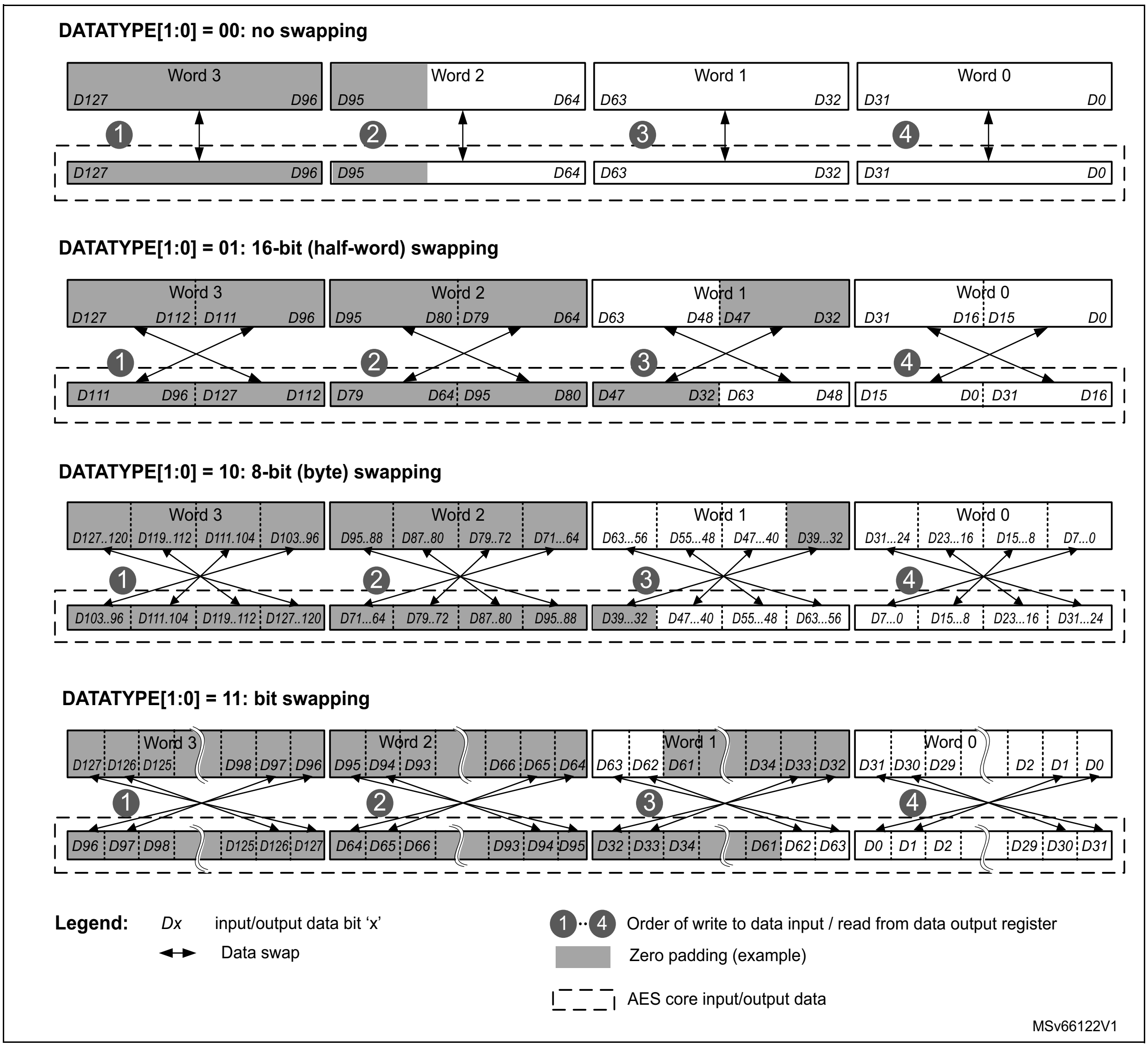
Table 188. AES data swapping example (continued)

DATATYPE[1:0]	Swapping performed	Data block
		System memory data (big-endian)

0x3	Bit swapping	Block[63..32]: 0x4E6F7720
		`0100 1110 0110 1111 0111 0111 0010 0000`
		Block[31..0]: 0x69732074
		`0110 1001 0111 0011 0010 0000 0111 0100`

		Address @, word[63..32]: 0x04EE F672
		`0000 0100 1110 1110 1111 0110 0111 0010`
		Address @ + 0x4, word[31..0]: 0x2E04 CE96
		`0010 1110 0000 0100 1100 1110 1001 0110`

Figure 129. 128-bit block construction according to the data type



Data padding

Figure 129 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:

- 84 message bits, with DATATYPE[1:0] = 0x0
- 48 message bits, with DATATYPE[1:0] = 0x1
- 56 message bits, with DATATYPE[1:0] = 0x2
- 34 message bits, with DATATYPE[1:0] = 0x3

25.4.16 AES key registers

The eight AES_KEYRx write-only registers store the encryption or decryption key information, as shown on *Table 189*. Reads are not allowed for security reason.

Note: In memory and in AES *key registers, keys are stored in little-endian format, with most significant byte on the highest address.*

Table 189. Key endianness in AES_KEYRx registers (128/256-bit keys)

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 registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield.

Write operations to the AES_KEYRx registers are ignored when AES peripheral is enabled (EN bit set). The application must check this before modifying key registers.

The entire key must be written before starting an AES computation.

In normal key mode (KMOD[1:0] at 0x0), the key registers must always be written in either ascending or descending order. The write sequence becomes:

- AES_KEYRx (x = 0 to 3 or x=3 to 0) for KEYSIZE cleared
- AES_KEYRx (x = 0 to 7 or x=7 to 0) for KEYSIZE set

Note: KEYSIZE must be written before the key.

As soon as the first key register is written, the KEYVALID flag is cleared. Once the key registers writing sequence is completed, KEYVALID is set and EN becomes writable. If an error occurs, KEYVALID is cleared and KEIF set (see *Section 25.4.18*).

25.4.17 AES initialization vector registers

The four AES_IVRx registers store the initialization vector (IV) information, as shown in *Table 190*. They can only be written if the AES peripheral is disabled (EN cleared).

Note: In memory and in AES IV registers, initialization vectors are stored in little-endian format, with most significant byte on the highest address.

Table 190. IVI bitfield spread over AES_IVRx registers

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]

Initialization vector information depends on the chaining mode selected. When used, AES_IVRx registers are updated upon each AES computation cycle (useful for managing suspend mode).

The initialization vector registers are not affected by the data swapping feature controlled through DATATYPE[1:0].

25.4.18 AES error management

The AES peripheral manages the errors described in this section.

Read error flag (RDERRF)

Unexpected read attempt of the AES_DOUTR register returns zero, setting the RDERRF flag and the RWEIF flag. RDERRF is triggered during the computation phase or during the input phase.

Note: AES is not disabled when RDERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 25.5: AES interrupts*.

The RDERRF and RWEIF flags are cleared by setting the RWEIF bit of the AES_ICR register.

Write error flag (WRERRF)

Unexpected write attempt of the AES_DINR register is ignored, setting the WRERRF and the RWEIF flags. WRERRF is triggered during the computation phase or during the output phase.

Note: AES is not disabled when WRERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 25.5: AES interrupts*.

The WRERRF and RWEIF flags are cleared by setting the RWEIF bit of the AES_ICR register.

Key error interrupt flag (KEIF)

There are multiple sources of errors that set the KEIF flag of the AES_ISR register and clear the KEYVALID bit of the AES_SR register:

- **Key writing sequence error:** triggered upon detecting an incorrect sequence of writing key registers. See *Section 25.4.16: AES key registers* for details.
- **Key sharing size mismatch error:** triggered when KMOD[1:0] is at 0x2 and KEYSIZE in AES peripheral does not match KEYSIZE in SAES peripheral.

- **Key sharing error:** triggered upon failing transfer of SAES shared key to AES peripheral. See *Section 25.4.14: AES key sharing with secure AES co-processor* for details.

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 25.5: AES interrupts*.

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 disabling and re-configuring AES.

25.5 AES interrupts

There are multiple individual maskable interrupt sources generated by the AES peripheral to signal the following events:

- computation completed (CCF)
- read error (RDERRF)
- write error (WRERRF)
- key error (KEIF)

See *Section 25.4.18: AES error management* for details on AES errors.

These sources are combined into a common interrupt signal from the AES peripheral that connects to the Cortex® CPU interrupt controller. Application can enable or disable AES interrupt sources individually by setting/clearing the corresponding enable bit of the AES_IER register.

The status of the individual maskable interrupt sources can be read from the AES_ISR register. They are cleared by setting the corresponding bit of the AES_ICR register.

Table 191 gives a summary of the available features.

Table 191. AES interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable bit	Interrupt clear method
AES	computation completed flag	CCF	CCFIE	set CCF(1)
	read error flag	RDERRF(2)	RWEIE	set RWEIF(1)
	write error flag	WRERRF(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.

25.6 AES DMA requests

The AES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the DMAINEN and DMAOUTEN bits of the AES_CR register. When key derivation is selected (MODE[1:0] is at 0x1), setting those bits has no effect.

AES only supports single DMA requests.

Suspend and resume operations are not supported in DMA mode.

Detailed usage of DMA with AES can be found in *Appending data using DMA* subsection of *Section 25.4.5: AES encryption or decryption typical usage.*

Data input using DMA

Setting DMAINEN enables DMA writing into AES. AES then initiates, during the input phase, a set of single DMA requests for each 16-byte data block to write to the AES_DINR register (quadruple 32-bit word, MSB first).

Note: According to the algorithm and the mode selected, special padding / ciphertext stealing might be required (see Section 25.4.7).

Data output using DMA

Setting DMAOUTEN enables DMA reading from AES. AES then initiates, during the output phase, a set of single DMA requests for each 16-byte data block to read from the AES_DOUTR register (quadruple 32-bit word, MSB first).

After the output phase, at the end of processing of a 16-byte data block, AES switches automatically to a new input phase for the next data block, if any.

In DMA mode, 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 managing the data transfers by software.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Stopping DMA transfers

All DMA request signals are de-asserted when AES is disabled (EN cleared) or the DMA enable bit (DMAINEN for input data, DMAOUTEN for output data) is cleared.

25.7 AES processing latency

The following tables provide the 16-byte data block processing latency per operating mode.

Table 192. Processing latency for ECB, CBC and CTR

Key size	Mode of operation	Chaining algorithm	Clock cycles
128-bit	Encryption or decryption(1)	ECB, CBC, CTR	51
	Key preparation	-	59
256-bit	Encryption or decryption(1)	ECB, CBC, CTR	75
	Key preparation	-	82

1. Excluding key preparation time (ECB and CBC only).

Table 193. Processing latency for GCM and CCM (in clock cycles)

Key size	Mode of operation	Chaining algorithm	Initialization phase	Header phase(1)	Payload phase(1)	Final phase(1)
128-bit	Encryption/ Decryption	GCM	64	35	51	59
		CCM	63	55	114	58
256-bit	Encryption/ 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).

25.8 AES registers

The registers are accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.

25.8.1 AES control register (AES_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPRST	Res.	Res.	Res.	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 are 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 kept 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. KEYSIZE must be correctly initialized when setting KMOD[1:0] different from zero.

0x0: Normal key mode. Key registers are freely usable.

0x2: Shared key mode. If shared key mode is properly initialized in SAES peripheral, the AES peripheral automatically loads its key registers with the data stored in the SAES key registers. The key value is available in AES 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.

Others: Reserved

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN 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

This padding information must be filled by software before processing the last block of GCM payload encryption or CCM payload decryption, otherwise authentication tag computation is incorrect.

0x0: All bytes are valid (no padding)

0x1: Padding for the last LSB byte

...

0xF: Padding for the 15 LSB bytes of last block.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **KEYSIZE:** Key size selection

This bitfield defines the key length in bits of the key used by AES.
0: 128-bit
1: 256-bit
Attempts to write the bit are ignored when BUSY is set, as well as when the EN 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, applicable only with GCM, GMAC or CCM chaining modes.
0x0: Initialization phase
0x1: Header phase
0x2: Payload phase
0x3: Final phase

Bit 12 **DMAOUTEN**: DMA output enable

This bit enables automatic generation of DMA requests during the data phase, for outgoing data transfers from AES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bit 11 **DMAINEN**: DMA input enable

This bit enables automatic generation of DMA requests during the data phase, for incoming data transfers to AES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bits 10:7 Reserved, must be kept at reset value.

Bits 16, 6:5 **CHMOD[2:0]**: Chaining mode

This bitfield selects the AES chaining mode:
0x0: Electronic codebook (ECB)
0x1: Cipher-block chaining (CBC)
0x2: Counter mode (CTR)
0x3: Galois counter mode (GCM) and Galois message authentication code (GMAC)
0x4: Counter with CBC-MAC (CCM)
others: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 4:3 **MODE[1:0]**: Operating mode

This bitfield selects the AES operating mode:
0x0: Encryption
0x1: Key derivation (or key preparation), for ECB/CBC decryption only
0x2: Decryption
0x3: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 2:1 **DATATYPE[1:0]**: Data type

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. This swapping is defined in *Section 25.4.15: AES data registers and data swapping*.

0x0: No swapping (32-bit data).

0x1: Half-word swapping (16-bit data)

0x2: Byte swapping (8-bit data)

0x3: Bit-level swapping

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bit 0 **EN**: 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. When KMOD[1:0] is different from 0x0, using IPRST bit is recommended instead.

This bit is automatically cleared by hardware upon the completion of the key preparation (MODE[1:0] at 0x1) and upon the completion of GCM/GMAC/CCM initialization phase.

– The bit cannot be set as long as KEYVALID is cleared

25.8.2 AES status register (AES_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.	KEYVALID	Res.	Res.	Res.	BUSY	WRERRF	RDERRF	Res.
								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 key of size defined by KEYSIZE is loaded in AES_KEYRx key registers.

0: Key not valid

1: Key valid

The EN bit can only be set when KEYVALID is set.

In normal mode when KMOD[1:0] is at zero, the key must be written in the key registers in the correct sequence, otherwise the KEIF flag is set and KEYVALID remains cleared.

When KMOD[1:0] is different from zero, the BUSY flag is automatically set by AES. When the key is loaded successfully, BUSY is cleared and KEYVALID set. Upon an error, KEIF is usually set, BUSY cleared and KEYVALID remains cleared.

If set, KEIF must be cleared through the AES_ICR register, otherwise KEYVALID cannot be set. See the KEIF flag description for more details.

For further information on key loading, refer to *Section 25.4.16: 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.

0: Idle

1: Busy

AES is flagged as idle when disabled (when EN is low) or when the last processing is completed.

AES is flagged as busy when processing a block data, preparing a key (ECB or CBC decryption only), or transferring a shared key from the SAES peripheral.

When GCM encryption payload phase is selected, this flag must be at zero before suspending current process to manage a higher-priority message.

Bit 2 **WRERRF**: Write error flag

This bit is set when an unexpected write to the AES_DINR register occurred. When set WRERRF bit has no impact on the AES operations.

0: No error

1: Unexpected write to AES_DINR register occurred during computation or data output phase.

The flag setting generates an interrupt if the RWEIE bit of the AES_IER register is set.

The flag is cleared by setting the RWEIF bit of the AES_ICR register.

Bit 1 **RDERRF**: Read error flag

This bit is set when an unexpected read to the AES_DOUTR register occurred. When set RDERRF bit has no impact on the AES operations.

0: No error

1: Unexpected read to AES_DOUTR register occurred during computation or data input phase.

The flag setting generates an interrupt if the RWEIE bit of the AES_IER register is set.

The flag is cleared by setting the RWEIF bit of the AES_ICR register.

Bit 0 Reserved, must be kept at reset value.

25.8.3 AES data input register (AES_DINR)

Address offset: 0x008

Reset value: 0x0000 0000

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]**: Data input

A four-fold sequential write to this bitfield during the Input phase results in writing a complete 16-bytes 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 16-bytes input buffer.

Reads return zero.

25.8.4 AES data output register (AES_DOUTR)

Address offset: 0x00C

Reset value: 0x0000 0000

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]**: Data output

This read-only bitfield fetches a 32-bit output buffer. A four-fold sequential read of this bitfield, upon the computation completion (CCF flag set), virtually reads a complete 16-byte 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].

25.8.5 AES key register 0 (AES_KEYR0)

Address offset: 0x010

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]

These are bits [31:0] of the write-only bitfield KEY[255:0] AES encryption or decryption key, depending on the MODE[1:0] bitfield of the AES_CR register.

Writes to AES_KEYRx registers are ignored when AES is enabled (EN bit set). When the key comes from the SAES peripheral (KMOD[1:0] at 0x2), writes to key registers are also ignored and they result in setting the KEIF bit of the AES_ISR register.

With KMOD[1:0] at 0x0, a special writing sequence is required. In this sequence, any valid write to AES_KEYRx register clears the KEYVALID flag except for the sequence-completing write that sets it. Also refer to the description of the KEYVALID flag in the AES_SR register.

25.8.6 AES key register 1 (AES_KEYR1)

Address offset: 0x014

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 and for information relative to writing AES_KEYRx registers.

25.8.7 AES key register 2 (AES_KEYR2)

Address offset: 0x018

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 and for information relative to writing AES_KEYRx registers.

25.8.8 AES key register 3 (AES_KEYR3)

Address offset: 0x01C

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 and for information relative to writing AES_KEYRx registers.

25.8.9 AES initialization vector register 0 (AES_IVR0)

Address offset: 0x020

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]

AES_IVRx registers store the 128-bit initialization vector or the nonce, depending on the chaining mode selected. This value is updated by hardware after each computation round (when applicable). Write to this register is ignored when EN bit is set in AES_CR register

25.8.10 AES initialization vector register 1 (AES_IVR1)

Address offset: 0x024

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.

25.8.11 AES initialization vector register 2 (AES_IVR2)

Address offset: 0x028

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.

25.8.12 AES initialization vector register 3 (AES_IVR3)

Address offset: 0x02C

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.

25.8.13 AES key register 4 (AES_KEYR4)

Address offset: 0x030

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 and for information relative to writing AES_KEYRx registers.

25.8.14 AES key register 5 (AES_KEYR5)

Address offset: 0x034

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 and for information relative to writing AES_KEYRx registers.

25.8.15 AES key register 6 (AES_KEYR6)

Address offset: 0x038

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 and for information relative to writing AES_KEYRx registers.

25.8.16 AES key register 7 (AES_KEYR7)

Address offset: 0x03C

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 and for information relative to writing AES_KEYRx registers.

25.8.17 AES suspend registers (AES_SUSPRx)

Address offset: 0x040 + 0x4 * 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
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]**: Suspend data

AES_SUSPRx registers contain the complete internal register states of the AES when the GCM, GMAC or CCM processing of the current task is suspended to process a higher-priority task. Refer to *Section 25.4.8: AES suspend and resume operations* for more details.

Clearing EN bit of the AES_CR register clears this register to zero.

AES_SUSPRx registers are not used in other chaining modes than GCM, GMAC or CCM.

25.8.18 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)

25.8.19 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 the key information fails to load into key registers.

0: No key error detected

1: Key information failed to load into key registers

The flag setting generates an interrupt if the KEIE bit of the AES_IER register is set. It also clears the key registers and the KEYVALID flag in the AES_SR register.

The flag is cleared by setting the corresponding bit of the AES_ICR register.

KEIF is raised upon any of the following events:

–AES_KEYRx register write does not respect the correct order. (For KEYSIZE cleared, AES_KEYR0 then AES_KEYR1 then AES_KEYR2 then AES_KEYR3 register, or reverse. For KEYSIZE set, 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).

–AES fails to load the key shared by the SAES peripheral (KMOD[1:0] = 0x2).

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 RDERRF or a WRERRF error flag is set in the AES_SR register.

0: No read or write error detected

1: Read or write error detected

The flag setting generates an interrupt if the RWEIE bit of the AES_IER register is set.

The flag is cleared by setting the corresponding bit of the AES_ICR register.

The flags has no meaning when key derivation mode is selected.

See the AES_SR register for details.

Bit 0 **CCF:** Computation complete flag

This flag indicates whether the computation is completed. It is significant only when the DMAOUTEN bit is cleared, and it may stay high when DMAOUTEN is set.

0: Not completed

1: Completed

The flag setting generates an interrupt if the CCFIE bit of the AES_IER register is set.

The flag is cleared by setting the corresponding bit of the AES_ICR register.

25.8.20 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 clears both RDERRF and WRERRF flags in the AES_SR register.

Bit 0 **CCF:** Computation complete flag clear

Setting this bit clears the CCF status bit of the AES_ISR register.

25.8.21 AES register map

Table 194. AES 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	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	WRERRF	RDERRF	Res.
	Reset value																									0				0	0	0	
0x008	AES_DINR	DIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	AES_DOUTR	DOUT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	AES_KEYR0	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	AES_KEYR1	KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 194. AES 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	AES_KEYR2	KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	AES_KEYR3	KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	AES_IVR0	IVI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	AES_IVR1	IVI[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	AES_IVR2	IVI[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	AES_IVR3	IVI[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	AES_KEYR4	KEY[159:128]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	AES_KEYR5	KEY[191:160]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	AES_KEYR6	KEY[223:192]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	AES_KEYR7	KEY[255:224]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	AES_SUSPR0	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_SUSPR1	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_SUSPR2	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_SUSPR3	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_SUSPR4	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_SUSPR5	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_SUSPR6	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_SUSPR7	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.

Table 194. AES 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	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
0x309-0x3FF	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: Memory organization* for the register boundary addresses.

26 Secure AES coprocessor (SAES)

This section only applies to STM32WBA52/54/55xx products.

26.1 Introduction

The secure AES coprocessor (SAES) encrypts or decrypts data in compliance with the advanced encryption standard (AES) defined by NIST. It incorporates a protection against side-channel attacks (SCA), including differential power analysis (DPA), certified SESIP and PSA security assurance level 3.

SAES supports ECB, CBC, CTR, GCM, GMAC, and CCM 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 master keys (boot hardware key BHK and derived hardware unique key DHUK), usable but not readable by the application.

The peripheral supports DMA single transfers for incoming and outgoing data (two DMA channels are required). It is hardware-linked with the true random number generator (TRNG) and with the AES peripheral.

26.2 SAES main features

- Compliant with NIST FIPS publication 197 *"Advanced encryption standard (AES)"* (November 2001)
- 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
- Protection against side-channel attacks (SCA), incl. differential power analysis (DPA), certified SESIP and PSA security assurance level 3
- 128-bit data block processing, supporting cipher key lengths of 128-bit and 256-bit
  - 480 or 680 clock cycle latency in ECB mode for processing one 128-bit block with, respectively, 128-bit or 256-bit key
- Hardware secret key encryption/ decryption (Wrapped-key mode)
- Using dedicated key bus, optional key sharing with faster AES peripheral (shared-key mode), controlled by SAES
- Integrated key scheduler to compute the last round key for ECB/CBC decryption
- 256-bit of write-only registers for storing cryptographic keys (eight 32-bit registers)
  - Optional 128-bit or 256-bit hardware loading of two hardware secret keys (BHK, DHUK) that can be XOR-ed together

- Security context enforcement for keys
- 128-bit of registers for storing initialization vectors (four 32-bit registers)
- 32-bit buffer for data input and output
- Automatic data flow control supporting two direct memory access (DMA) channels, one for incoming data, one for processed data. Only single transfers are supported.
- Data-swapping logic to support 1-, 8-, 16-, or 32-bit data
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.
- Possibility for software (in CPU mode only, not in DMA mode) to suspend a message if SAES needs to process another message with a higher priority, then resume the original message

26.3 SAES implementation

The devices have one SAES peripheral, implemented as per the following table. It shares the key with the AES peripheral. For comparison, the AES peripheral is also included in the table.

Table 195. AES versus SAES features

Modes or features[1]	AES	SAES
ECB, CBC chaining	X	X
CTR, CCM, GCM chaining	X	X
AES 128-bit ECB encryption in cycles	51	480
DHUK and BHK key selection	-	X
Resistance to side-channel attacks	-	X
Shared key between SAES and AES	X	
Key sizes in bits	128, 256	128, 256

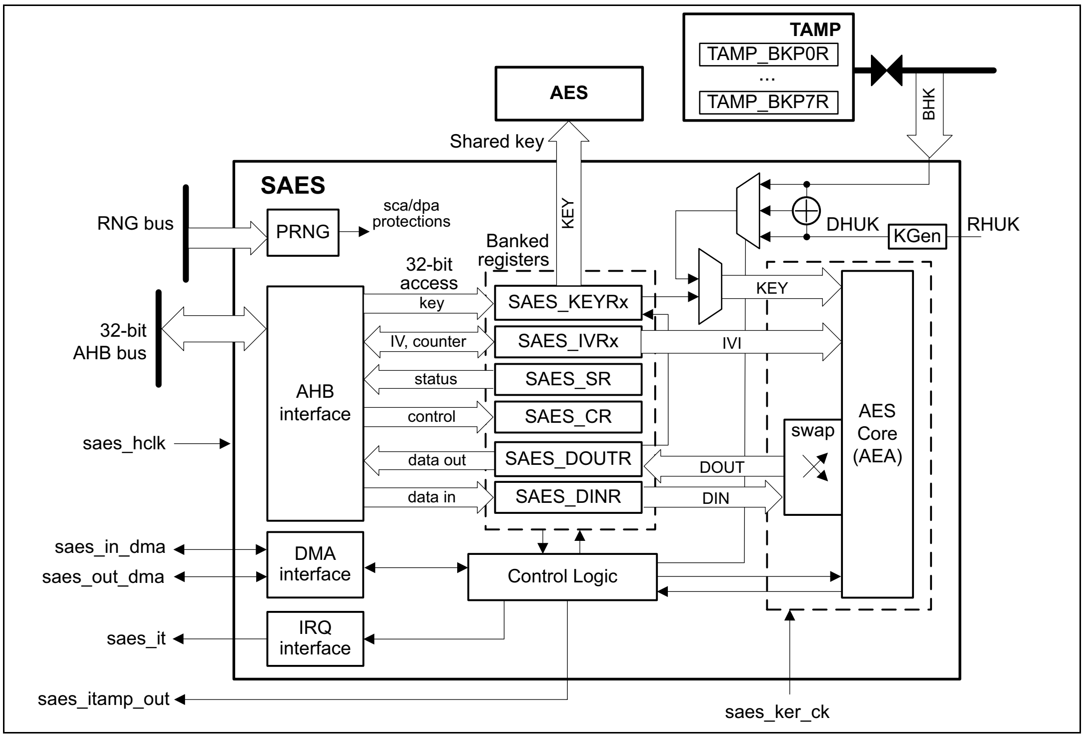
1. X = supported.

26.4 SAES functional description

26.4.1 SAES block diagram

Figure 130 shows the block diagram of SAES.

Figure 130. SAES block diagram



Note: AES represents the AES peripheral.

26.4.2 SAES internal signals

Table 196 describes the user relevant internal signals interfacing the SAES peripheral.

Table 196. SAES internal input/output signals

Signal name	Signal type	Description
saes_hclk	Input	AHB bus clock
saes_ker_ck	Input	SAES kernel clock.
saes_it	Output	SAES interrupt request
saes_in_dma	Input/Output	SAES incoming data DMA single request/acknowledge
saes_out_dma	Input/Output	SAES processed data DMA single request/acknowledge
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.
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 26.4.17* for details).

26.4.3 SAES reset and clocks

The SAES peripheral is clocked by the AHB bus clock. It has a dedicated reset bit and a dedicated kernel clock, controlled through the RCC.

The SAES reset in RCC is mandatory when a hardware tamper event occurs in SAES. Refer to *Managing tamper errors* in *Section 26.4.19* for details.

After clocking then releasing the reset of the SAES peripheral in the RCC, SAES automatically draws random numbers from the RNG, setting the BUSY bit of the SAES_SR register. Refer to the *RNG error interrupt flag (RNGEIF)* in *Section 26.4.19* for details.

This background task must be completed before the device enters a low-power mode.

26.4.4 SAES symmetric cipher implementation

The secure AES coprocessor (SAES) is a 32-bit AHB peripheral that encrypts or decrypts 16-byte blocks of data using the advanced encryption standard (AES). It also implements a set of approved AES symmetric key security functions summarized in *Table 197*. Those functions can be certified NIST PUB 140-3.

Table 197. SAES approved symmetric key functions

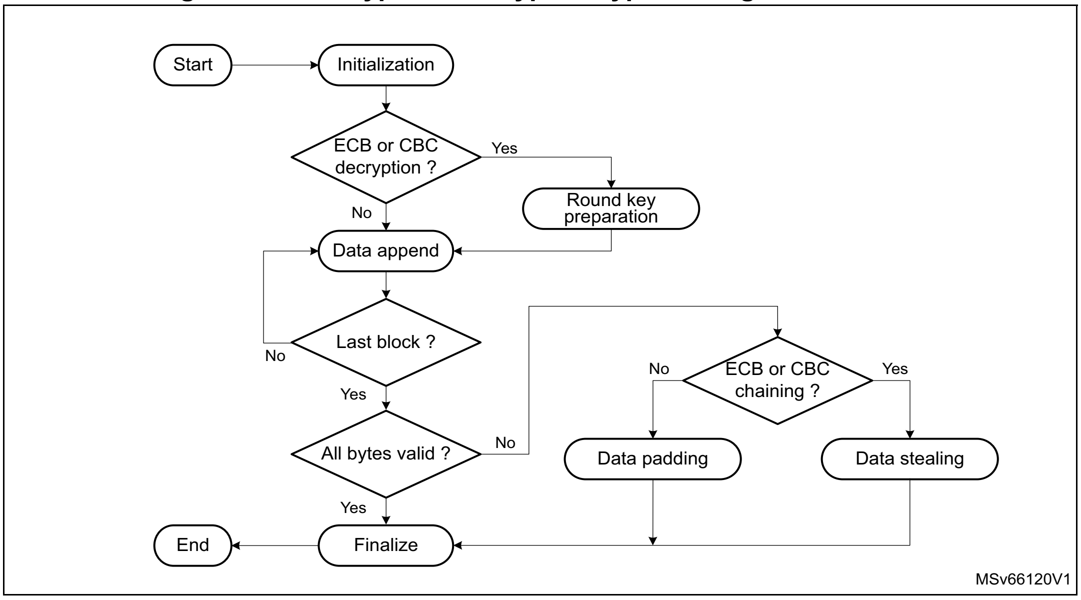
Operations	Algorithm	Specification	Key bit lengths	Chaining modes
Encryption, decryption	AES	FIPS PUB 197 NIST SP800-38A	128, 256	ECB, CBC, CTR
Authenticated encryption or decryption		NIST SP800-38C NIST SP800-38D		GCM, CCM
Cipher-based message authentication code		NIST SP800-38D		GMAC

SAES can be used directly by the CPU, or indirectly, using two DMA channels (one for the plaintext, one for the ciphertext).
It is possible to suspend then resume any SAES processing, following the sequence described in *Section 26.4.8*.

26.4.5 SAES encryption or decryption typical usage

The following figure shows a typical operation for encryption or decryption.

Figure 131. Encryption/ decryption typical usage



Initialization

The SAES peripheral is initialized according to the chaining mode. Refer to *Section 26.4.9: SAES basic chaining modes (ECB, CBC)* and *Section 26.4.10: SAES counter (CTR) mode* for details.

Data append

This section describes different ways of appending data for processing. For ECB or CBC chaining modes, refer to *Section 26.4.7: SAES ciphertext stealing and data padding* if the size of data to process is not a multiple of 16 bytes. The last block management in these cases is more complex than what is described in this section.

Appending data using the CPU in polling mode

This method uses flag polling to control the data append through the following sequence:

1. Enable the SAES peripheral when KEYVALID is set, by setting the EN bit of the SAES_CR register (if not already done).
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_ISR register, then read the four data words from the SAES_DOUTR register.
   c) Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
   d) If the next processing block is the last block, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (using

NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong).

3. As the data block just processed is the last block of the message, optionally discard the data that is not part of the message/payload, then disable the SAES peripheral by clearing EN.

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.

NPBLB[3:0] bitfield is not used in header phase of GCM, GMAC and CCM chaining modes.

Appending data using the CPU in interrupt mode

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 when KEYVALID is set, by setting EN (if not already done).
3. Write first four input data words into the SAES_DINR register.
4. Handle the data in the SAES interrupt service routine. Upon each 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_ICR register.
   c) If the next processing block is the last block of the message, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (through NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong). Then proceed with point 4e).
   d) If the data block just processed is the last block of the message, optionally discard the data that are not part of the message/payload, then disable the SAES peripheral by clearing EN 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.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

Appending data using DMA

With this method, all the transfers and processing are managed by DMA and SAES. Proceed as follows:

1. If the last block of the message to process is shorter than 16 bytes, prepare the last four-word data block 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 SAES peripheral input and the processed data from the SAES peripheral output to the memory, as described in *Section 26.6: SAES DMA requests*. Configure the DMA controller so as to generate an interrupt on transfer completion. For GCM payload encryption or CCM payload decryption, the DMA transfer **must not** include the last four-word block if padded with zeros. The sequence described in *Appending data using the CPU in polling mode* must be used instead for this last block, because the

NPBLB[3:0] bitfield must be set up before processing the block, for SAES to compute a correct tag.

3. Enable the SAES peripheral when KEYVALID is set, by setting EN (if not already done).
4. Enable DMA requests, by setting DMAINEN and DMAOUTEN.
5. Upon DMA interrupt indicating the transfer completion, get the SAES-processed data from the memory.

When appending data using DMA, the suspend/resume operation as described in *Section 26.4.8* is not supported.

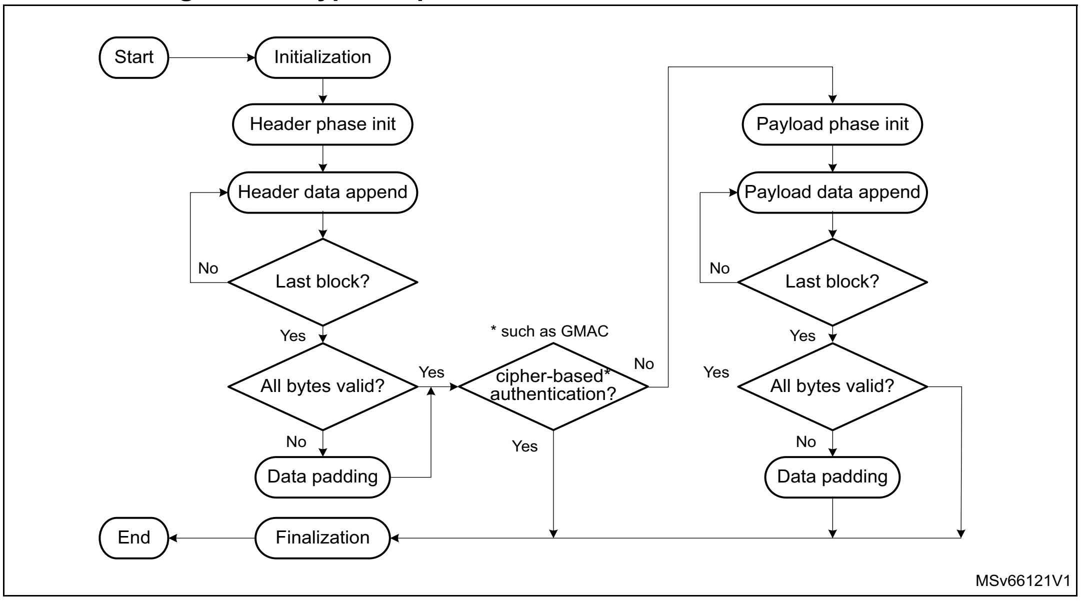
Note: The CCF flag has no use with this method because the reading of the SAES_DOUTR register is managed by DMA automatically, without any software action, at the end of the computation phase.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

26.4.6 SAES authenticated encryption, decryption, and cipher-based message authentication

The following figure shows a typical operation for authenticated encryption or decryption, and for cipher-based message authentication.

Figure 132. Typical operation with authentication



Section 26.4.11: SAES Galois/counter mode (GCM) and *Section 26.4.13: SAES counter with CBC-MAC (CCM)* describe detailed sequences supported by SAES.

Cipher-based message authentication flow omits the payload phase, as shown in the figure. Detailed sequence supported by SAES is described in *Section 26.4.12: SAES Galois message authentication code (GMAC)*.

26.4.7 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 (16 bytes), the application must use ciphertext stealing techniques 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 SAES does not implement 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, in modes other than ECB or CBC, an incomplete input data block (that is, a block with input data shorter than 16 bytes) 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. Since SAES does not implement automatic data padding operation to the *last block*, the application must follow the recommendation given in this document to manage messages the size of which is not a multiple of 16 bytes.

26.4.8 SAES suspend and resume operations

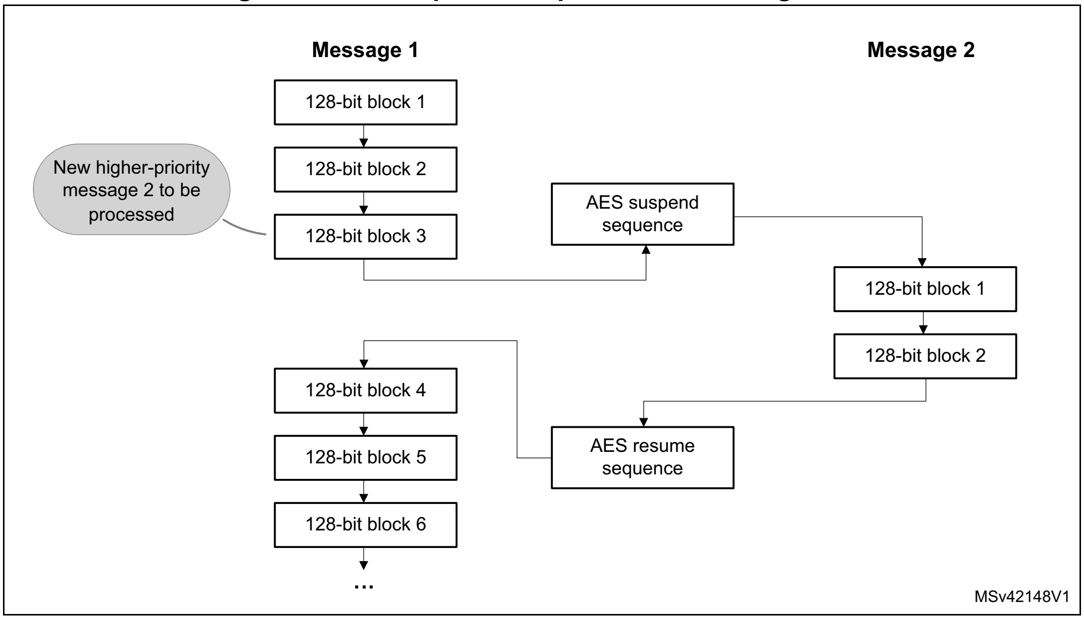
A message can be suspended to process another message with a higher priority. When the higher-priority message is sent, the suspended message can resume. This applies to both encryption and decryption mode.

Suspend and resume operations do not break the chaining operation. The message processing can resume as soon as SAES is enabled again, to receive a next data block.

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

Figure 133 gives an example of suspend and resume operations: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

Figure 133. Example of suspend mode management



A detailed description of suspend and resume operations is in the sections dedicated to each chaining mode.

26.4.9 SAES basic chaining modes (ECB, CBC)

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. When decrypting in ECB, a special key scheduling is required before processing the first block.

Figure 134 and *Figure 135* describe the electronic codebook (ECB) chaining implementation in encryption and in decryption, respectively. To select ECB chaining mode, write CHMOD[2:0] with 0x0.

Figure 134. ECB encryption

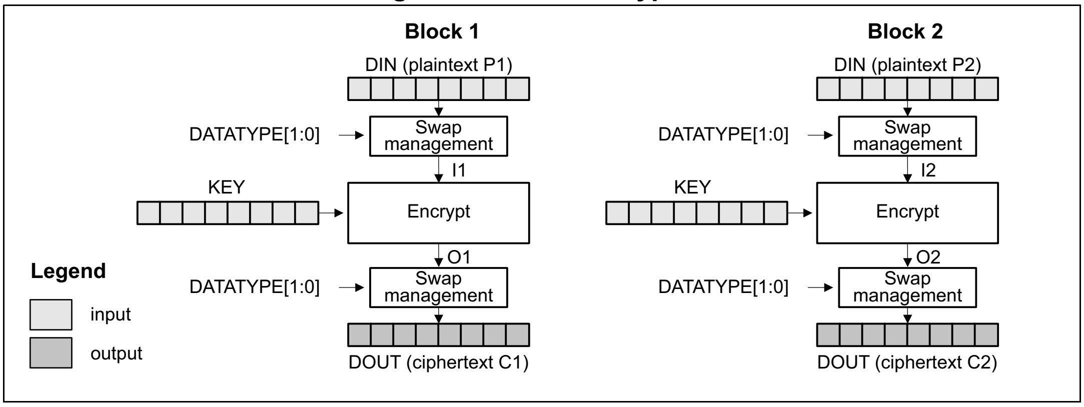
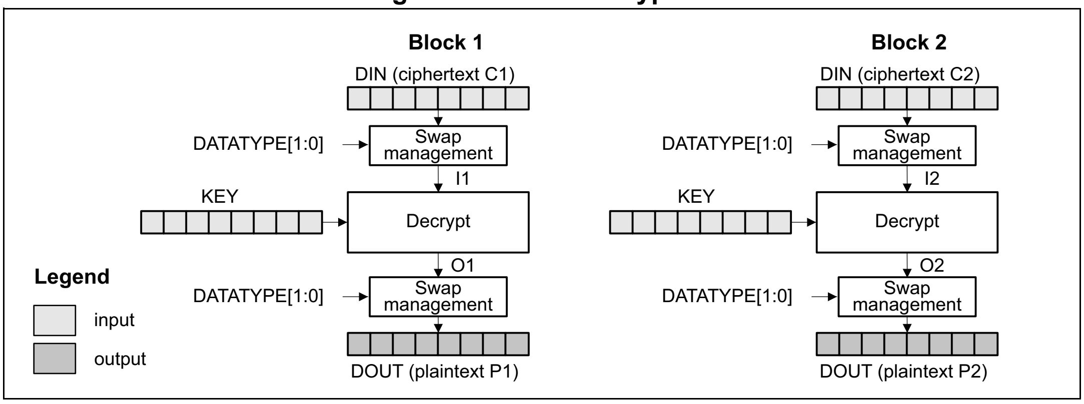


Figure 135. ECB decryption



In CBC encryption 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. When decrypting in CBC, a special key scheduling is required before processing the first block.

Figure 136 and *Figure 137* describe the cipher block chaining (CBC) implementation in encryption and in decryption, respectively. To select this chaining mode, write CHMOD[2:0] with 0x1.

Figure 136. CBC encryption

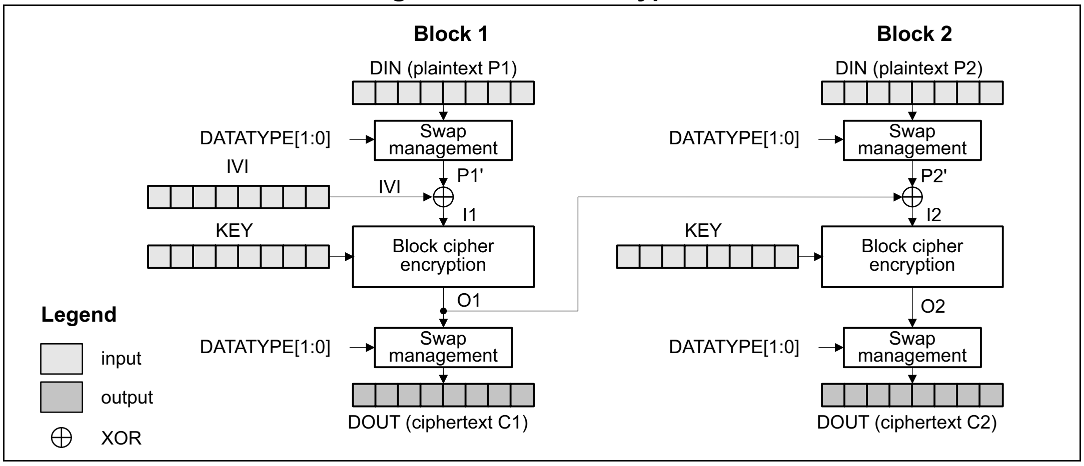
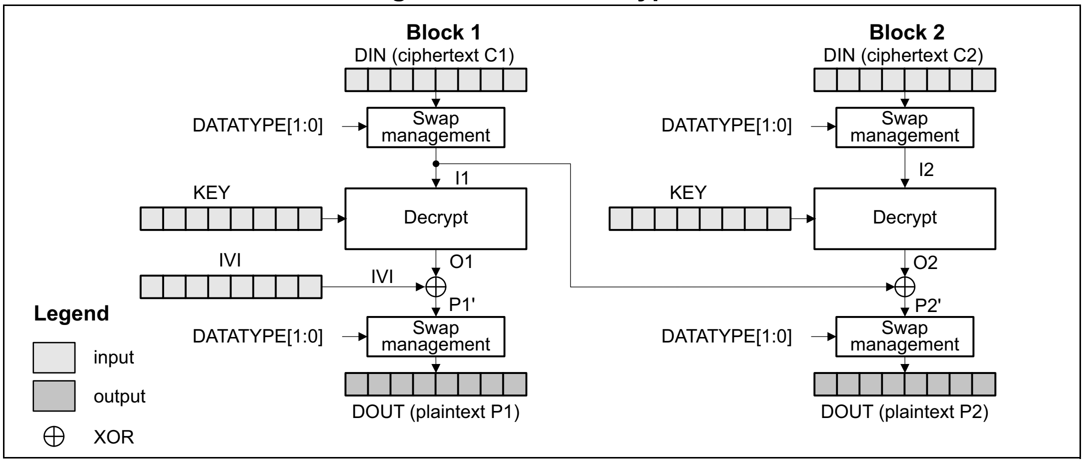


Figure 137. CBC decryption



For more details, refer to NIST Special Publication 800-38A, *Recommendation for Block Cipher Modes of Operation.*

ECB and CBC encryption process

This process is described in *Section 26.4.5*, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follows:
   – Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *encryption* mode (write MODE[1:0] with 0x0).
   – Configure the data type, through DATATYPE[1:0].
   – Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (different secure attribute), the KEYPROT bit must also be set.
   – Select normal key mode by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 26.4.14* (wrapped keys) and *Section 26.4.15* (shared keys).
4. Write the initialization vector into the SAES_IVRx registers if CBC mode is selected in the previous step.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 26.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Append cleartext data:
   a) If it is the second-last or the last block and the plaintext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 26.4.7*.
   b) Append the cleartext block into SAES as described in *Section 26.4.5*, then read the SAES_DOUTR register four times to save the ciphertext block.
   c) Repeat the step *b)* until the third-last plaintext block is encrypted. For the last two blocks, follow the steps *a)* and *b)*.
9. Finalize the sequence: disable the SAES peripheral, by clearing EN.

ECB/CBC decryption process

This process is described in *Section 26.4.5*, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follows:
   – Select the *key derivation* mode (write MODE[1:0] with 0x1). The CHMOD[2:0] bitfield is not significant during this operation.
   – Configure the data type, through DATATYPE[1:0].
   – Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (different secure attribute), the KEYPROT bit must also be set.
   – Select normal key mode by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 26.4.14* (wrapped keys) and *Section 26.4.15* (shared keys).

4. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 26.4.17: SAES key registers* for details.
5. Wait until KEYVALID is set (the key loading completed).
6. Enable the SAES peripheral, by setting EN. The peripheral immediately starts an AES round for key preparation.
7. Wait until the CCF flag in the SAES_ISR register is set.
8. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register. The decryption key is available in the AES core and SAES is disabled automatically.
9. Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *decryption* mode (write MODE[1:0] with 0x2). Do not change other parameters.
10. Write the initialization vector into the SAES_IVRx registers if CBC mode is selected in the previous step.
11. Enable the SAES peripheral, by setting EN.
12. Append encrypted data:
    a) If it is the second-last or the last block and the ciphertext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 26.4.7*.
    b) Append the ciphertext block into SAES as described in *Section 26.4.5*, then read the SAES_DOUTR register four times to save the cleartext block (MSB first).
    c) Repeat the step *b)* until the third-last ciphertext block is decrypted. For the last two blocks, follow the steps *a)* and *b)*.
13. Finalize the sequence: disable the SAES peripheral, by clearing EN.

Suspend/resume operations in ECB/CBC modes

The following sequences are valid for normal key mode (KMOD[1:0] at 0x0).

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

To suspend the processing of a message, proceed as follows:

1. Wait until the CCF flag in the SAES_ISR register is set (computation completed).
2. Read four times the SAES_DOUTR register to save the last processed block.
3. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
4. Save initialization vector registers (only required in CBC mode as the SAES_IVRx registers are altered during the data processing).
5. Disable the SAES peripheral, by clearing EN.
6. Save the SAES_CR register and clear the key registers if they are not needed, to process the higher-priority message.

To resume the processing of a message, proceed as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Restore the SAES_CR register (with correct KEYSIZE) then restore the SAES_KEYRx registers. For KEYSEL[2:0] selecting a key source different from key registers, refer to *Section 26.4.17: SAES key registers* for details.
3. Prepare the decryption key, as described in *ECB/CBC decryption process* (only required for ECB or CBC decryption).

4. Restore the SAES_IVRx registers, using the saved configuration (only required in CBC mode).
5. Enable the SAES peripheral, by setting EN.

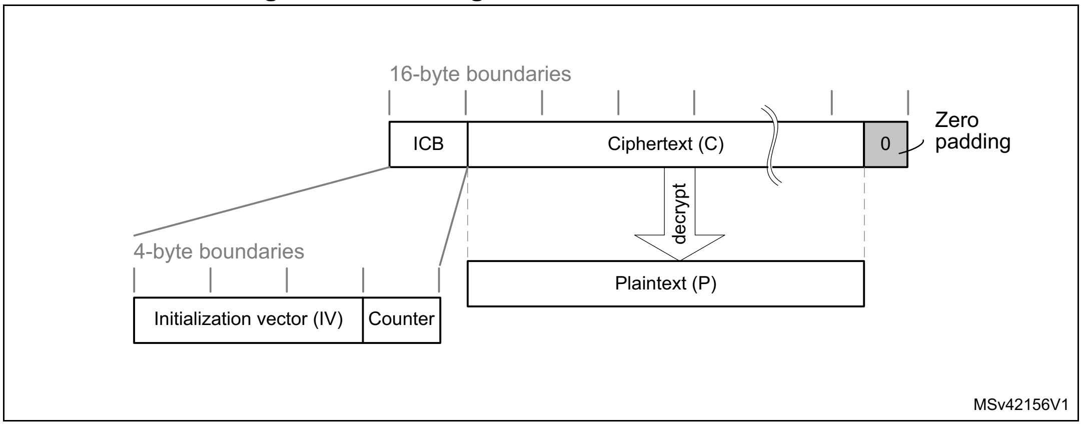
Note: It is not required to save the key registers as the application knows the original key.

26.4.10 SAES counter (CTR) mode

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. Unlike with ECB and CBC modes, no key scheduling is required for the CTR decryption since the AES core is always used in encryption mode.

A typical message construction in CTR mode is given in *Figure 138*.

Figure 138. Message construction in CTR mode



The structure of this message is:

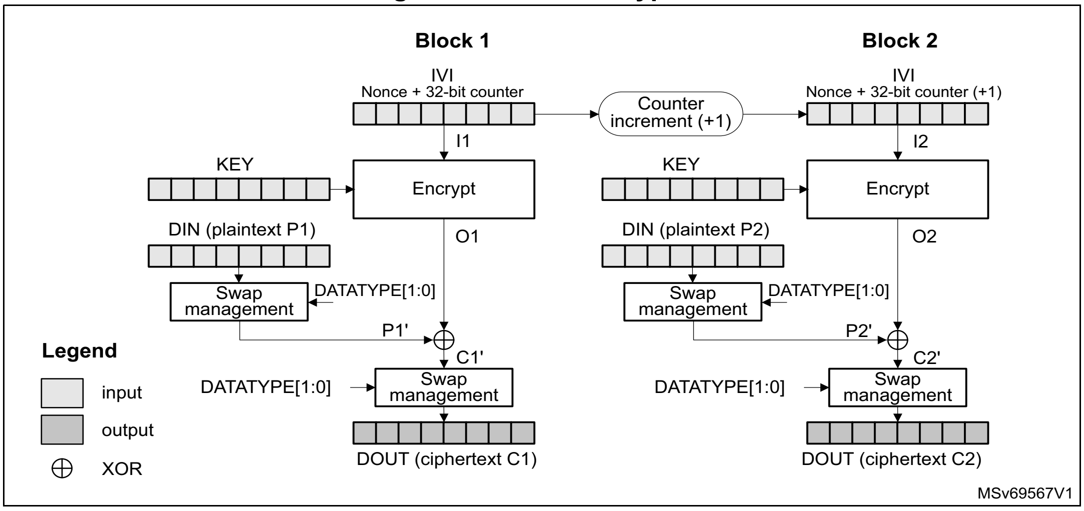
- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - **Initialization vector** (IV): a 96-bit value that must be unique for each encryption cycle with a given key.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. The 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.

For more details, refer to NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.*

CTR encryption and decryption

Figure 139 describes the counter (CTR) chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x2.

Figure 139. CTR encryption



Initialization vectors in SAES must be initialized as shown in *Table 198*.

Table 198. Counter mode initialization vector definition

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0] 32-bit counter = 0x0001

CTR encryption and decryption process

This process is described in *Section 26.4.5*, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   - Select CTR chaining mode (write CHMOD[2:0] with 0x2) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (different secure attribute), the KEYPROT bit must also be set.
   - Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 26.4.14* (wrapped keys) and *Section 26.4.15* (shared keys).
4. Write the initialization vector into the SAES_IVRx registers according to *Table 198*.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 26.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the SAES peripheral, by setting EN.

8. Append data:
   a) If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in the block is less than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into SAES as described in *Section 26.4.5*, then read the SAES_DOUTR register four times to save the resulting block (MSB first).
   c) Repeat the step *b)* until the second-last block is processed. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the message when the last block is smaller than 16 bytes.
9. Finalize the sequence: disable the SAES peripheral, by clearing EN.

Suspend/resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher-priority message, then resume the interrupted message. Detailed CBC suspend and resume sequence is described in *Section 26.4.9: SAES basic chaining modes (ECB, CBC)*.

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

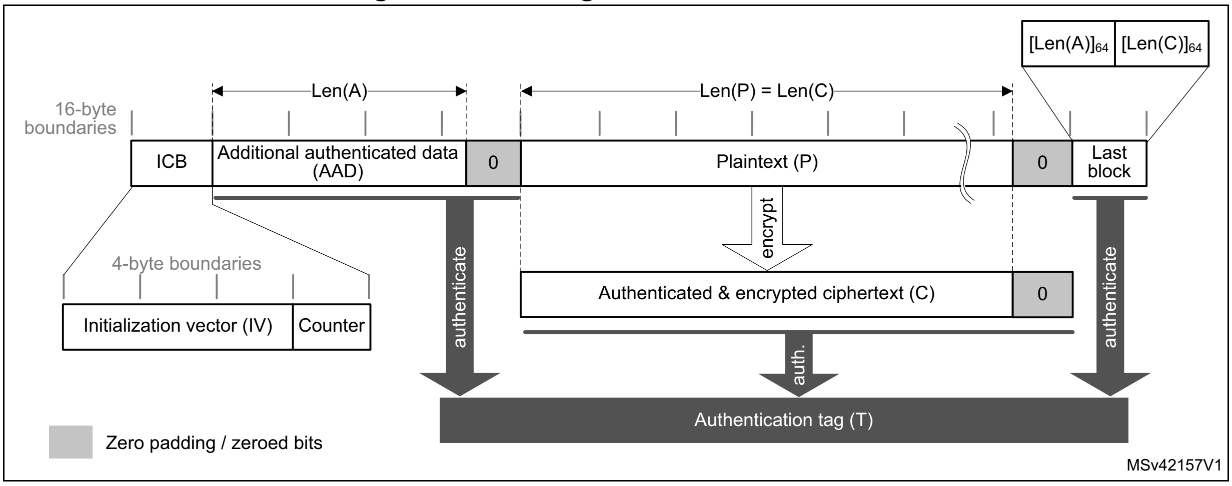
Note: Like for CBC mode, the IV registers must be reloaded during the resume operation.

26.4.11 SAES Galois/counter mode (GCM)

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).

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. The following figure shows a typical message construction in GCM mode.

Figure 140. 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. 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 16-byte blocks.
- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in *Table 200*.

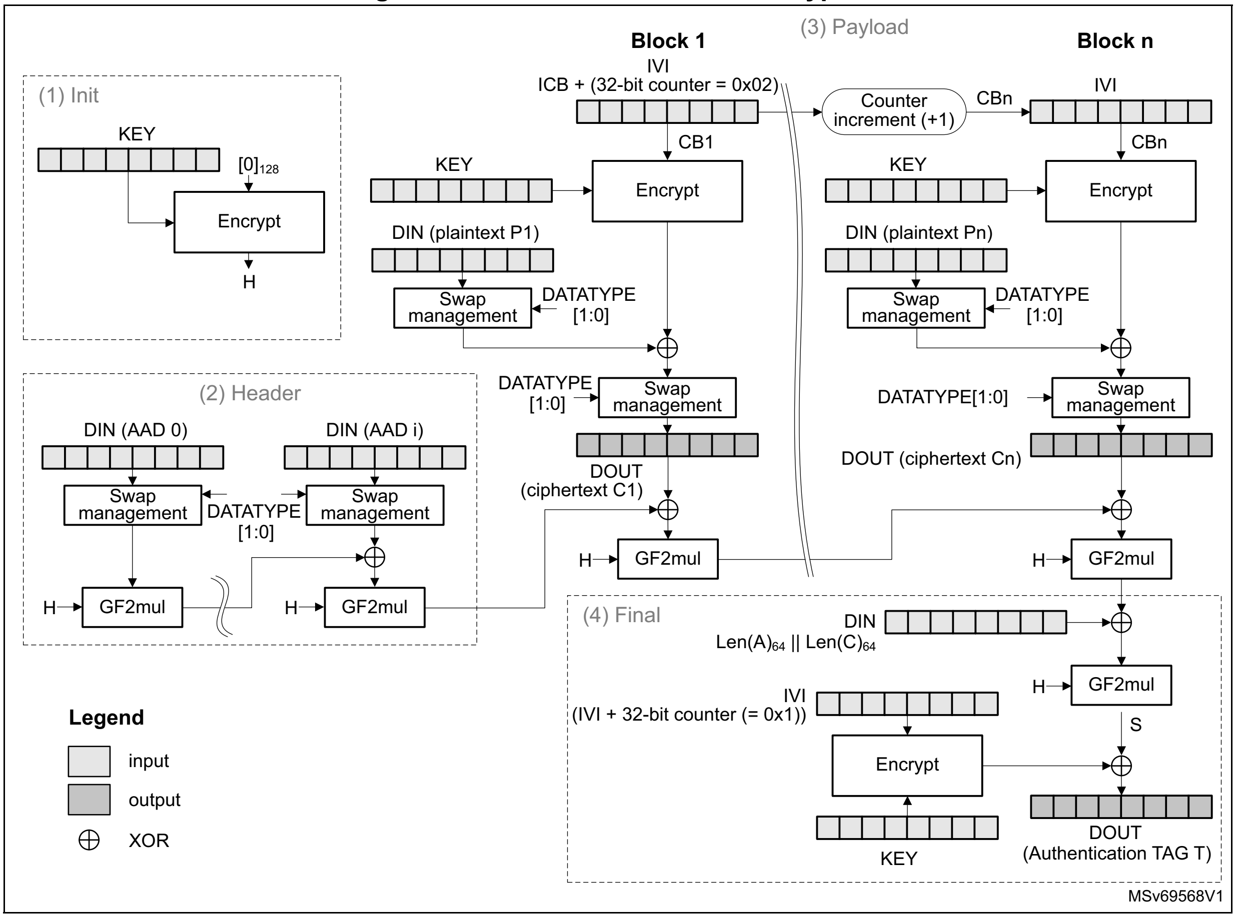
The GCM standard specifies that ciphertext C has the same bit length as the plaintext P.

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.

For more details, refer to NIST Special Publication 800-38D*, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 141 describes the GCM chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 141. GCM authenticated encryption



The first counter block (CB1) is derived from the initial counter block ICB by the application software, as defined in Table 199.

Table 199. Initialization of IV registers in GCM mode

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
ICB[127:96]	ICB[95:64]	ICB[63:32]	ICB[31:0] 32-bit counter = 0x0002

The last block of a GCM message contains the AAD header length and the payload length information, as shown in Table 200.

Table 200. GCM last block definition

Word order to SAES_DINR	First word	Second word	Third word	Fourth word
Input data	AAD length[63:32]	AAD length[31:0]	Payload length[63:32]	Payload length[31:0]

GCM encryption and decryption process

This process is described in *Section 26.4.6*, with the following sequence of events:

GCM initialize

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   - Select GCM chaining mode (write CHMOD[2:0] with 0x3) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (such as secure, nonsecure, specific CPU), also set KEYPROT.
   - Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 26.4.14* (wrapped keys) and *Section 26.4.15* (shared keys).
   - Select the GCM initialization phase, by writing GCMPH[1:0] with 0x0.
4. Write the initialization vector in SAES_IVRx registers according to *Table 199*.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 26.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Set EN to start the calculation of the hash key. EN is automatically cleared when the calculation is completed.
8. Wait until the CCF flag is set in the SAES_ISR register, indicating that the GCM hash subkey (H) computation is completed.
9. Clear the CCF flag by setting the CCF bit of the SAES_ICR register.

GCM header phase

10. Initialize header phase:
    a) Select the GCM header phase, by writing 0x1 to GCMPH[1:0]. Do not change the other configurations written during GCM initialization.
    b) Enable the SAES peripheral, by setting EN.
11. Append header data:
    a) If it is the last block and the AAD in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 26.4.5*.
    c) Repeat the step *b)* until the second-last AAD data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no AAD, that is, Len(A) = 0.

No data are read during header phase.

GCM payload phase

12. Initialize payload phase:
    a) Select the GCM payload phase, by writing GCMPH[1:0] with 0x2. Do not change the other configurations written during GCM initialization.
    b) If the header phase is skipped, enable the SAES peripheral by setting EN.
13. Append payload data:
    a) If it is the last block and the message in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 26.4.5*, then read the SAES_DOUTR register four times to save the resulting block
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(C)=0 (see GMAC mode).

GCM finalization

14. Encryption only: wait until the BUSY flag in the SAES_SR register is cleared.
15. Select the GCM final phase, by writing GCMPH[1:0] with 0x3. Do not change the other configurations written during GCM initialization.
16. Write the final GCM block into the SAES_DINR register. It is the concatenated AAD bit and payload bit lengths, as shown in *Table 200*.
17. Wait until the CCF flag in the SAES_ISR register is set.
18. Get the GCM authentication tag, by reading the SAES_DOUTR register four times.
19. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
20. Disable the SAES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message.

Note: In the final phase, data are written to SAES_DINR normally (no swapping), while swapping is applied to tag data read from SAES_DOUTR.

When transiting from the header or the payload phase to the final phase, the SAES peripheral must not be disabled, otherwise the result is wrong.

Suspend/resume operations in GCM mode

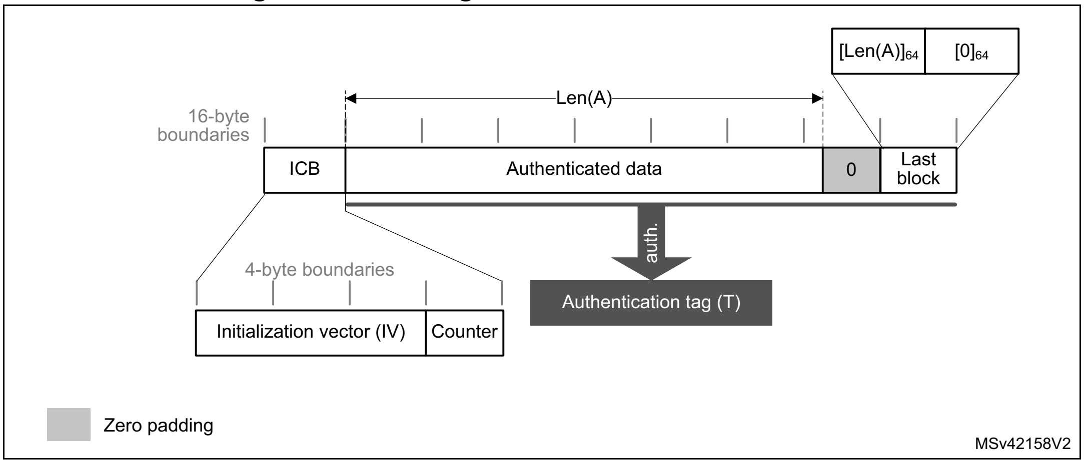
Suspend/resume operations are not supported in GCM mode.

26.4.12 SAES Galois message authentication code (GMAC)

The Galois message authentication code (GMAC) allows the authentication of a plaintext, generating the corresponding tag information (also known as message authentication code).

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). The following figure shows typical message construction for GMAC.

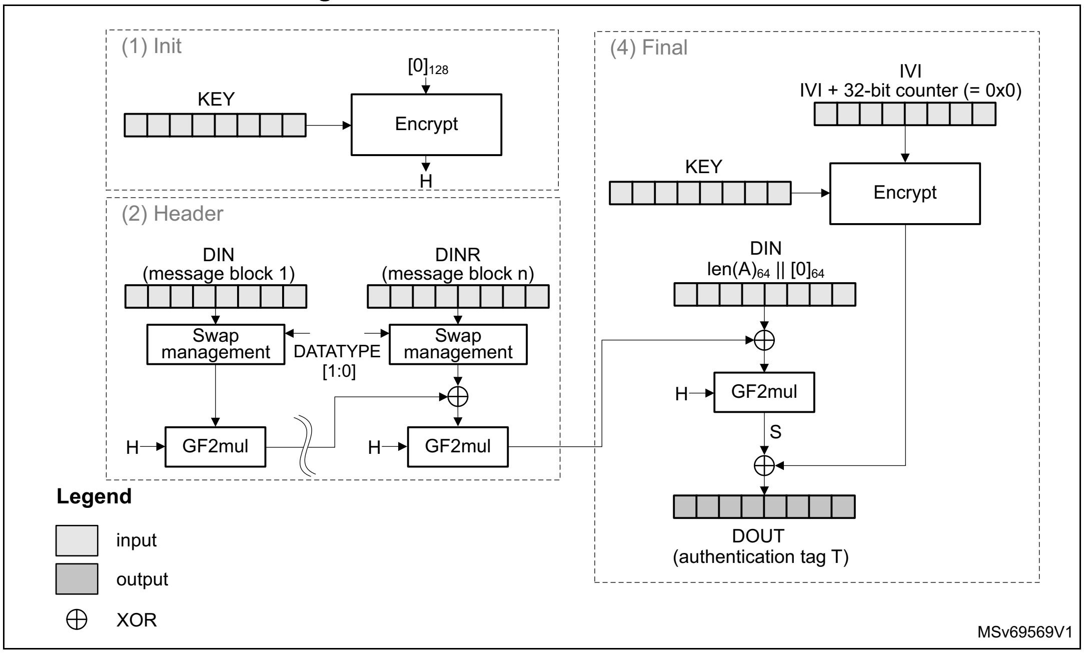
Figure 142. Message construction in GMAC mode



For more details, refer to NIST Special Publication 800-38D, *Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 143 describes the GMAC chaining implementation in the SAES peripheral. To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 143. GMAC authentication mode



The GMAC algorithm corresponds to the GCM algorithm applied on a message that only contains 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

Suspend/resume operations are not supported in GMAC mode.

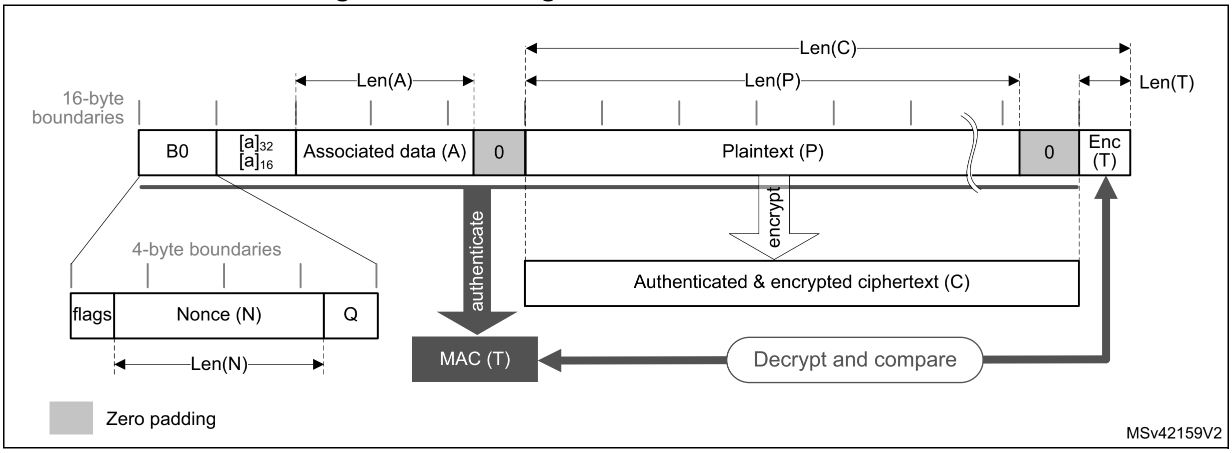
26.4.13 SAES counter with CBC-MAC (CCM)

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 counter mode processing. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

Note: NIST does not approve CBC-MAC as an authentication mode outside the context of the CCM specification.

The following figure shows typical message construction for CCM.

Figure 144. 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 144*). 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 144*).
- **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

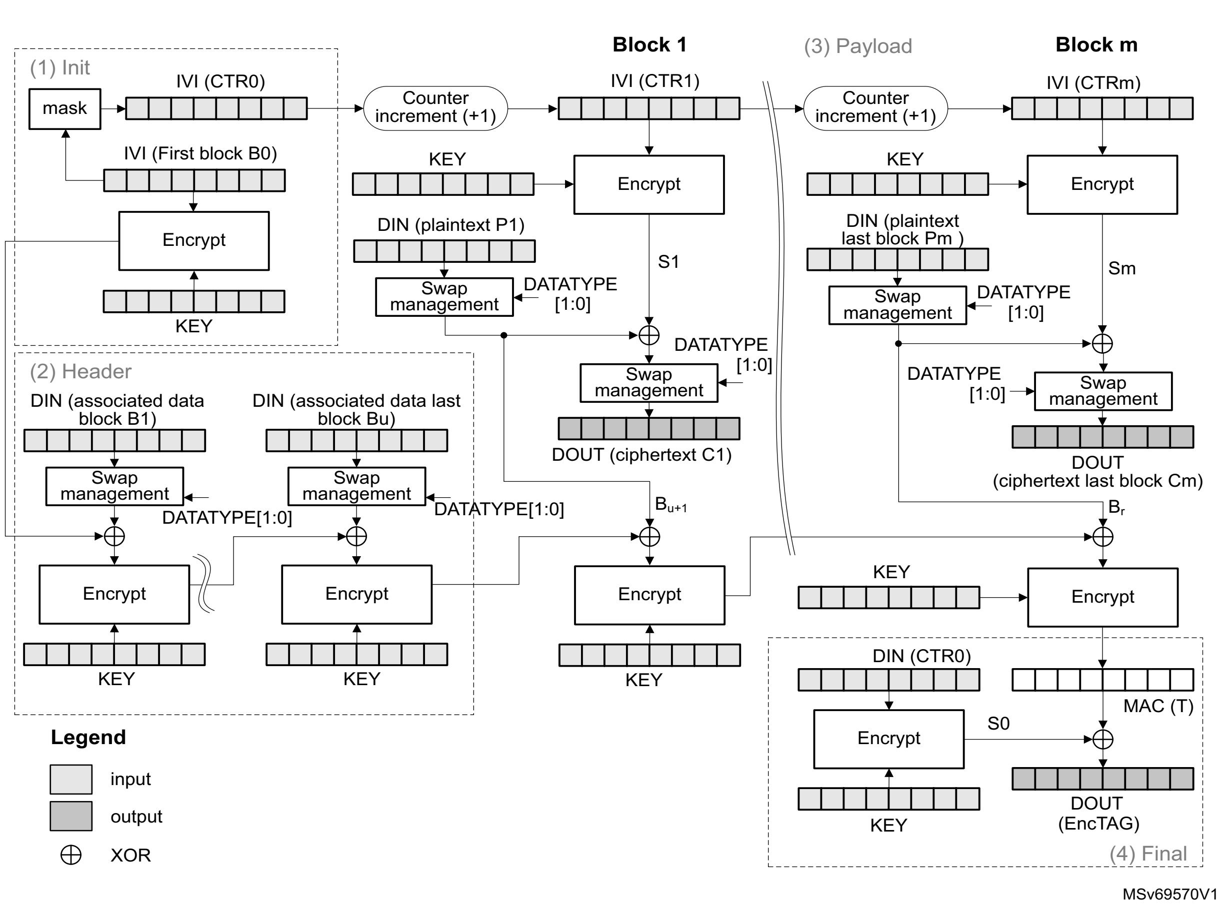
CTR0: 0710111213 141516 00000000 00000000

CTR1: 0710111213 141516 00000000 00000001

For more details, refer to NIST Special Publication 800-38C*, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*.

Figure 145 describes the CCM chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x4.

Figure 145. CCM mode authenticated encryption



The first block of a CCM message (B0) must be prepared by the application as defined in *Table 201*.

Table 201. Initialization of IV registers in CCM mode

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
B0[127:96](1)	B0[95:64]	B0[63:32]	B0[31:0](2)

1. The five most significant bits are cleared (flag bits).
2. Q length bits are cleared, except for the bit 0 that is set.

SAES supports counters up to 64 bits, as specified by NIST.

CCM encryption and decryption process

This process is described in *Section 26.4.6*, with the following sequence of events:

CCM initialize

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   – Select CCM chaining mode (write CHMOD[2:0] with 0x4) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   – Configure the data type, through DATATYPE[1:0]
   – Configure the key size, through KEYSIZE. If the key must not be shared with a different security context, also set the KEYPROT bit.
   – Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 26.4.14* (wrapped keys) and *Section 26.4.15* (shared keys).
   – Select the CCM initialization phase, by writing GCMPH[1:0] with 0x0.
4. Write the B0 data in SAES_IVRx registers according to *Table 201*.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 26.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Set EN to start the first mask calculation. The EN bit is automatically cleared when the calculation is completed.
8. Wait until the CCF flag in the SAES_ISR register is set.
9. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.

CCM header phase

10. Initialize header phase:
    a) Prepare the first block of the (B1) data associated with the message, in accordance with CCM chaining rules.
    b) Select the CCM header phase, by writing GCMPH[1:0] with 0x1. Do not change the other configurations written during the CCM initialization.
    c) Enable the SAES peripheral, by setting EN.
11. Append header data:
    a) If it is the last block and the associated data in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 26.4.5*.
    c) Repeat the step *b)* until the second-last associated data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no associated data, that is, Len(A) = 0

No data are read during the header phase.

CCM payload phase

12. Initialize payload phase:
    a) Select the CCM payload phase, by writing GCMPH[1:0] with 0x2. Do not change the other configurations written during the CCM initialization.
    b) If the header phase is skipped, enable the SAES peripheral, by setting EN.
13. Append payload data:
    a) In encryption only, if it is the last block and the plaintext in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 26.4.5*, then read the SAES_DOUTR register four times to save the resulting block.
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(P) = 0 or Len(C) = Len(T).

Remove $LSB_{Len(T)}(C)$ encrypted tag information when decrypting ciphertext C.

CCM finalization

14. Select the CCM final phase, by writing GCMPH[1:0] with 0x3. Do not change the other configurations written during the CCM initialization.
15. Wait until CCF flag in the SAES_ISR register is set.
16. Get the CCM authentication tag, by reading the SAES_DOUTR register four times.
17. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
18. Disable the SAES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message. Mask the authentication tag output with tag length to obtain a valid tag.

Note: In the final phase, swapping is applied to tag data read from SAES_DOUTR register.

When transiting from the header or the payload phase to the final phase, the SAES peripheral must not be disabled, otherwise the result is wrong.

Suspend and resume operations in CCM mode

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

To suspend the processing of a message in header or payload phase, proceed as follows:

1. Wait until the CCF flag of the SAES_ISR register is set (computation completed).
2. In the payload phase, read four times the SAES_DOUTR register to save the last-processed block.
3. Clear the CCF flag in the SAES_ISR register, by setting the CCF bit of the SAES_ICR register.
4. Save the SAES_SUSPRx registers in the memory.
5. Save the IV registers as they are altered during the data processing.
6. Disable the SAES peripheral, by clearing EN.
7. Save the current SAES_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.

To resume the processing of a message, proceed as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Write the suspend register values, previously saved in the memory, back into their corresponding SAES_SUSPRx registers.
3. Restore SAES_IVRx registers using the saved configuration.
4. Restore the initial setting values in the SAES_CR and SAES_KEYRx registers. For KEYSEL[2:0] selecting a key source different from the key registers, refer to *Section 26.4.17: SAES key registers* for details.
5. Enable the SAES peripheral, by setting EN.

26.4.14 SAES operation with wrapped keys

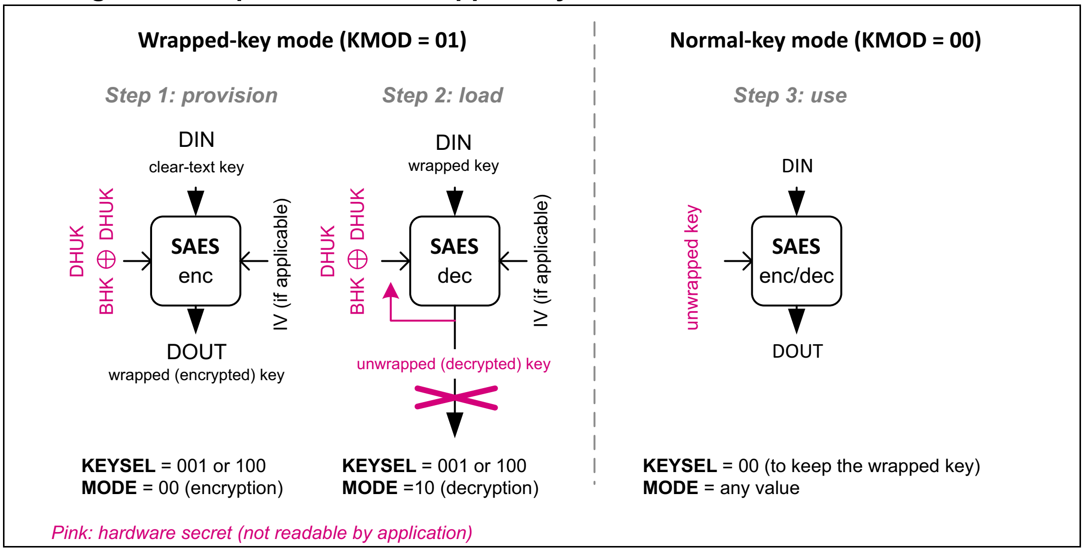
SAES peripheral can wrap (encrypt) and unwrap (decrypt) application keys using hardware-secret key DHUK, XOR-ed or not with application key BHK. With this feature, AES keys can be made usable by application software without being exposed in clear-text (unencrypted).

Wrapped key sequences are too small to be suspended/resumed. SAES cannot unwrap a key using an unwrapped key.

Operation with wrapped keys for SAES in ECB and CBC modes

Figure 146 summarizes how to wrap or unwrap keys for SAES in ECB and CBC modes. To protect the wrapped key, select DHUK by writing KEYSEL[2:0] with 0x1 or 0x4. Alternatively, select BHK by writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMP peripheral.

Figure 146. Operation with wrapped keys for SAES in ECB and CBC modes



Note: DHUK value depends on privilege, security, KMOD[1:0], KEYSEL[2:0], CHMOD[2:0], and KEYSIZE.

Key wrapping for SAES

The recommended sequence to wrap (that is, encrypt) a key is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *encryption* mode (MODE[1:0] at 0x0)
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size with KEYSIZE. This information is used both for the encryption key and for the key to be encrypt.
   - Select wrapped key mode by writing KMOD[1:0] with 0x1
4. Write the initialization vector in SAES_IVRx registers if CBC mode has been selected in previous step.
5. Select the DHUK key source by writing KEYSEL[2:0] with 0x1 or 0x4. Optionally, select the BHK, by writing KEYSEL[2:0] with 0x2. Refer to *Section 26.4.17* for details on the use of KEYSEL[2:0].
6. Wait until KEYVALID is set (DHUK loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Write the SAES_DINR register four times to input the key to encrypt (MSB first, see *Table 203*).
9. Wait until CCF flag is set in the SAES_ISR register.
10. 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.
11. Repeat steps *8.* to *10.* if KEYSIZE is set.
12. Disable the SAES peripheral, by clearing EN.

Note: Encryption in Wrapped-key mode is only supported when ECB or CBC is selected.

Key unwrapping for SAES

The recommended sequence to unwrap (or decrypt) a wrapped (encrypted) key using ECB/CBC is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select the chaining mode used during the wrapping process (CHMOD[2:0] at 0x0 or 0x1) in *key derivation* mode (MODE[1:0] at 0x1)
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size used during the wrapping process, with KEYSIZE. This information is used both for the decryption key and for the key to decrypt.
   - Select wrapped key mode, by writing KMOD[1:0] with 0x1.
4. With KEYSEL[2:0], select the same key source as when the key was wrapped/encrypted.
5. Wait until KEYVALID is set (the key loading completed).
6. Set EN bit in SAES_CR to enable the peripheral
7. Wait until CCF flag is set in the SAES_ISR register.
8. Clear the CCF flag, by setting the CCF bit in SAES_ICR register. The decryption key is available in the AES core, and SAES is disabled automatically.
9. Select the *decryption* mode (MODE[1:0] at 0x2). Other parameters are unchanged.
10. Write the initialization vector in SAES_IVRx registers if CBC mode has been selected in previous step.
11. Enable the SAES peripheral, by setting EN.
12. Write the SAES_DINR register four times to input the key to decrypt (MSB first, see *Table 203*).
13. Wait until CCF flag is set in the SAES_ISR register. Then clear the CCF flag by setting the CCF bit in SAES_ICR register. Reading SAES_DOUTR returns zero and triggers a read error (RDERRF).
14. Repeat steps *12.* and *13.* if KEYSIZE is set.
15. Disable the SAES peripheral, by clearing EN.

At the end of this sequence, the decrypted wrapped key is immediately usable by the application for any AES operation (normal key mode, that is, with KMOD[1:0] = 0x0).

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 3.

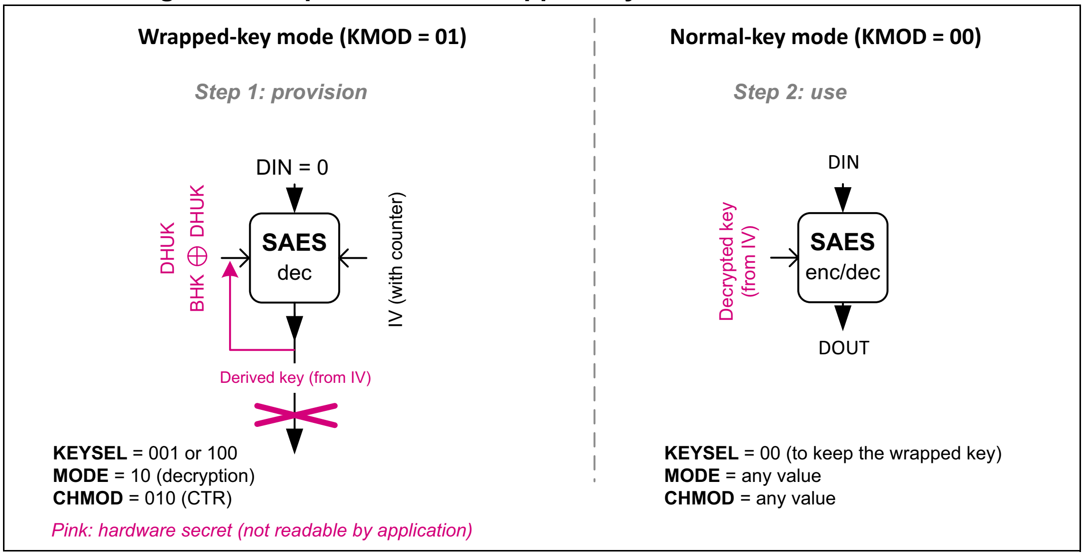
Note: When KMOD[1:0] = 0x1 (wrapped key) and MODE[1:0] = 0x2 (decryption) a read access to SAES_DOUTR register triggers a read error (RDERRF).

When KEYSEL[2:0] is at 0x1 (DHUK) or 0x4 (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.

Operation with wrapped keys for SAES in CTR mode

Figure 147 summarizes how to unwrap keys for SAES in CTR mode. To protect the derived key, select DHUK by writing KEYSEL[2:0] with 0x1 or 0x4. Alternatively, select BHK by writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMP peripheral.

Figure 147. Operation with wrapped keys for SAES in CTR mode



Note: DHUK value depends on privilege, security, KMOD[1:0], KEYSEL[2:0], CHMOD[2:0], and KEYSIZE.

The recommended sequence for SAES wrapped key mode using CTR is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select the CTR chaining mode (CHMOD[2:0] at 0x2) in decryption mode (MODE[1:0] at 0x2). Other MODE[1:0] values are not supported.
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size with KEYSIZE. It is used for encryption key and for the key to share.
   - Select wrapped key mode, by writing KMOD[1:0] with 0x1.
4. Write the initialization vector in SAES_IVRx registers, keeping the two least significant bits of SAES_IVR0 at zero.
5. Select the DHUK key source by writing KEYSEL[2:0] with 0x1 or 0x4. Optionally, select the BHK, by writing KEYSEL[2:0] with 0x2. Refer to *Section 26.4.17* for details on the use of KEYSEL[2:0].
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Wait until CCF flag is set in the SAES_ISR register.
9. Clear the CCF flag, by setting the CCF bit in SAES_ICR register. The derived hardware secret key is available in SAES_KEYRx registers.
10. Repeat steps *8.* and *9.* if KEYSIZE is set.
11. Disable the SAES peripheral, by clearing EN.

At the end of this sequence, the hardware secret key derived from the public data in the SAES_IVRx registers is then immediately usable by the application for any AES operation (normal key mode, that is, with KMOD[1:0] = 0x0).

Note: The configuration KMOD[1:0] at 0x1 (wrapped key), CHMOD[2:0] at 0x2 (CTR chaining), and MODE at 0x0 (encryption) disables the peripheral, by automatically clearing the EN bit of the SAES_CR register.

26.4.15 SAES operation with shared keys

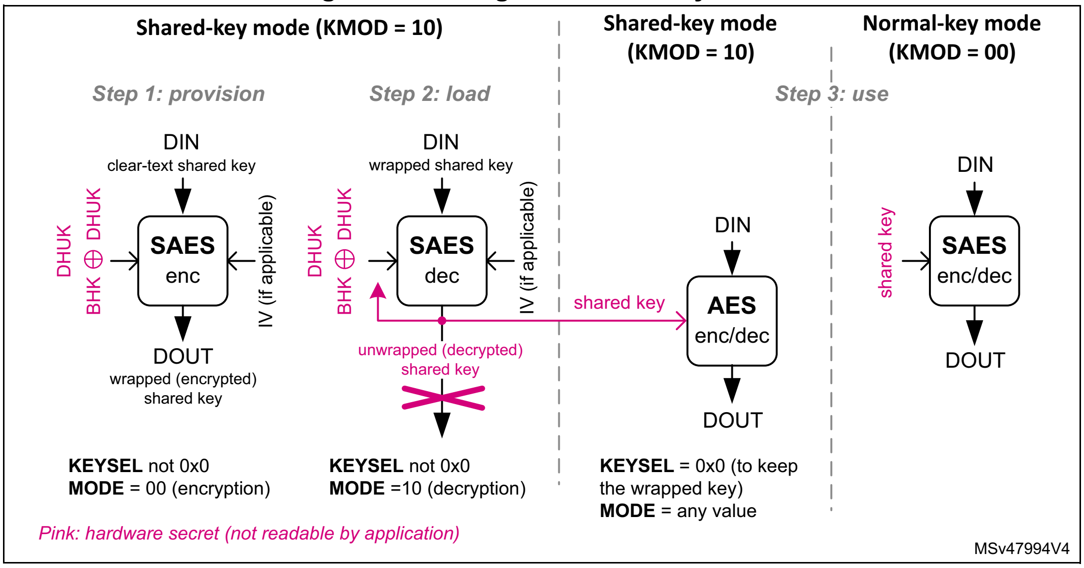
SAES peripheral can share application keys wrapped with hardware-secret key DHUK, XOR-ed or not with application key BHK. With this feature, the application software can make the AES keys available to the AES peripheral, without exposing them in clear-text (unencrypted).

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 26.4.17: SAES key registers.

Figure 148 summarizes how to wrap or unwrap keys to share with *AES* peripheral. To protect the shared key, DHUK must be selected, by writing KEYSEL[2:0] with 0x1 or 0x4. Alternatively, select BHK by writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMP peripheral.

Figure 148. Usage of Shared-key mode



Note: DHUK value depends on privilege, security, KMOD[1:0], KSHAREID, KEYSEL[2:0], CHMOD[2:0], and KEYSIZE.

In the step 3, AES represents the AES peripheral.

Key wrapping for AES peripheral

Before SAES can share a key with the AES peripheral, the key must be encrypted (wrapped) once. The encryption sequence of a shared key is the same as for a wrapped key, with KMOD[1:0] at 0x2 (shared key) and KSHAREID[1:0] kept at 0x0 in the step 3 in *Figure 148*. See *Key wrapping for SAES* for details.

Note: Encryption in Shared-key mode is only supported when ECB or CBC is selected.

Key unwrapping for AES peripheral (shared key)

Each time SAES needs to share a key with the AES peripheral, shared encrypted key must be decrypted (unwrapped) in SAES, then loaded by 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] at 0x2 (shared key) and KSHAREID[1:0] kept at 0x0 in the step 3 in *Figure 148*. See *Key unwrapping for SAES* for details.

In shared key mode when decryption mode is selected (MODE[1:0] at 0x2), a read access to the SAES_DOUTR register triggers a read error (RDERRF).

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 written with 0x0 (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] at 0x2 and KEYVALID set. The sequence in the AES key share target peripheral is described in *AES key sharing with secure AES co-processor* of the corresponding section in this document. It 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: When KMOD[1:0] is at 0x2 and BUSY set 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.

When KEYSEL[2:0] is at 0x1 (DHUK) or 0x4 (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.

26.4.16 SAES data registers and data swapping

Data input and output

A 16-byte data block enters 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 16-byte data block is retrieved from the SAES peripheral with four successive 32-bit word reads of 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 four 32-bit words of a 16-byte data block must be stored in the memory consecutively and in big-endian order, that is, with the most significant word on the lowest address. See *Table 202* “no swapping” option for details.

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 DATATYPE[1:0]. The selection applies to both SAES input and output.

Note: The data in SAES key registers (SAES_KEYRx) and initialization vector registers (SAES_IVRx) are not sensitive to the swap mode selection.

The SAES data swapping feature is summarized in *Table 202* and *Figure 149*.

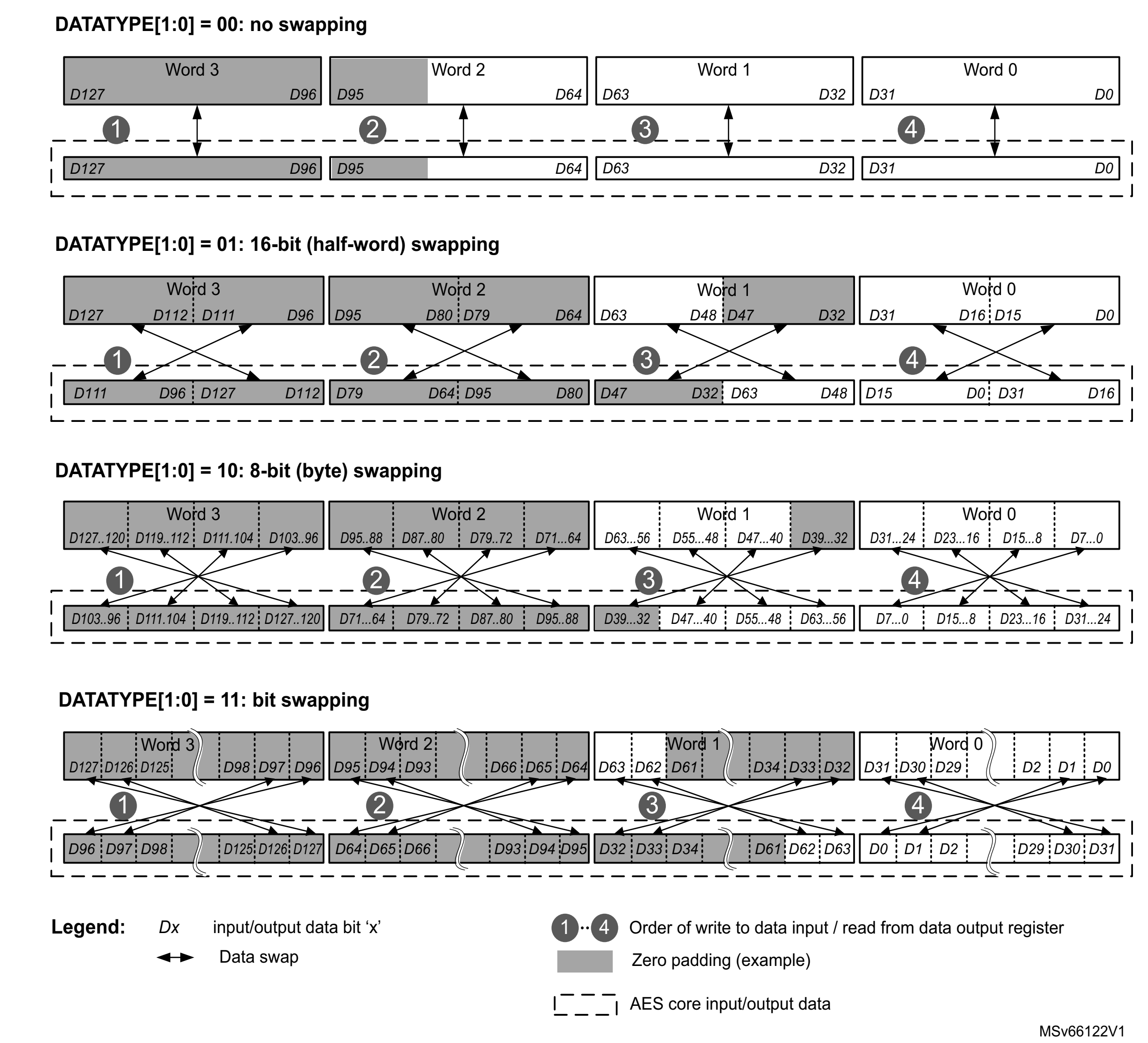
Table 202. AES data swapping example

DATATYPE[1:0]	Swapping performed	Data block System memory data (big-endian)
0x0	No swapping	Block[127..64]: 0x04EEF672 2E04CE96 Block[63..0]: 0x4E6F7720 69732074
		Address @, word[127..96]: 0x04EEF672 Address @ + 0x4, word[95..64]: 0x2E04CE96 Address @ + 0x8, word[63..32]: 0x4E6F7720 Address @ + 0xC, word[31..0]: 0x69732074
0x1	Half-word (16-bit) swapping	Block[63..0]: 0x**4E6F** 7720 **6973** 2074
		Address @, word[63..32]: 0x7720 **4E6F** Address @ + 0x4, word[31..0]: 0x2074 **6973**
0x2	Byte (8-bit) swapping	Block[63..0]: 0x**4E** 6F **77** 20 **69** 73 **20** 74
		Address @, word[63..32]: 0x20**77** 6F**4E** Address @ + 0x4, word[31..0]: 0x74**20** 73**69**

0x3	Bit swapping	Block[63..32]: 0x4E6F7720
		`0100 1110 0110 1111 0111 0111 0010 0000`
		Block[31..0]: 0x69732074
		`0110 1001 0111 0011 0010 0000 0111 0100`

		Address @, word[63..32]: 0x04EE F672
		`0000 0100 1110 1110 1111 0110 0111 0010`
		Address @ + 0x4, word[31..0]: 0x2E04 CE96
		`0010 1110 0000 0100 1100 1110 1001 0110`

Figure 149. 128-bit block construction according to the data type



Data padding

Figure 149 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:

- 84 message bits, with DATATYPE[1:0] = 0x0
- 48 message bits, with DATATYPE[1:0] = 0x1
- 56 message bits, with DATATYPE[1:0] = 0x2
- 34 message bits, with DATATYPE[1:0] = 0x3

26.4.17 SAES key registers

The eight SAES_KEYRx write-only registers store the encryption or decryption key information, as shown on *Table 203*. Reads are not allowed for security reason.

Note: In memory and in SAES *key registers, keys are stored in little-endian format, with most significant byte on the highest address.*

Table 203. Key endianness in SAES_KEYRx registers (128/256-bit keys)

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 registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield.

Write operations to the SAES_KEYRx registers are ignored when SAES peripheral is enabled (EN bit set) and KEYSEL[2:0] is different from zero. The application must check this before modifying key registers.

The entire key must be written before starting an AES computation.

In normal key mode (KMOD[1:0] at 0x0), with KEYSEL[2:0] at 0x0, the key registers must always be written in either ascending or descending order. The write sequence becomes:

- SAES_KEYRx (x = 0 to 3 or x=3 to 0) for KEYSIZE cleared
- SAES_KEYRx (x = 0 to 7 or x=7 to 0) for KEYSIZE set

Note: KEYSIZE must be written before the key.

As soon as the first key register is written, the KEYVALID flag is cleared. Once the key registers writing sequence is completed, KEYVALID is set and EN becomes writable. If an error occurs, KEYVALID is cleared and KEIF set (see *Section 26.4.19*).

Key selection

With KEYSEL[2:0] at 0x0, the application must write the key in the SAES_KEYRx registers.

With KEYSEL[2:0] at 0x1, a derived hardware unique key (DHUK), computed inside SAES from a non-volatile and secret root hardware unique key, is loaded directly into key registers, based on KEYSIZE information. Thanks to the key derivation function, the computed DHUK depends on the SAES usage (privilege, security, KMOD[1:0] / KSHAREID[1:0], CHMOD[2:0], CPU, and KEYSIZE context).

With KEYSEL[2:0] at 0x2, 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 cleared, x = 0 to 7 for KEYSIZE set) in ascending order. Refer to *Table 203*.

With KEYSEL[2:0] at 0x4, 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 cleared, x = 0 to 7 for KEYSIZE set) in ascending order. Refer to *Table 203*.

Repeated writing of KEYSEL[2:0] with the same non-zero value only triggers the loading of DHUK or BHK if KEYVALID is cleared. The recommended method to clear KEYVALID is to set IPRST. Such method is required for example when switching from ECB decryption to ECB encryption, selecting the same BHK (KEYSEL[2:0] at 0x2).

For all KEYSEL[2:0] values, 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 26.4.19*).

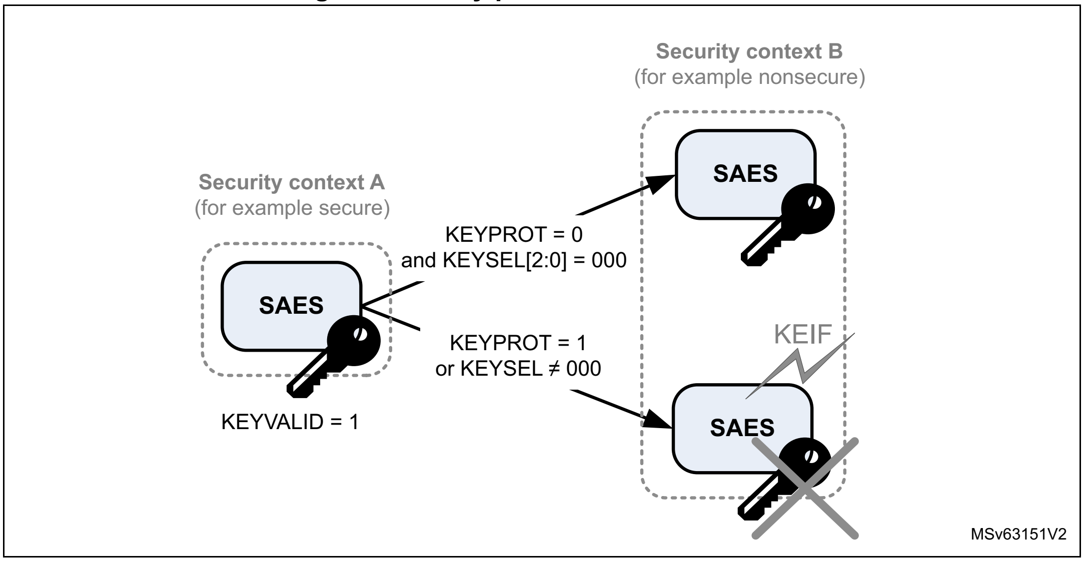
Note: DHUK, BHK and their XOR combination are not readable by any software (even secure).

Key protection

As depicted in *Figure 150*, when an application sets the KEYPROT bit before writing a key in SAES_KEYRx, any application executing in a different security context (that is, different security attribute) triggers a KEIF error flag upon access to any SAES register when KEYVALID is set.

Note: KEYSEL[2:0] values different from zero (normal key) automatically protect the key registers.

Figure 150. Key protection mechanisms



26.4.18 SAES initialization vector registers

The four SAES_IVRx registers store the initialization vector (IV) information, as shown in *Table 204*. They can only be written if the SAES peripheral is disabled (EN cleared).

Note: In memory and in SAES IV registers, initialization vectors are stored in little-endian format, with most significant byte on the highest address.

Table 204. IVI bitfield spread over SAES_IVRx registers

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0]

Initialization vector information depends on the chaining mode selected. When used, SAES_IVRx registers are updated upon each AES computation cycle (useful for managing suspend mode).

The initialization vector registers are not affected by the data swapping feature controlled through DATATYPE[1:0].

26.4.19 SAES error management

The SAES peripheral manages the errors described in this section.

Read error flag (RDERRF)

Unexpected read attempt of the SAES_DOUTR register returns zero, setting the RDERRF flag and the RWEIF flag. RDERRF is triggered during the computation phase or during the input phase.

Note: Unless otherwise indicated, SAES is not disabled when RDERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 26.5: SAES interrupts*.

The RDERRF and RWEIF flags are cleared by setting the RWEIF bit of the SAES_ICR register.

Write error flag (WRERRF)

Unexpected write attempt of the SAES_DINR register is ignored, setting the WRERRF and the RWEIF flags. WRERRF is triggered during the computation phase or during the output phase.

Note: Unless otherwise indicated, SAES is not disabled when WRERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 26.5: SAES interrupts*.

The WRERRF and RWEIF flags are cleared by setting the RWEIF bit of the SAES_ICR register.

Key error interrupt flag (KEIF)

There are multiple sources of errors that set the KEIF flag of the SAES_ISR register and clear the KEYVALID bit of the SAES_SR register:

- **Key protection error:** while KEYVALID is set, then if KEYPROT is set or KEYSEL[2:0] 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 (that is, different security attribute) accesses SAES.

- **Key writing sequence error:** triggered upon detecting an incorrect sequence of writing key registers. See *Section 26.4.17: SAES key registers* for details.
- **Key sharing size mismatch error:** triggered when KMOD[1:0] is at 0x2 and KEYSIZE in AES peripheral does not match KEYSIZE in SAES peripheral.
- **Key sharing error:** triggered upon failing transfer of SAES shared key to AES peripheral. See *Section 26.4.15: SAES operation with shared keys* for details.
- **Hardware secret key loading error:** triggered upon failing load of DHUK or BHK into SAES. KEYSEL[2:0] at 0x1 (DHUK), 0x2 (BHK) or 0x4 (DHUK XOR BHK) is not functional.

The KEIF flag is cleared with corresponding bit of the SAES_ICR register. An interrupt is generated if the KEIE bit of the SAES_IER register is set. For more details, refer to *Section 26.5: SAES interrupts*.

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.

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 disabling and 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 26.5: SAES interrupts*.

Upon an RNG error:

- Make sure that the RNG peripheral is properly configured.
- Verify that the RNG peripheral AHB clock is enabled and no noise source (or seed) error is pending in this peripheral.
- Clear RNGEIF or reset the peripheral by setting IPRST. The clearance of the BUSY flag 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.

Managing tamper 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, reset the SAES peripheral through RCC or global reset.

26.5 SAES interrupts

There are multiple individual maskable interrupt sources generated by the SAES peripheral to signal the following events:

- computation completed (CCF)
- read error (RDERRF)
- write error (WRERRF)
- key error (KEIF)
- RNG error (RNGEIF)

See *Section 26.4.19: SAES error management* for details on SAES errors.

These sources are combined into a common interrupt signal from the SAES peripheral that connects to the Cortex® CPU interrupt controller. Application can enable or disable SAES interrupt sources individually by setting/clearing the corresponding enable bit of the SAES_IER register.

The status of the individual maskable interrupt sources can be read from the SAES_ISR register. They are cleared by setting the corresponding bit of the SAES_ICR register.

Table 205 gives a summary of the available features.

Table 205. SAES interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable bit	Interrupt clear method
SAES	computation completed flag	CCF	CCFIE	set CCF(1)
	read error flag	RDERRF(2)	RWEIE	set RWEIF(1)
	write error flag	WRERRF(2)		
	key error flag	KEIF	KEIE	set KEIF(1)
	RNG error flag	RNGEIF	RNGEIE	set RNGEIF(1)

1. Bit of the SAES_ICR register.
2. Flag of the SAES_SR register, mirrored by the flag RWEIF of the SAES_ISR register.

26.6 SAES DMA requests

The SAES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the DMAINEN and DMAOUTEN bits of the SAES_CR register. When key derivation is selected (MODE[1:0] is at 0x1), setting those bits has no effect.

SAES only supports single DMA requests.

Suspend and resume operations are not supported in DMA mode.

Detailed usage of DMA with SAES can be found in *Appending data using DMA* subsection of *Section 26.4.5: SAES encryption or decryption typical usage*.

Data input using DMA

Setting DMAINEN enables DMA writing into SAES. SAES then initiates, during the input phase, a set of single DMA requests for each 16-byte data block to write to the SAES_DINR register (quadruple 32-bit word, MSB first).

Note: According to the algorithm and the mode selected, special padding / ciphertext stealing might be required (see Section 26.4.7).

Data output using DMA

Setting DMAOUTEN enables DMA reading from SAES. SAES then initiates, during the output phase, a set of single DMA requests for each 16-byte data block to read from the SAES_DOUTR register (quadruple 32-bit word, MSB first).

After the output phase, at the end of processing of a 16-byte data block, SAES switches automatically to a new input phase for the next data block, if any.

In DMA mode, 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 managing the data transfers by software.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Stopping DMA transfers

All DMA request signals are de-asserted when SAES is disabled (EN cleared) or the DMA enable bit (DMAINEN for input data, DMAOUTEN for output data) is cleared.

26.7 SAES processing latency

The following tables provide the 16-byte data block processing latency per operating mode.

Table 206. Processing latency for ECB, CBC and CTR

Key size	Mode of operation	Chaining algorithm	Clock cycles(1)
128-bit	Encryption or decryption(2)	ECB, CBC, CTR	**480**
	Key preparation	-	**145**
256-bit	Encryption or decryption(2)	ECB, CBC, CTR	**680**
	Key preparation	-	**230**

1. SAES kernel clock
2. Excluding key preparation time (ECB and CBC only).

Table 207. Processing latency for GCM and CCM (in SAES kernel clock cycles)

Key size	Mode of operation	Chaining algorithm	Initialization phase	Header phase(1)	Payload phase(1)	Final phase(1)
128-bit	Mode 1: Encryption/ Mode 3: Decryption	GCM	490	72(2)	480(3)	490
		CCM	490	490	800	490
256-bit	Mode 1: Encryption/ Mode 3: Decryption	GCM	650	72(2)	690(3)	650
		CCM	650	680	1350	650

1. Data insertion can include wait states forced by SAES on the AHB bus (maximum 3 cycles, typical 1 cycle).
2. SAES AHB clock cycles instead of kernel clock cycle (Galois multiplier only).
3. As a worst case in encryption mode, add extra 72 AHB clock cycles for the last block computation.

26.8 SAES registers

The registers are accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.

26.8.1 SAES control register (SAES_CR)

Address offset: 0x000

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]		NPBLB[3:0]				KEYPROT	KEYSIZE	Res.	CHMOD[2]
rw	rw	rw	rw	rw	rw	rw	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**: SAES peripheral software reset

Setting the bit resets the SAES peripheral, putting all registers to their default values, except the IPRST bit itself. Hence, any key-relative data are 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 kept 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.

0x0: Software key, loaded in key registers SAES_KEYx

0x1: Derived hardware unique key (DHUK)

0x2: Boot hardware key (BHK)

0x4: XOR of DHUK and BHK

Others: Reserved (if used, unfreeze SAES with IPRST)

When KEYSEL[2:0] 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 when KEYVALID is cleared.

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, nonsecure). 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] at 0x2 (shared key), which target can read the SAES key registers using a dedicated hardware bus.
0x0: AES peripheral
Others: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN 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. KEYSIZE must be correctly initialized when setting KMOD[1:0] different from zero.
0x0: Normal key mode. Key registers are freely usable and no specific use or protection applies to SAES_DINR and SAES_DOUTR registers.
0x1: Wrapped key for SAES mode. Key loaded in key registers can only be used to encrypt or decrypt AES keys. Hence, when a decryption is selected, read-as-zero SAES_DOUTR register is automatically loaded into SAES key registers after a successful decryption process.
0x2: Shared key mode. After a successful decryption process (unwrapping), SAES key registers are shared with the peripheral described in KSHAREID[1:0] bitfield. This sharing is valid only while KMOD[1:0] at 0x2 and KEYVALID=1. When a decryption is selected, read-as-zero SAES_DOUTR register is automatically loaded into SAES key registers after a successful decryption process.
Others: Reserved
With KMOD[1:0] other than zero, any attempt to configure the SAES peripheral for use by an application belonging to a different security context (such as secure or nonsecure) results in automatic key erasure and setting of the KEIF flag.
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN 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

This padding information must be filled by software before processing the last block of GCM payload encryption or CCM payload decryption, otherwise authentication tag computation is incorrect.
0x0: All bytes are valid (no padding)
0x1: Padding for the last LSB byte
...
0xF: Padding for the 15 LSB bytes of last block.

Bit 19 **KEYPROT**: Key protection

When set, hardware-based key protection is enabled.
0: When KEYVALID is set and KEYSEL[2:0] = 0 application can transfer the ownership of the SAES, with its loaded key, to an application running in another security context (such as nonsecure, 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, nonsecure) that does not match the one of the application that loaded the key.
Attempts to write the bit are ignored when BUSY is set, as well as when EN 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 key length in bits of the key used by SAES.
0: 128-bit
1: 256-bit
When KMOD[1:0] is at 0x1 or 0x2, KEYSIZE also defines the length of the key to encrypt or decrypt.
Attempts to write the bit are ignored when BUSY is set, as well as when the EN 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, applicable only with GCM, GMAC or CCM chaining modes.
0x0: Initialization phase
0x1: Header phase
0x2: Payload phase
0x3: Final phase

Bit 12 **DMAOUTEN**: DMA output enable

This bit enables automatic generation of DMA requests during the data phase, for outgoing data transfers from SAES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bit 11 **DMAINEN**: DMA input enable

This bit enables automatic generation of DMA requests during the data phase, for incoming data transfers to SAES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bits 10:7 Reserved, must be kept at reset value.

Bits 16, 6:5 **CHMOD[2:0]**: Chaining mode

This bitfield selects the AES chaining mode:
0x0: Electronic codebook (ECB)
0x1: Cipher-block chaining (CBC)
0x2: Counter mode (CTR)
0x3: Galois counter mode (GCM) and Galois message authentication code (GMAC)
0x4: Counter with CBC-MAC (CCM)
others: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 4:3 **MODE[1:0]**: Operating mode

This bitfield selects the SAES operating mode:
0x0: Encryption
0x1: Key derivation (or key preparation), for ECB/CBC decryption only
0x2: Decryption
0x3: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 2:1 **DATATYPE[1:0]**: Data type

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. This swapping is defined in *Section 26.4.16: SAES data registers and data swapping*.
0x0: No swapping (32-bit data).
0x1: Half-word swapping (16-bit data)
0x2: Byte swapping (8-bit data)
0x3: Bit-level swapping
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bit 0 **EN**: 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. When KMOD[1:0] is different from 0x0, using IPRST bit is recommended instead.

This bit is automatically cleared by hardware upon the completion of the key preparation (MODE[1:0] at 0x1) and upon the completion of GCM/GMAC/CCM initialization phase.

The bit cannot be set as long as KEYVALID is cleared, or when SAES is in one of the following configurations:

– KMOD[1:0] at 0x1 (wrap), CHMOD[2:0] at 0x3 (GCM)

– KMOD[1:0] at 0x1 (wrap), CHMOD[2:0] at 0x2 (CTR), MODE[1:0] at 0x0 (encryption).

26.8.2 SAES status register (SAES_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.	KEYVALID	Res.	Res.	Res.	BUSY	WRERRF	RDERRF	Res.
								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 key of size defined by KEYSIZE is loaded in SAES_KEYRx key registers.

0: Key not valid

1: Key valid

The EN bit can only be set when KEYVALID is set.

In normal mode when KEYSEL[2:0] is at zero, the key must be written in the key registers in the correct sequence, otherwise the KEIF flag is set and KEYVALID remains cleared.

When KEYSEL[2:0] is different from zero, the BUSY flag is automatically set by SAES. When the key is loaded successfully, BUSY is cleared and KEYVALID set. Upon an error, KEIF is set, BUSY cleared and KEYVALID remains cleared.

If set, KEIF must be cleared through the SAES_ICR register, otherwise KEYVALID cannot be set. See the KEIF flag description for more details.

For further information on key loading, refer to *Section 26.4.17: 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.

0: Idle

1: Busy

SAES is flagged as idle when disabled (when EN is low) or when the last processing is completed.

SAES is flagged as busy when processing a block data, preparing a key (ECB or CBC decryption only), fetching random number from the RNG, or transferring a shared key to the target peripheral.

When GCM encryption payload phase is selected, this flag must be at zero before suspending current process to manage a higher-priority message. BUSY must also be cleared before selecting the GCM final phase.

Bit 2 **WRERRF**: Write error flag

This bit is set when an unexpected write to the SAES_DINR register occurred. When set WRERRF bit has no impact on the SAES operations.

0: No error

1: Unexpected write to SAES_DINR register occurred during computation or data output phase.

The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.

The flag is cleared by setting the RWEIF bit of the SAES_ICR register.

Bit 1 **RDERRF**: Read error flag

This bit is set when an unexpected read to the SAES_DOUTR register occurred. When set RDERRF bit has no impact on the SAES operations.

0: No error

1: Unexpected read to SAES_DOUTR register occurred during computation or data input phase.

The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.

The flag is cleared by setting the RWEIF bit of the SAES_ICR register.

Bit 0 Reserved, must be kept at reset value.

26.8.3 SAES data input register (SAES_DINR)

Address offset: 0x008

Reset value: 0x0000 0000

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]**: Data input

A four-fold sequential write to this bitfield during the Input phase results in writing a complete 16-bytes 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 16-bytes input buffer.

Reads return zero.

26.8.4 SAES data output register (SAES_DOUTR)

Address offset: 0x00C

Reset value: 0x0000 0000

Read when KMOD[1:0] is at 0x1 or 0x2 while MODE[1:0] is at 0x2 and EN is set 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]**: Data output

This read-only bitfield fetches a 32-bit output buffer. A four-fold sequential read of this bitfield, upon the computation completion (CCF flag set), virtually reads a complete 16-byte 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].

26.8.5 SAES key register 0 (SAES_KEYR0)

Address offset: 0x010

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]

These are bits [31:0] of the write-only bitfield KEY[255:0] AES encryption or decryption key, depending on the MODE[1:0] bitfield of the SAES_CR register.

Writes to SAES_KEYRx registers are ignored when SAES is enabled (EN bit set). When KEYSEL[2:0] is different from 0 and KEYVALID is 0, writes to key registers are also ignored and they result in setting the KEIF bit of the SAES_ISR register.

With KMOD[1:0] at 0x0, a special writing sequence is required. In this sequence, any valid write to AES_KEYRx register clears the KEYVALID flag except for the sequence-completing write that sets it. Also refer to the description of the KEYVALID flag in the AES_SR register.

26.8.6 SAES key register 1 (SAES_KEYR1)

Address offset: 0x014

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 and for information relative to writing SAES_KEYRx registers.

26.8.7 SAES key register 2 (SAES_KEYR2)

Address offset: 0x018

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 and for information relative to writing SAES_KEYRx registers.

26.8.8 SAES key register 3 (SAES_KEYR3)

Address offset: 0x01C

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 and for information relative to writing SAES_KEYRx registers.

26.8.9 SAES initialization vector register 0 (SAES_IVR0)

Address offset: 0x020

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]

SAES_IVRx registers store the 128-bit initialization vector or the nonce, depending on the chaining mode selected. This value is updated by hardware after each computation round (when applicable). Write to this register is ignored when EN bit is set in SAES_CR register

26.8.10 SAES initialization vector register 1 (SAES_IVR1)

Address offset: 0x024

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.

26.8.11 SAES initialization vector register 2 (SAES_IVR2)

Address offset: 0x028

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.

26.8.12 SAES initialization vector register 3 (SAES_IVR3)

Address offset: 0x02C

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.

26.8.13 SAES key register 4 (SAES_KEYR4)

Address offset: 0x030

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 and for information relative to writing SAES_KEYRx registers.

26.8.14 SAES key register 5 (SAES_KEYR5)

Address offset: 0x034

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 and for information relative to writing SAES_KEYRx registers.

26.8.15 SAES key register 6 (SAES_KEYR6)

Address offset: 0x038

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 and for information relative to writing SAES_KEYRx registers.

26.8.16 SAES key register 7 (SAES_KEYR7)

Address offset: 0x03C

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 and for information relative to writing SAES_KEYRx registers.

26.8.17 SAES suspend registers (SAES_SUSPRx)

Address offset: 0x040 + 0x4 * 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
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]**: Suspend data

SAES_SUSPRx registers contain the complete internal register states of the SAES when the CCM processing of the current task is suspended to process a higher-priority task. Refer to *Section 26.4.8: SAES suspend and resume operations* for more details.

Clearing EN bit of the SAES_CR register clears this register to zero.

SAES_SUSPRx registers are not used in other chaining modes than CCM.

26.8.18 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)

26.8.19 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 (for example bad entropy).

0: RNG bus is functional

1: Error detected on RNG bus interface (random seed fetching error)

The flag setting generates an interrupt if the RNGEIE bit of the SAES_IER register is set.

The flag is cleared by setting the corresponding bit of the SAES_ICR register. The clear action triggers the reload of a new random number from the RNG peripheral.

Bit 2 **KEIF:** Key error interrupt flag

This read-only bit is set by hardware when the key information fails to load into key registers or when the key register use is forbidden.

0: No key error detected

1: Key information failed to load into key registers or the key register use is forbidden

The flag setting generates an interrupt if the KEIE bit of the SAES_IER register is set. It also clears the key registers and the KEYVALID flag in the SAES_SR register.

The flag is cleared by setting the corresponding bit of the SAES_ICR register.

KEIF is raised upon any of the following events:

–SAES fails to load the DHUK (KEYSEL[2:0] = 0x1 or 0x4).

–SAES fails to load the BHK (KEYSEL[2:0] = 0x2 or 0x4).

–AES fails to load the key shared by SAES peripheral (KMOD[1:0] = 0x2).

–KEYVALID is set and either KEYPROT is set or KEYSEL[2:0] is other than 0x0. The security context of the application that loads the key (secure or nonsecure) 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 cleared, SAES_KEYR0 then SAES_KEYR1 then SAES_KEYR2 then SAES_KEYR3 register, or reverse. For KEYSIZE set, 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 RDERRF or a WRERRF error flag is set in the SAES_SR register.
0: No read or write error detected
1: Read or write error detected
The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.
The flag is cleared by setting the corresponding bit of the SAES_ICR register.
The flags has no meaning when key derivation mode is selected.
See the SAES_SR register for details.

Bit 0 **CCF:** Computation complete flag
This flag indicates whether the computation is completed. It is significant only when the DMAOUTEN bit is cleared, and it may stay high when DMAOUTEN is set.
0: Not completed
1: Completed
The flag setting generates an interrupt if the CCFIE bit of the SAES_IER register is set.
The flag is cleared by setting the corresponding bit of the SAES_ICR register.

26.8.20 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 clears both RDERRF and WRERRF flags in the SAES_SR register.

Bit 0 **CCF:** Computation complete flag clear
Setting this bit clears the CCF status bit of the SAES_ISR register.

26.8.21 SAES register map

Table 208. SAES 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	SAES_CR	IPRST	KEYSEL[2:0]			KSHAREID[1:0]		KMOD[1:0]		NPBLB[3:0]				KEYPROT	KEYSIZE	Res.	CHMOD[2]	Res.	GCMPH[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	CHMOD[1]	CHMOD[0]	MODE[1:0]		DATATYPE[1:0]		EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0		0	0	0	0					0	0	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	WRERRF	RDERRF	Res.
	Reset value																									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

Table 208. SAES register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x040	SAES_SUSPR0	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	SAES_SUSPR1	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	SAES_SUSPR2	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	SAES_SUSPR3	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	SAES_SUSPR4	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	SAES_SUSPR5	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	SAES_SUSPR6	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	SAES_SUSPR7	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	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
0x309-0x3FF	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: Memory organization* for the register boundary addresses.

27 Hash processor (HASH)

27.1 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.

27.2 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

27.3 HASH implementation

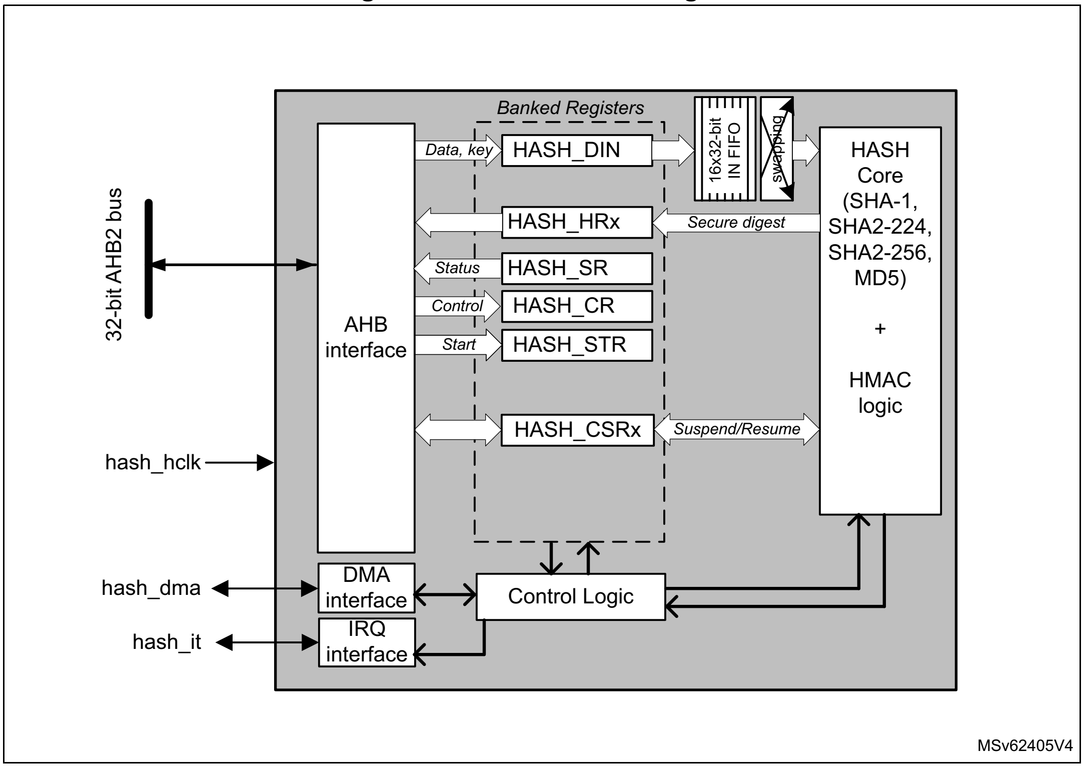
The devices have a single instance of HASH peripheral.

27.4 HASH functional description

27.4.1 HASH block diagram

Figure 151 shows the block diagram of the hash processor.

Figure 151. HASH block diagram



27.4.2 HASH internal signals

Table 209 describes a list of useful to know internal signals available at HASH level, not at product level (on pads).

Table 209. 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

27.4.3 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.

27.4.4 Message data feeding

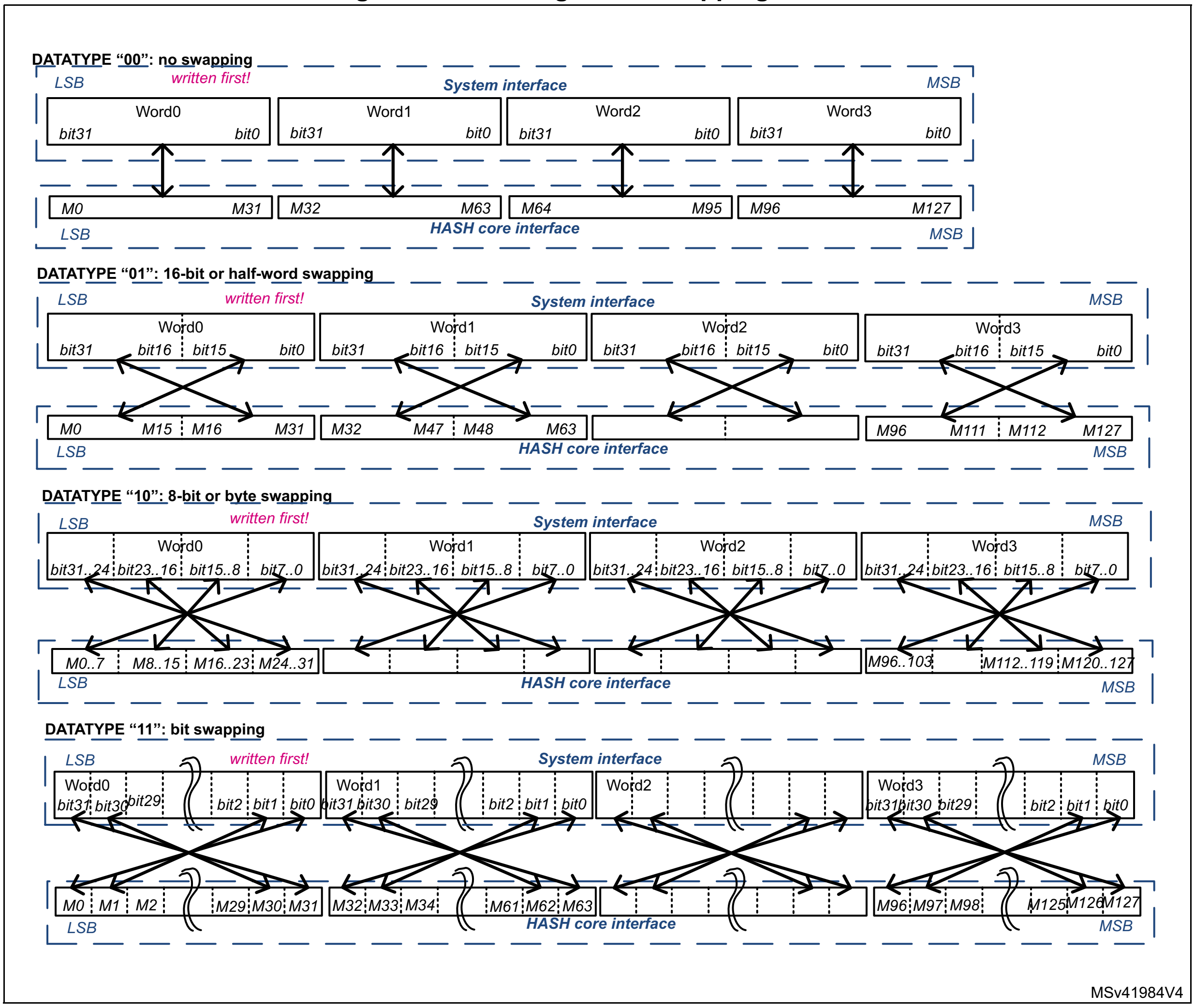
The message (or data file) to be processed by the HASH is considered as a bit string. Per FIPS PUB 180-4 standard this message bit string grows from left to right, with hexadecimal words expressed in “big-endian” convention, so that within each word, the most significant bit is stored in the left-most 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 152* 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 152. Message data swapping feature



27.4.5 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 27.4.4: 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 27.4.7: 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 27.4.6* 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 27.4.6 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 210.

Table 210. 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 27.4.7: HMAC operation.

27.4.6 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 27.4.5: 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
```

27.4.7 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 27.4.4: 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 27.4.4: 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 27.4.11: 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:

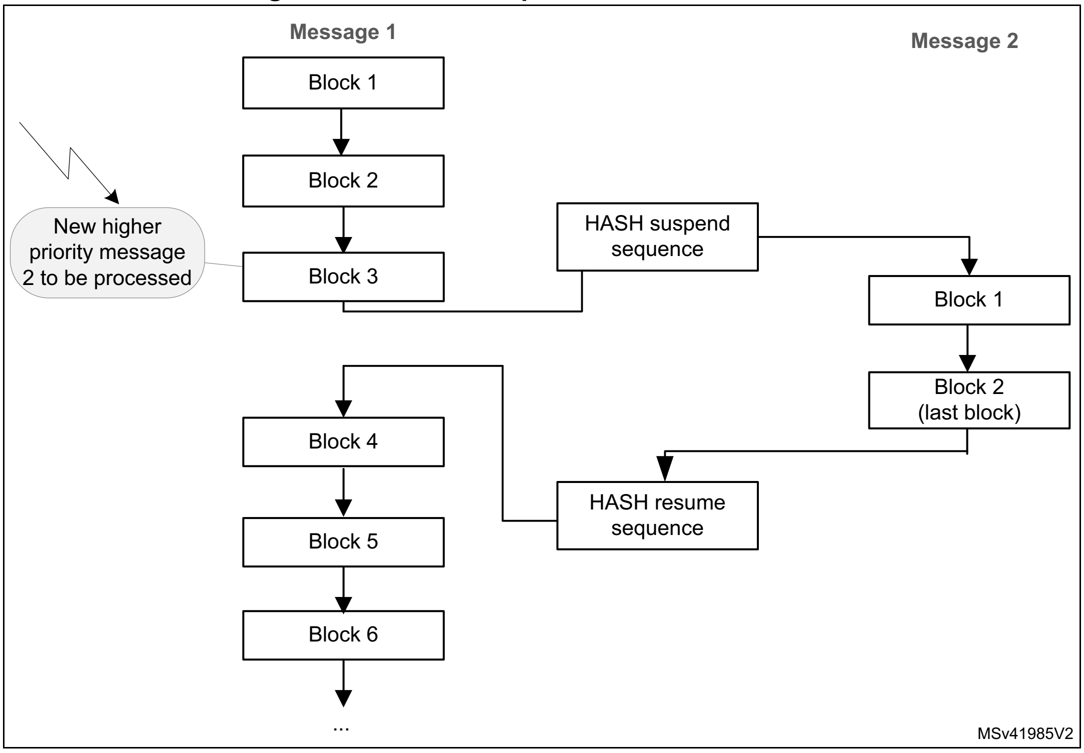
```
HASH_HR0 = 0x5FD596EE
HASH_HR1 = 0x78D5553C
HASH_HR2 = 0x8FF4E72D
HASH_HR3 = 0x266DFD19
HASH_HR4 = 0x2366DA29
```

27.4.8 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 153*.

Figure 153. 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 153*).

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.

27.4.9 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 27.4.5: 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.

27.4.10 HASH error management

No error flags are generated by the hash processor.

27.4.11 HASH processing time

Table 211 summarizes the time required to process an intermediate block for each mode of operation.

Table 211. 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

27.5 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 212* gives a summary of the available features.

Table 212. 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

27.6 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.

27.6.1 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.

27.6.2 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.

27.6.3 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].

27.6.4 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 + 0x4 * 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 27.6.4: HASH digest registers* introduction.

HASH digest register x (HASH_HRx)

Address offset: 0x310 + 0x4 * 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 27.6.4: HASH digest registers* introduction.

HASH supplementary digest register x (HASH_HRx)

Address offset: 0x310 + 0x4 * 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 27.6.4: HASH digest registers* introduction.

27.6.5 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

27.6.6 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.

27.6.7 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 27.6.7: HASH context swap registers* introduction.

27.6.8 HASH register map

Table 213. 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.	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- 0xF4	Reserved	Reserved																															
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
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- 0x30C	Reserved	Reserved																															
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

Table 213. 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
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: Memory organization* for the register boundary addresses.

28 Public key accelerator (PKA)

28.1 PKA 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.

For a given operation, 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.

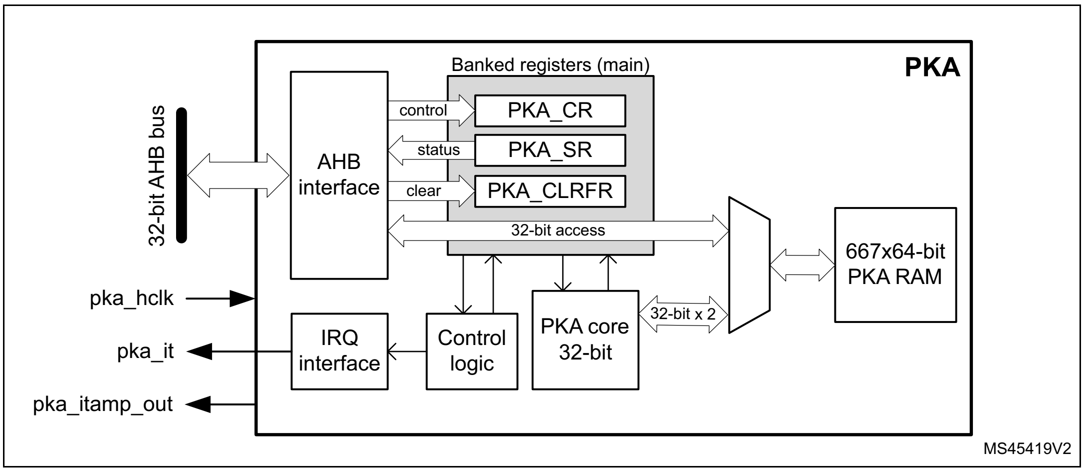
28.2 PKA main features

- 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)

28.3 PKA functional description

28.3.1 PKA block diagram

Figure 154. PKA block diagram



28.3.2 PKA internal signals

Table 214 lists the internal signals available at the PKA level, not necessarily available on product bonding pads.

Table 214. 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.

28.3.3 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. Once EN bit is set, refer to *Section 28.3.6* for details about PKA initialization sequence.

According to the security policy applied to the device, PKA RAM can also be reset following a tamper event. Refer to *Tamper detection and response* in the System security section (if applicable to this product).

28.3.4 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 28.5.1*.

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 215* and *Table 216*, respectively, for integer arithmetic functions and prime field (Fp) elliptic curve functions.

Table 215. PKA integer arithmetic functions list

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x01	000001	Montgomery parameter computation R2 mod n	Section 28.4.2
0x0E	001110	Modular addition (A+B) mod n	Section 28.4.3
0x0F	001111	Modular subtraction (A-B) mod n	Section 28.4.4
0x10	010000	Montgomery multiplication (AxB) mod n	Section 28.4.5
0x00	000000	Modular exponentiation A^e mod n	Section 28.4.6
0x02	000010	Modular exponentiation A^e mod n (fast mode)	
0x03	000011	Modular exponentiation A^e mod n (protected)	Section 28.4.6
0x08	001000	Modular inversion A^{-1} mod n	Section 28.4.7
0x0D	001101	Modular reduction A mod n	Section 28.4.8
0x09	001001	Arithmetic addition A+B	Section 28.4.9
0x0A	001010	Arithmetic subtraction A-B	Section 28.4.10
0x0B	001011	Arithmetic multiplication AxB	Section 28.4.11
0x0C	001100	Arithmetic comparison (A=B, A>B, A<B)	Section 28.4.12
0x07	000111	RSA CRT exponentiation	Section 28.4.13

Table 216. 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 28.4.14*
0x20	100000	ECC scalar multiplication kP (protected)	*Section 28.4.15*
0x23	100011	ECC complete addition	*Section 28.4.18*
0x24	100100	ECDSA sign (protected)	*Section 28.4.16*
0x26	100110	ECDSA verification	*Section 28.4.17*
0x27	100111	ECC double base ladder	*Section 28.4.19*
0x2F	101111	ECC projective to affine	*Section 28.4.20*

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 215* and *Table 216* 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) and any intermediate values are automatically erased when PKA RAM is cleared at the end of the protected operations (BUSY goes low). They are also protected against side channel attacks.

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 28.5.2: Computation times*.

The only operation using fast mode is modular exponentiation (MODE = 0x02).

28.3.5 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 mod 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.

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 28.5.1: Supported elliptic curves*.
- G is the chosen elliptic curve base point (also known as generator), with a large prime order n (that is, *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} mod *n*
5. calculate u_1 = zw mod *n* and u_2 = *rw* mod *n*
6. calculate the curve point (x_1, y_1) = u_1 x G + u_2 x Q_A
7. the signature is valid if *r* = x_1 (mod *n*), it is invalid otherwise.

Steps 4 to 7 are accelerated by PKA using *ECDSA verification*.

28.3.6 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.

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 28.3.7).

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 these 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 that 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 28.4*.

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 28.3.7* for details.

Using precomputed Montgomery parameters (PKA Fast mode)

As explained in *Section 28.3.4*, 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 28.4.5*). 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 28.5.1*.
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 28.4*.

28.3.7 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 0, while writes are ignored.
- The selected operating mode using MODE bitfield is not listed in PKA operating modes (see bitfield description), or PKA is running in limited mode (see LMF bit in PKA_SR). 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 28.6* for details).

ADDRERRF, OPERRF and RAMERRF errors are cleared by setting the corresponding bit in PKA_CLRFR.

OPERRF error must be cleared using OPERRFC bit in PKA_CLRFR before a new operation is written in PKA_CR register.

Note: The PKA can be re-initialized at any moment by resetting the EN bit in the PKA_CR register.

28.4 PKA operating modes

28.4.1 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: The validity of all input parameters to the PKA must be checked before starting any operation, as PKA assumes that all of them are valid and consistent with each other. Input parameters must not exceed the operand size specified 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 (that is, 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 */
```

28.4.2 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 217*.

Table 217. 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	

28.4.3 Modular addition

Modular addition operation consists in the computation of A + B mod n. Operation instructions are summarized in *Table 218*.

Table 218. 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	($0 \leq A < n$)	RAM@0xA50	ROS
	Operand B	($0 \leq B < n$)	RAM@0xC68	
	Modulus value n	($n < 2^{4160}$)	RAM@0x1088	
OUT	Result: A+B mod n	($0 \leq$ result $< n$)	RAM@0xE78	

28.4.4 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 219*.

Table 219. 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	($0 \leq A < n$)	RAM@0xA50	ROS
	Operand B	($0 \leq B < n$)	RAM@0xC68	
	Modulus value n	($n < 2^{4160}$)	RAM@0x1088	
OUT	Result: A-B mod n	($0 \leq$ result $< n$)	RAM@0xE78	

28.4.5 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) Assuming that 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) Assuming that 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 220*.

Table 220. 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	$(0 \leq A < n)$	RAM@0xA50	ROS
	Operand B	$(0 \leq B < n)$	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*).

28.4.6 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 221* and *Table 222*. RSA operation involving secret information (RSA decryption) must use the protected mode detailed on *Table 223*, 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, the error code is not affected when PKA automatically clears the PKA RAM at the end of this protected operation. When the error output equals 0xD60D, the result output is not affected either.

Operation instructions for modular exponentiation are summarized in *Table 221* (normal mode), *Table 222* (fast mode) and in *Table 223* (protected mode). Fast mode usage is explained in *Section 28.3.6*.

Table 221. 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)	($0 \leq A < n$)	RAM@0xC68	ROS
IN	Exponent e	($0 \leq e < n$)	RAM@0xE78	
	Modulus value n	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	
OUT	Result: A^e mod n	($0 \leq$ result $< n$)	RAM@0x838	

Table 222. 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)	($0 \leq A < n$)	RAM@0xC68	ROS
IN	Exponent e	($0 \leq e < n$)	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	($0 \leq$ result $< n$)	RAM@0x838	

Table 223. 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)	$(0 \le A < n)$	RAM@0x16C8	ROS
	Exponent e	$(0 \le e < n)$	RAM@0x14B8	
	Modulus value n	(odd integer only, $n < 2^{4096}$)	RAM@0x0838	
	Phi value(1)	-	RAM@0x0C68	
OUT	Result: A^e mod n	$(0 \le \text{result} < n)$	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).

28.4.7 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 ($1 \le A < n$) 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 224*.

Table 224. 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	$(0 \le A < n)$	RAM@0xA50	ROS
	Modulus value n	(odd integer only, $n < 2^{4160}$)	RAM@0xC68	
OUT	Result: A^{-1} mod n	$0 < \text{result} < n$	RAM@0xE78	

28.4.8 Modular reduction

Modular reduction operation consists in the computation of the remainder of A divided by n. Operation instructions are summarized in *Table 225*.

Table 225. 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	$(0 \le A < 2n < 2^{4160})$	RAM@0xA50	ROS
	Modulus value n	(odd integer only, $n < 2^{4160}$)	RAM@0xC68	
OUT	Result A mod n	(0 < result < n)	RAM@0xE78	

28.4.9 Arithmetic addition

Arithmetic addition operation consists in the computation of A + B. Operation instructions are summarized in *Table 226*.

Table 226. 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	$(0 \le A < 2^M)$	RAM@0xA50	ROS
	Operand B	$(0 \le B < 2^M)$	RAM@0xC68	
OUT	Result: A+B	$(0 \le \text{result} < 2^{M+1})$	RAM@0xE78	ROS + 1

28.4.10 Arithmetic subtraction

Arithmetic subtraction operation consists in the following computations:

- If $A \ge B$ 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 227*.

Table 227. 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	$(0 \le < A < 2^M)$	RAM@0xA50	ROS
	Operand B	$(0 \le < B < 2^M)$	RAM@0xC68	
OUT	Result: A-B	$(0 \le \text{result} < 2^M)$	RAM@0xE78	

28.4.11 Arithmetic multiplication

Arithmetic multiplication operation consists in the computation of AxB. Operation instructions are summarized in *Table 228*.

Table 228. 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	$(0 \leq \text{result} < 2^{2M})$	RAM@0xE78	2xROS

28.4.12 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 229*.

Table 229. 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

28.4.13 RSA CRT exponentiation

For efficiency many popular crypto libraries such as OpenSSL RSA use the following optimization for decryption and signing based on the chinese remainder theorem (CRT):

- *p* and *q* are precomputed primes, stored as part of the private key
- d_P= d mod (*p* -1)
- d_Q= d mod (*q* -1) and
- q_{inv}= q^{-1} mod *p*

These values allow the recipient to compute the exponentiation $m = A^d$ (mod pq) more efficiently as follows:

- $m_1 = A^{dP} \bmod p$
- $m_2 = A^{dQ} \bmod q$
- $h = q_{inv} (m_1 - m_2) \bmod p$, with $m_1 > m_2$
- $m = m_2 + hq \bmod pq$

Operation instructions for computing CRT exponentiation A^d mod pq are summarized in *Table 230*.

Table 230. 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	$(0 < d_P < 2^{M/2})$	RAM@0x730	ROS / 2
	Operand d_Q	$(0 < d_Q < 2^{M/2})$	RAM@0xE78	
	Operand q_{inv}	$(0 < q_{inv} < 2^{M/2})$	RAM@0x948	
	Prime *p*(1)	$(0 < p < 2^{M/2})$	RAM@0xB60	
	Prime *q*(1)	$(0 < q < 2^{M/2})$	RAM@0x1088	
IN	Operand A	$(0 < A < 2^M)$	RAM@0x12A0	ROS
OUT	Result: A^d mod *pq*	$(0 \le result < pq)$	RAM@0x838	

1. Must be different from 2.

28.4.14 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 231*.

Table 231. 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

28.4.15 ECC Fp scalar multiplication

This operation consists in the computation of 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 232*.

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 232. 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 232. 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	($0 \le k < 2^{640}$)	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 must be carried out.
- For k < 0 (that is, a negative scalar multiplication is required) the 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: The error code is not affected when PKA automatically clears the PKA RAM at the end of this protected operation. When the error output equals 0xD60D, the result output is not affected either.

28.4.16 ECDSA sign

ECDSA signing operation (outlined in *Section 28.3.5*) is summarized in *Table 233* (input parameters) and in *Table 234* (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 233. 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 234. 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: The error code is not affected when PKA automatically clears the PKA RAM at the end of this protected operation. When the error output equals 0xD60D, the result output is not affected either.

Extended ECDSA support

PKA also supports extended ECDSA signature, for which the inputs and the outputs are the same as ECDSA signature (*Table 233* and *Table 234*, respectively), with the addition of the coordinates of the point kG. This extra output is defined in *Table 235*.

Table 235. 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	

28.4.17 ECDSA verification

ECDSA verification operation (outlined in *Section 28.3.5*) is summarized in *Table 236* (input parameters) and *Table 237* (output parameters).

The application has to check if the output error is equal to 0xD60D, if different the signature is not verified.

Table 236. 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 237. ECDSA verification - Outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Result: ECDSA verify	– 0xD60D: valid signature – 0xA3B7: invalid signature	RAM@0x5D0	64 bits
	Computed signature part *r*	– (0 < *r* < n)	RAM@0x578	EOS

28.4.18 ECC complete addition

ECC complete addition computes the addition of two given points on an elliptic curve.

Operation instructions are summarized in *Table 238*.

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 238. 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	

28.4.19 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 239*.

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)

This operation requires the input point Z coordinate to be equal to 1 (input point represented in affine coordinates).

Table 239. 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

28.4.20 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/Z, 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 240*.

Table 240. 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

28.5 Example of configurations and processing times

28.5.1 Supported elliptic curves

The PKA supports all non-singular elliptic curves defined over prime fields. Those curves can 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 0000000**3**

Table 241 summarizes the family of curves supported by PKA for ECC operations.

Table 241. Family of supported curves for ECC operations

<table>
<tr><th>Curve name</th><th>Standard</th><th colspan="2">Reference</th></tr>
<tr><td>P-192</td><td rowspan="5">NIST</td><td colspan="2" rowspan="5">Digital Signature Standard (DSS), NIST FIPS 186-4</td></tr>
<tr><td>P-224</td></tr>
<tr><td>P-256</td></tr>
<tr><td>P-384</td></tr>
<tr><td>P-521</td></tr>
<tr><td>brainpoolP224r1, brainpoolP224t1</td><td rowspan="5">IETF</td><td rowspan="5">– Brainpool Elliptic Curves, IETF RFC 5639
– Brainpool Elliptic Curves for the Internet Key Exchange (IKE) Group Description Registry, IETF RFC 6932</td><td rowspan="5">https://tools.ietf.org</td></tr>
<tr><td>brainpoolP256r1, brainpoolP256t1</td></tr>
<tr><td>brainpoolP320r1, brainpoolP320t1</td></tr>
<tr><td>brainpoolP384r1, brainpoolP384t1</td></tr>
<tr><td>brainpoolP512r1, brainpoolP512t1</td></tr>
<tr><td>secp192k1, secp192r1</td><td rowspan="5">SEC</td><td rowspan="5">Standards for Efficient Cryptography SEC 2 curves</td><td rowspan="5">https://www.secg.org</td></tr>
<tr><td>secp224k1, secp224r1</td></tr>
<tr><td>secp256k1, secp256r1</td></tr>
<tr><td>secp384r1</td></tr>
<tr><td>secp521r1</td></tr>
<tr><td>Recommended curve parameters for public key cryptographic algorithm SM2</td><td>OSCCA</td><td colspan="2">– 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</td></tr>
</table>

28.5.2 Computation times

The following tables summarize the PKA computation times, expressed in AHB clock cycles.

Table 242. 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 243. 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 1s included in the scalar parameter, and include the computation of Montgomery parameter R2.

Table 244. 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 245. 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 246. 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 247. 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 248. 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 249. 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 250. 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.

28.6 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 28.3.7*
2. Access to unmapped address (ADDRERRF), see *Section 28.3.7*
3. PKA RAM access while PKA operation is in progress (RAMERRF), see *Section 28.3.7*
4. PKA end of operation (PROCENDF)

The 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 251 gives a summary of the available features.

Table 251. 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

28.7 PKA registers

28.7.1 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.	OPERRIE	ADDRERRIE	RAMERRIE	Res.	PROCENDIE	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 (protected)
100100: ECDSA sign (protected)
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. When PKA is configured in limited mode (LMF = 1 in PKA_SR), writing a MODE different from 0x26 with EN bit to 1 triggers OPERRF bit to be set and write to MODE bit is ignored.

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

Bit 1 **START**: start the operation

Set this bit to start the operation selected by the MODE[5:0] bitfield, 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.

28.7.2 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.	OPERRF	ADDRERRF	RAMERRF	Res.	PROCENDF	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.	LMF	INITOK
														r	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**: Busy flag

This bit is set whenever a PKA operation is in progress (START = 1 in PKA_CR). It is automatically cleared when the computation is complete, making PKA RAM accessible again.

0: No operation is in progress (default)

1: An operation is in progress

If PKA is started with a wrong opcode, it stays busy for a couple of cycles, then it aborts automatically the operation and goes back to ready (BUSY = 0).

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **LMF**: Limited mode flag

This bit is updated when EN bit in PKA_CR is set

0: All values documented in MODE bitfield can be used.

1: Only ECDSA verification (MODE = 0x26) is supported by the PKA.

Bit 0 **INITOK**: PKA initialization OK

This bit is asserted when PKA initialization is complete.

0: PKA is not initialized correctly. START bit cannot be set.

1: PKA is initialized correctly and can be used normally.

Note: When RNG is not able to output proper random numbers or when SAES is using the RNG bus, INITOK remains at 0. If it happens, make sure the RNG peripheral is properly clocked and initialized, and the SAES peripheral is clocked with BUSY bit cleared in SAES_SR to release the RNG bus, then enable PKA again.

28.7.3 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.	OPERRFC	ADDRERRFC	RAMERRFC	Res.	PROCENDFC	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.

28.7.4 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 28.3.3: PKA reset and clocks.

28.7.5 PKA register map

Table 252. 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.	LMF	INITOK
	Reset value											0	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* for the register boundary addresses.

29 Advanced-control timers (TIM1)

In this section, “TIMx” should be understood as “TIM1” since there is only one instance of this type of timer for the products to which this reference manual applies.

29.1 TIM1 introduction

The advanced-control timer (TIM1) consists of a 16-bit autoreload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The advanced-control (TIM1) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 29.3.30: Timer synchronization*.

29.2 TIM1 main features

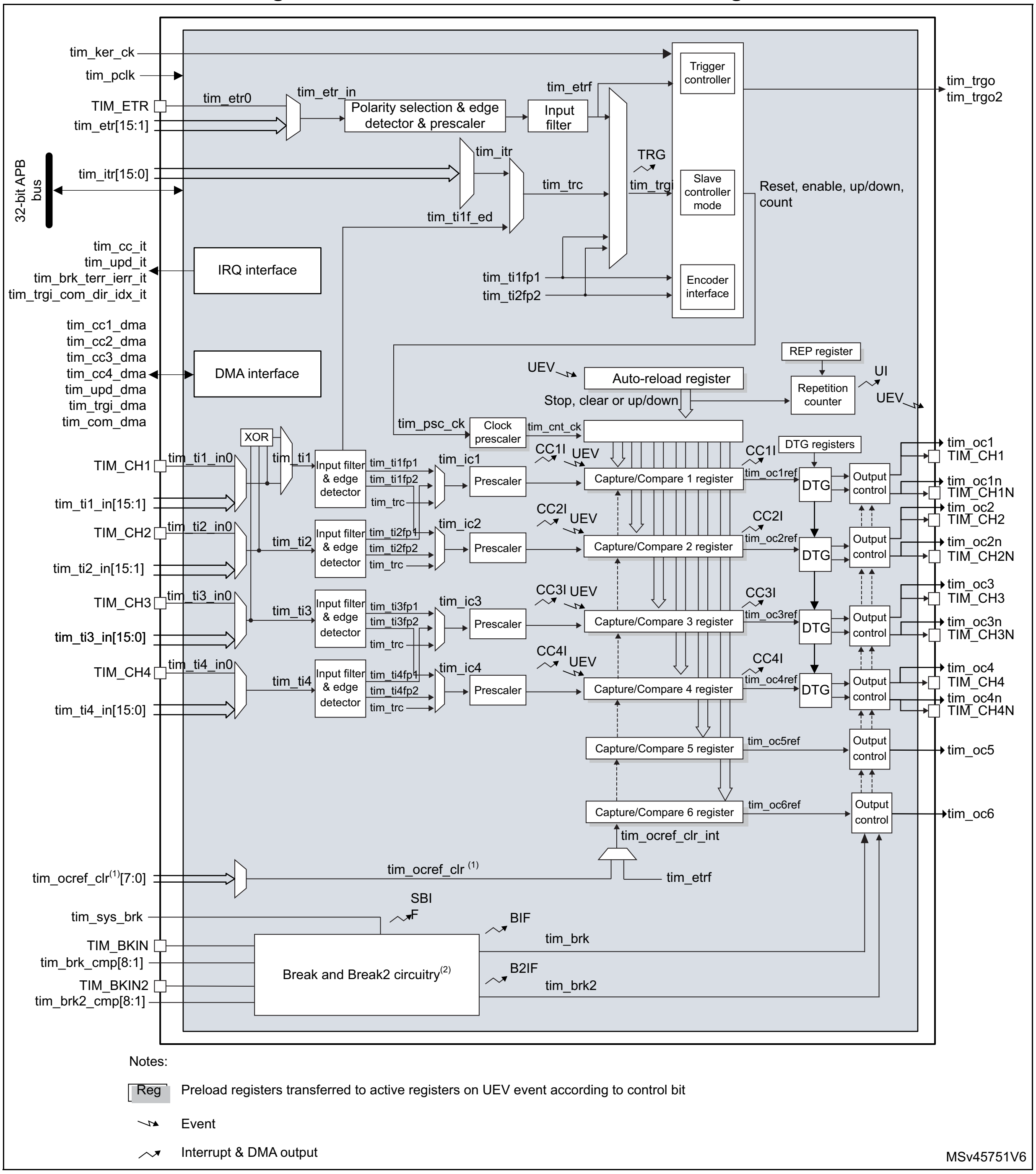
TIM1 timer features include:

- 16-bit up, down, up/down autoreload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency by any factor from 1 to 65536.
- Up to six 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

29.3 TIM1 functional description

29.3.1 Block diagram

Figure 155. Advanced-control timer block diagram



1. This feature is not available on all timers, refer to *Section 29.3.2: TIM1 pins and internal signals*.
2. See *Figure 202: Break and Break2 circuitry overview* for details.

29.3.2 TIM1 pins and internal signals

The tables in this section summarize the TIM inputs and outputs

Table 253. 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 254. 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 254. 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_trgi_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_trgi_dma	Output	Timer trigger dma request
tim_com_dma	Output	Timer commutation dma request

Table 255, Table 256, Table 257 and Table 258 list the sources connected to the tim_ti[4:1] input multiplexers.

Table 255. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources
	TIM1
tim_ti1_in0	TIM1_CH1
tim_ti1_in1	COMP1_OUT(1)
tim_ti1_in2	COMP2_OUT(1)
tim_ti1_in[15:3]	Reserved

1. Only available on STM32WBA54xx and STM32WBA55xx devices.

Table 256. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources
	TIM1
tim_ti2_in0	TIM1_CH2
tim_ti2_in[15:1]	Reserved

Table 257. Interconnect to the tim_ti3 input multiplexer

tim_ti3 inputs	Sources
	TIM1
tim_ti3_in0	TIM1_CH3
tim_ti3_in[15:1]	Reserved

Table 258. Interconnect to the tim_ti4 input multiplexer

tim_ti4 inputs	Sources
	TIM1
tim_ti4_in0	TIM1_CH4
tim_ti4_in[15:1]	Reserved

Table 259 lists the internal sources connected to the tim_itr input multiplexer.

Table 259. Internal trigger connection

Timer internal trigger input signal	TIM1
tim_itr0	Reserved
tim_itr1	tim2_trgo
tim_itr2	tim3_trgo

Table 259. Internal trigger connection (continued)

Timer internal trigger input signal	TIM1
tim_itr[6:3]	Reserved
tim_itr7	tim16_oc1
tim_itr8	tim17_oc1
tim_itr[15:9]	Reserved

Table 260 lists the internal sources connected to the tim_etr input multiplexer.

Table 260. Interconnect to the tim_etr input multiplexer

Timer external trigger input signal	Timer external trigger signals assignment
	TIM1
tim_etr0	TIM1_ETR
tim_etr1	COMP1_OUT(1)
tim_etr2	COMP2_OUT(1)
tim_etr3	Reserved
tim_etr4	HSI16
tim_etr[10:5]	Reserved
tim_etr11	adc4_awd1
tim_etr12	adc4_awd2
tim_etr13	adc4_awd3
tim_etr[15:14]	Reserved

1. Only available on STM32WBA54xx and STM32WBA55xx devices.

Table 261, *Table 262* and *Table 263* list the sources connected to the tim_brk and tim_brk2inputs.

Table 261. Timer break interconnect

tim_brk inputs	TIM1
TIM_BKIN	TIM1_BKIN pin
tim_brk_cmp1	COMP1_OUT(1)
tim_brk_cmp2	COMP2_OUT(1)
tim_brk_cmp[8:3]	Reserved

1. Only available on STM32WBA54xx and STM32WBA55xx devices.

Table 262. Timer break2 interconnect

tim_brk2 inputs	TIM1
TIM_BKIN2	TIM1_BKIN2 pin
tim_brk2_cmp1	COMP1_OUT(1)
tim_brk2_cmp2	COMP2_OUT(1)
tim_brk2_cmp[8:3]	Reserved

1. Only available on STM32WBA54xx and STM32WBA55xx devices.

Table 263. System break interconnect

tim_sys_brk inputs	TIM1
tim_sys_brk0	Cortex-M33 LOCKUP
tim_sys_brk1	Programmable Voltage Detector (PVD)
tim_sys_brk2	SRAM parity error
tim_sys_brk3	Flash memory ECC error
tim_sys_brk4	HSE32 lock Security System (HSECSS)

Table 264 lists the internal sources connected to the tim_ocref_clr input multiplexer.

Table 264. Interconnect to the ocref_clr input multiplexer

Timer OCREF clear signal	Timer OCREF clear signals assignment
	TIM1
tim_ocref_clr0	COMP1_OUT(1)
tim_ocref_clr1	COMP2_OUT(1)
tim_ocref_clr[7:2]	Reserved

1. Only available on STM32WBA54xx and STM32WBA55xx devices.

29.3.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related autoreload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the autoreload 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)
- Autoreload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The autoreload register is preloaded. Writing to or reading from the autoreload 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 autoreload 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 from 1 to 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 156 and *Figure 157* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 156. Counter timing diagram with prescaler division change from 1 to 2

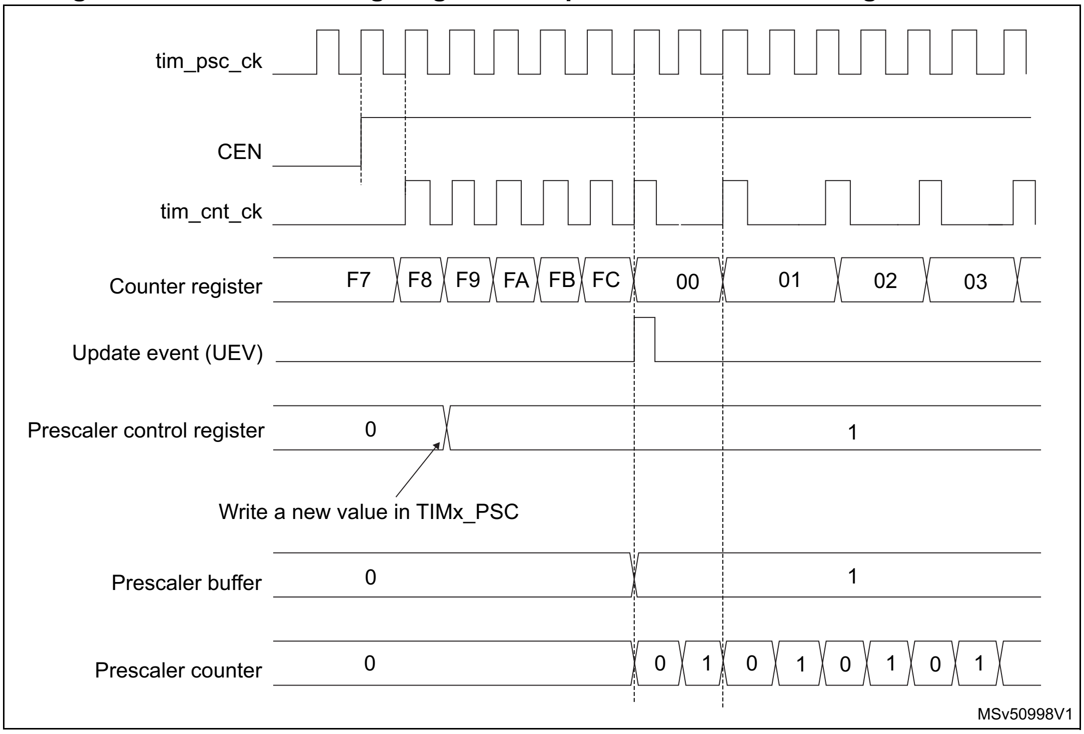
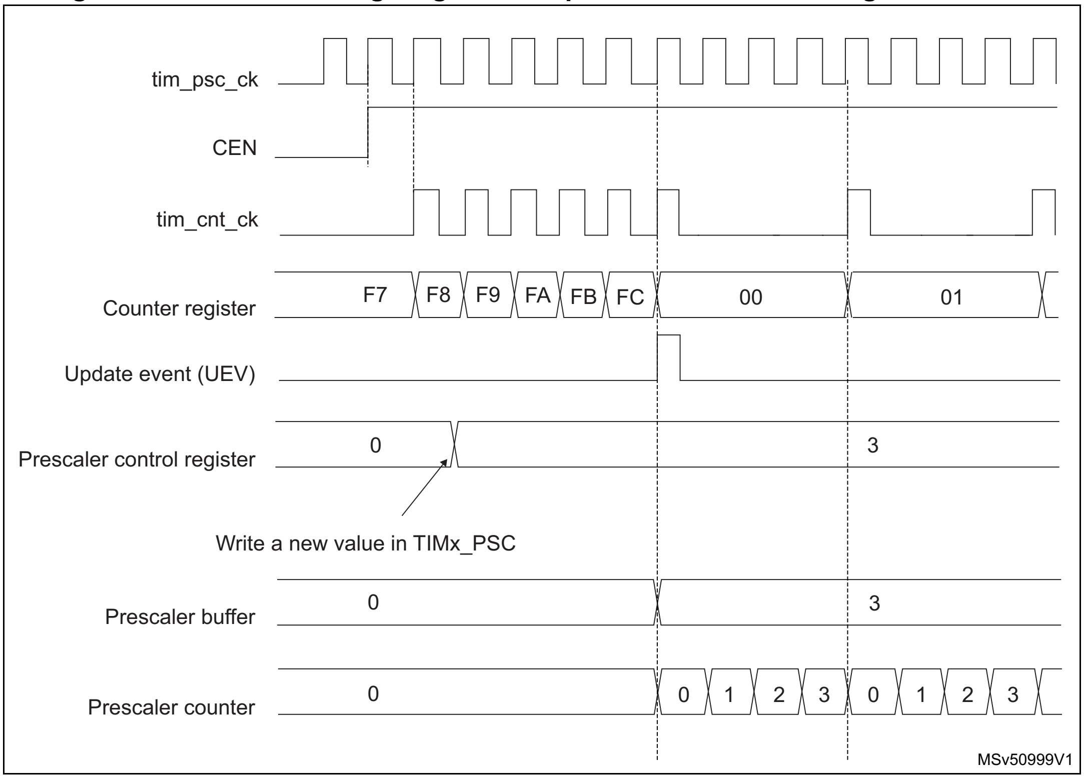


Figure 157. Counter timing diagram with prescaler division change from 1 to 4



29.3.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload 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 autoreload 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 158. Counter timing diagram, internal clock divided by 1

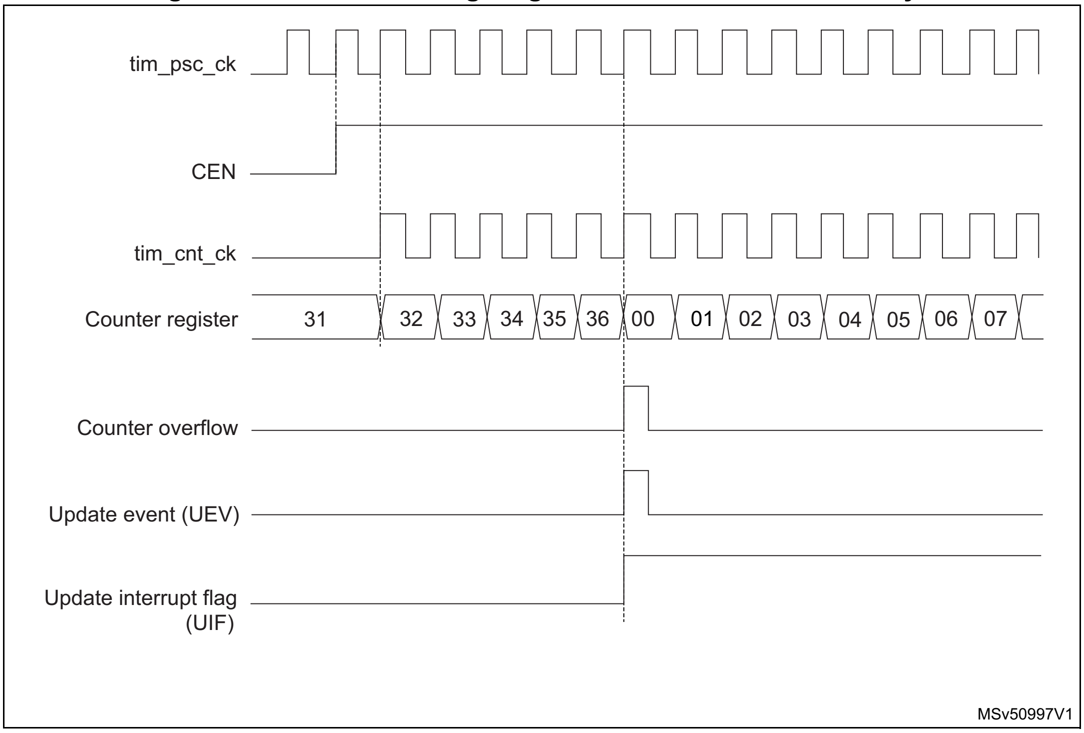


Figure 159. Counter timing diagram, internal clock divided by 2

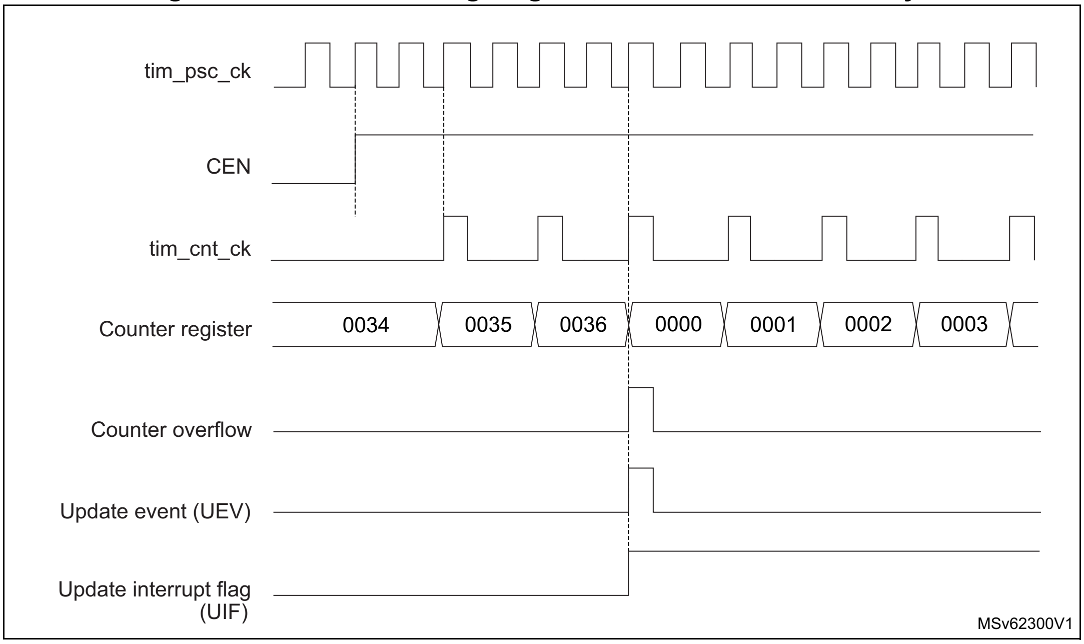


Figure 160. Counter timing diagram, internal clock divided by 4

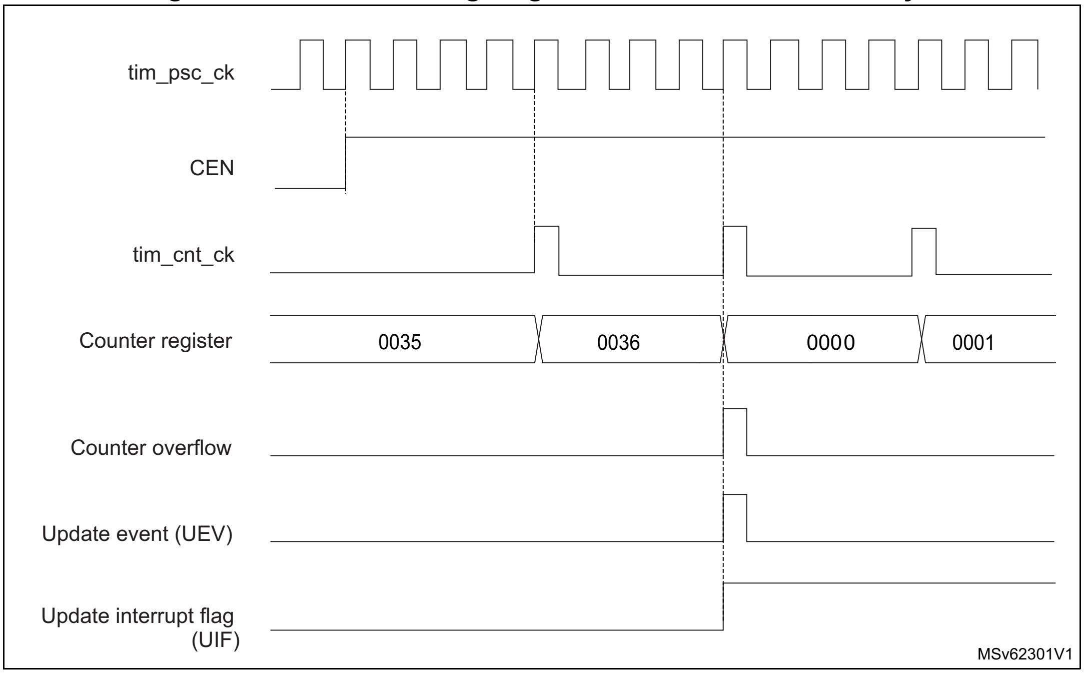


Figure 161. Counter timing diagram, internal clock divided by N

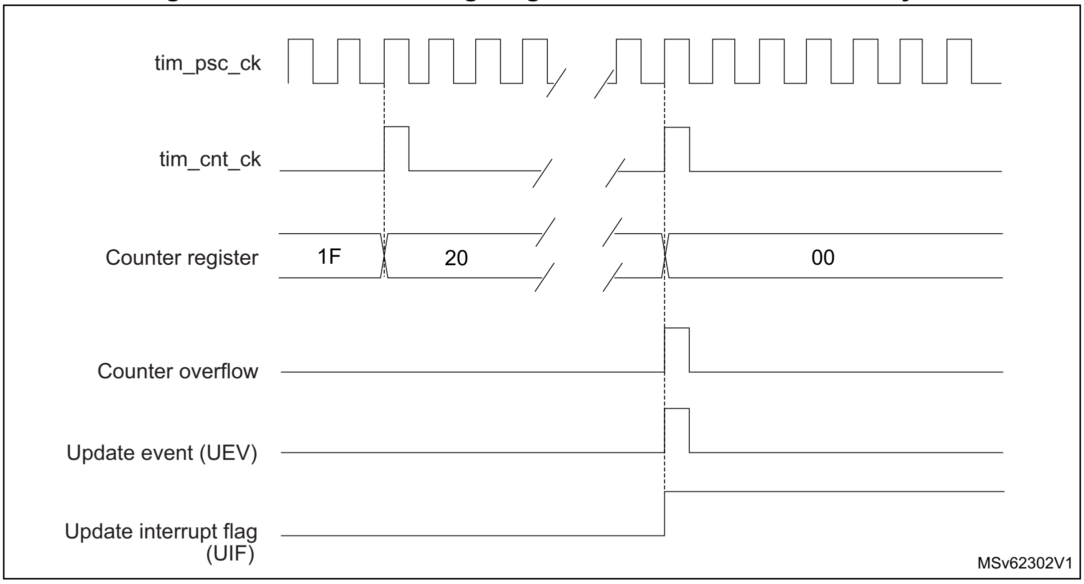


Figure 162. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

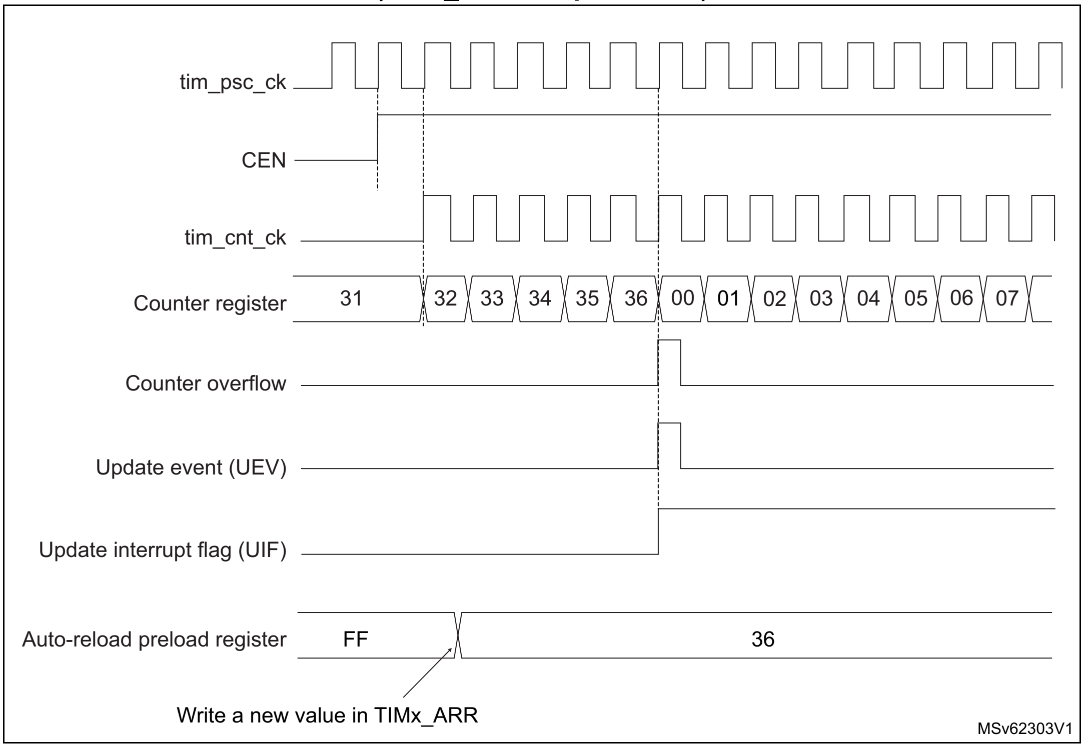
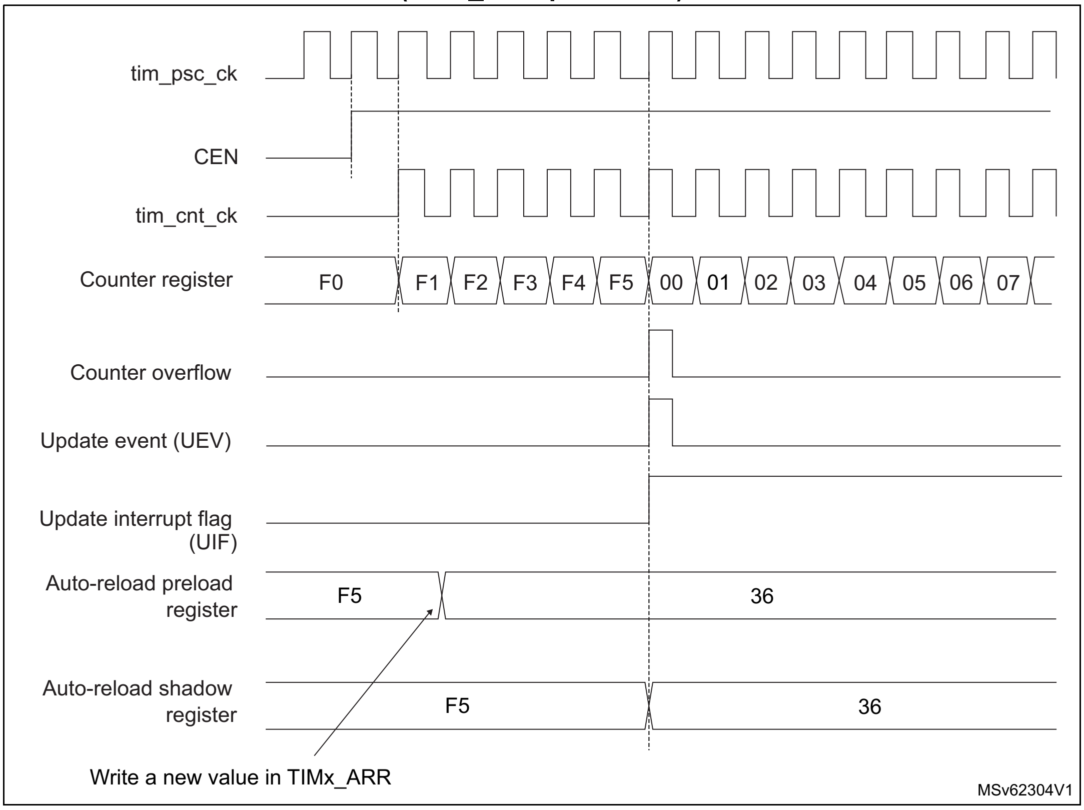


Figure 163. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the autoreload value (content of the TIMx_ARR register) down to 0, then restarts from the autoreload 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 autoreload 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 autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that the autoreload 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 164. Counter timing diagram, internal clock divided by 1

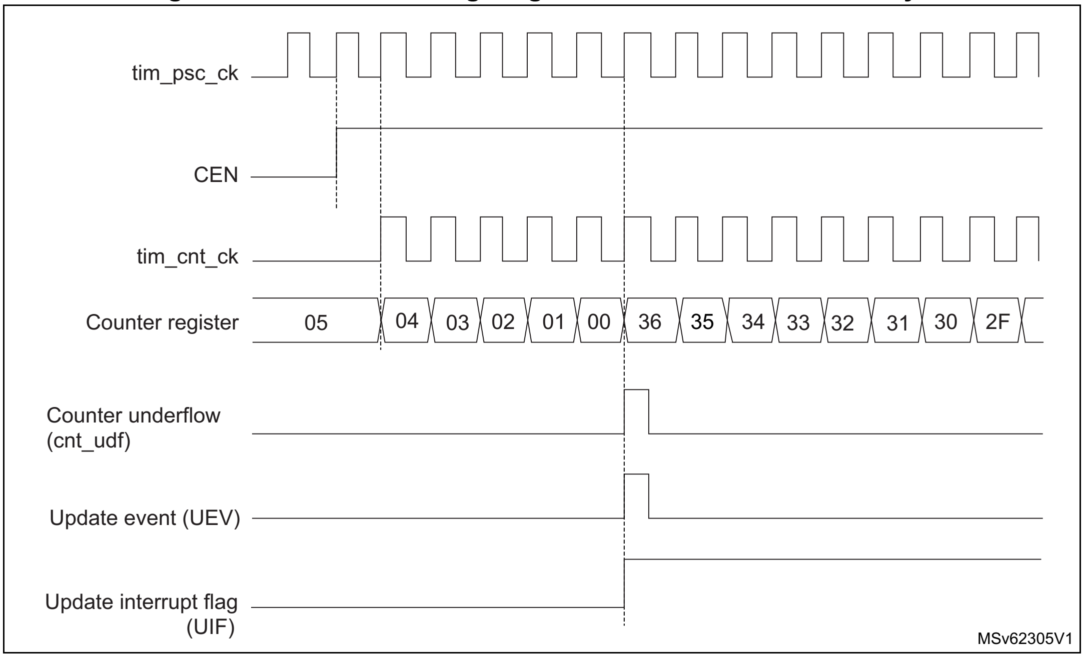


Figure 165. Counter timing diagram, internal clock divided by 2

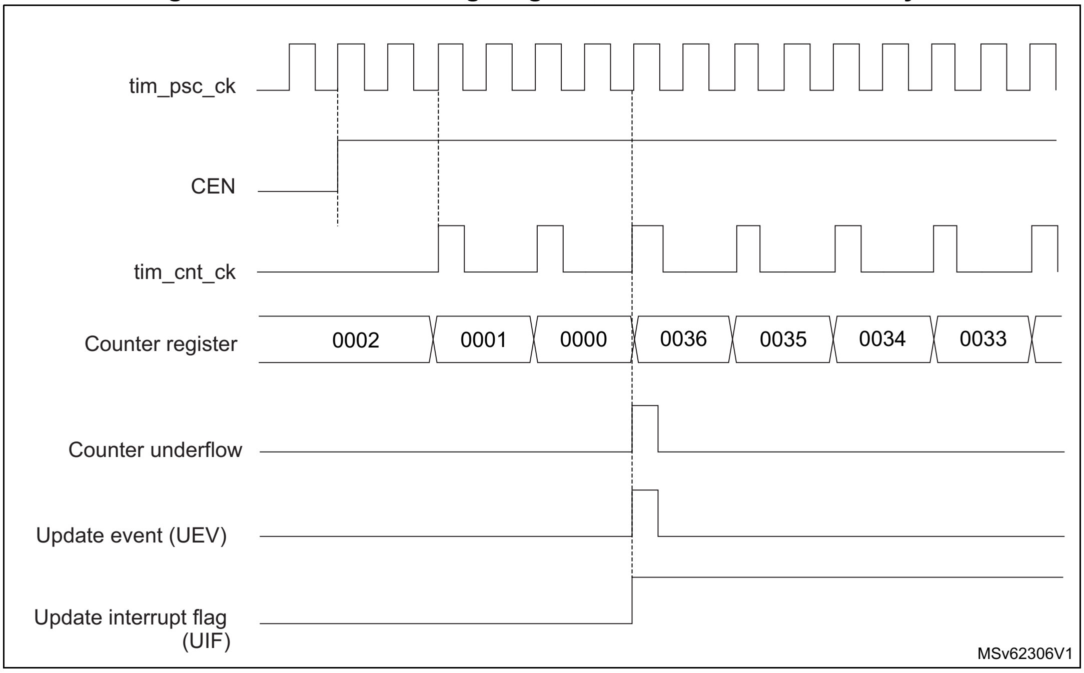


Figure 166. Counter timing diagram, internal clock divided by 4

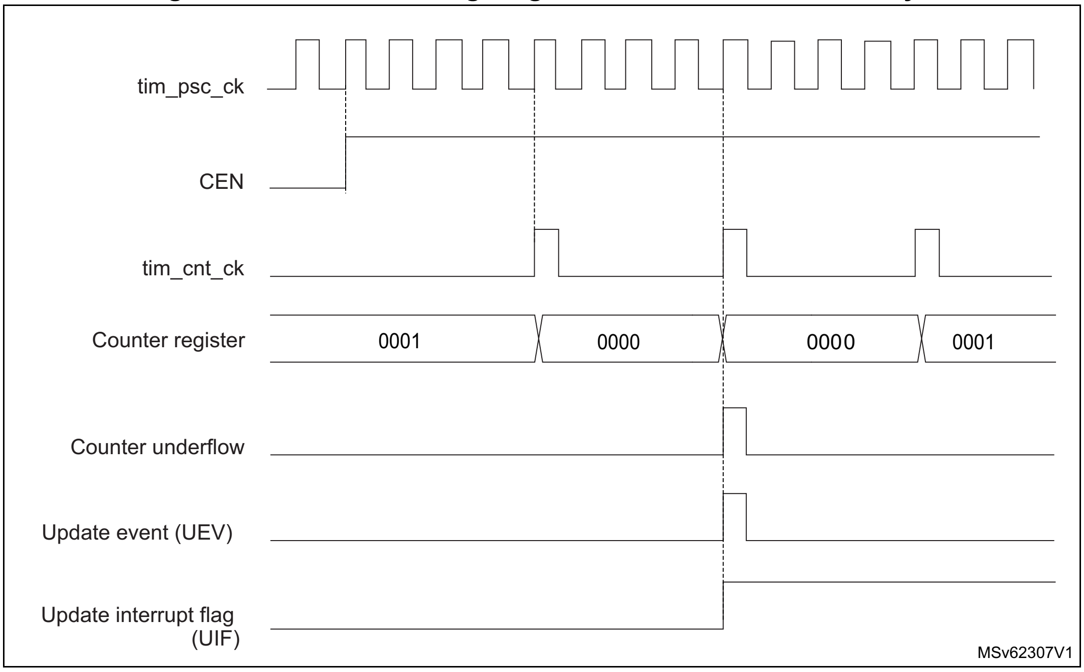


Figure 167. Counter timing diagram, internal clock divided by N

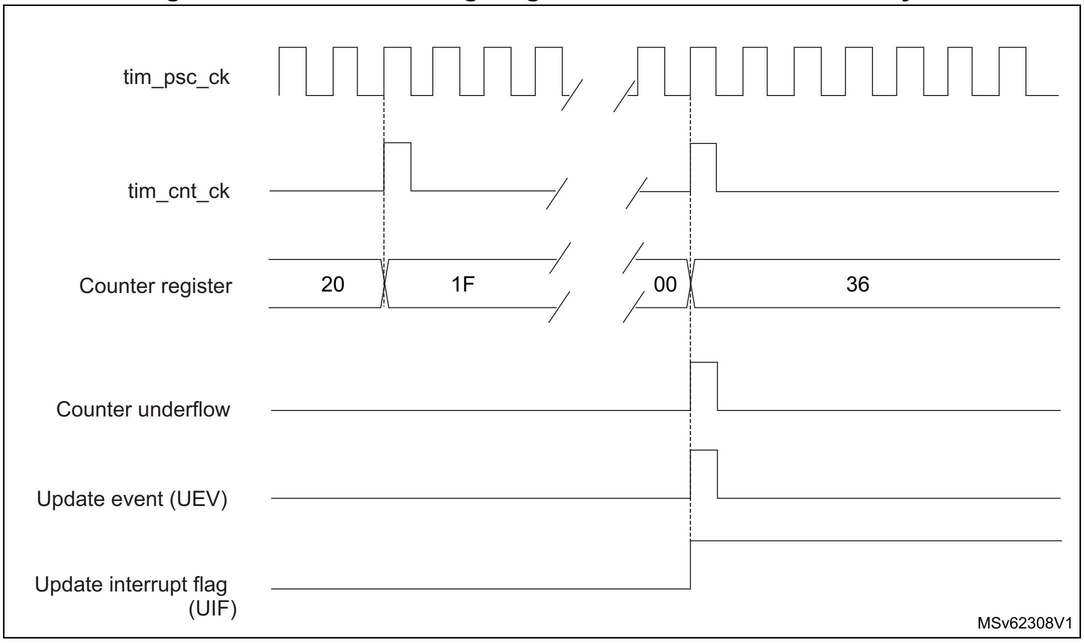
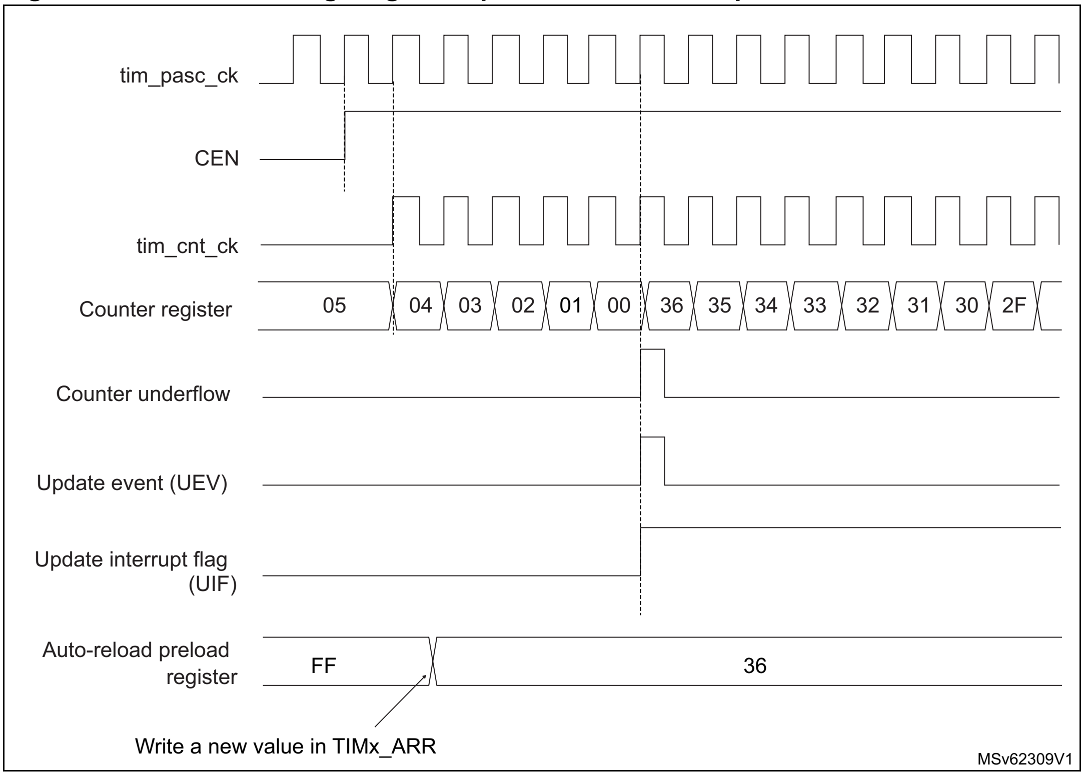


Figure 168. Counter timing diagram, update event when repetition counter is not used



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the autoreload 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 autoreload value.

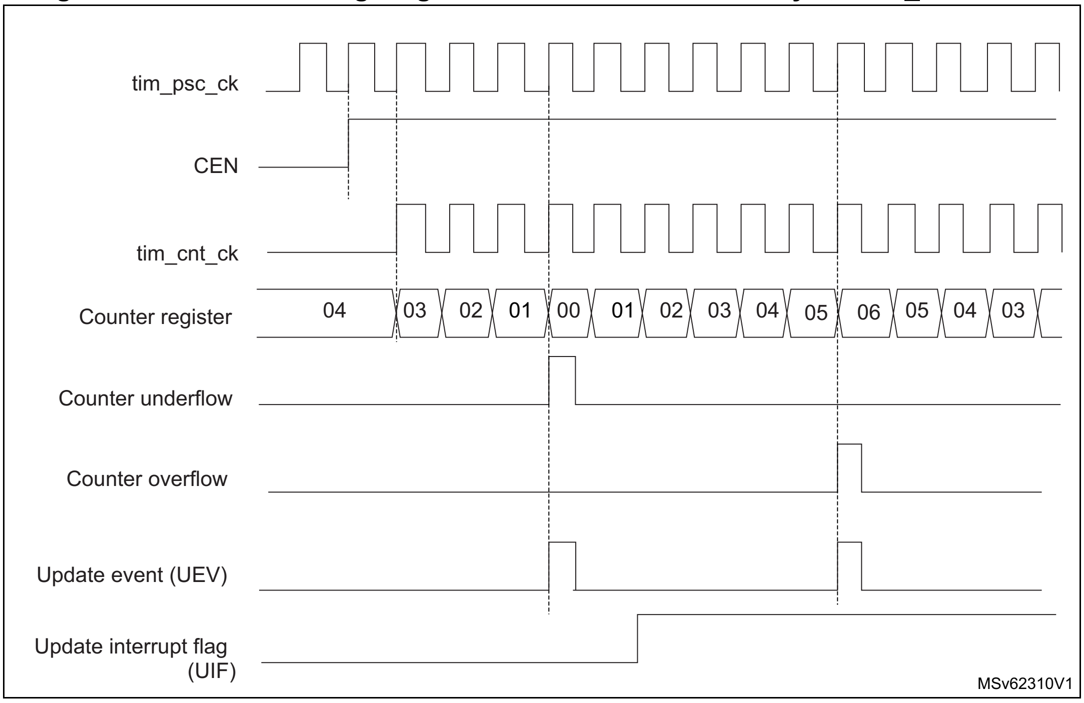
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on 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 autoreload 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 autoreload 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 169. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 29.6: TIM1 registers*).

Figure 170. Counter timing diagram, internal clock divided by 2

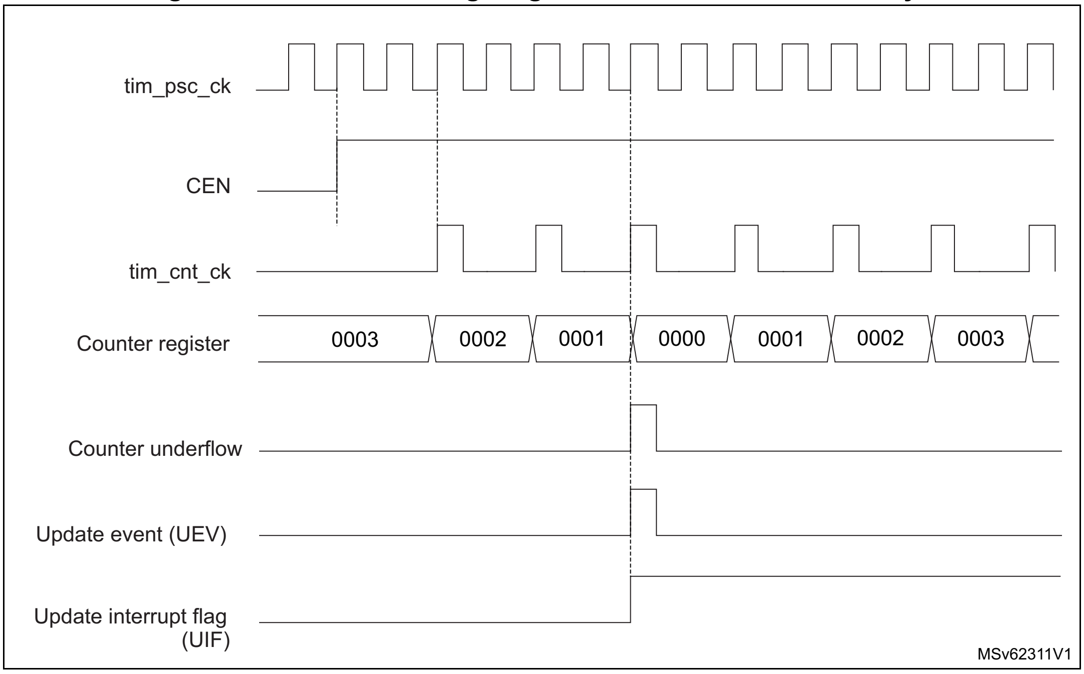


Figure 171. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36

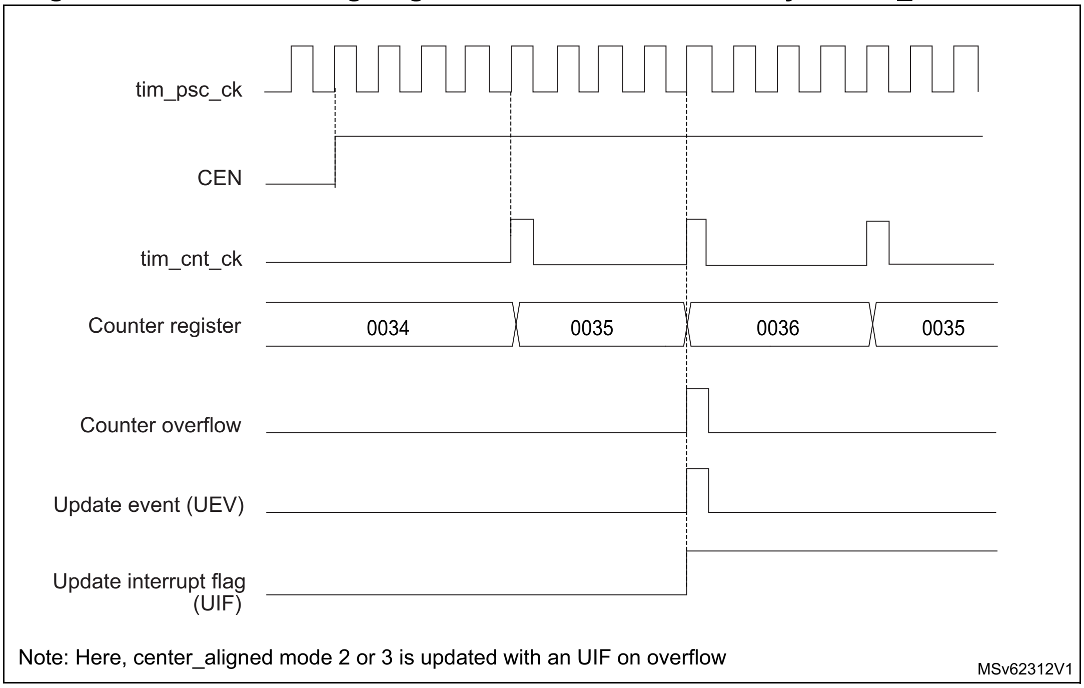


Figure 172. Counter timing diagram, internal clock divided by N

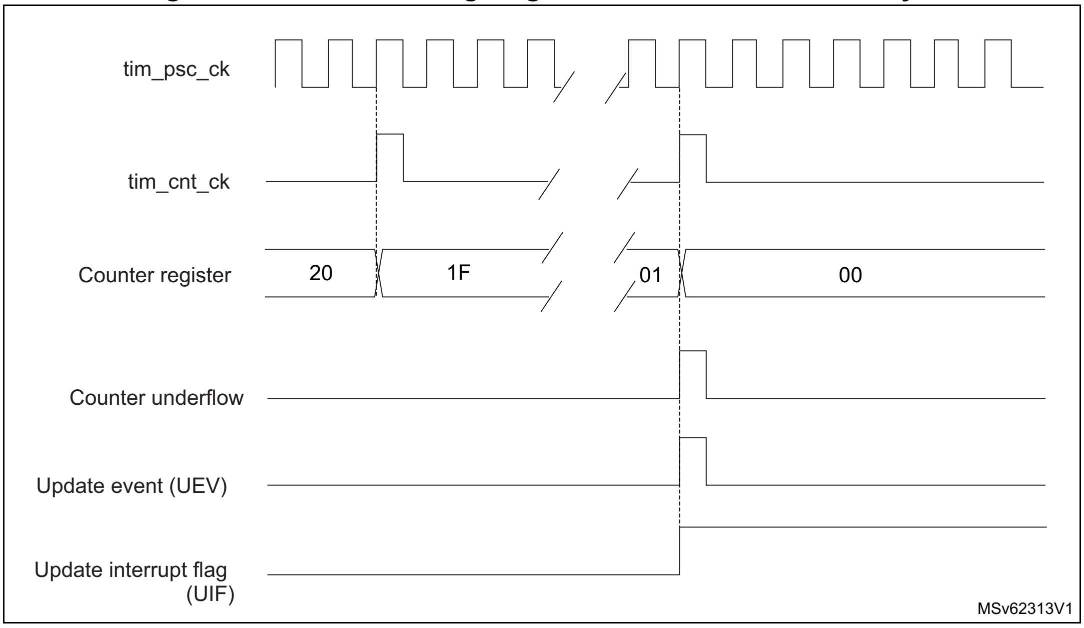


Figure 173. Counter timing diagram, update event with ARPE = 1 (counter underflow)

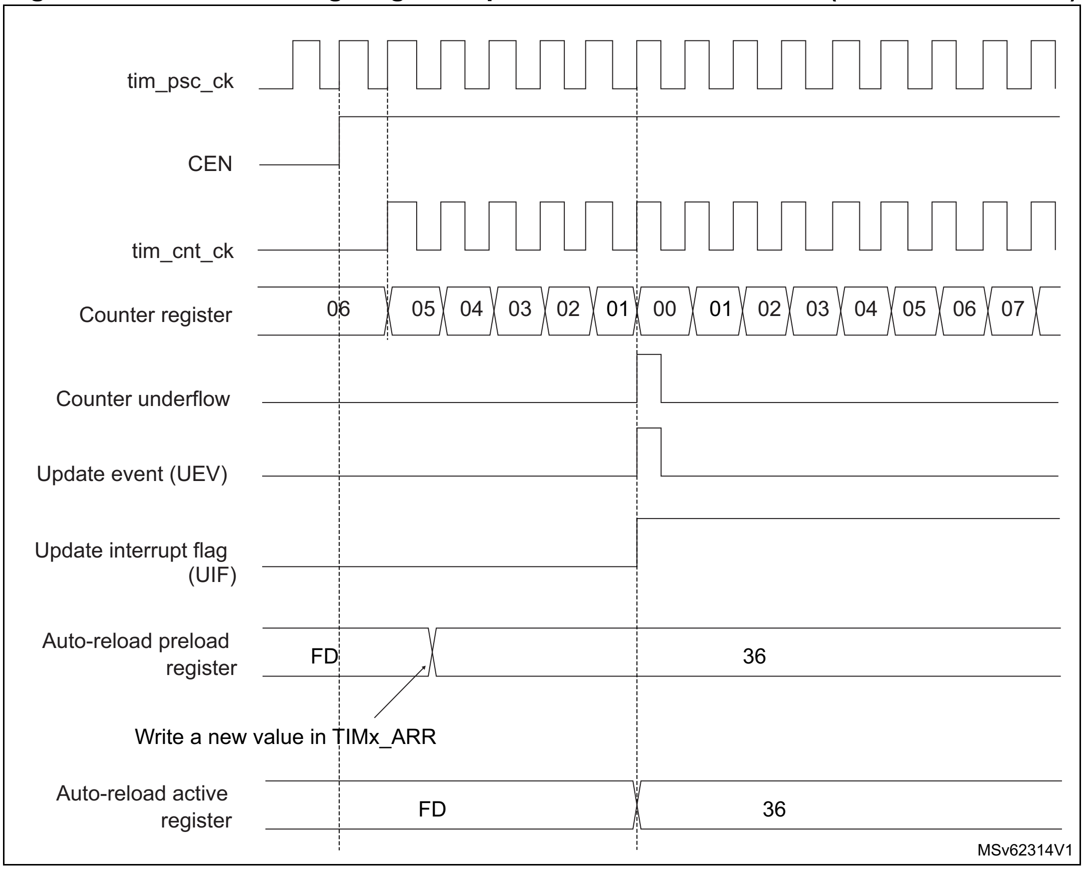
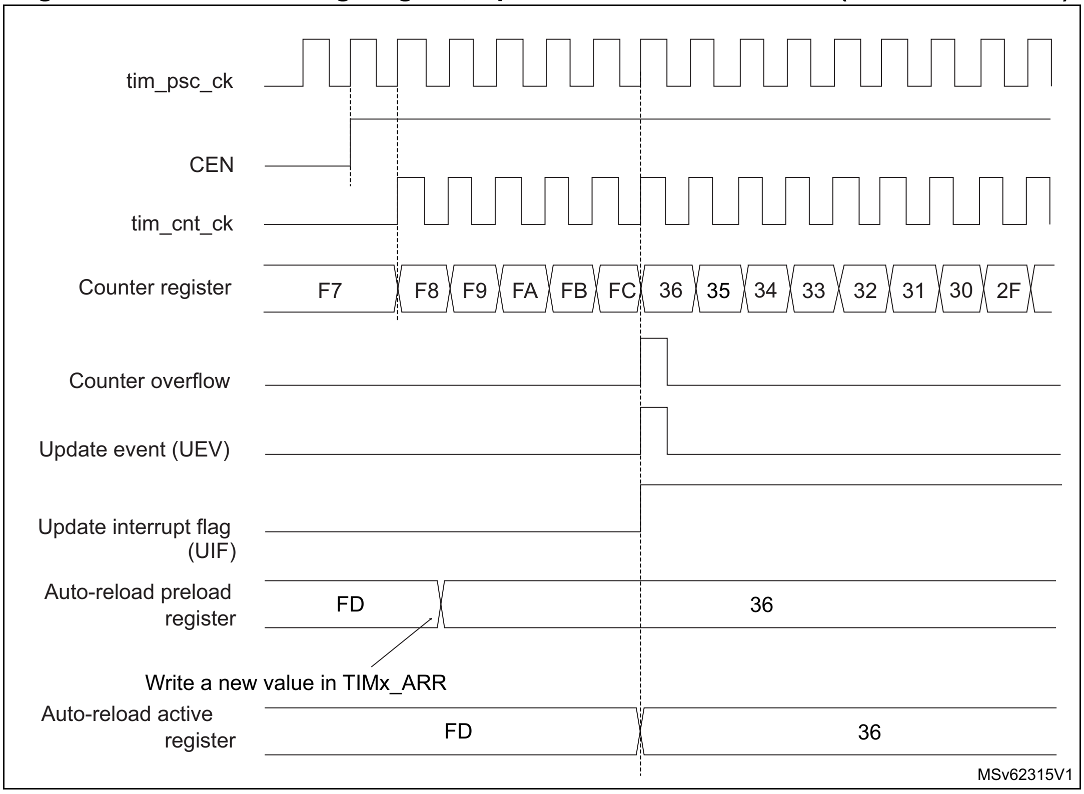


Figure 174. Counter timing diagram, Update event with ARPE = 1 (counter overflow)



29.3.5 Repetition counter

Section 29.3.3: 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 autoreload 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:

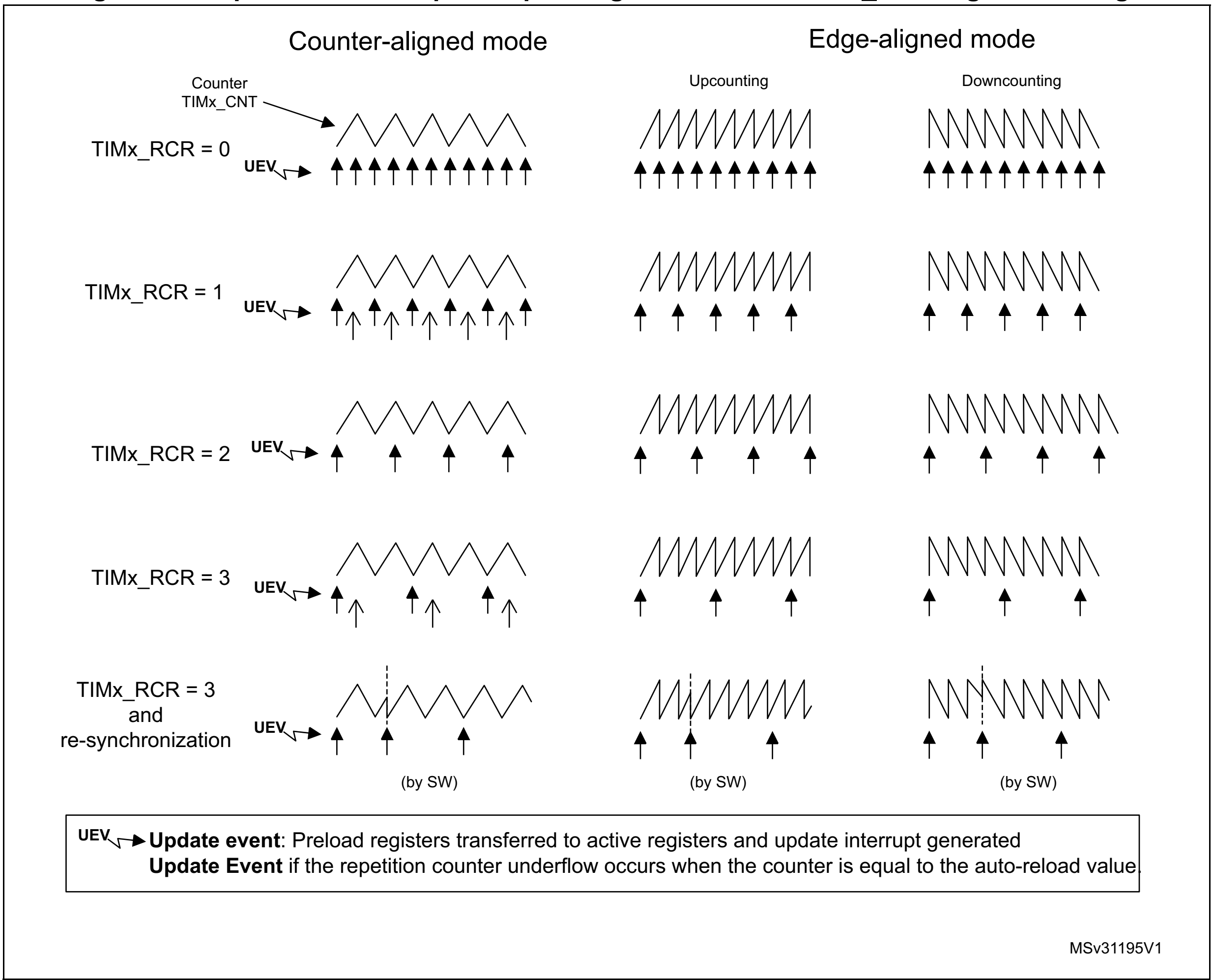
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice 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 autoreload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 175*). 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 175. Update rate examples depending on mode and TIMx_RCR register settings



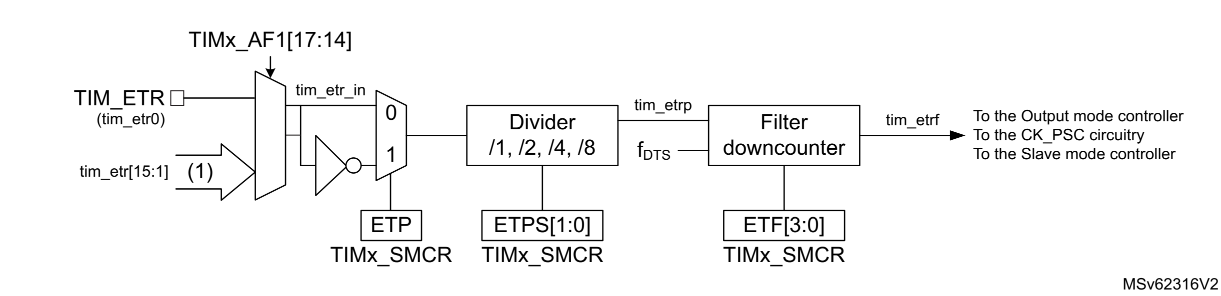
29.3.6 External trigger input

The timer features an external trigger input tim_etr_in. It can be used as:

- external clock (external clock mode 2, see *Section 29.3.7*)
- trigger for the slave mode (see *Section 29.3.30*)
- PWM reset input for cycle-by-cycle current regulation (see *Section 29.3.9*)

Figure 176 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 176. 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 29.3.2: TIM1 pins and internal signals* for the list of sources connected to the etr_in input in the product.

29.3.7 Clock selection

The counter clock can be provided by the following clock sources:

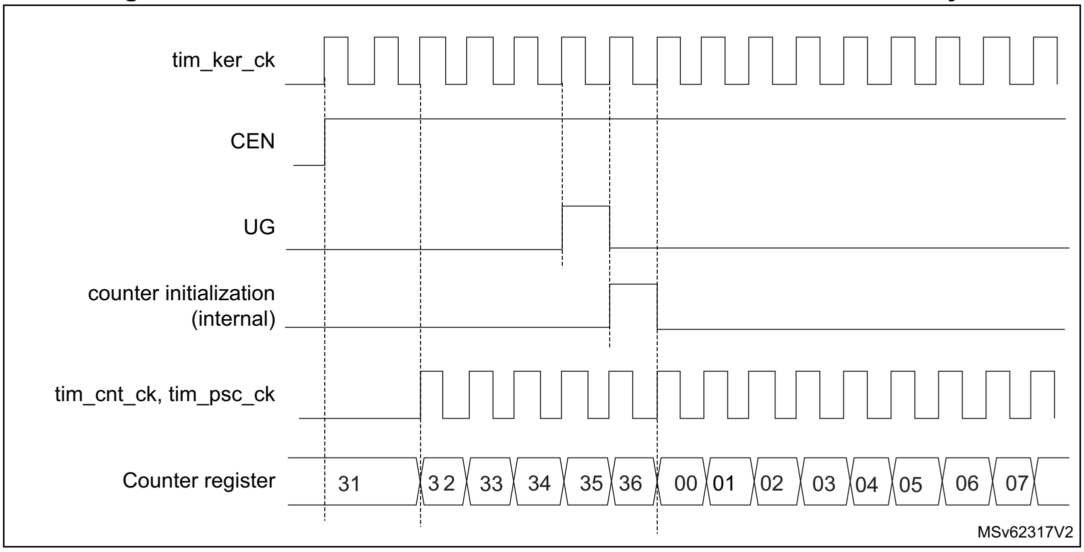
- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2)
- External clock mode2: external trigger input (tim_etr_in)
- Encoder mode

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 177 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

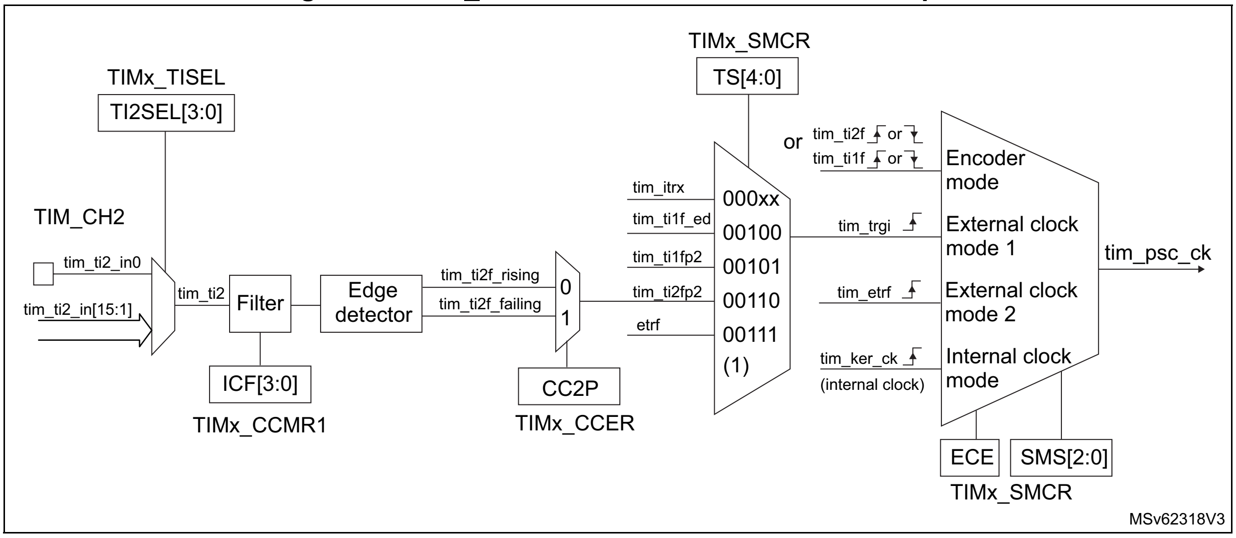
Figure 177. Control circuit in normal mode, internal clock divided by 1



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 178. 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:

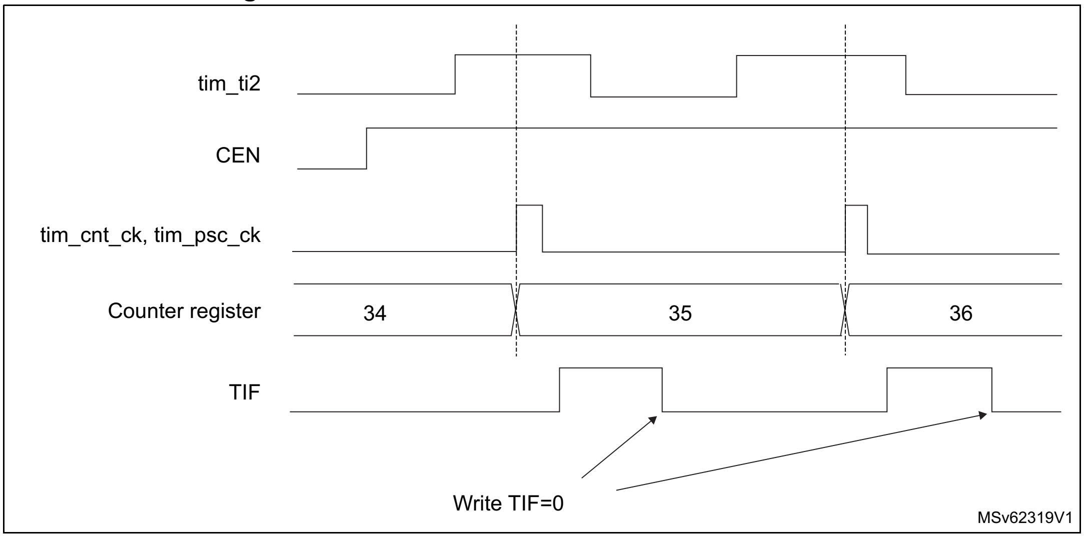
1. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F = 0000).
3. Select rising edge polarity by writing CC2P = 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 179. Control circuit in external clock mode 1



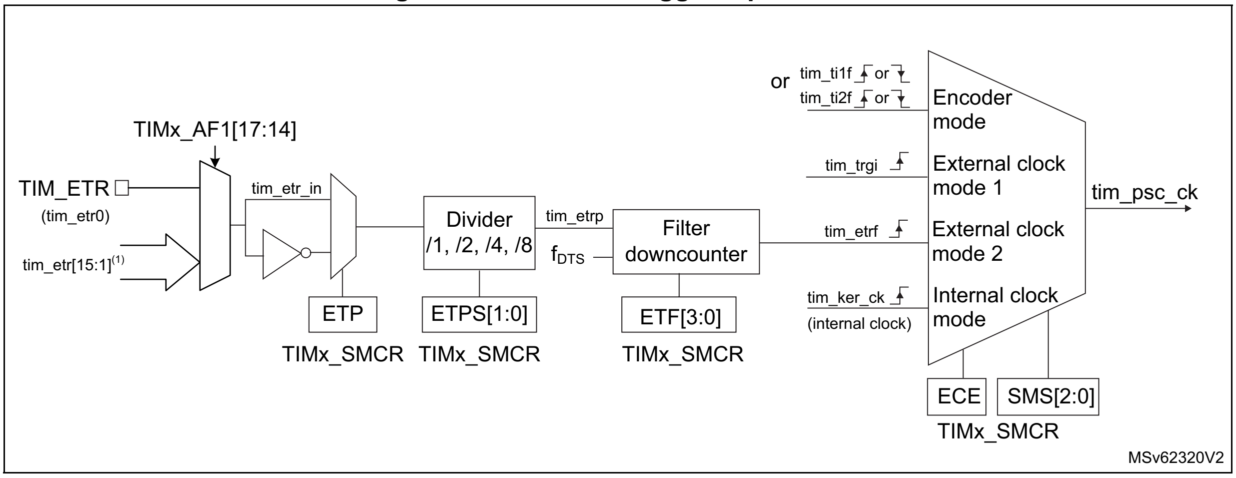
External clock source mode 2

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 180* gives an overview of the external trigger input block.

Figure 180. External trigger input block



1. Refer to *Section 29.3.2: TIM1 pins and internal signals*.

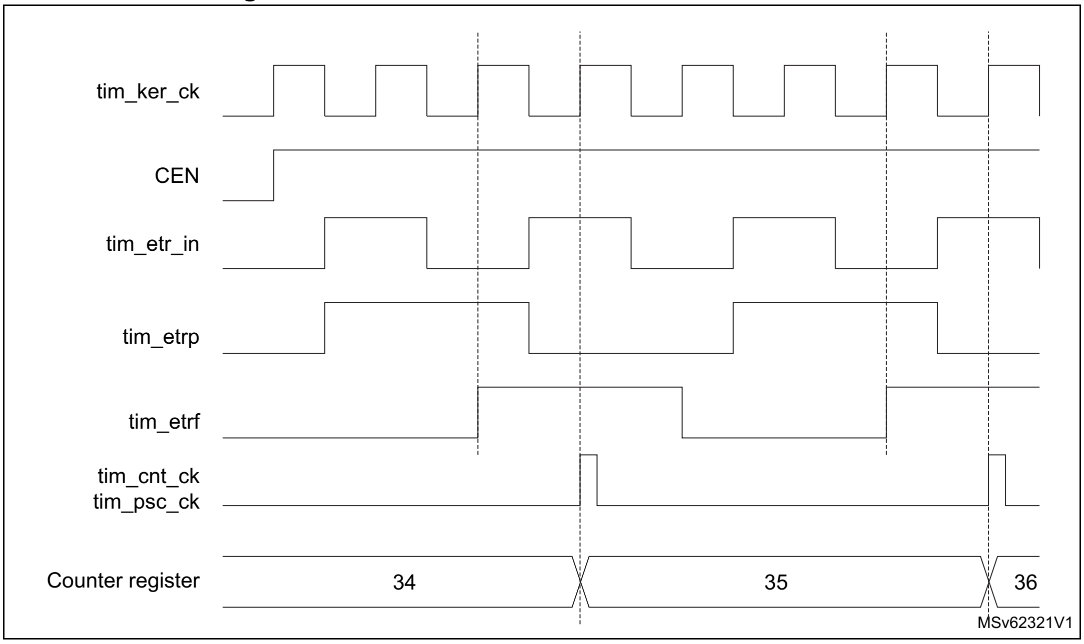
For example, to configure the upcounter to count each 2 rising edges on tim_etr_in, use the following procedure:

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 must apply a division of the external signal by a proper ETPS prescaler setting.

Figure 181. Control circuit in external clock mode 2



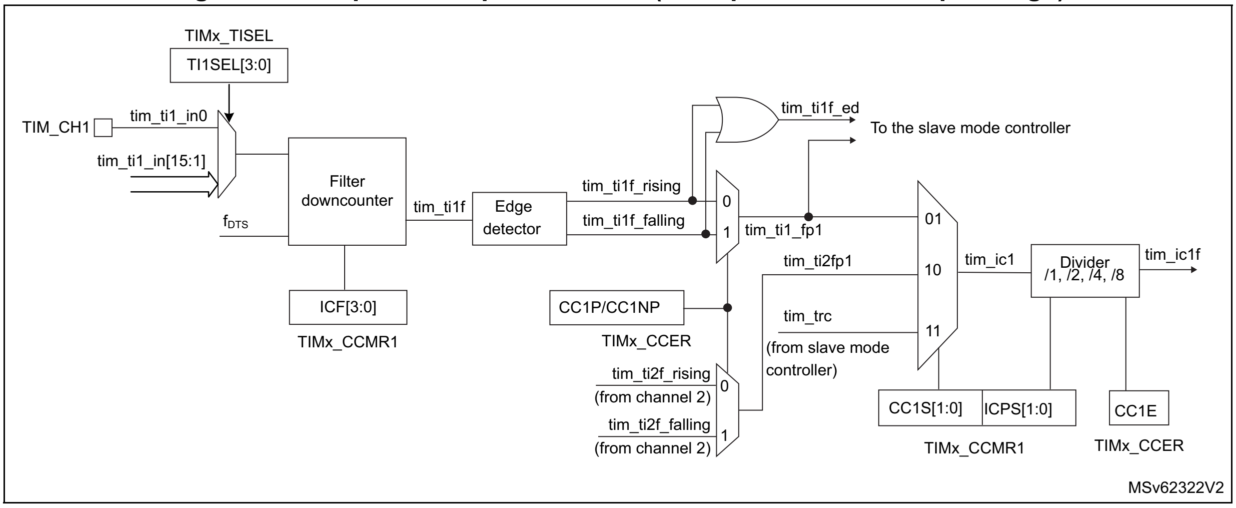
29.3.8 Capture/compare channels

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 182 to *Figure 185* 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 182. 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 183. Capture/compare channel 1 main circuit

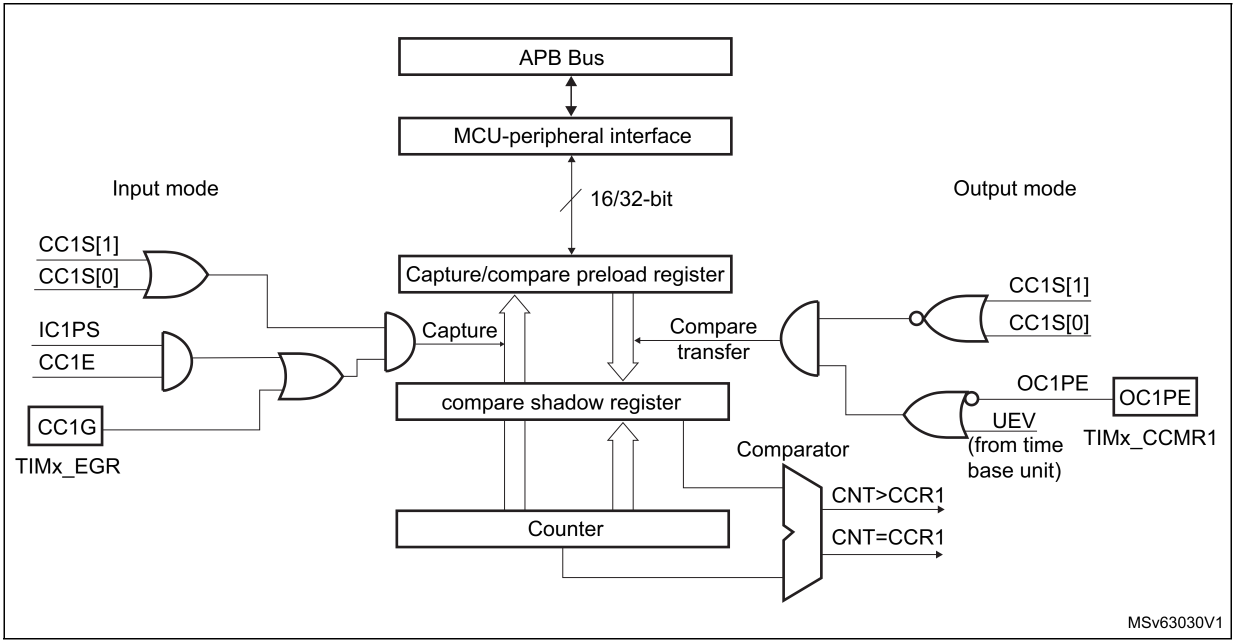
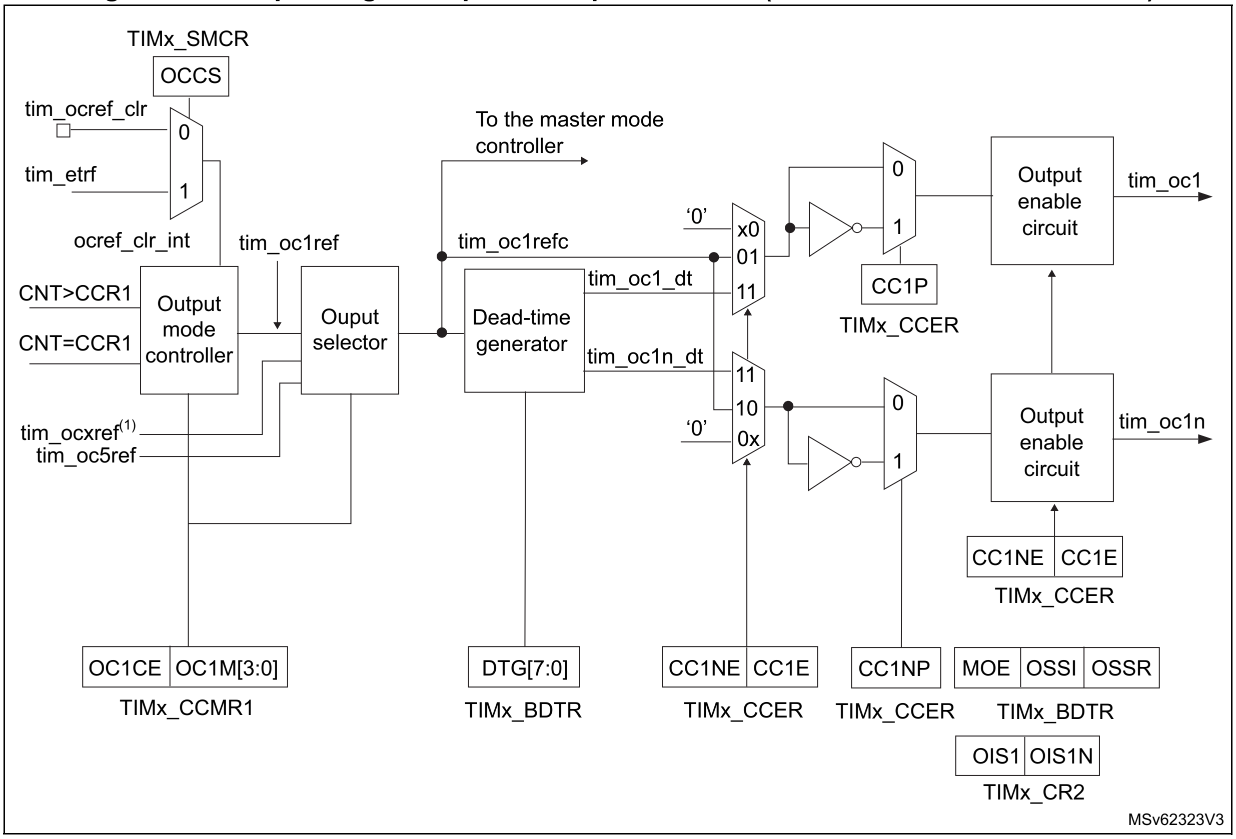
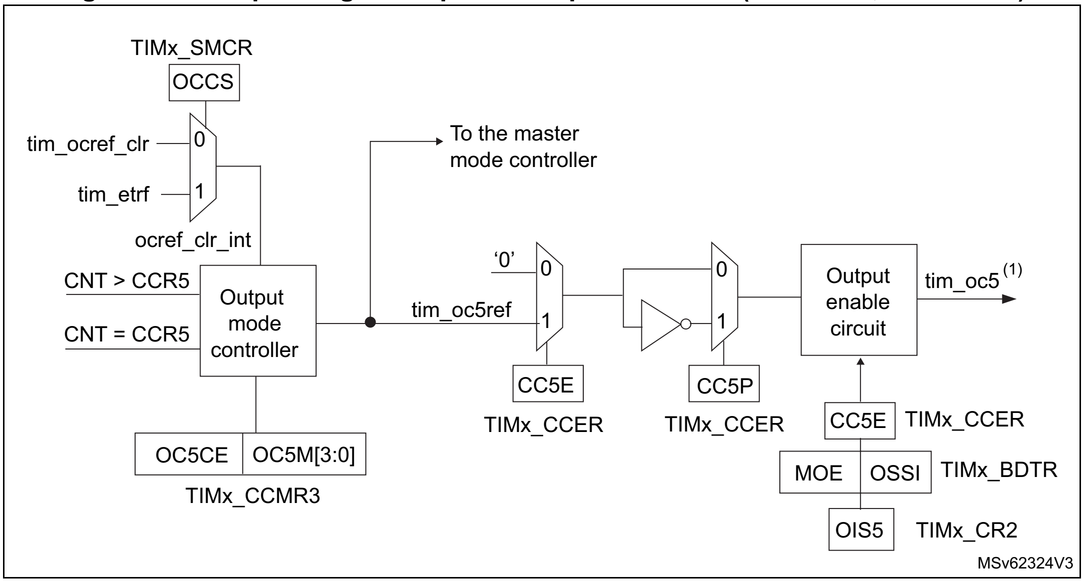


Figure 184. Output stage of capture/compare channel (channel 1, idem ch. 2, 3 and 4)



1. tim_ocxref, where x is the rank of the complementary channel

Figure 185. 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.

29.3.9 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 overcapture 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 five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on tim_ti1 when eight 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 may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

29.3.10 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.

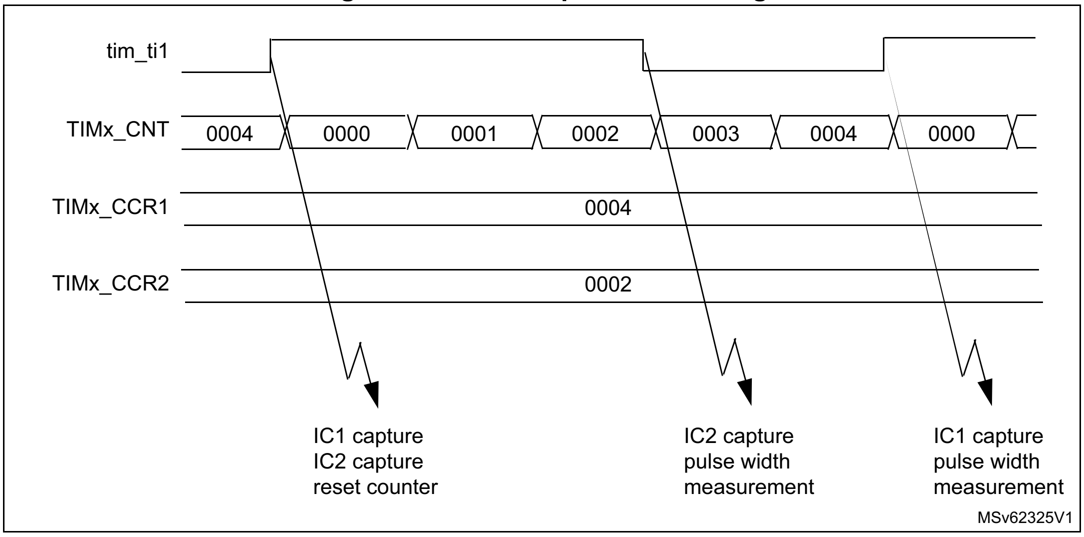
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tixfp1 input.
- These two 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 186. PWM input mode timing



29.3.11 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.

29.3.12 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.

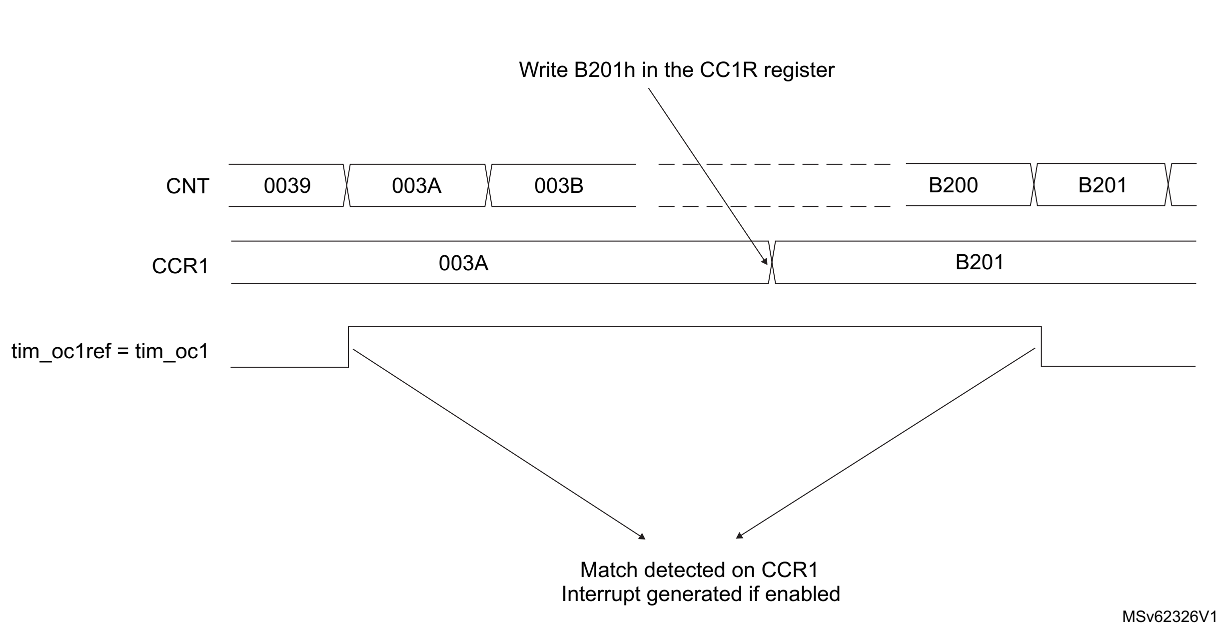
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

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 187*.

Figure 187. Output compare mode, toggle on tim_oc1



29.3.13 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 autoreload 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.

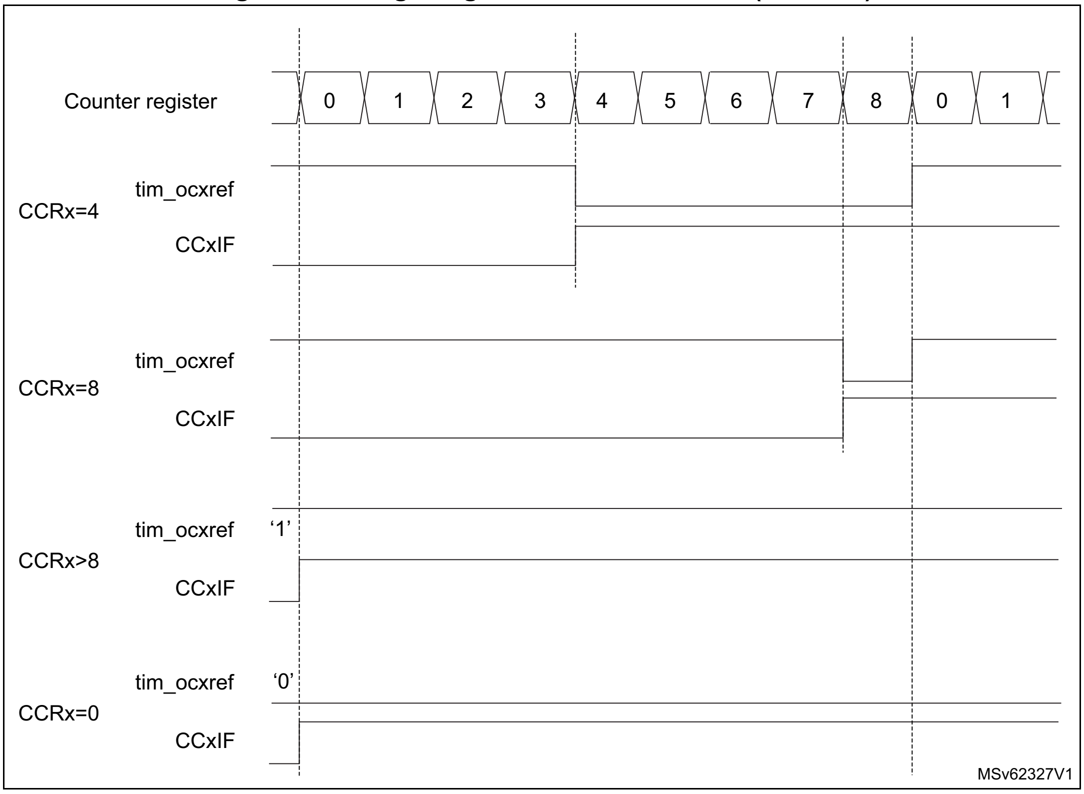
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Upcounting mode*.

  In the following example, the mode is 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 autoreload value (in TIMx_ARR) then tim_ocxref is held at 1. If the compare value is zero then tim_ocxref is held at 0. *Figure 188* shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

Figure 188. 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*

  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 autoreload value in TIMx_ARR, then tim_ocxref is held at 1. 0% PWM is not possible in this mode.

PWM center-aligned mode

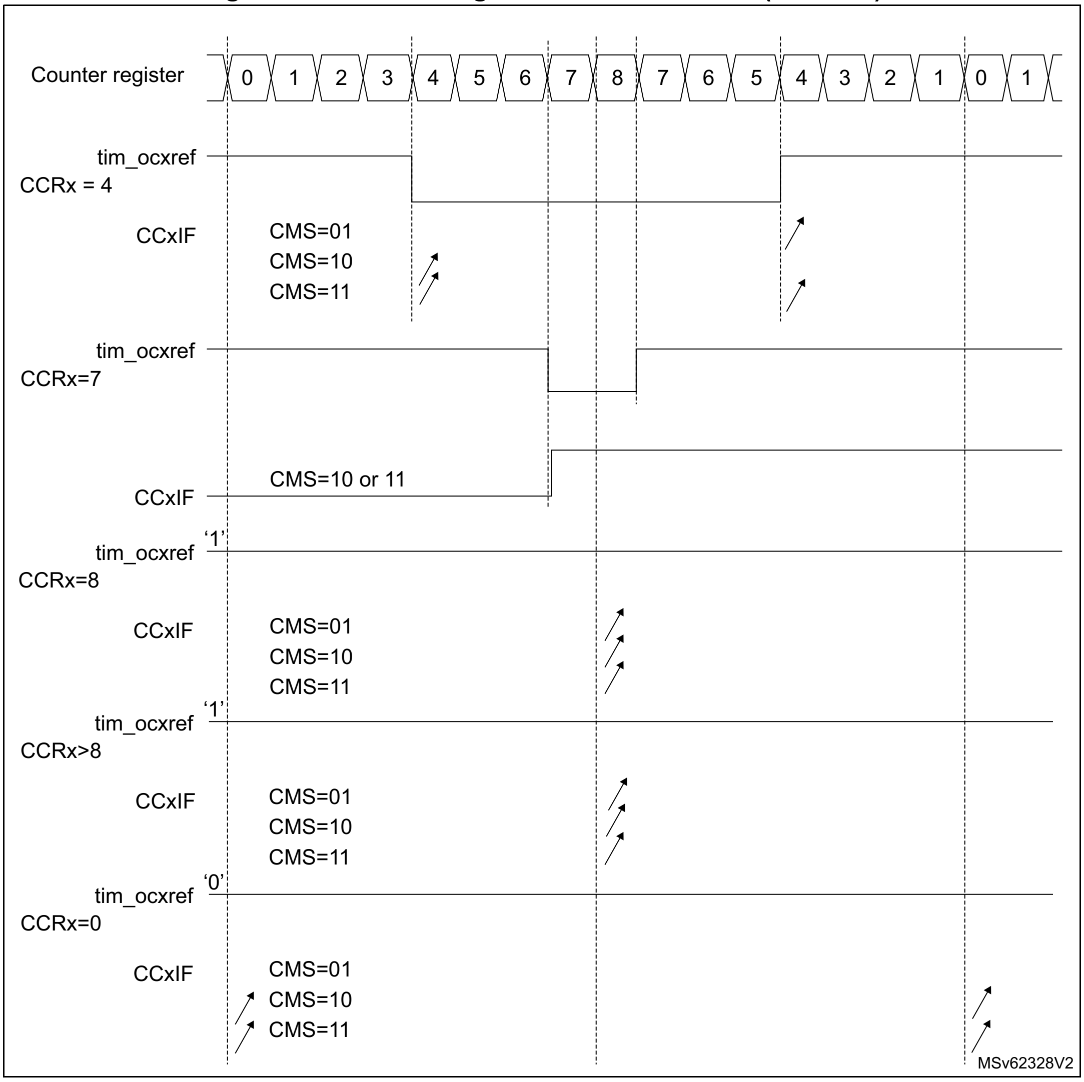
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending 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 *Center-aligned mode (up/down counting)*.

Figure 189 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 189. Center-aligned PWM waveforms (ARR = 8)



Hints on using center-aligned mode:

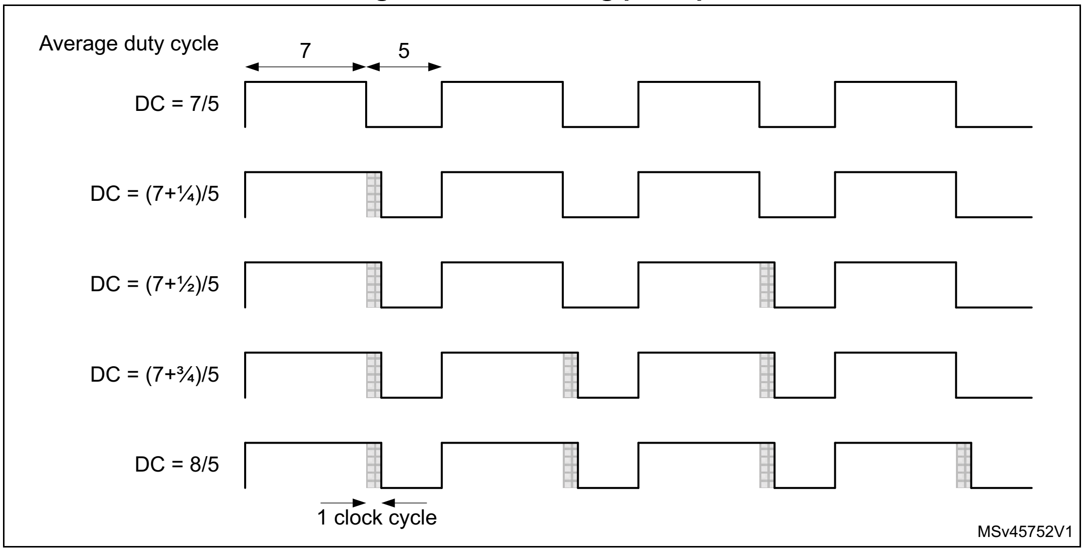
- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the 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 autoreload 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. *Figure 190* presents the dithering principle applied to four consecutive PWM cycles.

Figure 190. Dithering principle

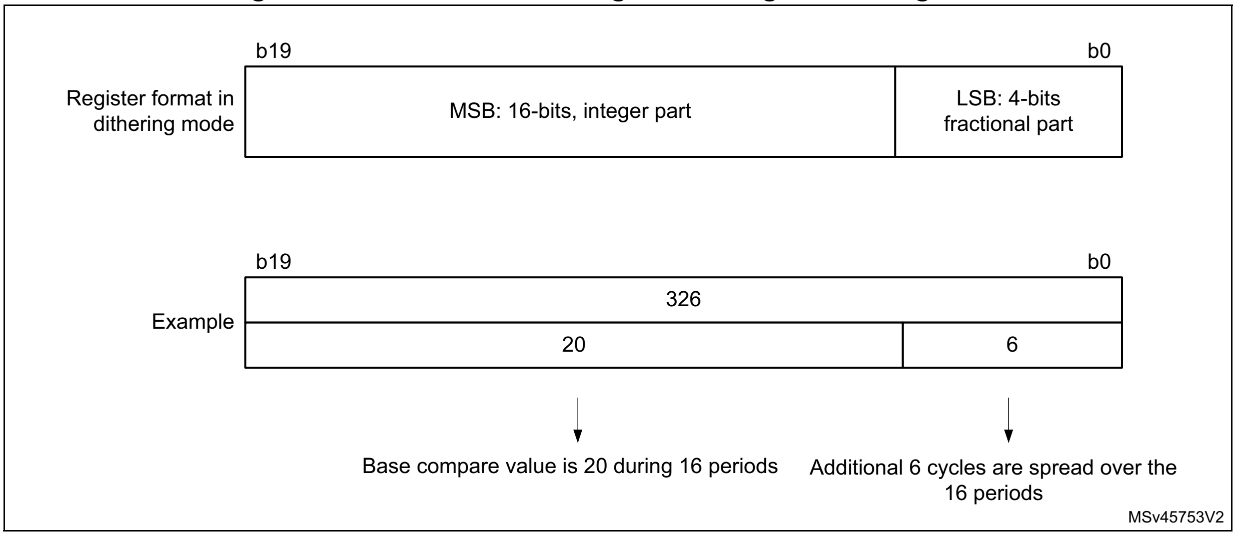


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 191* for example):

- The four 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 DITHEN bit must be reset.
3. The CCIF flags must be cleared.
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 191. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

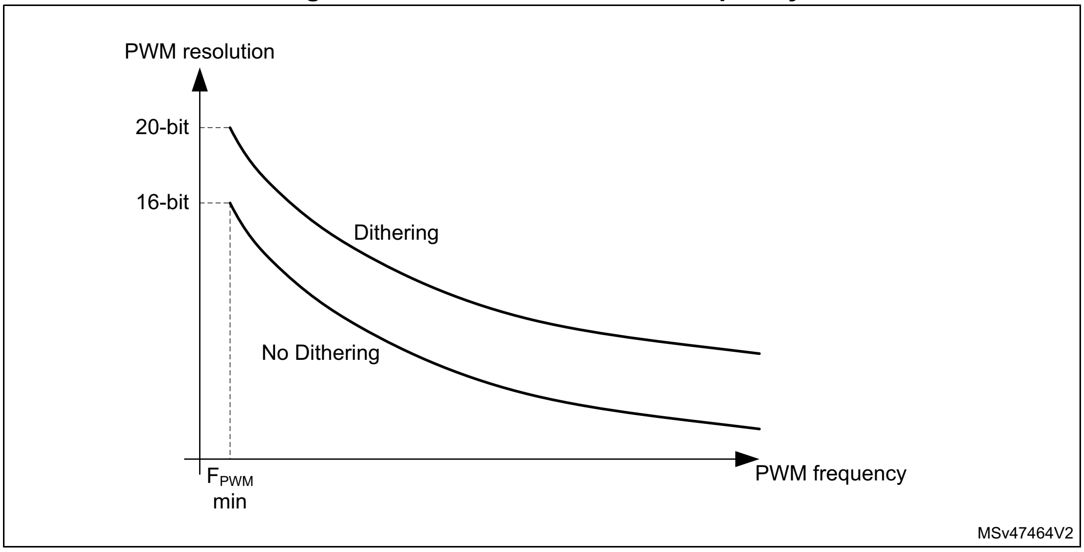
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

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).

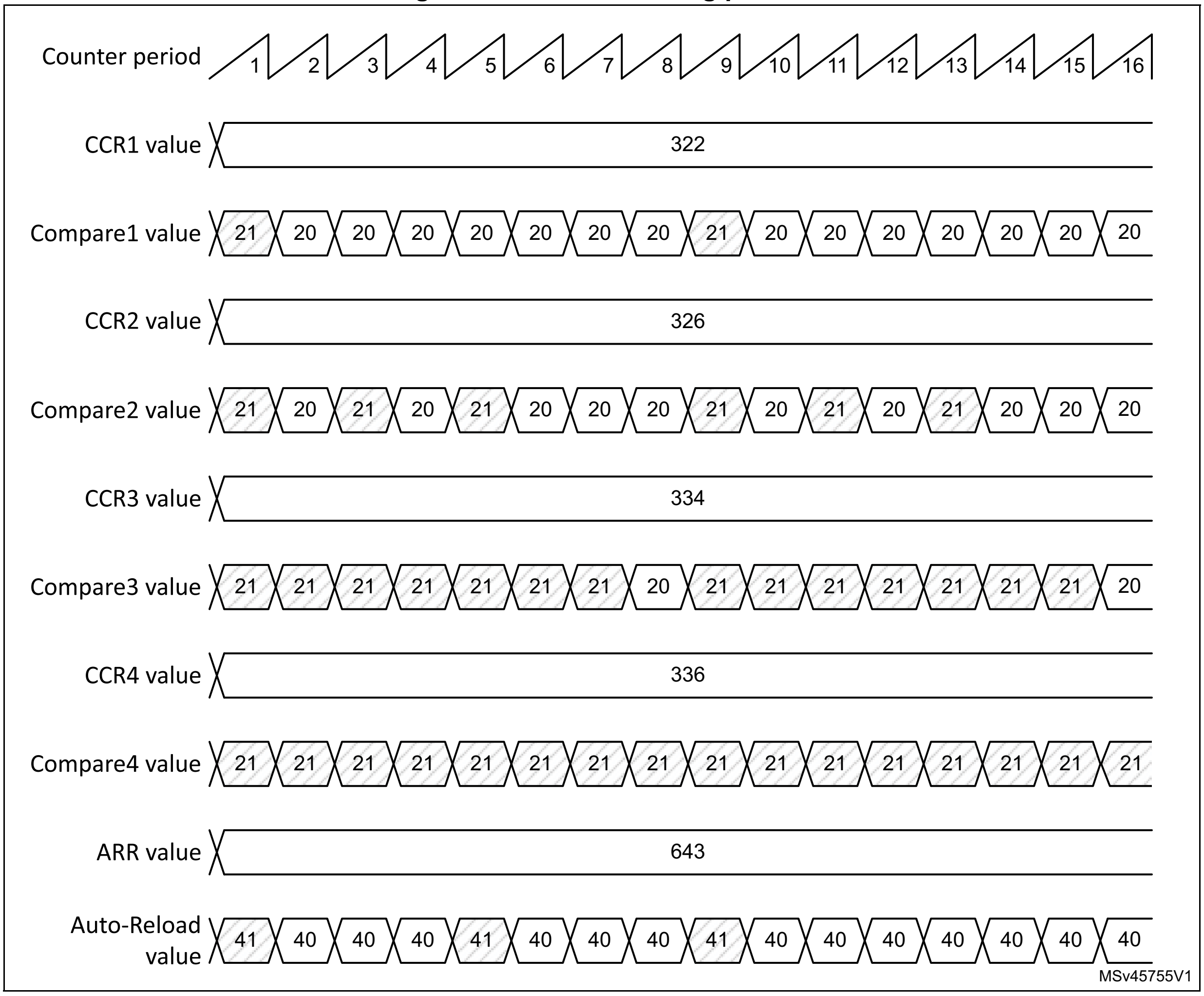
As shown on *Figure 192*, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 192. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 193*.

Figure 193. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 265*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 265. 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 265. 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 eight consecutive PWM periods, considering the up and down counting phases as shown in *Figure 194*.

Figure 194. Dithering effect on duty cycle in center-aligned PWM mode

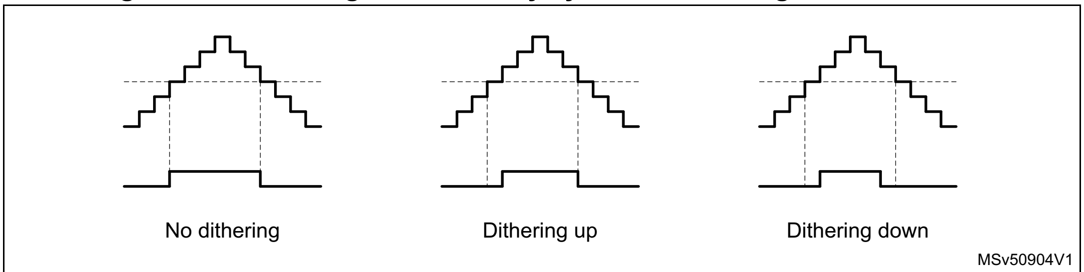


Table 266 shows how the dithering pattern is added in center-aligned PWM mode.

Table 266. 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 266. 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	-

29.3.14 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

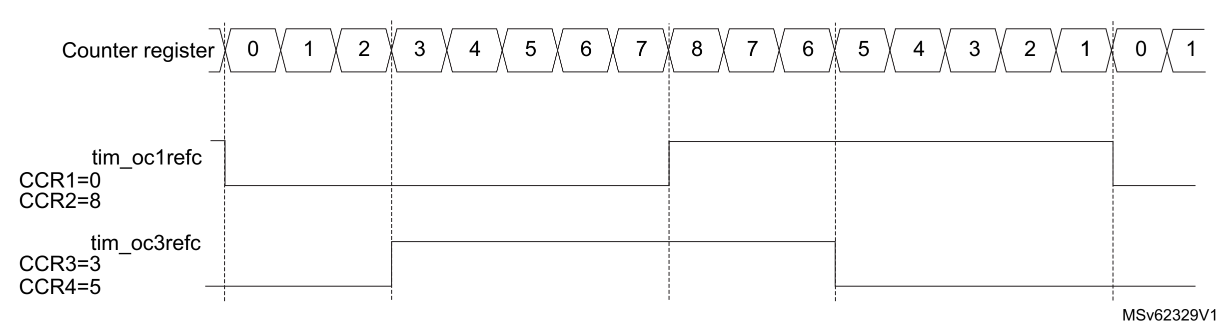
Asymmetric PWM mode can be selected independently on two channel (one tim_ocx output per pair of CCR registers) by writing 1110 (Asymmetric PWM mode 1) or 1111 (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three 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 195 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 195. Generation of 2 phase-shifted PWM signals with 50% duty cycle



29.3.15 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.

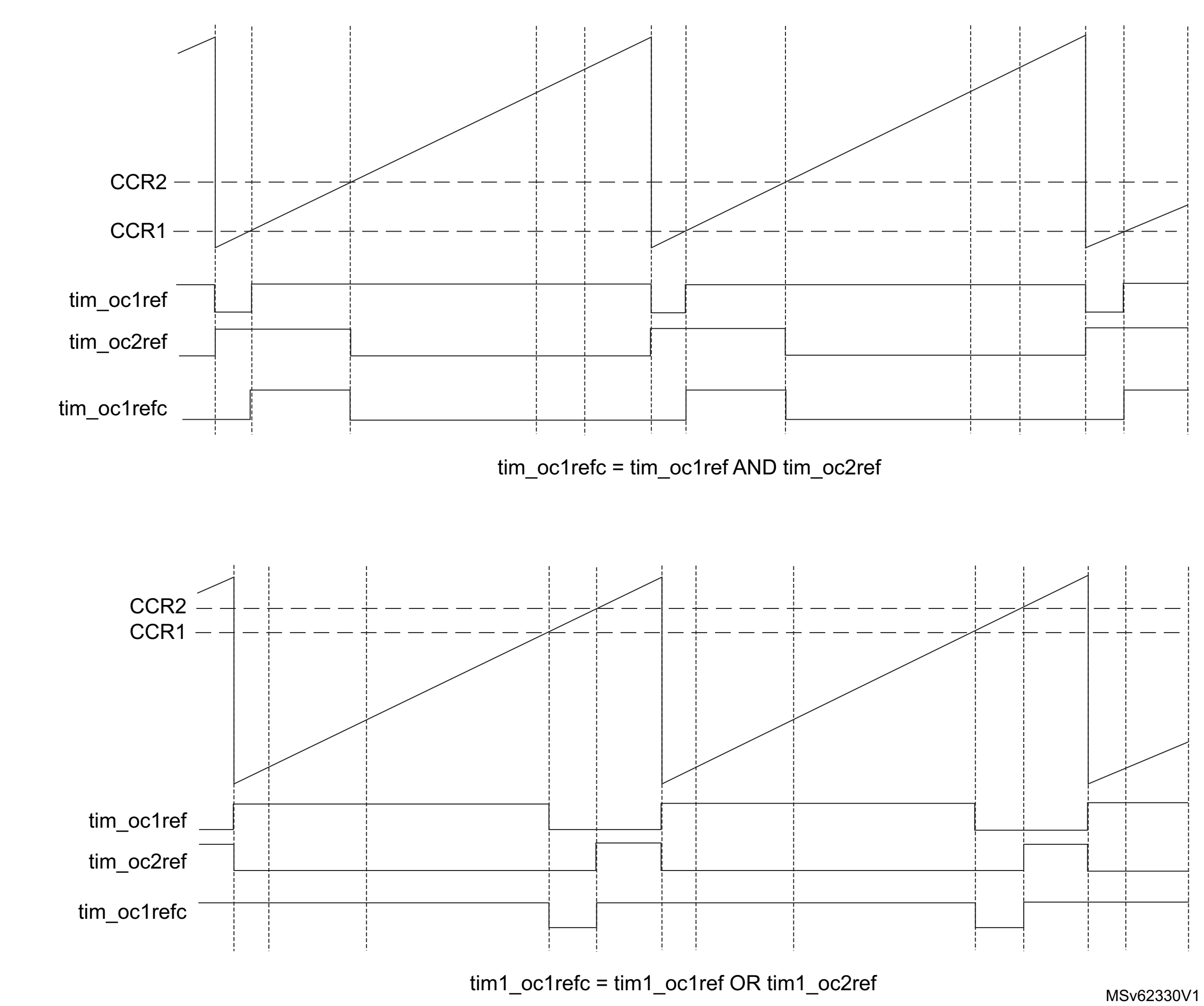
When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Figure 196 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 196. Combined PWM mode on channel 1 and 3



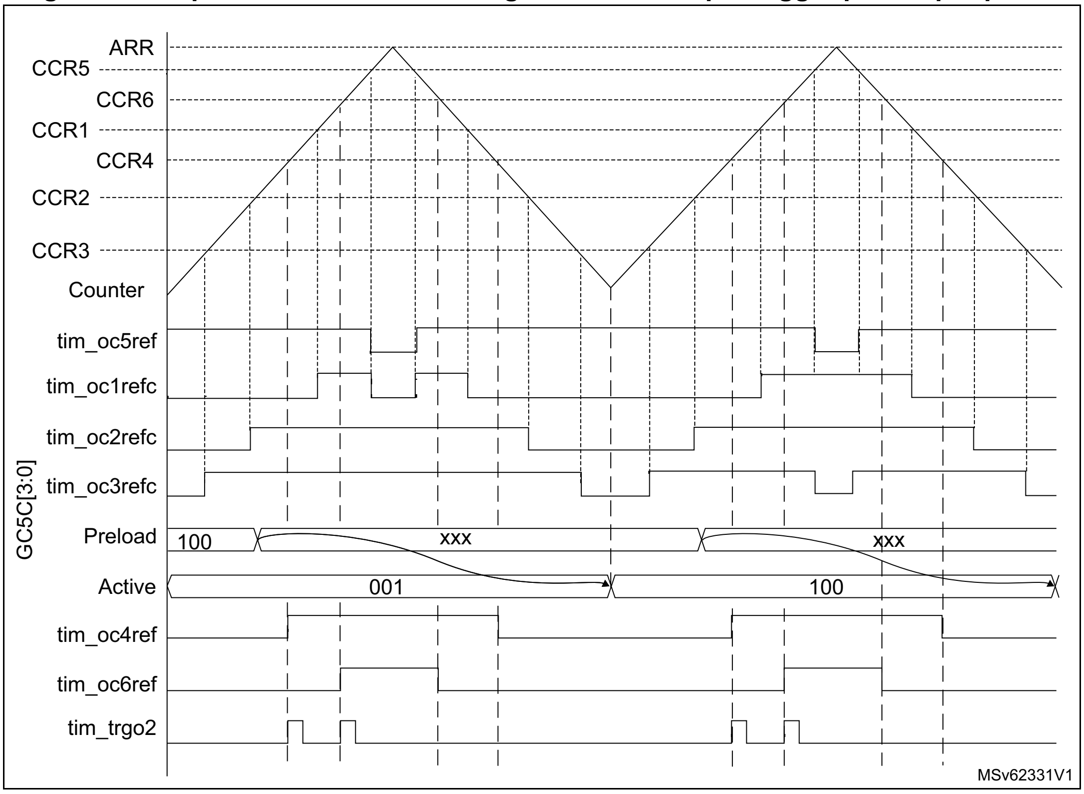
29.3.16 Combined 3-phase PWM mode

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 197. 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 29.3.31: ADC triggers* for more details.

29.3.17 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (such as intrinsic delays of level-shifters, or 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 274: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

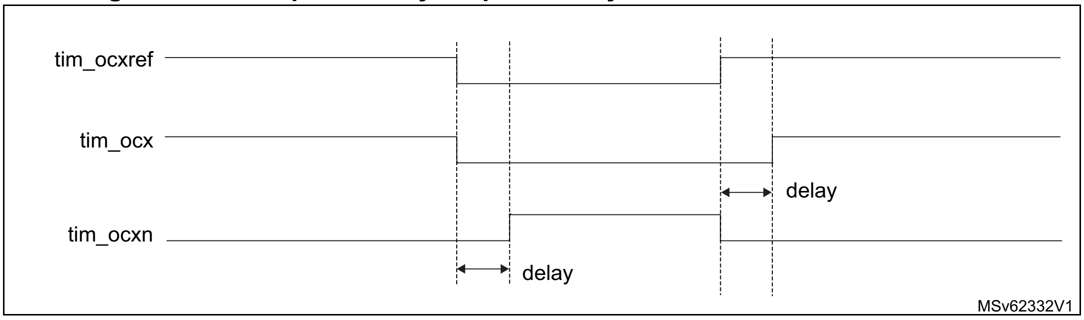
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform tim_ocxref, it generates two outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- 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 considering CCxP = 0, CCxNP = 0, MOE = 1, CCxE = 1 and CCxNE = 1 in these examples.

Figure 198. 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 199*.

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 199. Asymmetrical deadtime

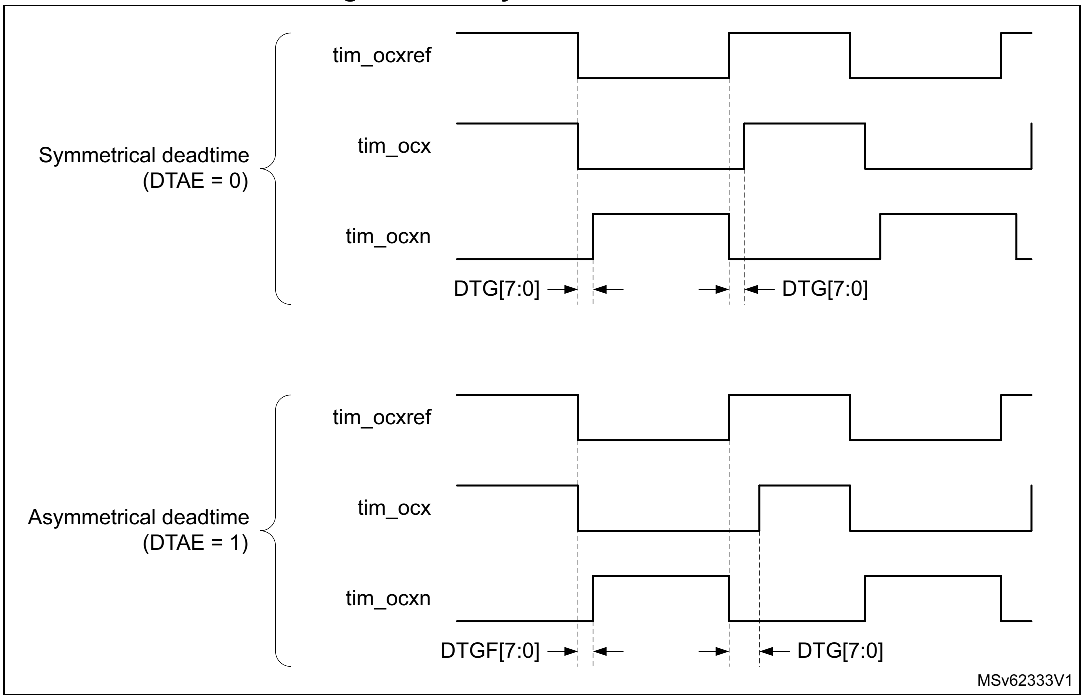


Figure 200. Dead-time waveforms with delay greater than the negative pulse

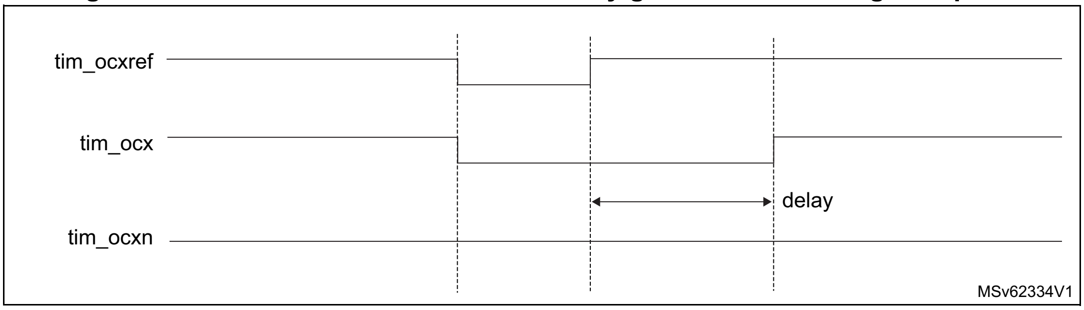
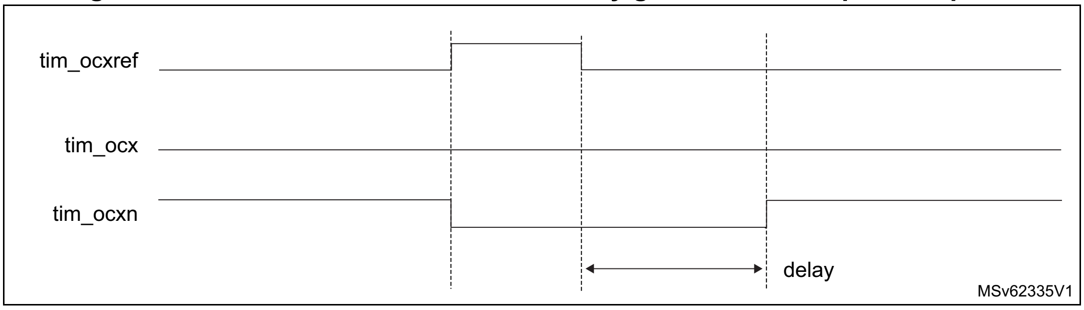


Figure 201. Dead-time waveforms with delay greater than the positive pulse



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 29.6.20: TIM1 break and dead-time register (TIM1_BDTR)* for delay calculation.

Redirecting tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be redirected 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.

29.3.18 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 274: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature* 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 one APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait one 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 29.3.2: TIM1 pins and internal signals* for product specific implementation)
  - coming from a system break request (refer to *Section 29.3.2: TIM1 pins and internal signals* for product specific implementation)

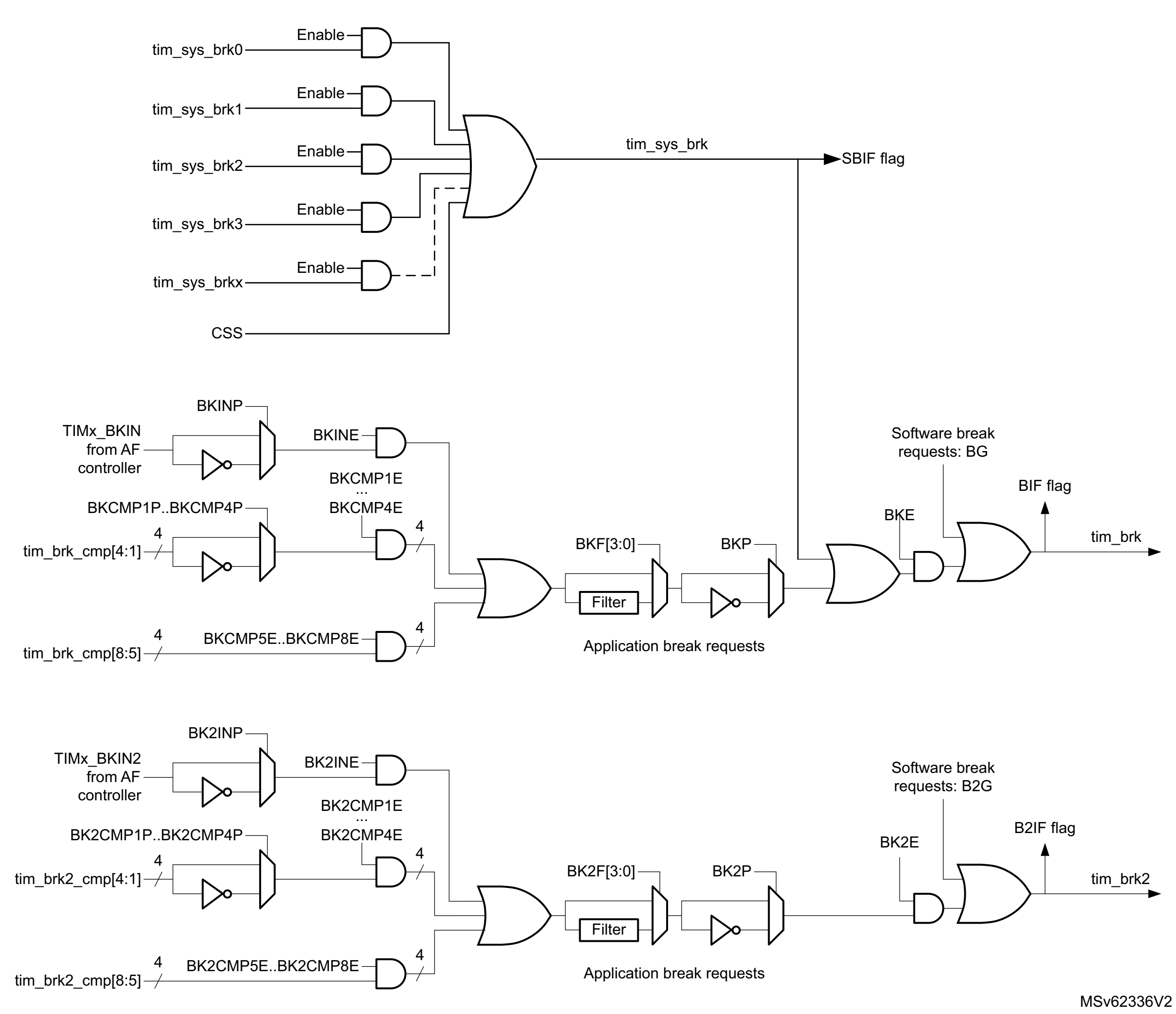
The sources for break2 (tim_brk2) are:

- 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 29.3.2: TIM1 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 202* below.

Figure 202. 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.

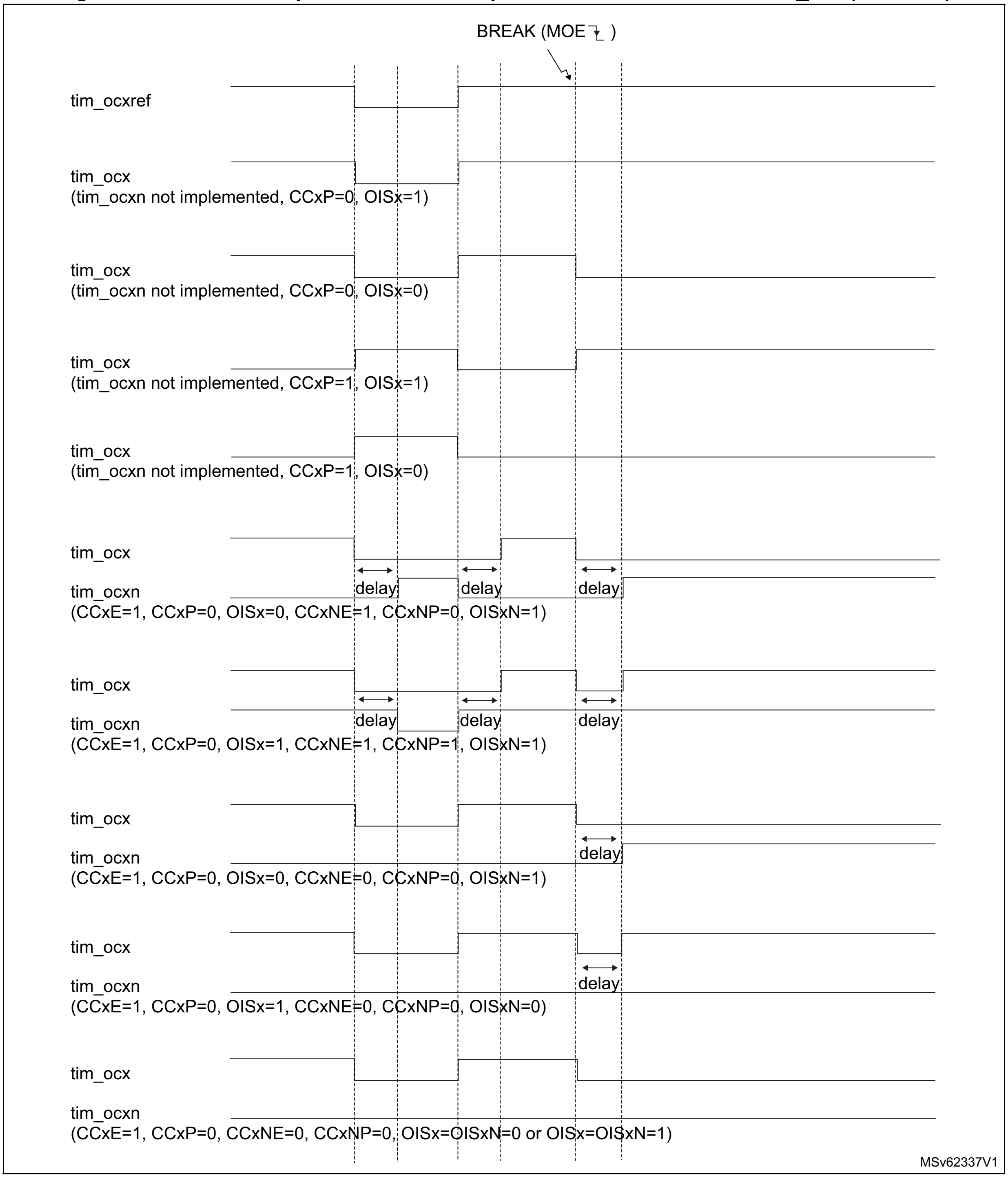
Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value. If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.

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 three levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to *Section 29.6.20*. The LOCK bits can be written only once after an MCU reset.

Figure 203 shows an example of behavior of the outputs in response to a break.

Figure 203. 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 267*.

Note: tim_brk2 must only be used with OSSR = OSSI = 1.

Table 267. 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 204 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 204. PWM output state following tim_brk and tim_brk2 assertion (OSSI = 1)

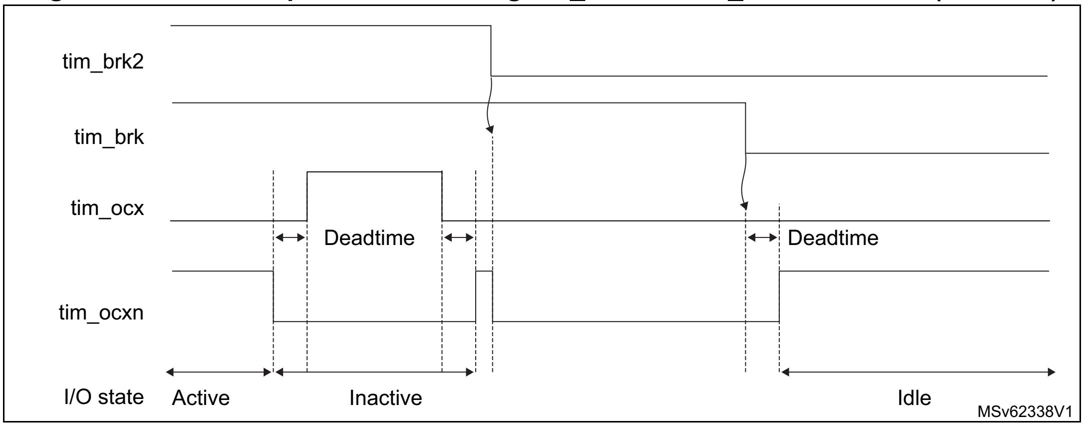
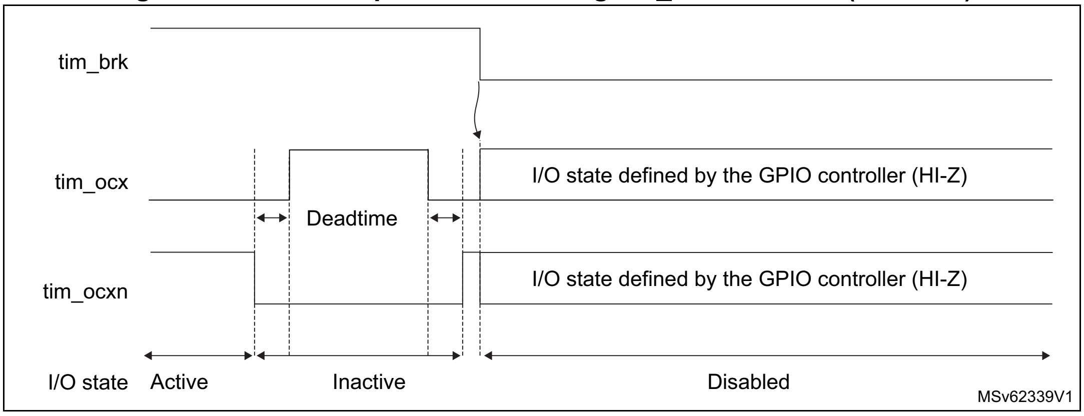


Figure 205. PWM output state following tim_brk assertion (OSSI = 0)



29.3.19 Bidirectional break inputs

The TIM1 features bidirectional break I/Os, as represented on *Figure 206*.

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 (for example 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 restarted 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 268*).

Table 268. 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 rearming break circuitry

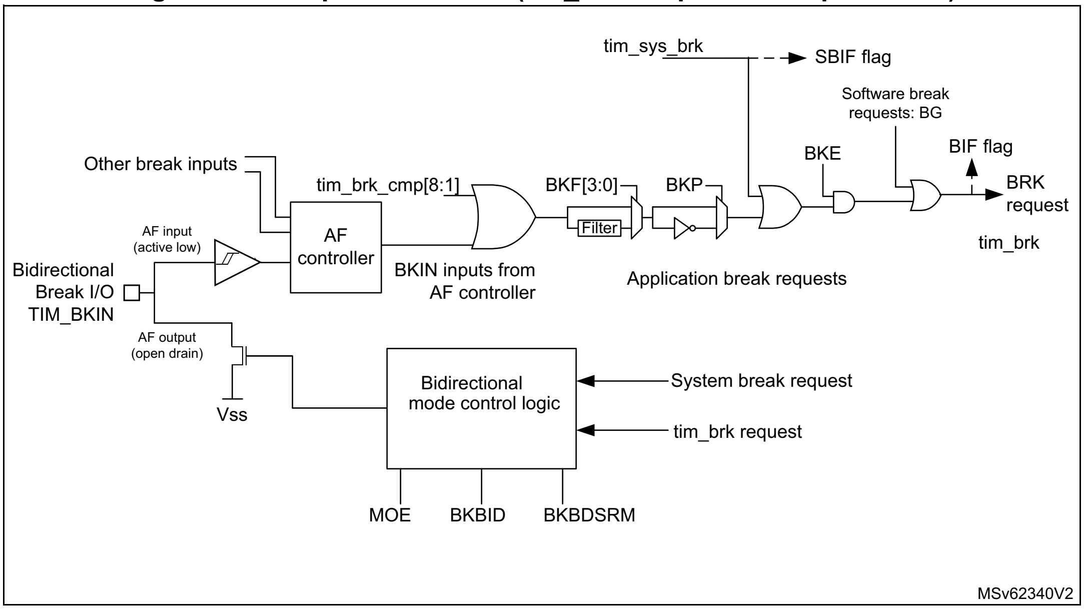
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

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 rearming).
- 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 206. Output redirection (tim_brk2 request not represented)

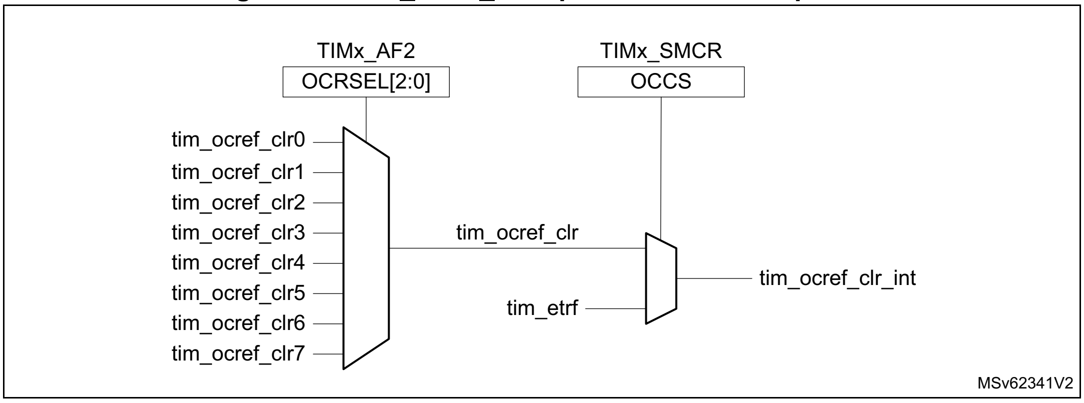


29.3.20 Clearing the tim_ocxref signal on an external event

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 transition to the active state, on the following PWM cycle. 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 207* below. Refer to *Section 29.3.2: TIM1 pins and internal signals* for a list of sources available in the product.

Figure 207. tim_ocref_clr input selection multiplexer

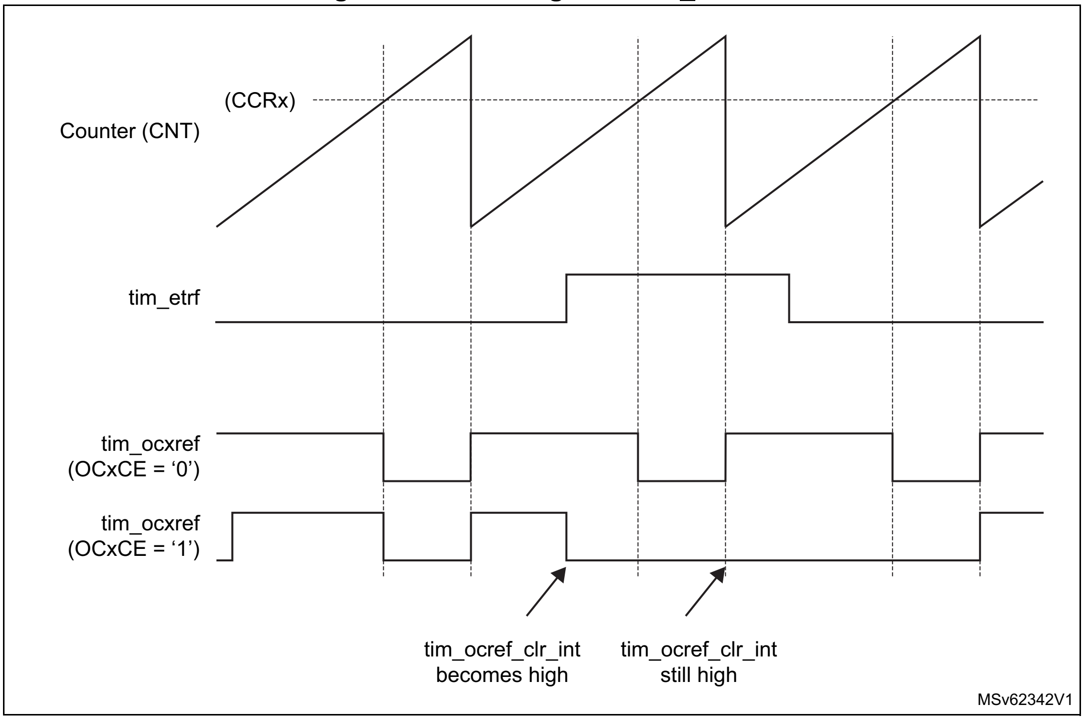


When tim_etrf is chosen, tim_etr_in must be configured as follows:

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 208 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 208. 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.

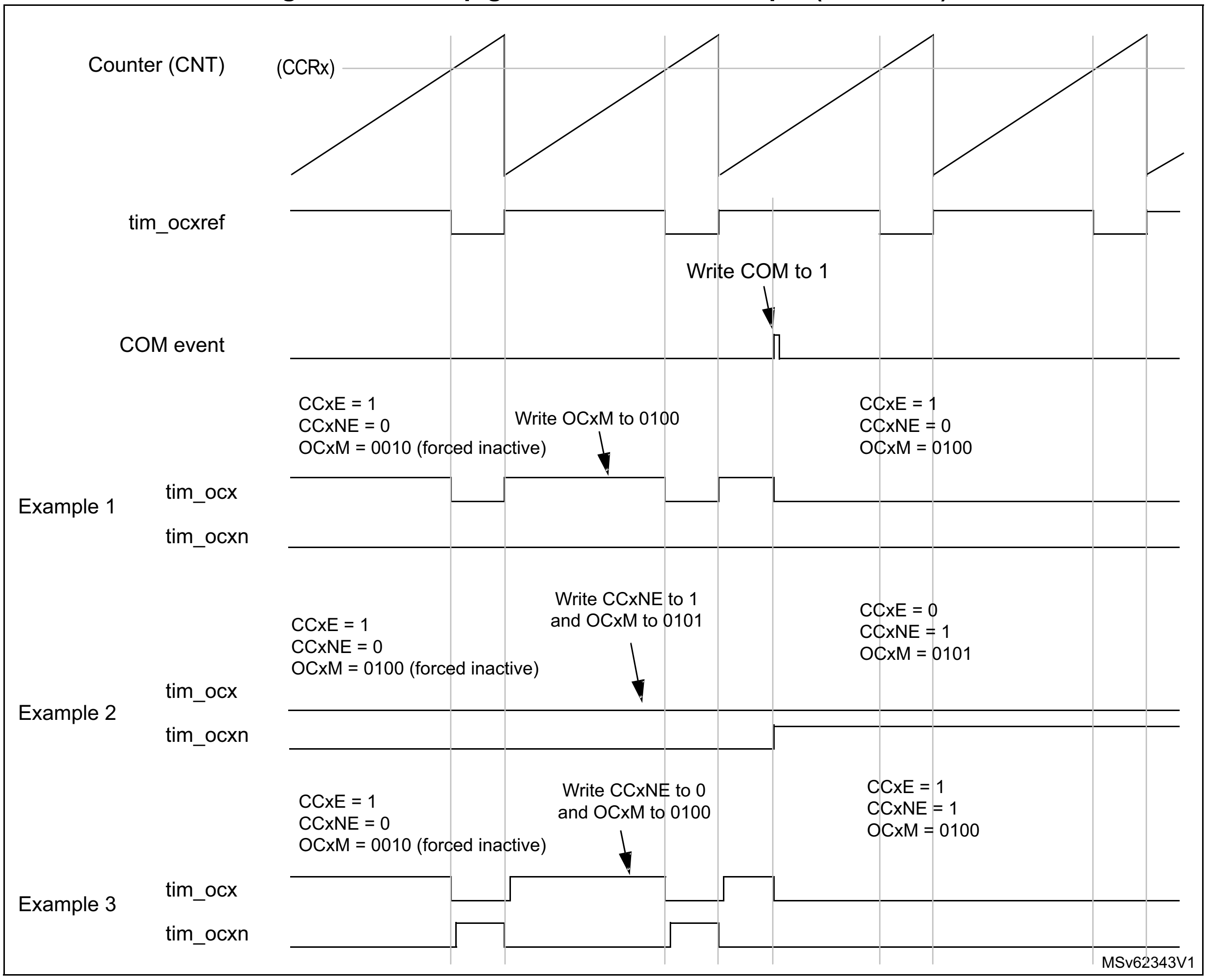
29.3.21 6-step PWM generation

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).

Figure 209 describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in three different examples of programmed configurations.

Figure 209. 6-step generation, COM example (OSSR = 1)



29.3.22 One-pulse mode

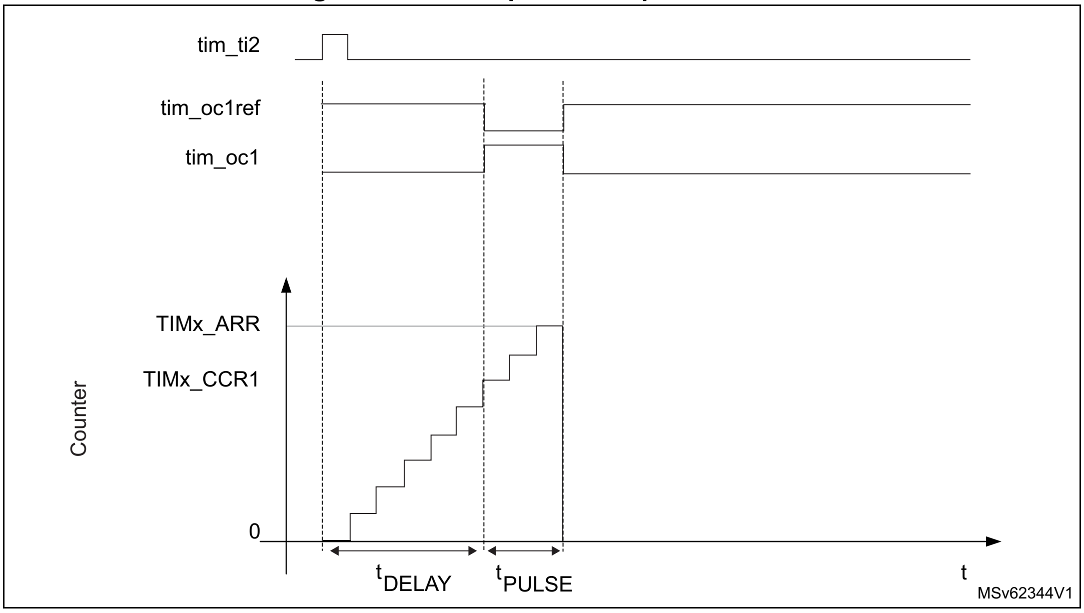
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

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 210. Example of one pulse mode.



In the following example, the user wants 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.

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 autoreload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Suppose the user wants 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. This is achieved by enabling PWM mode 2 (OC1M = 111 in TIMx_CCMR1). 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 autoreload 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 this example, the DIR and CMS bits in the TIMx_CR1 register must be low.

Since only one 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 autoreload 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 that can be achieved.

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.

29.3.23 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 nonretriggerable one-pulse mode described in *Section 29.3.22*:

- 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.

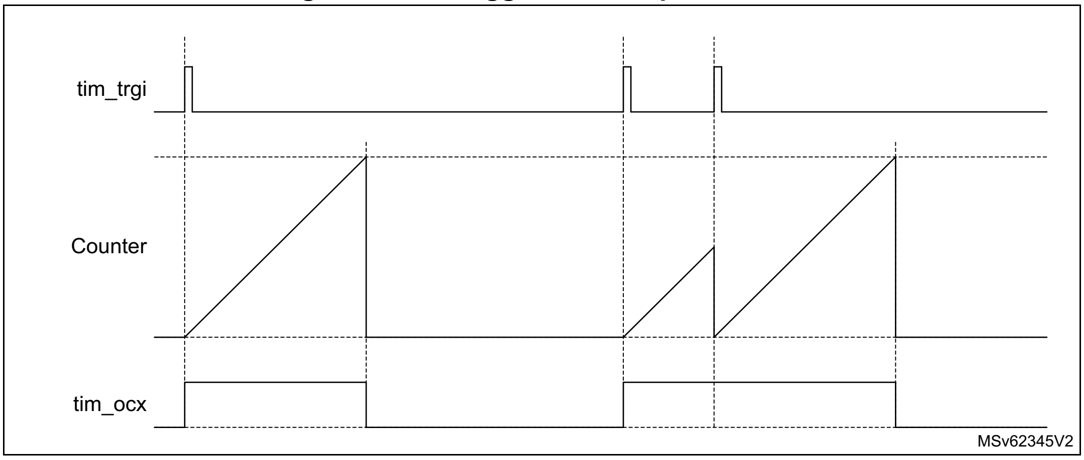
The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for retriggerable OPM mode 1 or 2.

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] bitfields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three 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 211. Retriggerable one-pulse mode



29.3.24 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 *Figure 212*.

Figure 212. Pulse generator circuitry

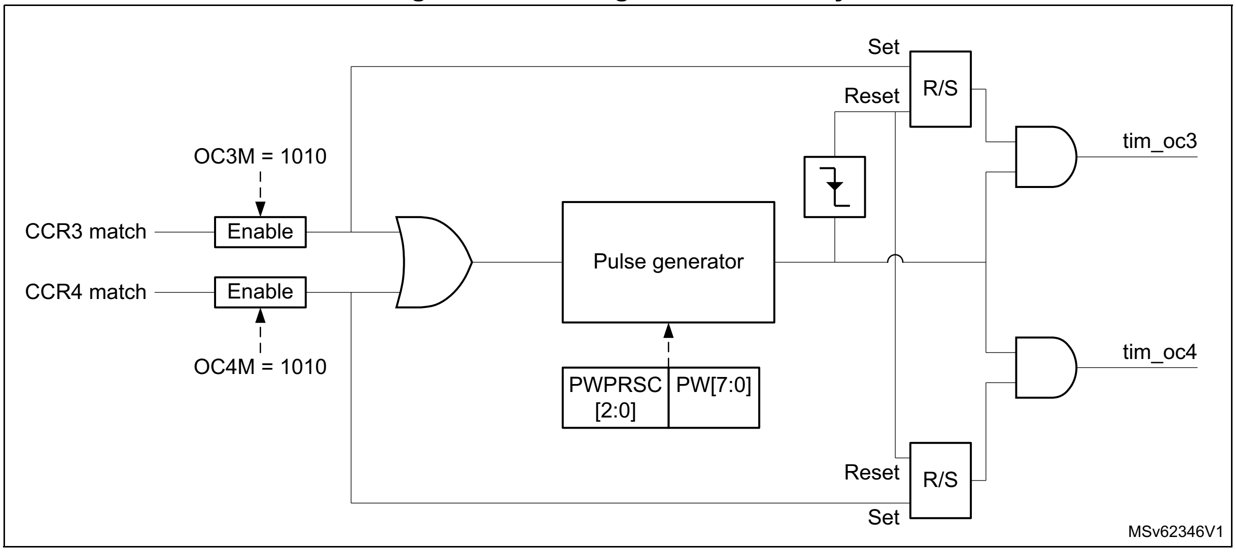
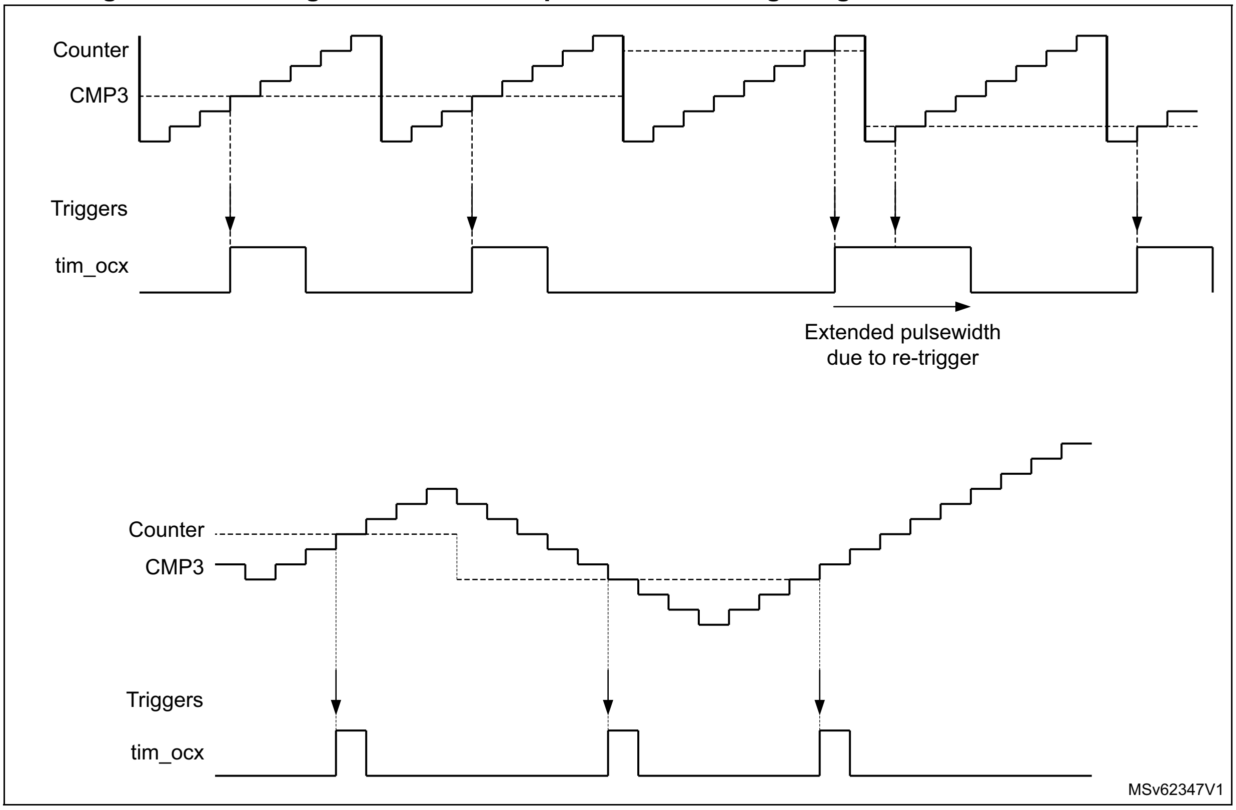


Figure 213 shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 213. 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] bitfields 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}

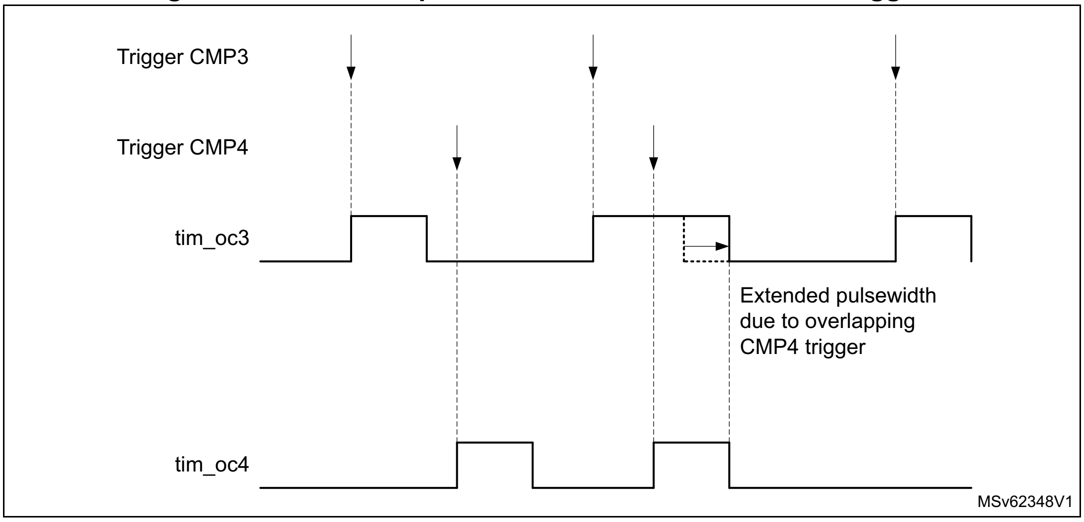
where $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim\_ker\_ck}$

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 first arriving trigger is extended (because of the retrigger), while the pulse width of the last arriving trigger is correct (as shown on Figure 214).

Figure 214. Extended pulsewidth in case of concurrent triggers



29.3.25 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 269*. 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 autoreload 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 269. 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.

Figure 215 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 the configuration is the following:

- CC1S = 01 (TIMx_CCMR1 register, tim_ti1fp1 mapped on tim_ti1).
- CC2S = 01 (TIMx_CCMR1 register, tim_ti2fp2 mapped on tim_ti2).
- CC1P = 0 and CC1NP = 0 (TIMx_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1 = tim_ti1).
- CC2P = 0 and CC2NP = 0 (TIMx_CCER register, tim_ti1fp2 noninverted, tim_ti1fp2= tim_ti2).
- SMS = 0011 (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIMx_CR1 register, Counter enabled).

Figure 215. Example of counter operation in encoder interface mode.

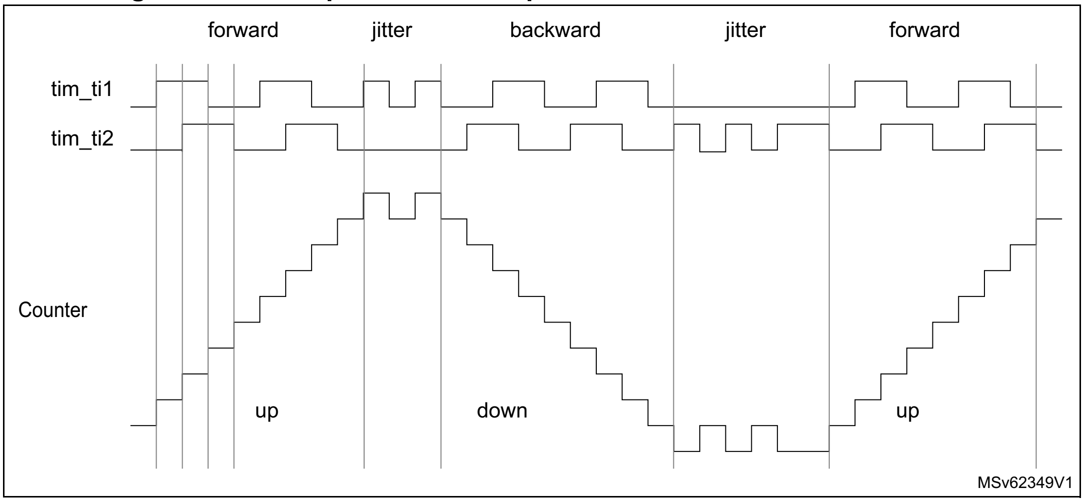


Figure 216 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 216. Example of encoder interface mode with tim_ti1fp1 polarity inverted.

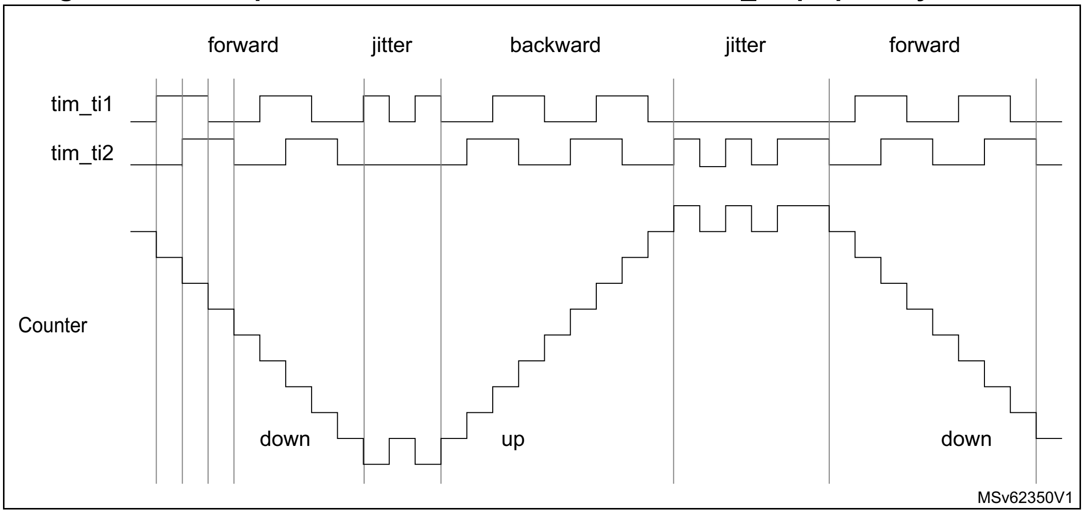
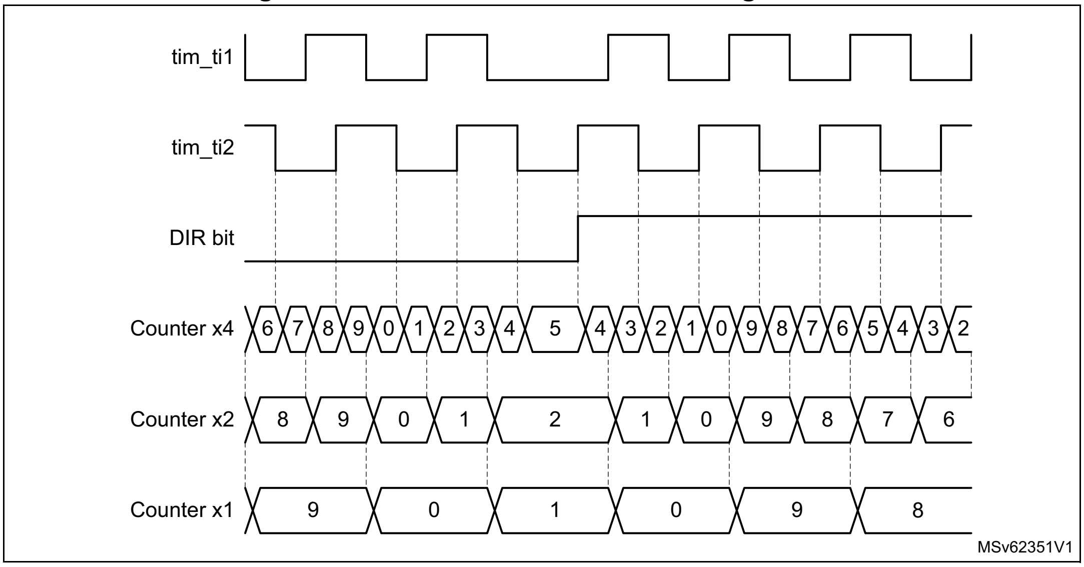


Figure 217 shows the timer counter value during a speed reversal, for various counting modes.

Figure 217. 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 for other types of encoders.

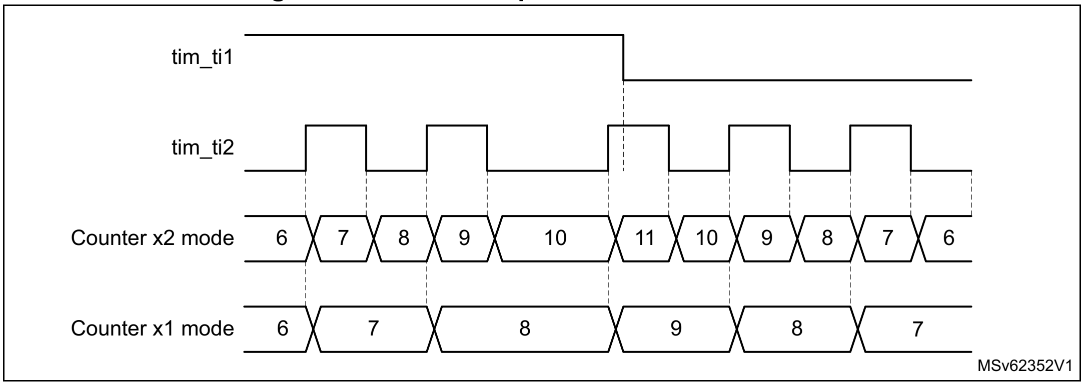
In the clock plus direction mode shown on *Figure 218*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

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 218. Direction plus clock encoder mode



Directional clock encoder mode

In the directional clock mode on *Figure 219*, 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 219*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 220*).
- 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 219*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 220*).

Figure 219. Directional clock encoder mode (CC1P = CC2P = 0)

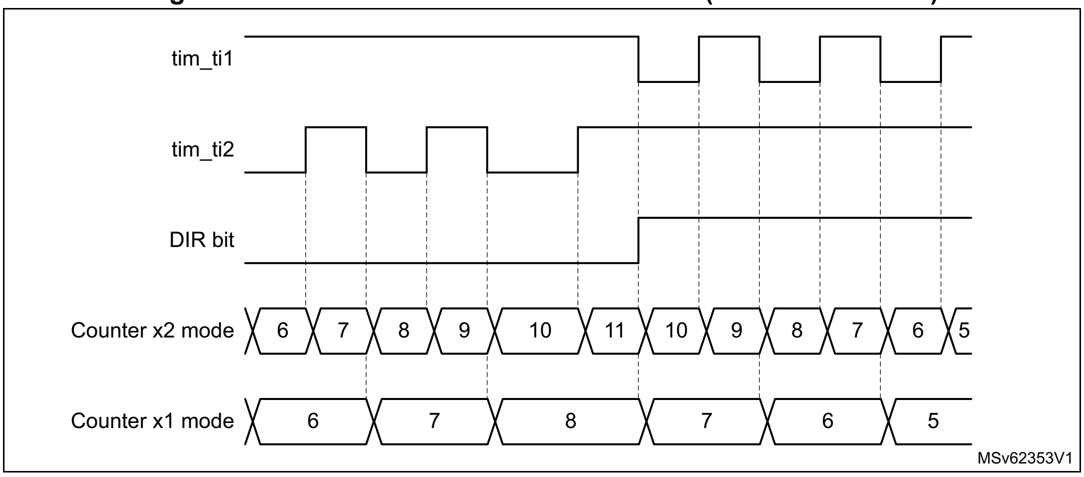


Figure 220. Directional clock encoder mode (CC1P = CC2P = 1)

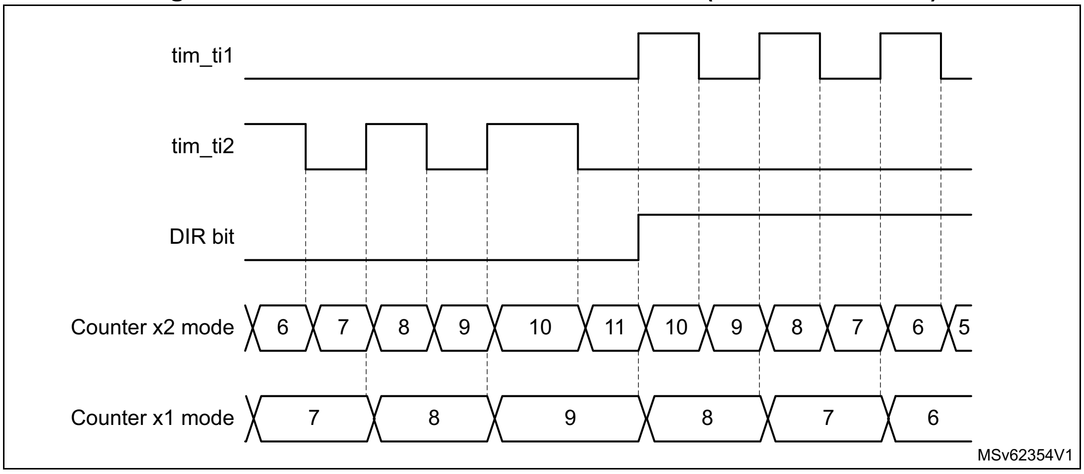


Table 270 here-below details how the directional clock mode operates, for any input transition.

Table 270. 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

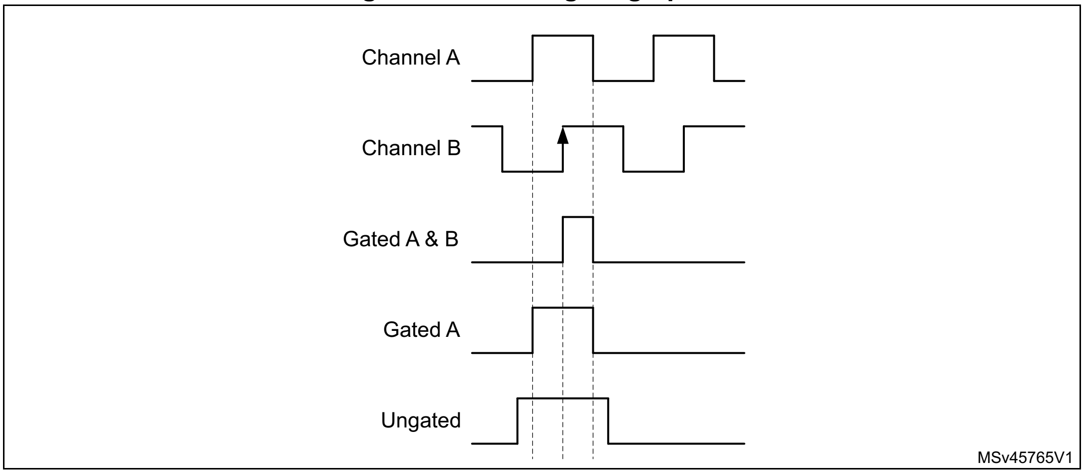
The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

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.

Available encoders are proposed with several options for index pulse conditioning, as per *Figure 221*:

- 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

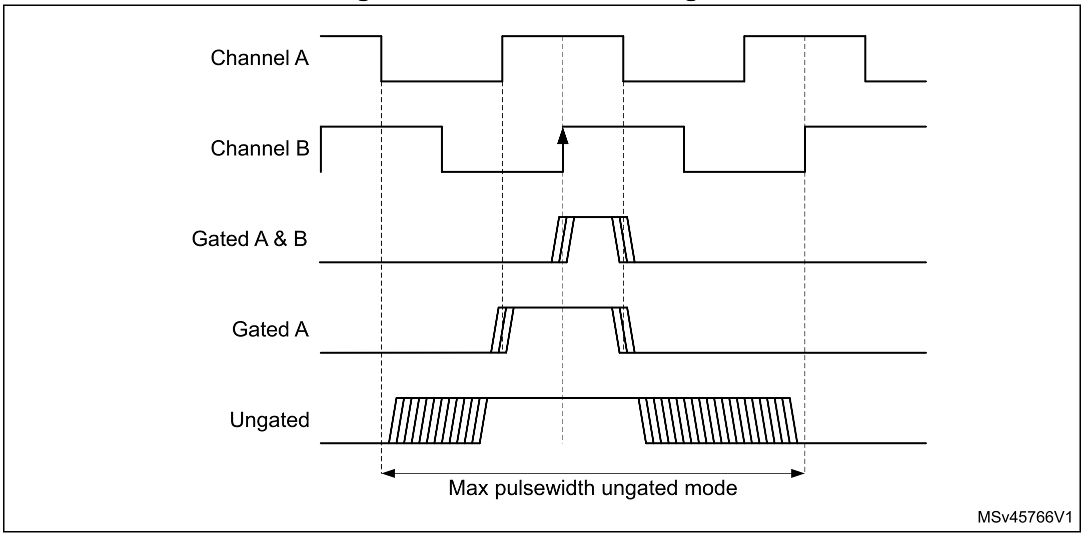
Figure 221. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as show on *Figure 222*.

In ungated mode, the signal must be strictly below two encoder periods. If the pulsewidth is greater or equal to two encoder period, the counter is reset multiple times.

Figure 222. Jittered Index signals



The timer supports the three gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (for example 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. *Figure 223* shows at which position is the index generated, for a simplistic example (an encoder providing four edges par mechanical rotation).

Figure 223. Index generation for IPOS[1:0] = 11

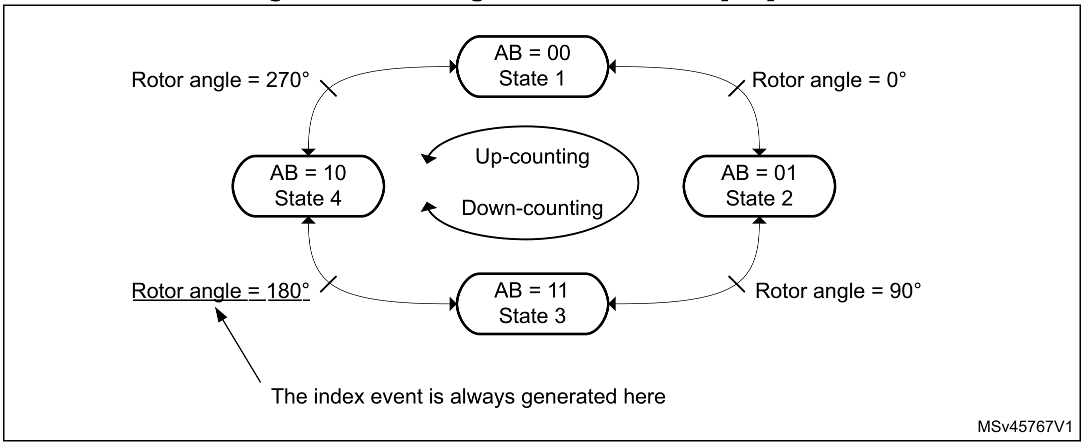


Figure 224 presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- 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 224. Counter reading with index gated on channel A (IPOS[1:0] = 11)

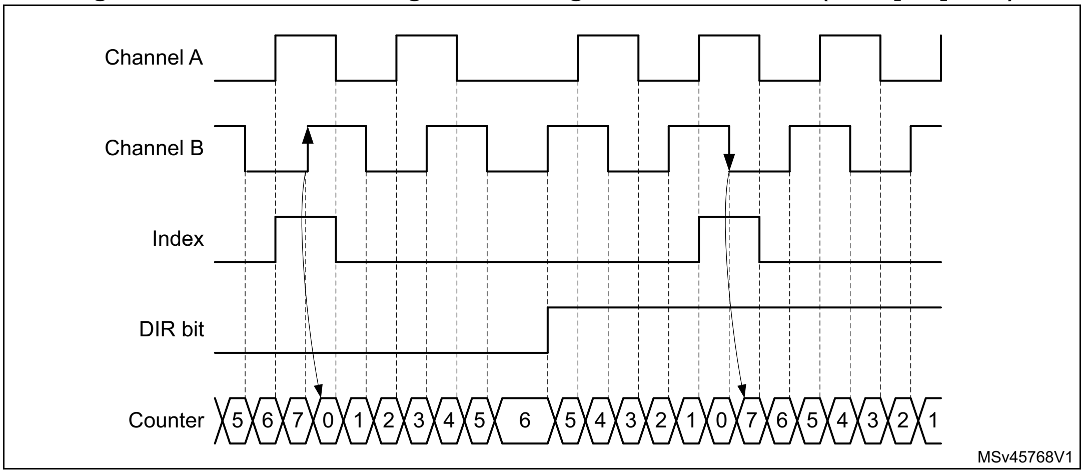


Figure 225. presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 225. Counter reading with index ungated (IPOS[1:0] = 00)

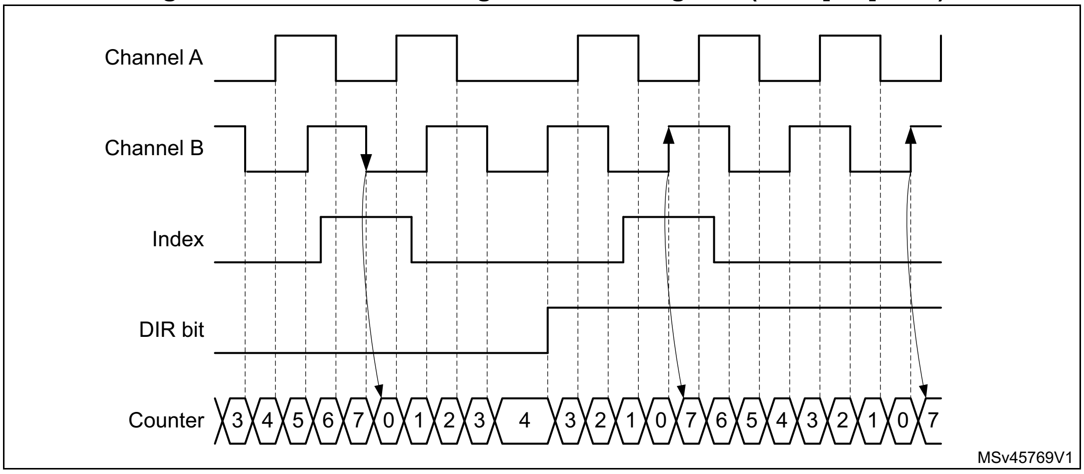


Figure 226. shows how the ‘gated on A & B’ mode is handled, for various pulse alignment scenario. The arrows are indicating on which transition is the index event generated.

Figure 226. Counter reading with index gated on channel A and B

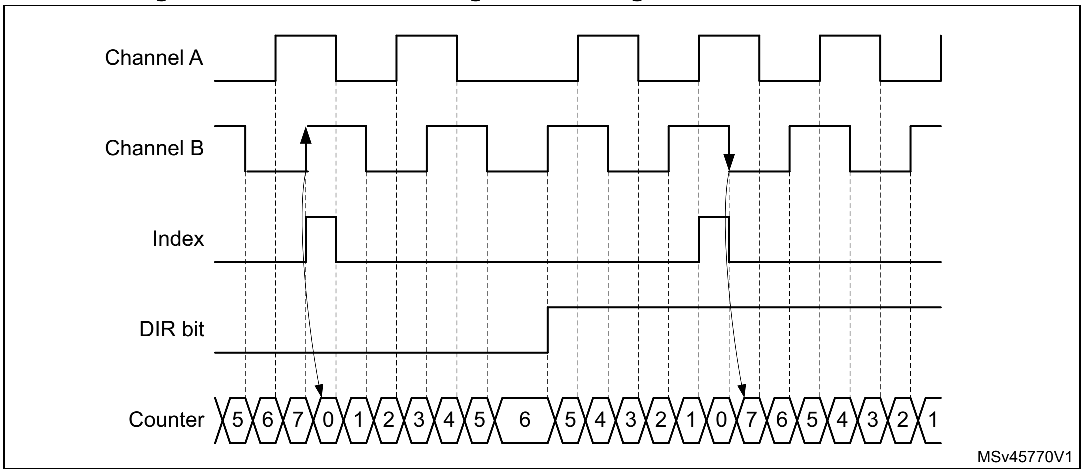


Figure 227 and *Figure 228* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 227. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

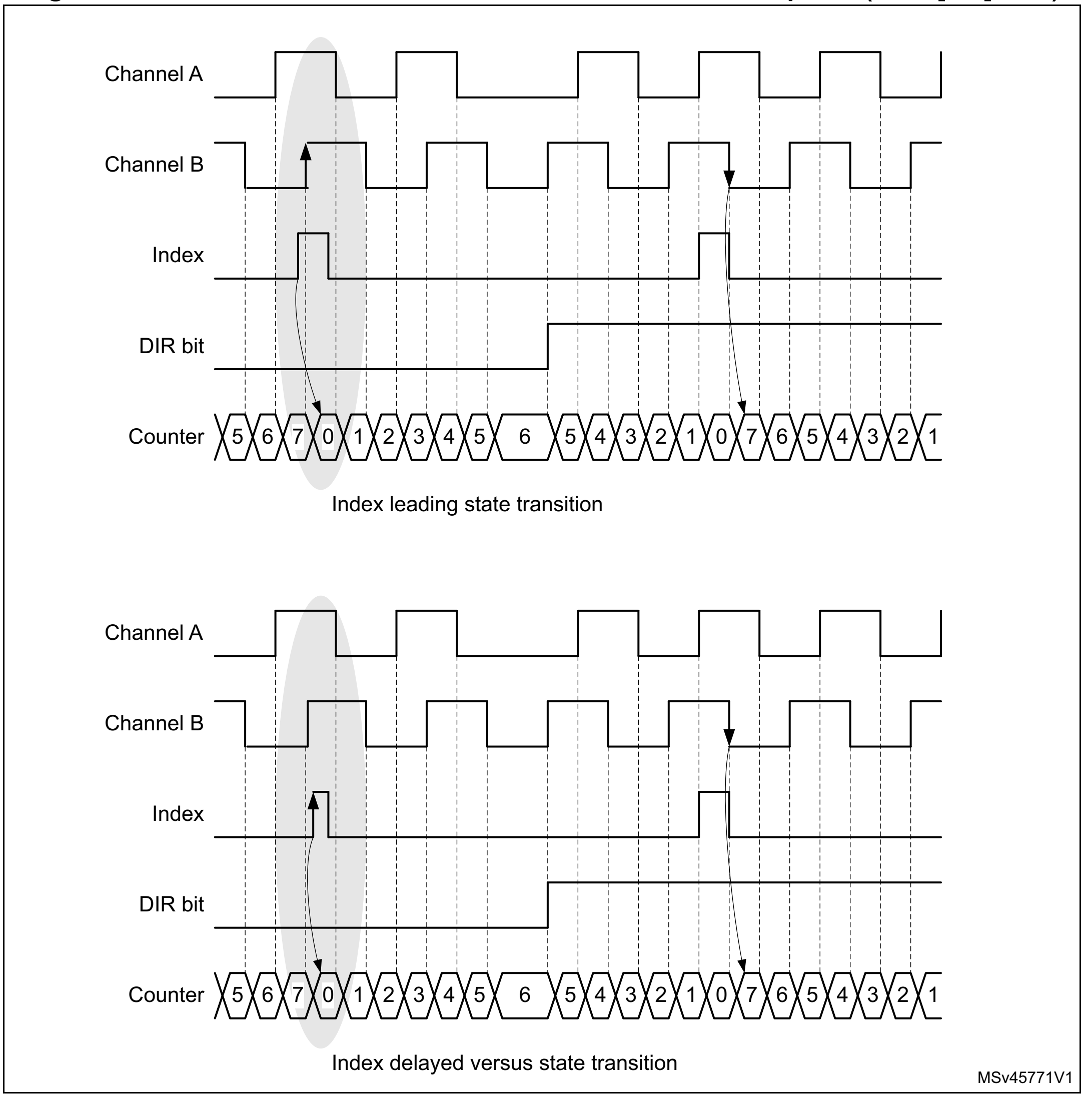


Figure 228. Counter reset Narrow index pulse (closer view, ARR = 0x07)

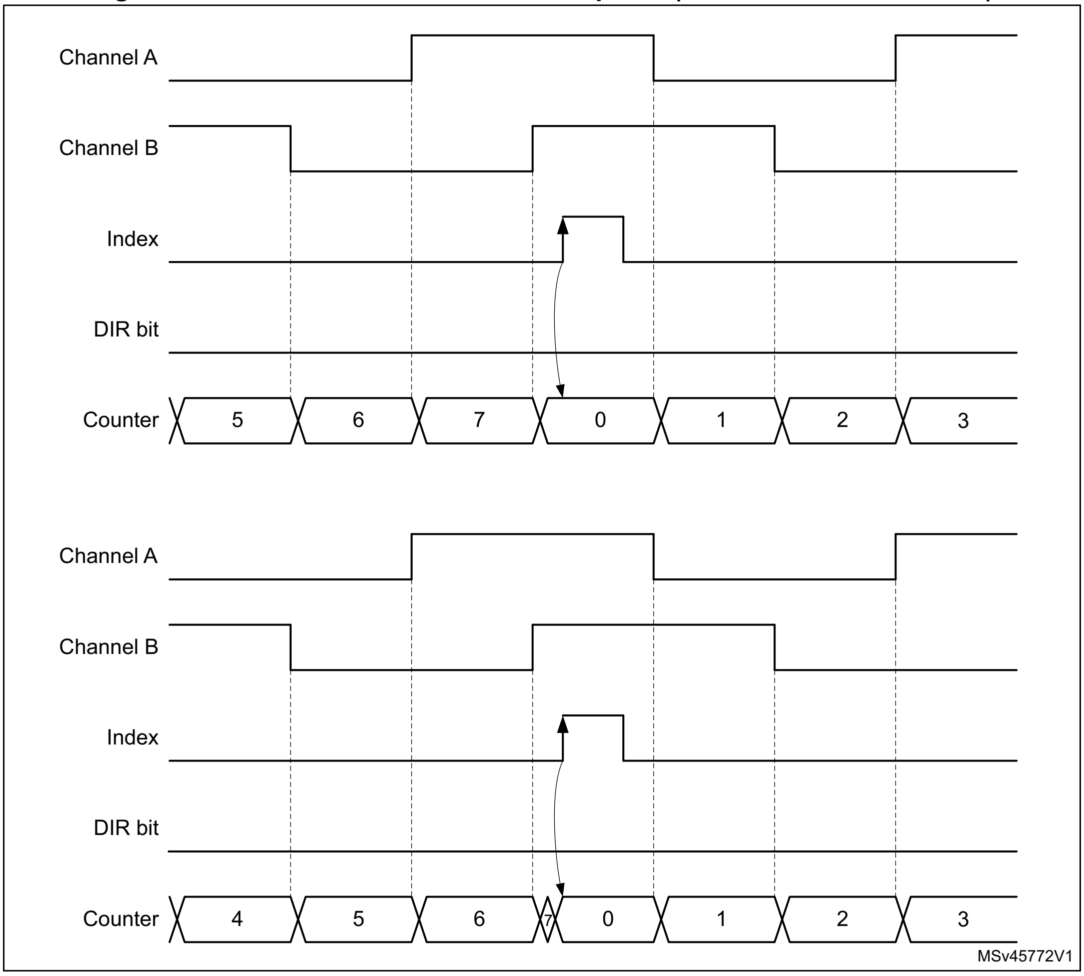
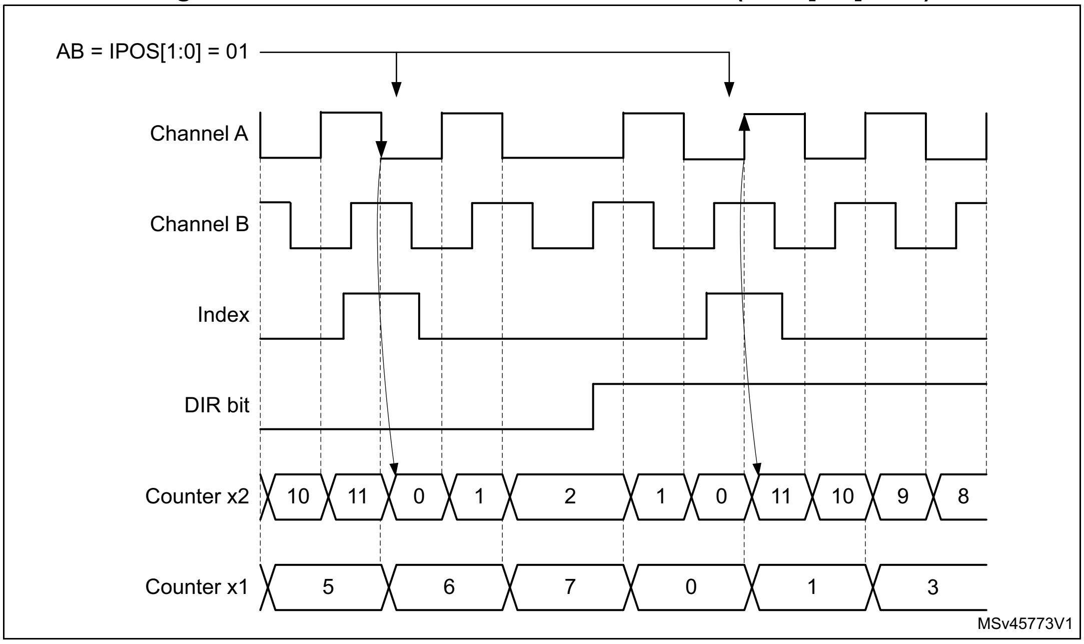


Figure 229 shows how the index is managed in x1 and x2 modes.

Figure 229. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)

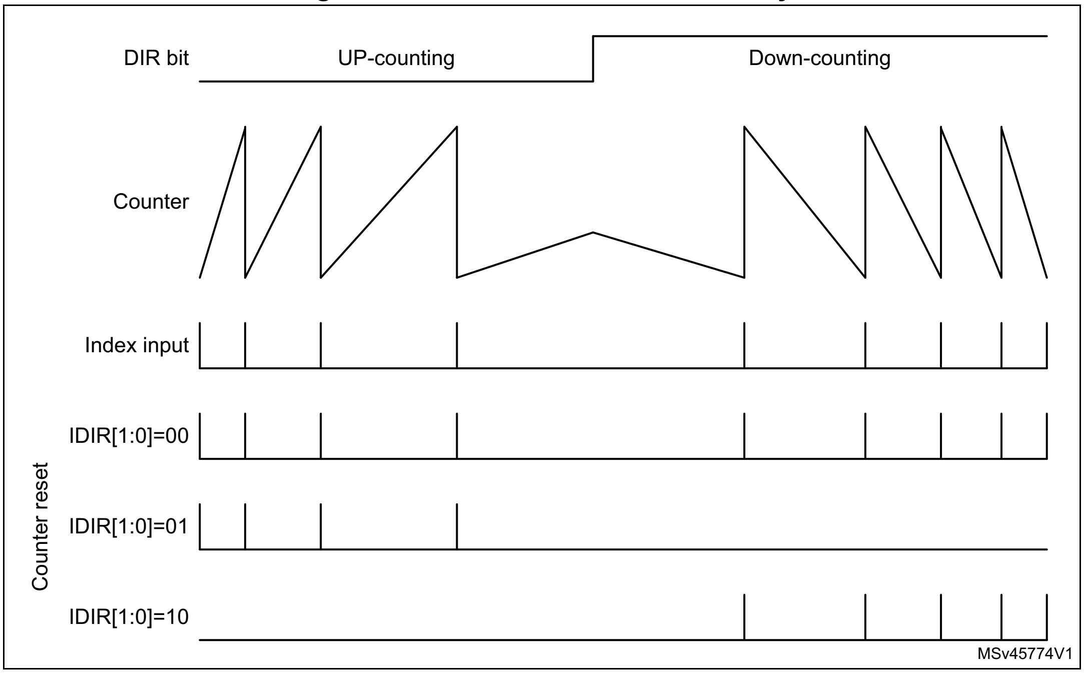


Directional index sensitivity

The IDIR[1:0] bitfield in the TIMx_ECR register allows the index to be active only in a selected counting direction.

Figure 230 shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

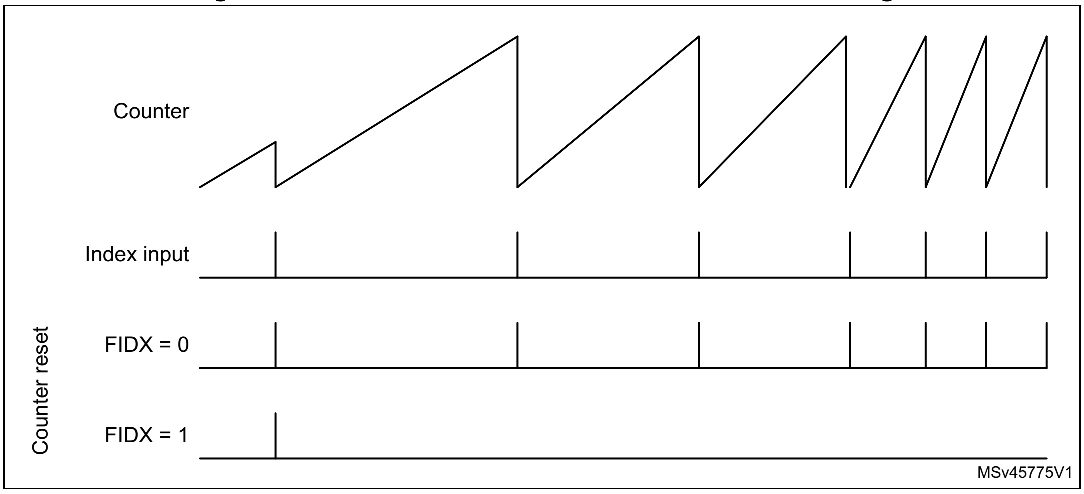
Figure 230. Directional index sensitivity



Special first index event management

The FIDX bit in the TIMx_ECR register allows the index to be taken only once, as shown on *Figure 231*. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be rearmed by writing the FIDX bit to 0 and setting it again to 1.

Figure 231. Counter reset as function of FIDX bit setting



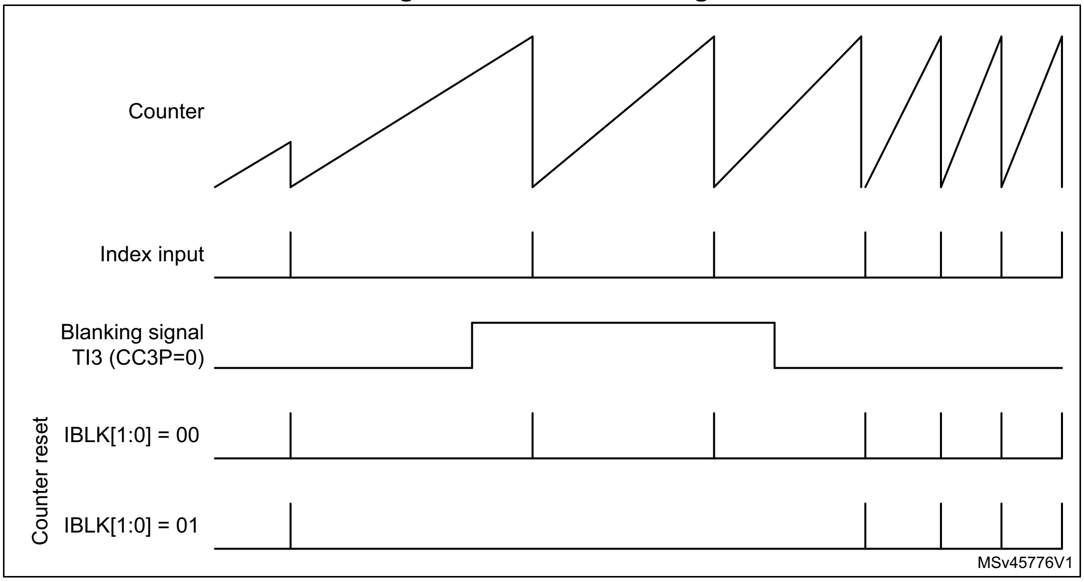
Index blanking

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 232* 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 232. Index blanking



Index management in nonquadrature mode

Figure 233 and *Figure 234* 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 233. Index behavior in clock + direction mode, IPOS[0] = 1

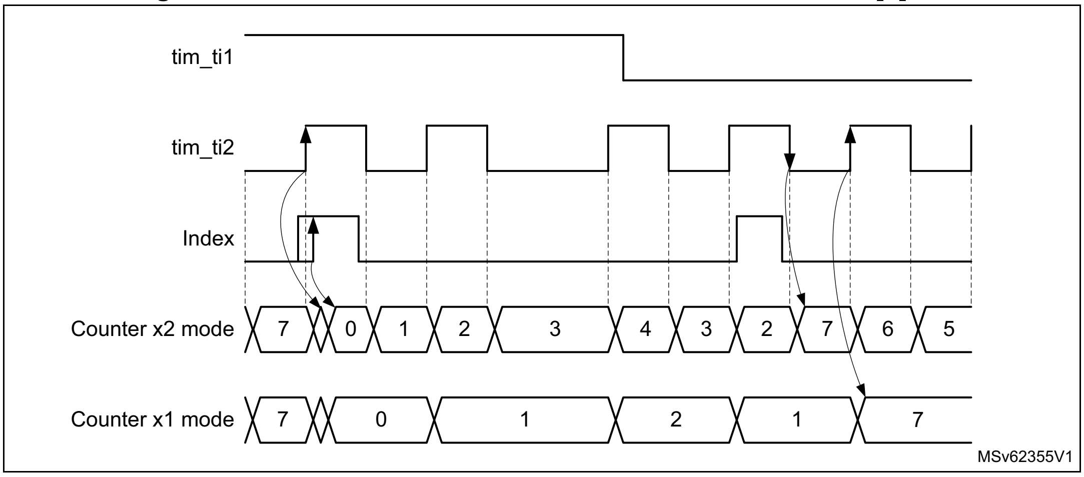
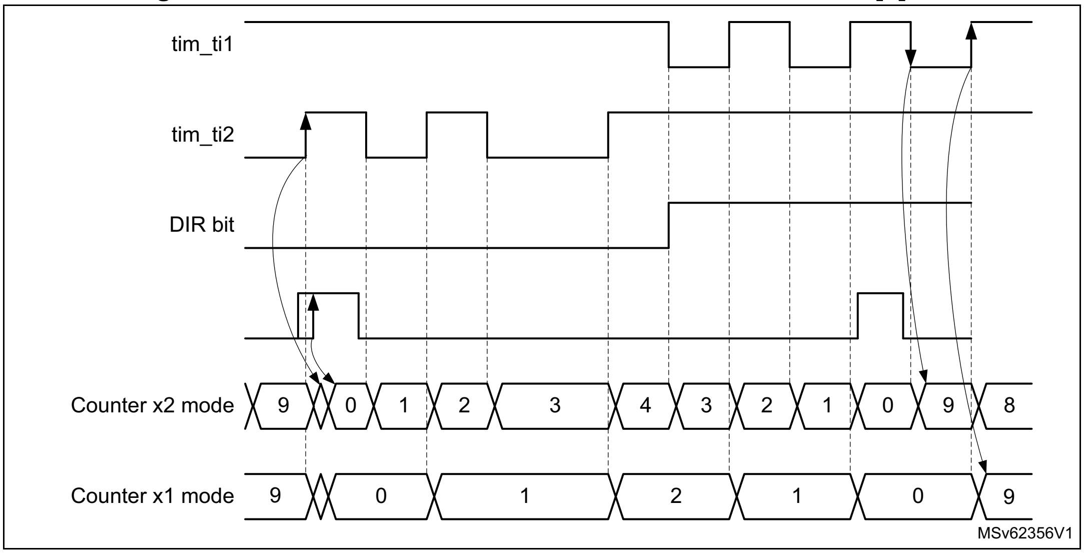


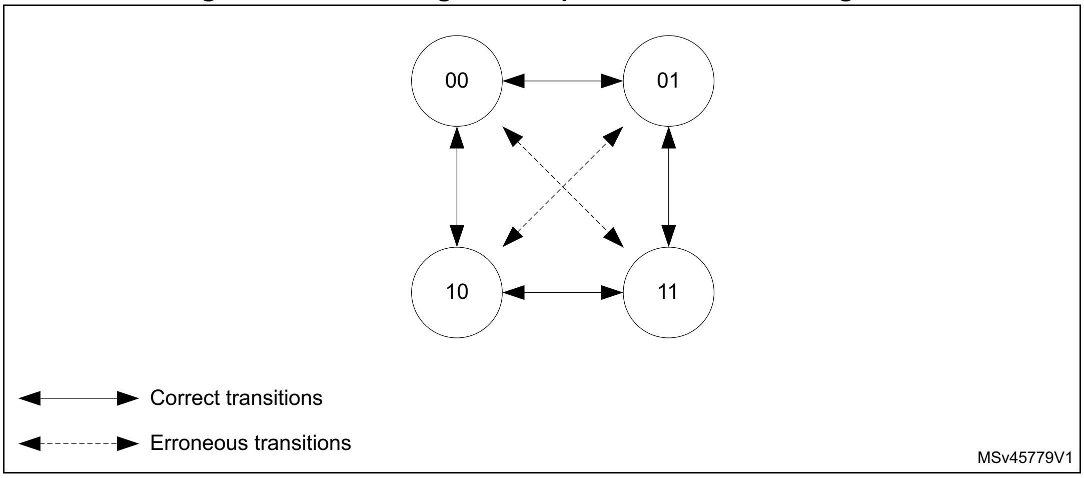
Figure 234. Index behavior in directional clock mode, IPOS[0] = 1



Encoder error management

For encoder configurations where two quadrature signals are available, it is possible to detect transition errors. The reading on the two inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on *Figure 235.*. 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 235. State diagram for quadrature encoded signals



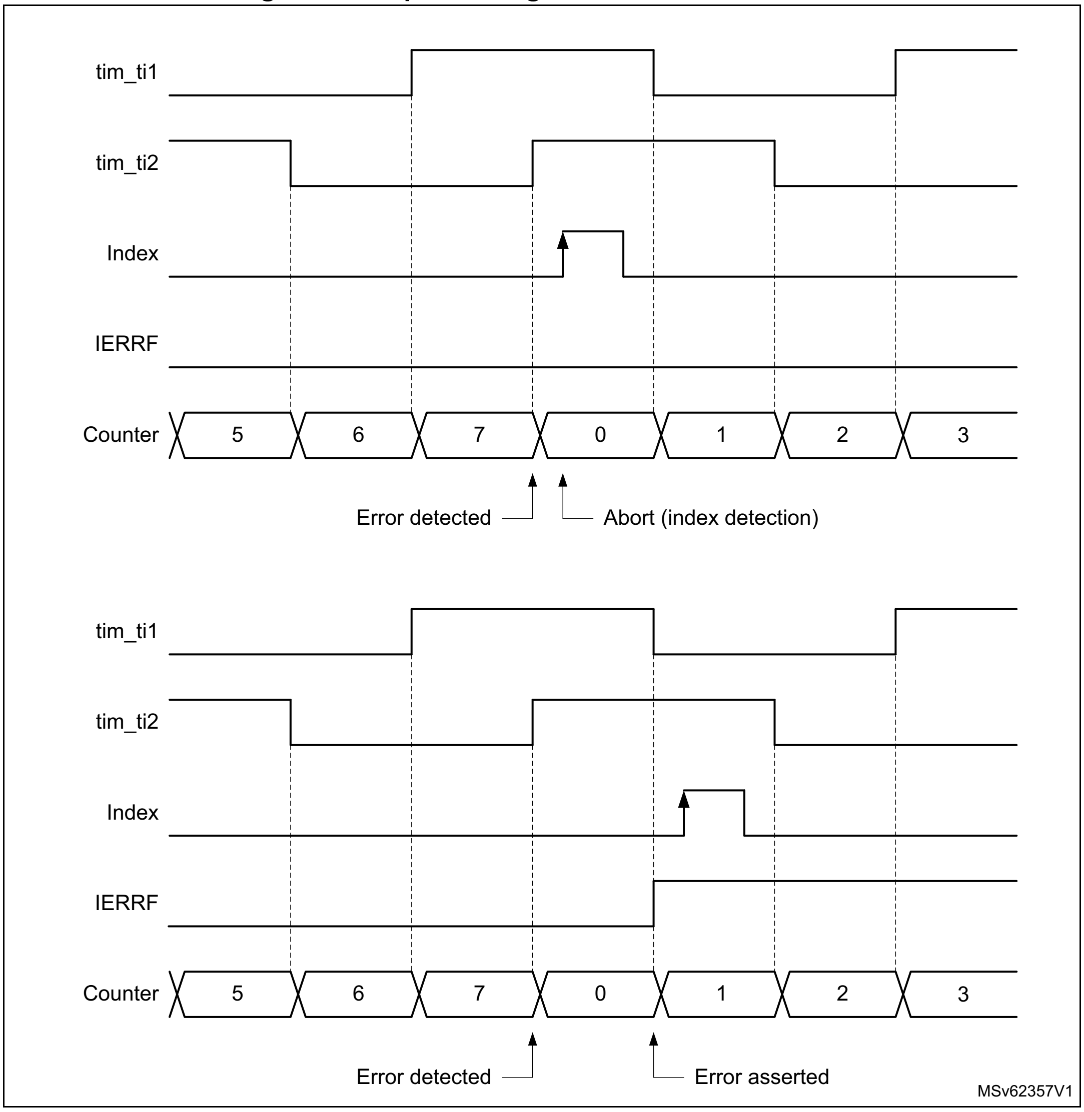
For encoder having an index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The index signal resets the counter every 4xN clock periods.

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.

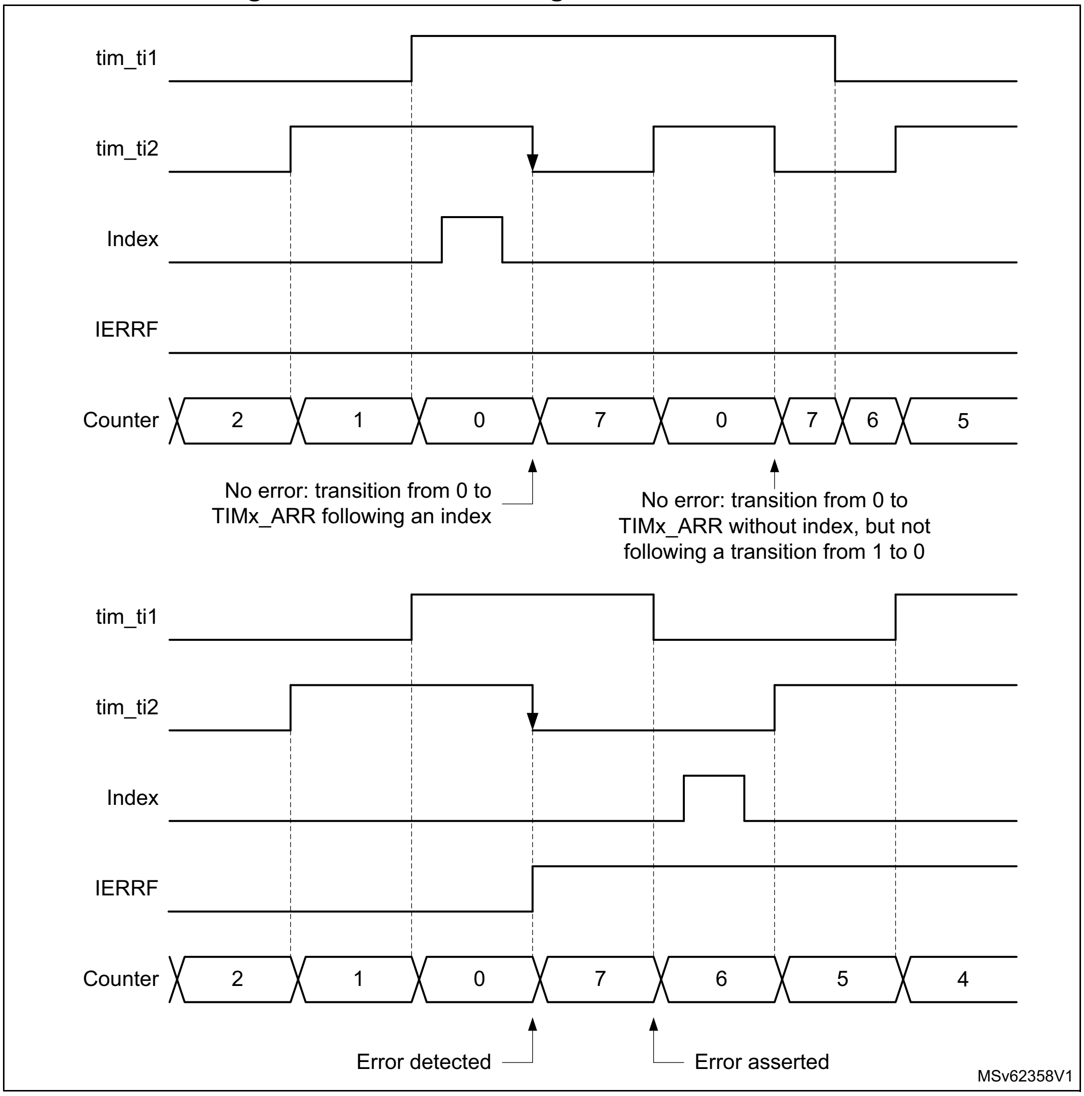
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is cope with narrow index pulses in gated A and B mode, as shown on Figure 236.

Figure 236. 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 237, to avoid any false error detection in case the encoder dithers between TIMx_ARR and 0 immediately after the index detection.

Figure 237. 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.

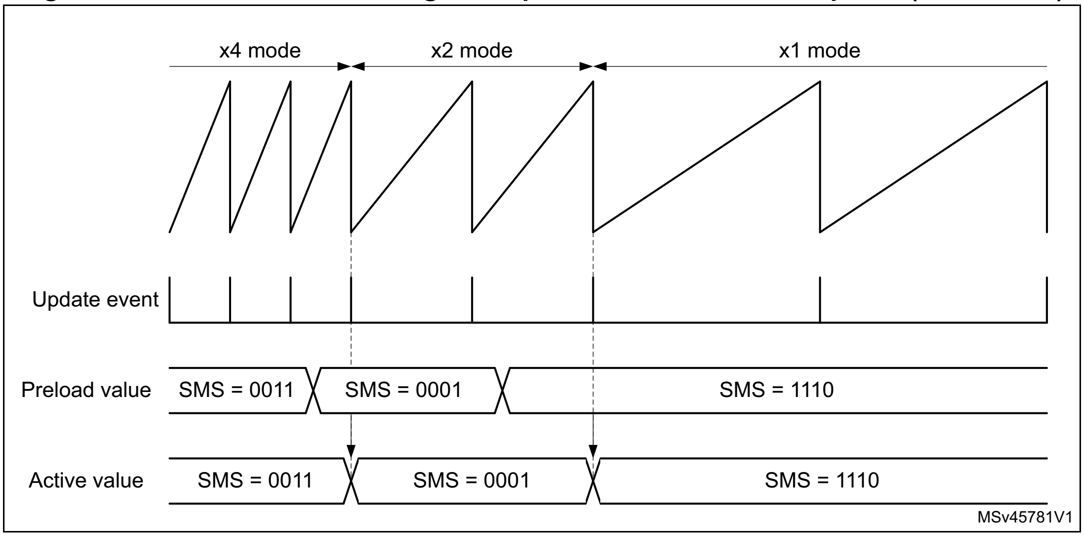
Slave mode selection preload for run-time encoder mode update

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 shown on *Figure 238*.

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.
- SMSPS = 1: the transfer is triggered by the index event.

Figure 238. Encoder mode change with preload transferred on update (SMSPS = 0)



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIMx_CR2 register.It is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

29.3.26 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] bitfield to 1011 in the TIMx_CCMR2 register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

29.3.27 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).

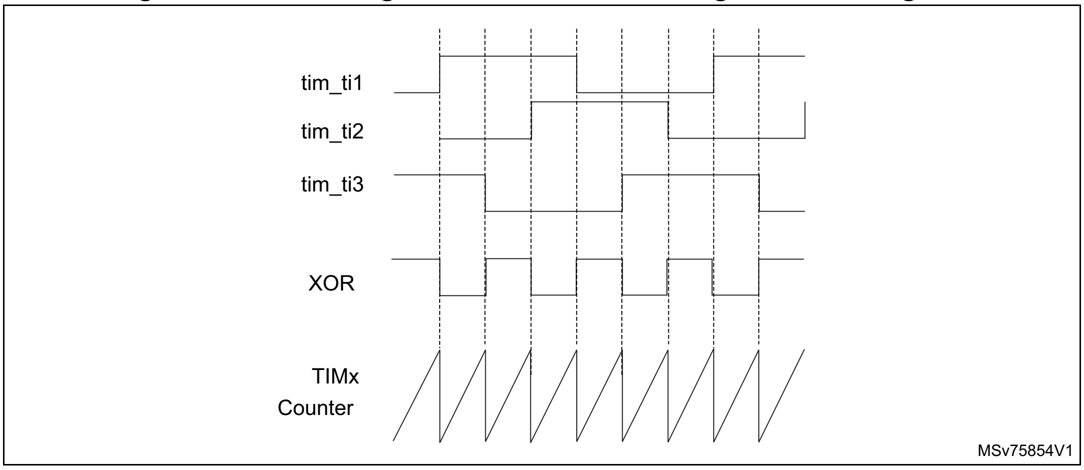
There is no latency between the UIF and UIFCPY flags assertion.

29.3.28 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 239*.

Figure 239. Measuring time interval between edges on three signals



29.3.29 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 240*. The "interfacing timer" captures the three 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 three 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 182*). The captured value, which corresponds to the time elapsed between two 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.

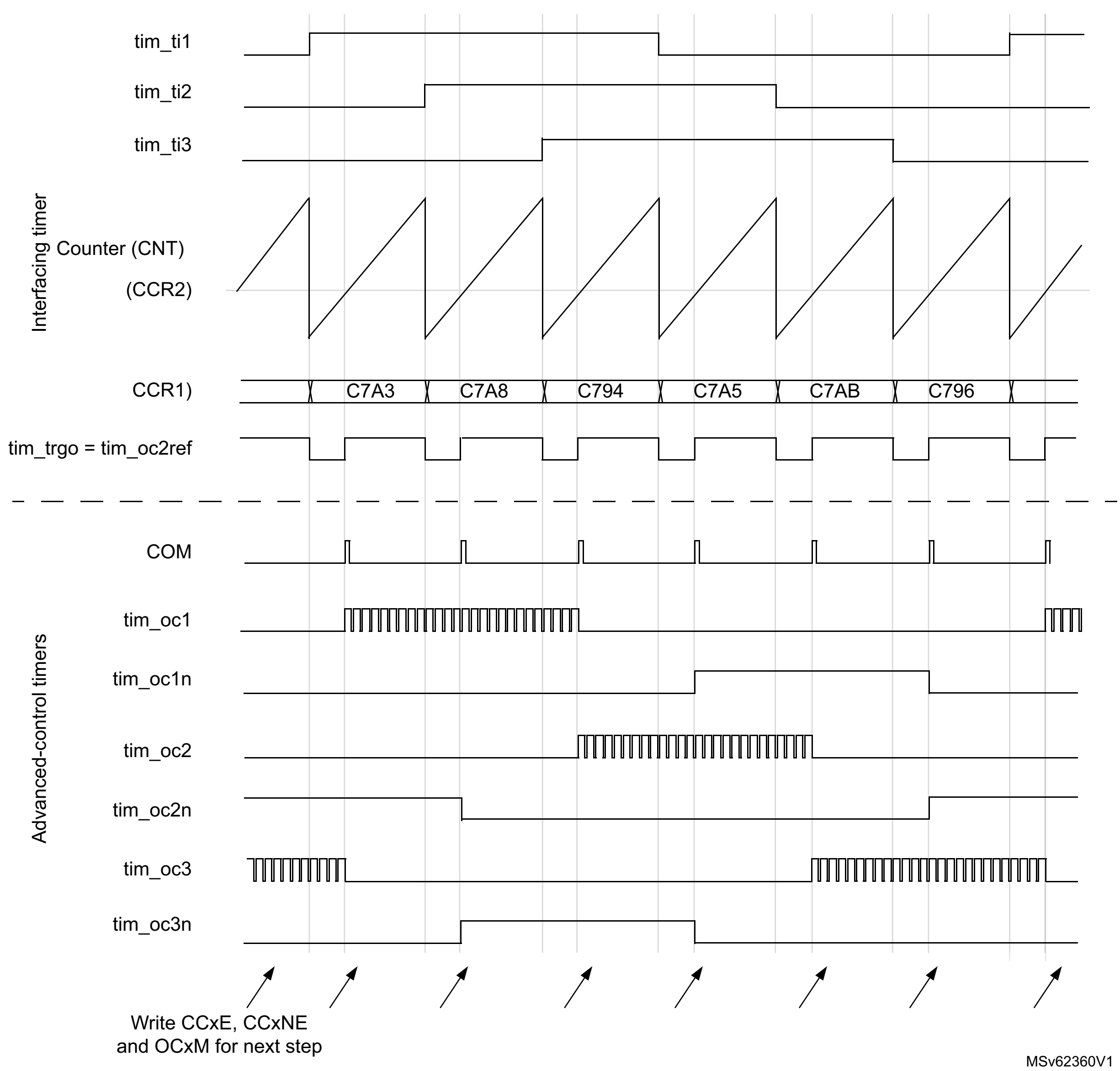
In this example the user wants to change the PWM configuration of the advanced-control timer after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure three 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 two 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).

Figure 240 describes this example.

Figure 240. Example of Hall sensor interface



29.3.30 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 30.4.23: 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.

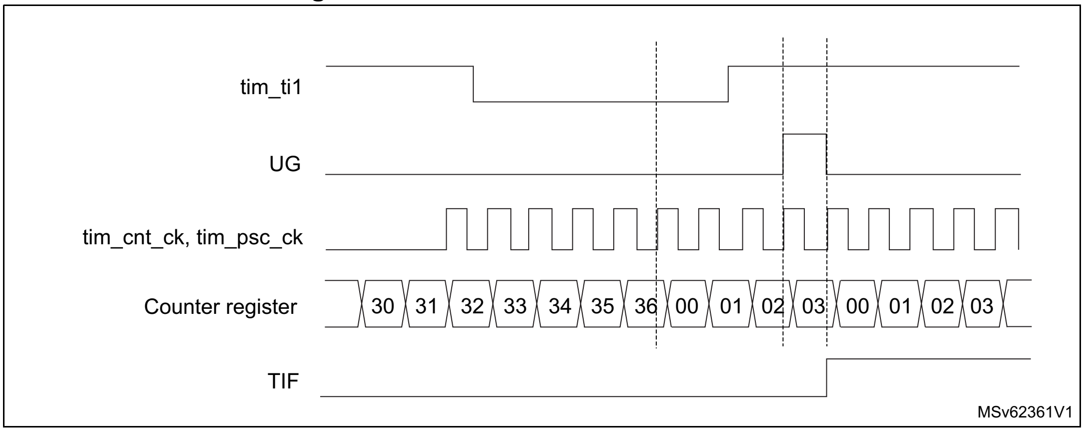
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

- 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 autoreload 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 241. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

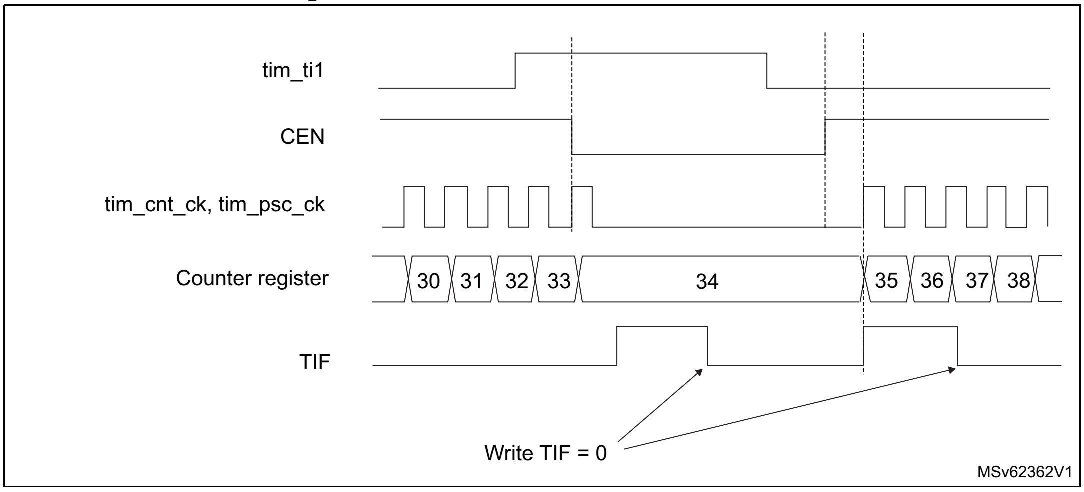
In the following example, the upcounter counts only when tim_ti1 input is low:

- 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 does not 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 242. Control circuit in Gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

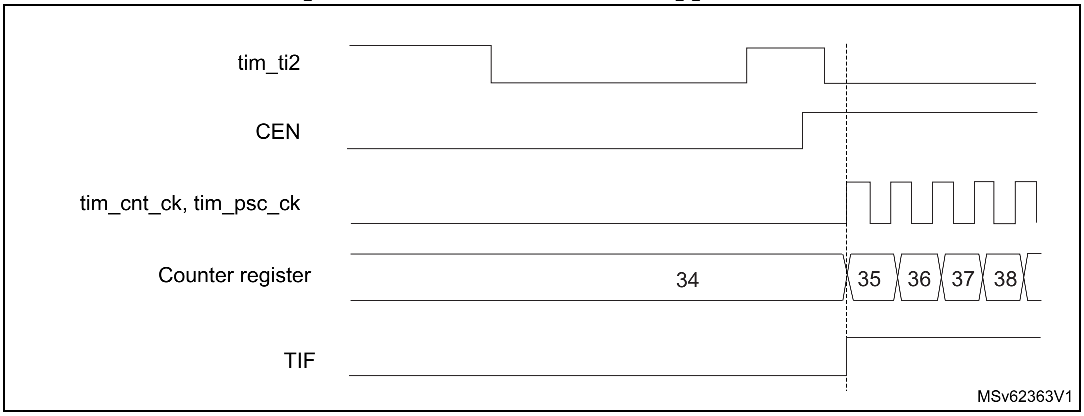
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

- 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 243. 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.

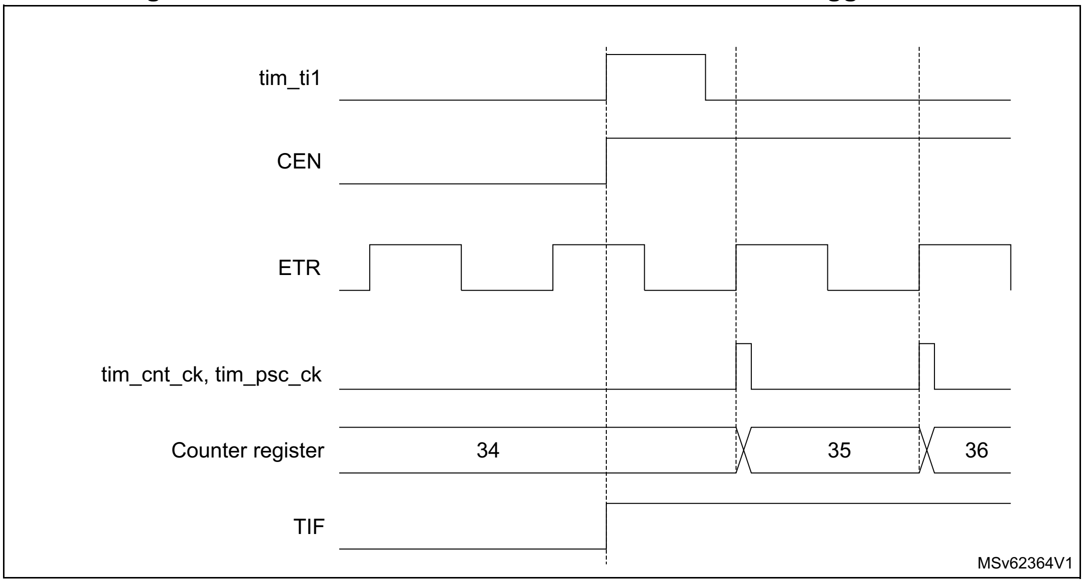
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

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 244. Control circuit in external clock mode 2 + trigger mode



Note: The clock of the slave peripherals (such as 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.

29.3.31 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 197*.

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.

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.

29.3.32 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 reprogram 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 29.6.29: TIM1 DMA control register (TIM1_DCR)* 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 bitfields 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 must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

29.3.33 TIM1 DMA requests

The TIM1 can generate a DMA request, as shown in the table below.

Table 271. 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_trgi_dma	Trigger	TDE

29.3.34 Debug mode

When the microcontroller enters debug mode (CPU1 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).

29.4 TIM1 low-power modes

Table 272. Effect of low-power modes on TIM1

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

29.5 TIM1 interrupts

The TIM1 can generate multiple interrupts, as shown in *Table 273*.

Table 273. 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_UPD		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_TRGI_COM_ DIR_IDX	TIM_COM	Commutation (COM)	COMIF	COMIE	write 0 in COMIF	Yes	No
	TIM_TRGI	Trigger	TIF	TIE	write 0 in TIF	Yes	No
	TIM_IDX	Index	IDXF	IDXIE	write 0 in IDXF	Yes	No
	TIM_DIR	Direction	DIRF	DIRIE	write 0 in DIRF	Yes	No
TIM_BRK_TERR_ IERR	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_TERR	Transition Error	TERRF	TERRIE	write 0 in TERRF	Yes	No

29.6 TIM1 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

29.6.1 TIM1 control register 1 (TIM1_CR1)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITH EN	UIFRE MAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This 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_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**: Autoreload 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.

29.6.2 TIM1 control register 2 (TIM1_CR2)

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.

29.6.3 TIM1 slave mode control register (TIM1_SMCR)

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 $\overline{\text{tim\_etr\_in}}$ 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 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 bitfield 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 bitfield 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 259: 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 (for example 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 (refer to ETP bit in TIMx_SMCR for tim_etr_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).

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.

29.6.4 TIM1 DMA/interrupt enable register (TIM1_DIER)

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

29.6.5 TIM1 status register (TIM1_SR)

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 29.6.3: TIM1 slave mode control register (TIM1_SMCR)*), if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

29.6.6 TIM1 event generation register (TIM1_EGR)

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 autoreload value (TIMx_ARR) if DIR = 1 (downcounting).

29.6.7 TIM1 capture/compare mode register 1 (TIM1_CCMR1)

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 (for example 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 bitfield 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 bitfield 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 bitfield 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 bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

29.6.8 TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1)

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 (for example 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 bitfield 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 can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the ouput keeps the state (active or inactive) it had before entering the frozen state.

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 when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Note: On channels having a complementary output, this bitfield 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 bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

29.6.9 TIM1 capture/compare mode register 2 (TIM1_CCMR2)

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 (for example 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 bitfield 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 bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

29.6.10 TIM1 capture/compare mode register 2 [alternate] (TIM1_CCMR2)

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 (for example 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 bitfield 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 bitfield 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 bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3

10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4

11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

29.6.11 TIM1 capture/compare enable register (TIM1_CCER)

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 274* 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 274. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature

<table>
<tr><th colspan="5">Control bits</th><th colspan="2">Output states(1)</th></tr>
<tr><th>MOE bit</th><th>OSSI bit</th><th>OSSR bit</th><th>CCxE bit</th><th>CCxNE bit</th><th>tim_ocx output state</th><th>tim_ocxn output state</th></tr>
<tr><td rowspan="6">1</td><td rowspan="6">X</td><td>X</td><td>0</td><td>0</td><td colspan="2">Output disabled (not driven by the timer: Hi-Z)
tim_ocx = 0, tim_ocxn = 0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Output disabled (not driven by the timer: Hi-Z)
tim_ocx = 0</td><td>tim_ocxref + Polarity tim_ocxn = tim_ocxref xor CCxNP</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>tim_ocxref + Polarity
tim_ocx = tim_ocxref xor CCxP</td><td>Output Disabled (not driven by the timer: Hi-Z)
tim_ocxn = 0</td></tr>
<tr><td>X</td><td>1</td><td>1</td><td>OCREF + Polarity + dead-time</td><td>Complementary to OCREF (not OCREF) + Polarity + dead-time</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Off-State (output enabled with inactive state)
tim_ocx = CCxP</td><td>tim_ocxref + Polarity
tim_ocxn = tim_ocxref x or CCxNP</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>tim_ocxref + Polarity
tim_ocx = tim_ocxref xor CCxP</td><td>Off-State (output enabled with inactive state)
tim_ocxn = CCxNP</td></tr>
<tr><td rowspan="5">0</td><td>0</td><td rowspan="5">X</td><td>X</td><td>X</td><td colspan="2" rowspan="2">Output disabled (not driven by the timer: Hi-Z).</td></tr>
<tr><td rowspan="4">1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td colspan="2" rowspan="3">Off-State (output enabled with inactive state)
Asynchronously: 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.</td></tr>
<tr><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table>

1. When both outputs of a channel are not used (control taken over by GPIO), 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.

29.6.12 TIM1 counter (TIM1_CNT)

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.

29.6.13 TIM1 prescaler (TIM1_PSC)

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”).

29.6.14 TIM1 autoreload register (TIM1_ARR)

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]**: Autoreload value

ARR is the value to be loaded in the actual autoreload register.

Refer to the *Section 29.3.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the autoreload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the autoreload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

29.6.15 TIM1 repetition counter register (TIM1_RCR)

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.

29.6.16 TIM1 capture/compare register 1 (TIM1_CCR1)

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.

29.6.17 TIM1 capture/compare register 2 (TIM1_CCR2)

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.

29.6.18 TIM1 capture/compare register 3 (TIM1_CCR3)

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.

29.6.19 TIM1 capture/compare register 4 (TIM1_CCR4)

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.

<u>Non-dithering mode (DITHEN = 0)</u>

The register holds the compare value in CCR4[15:0]. The CCR4[19:16] bits are reset.

<u>Dithering mode (DITHEN = 1)</u>

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.

<u>Non-dithering mode (DITHEN = 0)</u>

The register holds the capture value in CCR4[15:0]. The CCR4[19:16] bits are reset.

<u>Dithering mode (DITHEN = 1)</u>

The register holds the capture in CCR4[19:4]. The CCR4[3:0] bits are reset.

29.6.20 TIM1 break and dead-time register (TIM1_BDTR)

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, see *Figure 202: 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 bitfield 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 bitfield 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 29.6.11: TIM1 capture/compare enable register (TIM1_CCER)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if none of the break inputs 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 tim_sys_brk and BKIN sources, as per *Figure 202: 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 29.6.11: TIM1 capture/compare enable register (TIM1_CCER)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic, which forces a Hi-Z state).

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE = 1 or CCxNE = 1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for idle mode

This bit is used when MOE = 0 due to a break event or by a software write, on channels configured as outputs.

See OC/OCN enable description for more details (*Section 29.6.11: TIM1 capture/compare enable register (TIM1_CCER)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic and which imposes a Hi-Z state).

1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the deadtime. The timer maintains its control over the output.

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected.

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and 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 bitfield 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} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

29.6.21 TIM1 capture/compare register 5 (TIM1_CCR5)

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.

29.6.22 TIM1 capture/compare register 6 (TIM1_CCR6)

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.

29.6.23 TIM1 capture/compare mode register 3 (TIM1_CCMR3)

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.

29.6.24 TIM1 timer deadtime register 2 (TIM1_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 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 bitfield 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} = 125 ns (8 MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps
Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

29.6.25 TIM1 timer encoder control register (TIM1_ECR)

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

Bit 0 **IE**: Index enable

This bit indicates if the Index event resets the counter.

0: Index disabled
1: Index enabled

29.6.26 TIM1 timer input selection register (TIM1_TISEL)

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 29.3.2: TIM1 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 29.3.2: TIM1 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 29.3.2: TIM1 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 29.3.2: TIM1 pins and internal signals* for interconnects list.

29.6.27 TIM1 alternate function option register 1 (TIM1_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.	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 29.3.2: TIM1 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 29.3.2: TIM1 pins and internal signals for product specific implementation.

29.6.28 TIM1 alternate function register 2 (TIM1_AF2)

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 29.3.2: TIM1 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 29.3.2: TIM1 pins and internal signals for product specific implementation.

29.6.29 TIM1 DMA control register (TIM1_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
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 is given by the following equation:

(TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data are copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

–If the DMA Data Size is configured in half-words, 16-bit data are transferred to each of the 7 registers.

–If the DMA Data Size is configured in bytes, the data are also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:
00000: TIMx_CR1
00001: TIMx_CR2
00010: TIMx_SMCR
...

29.6.30 TIM1 DMA address for full transfer (TIM1_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 (TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

29.6.31 TIM1 register map

TIM1 registers are mapped as 16-bit addressable registers as described in the table below:

Table 275. TIM1 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
	Reset value							0	0			0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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 275. TIM1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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 275. TIM1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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.	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.	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: Memory organization* for the register boundary addresses.

30 General-purpose timers (TIM2/TIM3)

30.1 TIM2/TIM3 introduction

The general-purpose timers consist of a 16-bit or 32-bit autoreload counter driven by a programmable prescaler.

They can 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 30.4.23: Timer synchronization*.

30.2 TIM2/TIM3 main features

General-purpose TIMx timer features include:

- 16-bit or 32-bit up, down, up/down autoreload 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 four 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.

30.3 TIM2/TIM3 implementation

Table 276. General purpose timers(1)

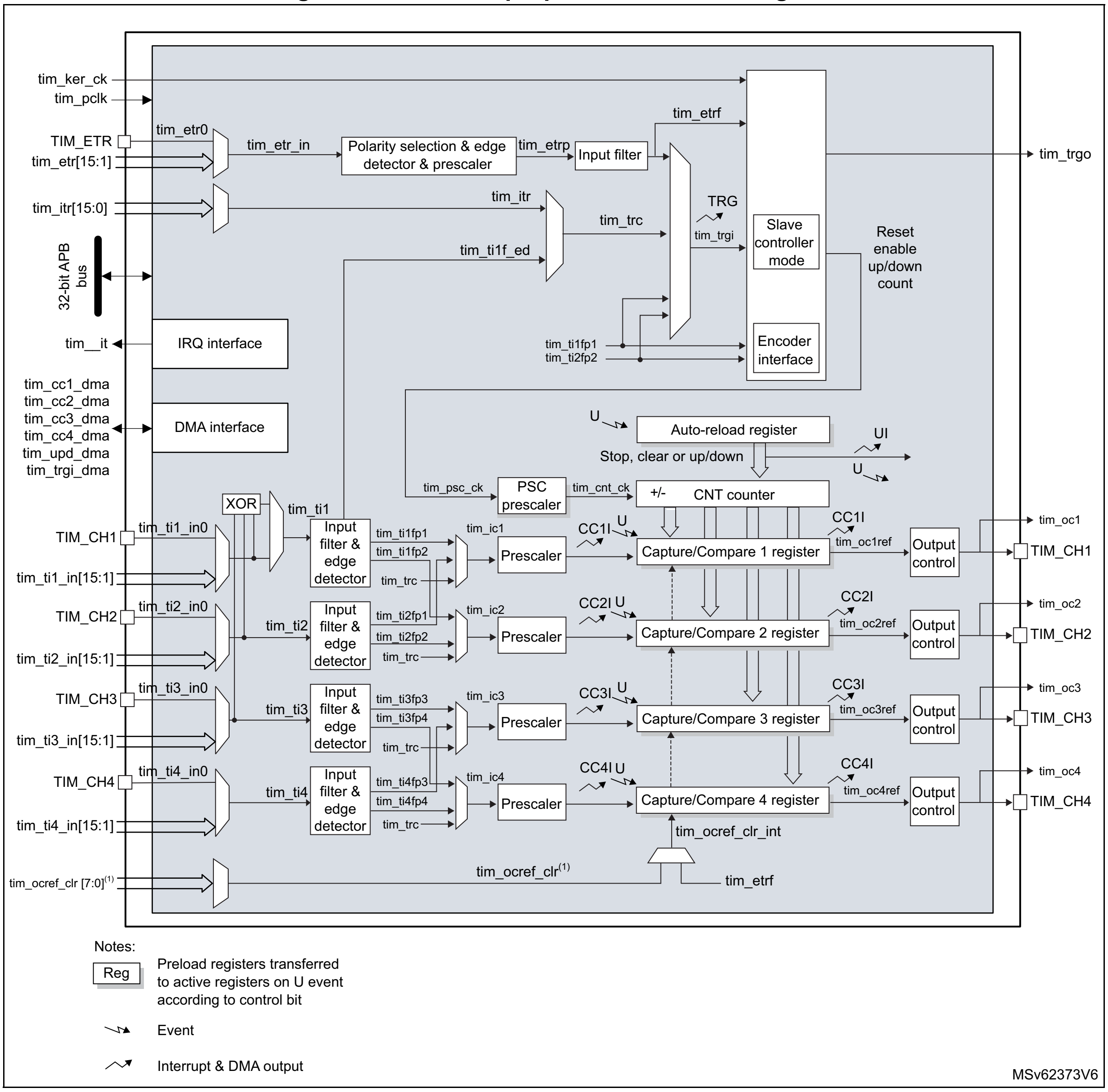
Timer instance	TIM2	TIM3(2)
CHx	1 to 4	1 to 4
ETR	X	X
Resolution	32-bit	16-bit
DMA request channel	1 to 4	1 to 4
DMA request update	Yes	Yes
DMA request trigger	No	Yes
OCREF clear sources	tim2_etrf tim2_ocref_clr[7:0]	tim3_etrf tim3_ocref_clr[7:0]

1. Note: ‘X’ = supported, ‘-’= not supported.
2. Available only on STM32WBA52/54/55xx devices.

30.4 TIM2/TIM3 functional description

30.4.1 Block diagram

Figure 245. General-purpose timer block diagram



1. This feature is not available on all timers, refer to *Section 30.3: TIM2/TIM3 implementation*.

30.4.2 TIM2/TIM3 pins and internal signals

Table 277 and *Table 278* in this section summarize the TIM inputs and outputs.

Table 277. 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 278. 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 278. 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 [4:1] dma requests.
tim_upd_dma	Output	Timer update dma request.
tim_trgi_dma	Output	Timer trigger dma request.

Table 279, *Table 280*, *Table 281* and *Table 282* are listing the sources connected to the tim_ti[4:1] input multiplexers.

Table 279. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources	
	TIM2	TIM3
tim_ti1_in0	TIM2_CH1	TIM3_CH1
tim_ti1_in1	COMP1_OUT(1)	COMP1_OUT(1)
tim_ti1_in2	COMP2_OUT(1)	COMP2_OUT(1)
tim_ti1_in[15:3]	Reserved	

1. Only available on STM32WBA54xx and STM32WBA55xx devices.

Table 280. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources	
	TIM2	TIM3
tim_ti2_in0	TIM2_CH2	TIM3_CH2
tim_ti2_in1	COMP1_OUT(1)	COMP1_OUT(1)
tim_ti2_in2	COMP2_OUT(1)	COMP2_OUT(1)
tim_ti2_in[15:3]	Reserved	

1. Only available on STM32WBA54xx and STM32WBA55xx devices.

Table 281. Interconnect to the tim_ti3 input multiplexer

tim_ti3 inputs	Sources	
	TIM2	TIM3
tim_ti3_in0	TIM2_CH3	TIM3_CH3
tim_ti3_in[15:1]	Reserved	

Table 282. Interconnect to the tim_ti4 input multiplexer

tim_ti4 inputs	Sources	
	TIM2	TIM3
tim_ti4_in0	TIM2_CH4	TIM3_CH4
tim_ti4_in1	COMP1_OUT[1]	Reserved
tim_ti4_in2	COMP2_OUT[1]	Reserved
tim_ti4_in[15:3]	Reserved	

1. Only available on STM32WBA54xx and STM32WBA55xx devices.

Table 283 lists the internal sources connected to the tim_etr input multiplexer.

Table 283. TIMx internal trigger connection

TIMx	TIM2	TIM3
tim_itr0	tim1_trgo	tim1_trgo
tim_itr1	Reserved	tim2_trgo
tim_itr2	tim3_trgo	Reserved
tim_itr[6:3]	Reserved	
tim_itr7	tim16_oc1	tim16_oc1
tim_itr8	tim17_oc1	tim17_oc1
tim_itr[15:9]	Reserved	

Table 284 lists the internal sources connected to the tim_etr input multiplexer.

Table 284. Interconnect to the tim_etr input multiplexer

Timer external trigger input signal	Timer external trigger signals assignment	
	TIM2	TIM3
tim_etr0	TIM2_ETR	TIM3_ETR
tim_etr1	COMP1_OUT[1]	COMP1_OUT[1]
tim_etr2	COMP2_OUT[1]	COMP2_OUT[1]
tim_etr3	Reserved	
tim_etr4	HSI16	HSI16
tim_etr[7:5]	Reserved	
tim_etr8	TIM3_ETR	TIM2_ETR
tim_etr[10:9]	Reserved	
tim_etr11	LSE	adc4_awd1
tim_etr12	Reserved	adc4_awd2
tim_etr13	Reserved	adc4_awd3
tim_etr[15:14]	Reserved	

1. Only available on STM32WBA54xx and STM32WBA55xx devices.

Table 285 lists the internal sources connected to the tim_ocref_clr input multiplexer.

Table 285. Interconnect to the tim_ocref_clr input multiplexer

Timer tim_ocref_clr signal	Timer tim_ocref_clr signals assignment	
	TIM2	TIM3
tim_ocref_clr0	COMP1_OUT(1)	COMP1_OUT(1)
tim_ocref_clr1	COMP2_OUT(1)	COMP2_OUT(1)
tim_ocref_clr[7:2]	Reserved	

1. Only available on STM32WBA54xx and STM32WBA55xx devices.

30.4.3 Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related autoreload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the autoreload 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)
- Autoreload register (TIMx_ARR).

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when down-counting) 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 one 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 246 and *Figure 247* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 246. Counter timing diagram with prescaler division change from 1 to 2

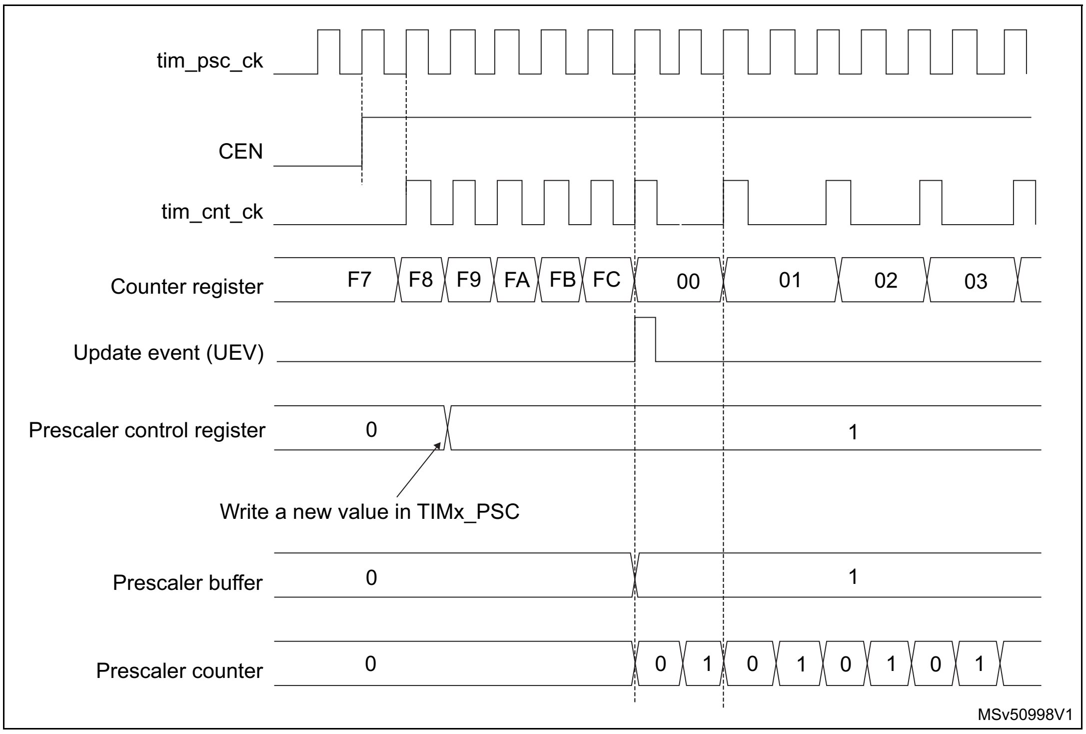
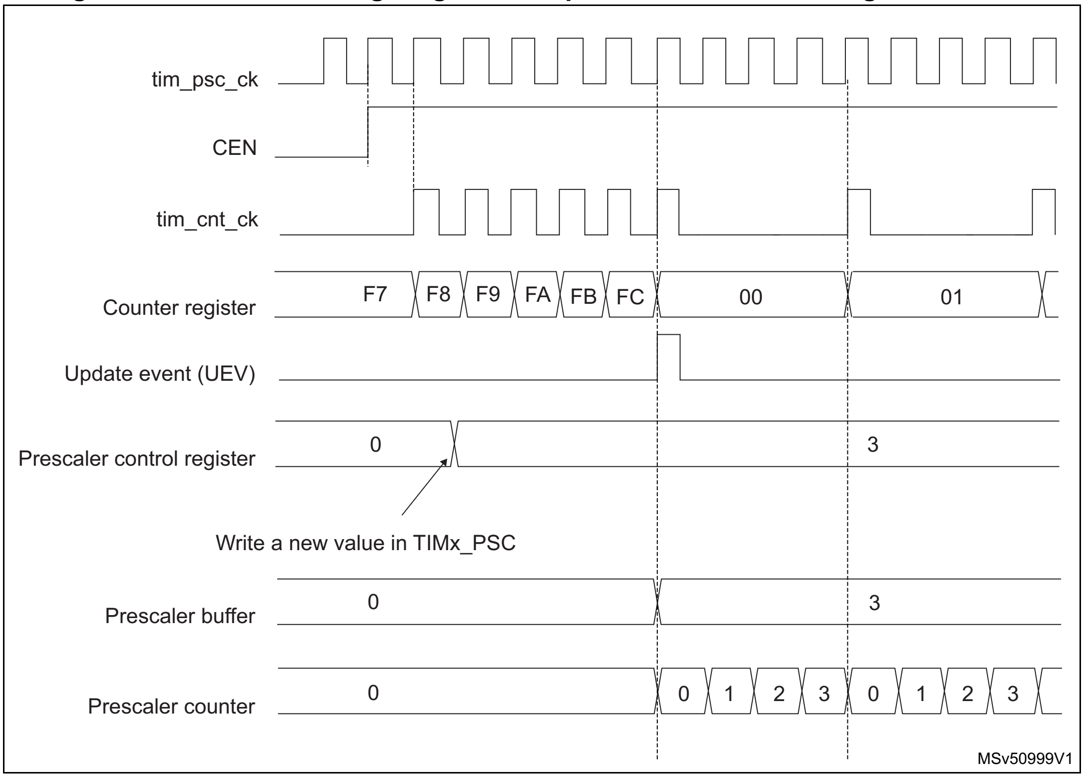


Figure 247. Counter timing diagram with prescaler division change from 1 to 4



30.4.4 Counter modes

Up-counting mode

In up-counting mode, the counter counts from 0 to the autoreload 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 autoreload 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 248. Counter timing diagram, internal clock divided by 1

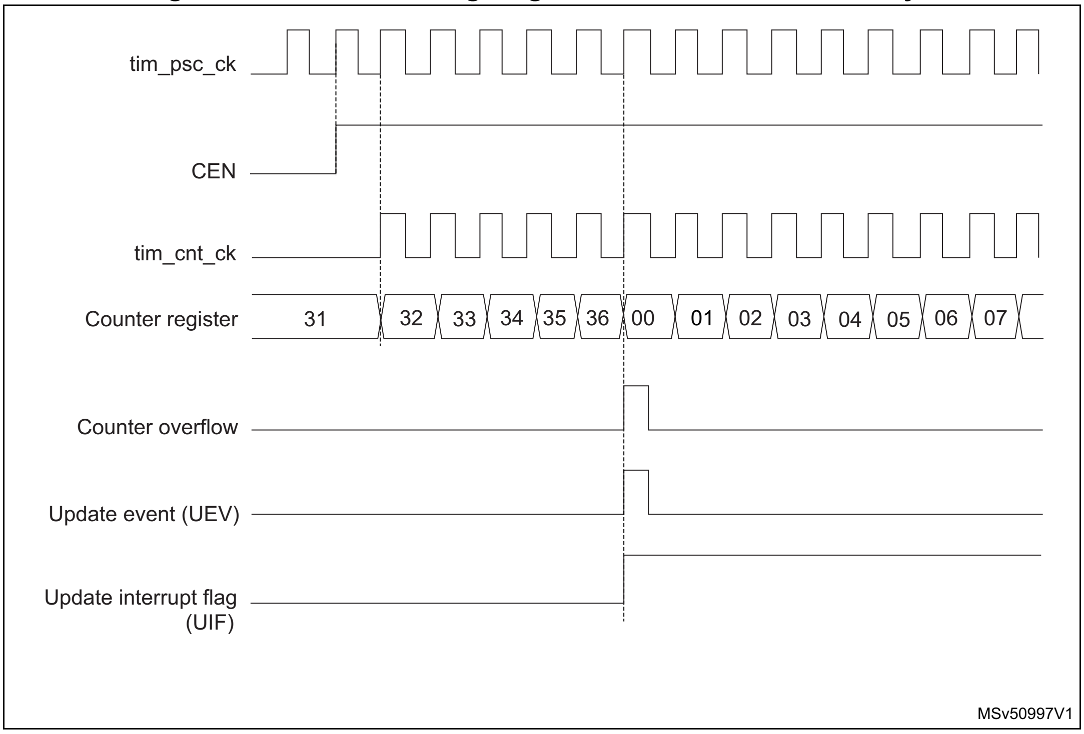


Figure 249. Counter timing diagram, internal clock divided by 2

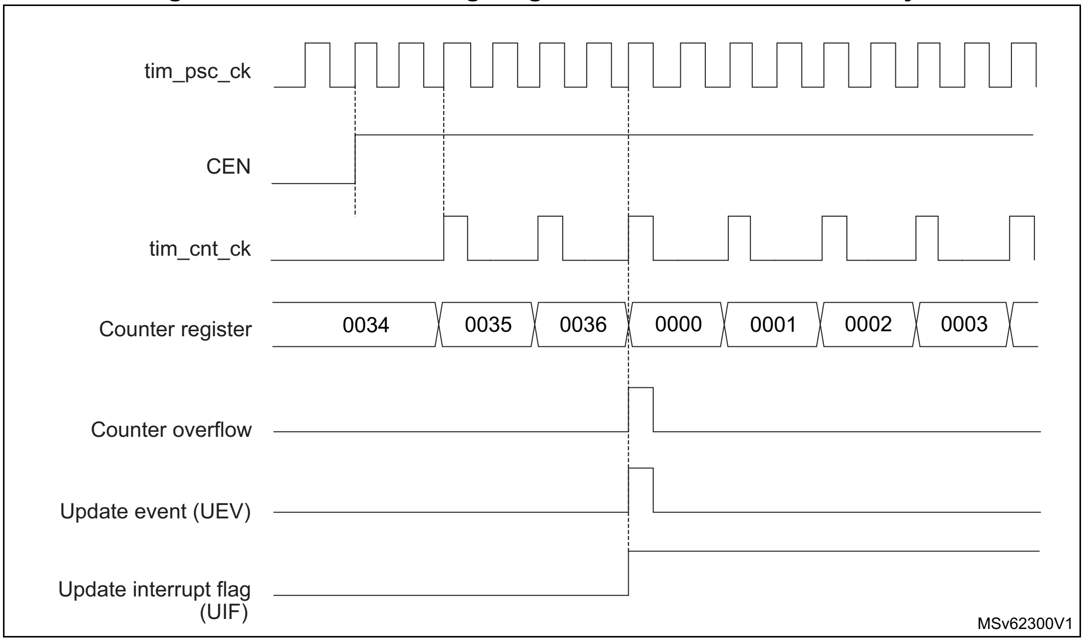


Figure 250. Counter timing diagram, internal clock divided by 4

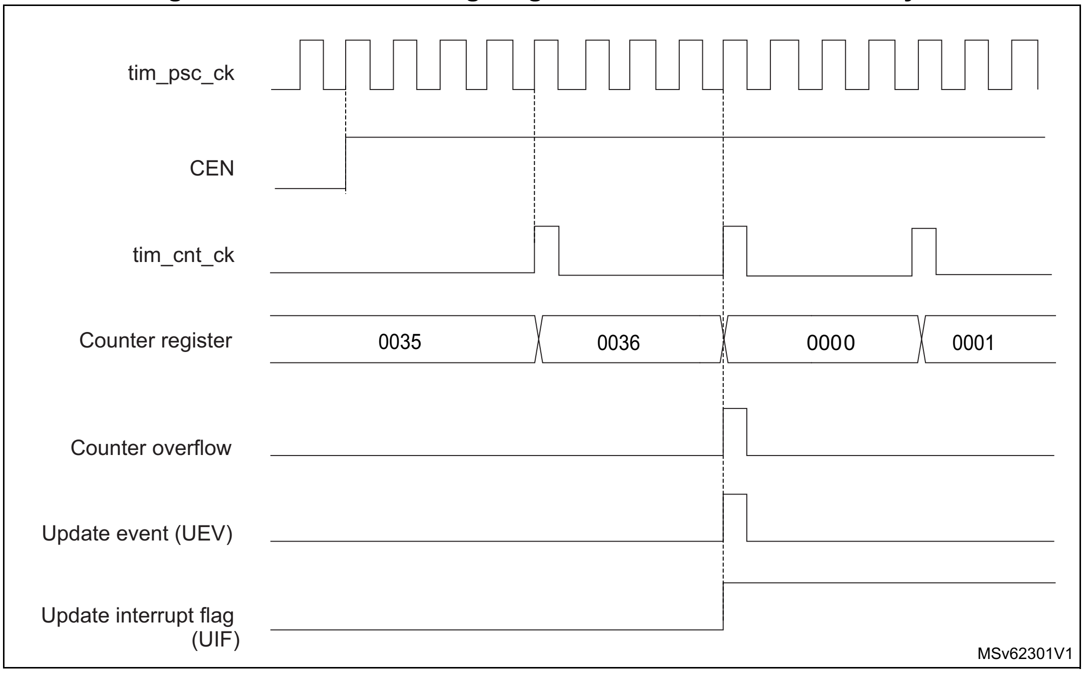


Figure 251. Counter timing diagram, internal clock divided by N

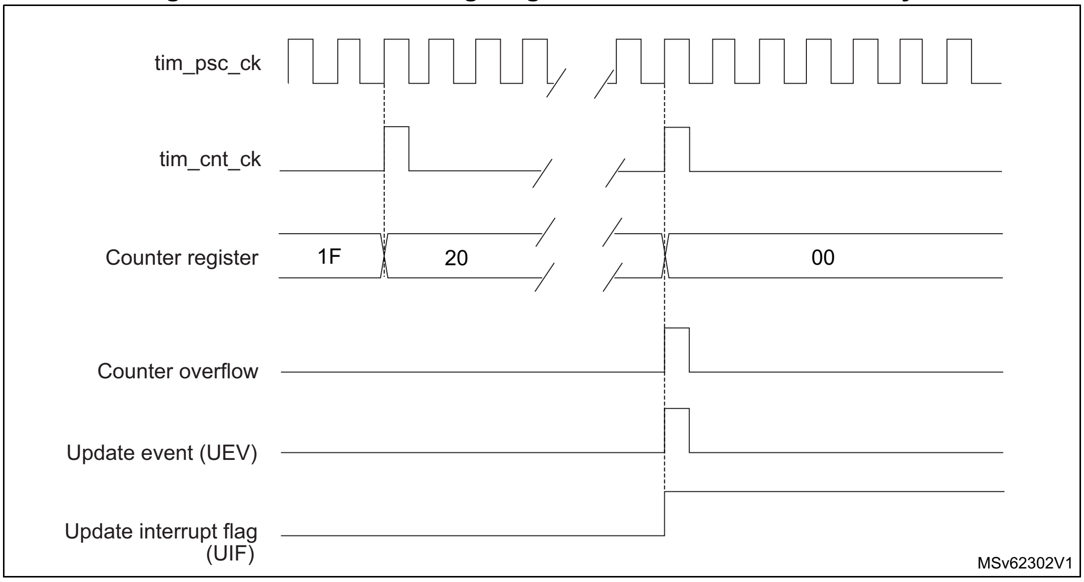


Figure 252. Counter timing diagram, Update event when ARPE = 0 (TIMx_ARR not preloaded)

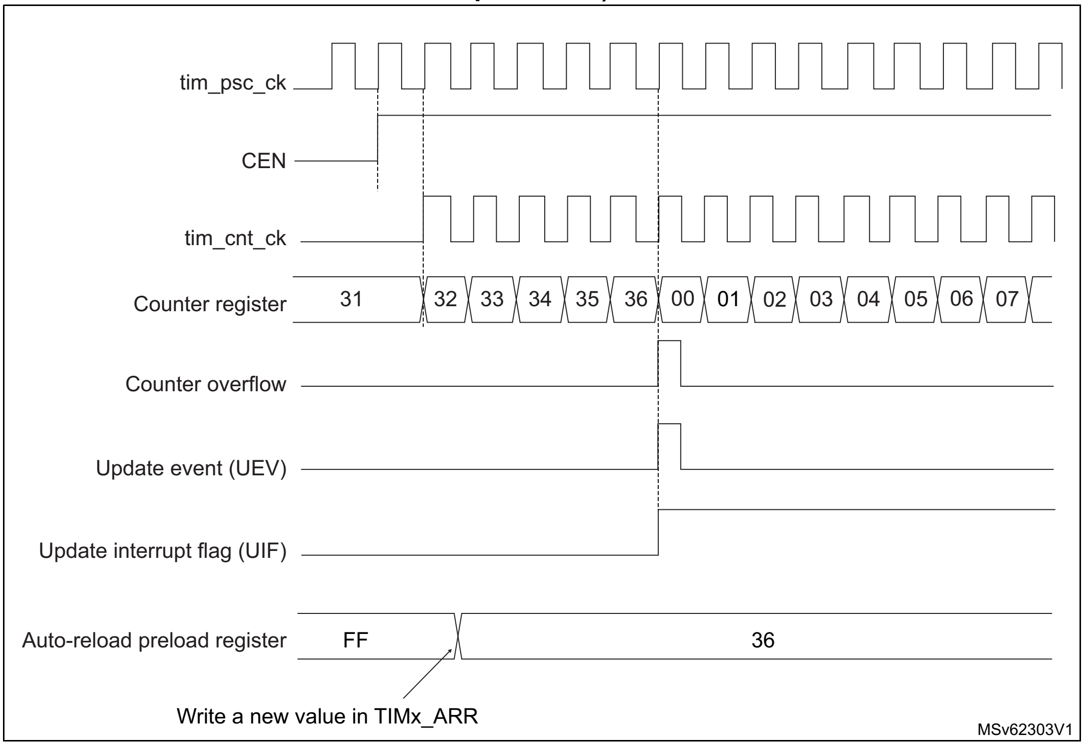
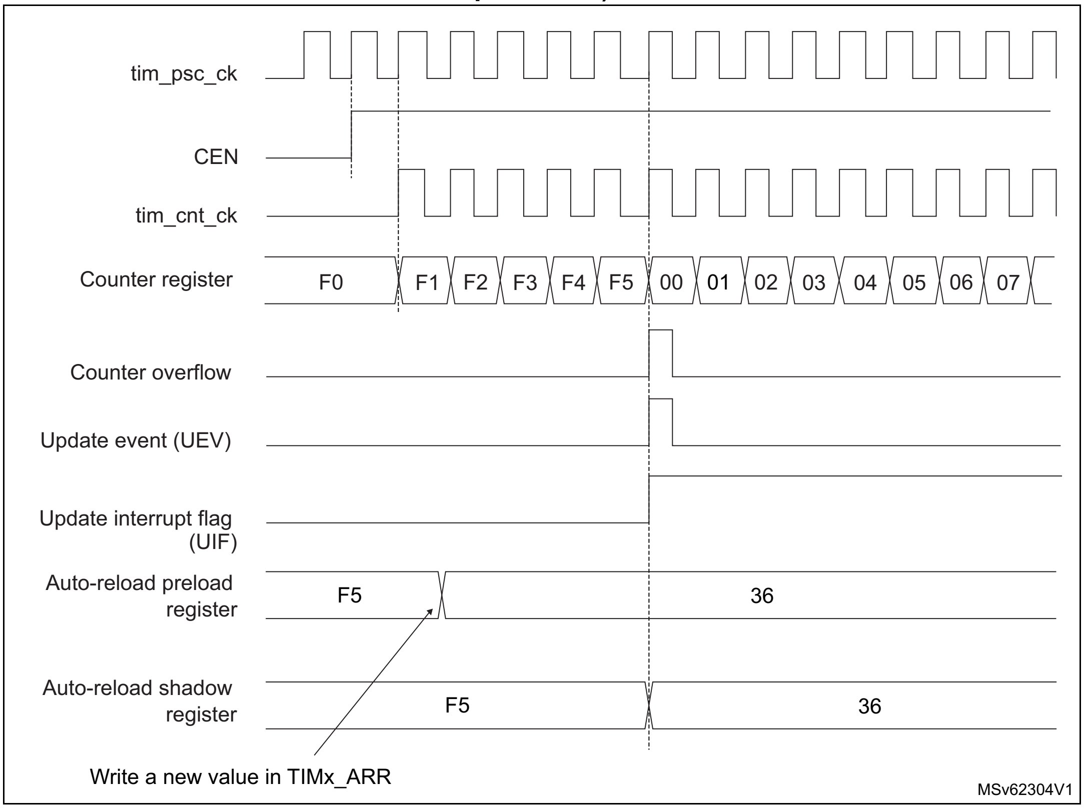


Figure 253. Counter timing diagram, Update event when ARPE = 1 (TIMx_ARR preloaded)



Down-counting mode

In down-counting mode, the counter counts from the autoreload value (content of the TIMx_ARR register) down to 0, then restarts from the autoreload value and generates a counter underflow event.

An update event can be generated 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 autoreload value, whereas the counter of the prescaler restarts from 0 (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 autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that the autoreload 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 254. Counter timing diagram, internal clock divided by 1

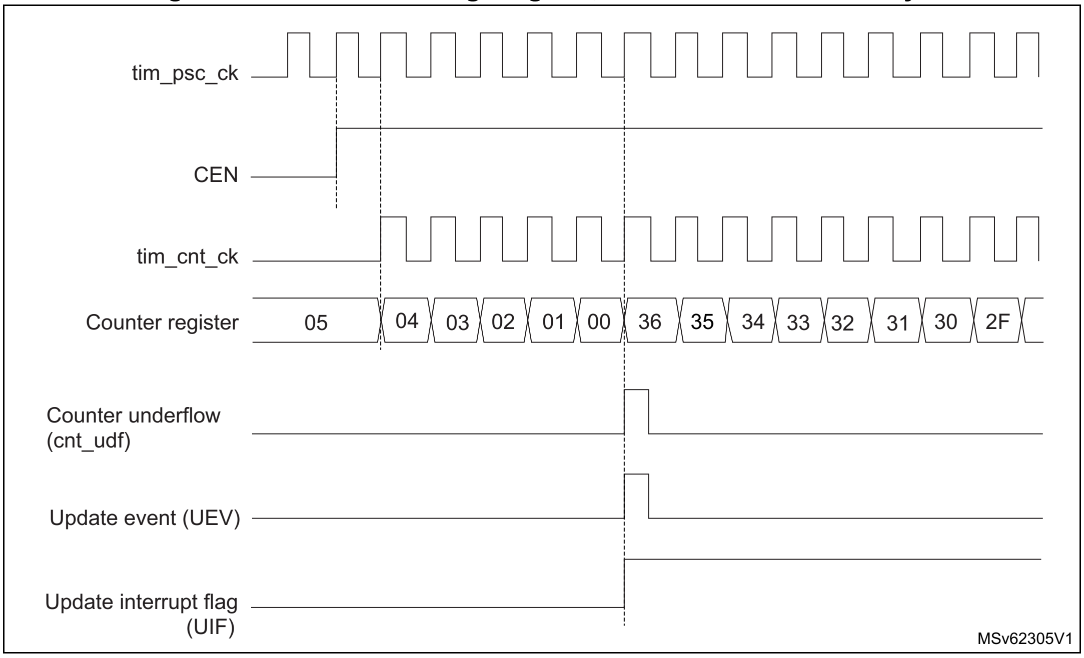


Figure 255. Counter timing diagram, internal clock divided by 2

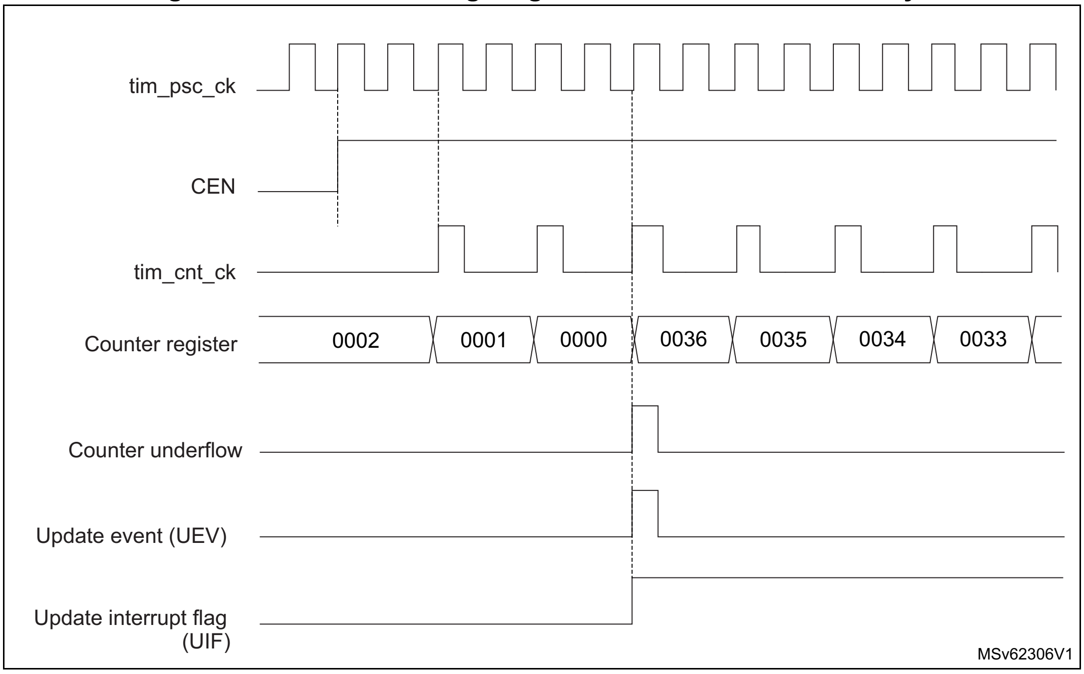


Figure 256. Counter timing diagram, internal clock divided by 4

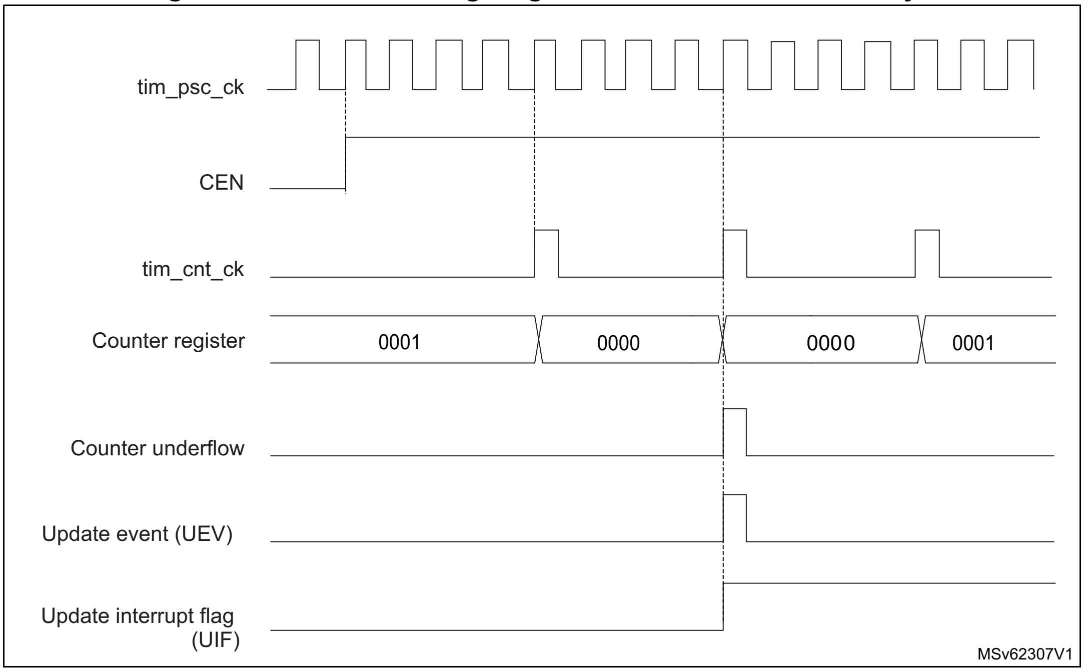


Figure 257. Counter timing diagram, internal clock divided by N

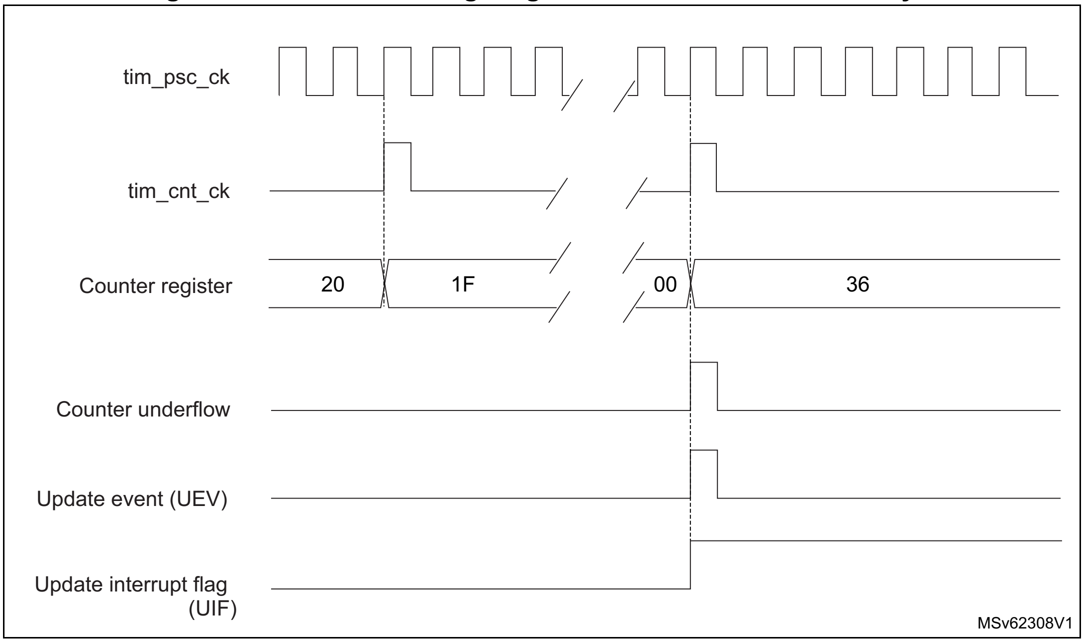
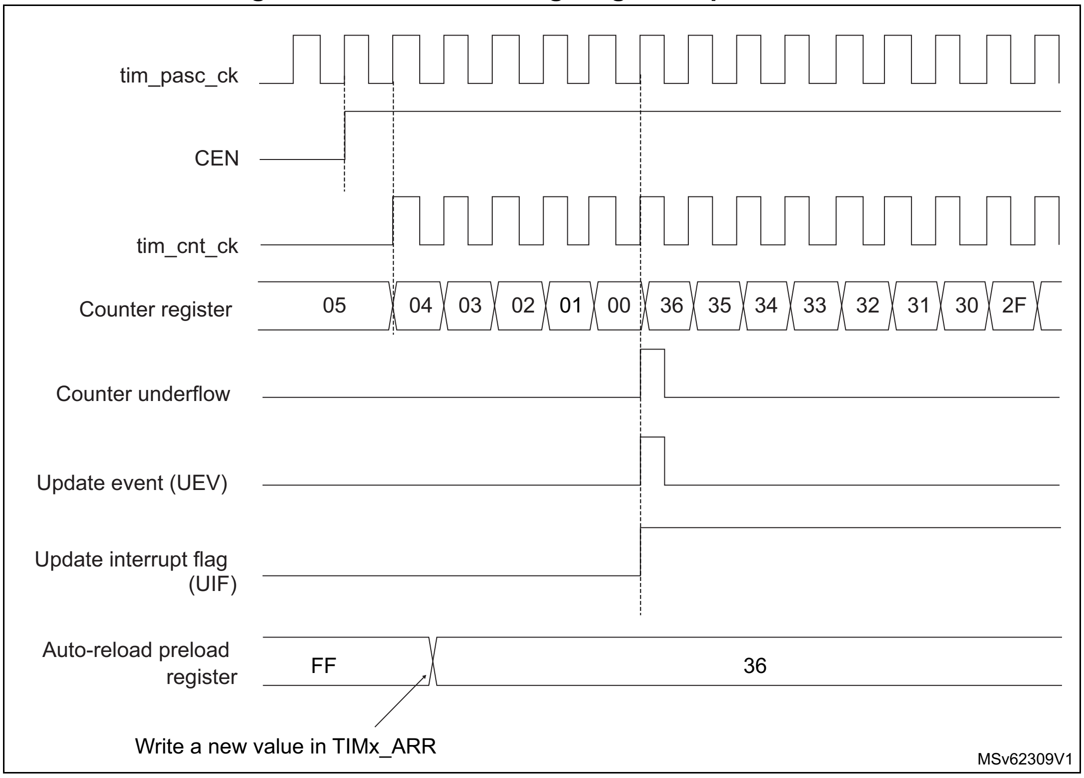


Figure 258. Counter timing diagram, Update event



Center-aligned mode (up/down-counting)

In center-aligned mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the autoreload 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 autoreload value.

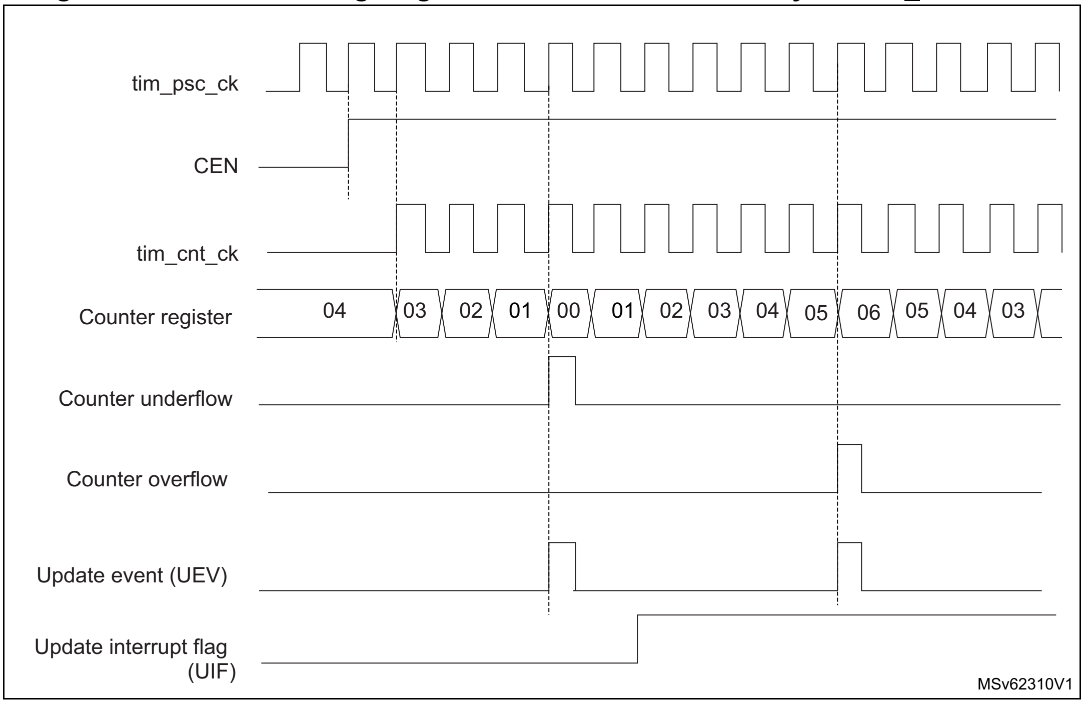
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on 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 autoreload 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 autoreload 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 259. 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 30.5.1: TIMx control register 1 (TIMx_CR1)(x = 2, 3)*).

Figure 260. Counter timing diagram, internal clock divided by 2

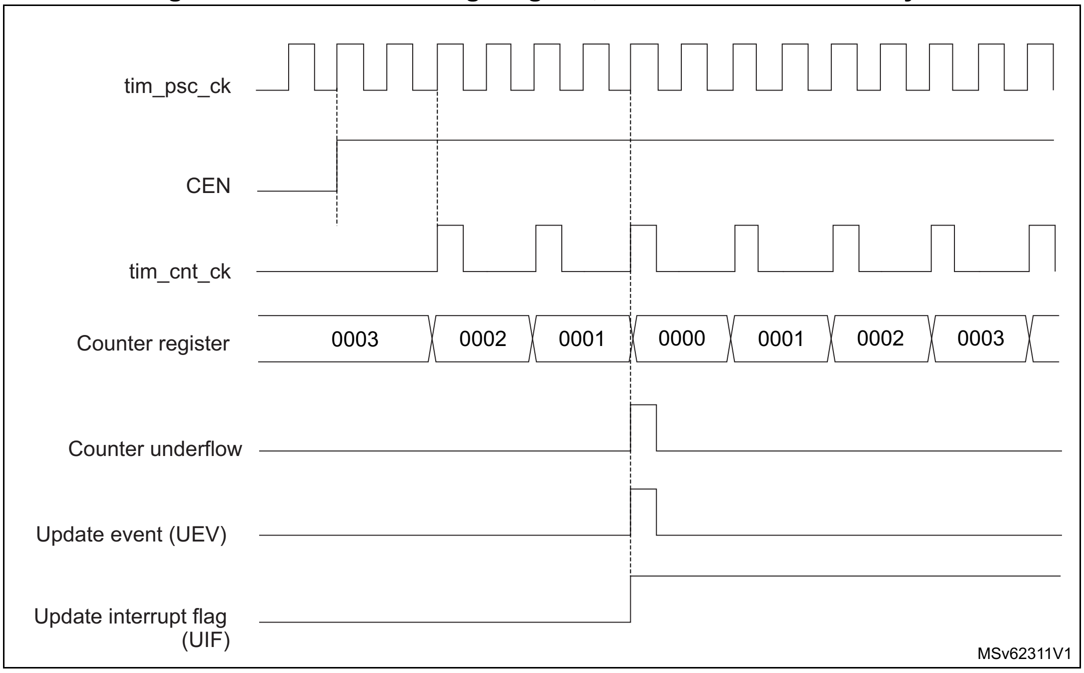
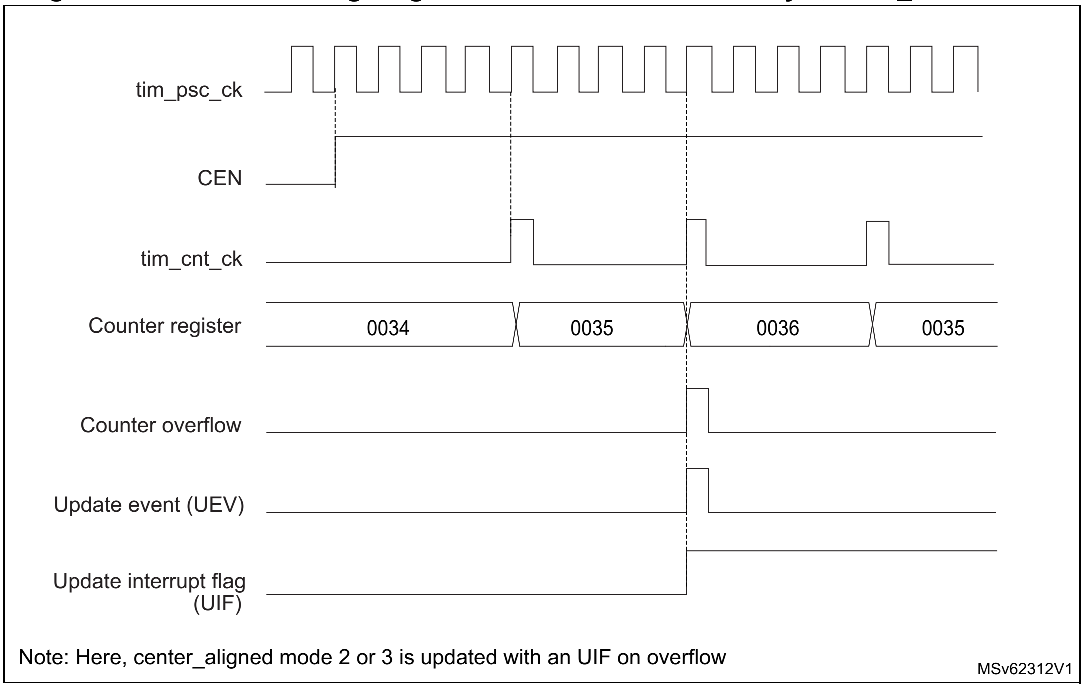


Figure 261. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36



1. Center-aligned mode 2 or 3 is used with a UIF on overflow.

Figure 262. Counter timing diagram, internal clock divided by N

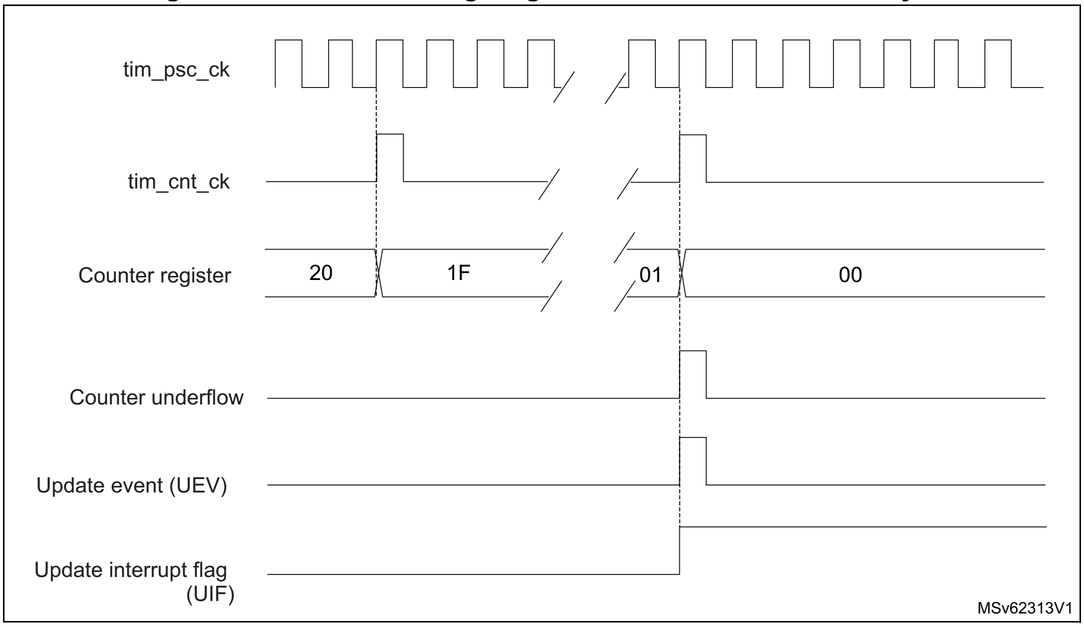


Figure 263. Counter timing diagram, Update event with ARPE = 1 (counter underflow)

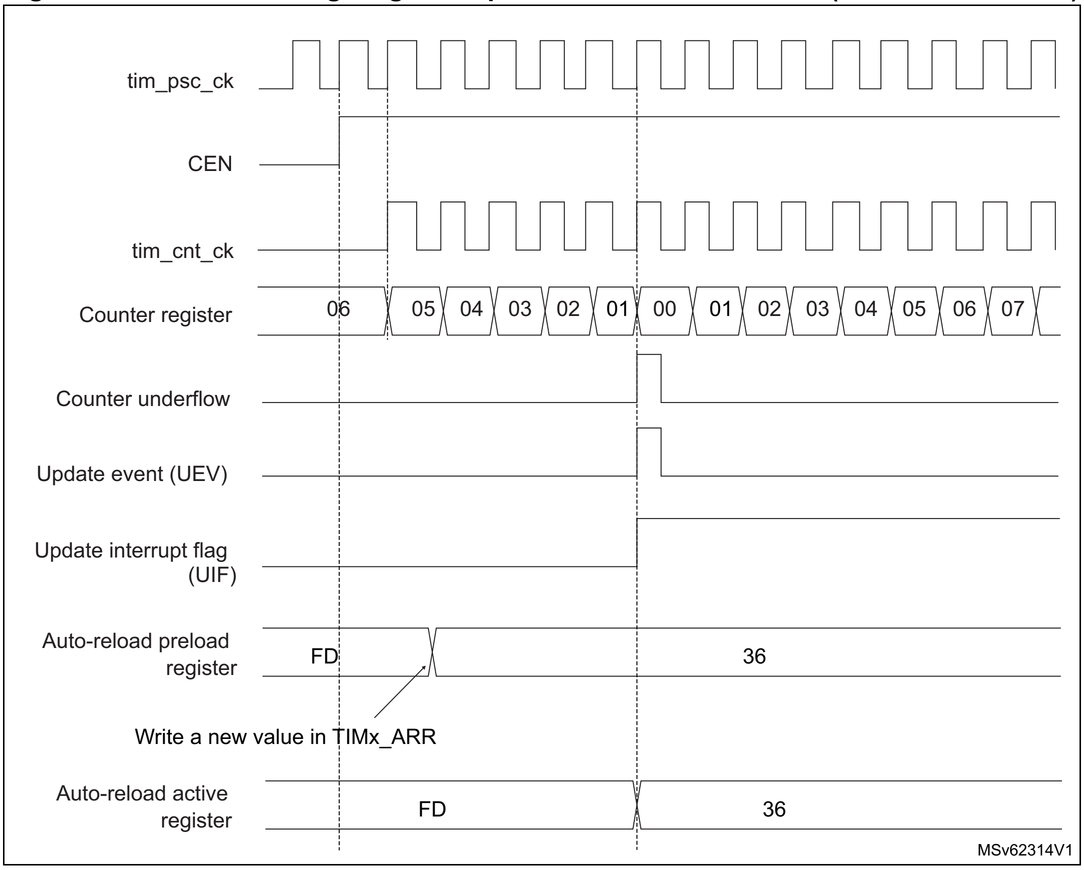
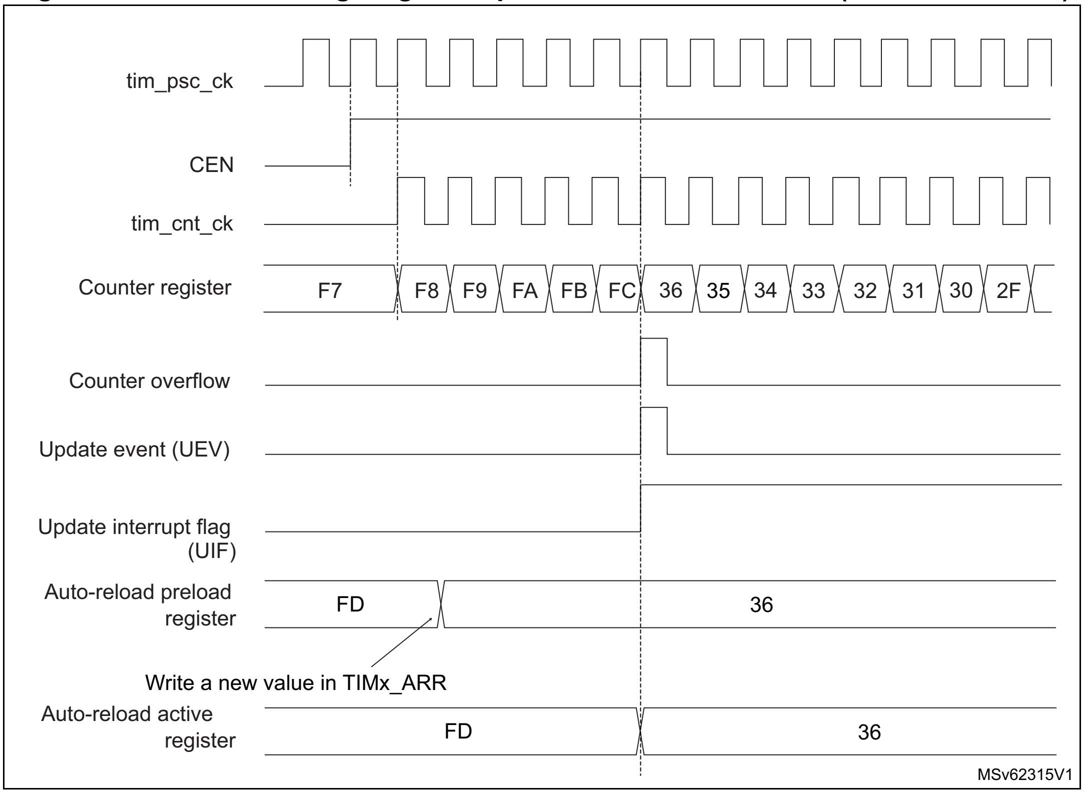


Figure 264. Counter timing diagram, Update event with ARPE = 1 (counter overflow)



30.4.5 Clock selection

The counter clock can be provided by the following clock sources:

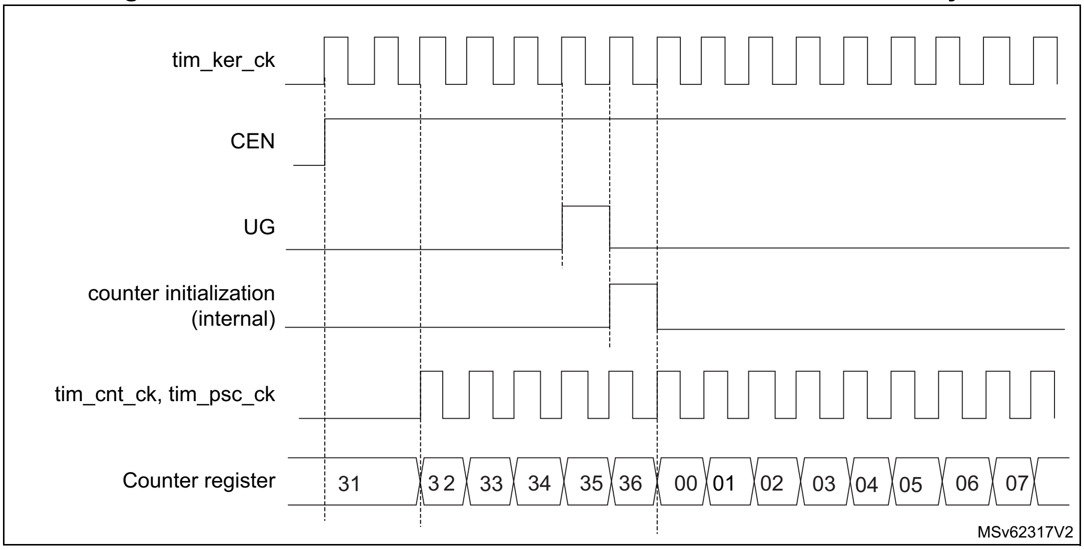
- Internal clock (tim_ker_ck).
- External clock mode1: external input pin (tim_ti1 or tim_ti2).
- External clock mode2: external trigger input (tim_etr_in).
- Internal trigger inputs (tim_itr): using one timer as prescaler for another timer, for example, timer 1 can be configured to act as a prescaler for timer 2. Refer to *Using one timer as prescaler for another timer* 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 265 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

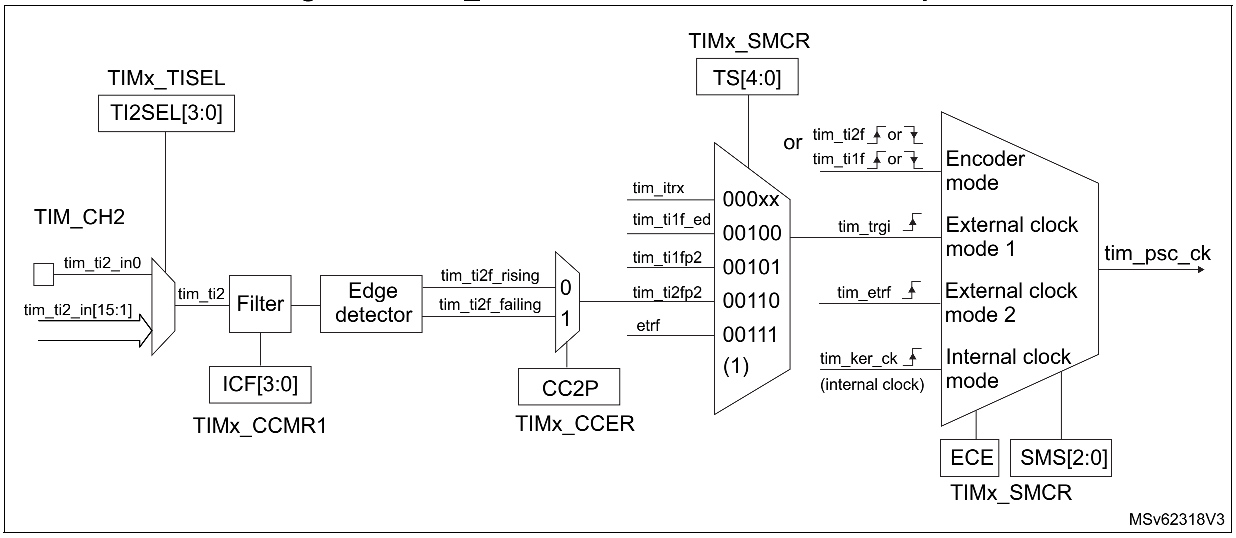
Figure 265. Control circuit in normal mode, internal clock divided by 1



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 266. 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:

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).

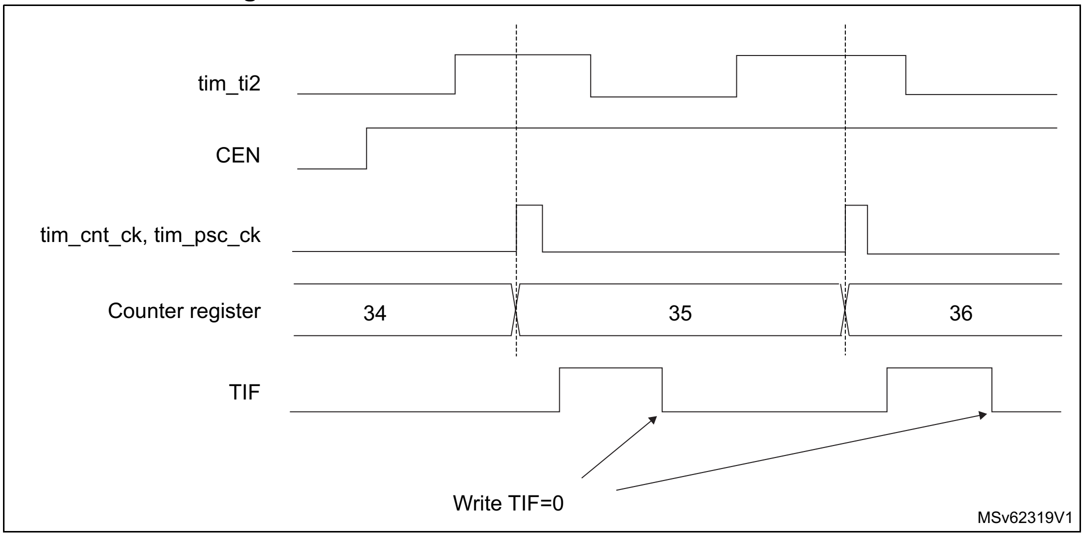
Note: The capture prescaler is not used for triggering, so it does not need to be configured.

4. Select rising edge polarity by writing CC2P = 0 and CC2NP = 0 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 267. Control circuit in external clock mode 1



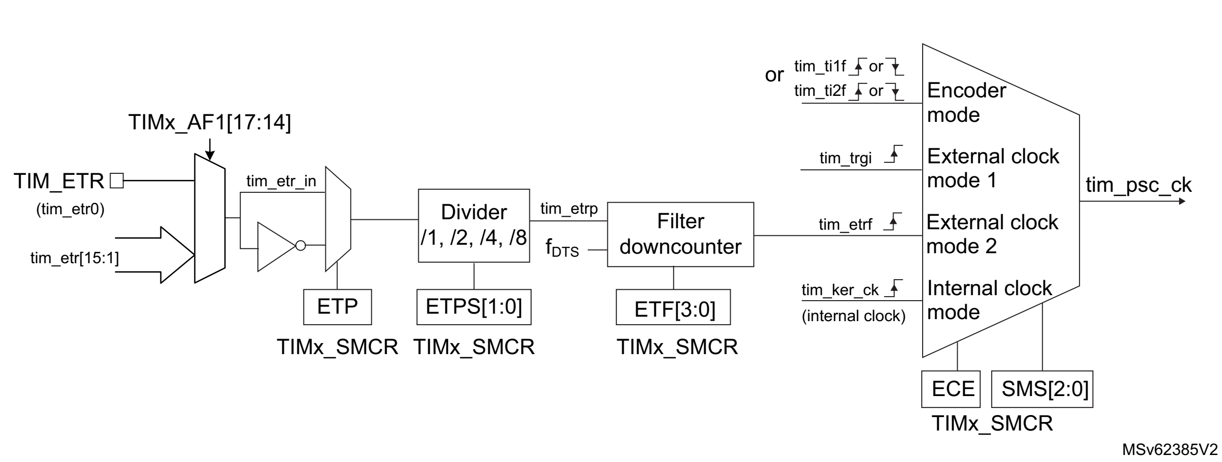
External clock source mode 2

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 268 gives an overview of the external trigger input block.

Figure 268. External trigger input block



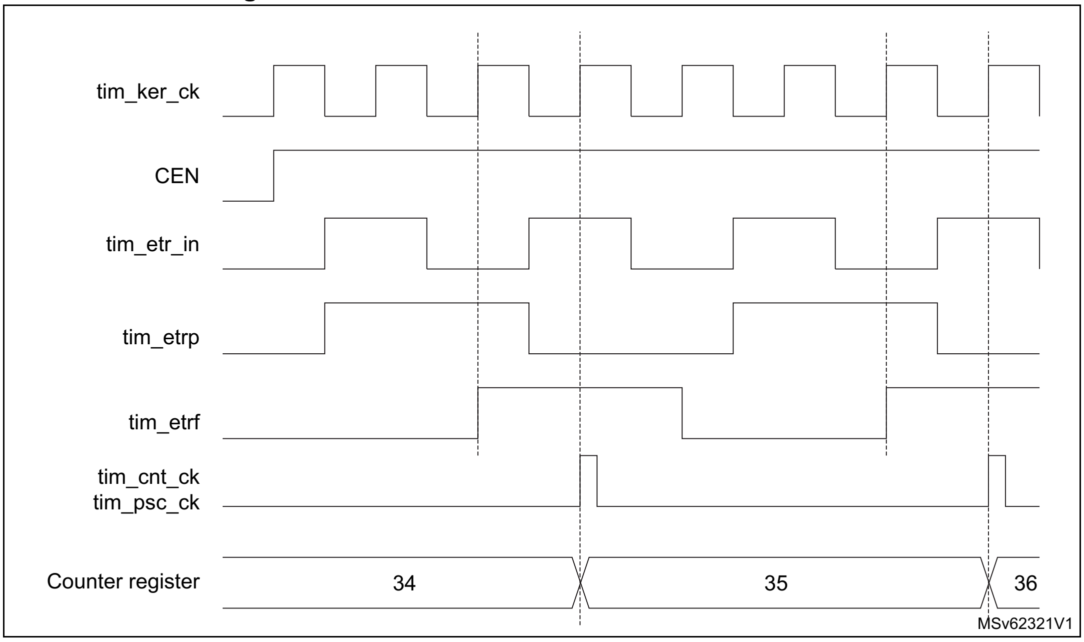
For example, to configure the upcounter to count each two rising edges on tim_etr_in, use the following procedure:

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 two 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 must apply a division of the external signal by a proper ETPS prescaler setting.

Figure 269. Control circuit in external clock mode 2



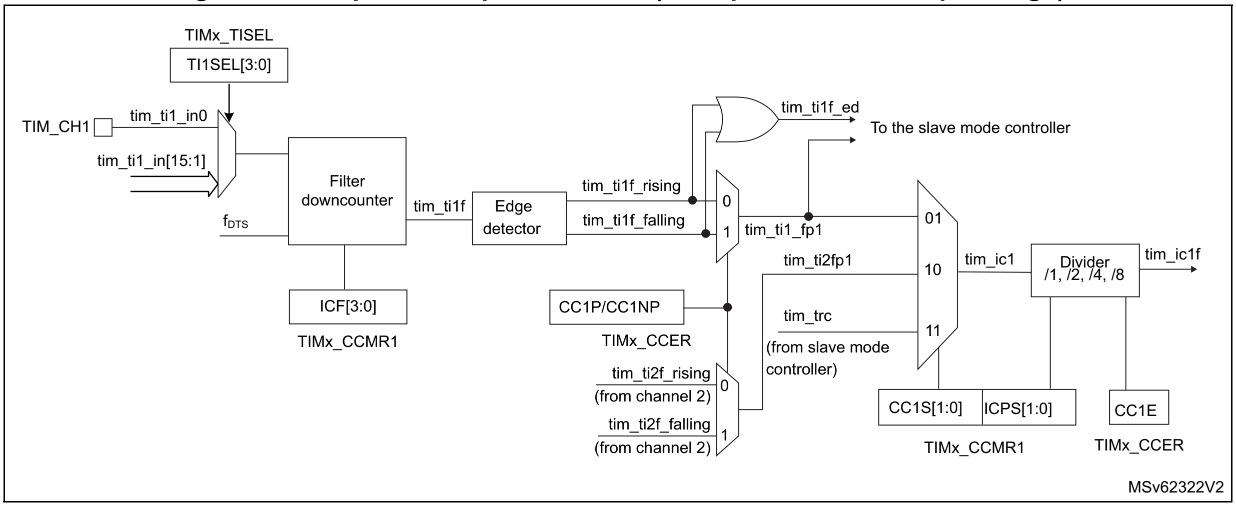
30.4.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) 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 270. 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 271. Capture/compare channel 1 main circuit

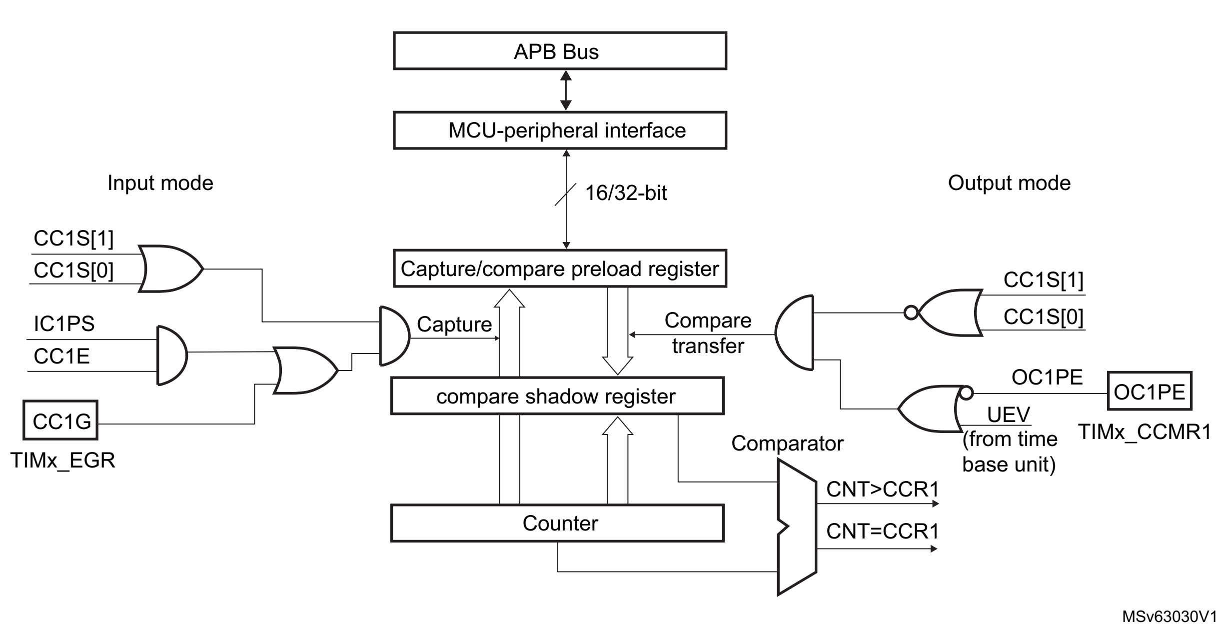
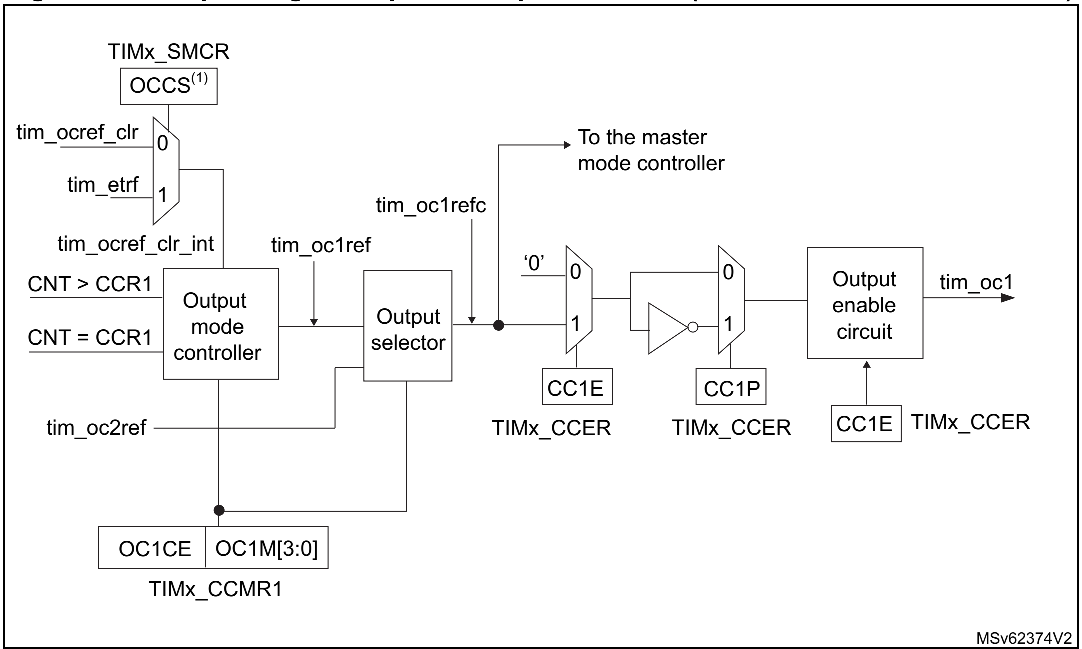


Figure 272. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4)



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.

30.4.7 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 overcapture 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 most five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on tim_ti1 when eight 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 bits to 000 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the capture is 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 may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

30.4.8 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).

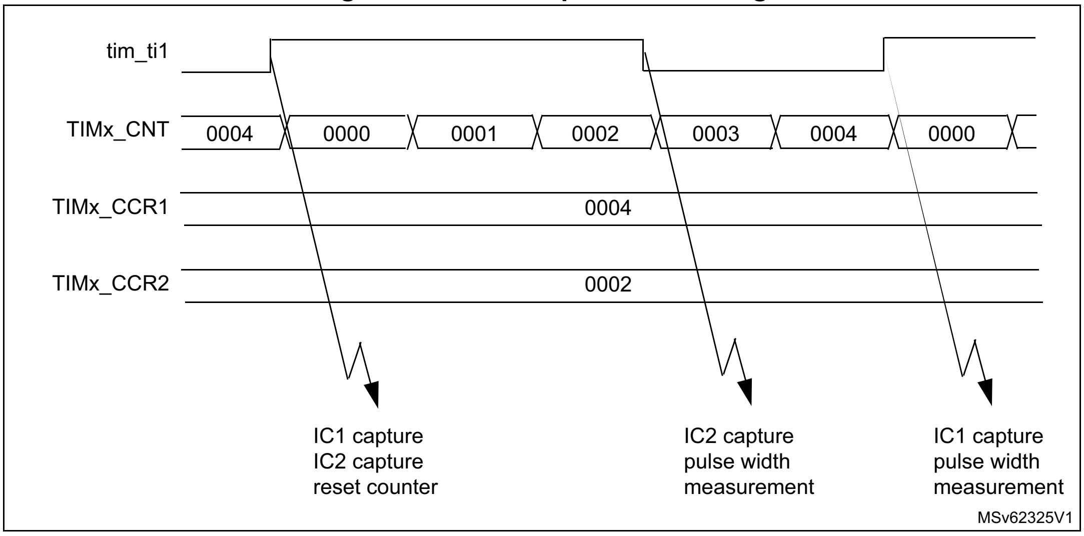
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tix input.
- These two 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 273. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only tim_ti1fp1 and tim_ti2fp2 are connected to the slave mode controller.

30.4.9 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, the user 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.

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.

30.4.10 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.

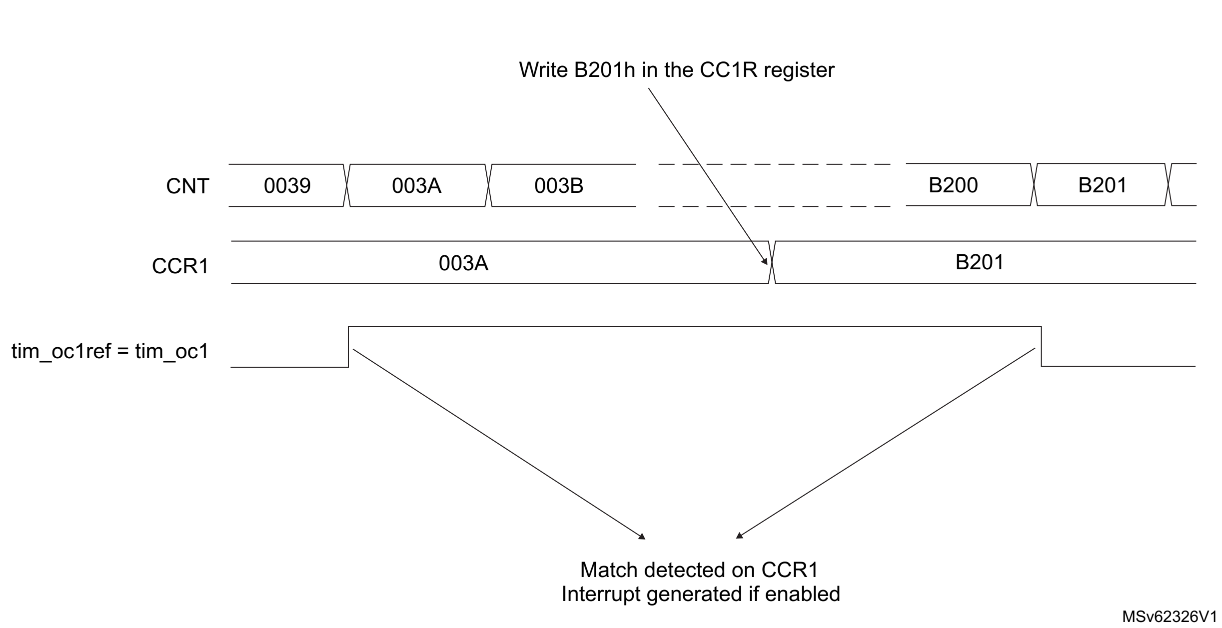
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

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 274*.

Figure 274. Output compare mode, toggle on tim_oc1



30.4.11 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 autoreload preload register (in up-counting 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.

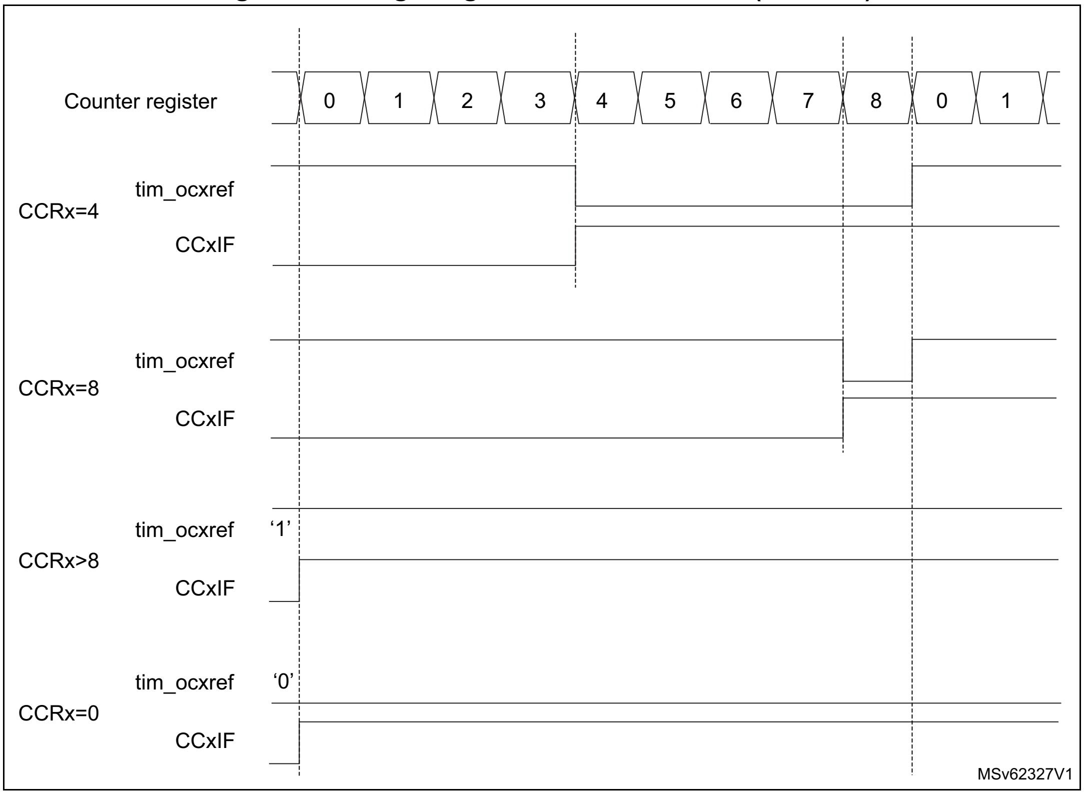
The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

PWM edge-aligned mode

- Up-counting configuration
- Up-counting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Up-counting mode*.

  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 autoreload 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 275* shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

Figure 275. Edge-aligned PWM waveforms (ARR = 8)



Down-counting configuration

- Down-counting is active when DIR bit in TIMx_CR1 register is high. Refer to *Down-counting mode*.

  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 autoreload value in TIMx_ARR, then tim_ocxref is held at 100%. PWM is not possible in this mode.

PWM center-aligned mode

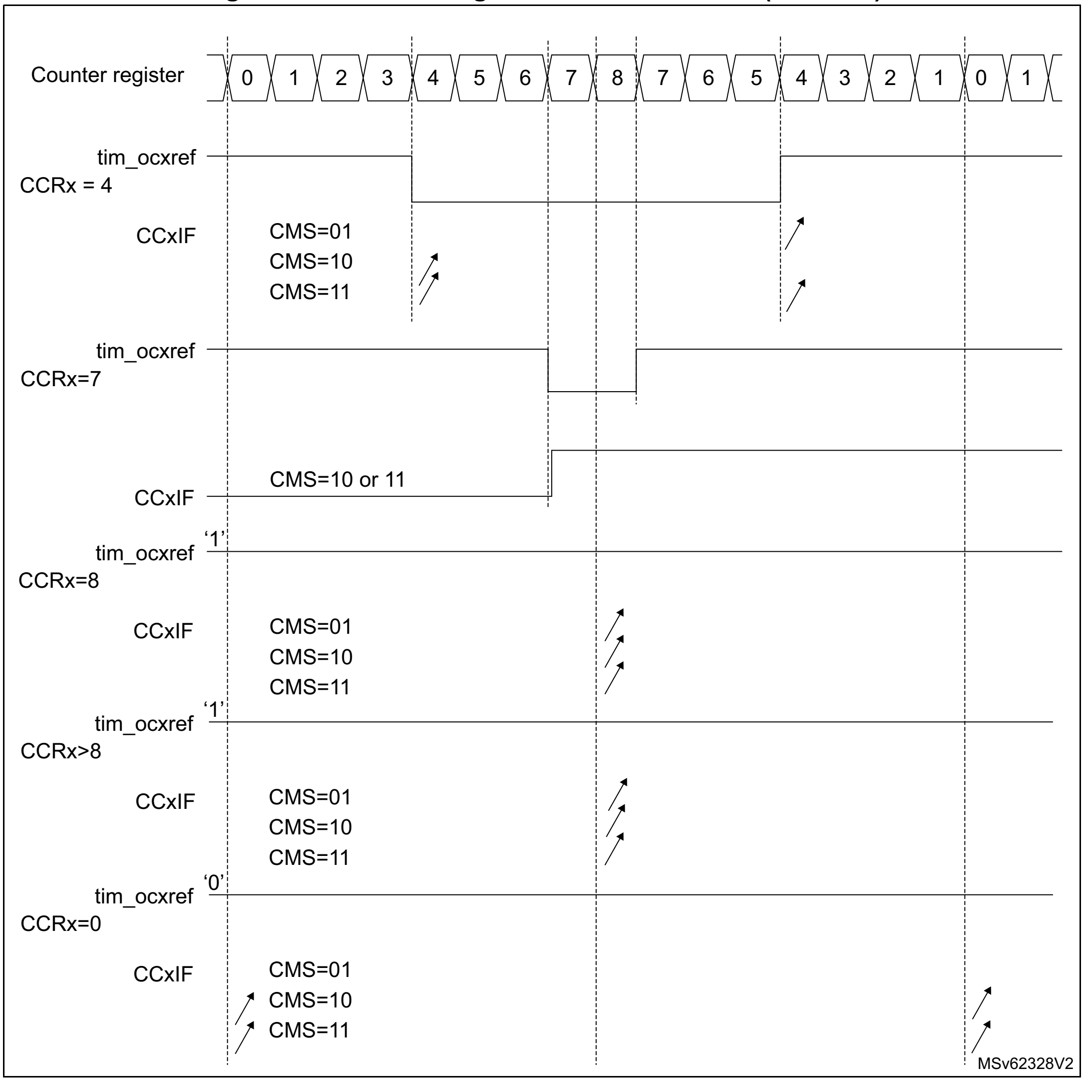
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending 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)*.

Figure 276 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 276. 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.

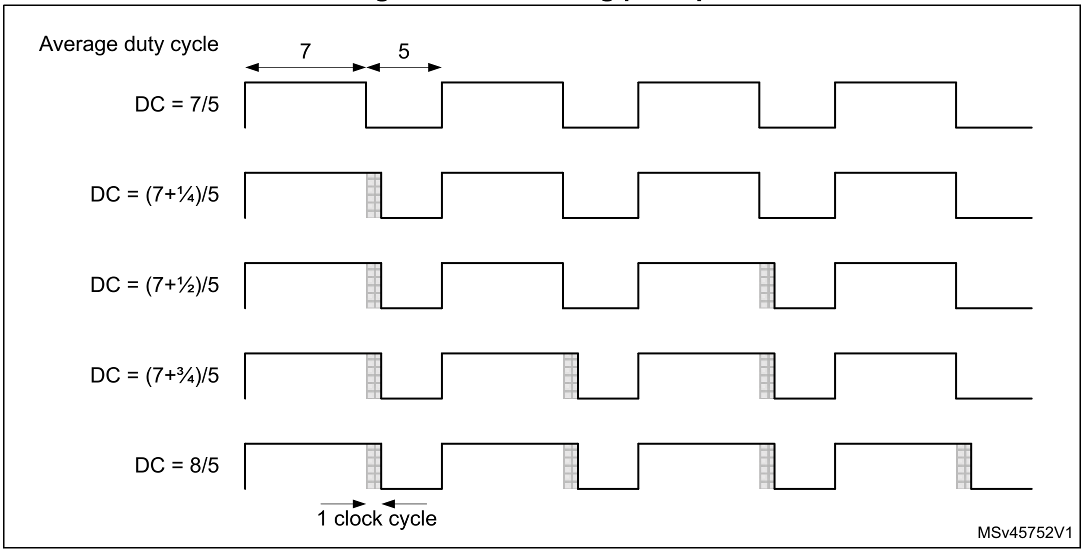
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the autoreload value is written in the counter (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, 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. *Figure 277* presents the dithering principle applied to four consecutive PWM cycles.

Figure 277. Dithering principle

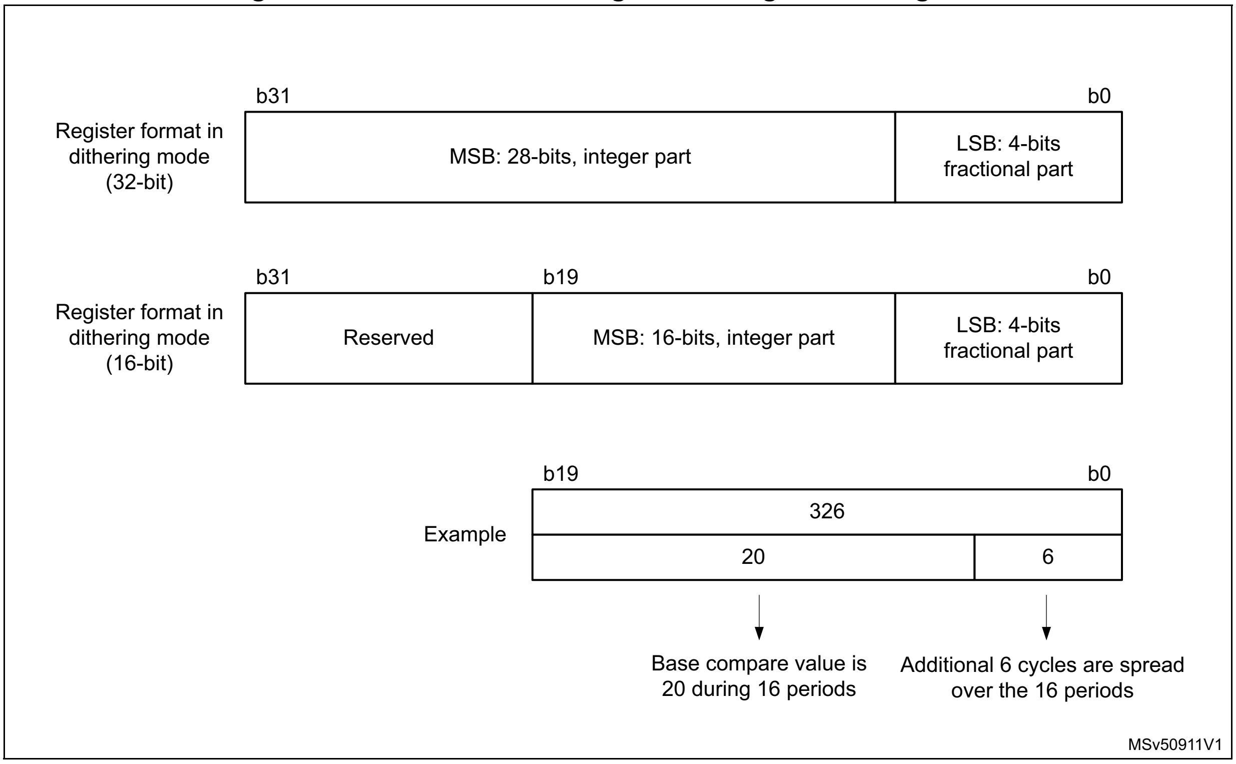


When the dithering mode is enabled, the register coding is changed as following (see *Figure 278* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted by four 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 DITHEN bit must be reset.*
3. *The CCIF flags must be cleared.*
4. *The CEN bit can be set (eventually with ARPE = 1).*

Figure 278. 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 *Figure 279* and *Figure 280*, the dithering mode is used to increase the PWM resolution.

Figure 279. PWM resolution vs frequency (16-bit mode)

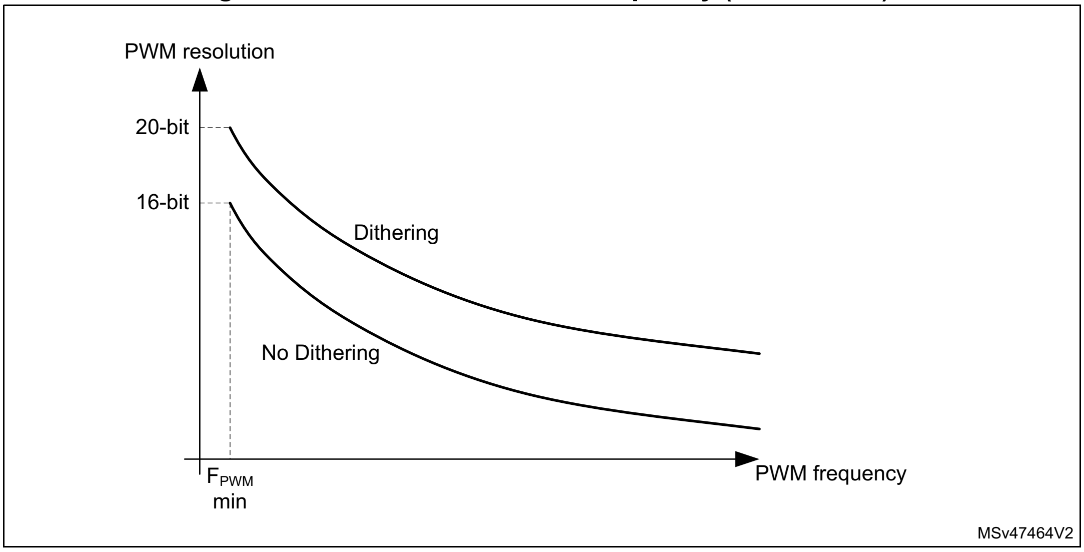
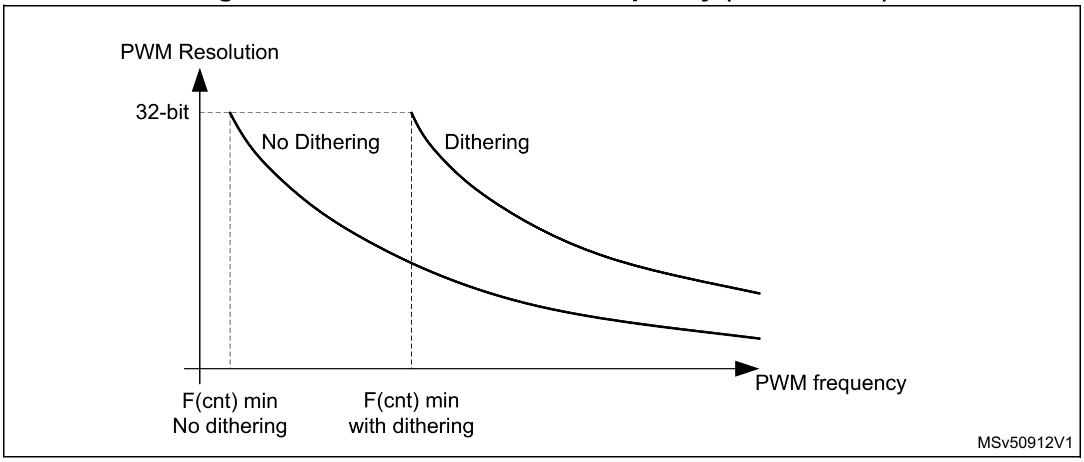
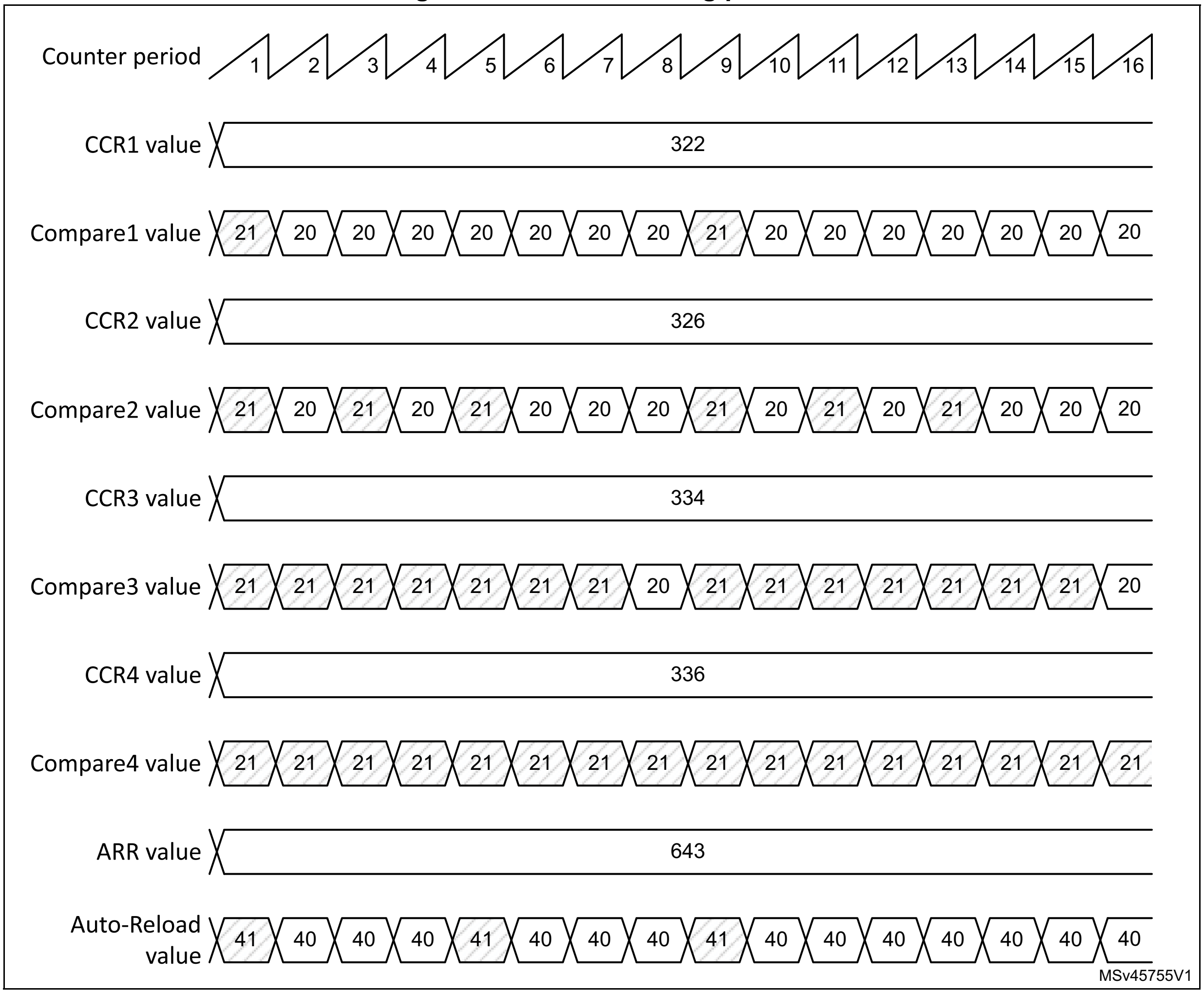


Figure 280. PWM resolution vs frequency (32-bit mode)



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 281*.

Figure 281. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 286*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 286. 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 eight consecutive PWM periods, considering the up and down-counting phases as shown in *Figure 282*.

Figure 282. Dithering effect on duty cycle in center-aligned PWM mode

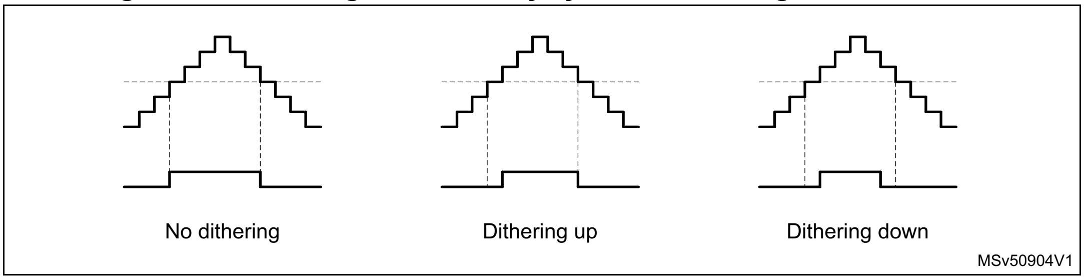


Table 287 shows how the dithering pattern is added in center-aligned PWM mode.

Table 287. 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	-

30.4.12 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.

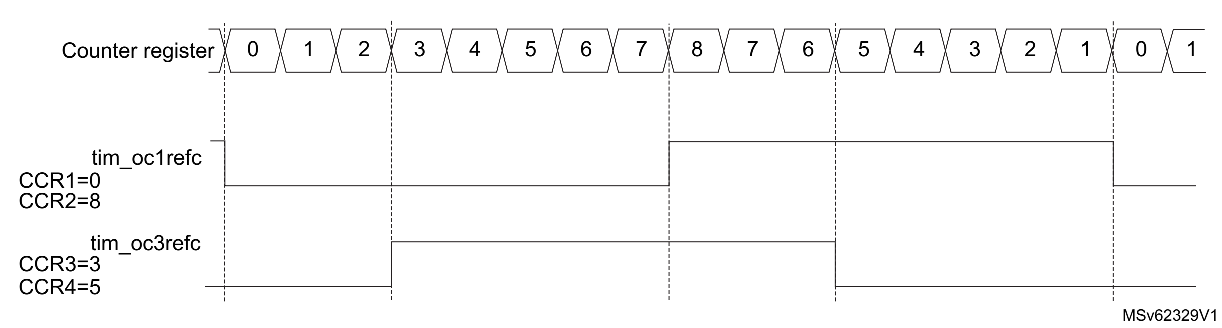
Asymmetric PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1110 (Asymmetric PWM mode 1) or 1111 (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three 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 283 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 283. Generation of two phase-shifted PWM signals with 50% duty cycle



30.4.13 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.

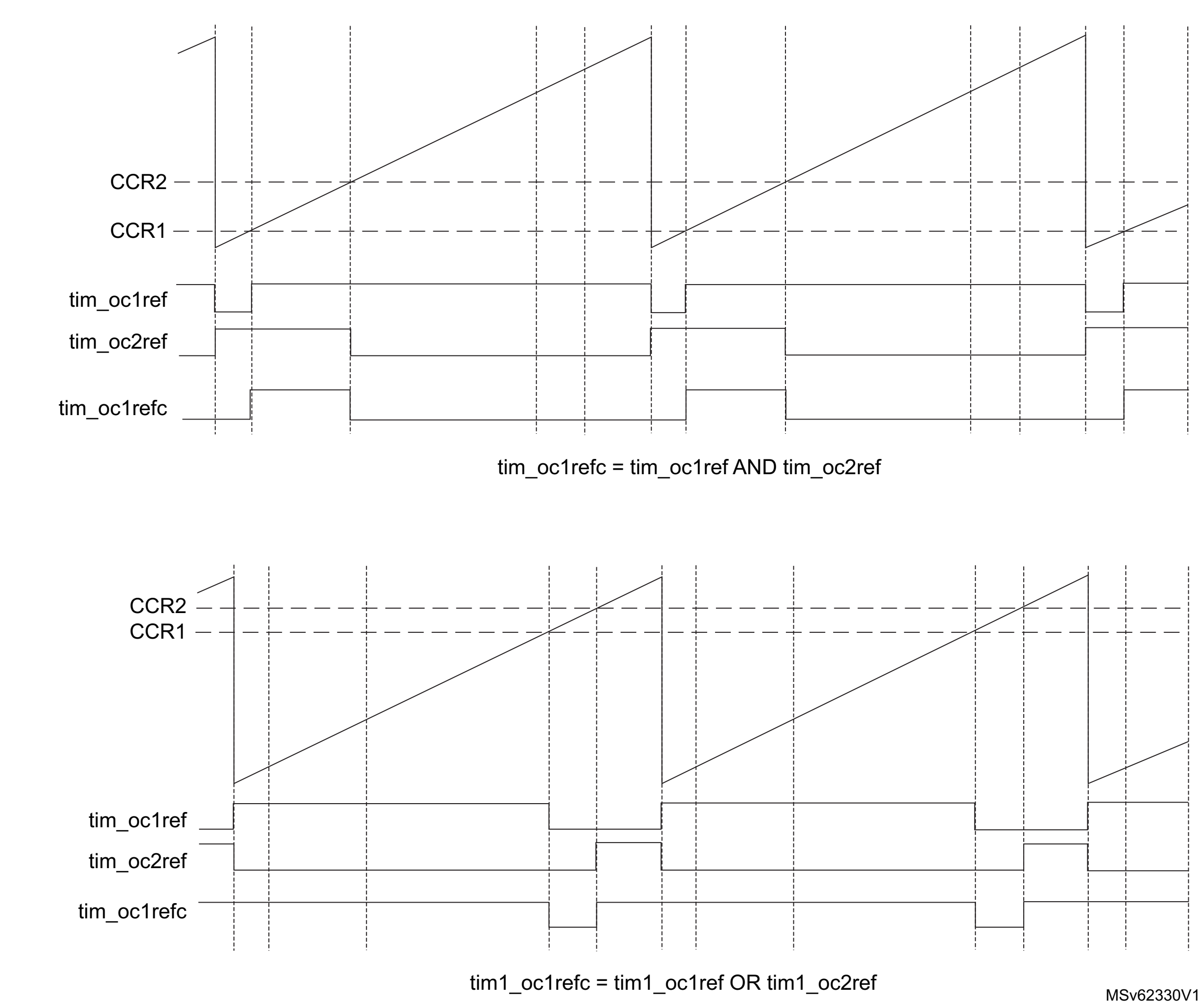
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Figure 284 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 284. Combined PWM mode on channels 1 and 3



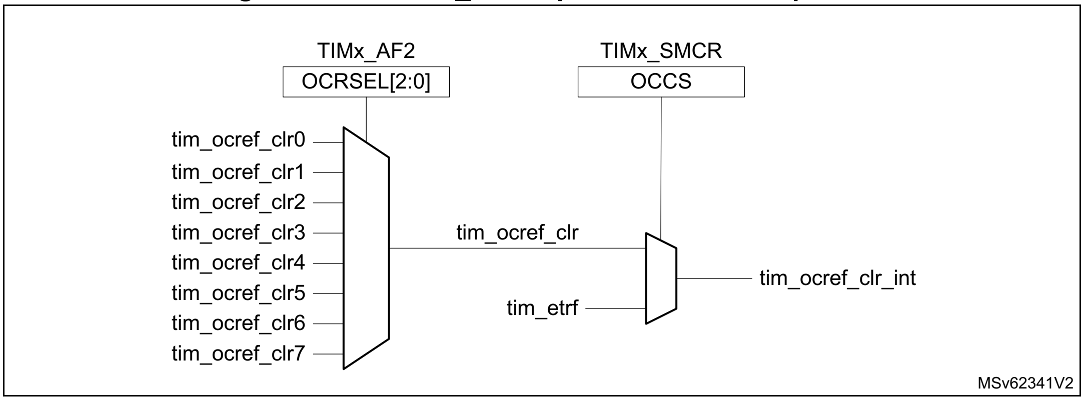
30.4.14 Clearing the tim_ocxref signal on an external event

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 transition to the active state, on the following PWM cycle. 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 30.3: TIM2/TIM3 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 285*.

Figure 285. OCREF_CLR input selection multiplexer



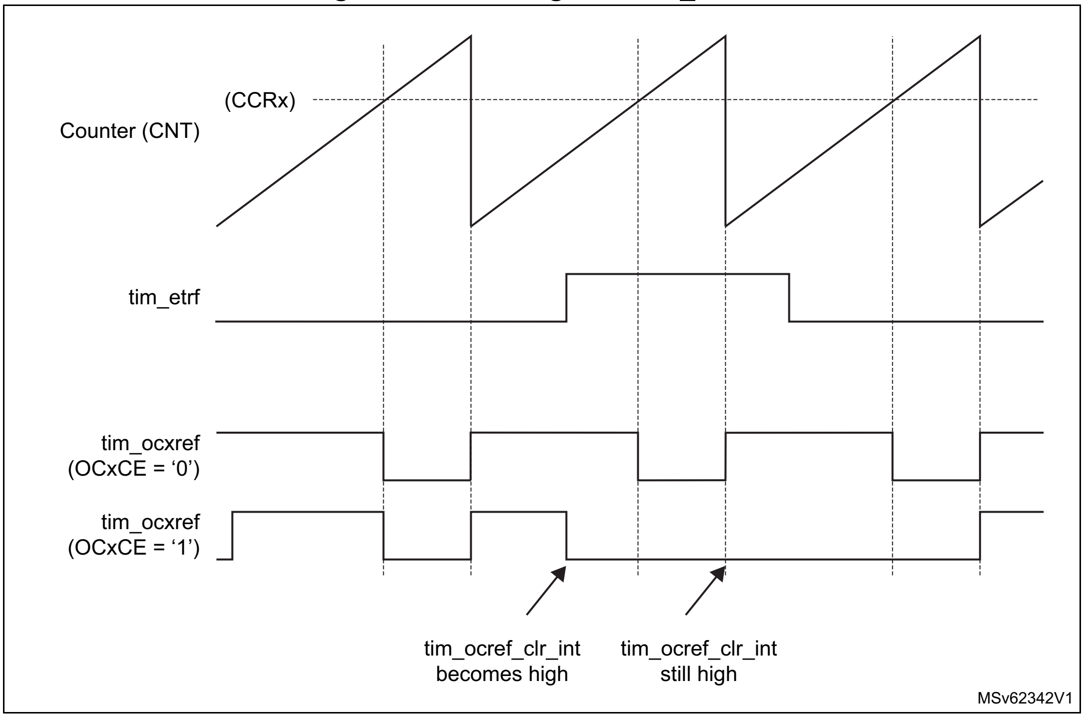
If the OCREF clear selection feature is not implemented, the tim_ocref_clr_int input is directly connected to the tim_etrf input.

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 must 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 286 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 286. 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.

30.4.15 One-pulse mode

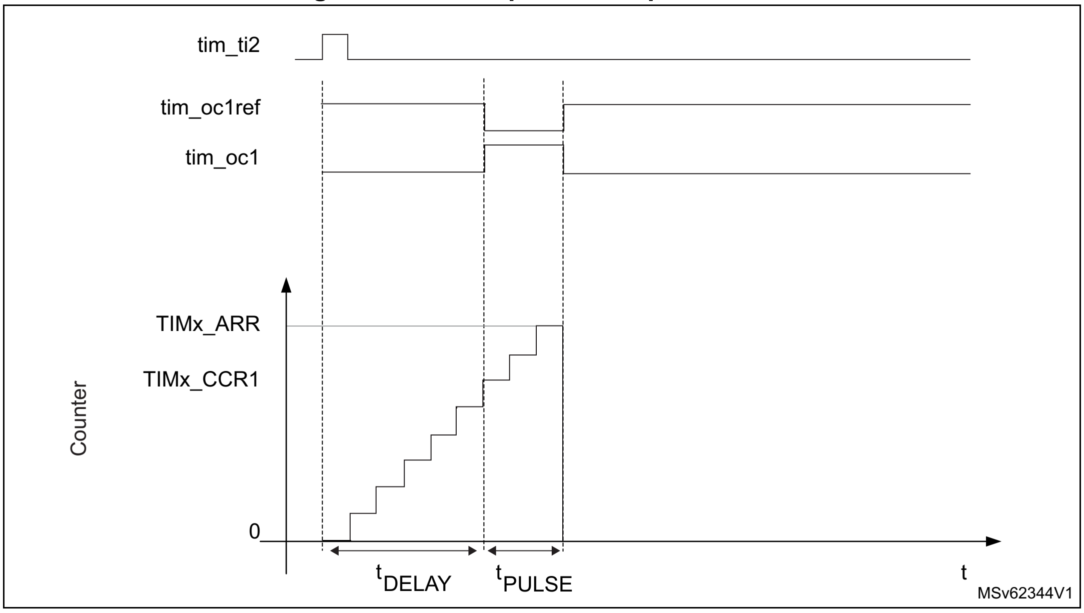
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

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 287. Example of One-pulse mode



For example if the user wants 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.

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 autoreload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Suppose the user wants 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 autoreload 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 autoreload 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 this example, the DIR and CMS bits in the TIMx_CR1 register must be low.

Since only one pulse (Single mode) is needed, a one 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 autoreload 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.

30.4.16 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 30.4.15*:

- 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.

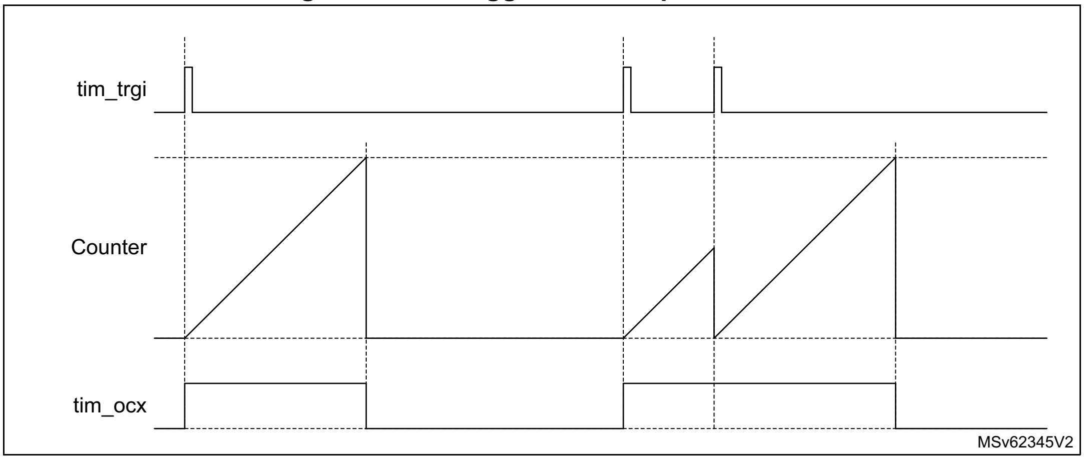
If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode CCRx must be above or equal to ARR.

Note: In Retriggerable one-pulse mode, the CCxIF flag is not significant.

The OCxM[3:0] and SMS[3:0] bitfields are split into two parts for compatibility reasons, the most significant bit is not contiguous with the three 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 288. Retriggerable one-pulse mode

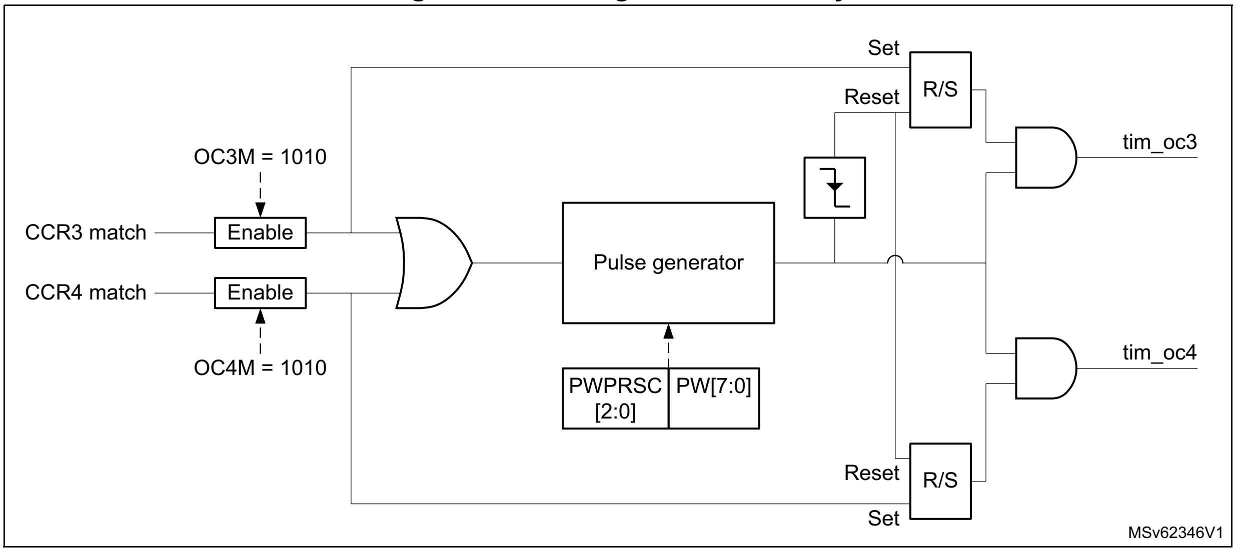


30.4.17 Pulse on compare mode

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.

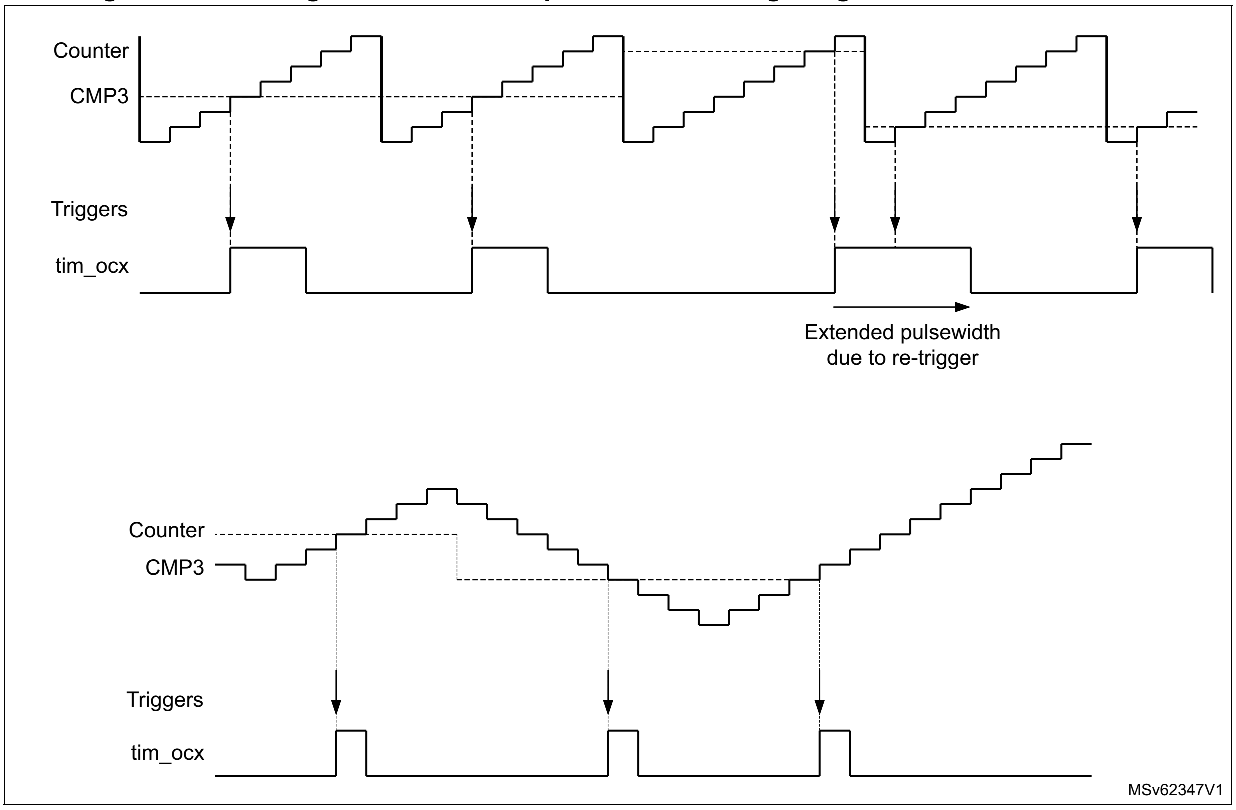
This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on *Figure 289*.

Figure 289. Pulse generator circuitry



[Figure 290](#) shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 290. 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] bitfields 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}

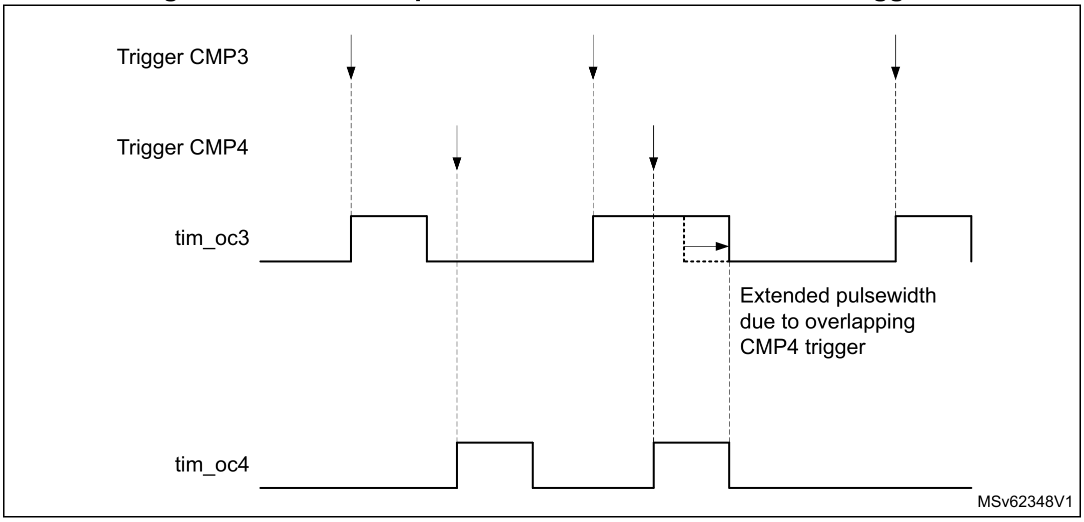
where $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim\_ker\_ck}$.

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 first arriving trigger is extended (because of the retrigger), while the pulse width of the last arriving trigger is correct (as shown on Figure 291).

Figure 291. Extended pulse width in case of concurrent triggers



30.4.18 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 288*. 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 autoreload 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 corresponds 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 288. 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 indicates the mechanical zero position, can be connected to the external trigger input and trigger a counter reset.

Figure 292 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_CCMR1 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 = 0011 (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIMx_CR1 register, counter is enabled).

Figure 292. Example of counter operation in encoder interface mode

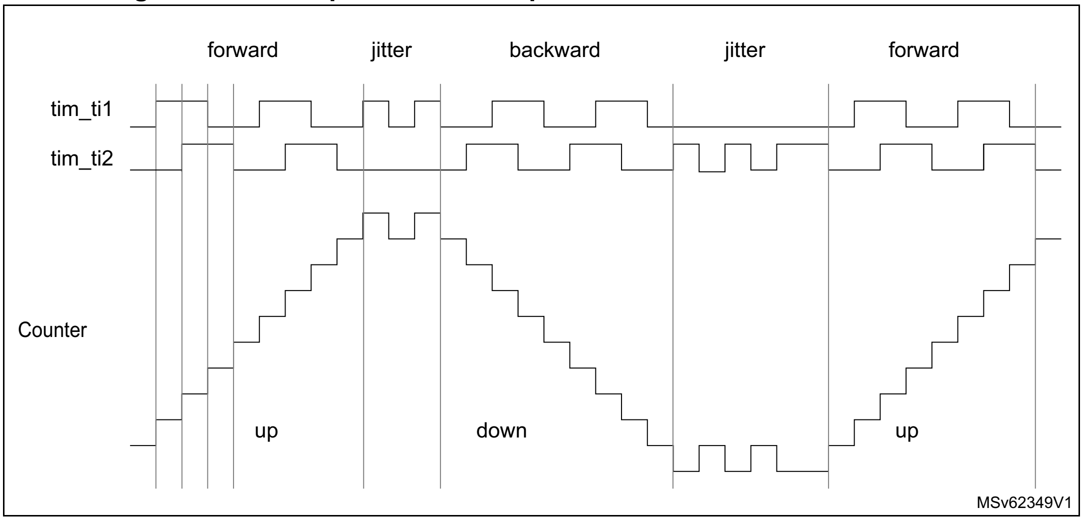


Figure 293 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 293. Example of encoder interface mode with tim_ti1fp1 polarity inverted

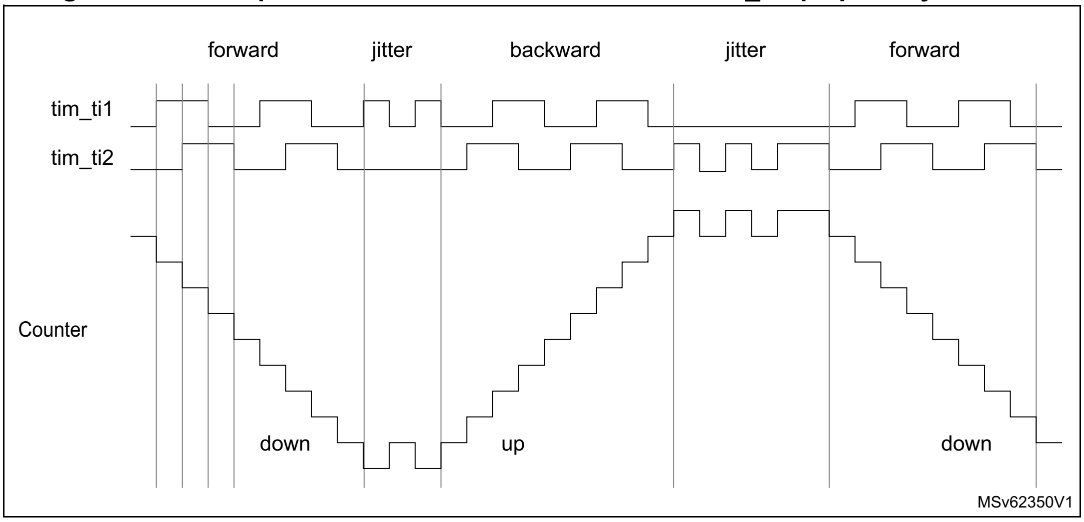
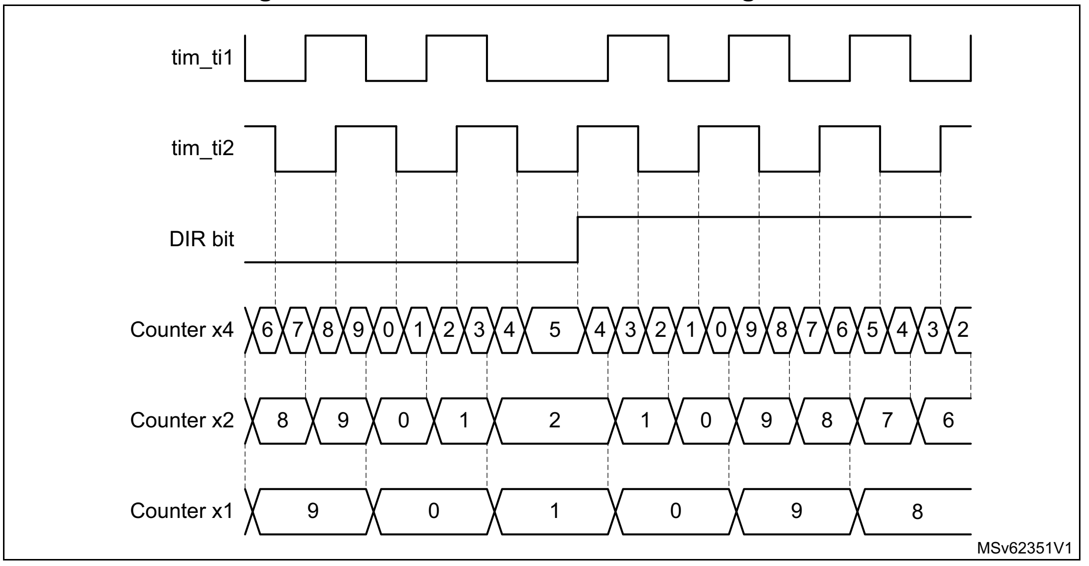


Figure 294 shows the timer counter value during a speed reversal, for various counting modes.

Figure 294. 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 for other types of encoders.

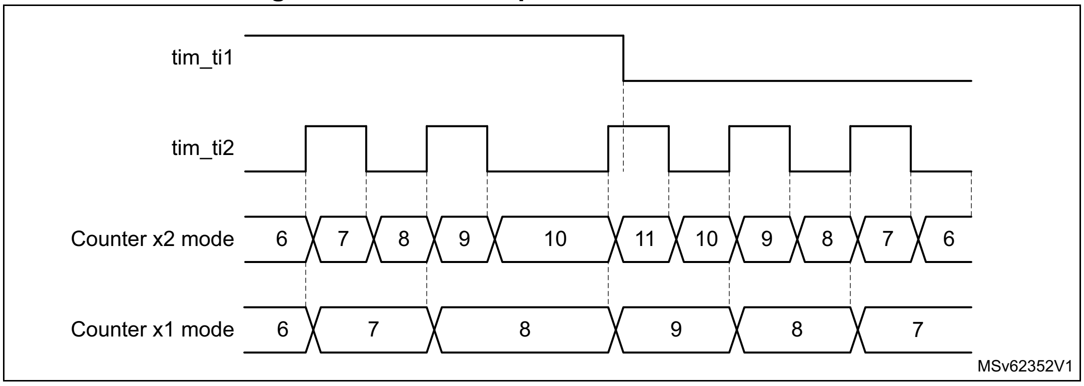
In the "clock plus direction" mode shown on *Figure 295*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

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 295. Direction plus clock encoder mode



Directional clock encoder mode

In the “directional clock” mode on *Figure 296*, 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 lines. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 296*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 297*).
- 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 296*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 297*).

Figure 296. Directional clock encoder mode (CC1P = CC2P = 0)

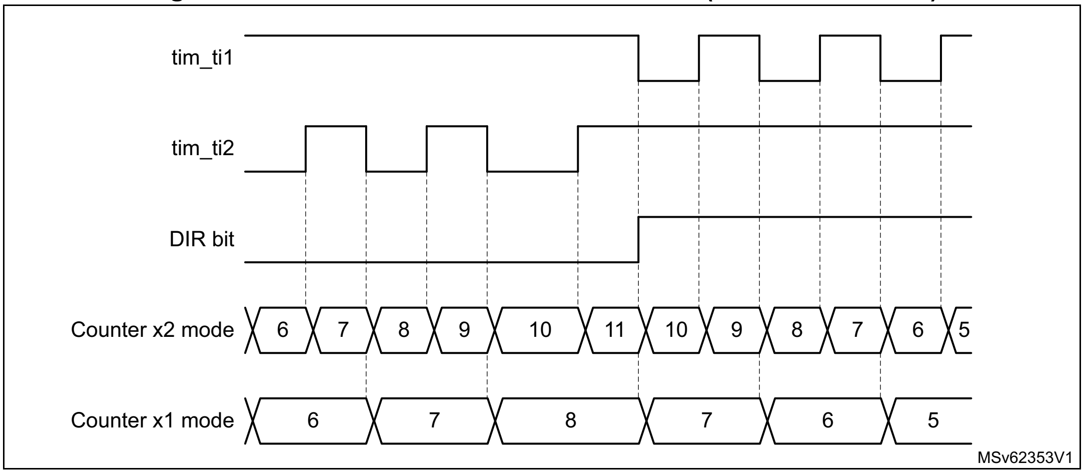


Figure 297. Directional clock encoder mode (CC1P = CC2P = 1)

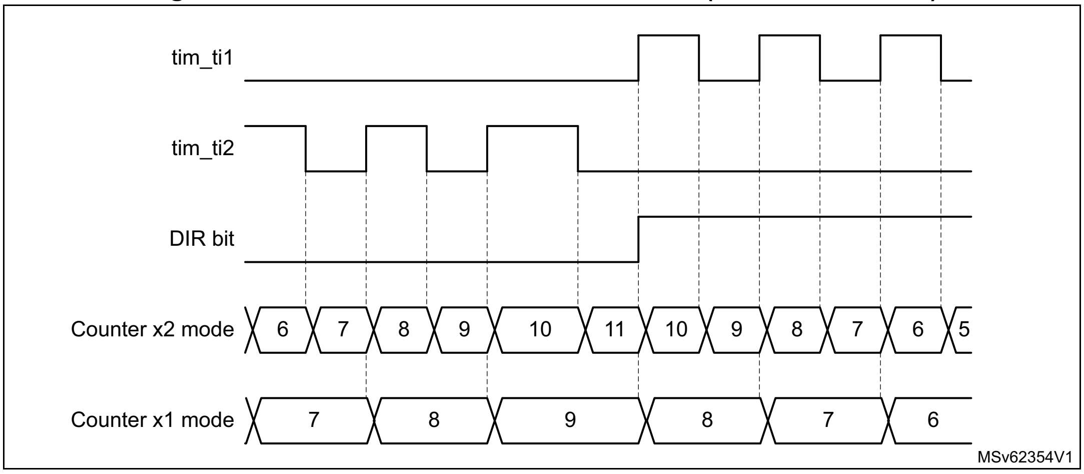


Table 289 details how the directional clock mode operates, for any input transition.

Table 289. 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

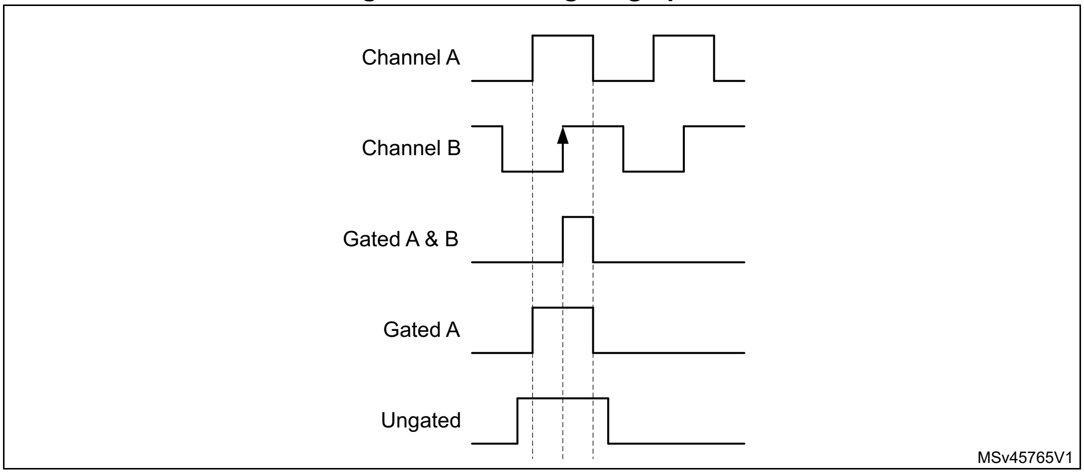
The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

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.

Available encoders are proposed with several options for index pulse conditioning, as per *Figure 298*:

- 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.

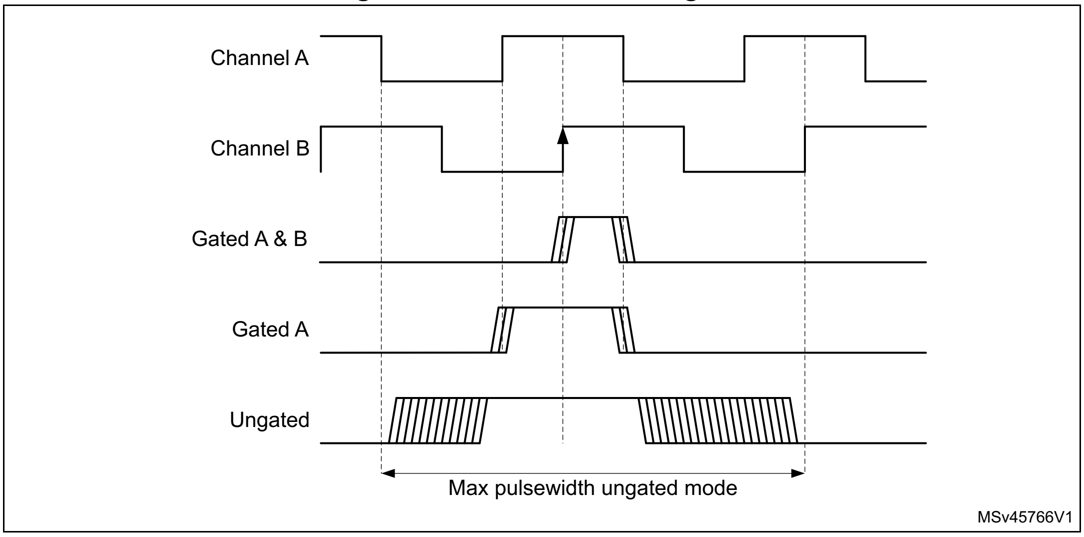
Figure 298. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as shown on *Figure 299*.

In ungated mode, the signal must be strictly below two encoder periods. If the pulse width is greater or equal to two encoder period, the counter is reset multiple times.

Figure 299. Jittered Index signals



The timer supports the three gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (for example 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. *Figure 300* shows at which position is the index generated, for a simplistic example (an encoder providing four edges par mechanical rotation).

Figure 300. Index generation for IPOS[1:0] = 11

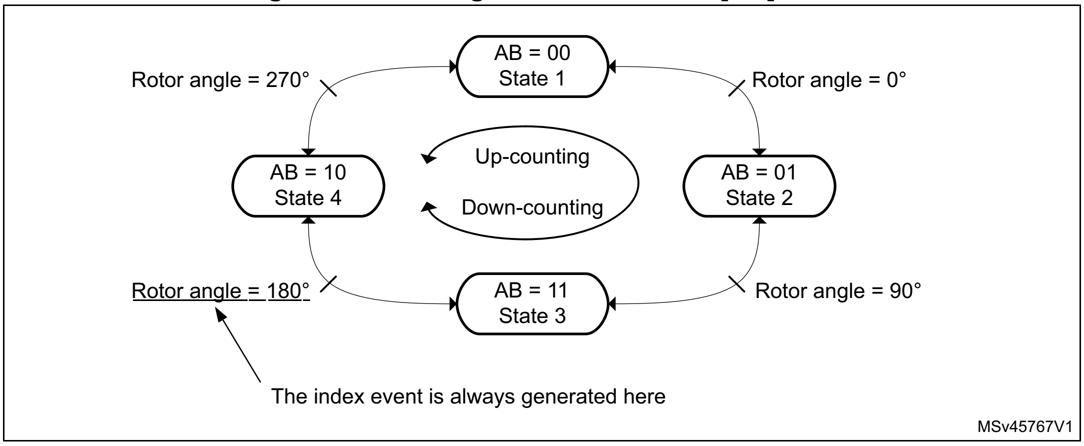


Figure 301 presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- 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 301. Counter reading with index gated on channel A (IPOS[1:0] = 11)

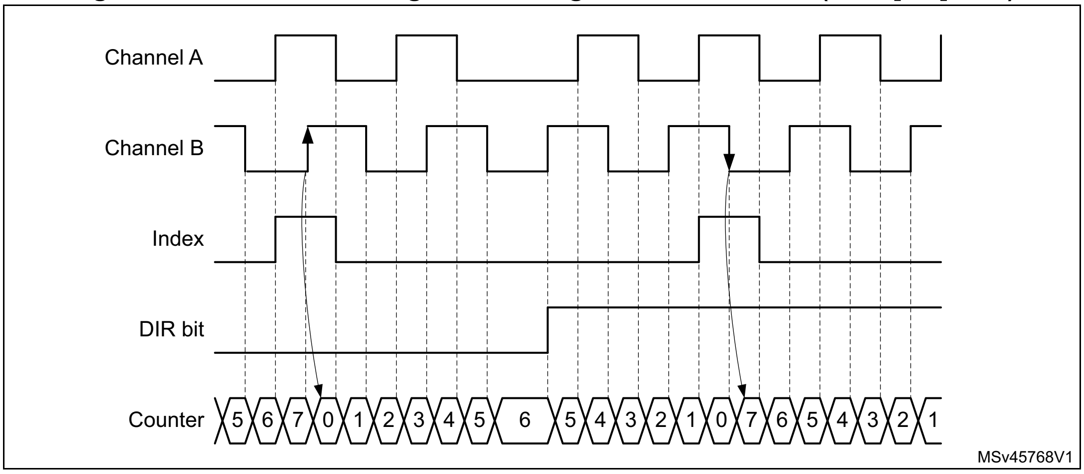


Figure 302 presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 302. Counter reading with index ungated (IPOS[1:0] = 00)

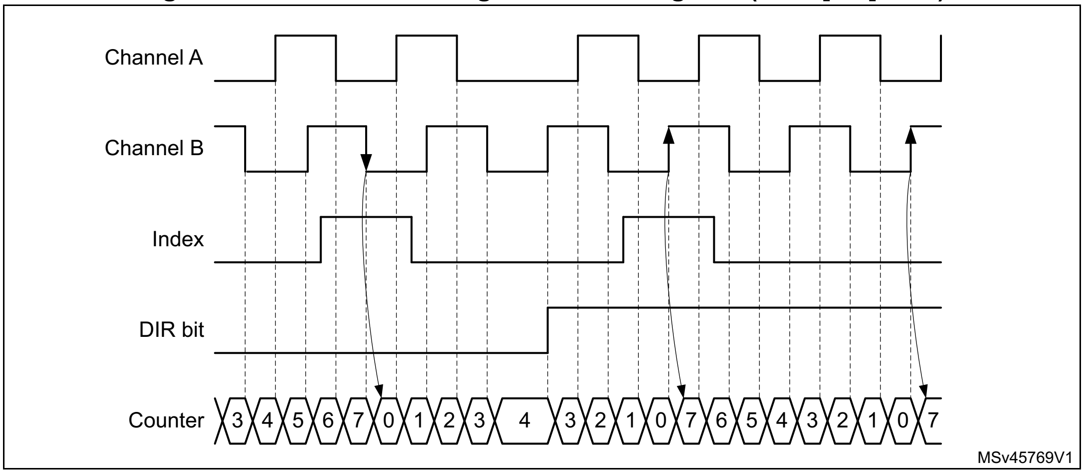


Figure 303 shows how the gated on A & B mode is handled, for various pulse alignment scenarios. The arrows are indicating on which transition is the index event generated.

Figure 303. Counter reading with index gated on channel A and B

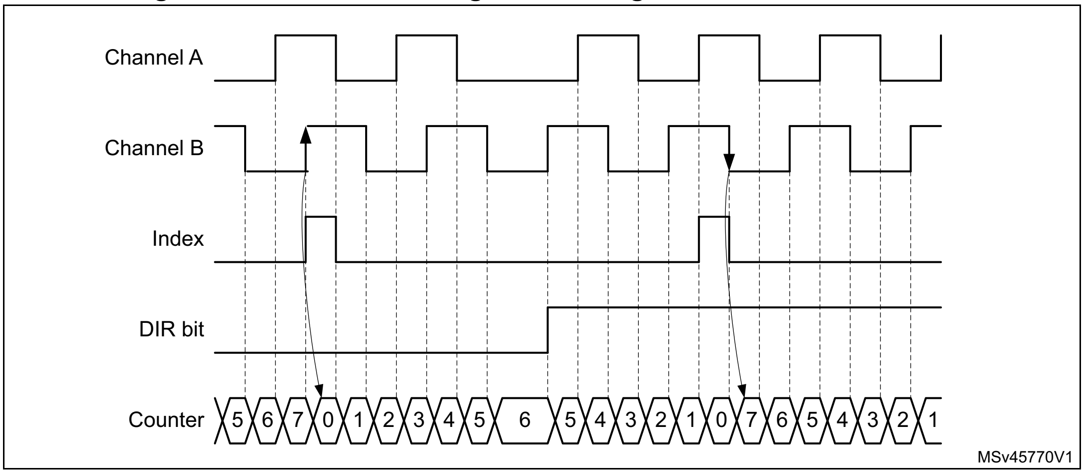


Figure 304 and *Figure 305* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 304. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

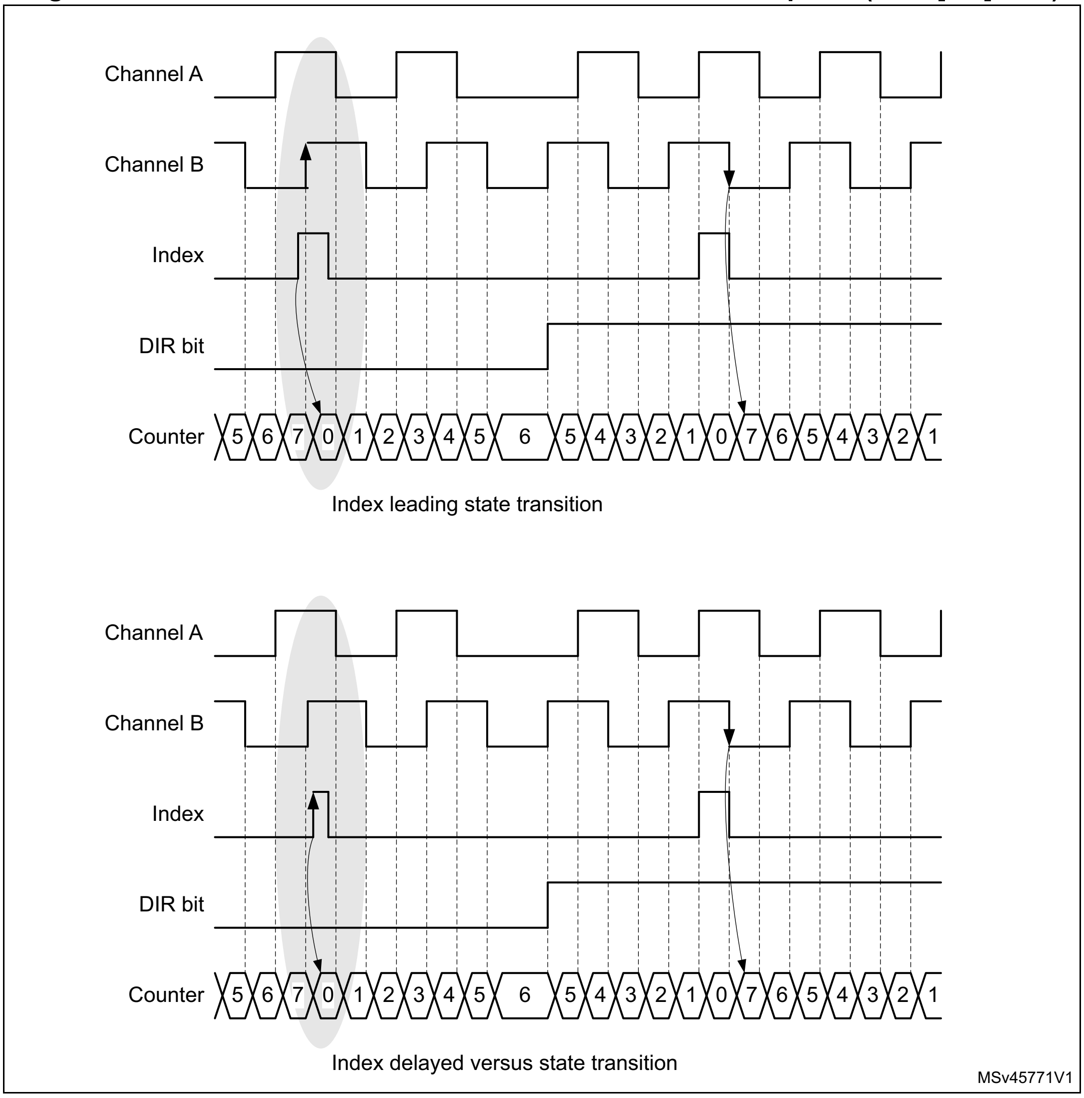


Figure 305. Counter reset Narrow index pulse (closer view, ARR = 0x07)

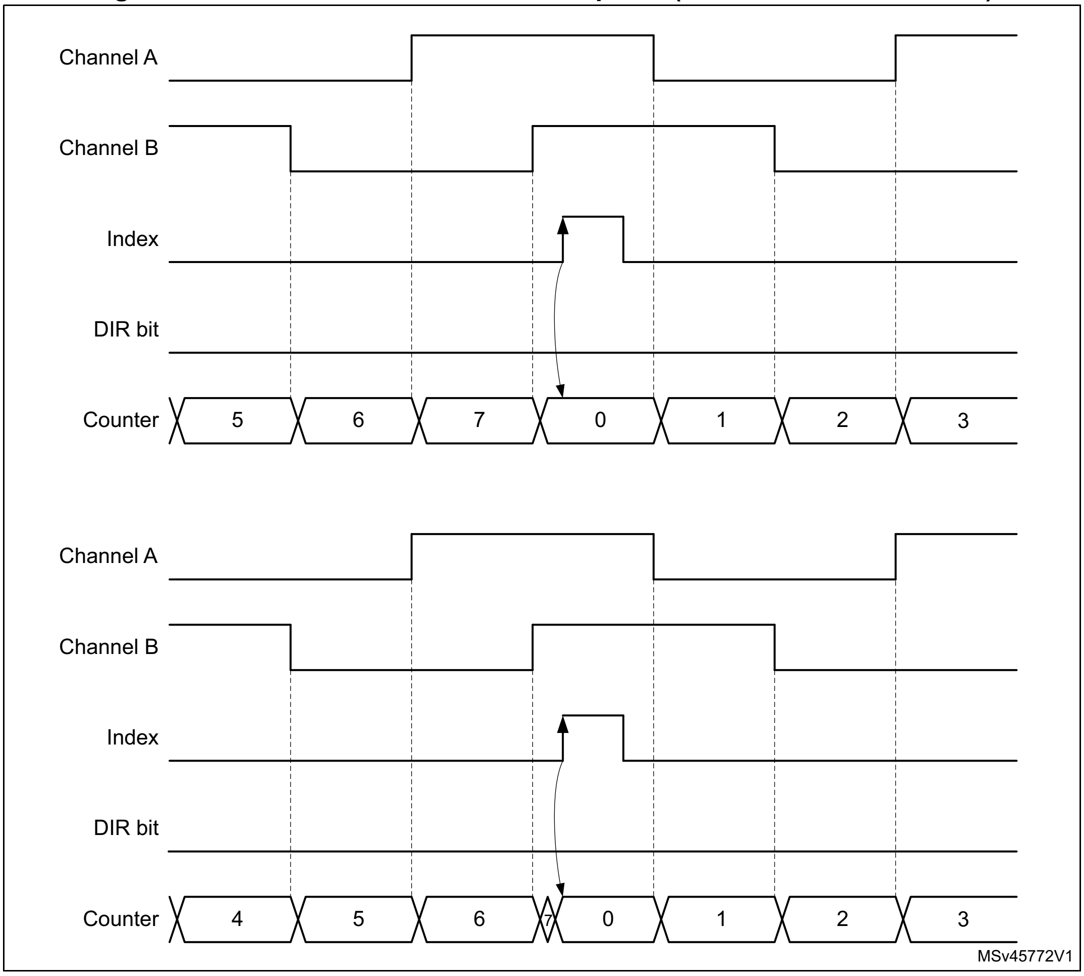
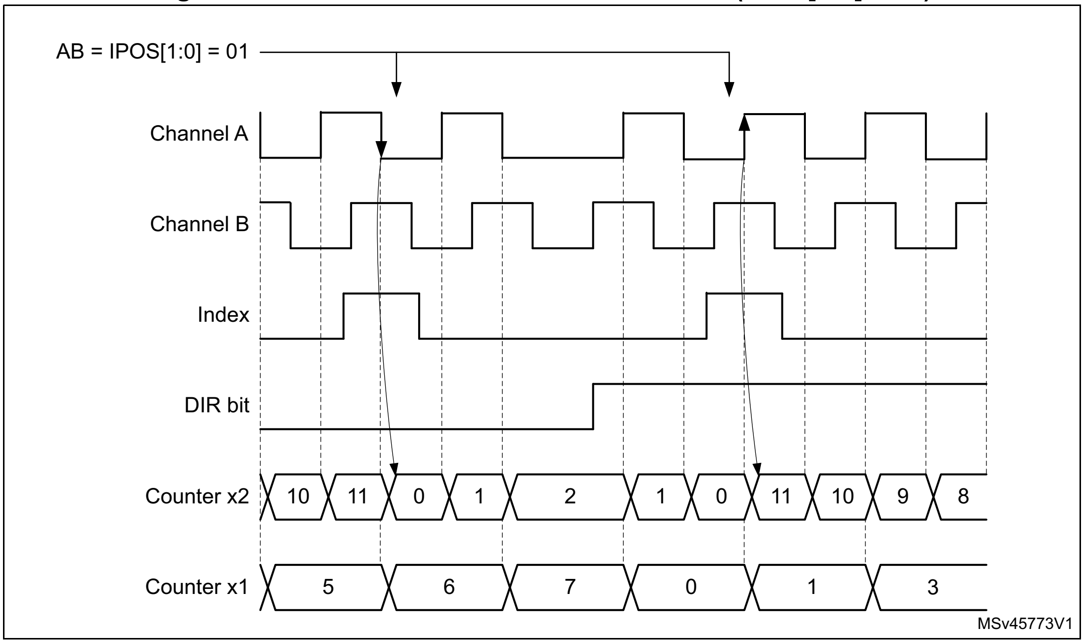


Figure 306 shows how the index is managed in x1 and x2 modes.

Figure 306. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)

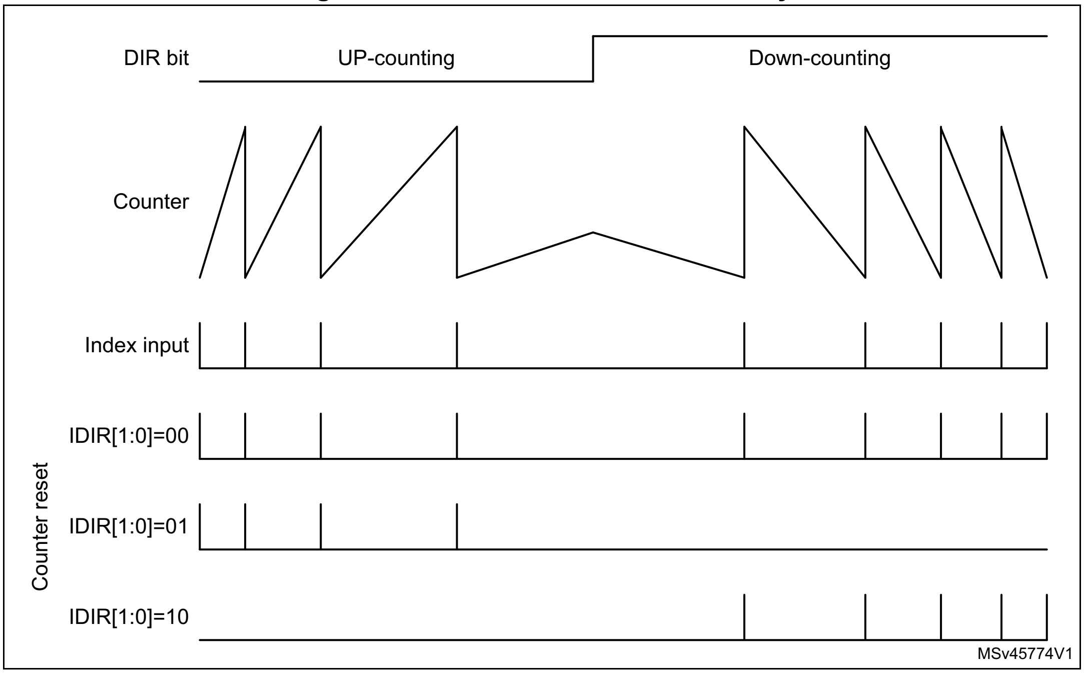


Directional index sensitivity

The IDIR[1:0] bitfield in the TIMx_ECR register allows the index to be active only in a selected counting direction.

Figure 307 shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

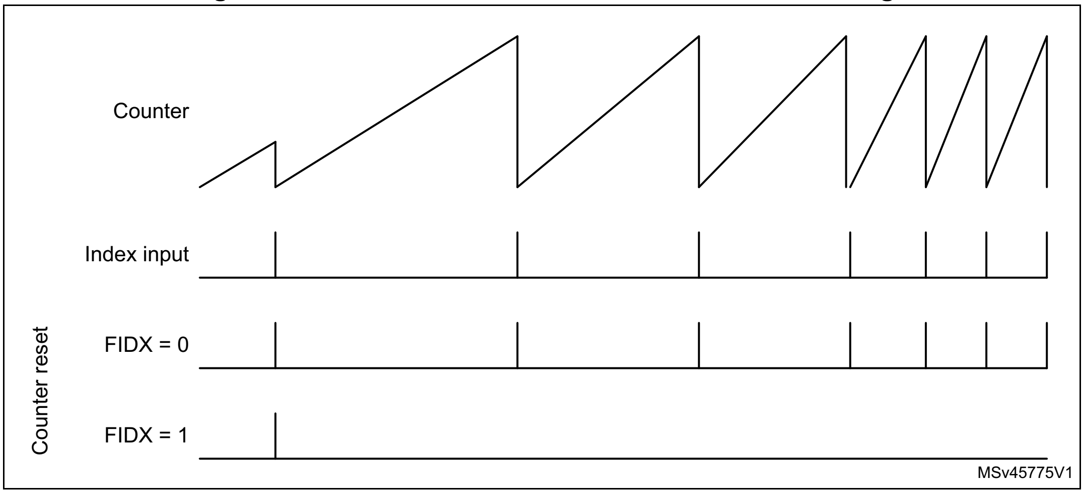
Figure 307. Directional index sensitivity



Special first index event management

The FIDX bit in the TIMx_ECR register allows the index to be taken only once, as shown on *Figure 308*. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be rearmed by writing the FIDX bit to 0 and setting it again to 1.

Figure 308. Counter reset as function of FIDX bit setting



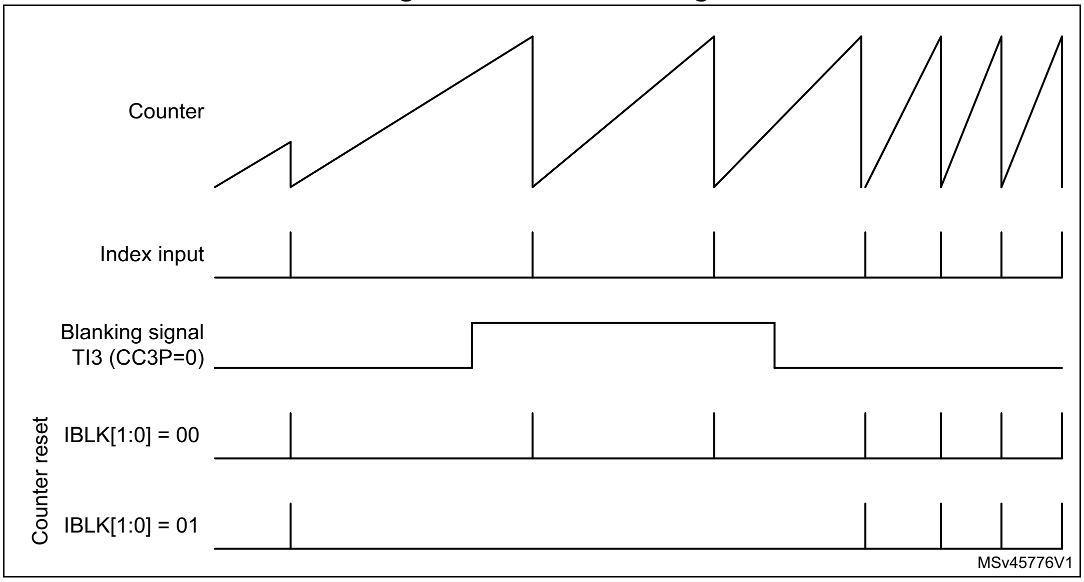
Index blanking

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 *Figure 309*.

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 309. Index blanking



Index management in nonquadrature mode

Figure 310 and *Figure 311* 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 310. Index behavior in clock + direction mode, IPOS[0] = 1

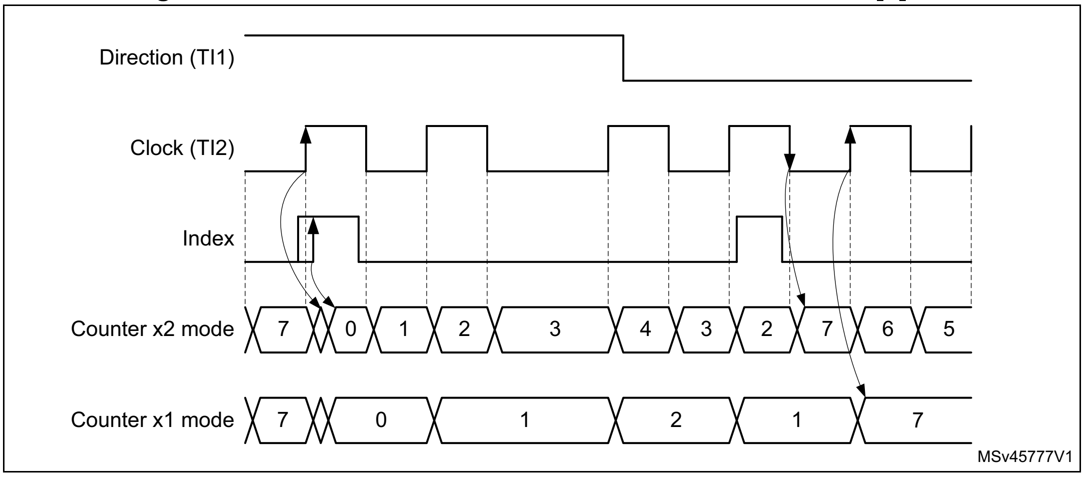
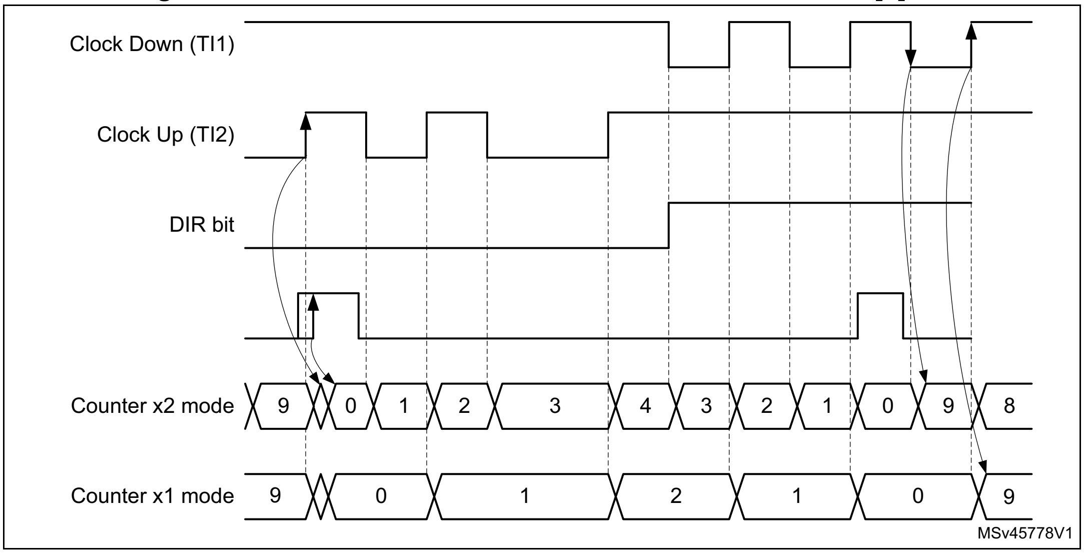


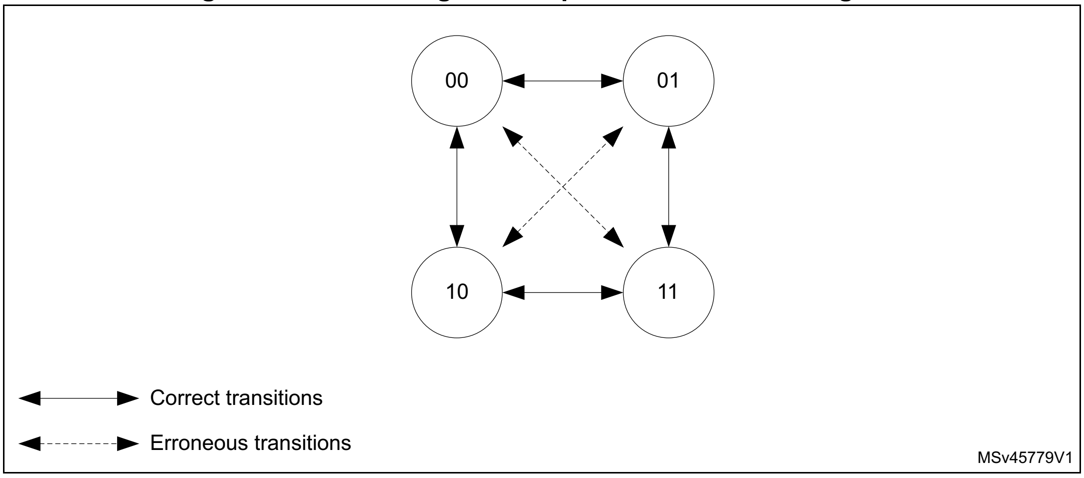
Figure 311. Index behavior in directional clock mode, IPOS[0] = 1



Encoder error management

For encoder configurations where two quadrature signals are available, it is possible to detect transition errors. The reading on the two inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on *Figure 312*. 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 312. State diagram for quadrature encoded signals



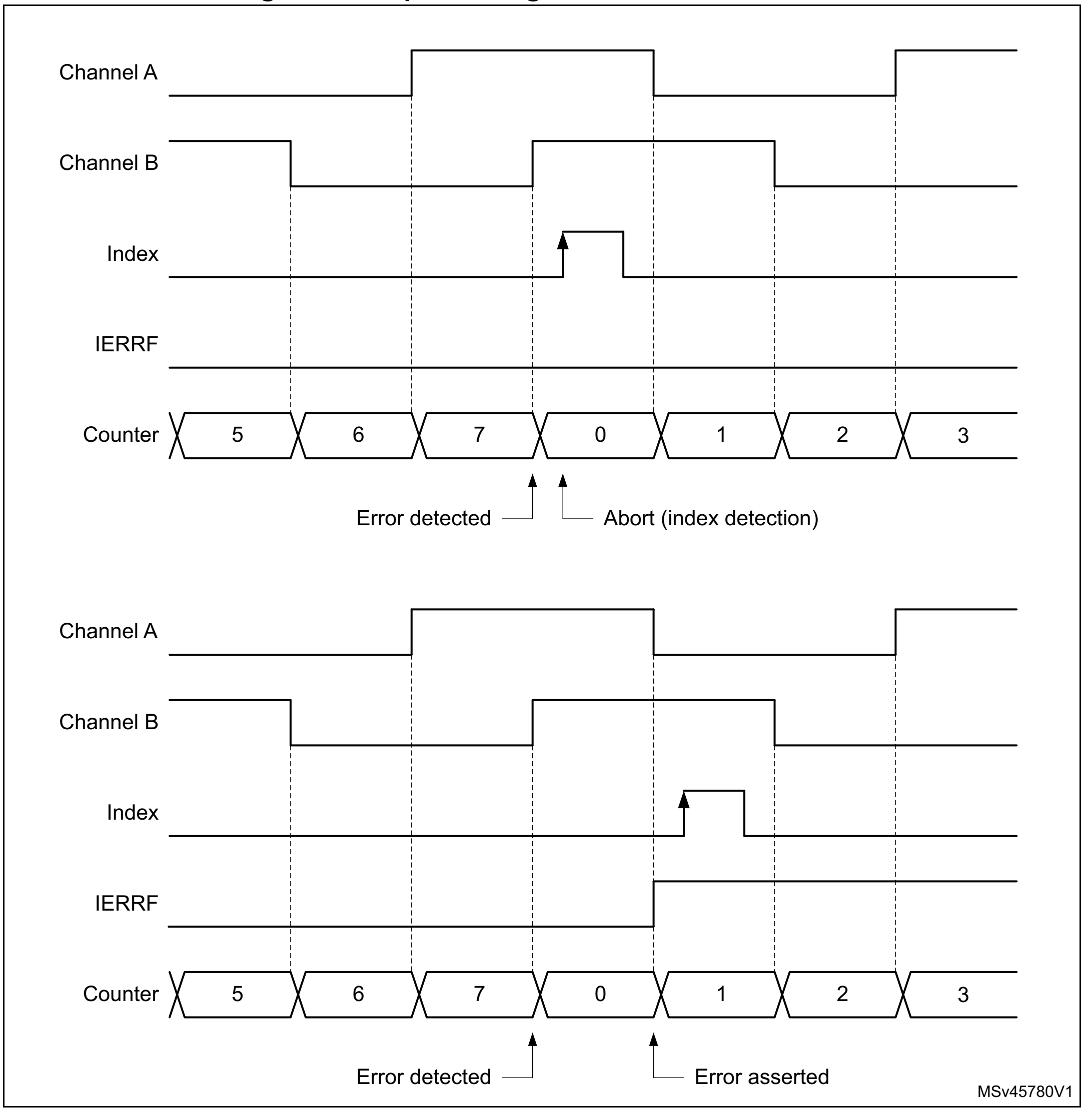
For encoder having an index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The index signal resets the counter every 4xN clock periods.

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.

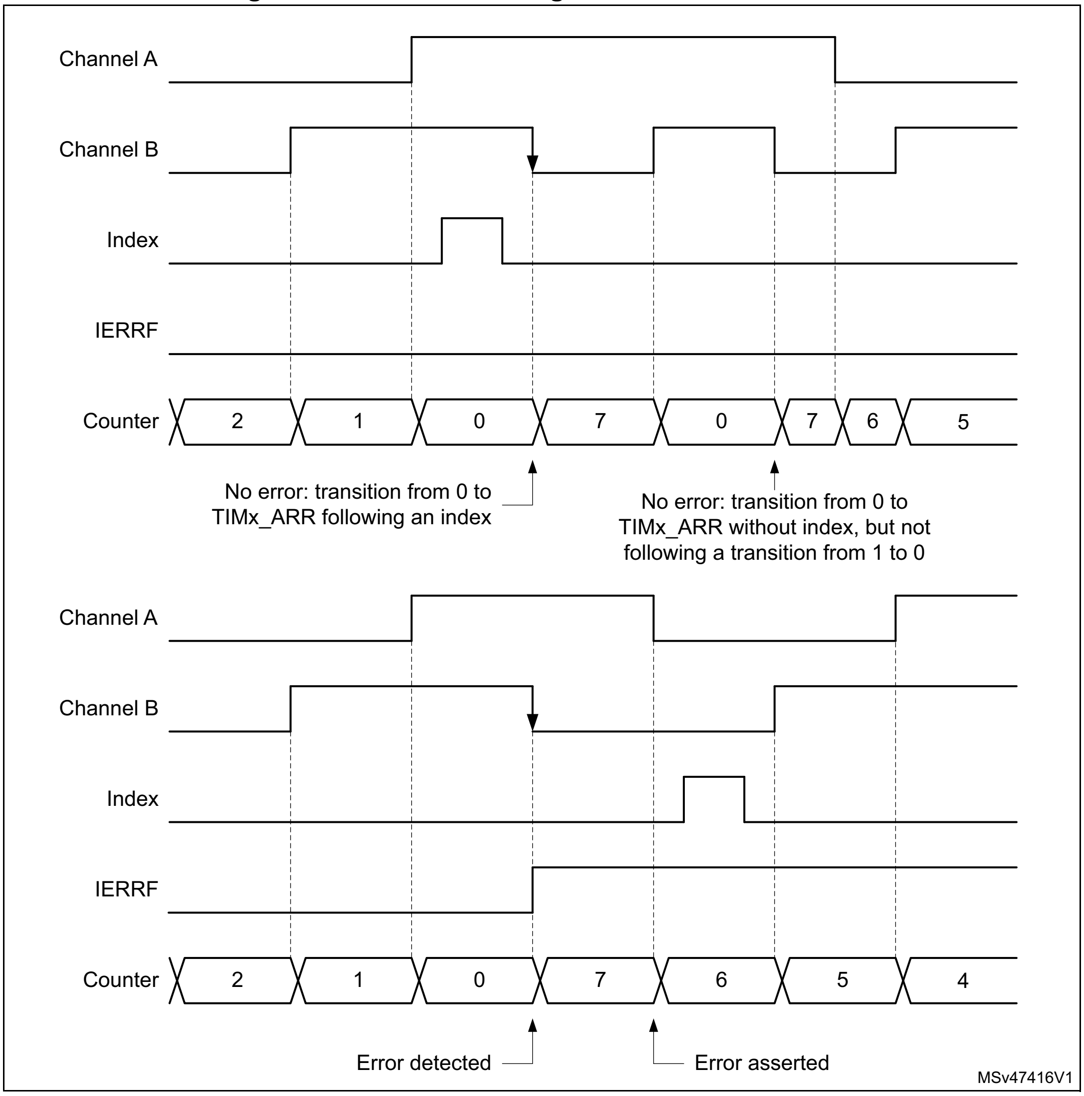
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 313*.

Figure 313. 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 314*, to avoid any false error detection in case the encoder dithers between TIMx_ARR and 0 immediately after the index detection.

Figure 314. 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.

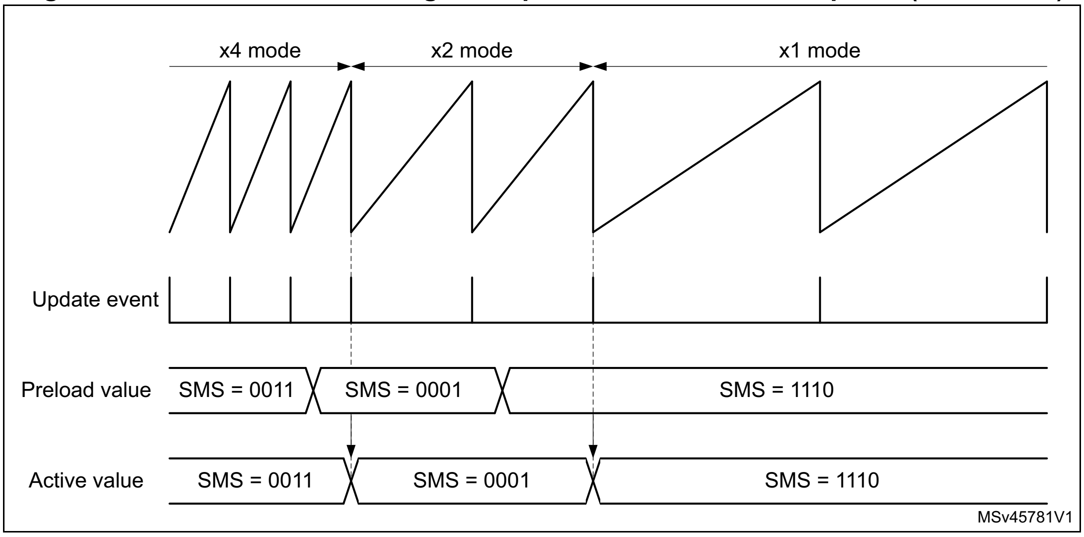
Slave mode selection preload for run-time encoder mode update

It can 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 shown on *Figure 315*.

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 up-counting, and underflows when down-counting.
- SMSPS = 1: the transfer is triggered by the index event.

Figure 315. 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.

30.4.19 Direction bit output

It 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] bitfield 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.

30.4.20 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register'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).

30.4.21 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 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.

An example of this feature used to interface Hall sensors is given in *Section 29.3.29: Interfacing with Hall sensors*.

30.4.22 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.

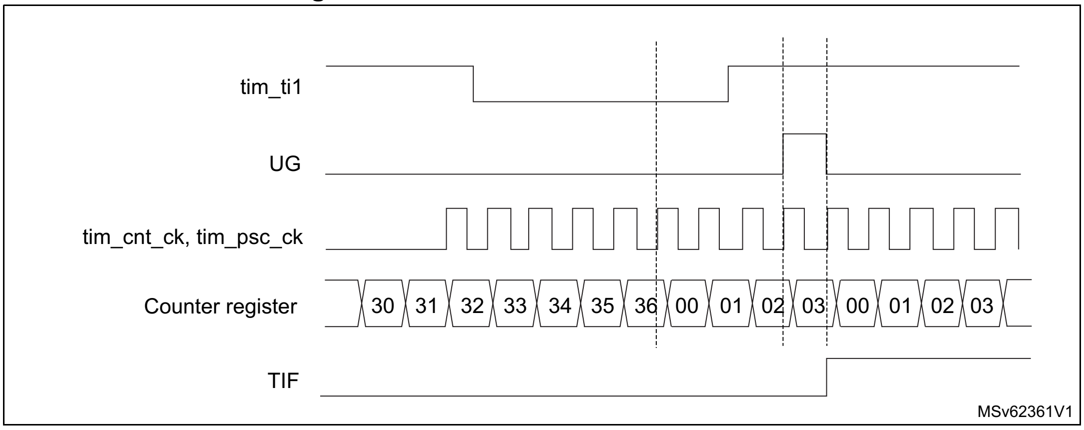
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

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 autoreload 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 316. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

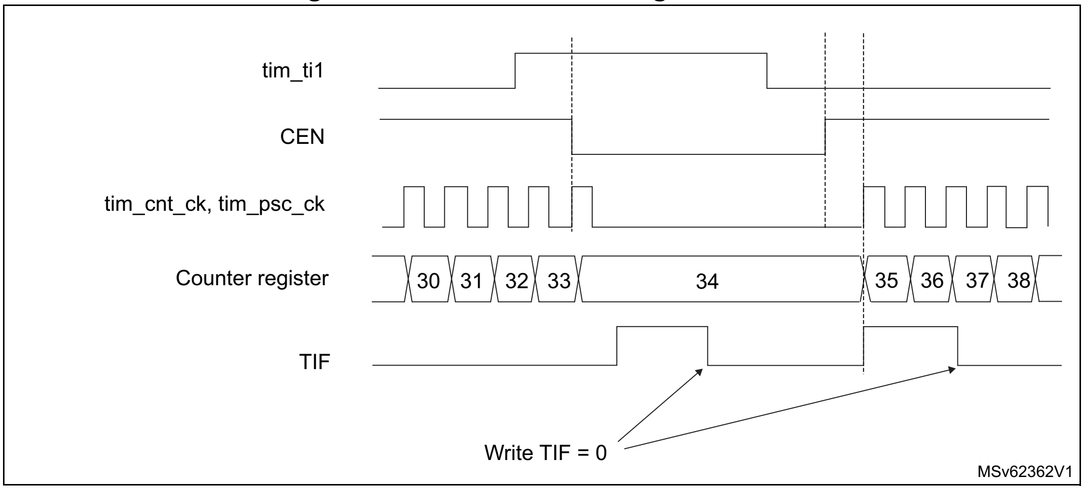
In the following example, the upcounter counts only when tim_ti1 input is low:

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 does not 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 317. 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.

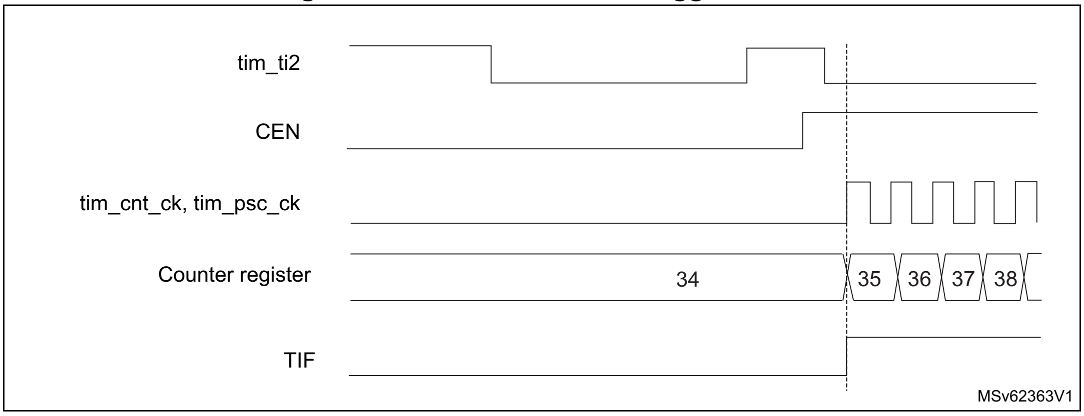
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

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 318. 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.

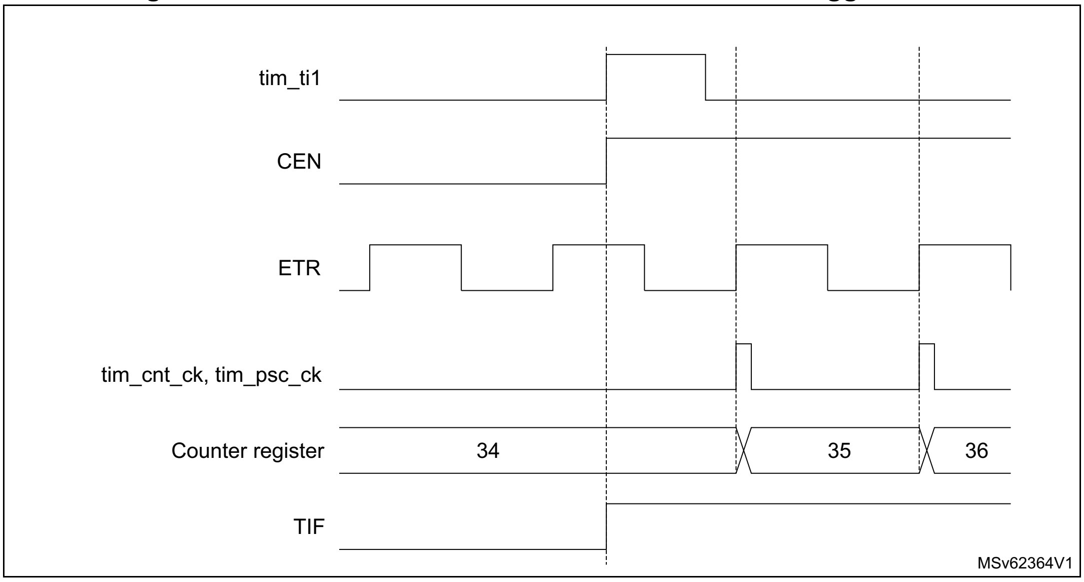
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

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 319. Control circuit in external clock mode 2 + trigger mode



30.4.23 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 320 and *Figure 321* show examples of master/slave timer connections.

Figure 320. Master/Slave timer example

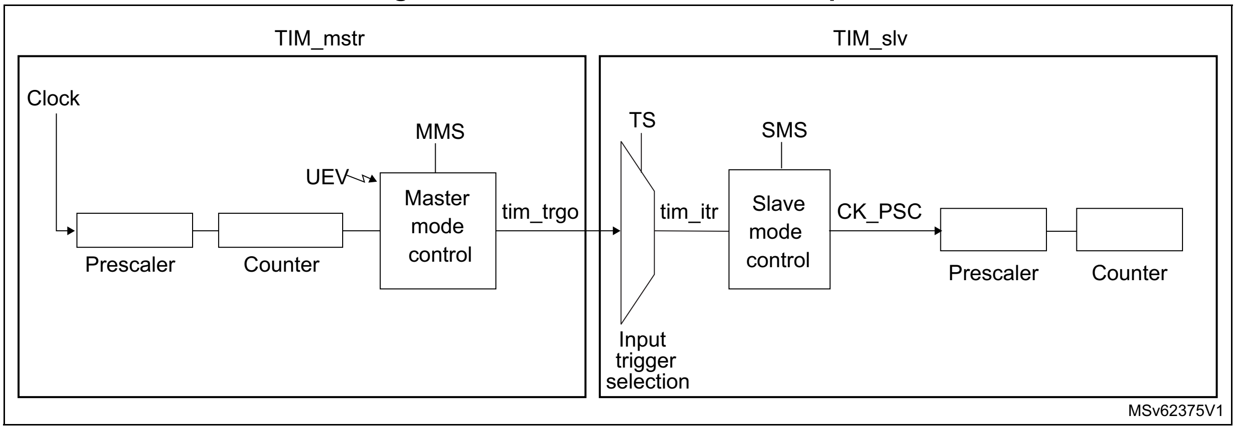
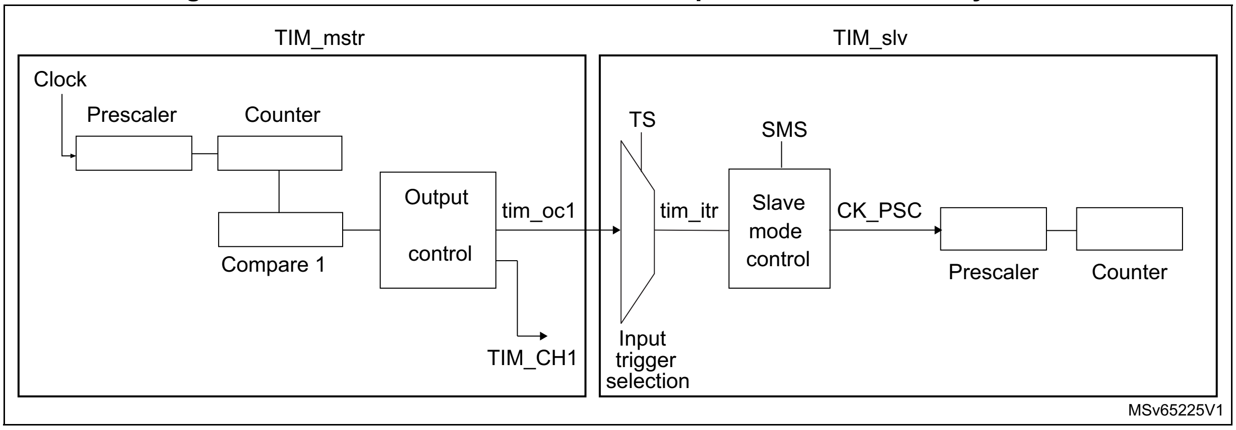


Figure 321. Master/slave connection example with 1 channel only timers



Note: The timers with one channel only (see Figure 321) 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 30.4.2: TIM2/TIM3 pins and internal signals).
The tim_oc1 signal pulse width must be programmed to be at least two 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 four times slower than the source timer, the OC1 pulse width must be eight 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 320*. 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.

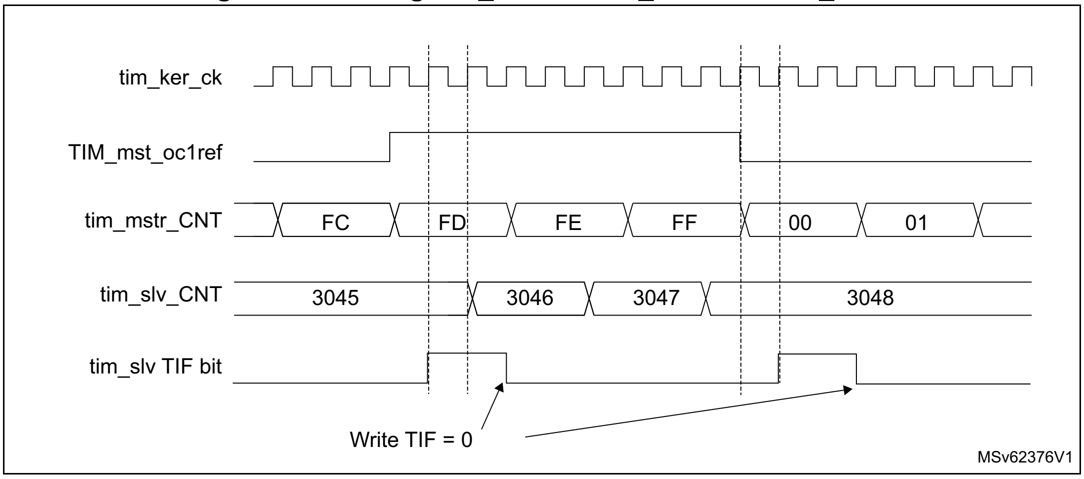
Using one timer to enable another timer

In this example, we control the enable of TIM_slv with the output compare 1 of TIM_mstr. Refer to *Figure 320* 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 322. Gating TIM_slv with tim_oc1ref of TIM_mstr

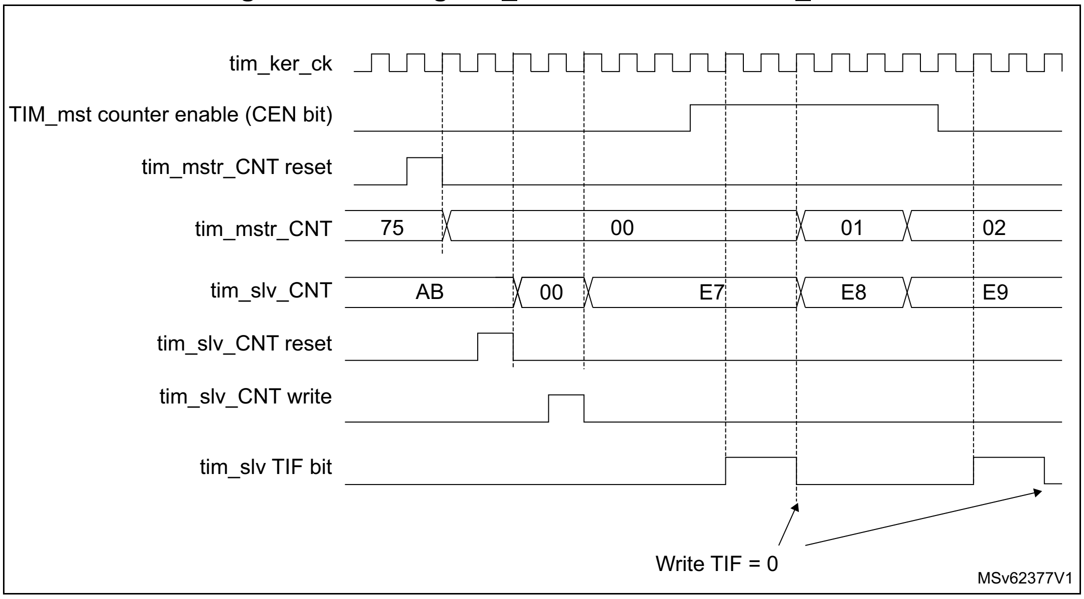


In the example in *Figure 322*, the TIM_slv counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM_mstr. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example (refer to *Figure 323*), 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 323. Gating TIM_slv with Enable of TIM_mstr

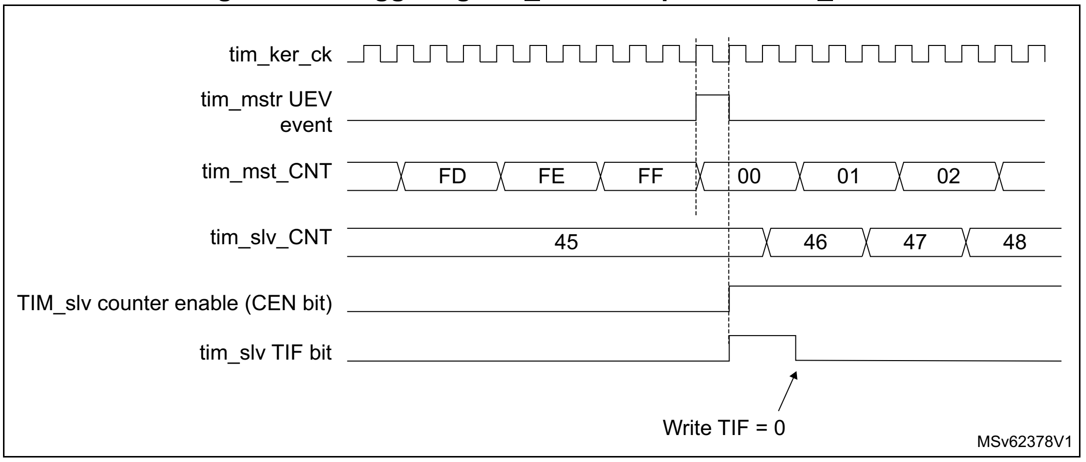


Using one timer to start another timer

In this example, we set the enable of TIM_slv with the update event of TIM_mstr. Refer to *Figure 320* for connections. TIM_slv starts counting from its current value (which can be nonzero) 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 ($f_{tim\_cnt\_ck} = f_{tim\_ker\_ck}/3$).

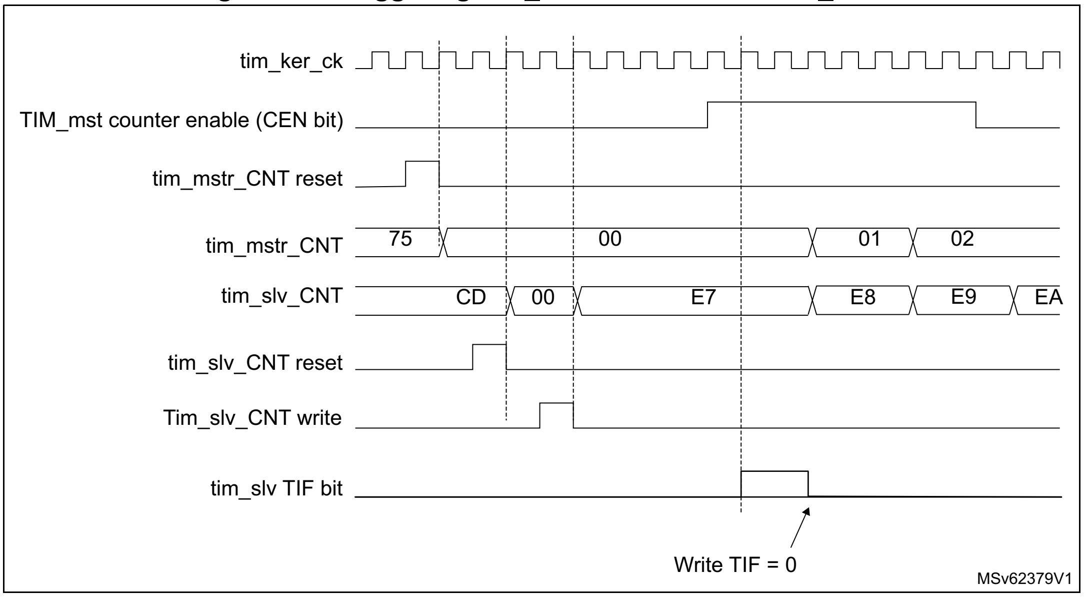
1. Configure TIM_mstr master mode to send its update event (UEV) as trigger output (MMS = 010 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr period (TIM_mstr_ARR registers).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv 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 324. Triggering TIM_slv with update of TIM_mstr



As in the previous example, both counters can be initialized before starting counting. *Figure 325* shows the behavior with the same configuration as in *Figure 324* but in trigger mode (SMS = 110 in the TIM_slv_SMCR register) instead of gated mode.

Figure 325. Triggering TIM_slv with Enable of TIM_mstr



Starting two timers synchronously in response to an external trigger

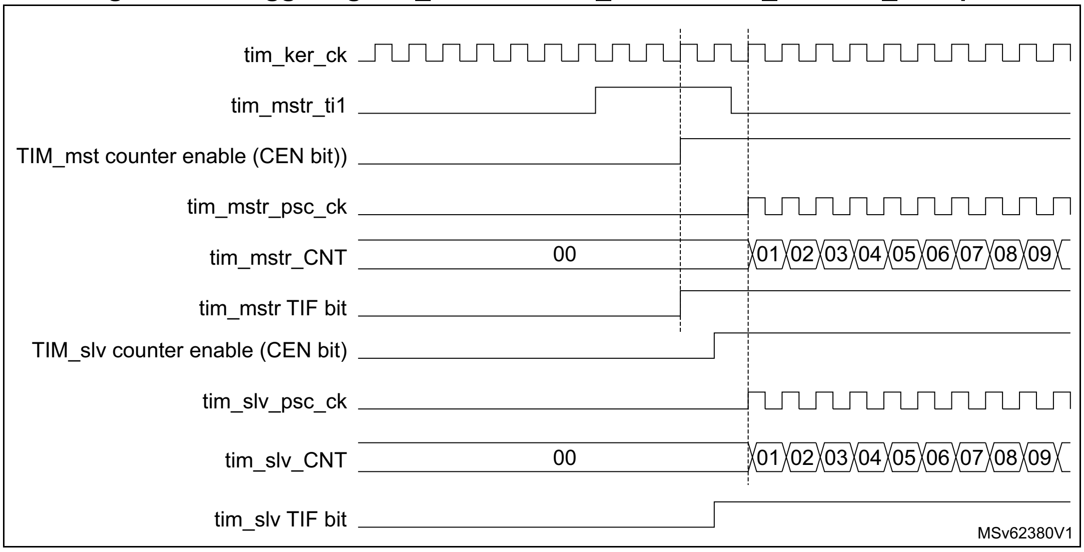
In this example, we set the enable of TIM_mstr when its tim_ti1 input rises, and the enable of TIM_slv with the enable of TIM_mstr. Refer to *Figure 320* for connections. To ensure the counters are aligned, TIM_mstr must be configured in Master/Slave mode (slave with respect to tim_ti1, master with respect 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 start 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 inserts a delay between CNT_EN and CK_PSC on TIM_mstr.

Figure 326. Triggering TIM_mstr and TIM_slv with TIM_mstr tim_ti1 input



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.

30.4.24 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.

30.4.25 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 reprogram 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 accesses 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 30.5.29: TIMx DMA control register (TIMx_DCR)(x = 2, 3)* 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 bitfields 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 must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

30.4.26 TIM2/TIM3 DMA requests

The TIM2/TIM3 can generate a DMA requests, as shown in *Table 290*.

Table 290. 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_trgi_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.

30.4.27 Debug mode

When the microcontroller enters debug mode (CPU1 Cortex®-M33 core halted), the TIMx counter can either continue 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).

30.4.28 TIM2/TIM3 low-power modes

Table 291. Effect of low-power modes on TIM2/TIM3

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

30.4.29 TIM2/TIM3 interrupts

The TIM2/TIM3 can generate multiple interrupts, as shown in *Table 292*.

Table 292. 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

30.5 TIM2/TIM3 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).

30.5.1 TIMx control register 1 (TIMx_CR1)(x = 2, 3)

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 bitfield 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**: Autoreload 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.

30.5.2 TIMx control register 2 (TIMx_CR2)(x = 2, 3)

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 29.3.29: Interfacing with Hall sensors*.

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.

30.5.3 TIMx slave mode control register (TIMx_SMCR)(x = 2, 3)

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 bitfield 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 bitfield 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 30.4.2: TIM2/TIM3 pins and internal signals* for product specific implementation details.

Note: These bits must be changed only when they are not used (for example 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 30.3: TIM2/TIM3 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 (refer to ETP bit in TIMx_SMCR for tim_etr_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).

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 (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.

30.5.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2, 3)

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.

30.5.5 TIMx status register (TIMx_SR)(x = 2, 3)

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 down-counting mode). There are three 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.

30.5.6 TIMx event generation register (TIMx_EGR)(x = 2, 3)

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 (up-counting), else it takes the autoreload value (TIMx_ARR) if DIR = 1 (down-counting).

30.5.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 2, 3)

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 (for example 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 bitfield 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 bitfield 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 bitfield 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 bitfield 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).

30.5.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2, 3)

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 (for example 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 bitfield 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 can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the ouput keeps the state (active or inactive) it had before entering the frozen state.

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 up-counting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In down-counting, channel 1 is inactive (tim_oc1ref = 0) as long as TIMx_CNT>TIMx_CCR1 else active (tim_oc1ref = 1).

0111: PWM mode 2 - In up-counting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In down-counting, 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 OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” 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 three 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 bitfield 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).

30.5.9 TIMx capture/compare mode register 2 (TIMx_CCMR2)(x = 2, 3)

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 (for example 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 bitfield 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 bitfield 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).

30.5.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2, 3)

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 (for example 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 OC3M[3:0]

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 bitfield 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 up-counting, channel 3 is active as long as TIMx_CNT<TIMx_CCR3 else inactive. In down-counting, channel 3 is inactive (tim_oc3ref = 0) as long as TIMx_CNT>TIMx_CCR3 else active (tim_oc3ref = 1).

0111:PWM mode 2 - In up-counting, channel 3 is inactive as long as TIMx_CNT<TIMx_CCR3 else active. In down-counting, 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 bitfield 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 bitfield 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).

30.5.11 TIMx capture/compare enable register (TIMx_CCER)(x = 2, 3)

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 293. 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.

30.5.12 TIM3 counter (TIM3_CNT)

Address offset: 0x024

Reset value: 0x0000 0000

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

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

Bit 31 **UIFCPY**: Value depends on IUFREMAP in TIMx_CR1.
If UIFREMAP = 0
Reserved
If UIFREMAP = 1
UIFCPY: UIF Copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register

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

Bits 15:0 **CNT[15:0]**: Counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register holds the non-dithered part in CNT[15:0]. The fractional part is not available.

30.5.13 TIM2 counter (TIM2_CNT)

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_ 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.

30.5.14 TIMx prescaler (TIMx_PSC)(x = 2, 3)

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”).

30.5.15 TIM3 autoreload register (TIM3_ARR)

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]**: Low autoreload value

ARR is the value to be loaded in the actual autoreload register.
Refer to the *Section 30.4.3: Time-base unit* for more details about ARR update and behavior.
The counter is blocked while the autoreload value is null.
Non-dithering mode (DITHEN = 0)
The register holds the autoreload value.
Dithering mode (DITHEN = 1)
The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

30.5.16 TIM2 autoreload register (TIM2_ARR)

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]**: Autoreload value

ARR is the value to be loaded in the actual autoreload register.

Refer to the *Section 30.4.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the autoreload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the autoreload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[31:4]. The ARR[3:0] bitfield contains the dithered part.

30.5.17 TIM3 capture/compare register 1 (TIM3_CCR1)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The CCR1[15:0] bits hold the capture value. The CCR1[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:0]. The CCR1[3:0] bits are reset.

30.5.18 TIM2 capture/compare register 1 (TIM2_CCR1)

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.

30.5.19 TIM3 capture/compare register 2 (TIM3_CCR2)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 19:0 **CCR2[19:0]**: Capture/compare 1 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc2 output.

<u>Non-dithering mode (DITHEN = 0)</u>

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

<u>Dithering mode (DITHEN = 1)</u>

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.

<u>Non-dithering mode (DITHEN = 0)</u>

The CCR2[15:0] bits hold the capture value. The CCR2[19:16] bits are reserved.

<u>Dithering mode (DITHEN = 1)</u>

The register holds the capture in CCR2[19:0]. The CCR2[3:0] bits are reset.

30.5.20 TIM2 capture/compare register 2 (TIM2_CCR2)

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.

<u>Non-dithering mode (DITHEN = 0)</u>

The register holds the compare value.

<u>Dithering mode (DITHEN = 1)</u>

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.

<u>Non-dithering mode (DITHEN = 0)</u>

The register holds the capture value.

<u>Dithering mode (DITHEN = 1)</u>

The register holds the capture in CCR2[31:0]. The CCR2[3:0] bits are reset.

30.5.21 TIM3 capture/compare register 3 (TIM3_CCR3)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 19:0 **CCR3[19:0]**: Capture/compare 3 value

If channel CC3 is configured as output:

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc3 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR3[15:0]. The CCR3[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR3[19:4]. The CCR3[3:0] bitfield contains the dithered part.

If channel CC3 is configured as input:

CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIMx_CCR3 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The CCR3[15:0] bits hold the capture value. The CCR3[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR3[19:0]. The CCR3[3:0] bits are reset.

30.5.22 TIM2 capture/compare register 3 (TIM2_CCR3)

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.

30.5.23 TIM3 capture/compare register 4 (TIM3_CCR4)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 19:0 **CCR4[19:0]**: Capture/compare 4 value

If channel CC4 is configured as output:

CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[19:4]. The CCR4[3:0] bitfield contains the dithered part.

If channel CC4 is configured as input:

CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIMx_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The CCR4[15:0] bits hold the capture value. The CCR4[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[19:0]. The CCR4[3:0] bits are reset.

30.5.24 TIM2 capture/compare register 4 (TIM2_CCR4)

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.

30.5.25 TIMx timer encoder control register (TIMx_ECR)(x = 2, 3)

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

Bit 0 **IE**: Index enable

This bit indicates if the Index event resets the counter.

0: Index disabled
1: Index enabled

30.5.26 TIMx timer input selection register (TIMx_TISEL)(x = 2, 3)

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 30.4.2: TIM2/TIM3 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 30.4.2: TIM2/TIM3 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 30.4.2: TIM2/TIM3 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 30.4.2: TIM2/TIM3 pins and internal signals* for product specific implementation.

30.5.27 TIMx alternate function register 1 (TIMx_AF1)(x = 2, 3)

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 30.4.2: TIM2/TIM3 pins and internal signals* for product specific implementation.

Bits 13:0 Reserved, must be kept at reset value.

30.5.28 TIMx alternate function register 2 (TIMx_AF2)(x = 2, 3)

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 30.4.2: TIM2/TIM3 pins and internal signals* for product specific implementation.

Bits 15:0 Reserved, must be kept at reset value.

30.5.29 TIMx DMA control register (TIMx_DCR)(x = 2, 3)

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 is 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,
...

30.5.30 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2, 3)

Address offset: 0x3E0

Reset value: 0x0000 0000

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

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

30.5.31 TIMx register map

TIMx registers are mapped as described in the table below.

Table 294. TIM2/TIM3 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

Table 294. TIM2/TIM3 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x020	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
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 = 3)	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
0x02C	TIMx_ARR (x = 2)	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																																
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.
	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.																															

Table 294. TIM2/TIM3 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
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: Memory organization* for the register boundary addresses.

31 General purpose timers (TIM16/TIM17)

TIM17 is available only on STM32WBA52/54/55xx devices.

31.1 TIM16/TIM17 introduction

The TIM16/TIM17 timers consist of a 16-bit autoreload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM16/TIM17 timers are completely independent, and do not share any resources.

31.2 TIM16/TIM17 main features

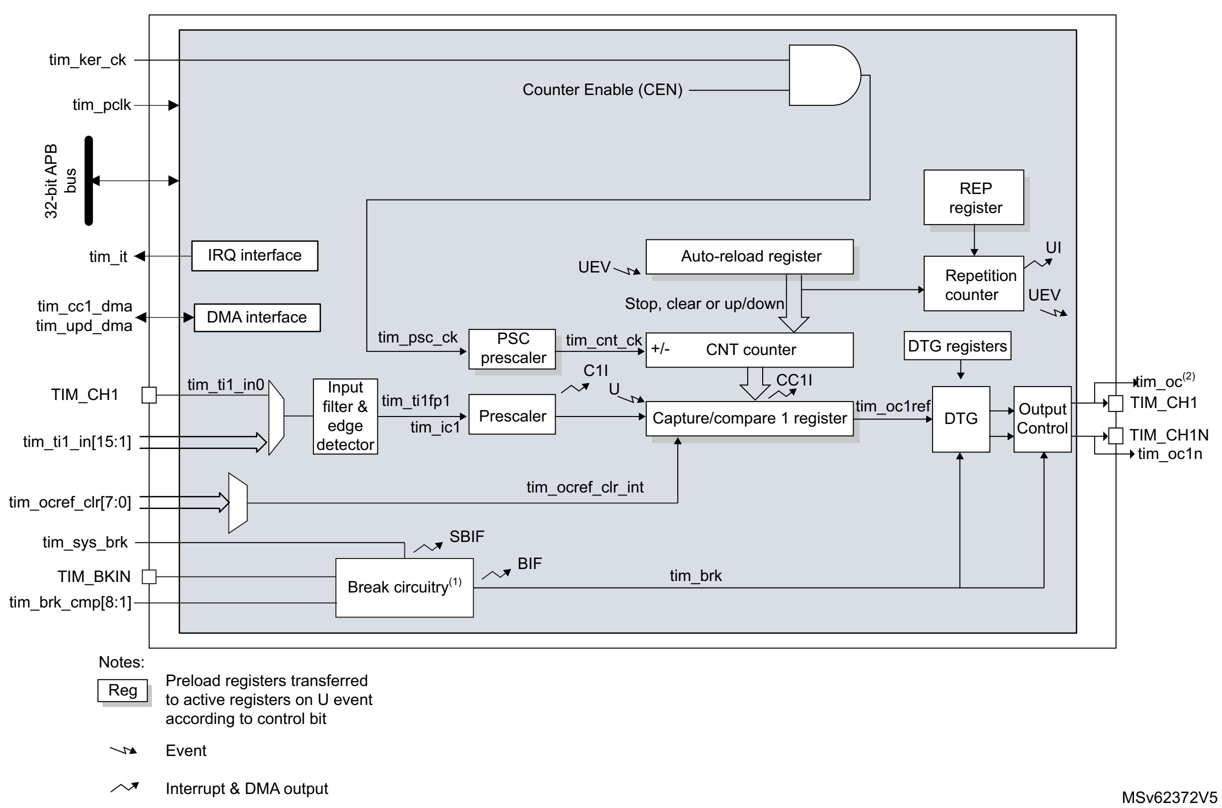
The TIM16/TIM17 timers include the following features:

- 16-bit autoreload 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

31.3 TIM16/TIM17 functional description

31.3.1 Block diagram

Figure 327. TIM16/TIM17 block diagram



1. Refer to *Section 31.3.13: 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 31.3.19: Using timer output as trigger for other timers (TIM16/TIM17 only)* for details.

31.3.2 TIM16/TIM17 pins and internal signals

Table 295 and *Table 296* in this section summarize the TIM inputs and outputs.

Table 295. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 can also be used as external clock (below 1/4 of the tim_ker_ck clock) and external trigger input.
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.

Table 296. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0]	Input	Internal timer input 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]	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 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

Table 297 lists the sources connected to the tim_ti1 input multiplexer.

Table 297. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources	
	TIM16	TIM17
tim_ti1_in0	TIM16_CH1	TIM17_CH1
tim_ti1_in1	Reserved	
tim_ti1_in2	MCO	
tim_ti1_in3	HSE32 / 32	
tim_ti1_in4	rtc_wut_trg	
tim_ti1_in5	LSE	
tim_ti1_in6	LSI	
tim_ti1_in[8:7]	Reserved	
tim_ti1_in9	HSI16 / 256	
tim_ti1_in[15:10]	Reserved	

Table 298 and Table 299 list the sources connected to the tim_brk and tim_brk2 inputs.

Table 298. Timer break interconnect

tim_brk inputs	TIM16	TIM17
TIM_BKIN	TIM16_BKIN	TIM17_BKIN
tim_brk_cmp1	COMP1_OUT(1)	COMP1_OUT(1)
tim_brk_cmp2	COMP2_OUT(1)	COMP2_OUT(1)
tim_brk_cmp[8:3]	Reserved	

1. Only available on STM32WBA54xx and STM32WBA55xx devices.

Table 299. System break interconnect

tim_sys_brk inputs	TIM16	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 parity error	SPL
tim_sys_brk3	Flash ECC error	ECCL
tim_sys_brk4	HSE32 clock security system (HSECSS)	None (always enabled)

Table 300 lists the internal sources connected to the tim_ocref_clr input multiplexer.

Table 300. Interconnect to the ocref_clr input multiplexer

Timer OCREF clear signal	Timer OCREF clear signals assignment	
	TIM16	TIM17
tim_ocref_clr0	COMP1_OUT(1)	COMP1_OUT(1)
tim_ocref_clr1	COMP2_OUT(1)	COMP2_OUT(1)
tim_ocref_clr[7:2]	Reserved	

1. Only available on STM32WBA54xx and STM32WBA55xx devices.

31.3.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related autoreload register. The counter clock can be divided by a prescaler.

The counter, the autoreload 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)
- Autoreload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload 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 one 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.

Table 338 and *Table 329* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 328. Counter timing diagram with prescaler division change from 1 to 2

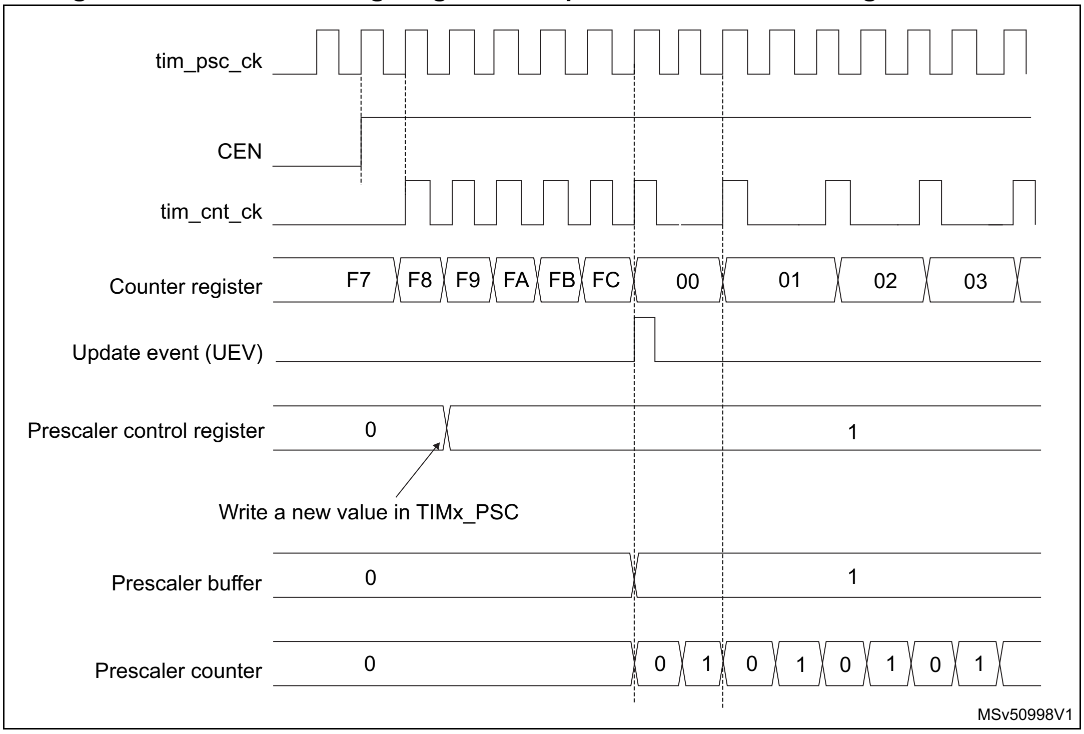
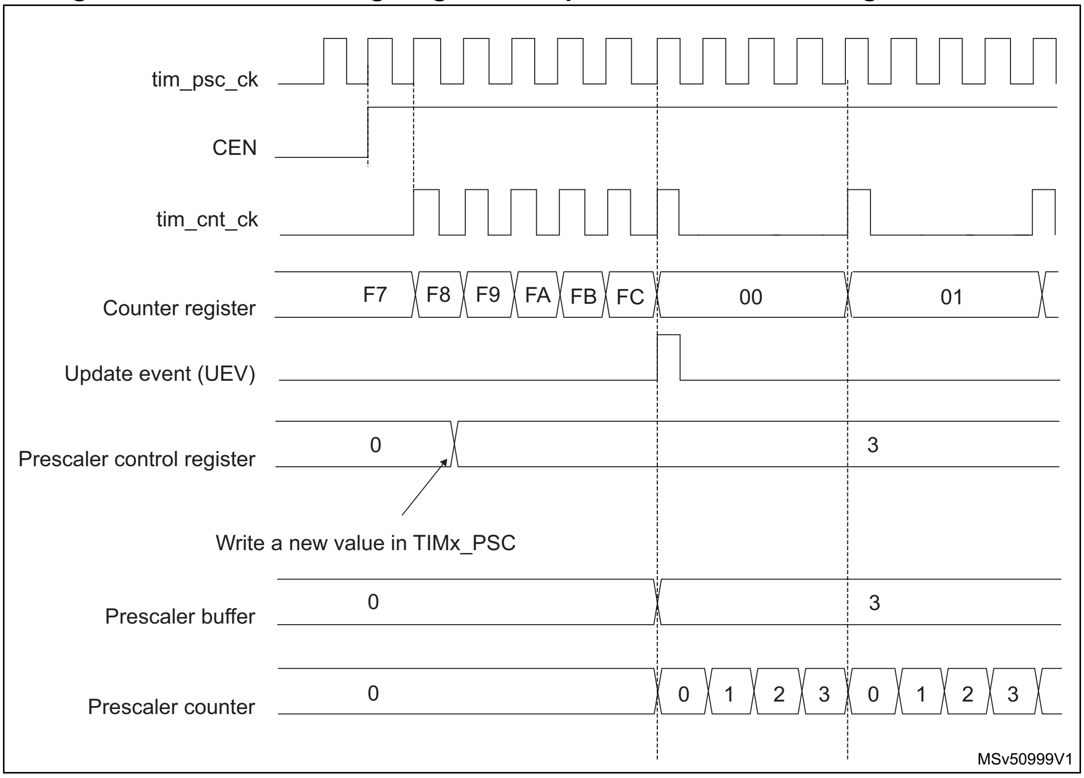


Figure 329. Counter timing diagram with prescaler division change from 1 to 4



31.3.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload 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 autoreload 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 330. Counter timing diagram, internal clock divided by 1

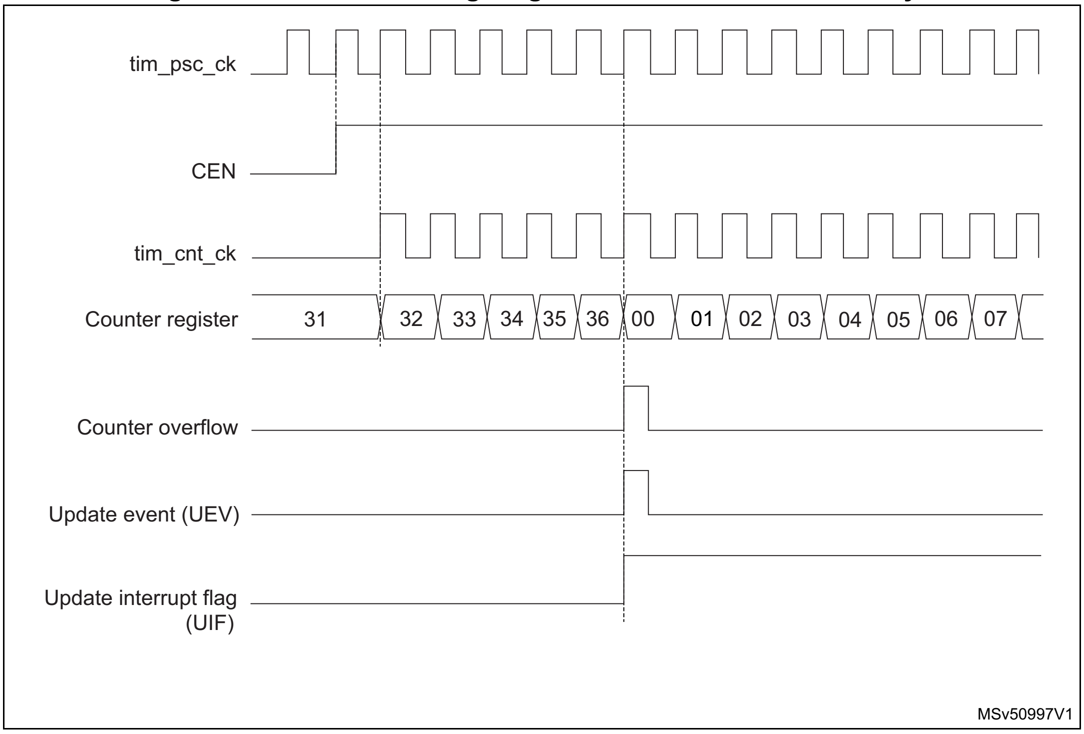


Figure 331. Counter timing diagram, internal clock divided by 2

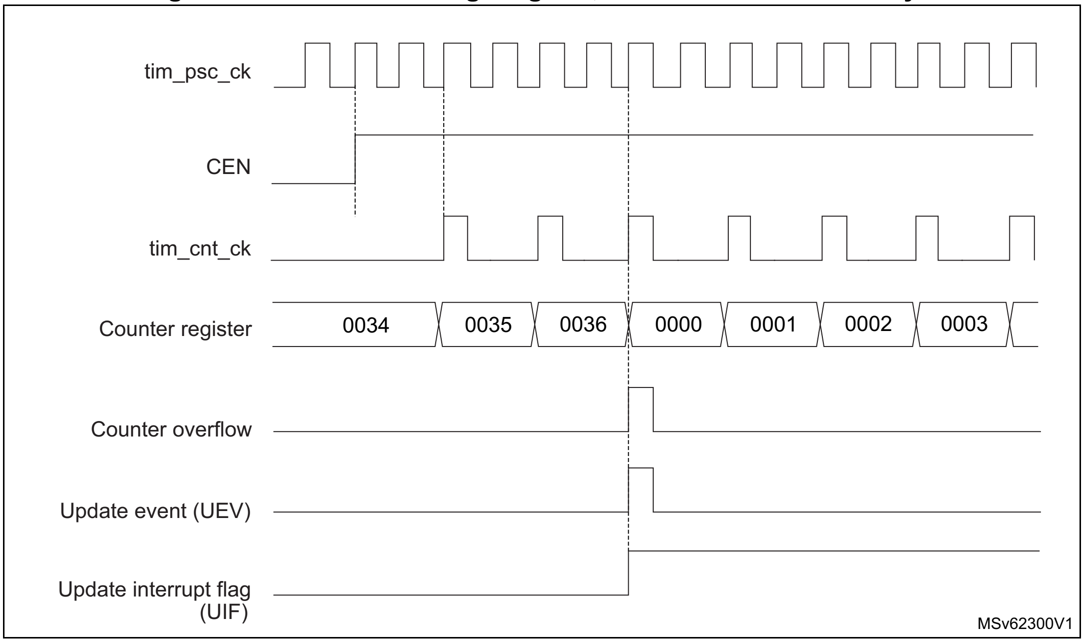


Figure 332. Counter timing diagram, internal clock divided by 4

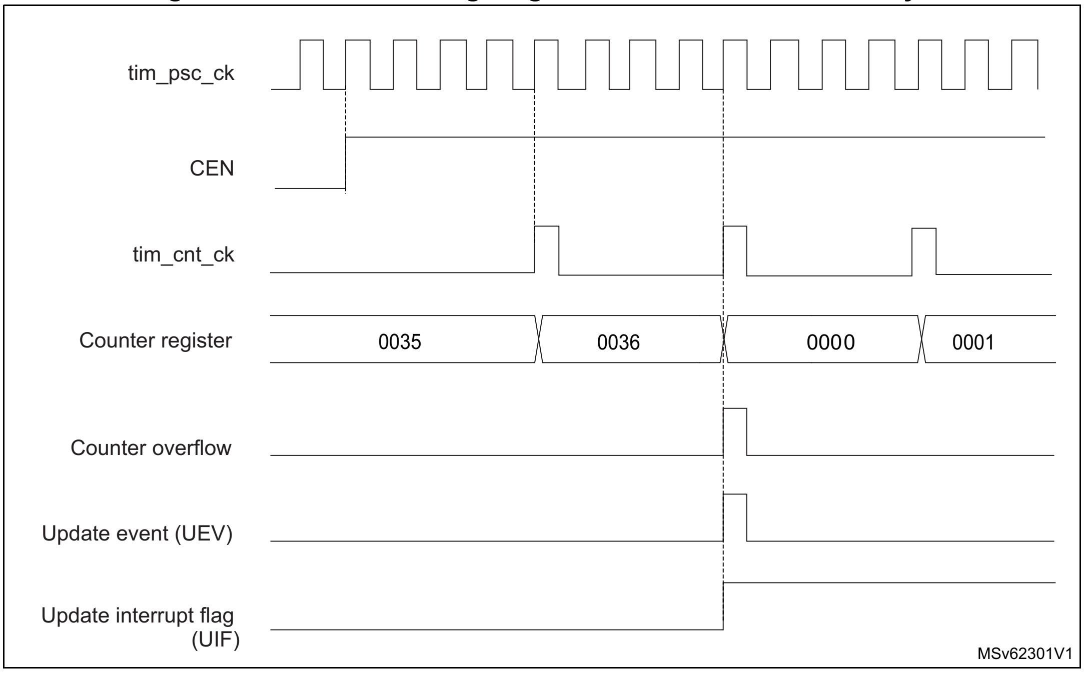


Figure 333. Counter timing diagram, internal clock divided by N

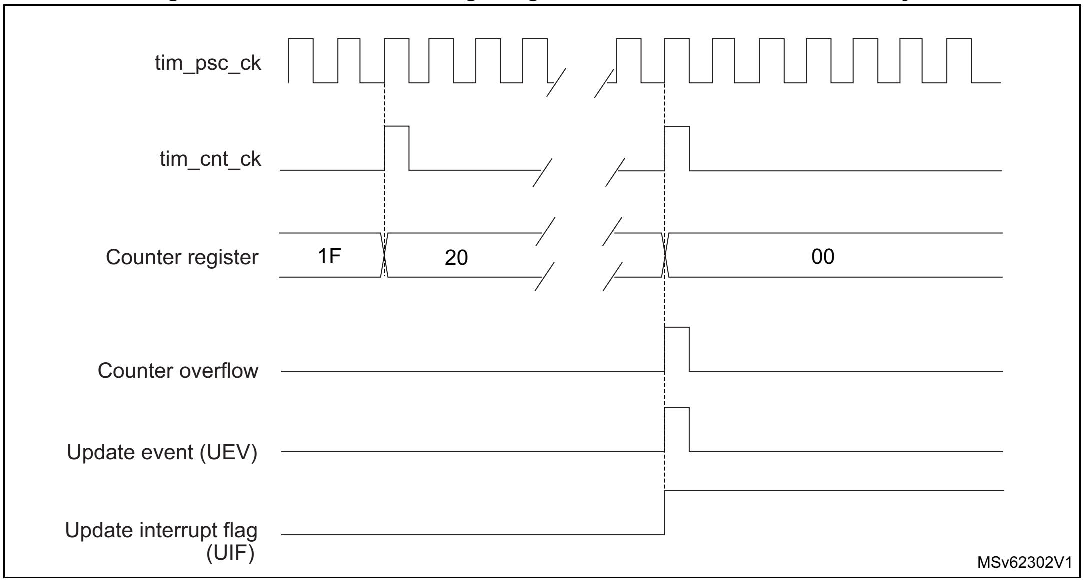


Figure 334. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

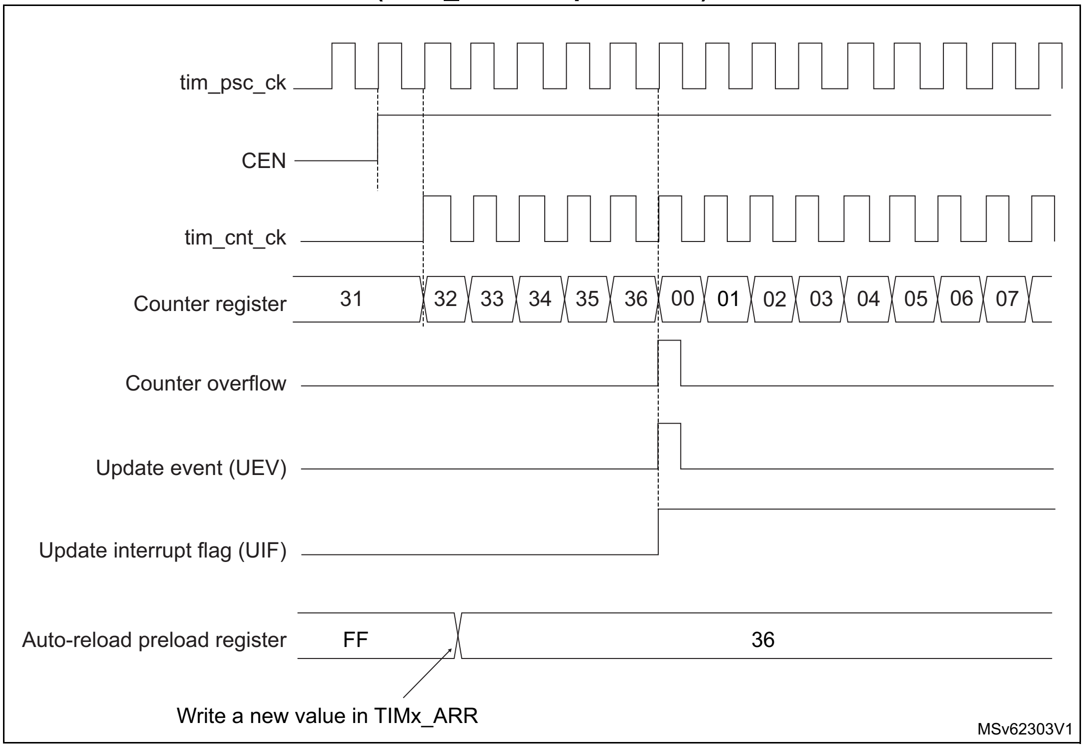
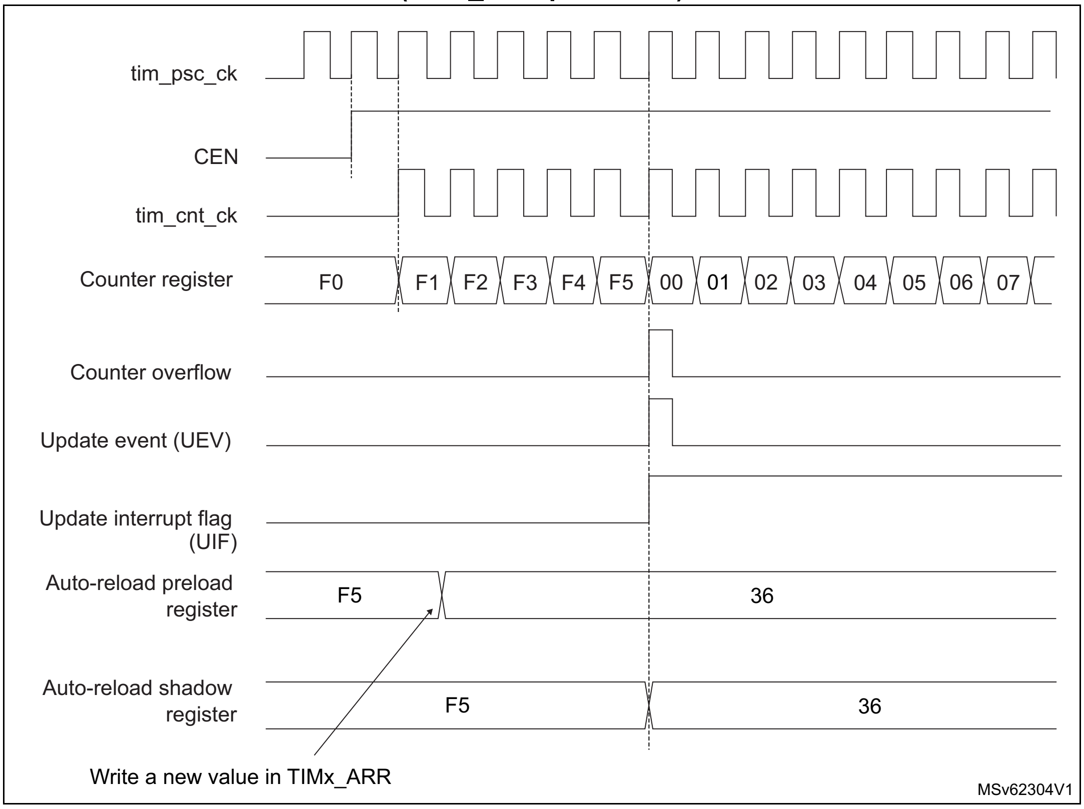


Figure 335. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)



31.3.5 Repetition counter

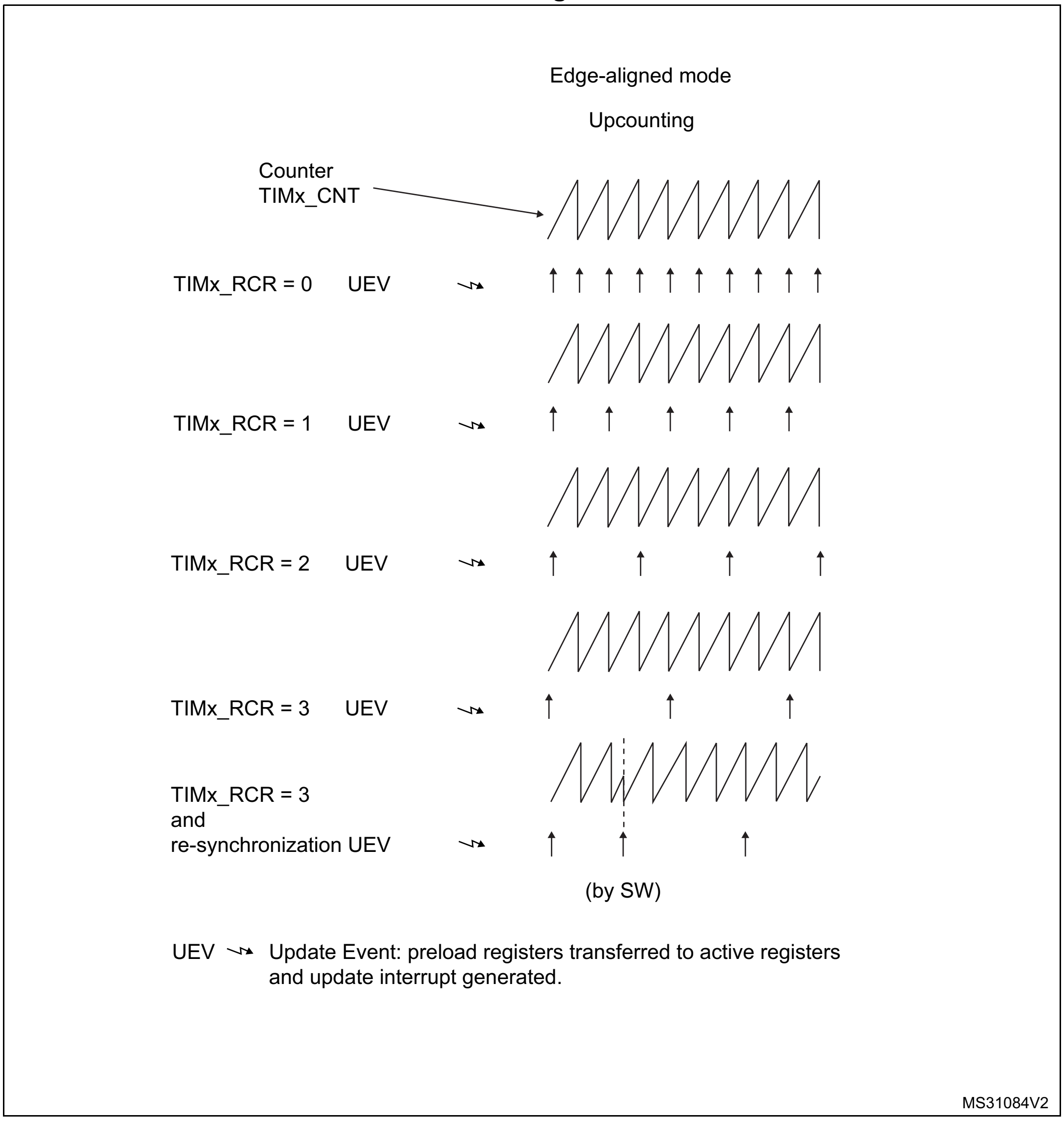
Section 31.3.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR autoreload 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 autoreload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 336*). 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 336. Update rate examples depending on mode and TIMx_RCR register settings



31.3.6 Clock selection

The counter clock can be provided by the following clock sources:

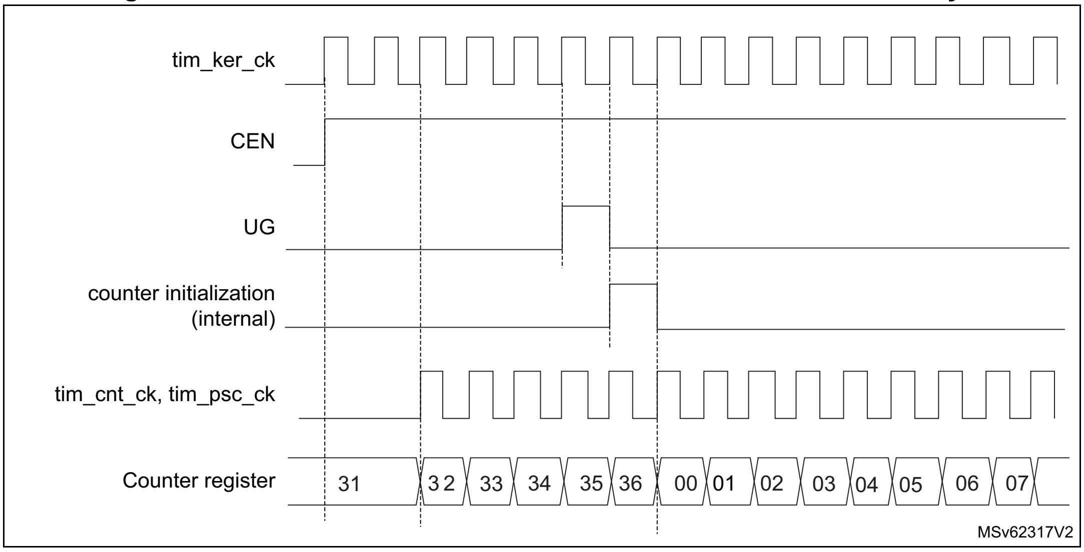
- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2, if available)

Internal 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 337 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

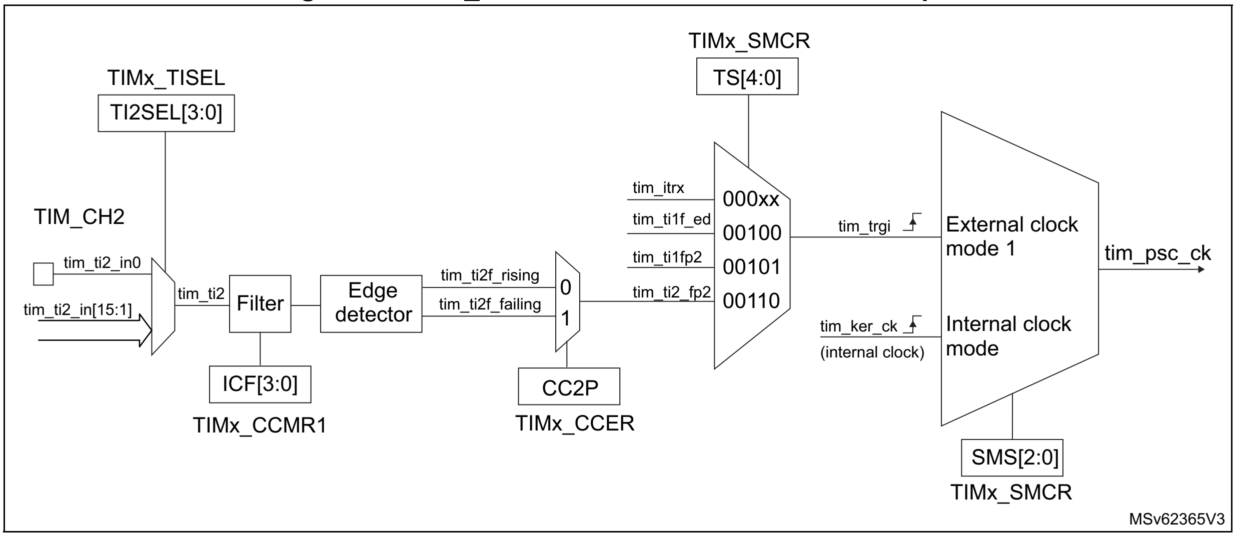
Figure 337. Control circuit in normal mode, internal clock divided by 1



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 338. 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:

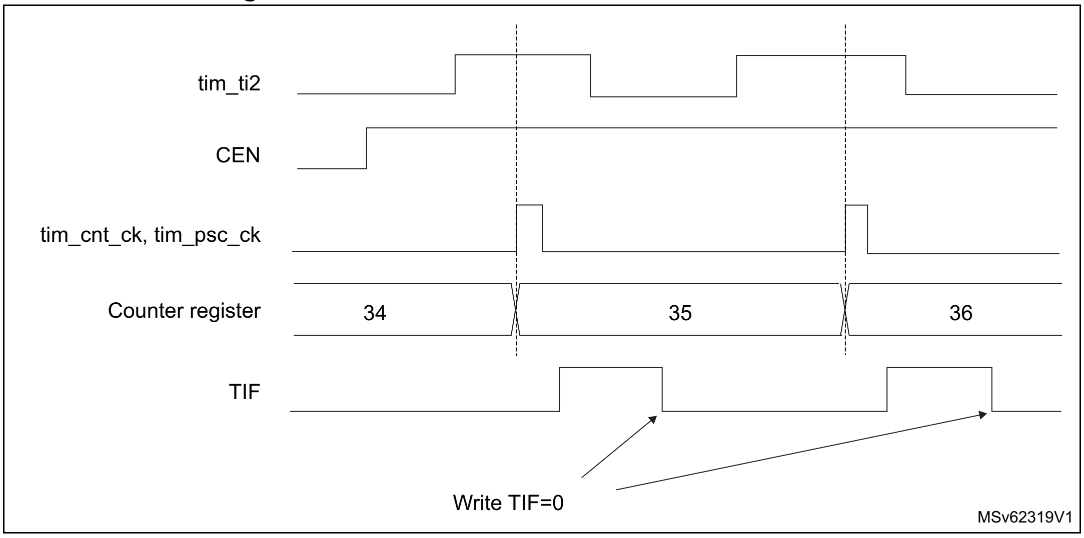
1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] 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 339. Control circuit in external clock mode 1



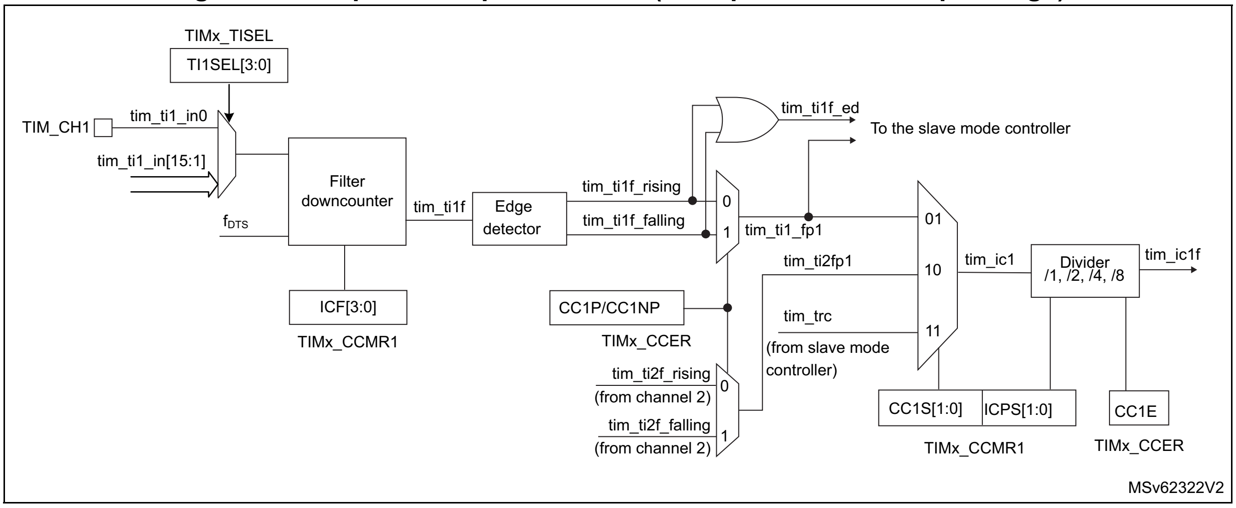
31.3.7 Capture/compare channels

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 340 to *Figure 342* 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 340. 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 341. Capture/compare channel 1 main circuit

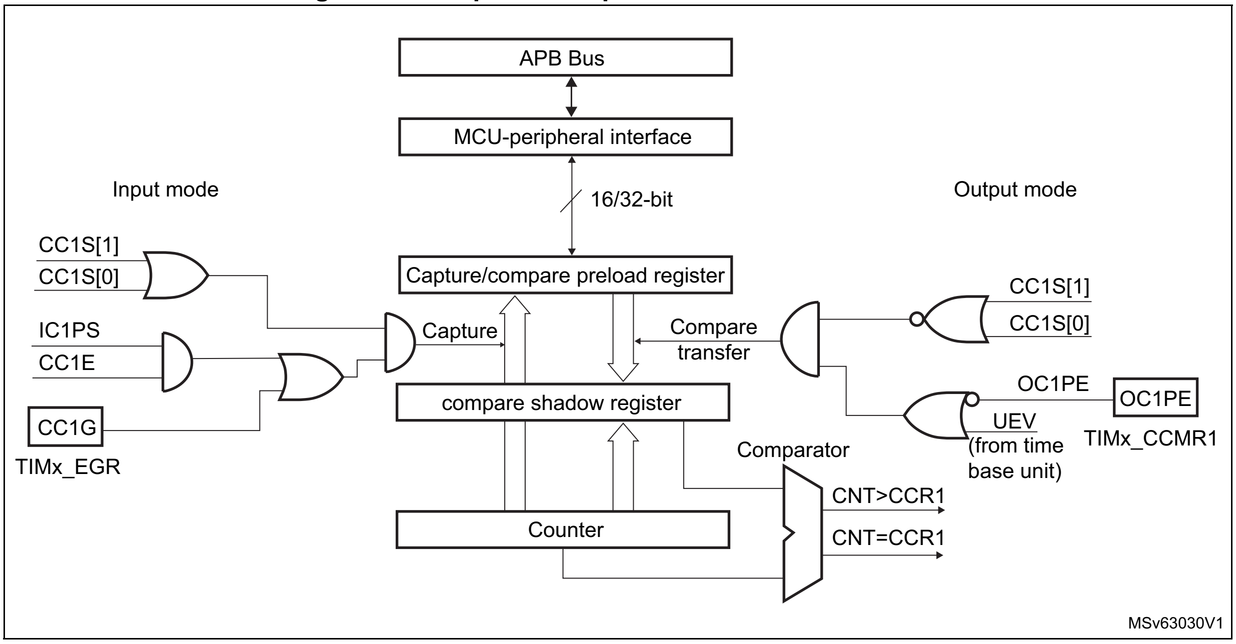
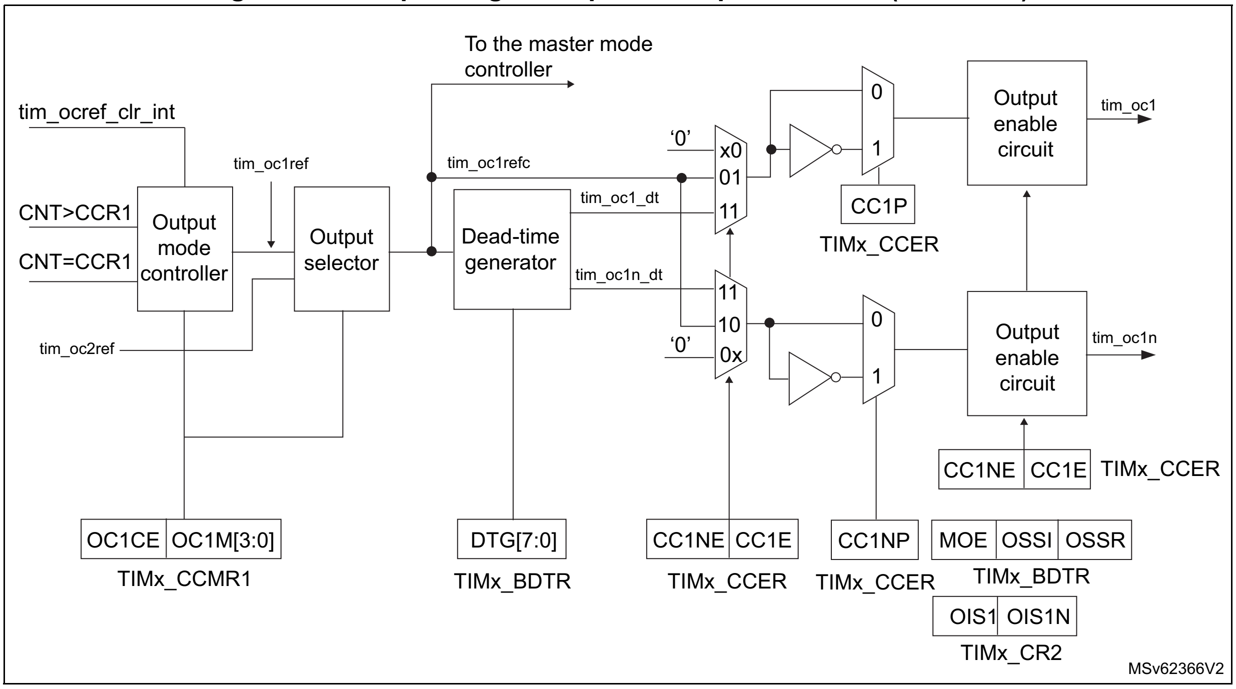


Figure 342. Output stage of capture/compare channel (channel 1)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

31.3.8 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 overcapture 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. The user must program a filter duration longer than these five clock cycles. The user can validate a transition on tim_ti1 when eight 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 this example, the user wants 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 may 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.

31.3.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/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.

31.3.10 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.

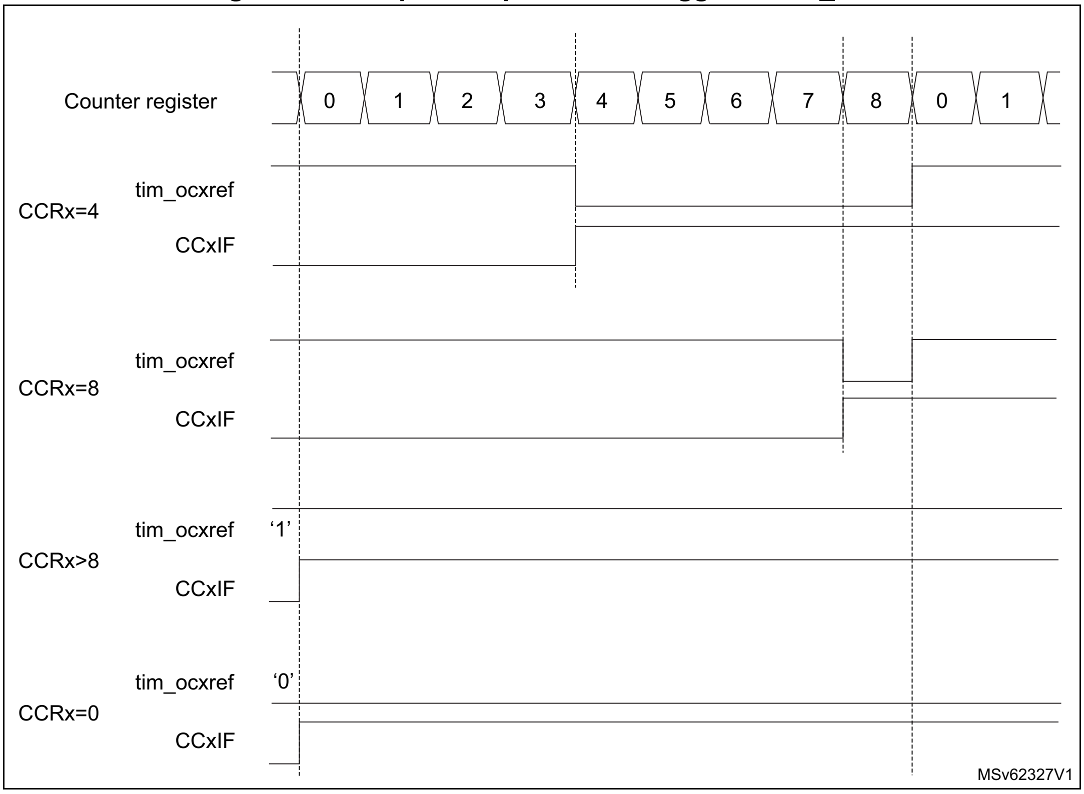
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

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 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 343*.

Figure 343. Output compare mode, toggle on tim_oc1



31.3.11 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 autoreload 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.

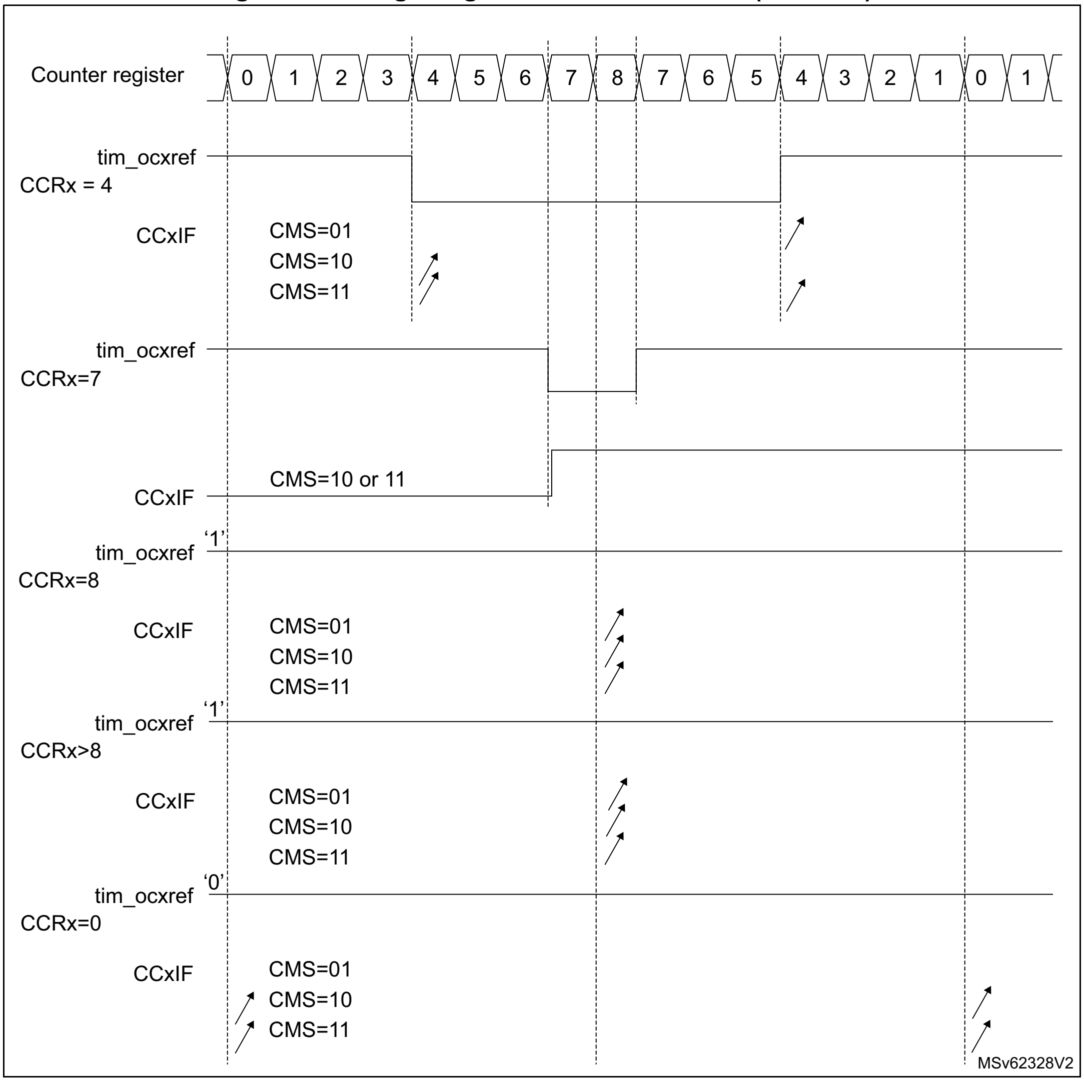
In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The TIM16/TIM17 are capable of upcounting only. Refer to *Upcounting mode*.

In the following example applies to 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 autoreload 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 344 shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

Figure 344. Edge-aligned PWM waveforms (ARR = 8)

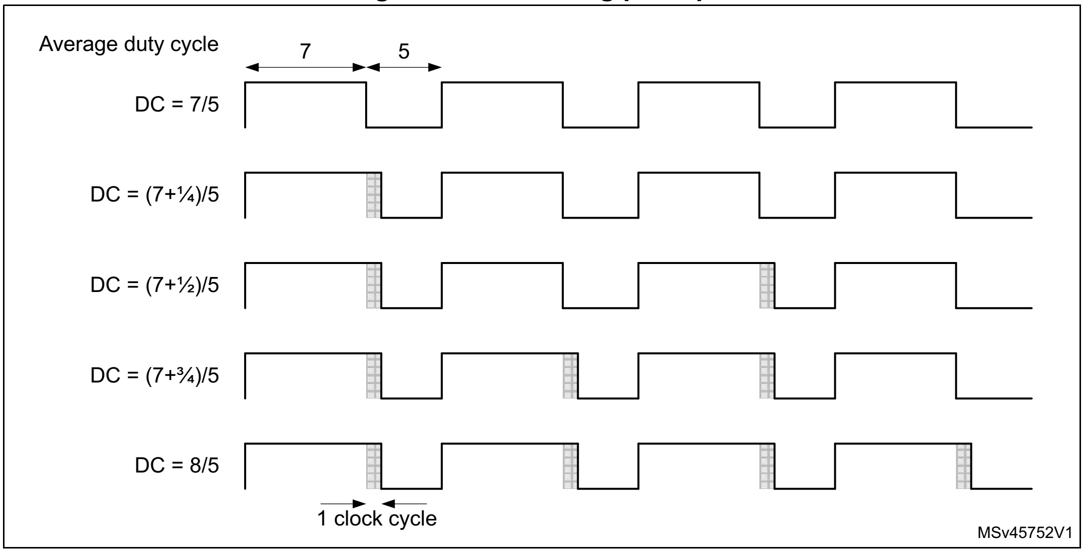


Dithering mode

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 345 below presents the dithering principle applied to four consecutive PWM cycles.

Figure 345. Dithering principle

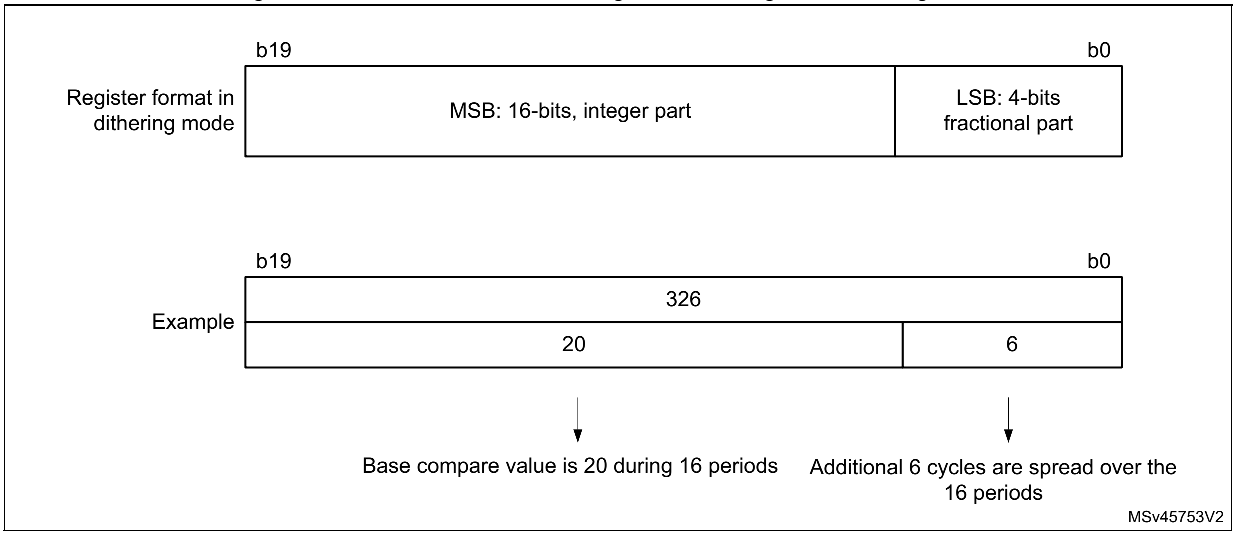


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 346* for example):

- The four 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 CCIF flags must be cleared
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 346. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

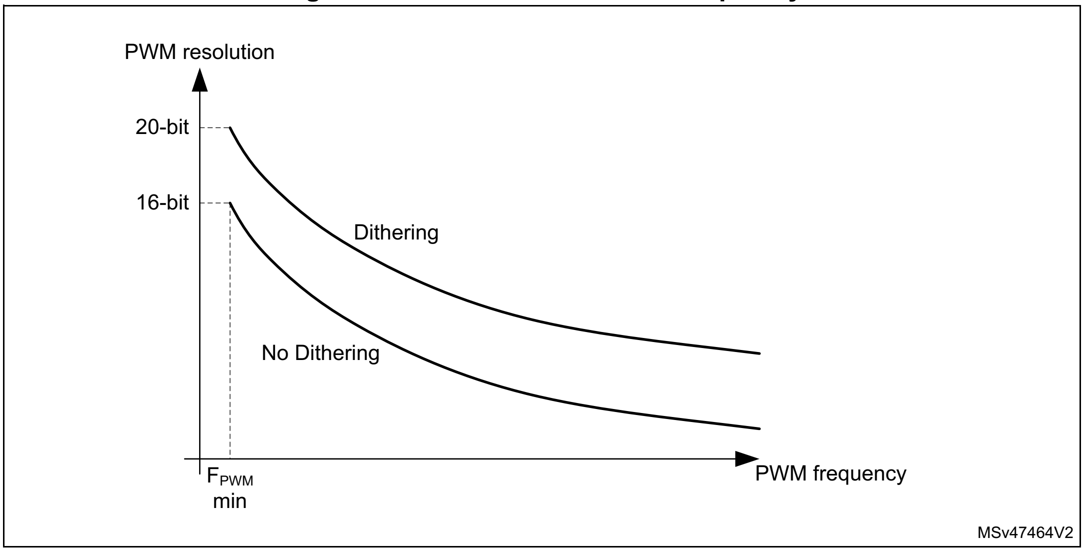
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

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).

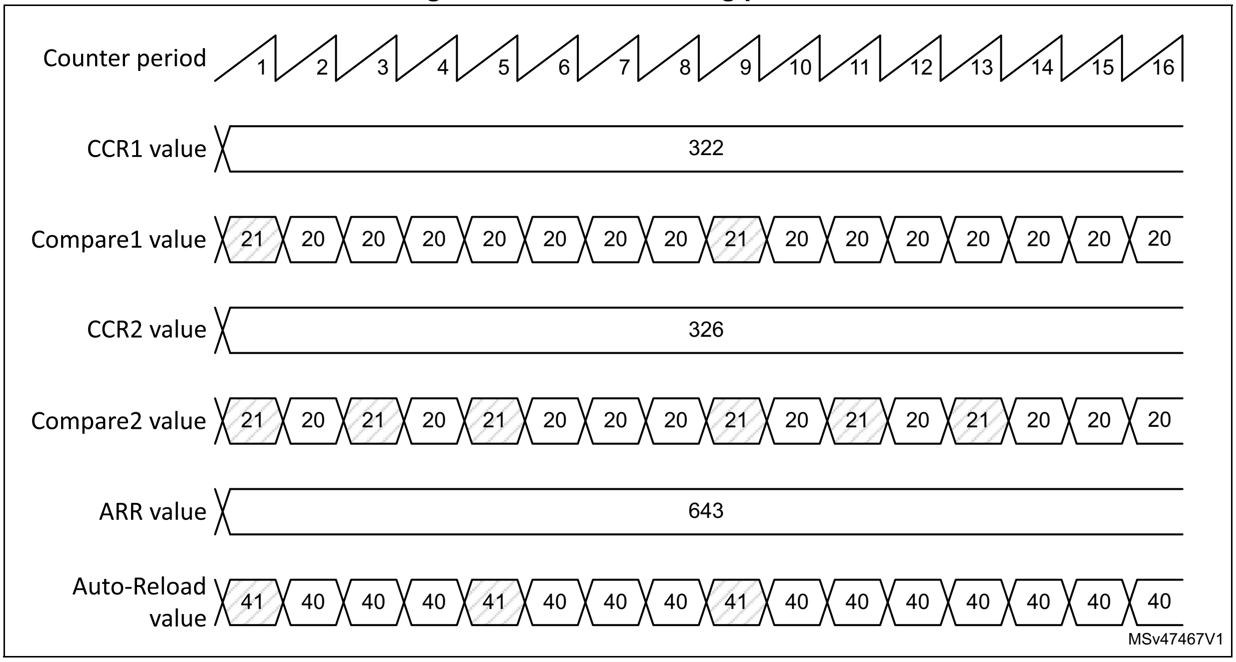
As shown on the *Figure 347* below, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 347. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in the *Figure 348* below.

Figure 348. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in the *Table 301* below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 301. 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 301. 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	-

31.3.12 Complementary outputs and dead-time insertion

The TIM16/TIM17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (such as intrinsic delays of level-shifters and 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 306: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

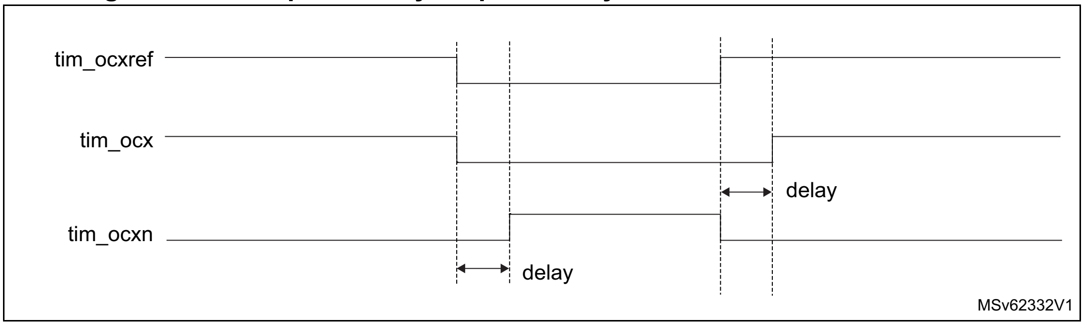
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform tim_ocxref, it generates two outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- 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. (in these examples CCxP = 0, CCxNP = 0, MOE = 1, CCxE = 1 and CCxNE = 1)

Figure 349. 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 350*.

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 350. Asymmetrical deadtime

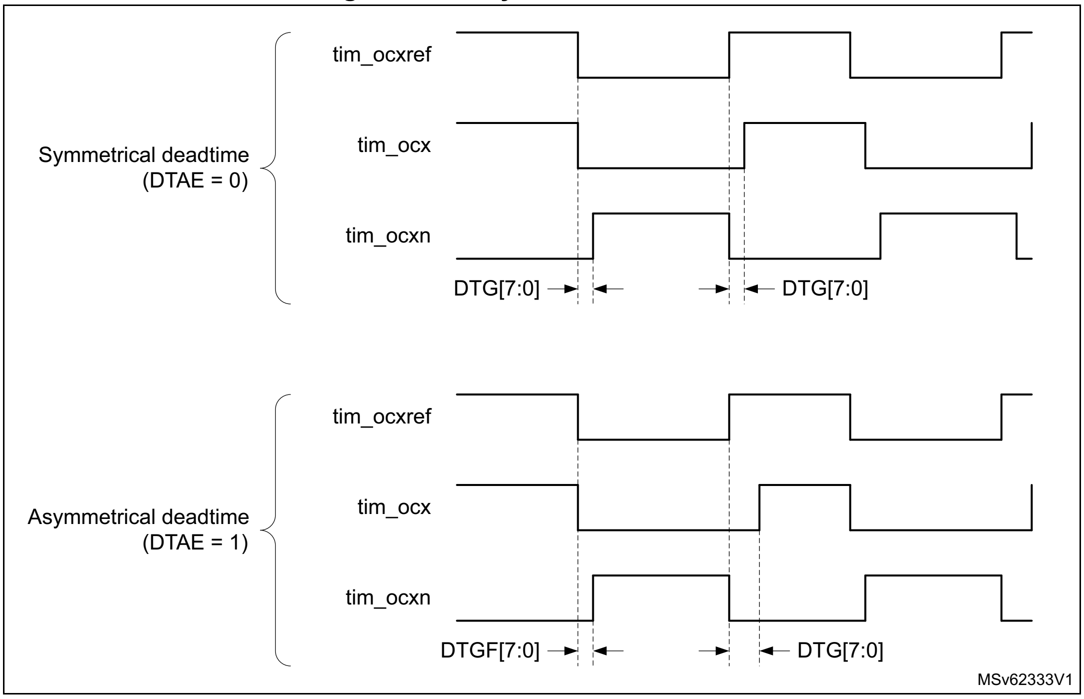


Figure 351. Dead-time waveforms with delay greater than the negative pulse.

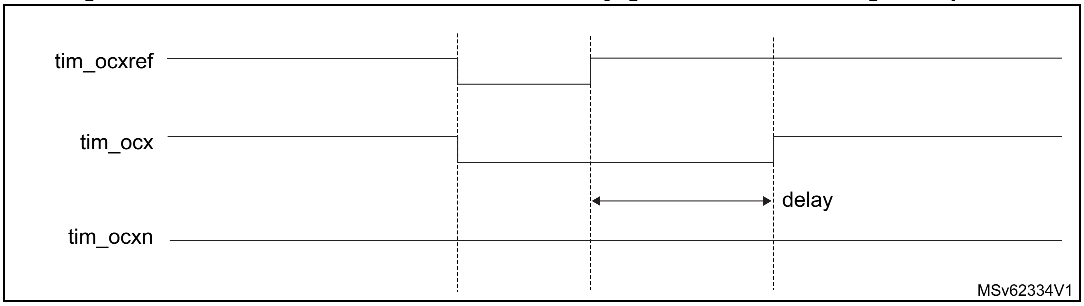
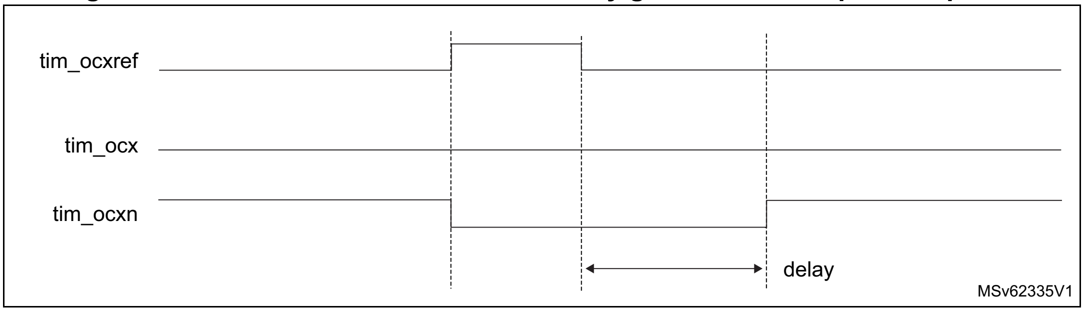


Figure 352. Dead-time waveforms with delay greater than the positive pulse.



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 31.6.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)* for delay calculation.

Redirecting tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be redirected 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.

31.3.13 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 (such as clock failure, ECC/parity, and 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 306: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)* 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 one APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait one 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 have:

- 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.

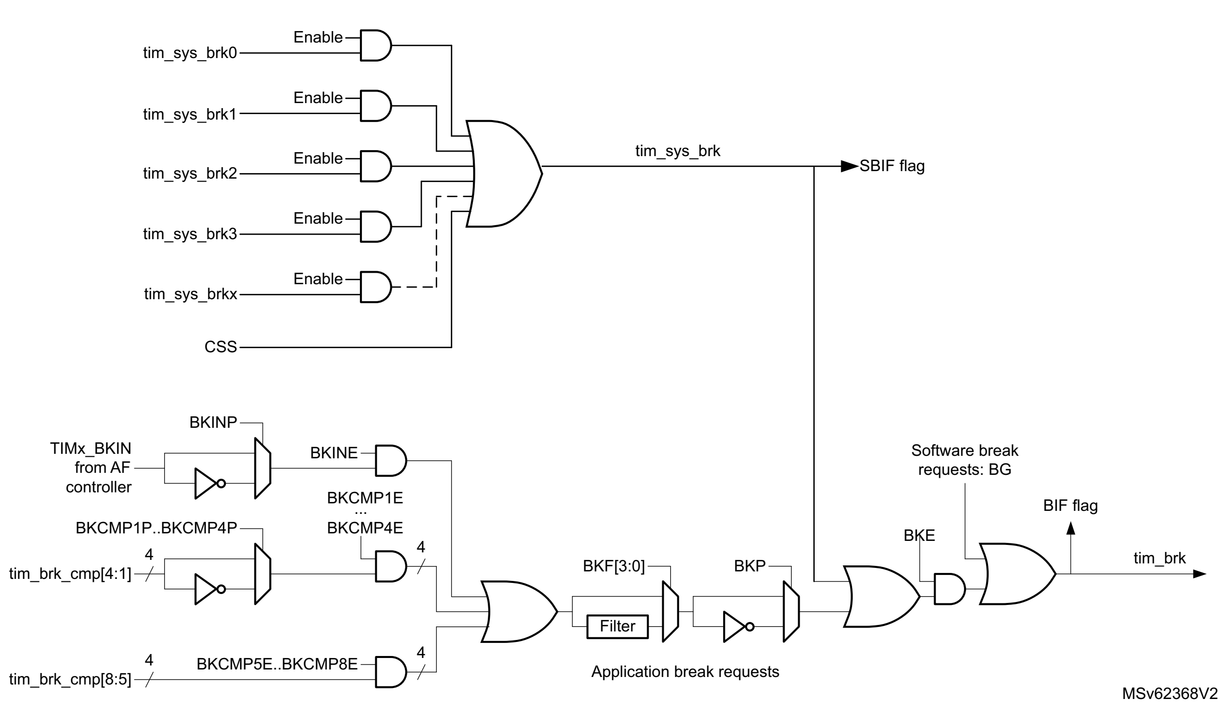
The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx_AF1 register.

The sources for break (tim_brk) channel are:

- External sources connected to one of the TIM_BKIN pins (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 31.3.2: 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 31.3.2: 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 353* below.

Figure 353. 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.

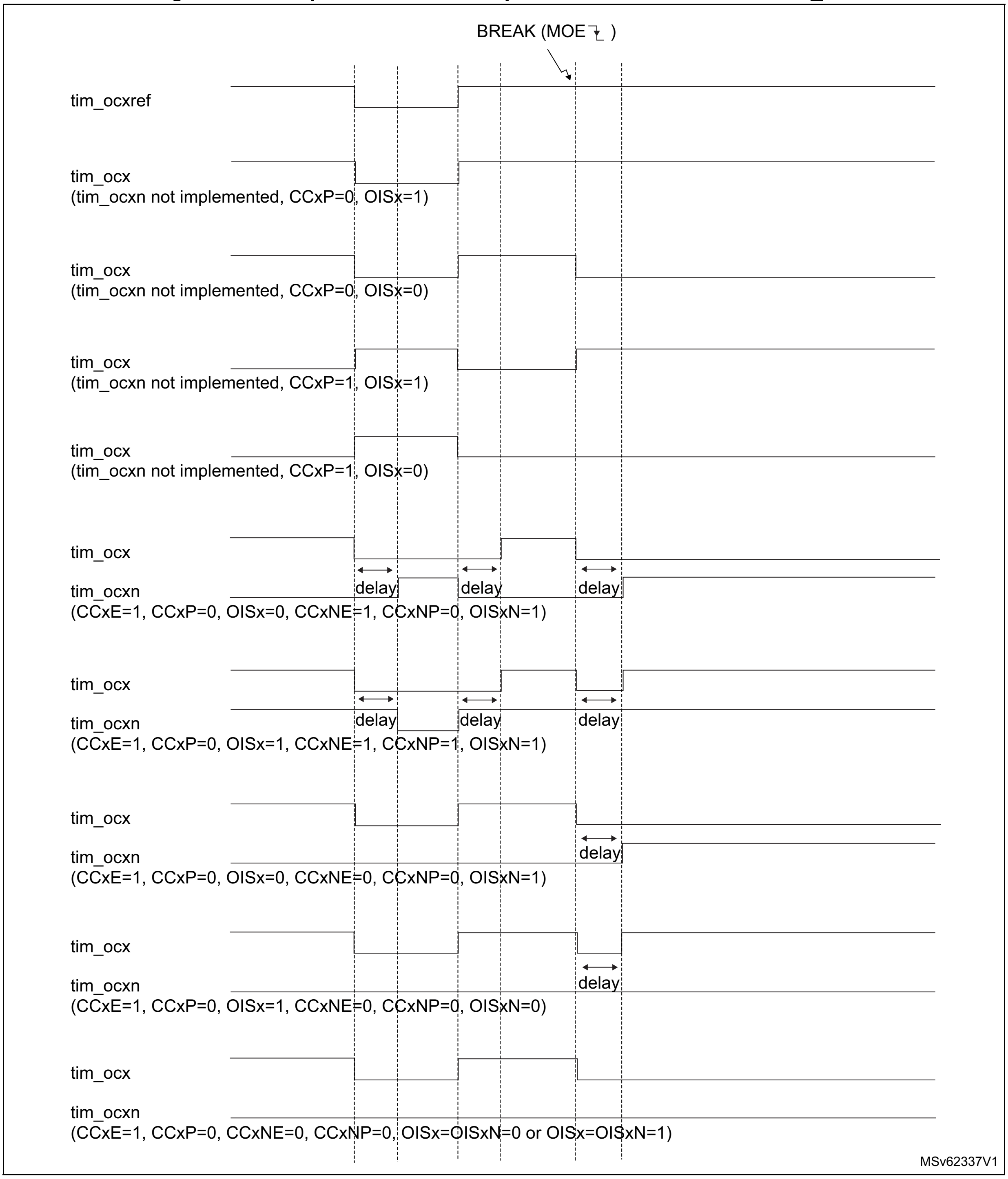
The break inputs are acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

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 31.6.14: 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 354* shows an example of behavior for the outputs in response to a break.

Figure 354. Output behavior in response to a break event on tim_brk



31.3.14 Bidirectional break input

The TIM16/TIM17 are featuring bidirectional break I/Os, as represented on *Figure 355*.

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 (for example 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 restarted 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 302*).

Table 302. 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 rearming break circuitry

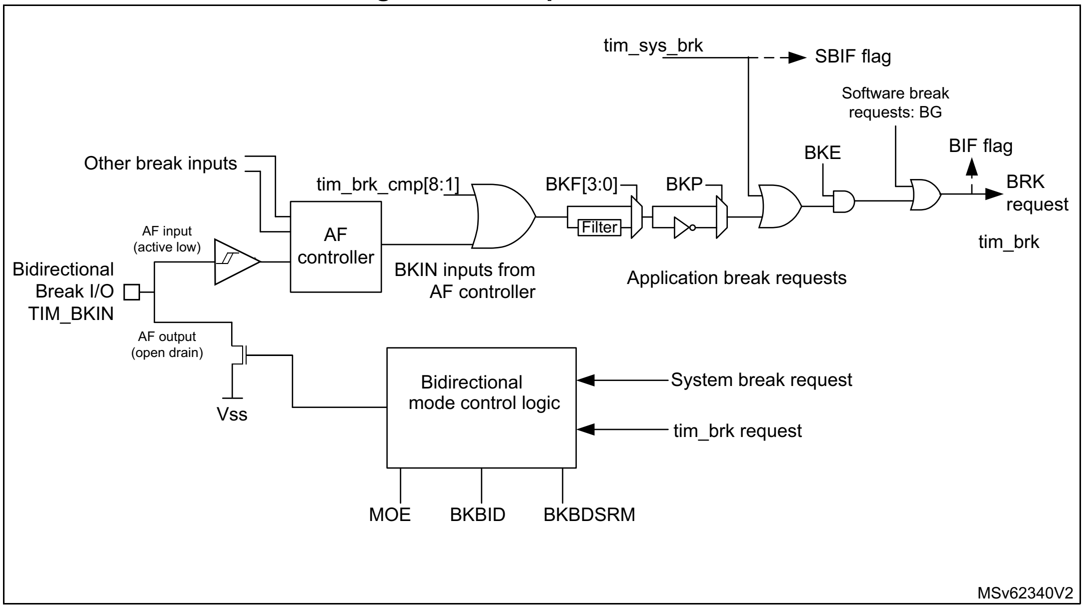
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

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 rearming).
- 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 355. Output redirection

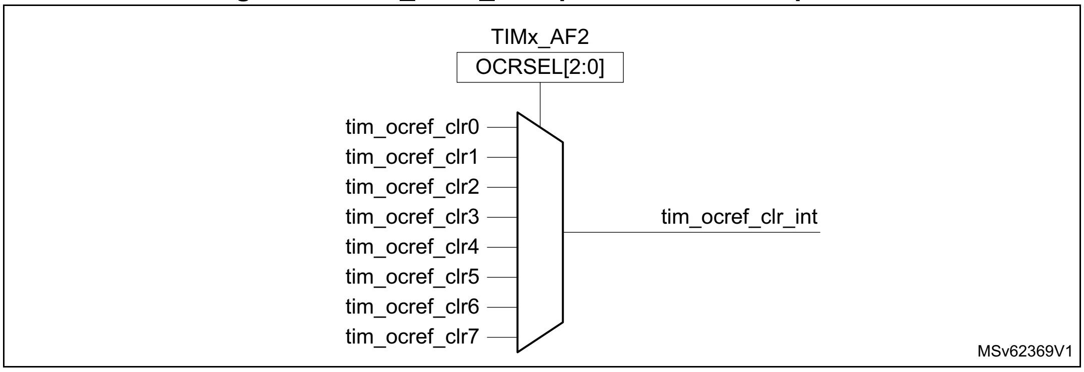


31.3.15 Clearing the tim_ocxref signal on an external event

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 transition to the active state, on the following PWM cycle. 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 509* below.

Figure 356. tim_ocref_clr input selection multiplexer



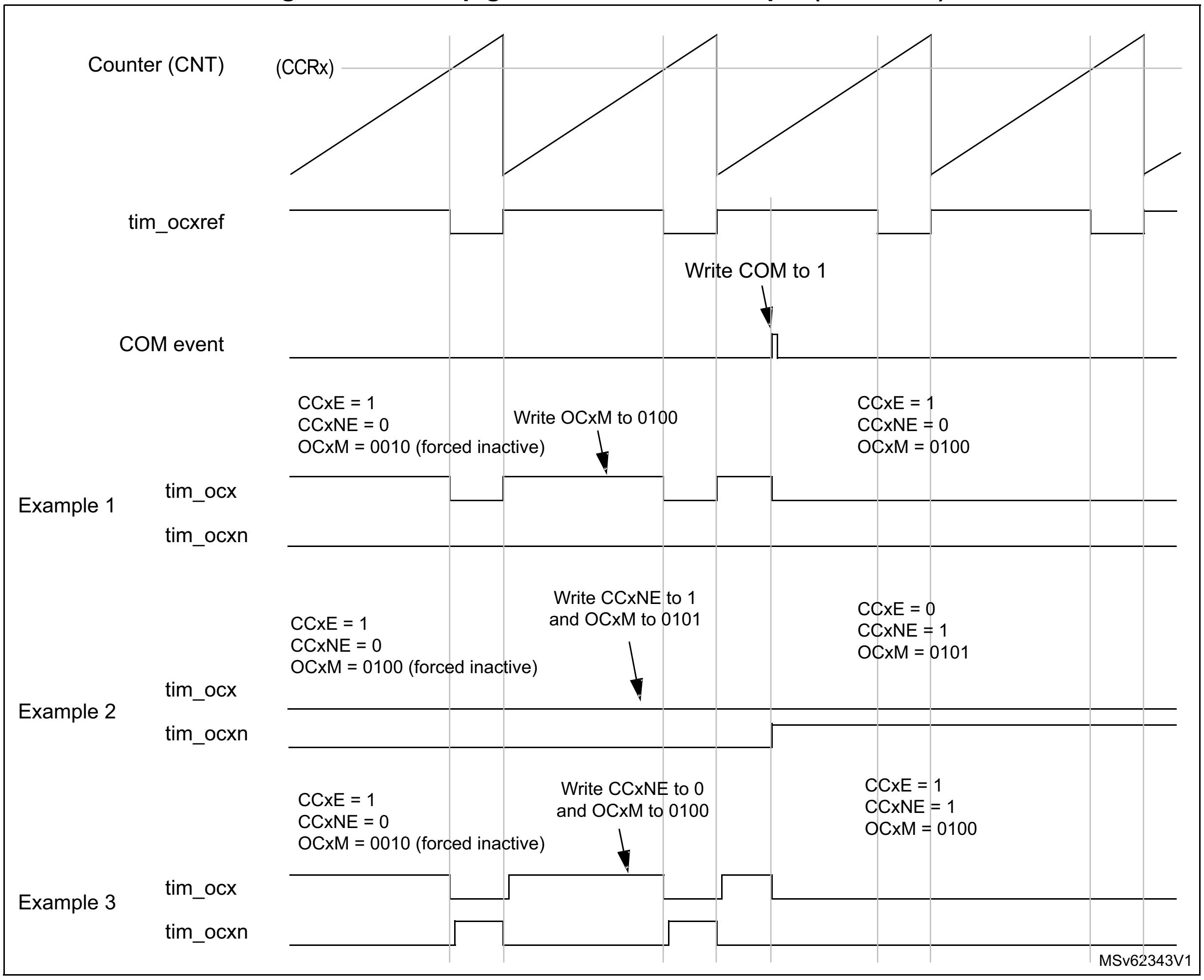
31.3.16 6-step PWM generation

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 357* describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 357. 6-step generation, COM example (OSSR = 1)



31.3.17 One-pulse mode

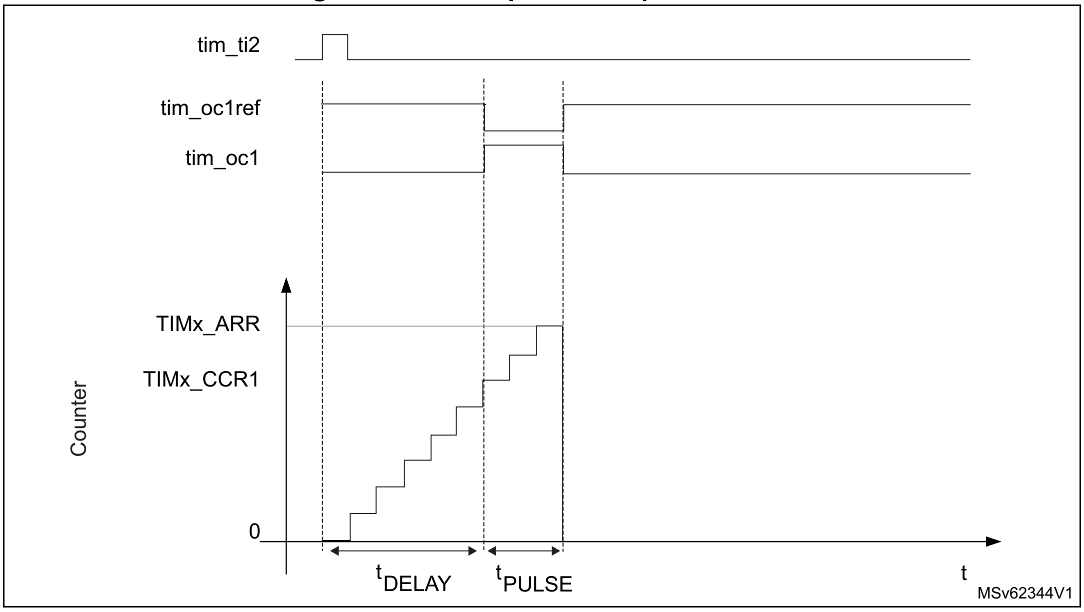
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

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 358. 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 autoreload 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 autoreload 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 autoreload 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 one 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 autoreload 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 that can be obtained.

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.

31.3.18 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.

31.3.19 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 two 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 four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

31.3.20 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 reprogram 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 accesses 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 31.6.20: TIMx DMA control register (TIMx_DCR)(x = 16 to 17)* for details).

For example, the timer DMA burst feature can 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 must 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.

31.3.21 TIM16/TIM17 DMA requests

The TIM16/TIM17 can generate a DMA request, as shown in *Table 303*.

Table 303. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE

31.3.22 Debug mode

When the microcontroller enters debug mode (CPU1 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.

31.4 TIM16/TIM17 low-power modes

Table 304. Effect of low-power modes on 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.

31.5 TIM16/TIM17 interrupts

The TIM16/TIM17 can generate multiple interrupts, as shown in *Table 305*.

Table 305. 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	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Commutation (COM)	COMIF	COMIE	write 0 in COMIF	Yes	No
	Trigger	TIF	TIE	write 0 in TIF	Yes	No
	Break	BIF	BIE	write 0 in BIF	Yes	No

31.6 TIM16/TIM17 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

31.6.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17)

Address offset: 0x00

Reset value: 0x0000

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

31.6.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
						rw	rw					rw	rw		rw

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

Bit 9 **OIS1N**: Output Idle state 1 (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.

31.6.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
						rw	rw	rw		rw				rw	rw

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

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

31.6.4 TIMx status register (TIMx_SR)(x = 16 to 17)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
						rc_w0		rc_w0		rc_w0				rc_w0	rc_w0

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

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.

0: No overcapture has been detected

1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the 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.

31.6.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
								w		w				w	w

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

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMG:** Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

31.6.6 TIMx capture/compare mode register 1 (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 (for example 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 bitfield 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 bitfield 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 bitfield 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).

31.6.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example 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. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

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, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” 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 bitfield 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).

31.6.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: 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).

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).

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).

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 306* for details.

Table 306. Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)

<table>
<tr><th colspan="5">Control bits</th><th colspan="2">Output states(1)</th></tr>
<tr><th>MOE bit</th><th>OSSI bit</th><th>OSSR bit</th><th>CC1E bit</th><th>CC1NE bit</th><th>tim_oc1 output state</th><th>tim_oc1n output state</th></tr>
<tr><td rowspan="6">1</td><td rowspan="6">X</td><td>X</td><td>0</td><td>0</td><td colspan="2">Output Disabled (not driven by the timer: Hi-Z)
tim_oc1 = 0
tim_oc1n = 0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Output Disabled (not driven by the timer: Hi-Z)
tim_oc1 = 0</td><td>tim_oc1ref + Polarity
tim_oc1n = tim_oc1ref XOR CC1NP</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>tim_oc1ref + Polarity
tim_oc1 = tim_oc1ref XOR CC1P</td><td>Output Disabled (not driven by the timer: Hi-Z)
tim_oc1n = 0</td></tr>
<tr><td>X</td><td>1</td><td>1</td><td>tim_oc1ref + Polarity + dead-time</td><td>Complementary to tim_oc1ref (not tim_oc1ref) + Polarity + dead-time</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Off-State (output enabled with inactive state)
tim_oc1 = CC1P</td><td>tim_oc1ref + Polarity
tim_oc1n = tim_oc1ref XOR CC1NP</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>tim_oc1ref + Polarity
tim_oc1 = tim_oc1ref XOR CC1P</td><td>Off-State (output enabled with inactive state)
tim_oc1n = CC1NP</td></tr>
<tr><td rowspan="5">0</td><td>0</td><td rowspan="5">X</td><td>X</td><td>X</td><td colspan="2" rowspan="2">Output disabled (not driven by the timer: Hi-Z)</td></tr>
<tr><td rowspan="4">1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td colspan="2" rowspan="3">Off-State (output enabled with inactive state)
Asynchronously: 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</td></tr>
<tr><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table>

1. When both outputs of a channel are not used (control taken over by the 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.

31.6.9 TIMx counter (TIMx_CNT)(x = 16 to 17)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved.

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

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

31.6.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (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”).

31.6.11 TIMx auto-reautoreload 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 31.3.3: Time-base unit on page 1152* 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.

31.6.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17)

Address offset: 0x30

Reset value: 0x0000

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

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

Bits 7:0 **REP[7:0]**: Repetition counter 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.

31.6.13 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.

<u>Non-dithering mode (DITHEN = 0)</u>

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

<u>Dithering mode (DITHEN = 1)</u>

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).

<u>Non-dithering mode (DITHEN = 0)</u>

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

<u>Dithering mode (DITHEN = 1)</u>

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

31.6.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)

Address offset: 0x44

Reset value: 0x0000 0000

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

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 bitfield 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 31.6.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)*).

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 31.6.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)*).

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 31.6.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)*).

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 bitfield 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}=125 ns (8 MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

31.6.15 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 bitfield 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} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

31.6.16 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 31.3.2: TIM16/TIM17 pins and internal signals* for interconnects list.

31.6.17 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 31.3.2: 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. 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).

31.6.18 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 31.3.2: 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.

31.6.19 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.	HSE32EN	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 297: 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.

31.6.20 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-bitfield 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-bitfield 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.

31.6.21 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).

31.6.22 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

Table 307. 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 307. 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: Memory organization* for the register boundary addresses.

32 Low-power timer (LPTIM)

32.1 Introduction

The LPTIM is a 16-bit timer that benefits from the ultimate developments in power consumption reduction. Thanks to its diversity of clock sources, the LPTIM is able to keep running in all power modes except for Standby mode. Given its capability to run even with no internal clock source, the LPTIM can be used as a “Pulse Counter” which can be useful in some applications. The LPTIM capability to wake up the system from low-power modes, makes it suitable to realize “Timeout functions” with extremely low power consumption.

The LPTIM introduces a flexible clock scheme that provides the needed functionalities and performance, while minimizing the power consumption.

32.2 LPTIM main features

- 16-bit upcounter
- 3-bit prescaler with 8 possible dividing factors (1,2,4,8,16,32,64,128)
- Selectable clock
  - Internal clock sources: 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

32.3 LPTIM implementation

Table 308. LPTIM features

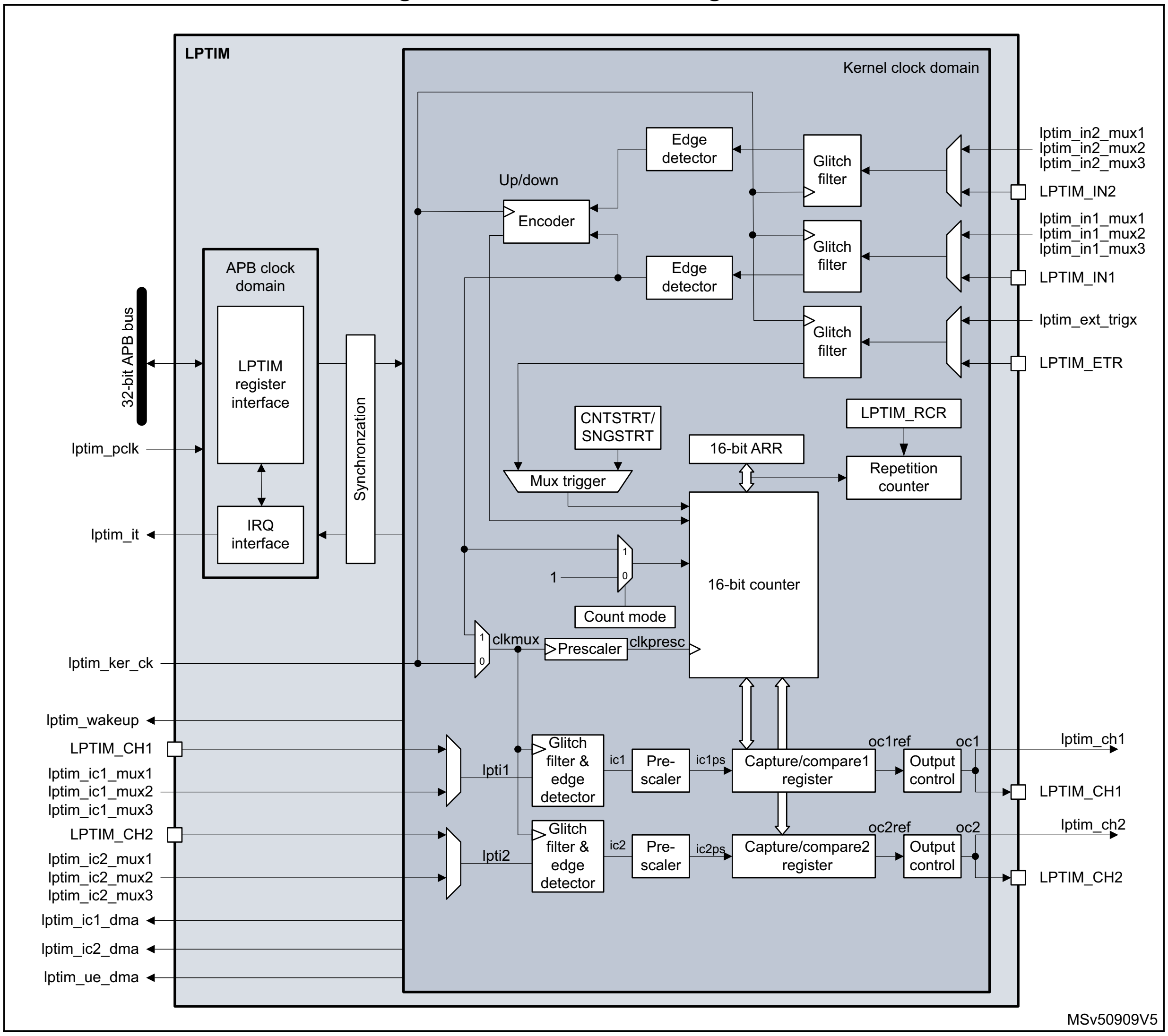
LPTIM modes/features(1)	LPTIM1	LPTIM2(2)
INx	1 to 2	1 to 2
CHx	1 to 2	1 to 2
ETR	X	X
Encoder mode	X	X
PWM mode	X	X
Input Capture	X	X
Number of channels	2	2
Number of DMA requests	3	3
Wake-up from Stop mode	X	X
Autonomous mode	X	X

1. Note ‘X’ = supported, ‘-’ = not supported.
2. Available only on STM32WBA52/54/55xx devices.

32.4 LPTIM functional description

32.4.1 LPTIM block diagram

Figure 359. LPTIM block diagram(1)



1. Some I/Os may not be available, refer to *Section 32.4.2: LPTIM pins and internal signals*.

32.4.2 LPTIM pins and internal signals

The following tables provide the list of LPTIM pins and internal signals, respectively.

Table 309. LPTIM1/2 input/output pins

Pin name	Pin type	Description
LPTIM_IN1	Digital input	LPTIM Input 1 from GPIO pin on mux input 0
LPTIM_IN2	Digital input	LPTIM Input 2 from GPIO pin on mux input 0

Table 309. LPTIM1/2 input/output pins (continued)

Pin name	Pin type	Description
LPTIM_ETR	Digital input	LPTIM external trigger GPIO pin
LPTIM_CH1	Digital input/output	LPTIM channel 1 input/output GPIO pin
LPTIM_CH2	Digital input/output	LPTIM channel 2 input/output GPIO pin

Table 310. LPTIM1/2 internal signals

Signal name	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	Digital input	Internal LPTIM input 2 connected to mux input 1
lptim_in2_mux2	Digital input	Internal LPTIM input 2 connected to mux input 2
lptim_in2_mux3	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 wake-up 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
lptim_ch1	Digital output	LPTIM channel 1 DMA trigger
lptim_ch2	Digital output	LPTIM channel 2 DMA trigger

32.4.3 LPTIM input and trigger mapping

The LPTIM external trigger and input connections are detailed hereafter.

Table 311. LPTIM1/2 external trigger connections

TRIGSEL	External trigger	
	LPTIM1	LPTIM2
lptim_ext_trig0	LPTIM1_ETR	LPTIM2_ETR
lptim_ext_trig1	rtc_alra_trg	rtc_alra_trg
lptim_ext_trig2	rtc_alrb_trg	rtc_alrb_trg
lptim_ext_trig3	tamp_trg1	tamp_trg1
lptim_ext_trig4	tamp_trg2	gpdma_ch0_tc
lptim_ext_trig5	Reserved	gpdma_ch4_tc
lptim_ext_trig6	comp1_out(1)	comp1_out(1)
lptim_ext_trig7	comp2_out(1)	comp2_out(1)

1. Only available on STM32WBA54xx and STM32WBA55xx devices.

Table 312. LPTIM1/2 input 1 connections

lptim_in1_mux	LPTIM1	LPTIM2
lptim_in1_mux0	LPTIM1_IN1	LPTIM2_IN1
lptim_in1_mux1	comp1_out(1)	comp1_out(1)
lptim_in1_mux2..3	Reserved	Reserved

1. Only available on STM32WBA54xx and STM32WBA55xx devices.

Table 313. LPTIM1/2 input 2 connections

lptim_in2_mux	LPTIM1	LPTIM2
lptim_in2_mux0	LPTIM1_IN2	LPTIM2_IN2
lptim_in2_mux1	comp2_out(1)	comp2_out(1)
lptim_in2_mux2..3	Reserved	Reserved

1. Only available on STM32WBA54xx and STM32WBA55xx devices.

Table 314. LPTIM1/2 input capture 1 connections

lptim_ic1_mux	LPTIM1	LPTIM2
lptim_ic1_mux0	LPTIM1_CH1	LPTIM2_CH1
lptim_ic1_mux1	comp1_out(1)	comp1_out(1)
lptim_ic1_mux2	comp2_out(1)	comp2_out(1)
lptim_ic1_mux3	Reserved	Reserved

1. Only available on STM32WBA54xx and STM32WBA55xx devices.

Table 315. LPTIM1/2 input capture 2 connections

lptim_ic2_mux	LPTIM1	LPTIM2
lptim_ic2_mux0	LPTIM1_CH2	LPTIM2_CH2
lptim_ic2_mux1	LSI	HSI16 / 256
lptim_ic2_mux2	LSE	Reserved
lptim_ic2_mux3	Reserved	Reserved

32.4.4 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 can run in one of the following 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 must also be provided (first configuration). In this case, the internal clock signal frequency must be at least four times higher than the external clock signal frequency.

32.4.5 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 must first be provided to the LPTIM. This is necessary to guarantee the proper operation of the filters.

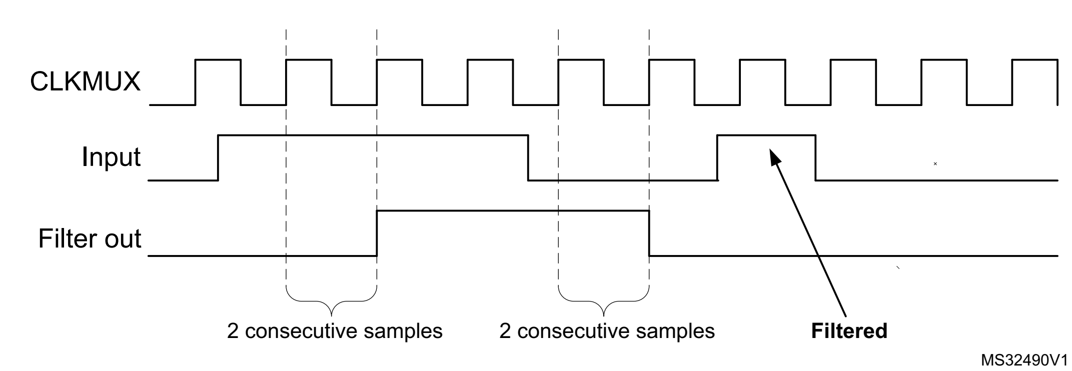
The digital filters are divided into three groups:

- 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 is detected on one of the LPTIM inputs to consider a signal level change as a valid transition. *Figure 360* shows an example of glitch filter behavior in case of a two consecutive samples programmed.

Figure 360. 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 this case, an external analog filter can be used to protect the LPTIM external inputs against glitches.

32.4.6 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] field. The table below lists all the possible division ratios:

Table 316. Prescaler division ratios

Programming	Dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

32.4.7 Trigger multiplexer

The LPTIM counter can be started either by software or after the detection of an active edge on one of the eight 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 eight trigger inputs is used to start the counter.

The external triggers are considered asynchronous signals for the LPTIM. 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.

32.4.8 Operating mode

The LPTIM features two operating modes:

- 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.

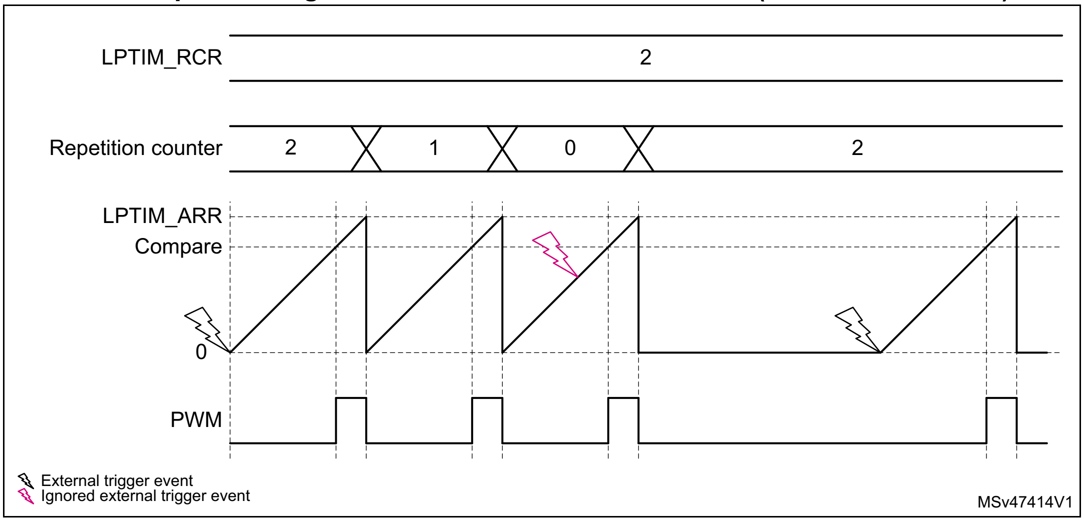
One-shot mode

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 361*.

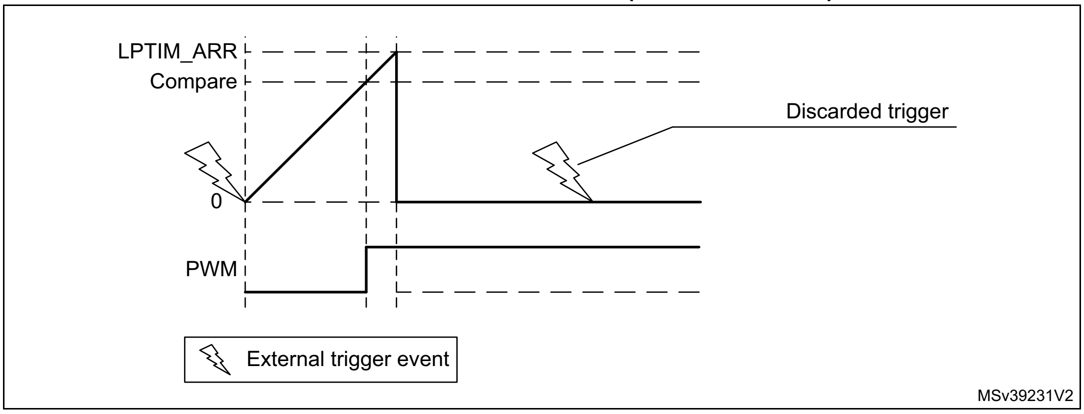
Figure 361. LPTIM output waveform, single-counting mode configuration when repetition register content is different than zero (with PRELOAD = 1)



- Set-once mode activated:

Note 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 362*.

Figure 362. 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.

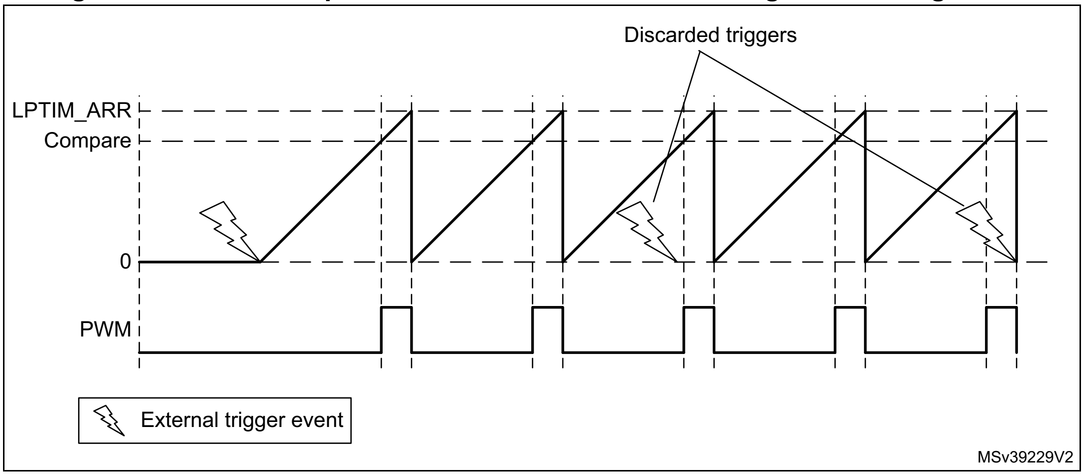
Continuous mode

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 363*.

In case of software start (TRIGEN[1:0] = 00), setting CNTSTRT starts the counter for continuous counting.

Figure 363. LPTIM output waveform, Continuous counting mode configuration



SNGSTRT and CNTSTRT bits can only be set when the timer is enabled (ENABLE bit 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.

32.4.9 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.

32.4.10 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 32.4.19: 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 the LPTIM_ARR register value to be strictly greater than the LPTIM_CCRx register value.

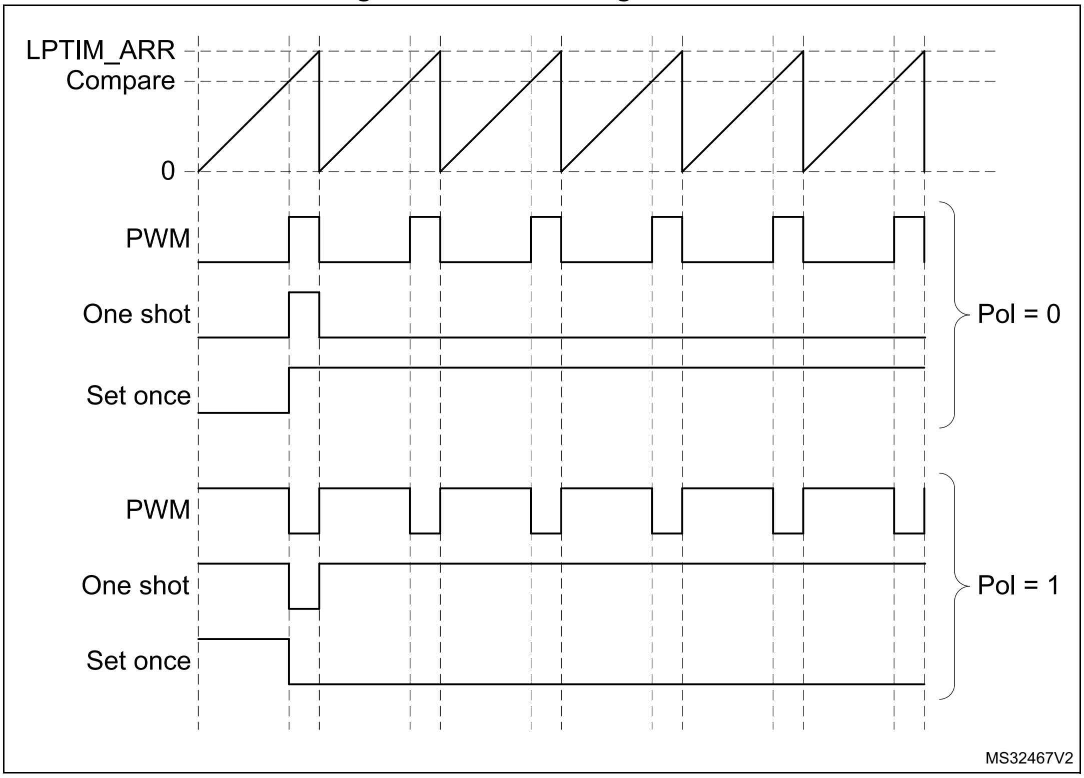
The LPTIM output waveform can be configured through the WAVE bit as follow:

- 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 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 two can be generated. *Figure 364* 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 CCxP bit.

Figure 364. Waveform generation



32.4.11 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.

32.4.12 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 must never exceed the frequency of the internal clock provided to the LPTIM. Also, the internal clock provided to the LPTIM must not be prescaled (PRESC[2:0] = 000).
- CKSEL = 1: the LPTIM is clocked by an external clock source

  COUNTMODE value is don’t care.

  In this configuration, the LPTIM has no need for an internal clock source (except if the glitch filters are enabled). The signal injected on the LPTIM external input1 is used as system clock for the LPTIM. This configuration is suitable for operation modes where no embedded oscillator is enabled.

For this configuration, the LPTIM counter can be updated either on rising edges or falling edges of the input1 clock signal but not on both rising and falling edges.

Since the signal injected on the LPTIM external input1 is also used to clock the LPTIM kernel logic, there is some initial latency (after the LPTIM is enabled) before the counter is incremented. More precisely, the first five active edges on the LPTIM external Input1 (after LPTIM is enable) are lost.

32.4.13 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 cycles is needed before the LPTIM is actually enabled.

The LPTIM_CFGR register must be modified only when the LPTIM is disabled.

32.4.14 Timer counter reset

In order to reset the content of LPTIM_CNT register, 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 triggered 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.

Note: The software should ensure that COUNRST bit is '0' before generating every synchronous reset.

- 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 must 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 must be applied only when there is enough insurance that no toggle occurs on the LPTIM Input1.
  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. The developer must make sure that these two mechanisms are used exclusively.

32.4.15 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. 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.

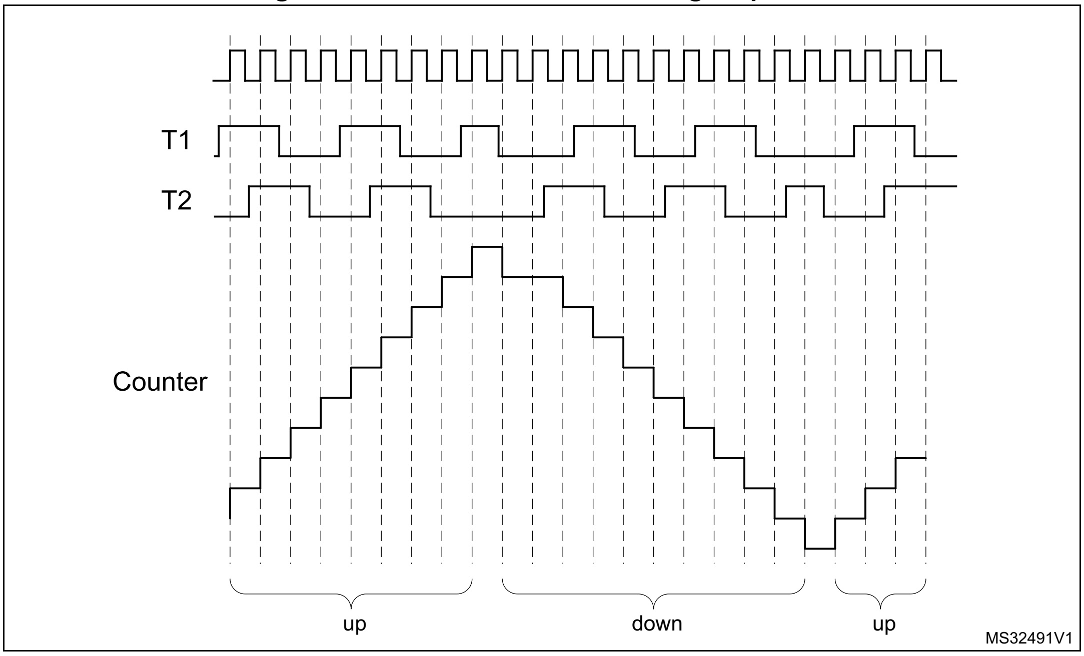
Table 317. Encoder counting scenarios

Active edge	Level on opposite signal (Input1 for Input2, Input2 for Input1)	Input1 signal		Input2 signal	
		Rising	Falling	Rising	Falling
Rising Edge (Encoder sub-mode 1)	High	Down	No count	Up	No count
	Low	Up	No count	Down	No count
Falling Edge (Encoder sub-mode 2)	High	No count	Up	No count	Down
	Low	No count	Down	No count	Up
Both Edges (Encoder sub-mode 3)	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 365. Encoder mode counting sequence



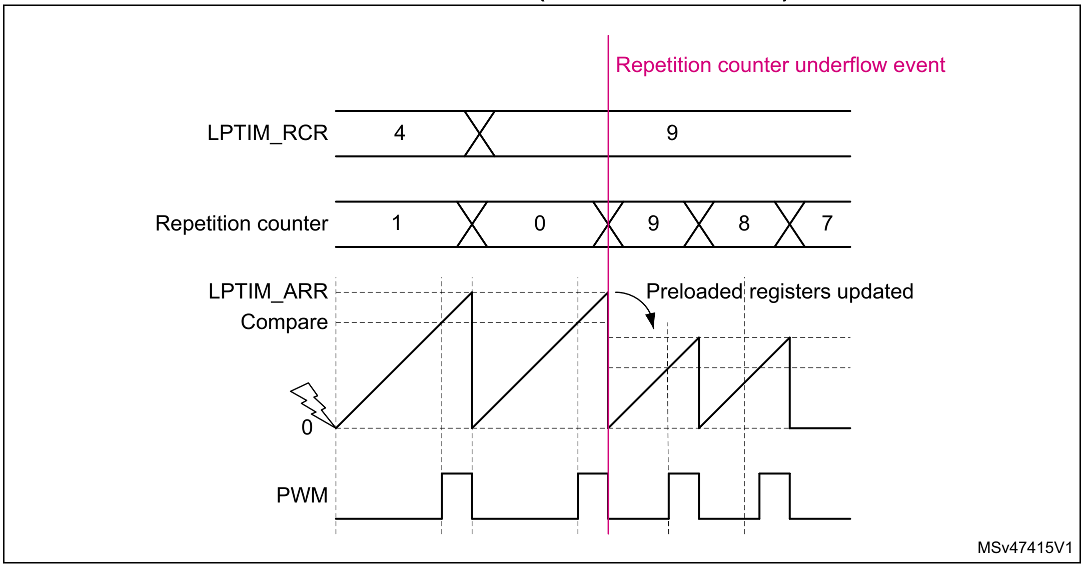
32.4.16 Repetition counter

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 366*.

Figure 366. 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 32.4.11: 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, this new content is propagated from the APB bus interface clock domain to the LPTIM kernel clock domain. The new written value is then 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 writes before the REPOK flag is set, lead 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.

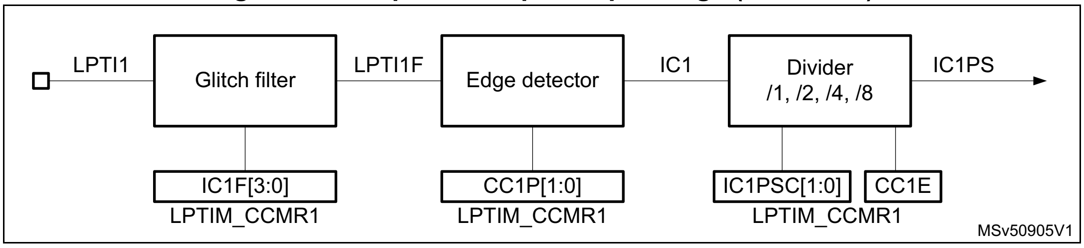
32.4.17 Capture/compare channels

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).

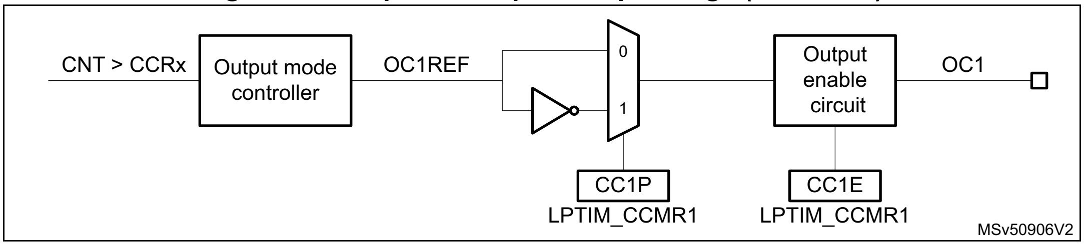
Figure 367. Capture/compare input stage (channel 1)



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 368. Capture/compare output stage (channel 1)



32.4.18 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 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 318*) 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 must 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 318*, 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 318. 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
5	0	2
	1	2
	2	2
	3	2

Table 318. Input capture Glitch filter latency (in counter step unit) (continued)

Prescaler PRESC[2:0]	ICxF[1:0]	Offset
6	0	2
	1	2
	2	2
	3	2
7	0	2
	1	2
	2	2
	3	2

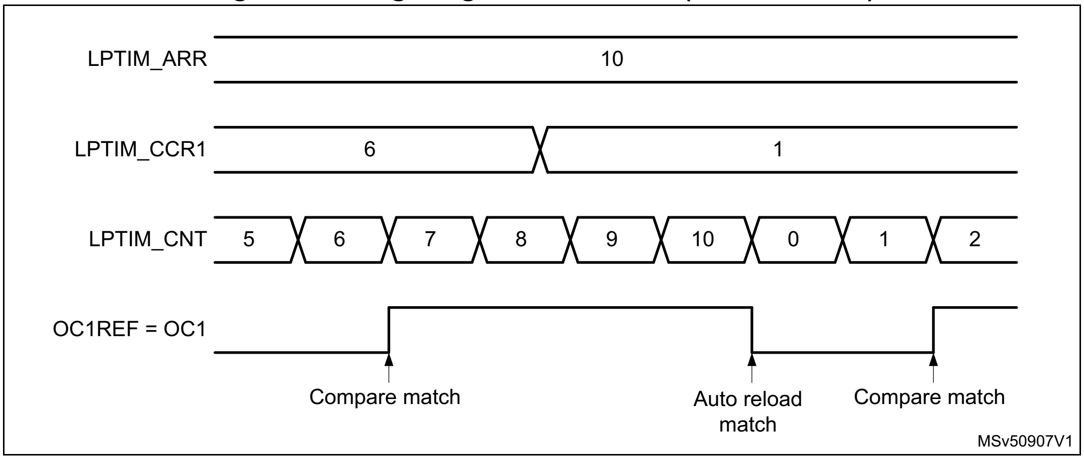
32.4.19 PWM mode

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 369 gives an example where the LPTIM channel 1 is configured in PWM mode with LPTIM_CCR1 = 6 then 1 and LPTIM_ARR=10.

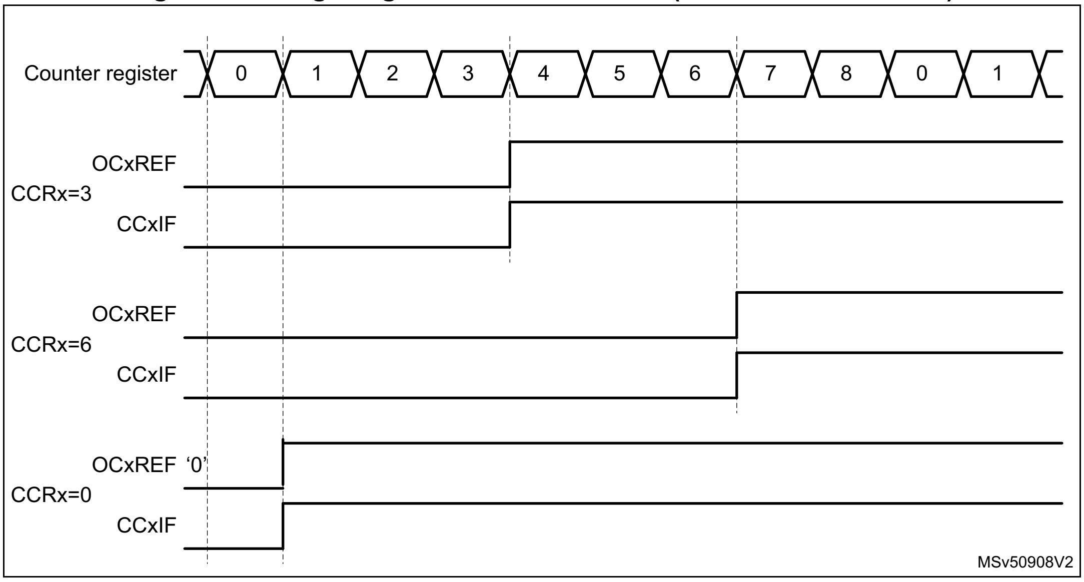
Figure 369. Edge-aligned PWM mode (PRELOAD = 1)



In the following example the reference PWM signal OCxREF is low as long as LPTIM_CNT ≤ LPTIM_CCRx else it becomes high.

Figure 370 shows some edge-aligned PWM waveforms in an example where LPTIM_ARR = 8.

Figure 370. 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.

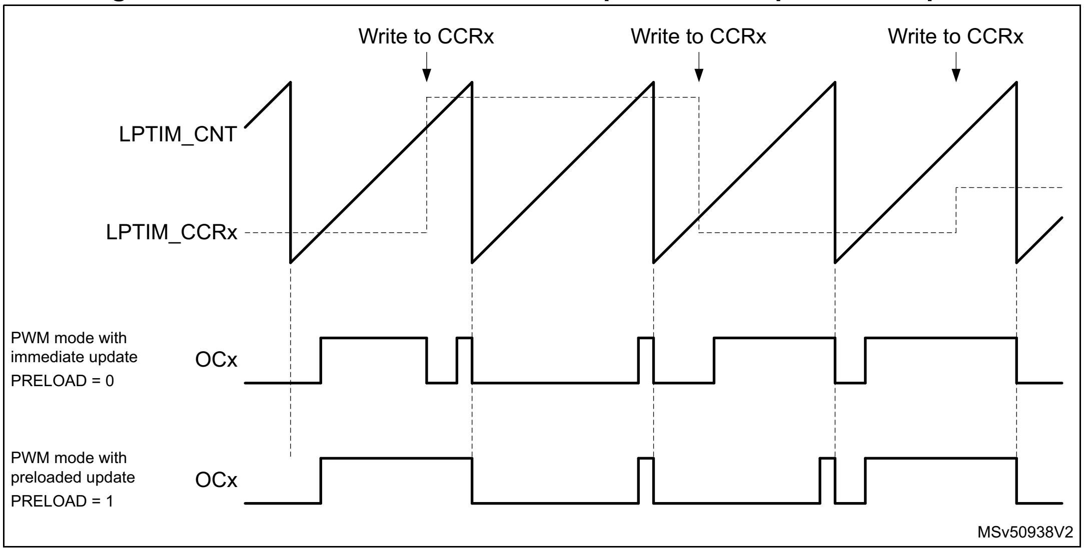
When the (PRELOAD = 0), the OCxREF signal level can be changed on-the-fly by software (or DMA) by updating the compare value in the LPTIM_CCRx register.

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 371* illustrates the behavior of the OCxREF signal level when PRELOAD = 0 and PRELOAD = 1.

Figure 371. 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 369 illustrates this behavior.

32.4.20 Autonomous mode

When clocked by oscillators available in this mode (refer to RCC), the LPTIM can operate in autonomous mode, remaining then 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 must 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 must 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 must 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 must 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.

32.4.21 DMA requests

The LPTIM can generate two categories of DMA requests:

- DMA requests used to retrieve the input-capture counter values
- DMA update requests used to re-program part of the LPTIM, 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 LPTIM_CCRx register. The captured values in LPTIM_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 LPTIM_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)*
- *or if the channel x is disabled (clear CCxE bit) and all the other channels are already disabled*

32.4.22 Debug mode

When the microcontroller enters debug mode (core halted), the LPTIM counter either continues to work normally or stops, depending on the timer dedicated bit configuration in the debug support (DBG) peripheral.

For further details, refer to section debug support (DBG).

32.5 LPTIM low-power modes

Table 319. Effect of low-power modes on the LPTIM

Mode	Description
Sleep	No effect. LPTIM interrupts cause the device to exit Sleep mode.
Stop	If the LPTIM is clocked by an oscillator available in Stop mode, LPTIM is functional and the interrupts cause the device to exit the Stop mode. The DMA requests are functional if the instance supports the autonomous mode (refer to *Section 32.3: LPTIM implementation*).
Standby	The LPTIM peripheral is powered down and must be reinitialized after exiting Standby mode.

32.6 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 320. 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	CCxIF	CCxIE	Write 1 to CCxCF	Yes	Yes
	Over-capture	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(2)	UP	UPIE	Write 1 to UPCF	Yes	Yes
	Direction change to down(2)	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 wake-up from Stop mode feature. Refer to *Section 32.3: LPTIM implementation*.
2. If LPTIM does not support Encoder mode feature, this event does not exist. Refer to *Section 32.3: LPTIM implementation*.

32.7 LPTIM registers

Refer to *Section 1.2: List of abbreviations for registers on page 70* for a list of abbreviations used in register descriptions.

The peripheral registers can only be accessed by words (32-bit).

32.7.1 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1, 2)

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.	DIER OK	Res.	Res.	Res.	Res.	CMP2 OK	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	REP OK	UE	DOWN	UP	ARR OK	CMP1 OK	EXT TRIG	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 32.3.

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 32.3.

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 32.3.

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 32.3.

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

32.7.2 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1, 2)

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 32.3.

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 32.3.

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 32.3.

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. The 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 32.3.

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 32.3.

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.

32.7.3 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1, 2)

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 32.3.

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 32.3.

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 32.3.

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 32.3.

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.

32.7.4 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1, 2)

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 32.3.

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 32.3.

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 32.3.

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 32.3.

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 32.3.

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.

32.7.5 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1, 2)

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.	CMP2 OKIE	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	REPOK IE	UEIE	DOWNI E	UPIE	ARRO KIE	CMP1 OKIE	EXT TRIGIE	ARRM IE	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 32.3.

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 32.3.

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 32.3.

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 32.3.

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 32.3.

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

32.7.6 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1, 2)

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 32.3.

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 32.3.

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 32.3.

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 32.3.

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 32.3.

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 32.3.

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 32.3.

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 32.3.

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

32.7.7 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	Res.	WAVE	TIMOUT	TRIGEN[1:0]		Res.
							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 32.3.

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 Reserved, must be kept at reset value.

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 32.4.3: 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 are 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 are 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 32.4.15: Encoder mode* for more details about Encoder 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).

32.7.8 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 can 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 must 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

32.7.9 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.

Caution: The LPTIM_CCR1 register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

32.7.10 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).

32.7.11 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, reading the LPTIM_CNT register may return unreliable values. In this case it is necessary to perform consecutive reads until two returned values are identical.

32.7.12 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 32.4.3: 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 32.4.3: 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 32.4.3: 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 32.4.3: LPTIM input and trigger mapping*.

32.7.13 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.

32.7.14 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 are 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 are 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).

32.7.15 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 32.3: LPTIM implementation*.

32.7.16 LPTIM register map

The following table summarizes the LPTIM registers.

Table 321. 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	LPTIMx_ISR (x = 1, 2) 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, 2) 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	LPTIMx_ICR (x = 1, 2) 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, 2) 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	LPTIMx_DIER (x = 1, 2) 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, 2) 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

Table 321. 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
0x00C	LPTIM_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC(2)	COUNTMODE	PRELOAD	Res.	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
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	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
0x034	LPTIM_CCR2(3)	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 32.3: LPTIM implementation*.
2. If LPTIM does not support Encoder mode feature, this bit is reserved. Refer to *Section 32.3: LPTIM implementation*.
3. If the LPTIM implements less than 2 channels this register is reserved. Refer to *Section 32.3: LPTIM implementation*.

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

33 Infrared interface (IRTIM)

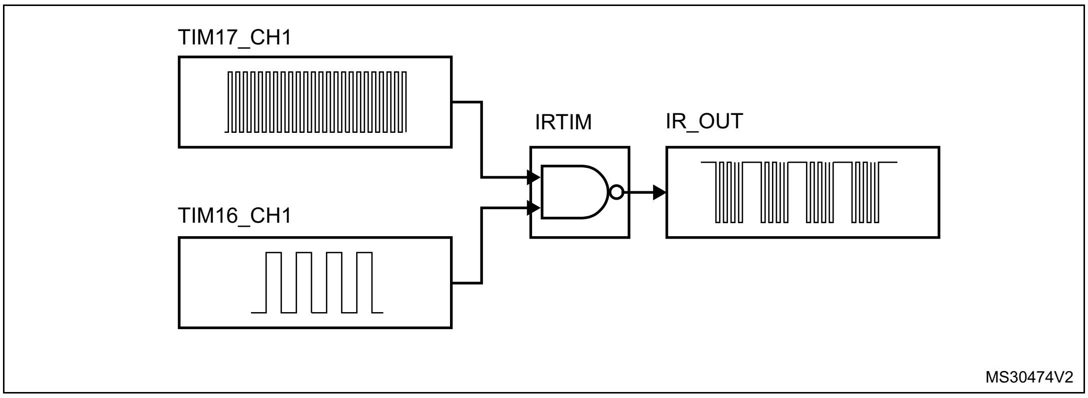
An infrared interface (IRTIM) for remote control is available on the device. It can be used with an infrared LED to perform remote control functions.

It uses internal connections with TIM16, and TIM17 as shown in *Figure 372*.

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 372. 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.

34 Independent watchdog (IWDG)

34.1 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 34.3* 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.

34.2 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

34.3 IWDG implementation

Table 322. IWDG features (1)

IWDG modes/features	IWDG
LSI used as IWDG kernel clock (iwdg_ker_ck)	X
Window function	X
Early wake-up interrupt generation	X
System reset generation (2)	X
Capability to work in system Stop	X
Capability to work in system Standby	X
Capability to generate an interrupt in system Stop	X
Capability to generate an interrupt in system Standby	X
Capability to be frozen when the microcontroller enters in Debug mode (3)	X
Option bytes to control the activity in Stop mode (4)	X
Option bytes to control the activity in Standby mode (5)	X
Option bytes to control the Hardware mode (6)	X

1. 'X' = supported, '-' = not supported.
2. Refer to the RCC section for additional information.
3. Controlled via DBG_IWDG_STOP in DBG section.

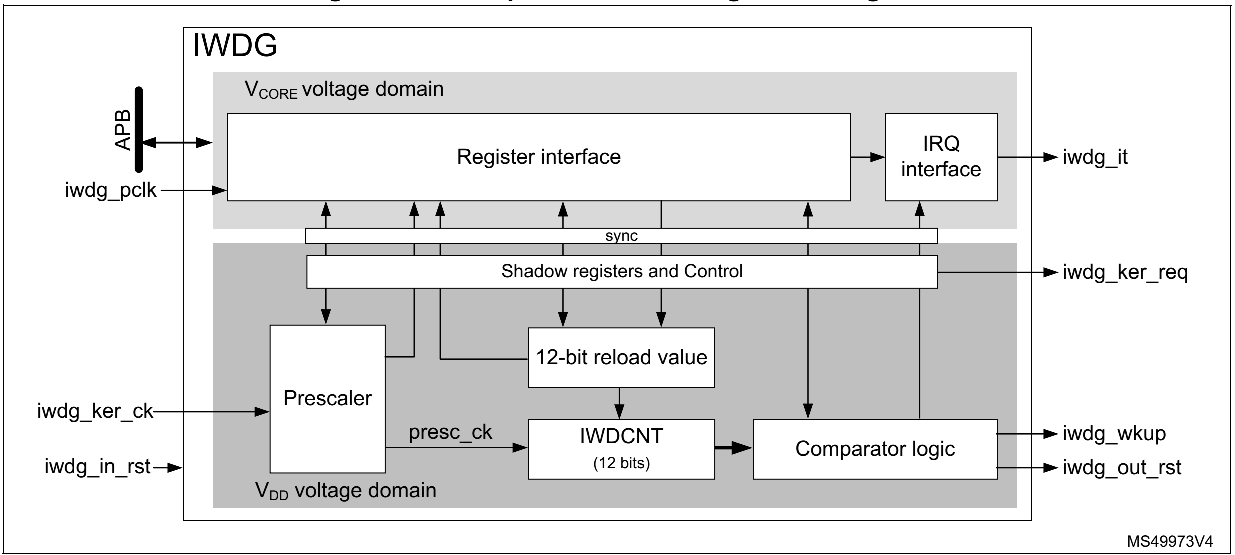
4. Controlled via the option byte IWDG_STOP in FLASH section.
5. Controlled via the option byte IWDG_STDBY in FLASH section.
6. Controlled via the option byte IWDG_SW in FLASH section.

34.4 IWDG functional description

34.4.1 IWDG block diagram

Figure 373 shows the functional blocks of the independent watchdog module.

Figure 373. 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 34.3* 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.

The table below gives the timing delays according to the actions performed on the IWDG: changing the prescaler, the timeout, the window or the early wake-up comparator values or doing a refresh.

Table 323. IWDG delays versus actions (1)

TD_{RVU}, TD_{PVU}		TD_{WVU}		TD_{EWU}		$TD_{Refresh}$		$TD_{RefAuto}$	
Min	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**
$5\ T_k$	$6\ T_k$	-	$6\ T_k$	-	$6\ T_k$	$2\ T_k$	$2\ T_k + T_p$	-	T_p

1. Tk represents a period of the kernel clock input, Tp represents a period of presc_ck.

34.4.2 IWDG internal signals

The list of IWDG internal signals is detailed in *Table 324*.

Table 324. 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 wake-up event
iwdg_it	Output	IWDG early wake-up interrupt

34.4.3 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 34.3* for details.

When the IWDG is enabled the ONF flag is set to 1.

When the IWDCNT reaches 0x000, a reset signal is generated (iwdg_out_rst asserted).

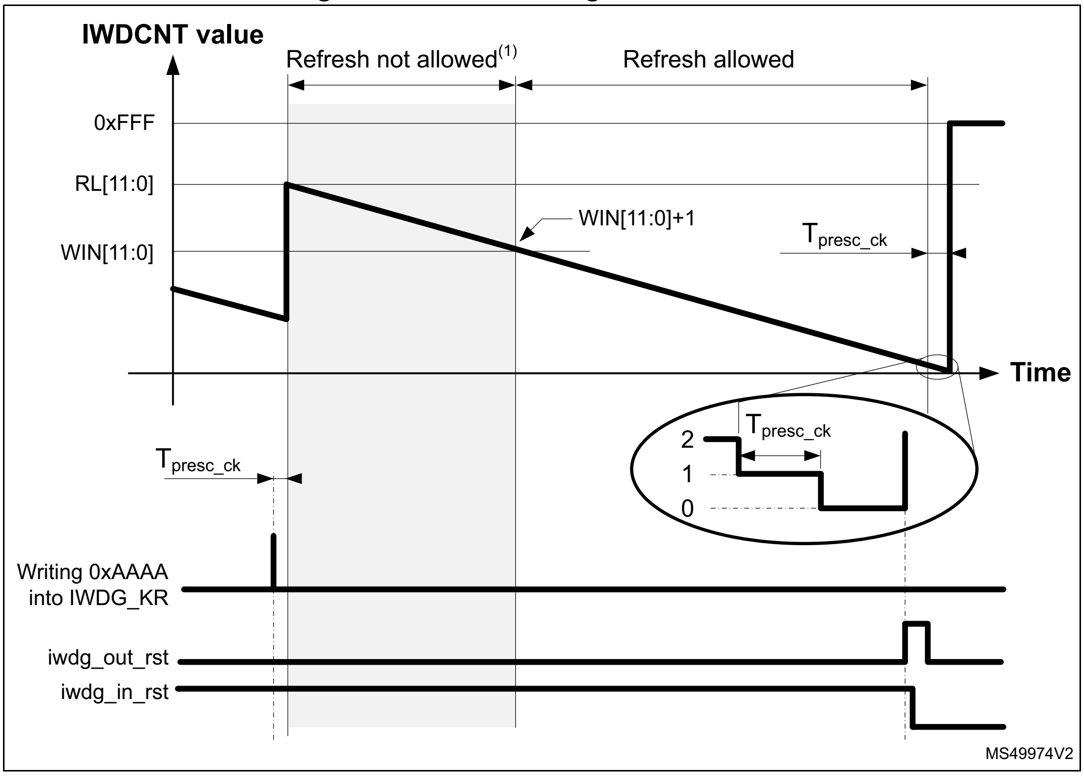
Whenever the key value 0x0000 AAAA is written in the *IWDG key register (IWDG_KR)*, the IWDG_RLR value is reloaded into the IWDCNT, and the watchdog reset is prevented.

Due to re-synchronization delays, the IWDG must be refreshed before the IWDCNT down-counter reaches 1.

Once started, the IWDG can be stopped only when it is reset (iwdg_in_rst asserted).

As shown in *Figure 374*, when the refresh command is executed, one period of presc_ck later, the IWDCNT is reloaded with the content of RL[11:0].

Figure 374. Reset timing due to timeout



1. If window option activated.

If the IWDG is not refreshed before the IWDCNT reaches 1, the IWDG generates a reset (iwdg_out_rst is asserted). In return, the RCC resets the IWDG (assertion of iwdg_in_rst) to clear the reset source.

34.4.4 Window option

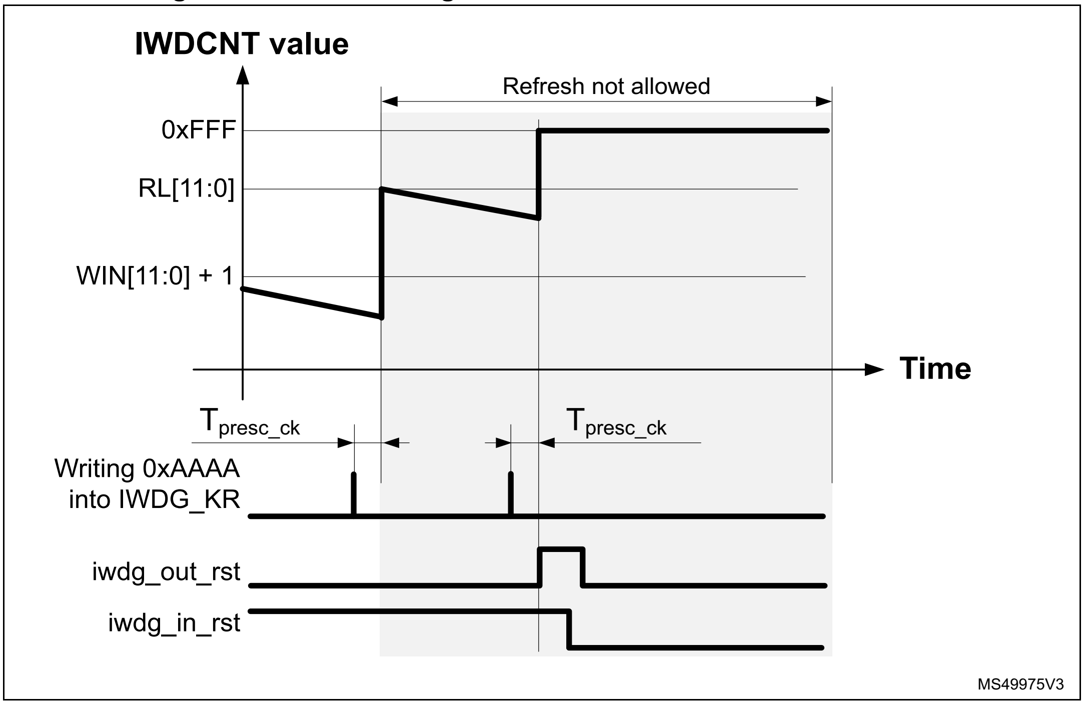
The IWDG can also work as a window watchdog, by setting the appropriate window in the *IWDG window register (IWDG_WINR)*.

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 375*, 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 375. 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 wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up 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.

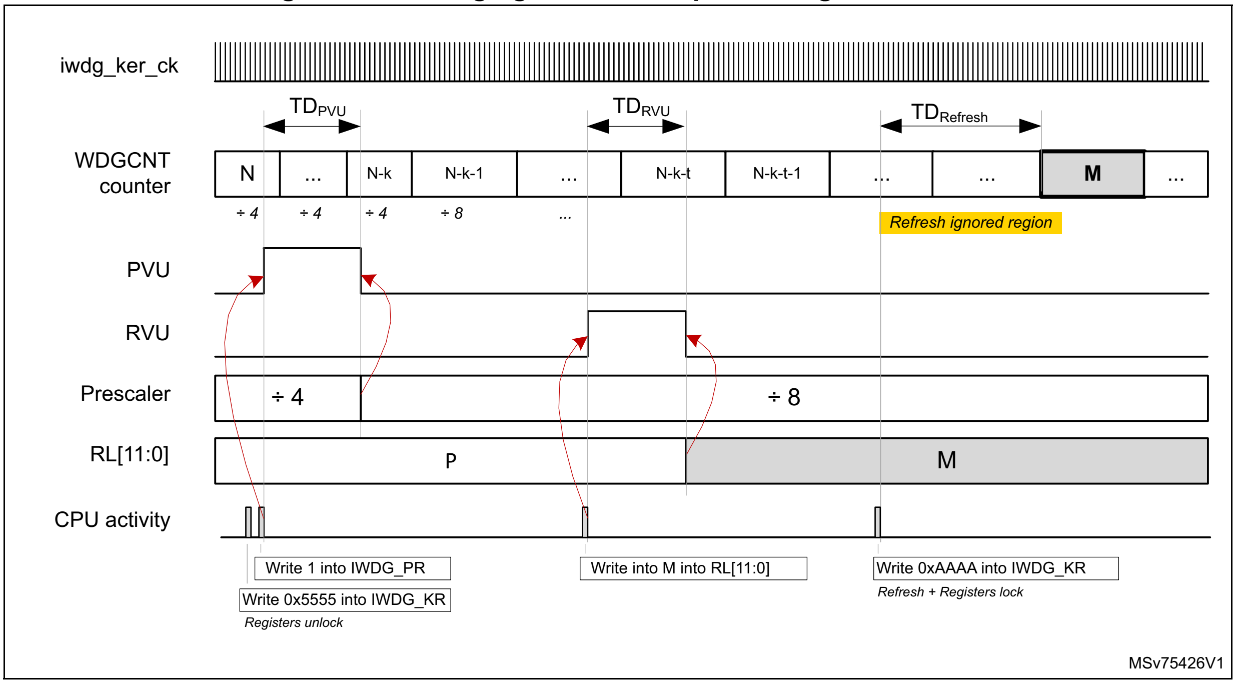
Configuring the IWDG when the window option is disabled

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 wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up 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)*.

The figure below shows a sequence example changing the prescaler, the reload value, and then performing a refresh.

Figure 376. Changing PR, RL, and performing a refresh(1)



1. Refer to *Table 323: IWDG delays versus actions* for details on timing values.

Note: When the new prescaler value is accepted by the IWDG (falling edge of PVU), the new division ratio is effective when the current division sequence is completed (N-k in the drawing).

Note: The timeout delay between the refresh (write 0xAAAA into IWDG_KR) and the watchdog reset is increased by TDRefresh.

If a refresh command is sent while the previous refresh command is not yet completed, this last refresh command is ignored.

Updating the window comparator

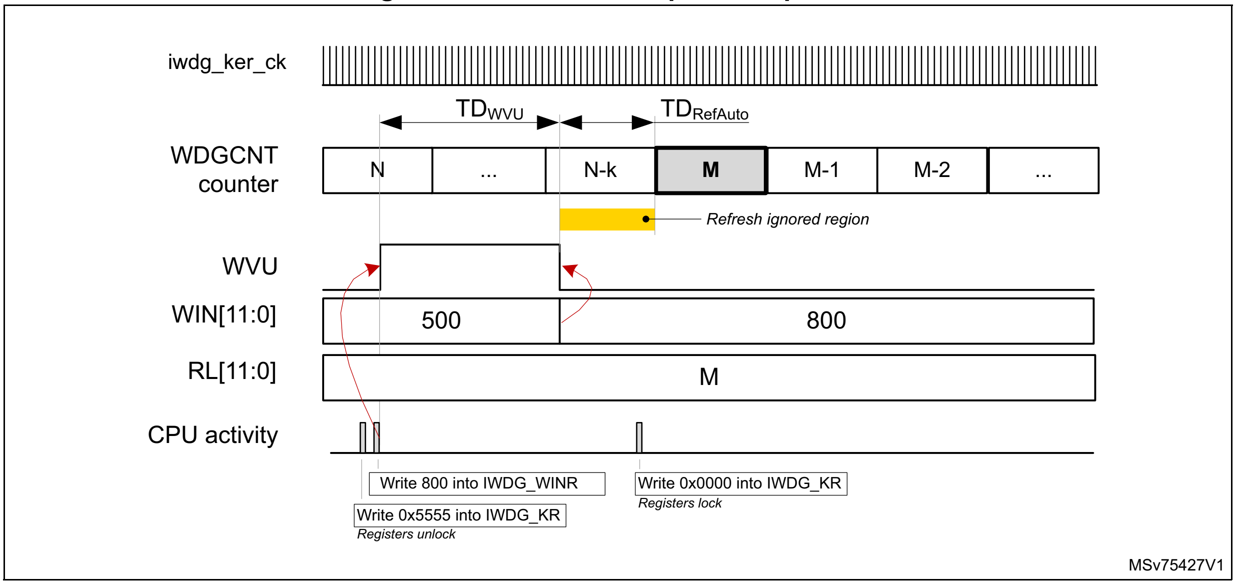
It is possible to update the window comparator when the IWDG is already running. The IWDCNT is reloaded as well. The following sequence can be performed to update the window comparator:

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

Step 3 can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

Figure 377 shows this sequence. As soon as the IWDG_WINR register is written, the WVU flag goes high for TDwvu. After a window comparator update, a refresh is automatically performed. The refresh is effective in the worst case, on the next prescaler clock active edge after the falling edge of WVU (TDRefAuto)

Figure 377. Window comparator update(1)



1. Refer to Table 323: IWDG delays versus actions for details on timing values.

34.4.5 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 34.3 for details on the capabilities of this product.

34.4.6 Register access protection

Write accesses to *IWDG prescaler register (IWDG_PR)*, *IWDG reload register (IWDG_RLR)*, *IWDG early wake-up 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.

34.5 IWDG low power modes

Depending on option bytes configuration, the IWDG can continue counting or not during the low power modes. Refer to *Section 34.3* for details.

Table 325. Effect of low power modes on IWDG

Mode	Description
Sleep	No effect. IWDG interrupts cause the device to exit from the mode.
Stop	The IWDG remains active or not, depending on option bytes configuration. Refer to *Section 34.3* 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 34.3* for details. IWDG interrupts cause the device to exit from Standby mode.

34.6 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 wake-up interrupt register (IWDG_EWCR)* to 1.

A comparator value (EWIT[11:0]) allows the application to define the position where 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 modes, if needed.

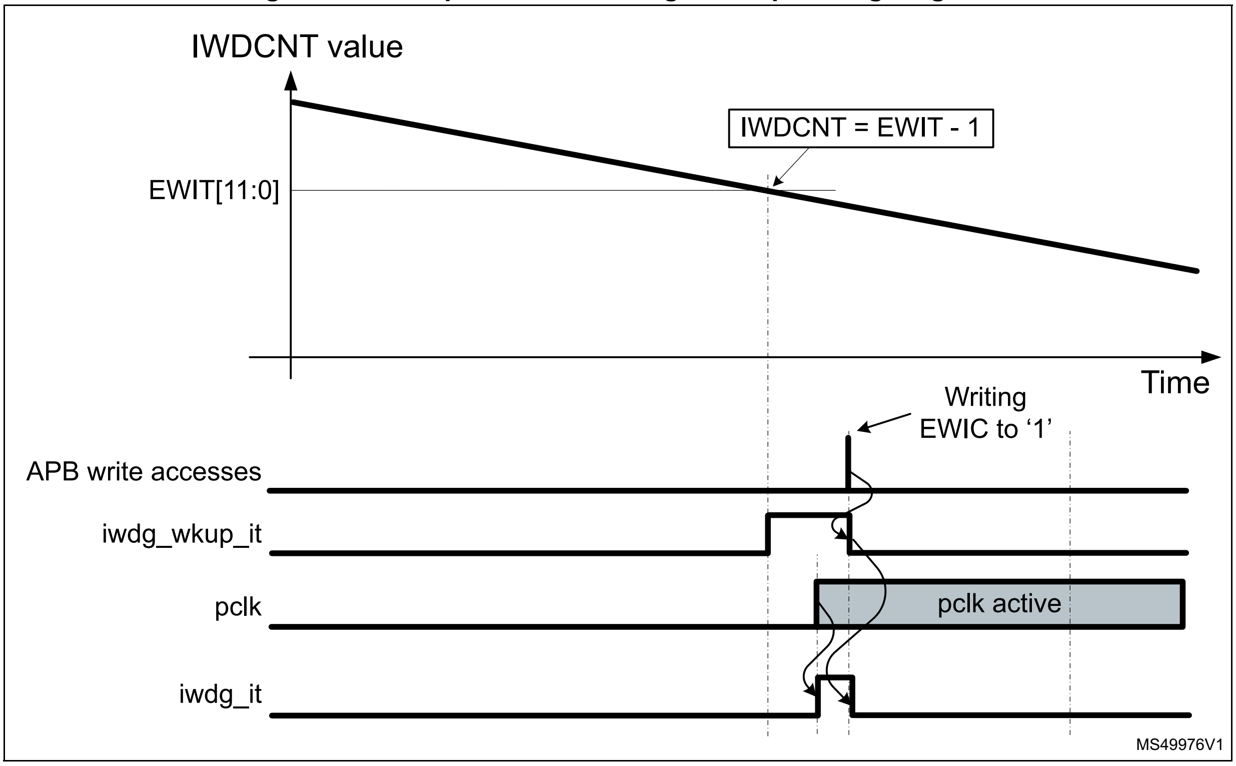
When the APB clock is available, the iwdg_it is activated as well.

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 wake-up 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 378. Independent watchdog interrupt timing diagram



The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the watchdog reset is generated.

Changing the early wake-up comparator value

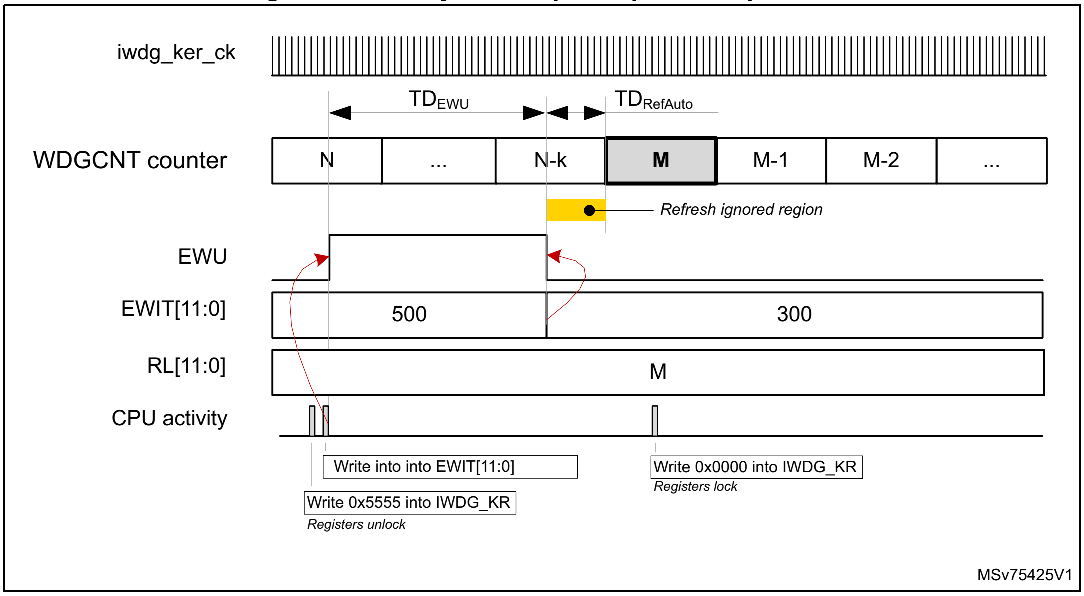
It is possible to change the early wake-up comparator value or to enable/disable the interrupt generation at any time, by performing the following sequence:

1. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
2. Enable or disable the early wake-up interrupt, and/or program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up 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.

Figure 378 shows this sequence. During the early wake-up comparator update operation the flag EWU remains to 1 for TDewu. A refresh is automatically performed. The refresh is effective on the worst case, on the next prescaler clock active edge after the falling edge of EWU (TDRefAuto).

Figure 379. Early wake-up comparator update(1)



1. Refer to *Table 323: IWDG delays versus actions* for details on timing values.

Table 326 summarizes the IWDG interrupt request.

Table 326. 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.

34.7 IWDG registers

Refer to *Section 1.2 on page 70* 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.

34.7.1 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.

34.7.2 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 34.4.6*. 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.

34.7.3 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 34.4.6*. 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.

34.7.4 IWDG status register (IWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000 (0xFFFF FEFF)

This register contains various status flags. Note that the mask value between parenthesis means that the reset value of ONF bit is not defined. When the IWDG is configured in software mode, the reset value of ONF bit is 0, when the IWDG is configured in hardware mode, the reset value of ONF bit is 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.	EWIF	Res.	Res.	Res.	Res.	Res.	ONF	Res.	Res.	Res.	Res.	EWU	WVU	RVU	PVU
	r						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 13:9 Reserved, must be kept at reset value.

Bit 8 **ONF**: Watchdog enable status bit

Set to ‘1’ by hardware as soon as the IWDG is started. In software mode, it remains to '1' until the IWDG is reset. In hardware mode, this bit is always set to '1'.

0: The IWDG is not activated

1: The IWDG is activated and needs to be refreshed regularly by the application

Bits 7:4 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. Refer to *Table 323: IWDG delays versus actions* for delay values.

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. Refer to *Table 323: IWDG delays versus actions* for delay values.

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. Refer to *Table 323: IWDG delays versus actions* for delay values.

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. Refer to *Table 323: IWDG delays versus actions* for delay values.

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.*

34.7.5 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 34.4.6*.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.

34.7.6 IWDG early wake-up 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 wake-up 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 34.4.6*). They are written by software to define at which position of the IWDCNT down-counter the early wake-up 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 wake-up 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.

34.7.7 IWDG register map

Table 327. 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
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.	ONF	Res.	Res.	Res.	Res.	EWU	WVU	RVU	PVU
	Reset value																		0						x					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* for the register boundary addresses.

35 System window watchdog (WWDG)

35.1 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 is cleared. A reset is also generated if the 7-bit down-counter value (in the control register) is refreshed before the down-counter reaches 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 requiring the watchdog to react within an accurate timing window.

35.2 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 381*)
- Early wake-up interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40

35.3 WWDG implementation

Table 328. WWDG features[1]

WWDG mode / feature	WWDG
Window function	X
Early wake-up 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 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.

35.4 WWDG functional description

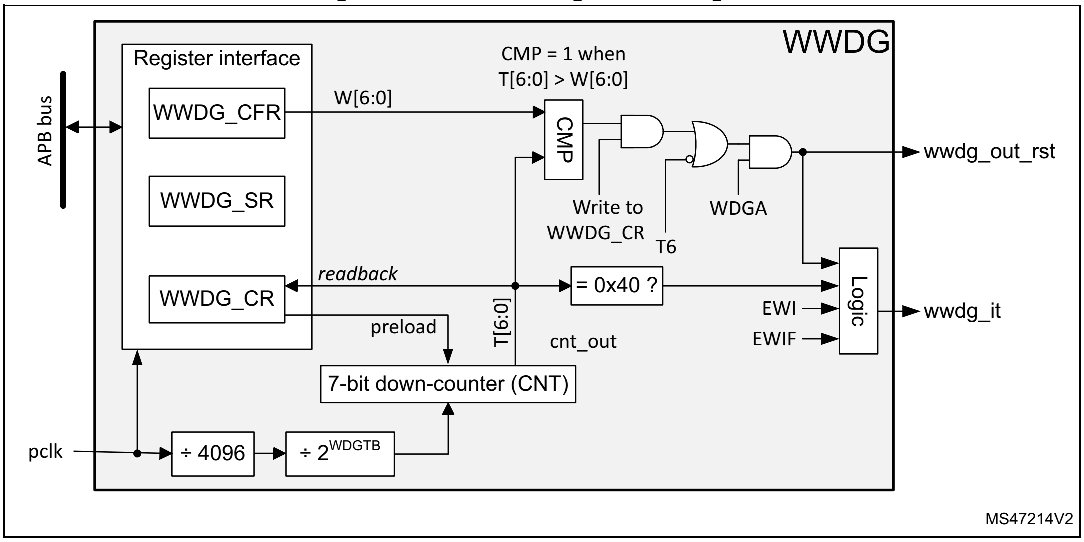
If the watchdog is activated (the WDGA bit is set in the WWDG_CR register), and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent a reset. This operation can take place only when the counter value is lower than or equal to 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 380* for the WWDG block diagram.

35.4.1 WWDG block diagram

Figure 380. Watchdog block diagram



35.4.2 WWDG internal signals

Table 329 gives the list of WWDG internal signals.

Table 329. 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

35.4.3 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.

35.4.4 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 381*). 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 or equal to the window register value, and greater than 0x3F. *Figure 381* describes the window watchdog process.

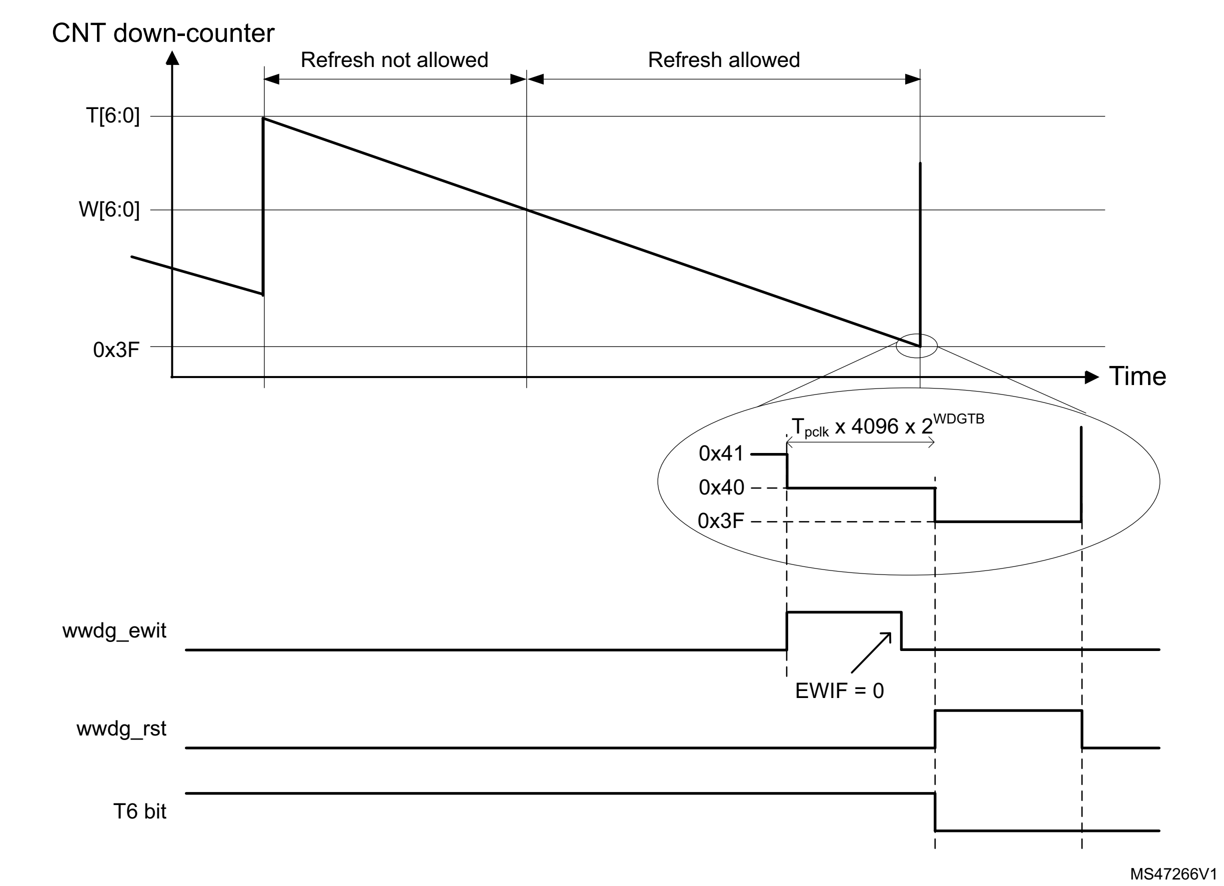
Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

35.4.5 How to program the watchdog timeout

Use the formula in *Figure 381* 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 381. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$t_{WWDG} = t_{PCLK} \times 4096 \times 2^{WDGTB[2:0]} \times (T[5:0] + 1) \qquad (ms)$$

where:

- t_{WWDG}: WWDG timeout
- t_{PCLK}: APB clock period measured in ms
- 4096: value corresponding to internal divider

As an example, if APB frequency is 48 MHz, WDGTB[2:0] is set to 3, and T[5:0] is set to 63:

$$t_{WWDG} = (1 / 48000) \times 4096 \times 2^{3} \times (63 + 1) = 43.69ms$$

Refer to the datasheet for the minimum and maximum values of t_{WWDG}.

35.4.6 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 43: Debug support (DBG)*.

35.5 WWDG interrupts

The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the reset is generated. To enable the early wake-up interrupt, the application must:

- Write EWIF bit of WWDG_SR register to 0, to clear unwanted pending interrupt
- Write EWI bit of WWDG_CFR register to 1, to enable interrupt

When the down-counter reaches the value 0x40, a watchdog 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 must reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The watchdog interrupt is cleared by writing 0 to the EWIF bit in the WWDG_SR register.

Note: When the watchdog interrupt cannot be served (for example due to a system lock in a higher priority task), the WWDG reset is eventually generated.

Table 330. 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 wake-up interrupt	EWIF	EWI	Write EWIF flag to 0	Yes	No	No

1. The WWDG interrupt can have additional capabilities, refer to *Section 35.3* for details.
2. WWDG vector corresponds to the assertion of the wwdg_it signal.

35.6 WWDG registers

Refer to *Section 1.2: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by halfwords (16-bit) or words (32-bit).

35.6.1 WWDG control register (WWDG_CR)

Address offset: 0x000

Reset value: 0x0000 007F

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>Res.</td><td>Res.</td><td>Res.</td><td>Res.</td><td>Res.</td><td>Res.</td><td>Res.</td><td>Res.</td><td>Res.</td><td>Res.</td><td>Res.</td><td>Res.</td><td>Res.</td><td>Res.</td><td>Res.</td><td>Res.</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Res.</td><td>Res.</td><td>Res.</td><td>Res.</td><td>Res.</td><td>Res.</td><td>Res.</td><td>Res.</td><td>WDGA</td><td colspan="7">T[6:0]</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>rs</td><td>rw</td><td>rw</td><td>rw</td><td>rw</td><td>rw</td><td>rw</td><td>rw</td></tr>
</table>

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 ($4096 \times 2^{WDGTB[2:0]}$) PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

35.6.2 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 wake-up interrupt enable

Set by software and cleared by hardware after a reset. When set, an interrupt occurs whenever the counter reaches the value 0x40.

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.

35.6.3 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 wake-up 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.

35.6.4 WWDG register map

The following table gives the WWDG register map and reset values.

Table 331. WWDG 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	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* for the register boundary addresses.

36 Real-time clock (RTC)

36.1 Introduction

The RTC provides an automatic wake-up 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).

36.2 RTC main features

The RTC supports the following features (see *Figure 382: 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.
- 17-bit auto-reload wake-up timer (WUT) for periodic events with programmable resolution and period.
- TrustZone support:
  - RTC fully securable
  - Alarm A, alarm B, wake-up Timer and timestamp individual secure or nonsecure configuration
- Alarm A, alarm B, wake-up Timer and timestamp individual privilege protection

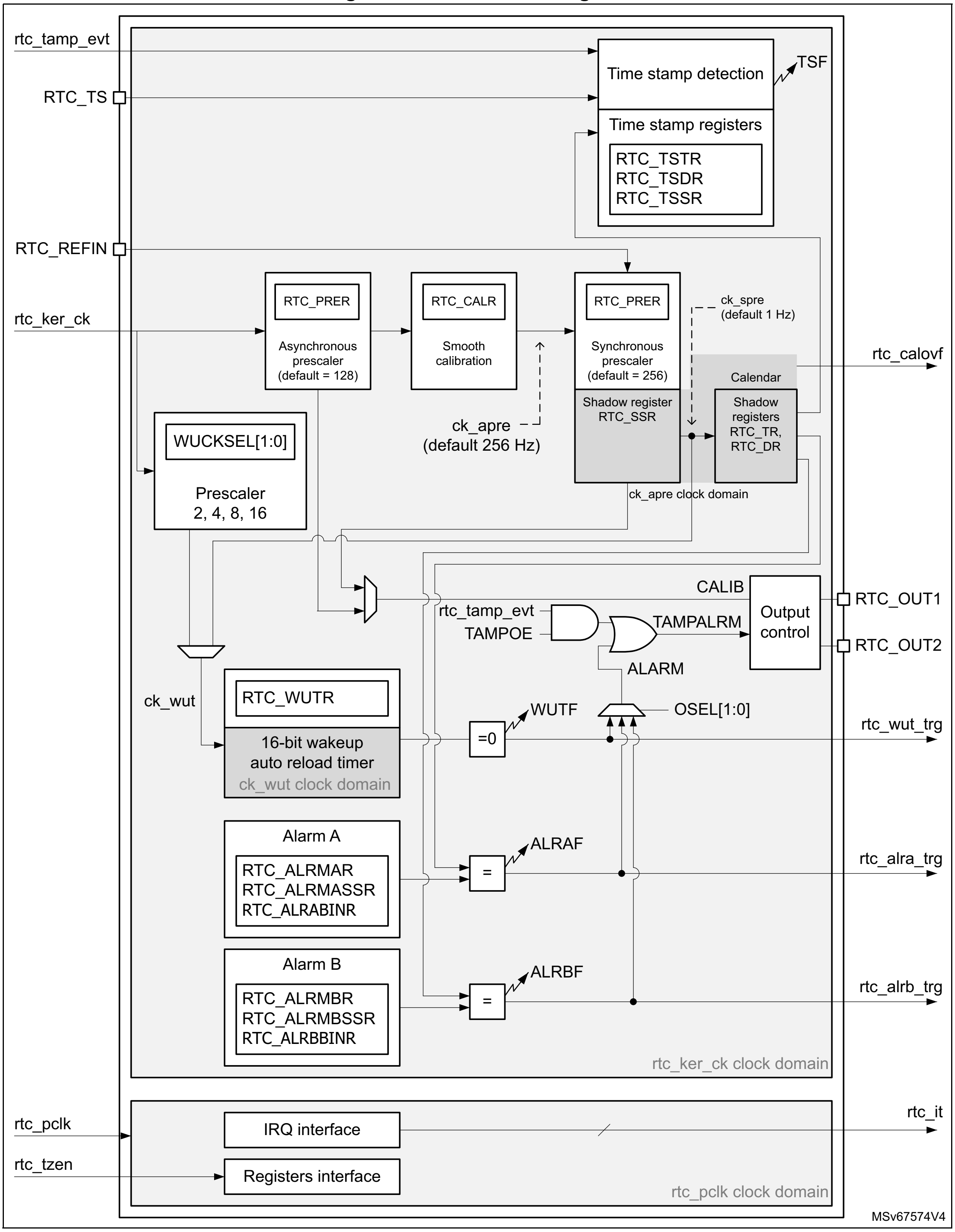
The RTC is functional in all low-power modes when it is clocked by the LSE.

All RTC events (Alarm, wake-up Timer, Timestamp) can generate an interrupt and wake-up the device from the low-power modes.

36.3 RTC functional description

36.3.1 RTC block diagram

Figure 382. RTC block diagram



36.3.2 RTC pins and internal signals

Table 332. 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 rtc_tamp_evt and ALARM signals.

ALARM is enabled by configuring the OSEL[1:0] bits in the RTC_CR register which select the alarm A, alarm B or wake-up outputs. rtc_tamp_evt is enabled by setting the TAMPOE bit in the RTC_CR register which selects the tamper event outputs.

Table 333. 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_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 36.5: 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 wake-up timer event detection trigger
rtc_calovf	Output	RTC calendar overflow: this signal is generated when the RTC calendar reaches its maximum value, on the 31^{st} 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 when the selected clock is not LSE. Refer to *Section 36.4: RTC low-power modes* for more details.

Table 334. RTC interconnection

Signal name	Source/destination
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 nonsecure.

The triggers outputs can be used as triggers for other peripherals.

36.3.3 GPIOs controlled by the RTC and TAMP

The GPIOs included in the backup domain (V_{DD}) are directly controlled by the peripherals providing functions on these I/Os, whatever the GPIO configuration.

Both RTC and TAMP peripherals provide functions on these I/Os (refer to *Section 37: Tamper and backup registers (TAMP)*).

RTC_OUT1, RTC_TS, TAMP_IN4 and TAMP_OUT5 are mapped on the same pin (PC13). The RTC and TAMP functions mapped on PC13 are available in all low-power modes.

The output mechanism follows the priority order shown in *Table 335: RTC pin PC13 configuration*.

Table 335. RTC pin PC13 configuration(1)

<table>
<tr><th>PC13 Pin function</th><th>OSEL[1:0] (ALARM output enable)</th><th>TAMPOE (TAMPER output enable)</th><th>COE (CALIB output enable)</th><th>OUT2EN</th><th>TAMPALRM_TYPE</th><th>TAMPALRM_PU</th><th>TAMP5E=TAMP5AM=1 with ATOSHARE=0, or TAMPxE=TAMPxAM=1 with ATOSHARE=1 and ATOSELx=1</th><th>TAMP4E (TAMP_IN4 input enable)</th><th>TSE (RTC_TS input enable)</th></tr>
<tr><td rowspan="3">TAMPALRM output Push-Pull</td><td>01 or 10 or 11</td><td>0</td><td rowspan="3">Don't care</td><td rowspan="3">Don't care</td><td rowspan="3">0</td><td rowspan="3">0</td><td rowspan="3">Don't care</td><td rowspan="3">Don't care</td><td rowspan="3">Don't care</td></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01 or 10 or 11</td><td>1</td></tr>
</table>

Table 335. RTC pin PC13 configuration[1] (continued)

<table>
<tr><th colspan="2">PC13 Pin function</th><th>OSEL[1:0] (ALARM output enable)</th><th>TAMPOE (TAMPER output enable)</th><th>COE (CALIB output enable)</th><th>OUT2EN</th><th>TAMPALRM_TYPE</th><th>TAMPALRM_PU</th><th>TAMP5E=TAMP5AM=1 with ATOSHARE=0, or TAMPxE=TAMPxAM=1 with ATOSHARE=1 and ATOSELx=1</th><th>TAMP4E (TAMP_IN4 input enable)</th><th>TSE (RTC_TS input enable)</th></tr>
<tr><td rowspan="6">TAMPALRM output Open-Drain[2]</td><td rowspan="3">No pull</td><td>01 or 10 or 11</td><td>0</td><td rowspan="3">Don't care</td><td rowspan="3">Don't care</td><td rowspan="3">1</td><td rowspan="3">0</td><td rowspan="3">Don't care</td><td rowspan="3">Don't care</td><td rowspan="3">Don't care</td></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01 or 10 or 11</td><td>1</td></tr>
<tr><td rowspan="3">Internal pull-up</td><td>01 or 10 or 11</td><td>0</td><td rowspan="3">Don't care</td><td rowspan="3">Don't care</td><td rowspan="3">1</td><td rowspan="3">1</td><td rowspan="3">Don't care</td><td rowspan="3">Don't care</td><td rowspan="3">Don't care</td></tr>
<tr><td>00</td><td>1</td></tr>
<tr><td>01 or 10 or 11</td><td>1</td></tr>
<tr><td colspan="2">CALIB output PP</td><td>00</td><td>0</td><td>1</td><td>0</td><td>Don't care</td><td>Don't care</td><td>Don't care</td><td>Don't care</td><td>Don't care</td></tr>
<tr><td colspan="2">TAMP_OUT5 output PP</td><td>00</td><td>0</td><td>0</td><td>0</td><td>Don't care</td><td>Don't care</td><td>1</td><td>Don't care</td><td>Don't care</td></tr>
<tr><td colspan="2" rowspan="3">TAMP_IN4 input floating</td><td>00</td><td>0</td><td>0</td><td>Don't care</td><td rowspan="3">Don't care</td><td rowspan="3">Don't care</td><td rowspan="3">0</td><td rowspan="3">1</td><td rowspan="3">0</td></tr>
<tr><td>00</td><td>0</td><td>1</td><td rowspan="2">1</td></tr>
<tr><td>Don't care</td><td>Don't care</td><td>0</td></tr>
<tr><td colspan="2" rowspan="3">RTC_TS and TAMP_IN4 input floating</td><td>00</td><td>0</td><td>0</td><td>Don't care</td><td rowspan="3">Don't care</td><td rowspan="3">Don't care</td><td>0</td><td rowspan="3">1</td><td rowspan="3">1</td></tr>
<tr><td>00</td><td>0</td><td>1</td><td rowspan="2">1</td><td>0</td></tr>
<tr><td>Don't care</td><td>Don't care</td><td>0</td><td>0</td></tr>
<tr><td colspan="2" rowspan="3">RTC_TS input floating</td><td>00</td><td>0</td><td>0</td><td>Don't care</td><td rowspan="3">Don't care</td><td rowspan="3">Don't care</td><td>0</td><td rowspan="3">0</td><td rowspan="3">1</td></tr>
<tr><td>00</td><td>0</td><td>1</td><td rowspan="2">1</td><td>0</td></tr>
<tr><td>Don't care</td><td>Don't care</td><td>0</td><td>0</td></tr>
</table>

Table 335. 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	TAMP5E=TAMP5AM=1 with ATOSHARE=0, or TAMPxE=TAMPxAM=1 with ATOSHARE=1 and ATOSELx=1	TAMP4E (TAMP_IN4 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 336: RTC_OUT mapping*.

Table 336. RTC_OUT mapping

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

36.3.4 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 nonsecure 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 nonsecure 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 clear, all the RTC registers can be read and written in secure and nonsecure 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 nonsecure 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 nonsecure 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, TSE, TSIE, TSEDGE control bits in the RTC_CR, CTSOVF and CTSF bits in the RTC_SCR, TSF, TSOVF in RTC_SR, and TSMF, TSOVMF 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 nonsecure 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.

36.3.5 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, TSE, TSIE, TSEDGE control bits in the RTC_CR, CTSOVF and CTSF bits in the RTC_SCR, TSF, TSOVF in RTC_SR, and TSMF, TSOVMF 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.

36.3.6 Clock and prescalers

The RTC clocks must respect this ratio: frequency(PCLK) ? 2 × frequency(RTCCLK).

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 382: 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 wake-up auto-reload timer. To obtain short timeout periods, the 16-bit wake-up auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see *Section 36.3.10: Periodic auto-wake-up* 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.

36.3.7 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 36.6.12: 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.

36.3.8 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.

36.3.9 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_ALRABINR.

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.

36.3.10 Periodic auto-wake-up

The periodic wake-up flag is generated by a 16-bit programmable auto-reload down-counter. The wake-up timer range can be extended to 17 bits.

The wake-up function is enabled through the WUTE bit in the RTC_CR register.

The wake-up 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 wake-up 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 wake-up 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 wake-up timer on page 1296*), the timer starts counting down. When the wake-up 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 wake-up 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 wake-up flag is output on an internal signal rtc_wut that can be used by other peripherals (refer to section *Section 36.3.1: RTC block diagram*).

When the periodic wake-up interrupt is enabled by setting the WUTIE bit in the RTC_CR register, it can exit the device from low-power modes.

The periodic wake-up 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 wake-up timer.

36.3.11 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 power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable RTC registers write access.

After Backup domain reset, some of the RTC registers are write-protected: RTC_TR, RTC_DR, RTC_PRER, RTC_CALR, RTC_SHIFTR, the bits INIT, BIN and BCDU in RTC_ICSR and the bits FMT, SUB1H, ADD1H, REFCKON in RTC_CR.

The following steps are required to unlock the write protection on the protected RTC registers.

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

The registers protected by INITPRIV are write-protected by the INIT KEY.

The registers protected by CALPRIV 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 ?s 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_ALRABINR).
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 wake-up timer

The following sequence is required to configure or change the wake-up timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wake-up timer.
2. Poll WUTWF until it is set in RTC_ICSR to make sure the access to wake-up 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 wake-up auto-reload value WUT[15:0], WUTOCLR[15:0] and the wake-up clock selection (WUCKSEL[2:0] bits in RTC_CR). Set WUTE in RTC_CR to enable the timer again. The wake-up 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.

36.3.12 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 wake-up 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 1295*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to *Section 36.3.14: 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.

36.3.13 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 wake-up 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.

36.3.14 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.

36.3.15 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.

36.3.16 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.

36.3.17 Timestamp function

Timestamp is enabled by setting the TSE bit 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. Refer to *RTC control register (RTC_CR)* and refer to *Section : Timestamp on tamper event*.

When a timestamp event occurs, the timestamp flag bit (TSF) in RTC_SR register is set.

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 up to 2 ck_apre cycles after the timestamp event from RTC_TS pin occurs, due to synchronization process. TSF is set up to 3 ck_apre cycles after tamper flags.

TSOVF is set up to only 1 ck_apre cycle after the event occurs. 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.

36.3.18 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the CALIB device output.

If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the CALIB frequency is $f_{RTCCLK/64}$. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz. The CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and “PREDIV_S+1” is a non-zero multiple of 256 (i.e: PREDIV_S[7:0] = 0xFF), the CALIB frequency is fRTCCLK/(256 * (PREDIV_A+1)). This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = Ox7F, PREDIV_S = 0xFF), with an RTCCLK frequency at 32.768 kHz.

Note: When 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.

36.3.19 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 in 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 wake-up 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.*

36.4 RTC low-power modes

Table 337. 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.

The table below summarizes the RTC pins and functions capability in all modes.

Table 338. RTC pins functionality over modes

Functions	Functional in all low-power Stop modes	Functional in Standby mode
RTC_TS	Yes	Yes
RTC_REFIN	Yes	No
RTC_OUT1	Yes	Yes
RTC_OUT2	Yes	Yes

36.5 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 nonsecure interrupt output or the secure interrupt output is also activated.

Table 339. Nonsecure 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)
	Wake-up timer	WUTF	WUTIE and (WUTSEC=0 and SEC=0)	write 1 in CWUTF	Yes(3)
	SSR underflow (reload)	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 340. 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]
	Wake-up timer	WUTF	WUTIE and (WUTSEC=1 or SEC=1)	write 1 in CWUTF	Yes[3]
	SSR underflow (reload)	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_SMISR register.
3. When the RTC is clocked by an oscillator functional in the low-power mode.

36.6 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).

36.6.1 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 1295* and *Reading the calendar on page 1297*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1295*.

This register can be write-protected against nonsecure access. Refer to *Section 36.3.4: RTC secure protection modes*.

This register can be write-protected against non-privileged access. Refer to *Section 36.3.5: 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

36.6.2 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 1295* and *Reading the calendar on page 1297*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1295*.

This register can be write-protected against nonsecure access. Refer to *Section 36.3.4: RTC secure protection modes*.

This register can be write-protected against non-privileged access. Refer to *Section 36.3.5: 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.

36.6.3 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.

36.6.4 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 1295*.

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 36.3.4: 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 36.3.5: 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**: Wake-up 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: Wake-up timer configuration update not allowed except in initialization mode

1: Wake-up timer configuration update allowed

Bits 1:0 Reserved, must be kept at reset value.

36.6.5 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 1295*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1295*.

This register can be write-protected against nonsecure access. Refer to *Section 36.3.4: RTC secure protection modes*.

This register can be write-protected against non-privileged access. Refer to *Section 36.3.5: 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)

36.6.6 RTC wake-up 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 nonsecure access. Refer to *Section 36.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 36.3.5: 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]**: Wake-up 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]**: Wake-up auto-reload value bits

When the wake-up 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 wake-up 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.

36.6.7 RTC control register (RTC_CR)

This register is write protected. The write access procedure is described in RTC register write protection on page 1295.

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 36.3.4: 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 36.3.5: 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	Res.	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
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.
Note: TAMPTS must be cleared before entering RTC initialization mode.

Bit 24 Reserved, must be kept at reset value.

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: Wake-up 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 36.3.18: Calibration clock output*.

Bit 18 **BKP**: Backup
This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: *S*ubtract 1 hour (winter time change)
When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.
Setting this bit has no effect when current hour is 0.
0: No effect
1: Subtracts 1 hour to the current time. This can be used for winter time change.

Bit 16 **ADD1H**: Add 1 hour (summer time change)
When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.
0: No effect
1: Adds 1 hour to the current time. This can be used for summer time change

Bit 15 **TSIE**: Timestamp interrupt enable
0: Timestamp interrupt disable
1: Timestamp interrupt enable

Bit 14 **WUTIE**: Wake-up timer interrupt enable
0: Wake-up timer interrupt disabled
1: Wake-up 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**: Wake-up timer enable
0: Wake-up timer disabled
1: Wake-up timer enabled
Note: When the wake-up 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 wake-up 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 = wake-up 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 may mask the incrementation of the calendar hour.

ADD1H and SUB1H changes are effective in the next second.

36.6.8 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 nonsecure access depending on the RTC_SECCFGR configuration (refer to *Section 36.3.5: 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**: Wake-up timer privilege protection

0: RTC wake-up timer configuration and interrupt clear can be written when the APB access is privileged or non-privileged.

1: RTC wake-up 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 36.3.5: RTC privilege protection modes for details on the read protection.

36.6.9 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 36.3.5: 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 nonsecure, 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 nonsecure.

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 nonsecure.

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 nonsecure.

1: RTC timestamp configuration and interrupt clear can be written only when the APB access is secure.

Bit 2 **WUTSEC**: Wake-up timer protection

0: RTC wake-up timer configuration and interrupt clear can be written when the APB access is secure or nonsecure.

1: RTC wake-up 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 nonsecure.

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 nonsecure.

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 36.3.4: RTC secure protection modes for details on the read protection.

36.6.10 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.

36.6.11 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 1295*.

This register can be write-protected against nonsecure access. Refer to *Section 36.3.4: RTC secure protection modes*.

This register can be write-protected against non-privileged access. Refer to *Section 36.3.5: 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 36.3.16: 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 36.3.16: 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 36.3.16: 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 36.3.16: RTC smooth digital calibration on page 1300*.

36.6.12 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 1295*.

This register can be protected against nonsecure access. Refer to *Section 36.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 36.3.5: 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.

36.6.13 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 nonsecure access. Refer to *Section 36.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 36.3.5: 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.

36.6.14 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 nonsecure access. Refer to *Section 36.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 36.3.5: 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

36.6.15 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 nonsecure access. Refer to *Section 36.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 36.3.5: 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.

36.6.16 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 nonsecure access. Refer to *Section 36.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 36.3.5: 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.

36.6.17 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 nonsecure access. Refer to *Section 36.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 36.3.5: 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_ALRABINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one ck_apre cycle after reaching RTC_ALRABINR.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_ALRABINR, and so can also be read or written through RTC_ALRABINR.

Note: SS[3:0] must be 0000 when SSCLR is set with ATCKSEL[3] = 1 in TAMP_ATCR1.

36.6.18 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 nonsecure access. Refer to *Section 36.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 36.3.5: 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	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 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

36.6.19 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 nonsecure access. Refer to *Section 36.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 36.3.5: 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_ALRBBINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one ck_apre cycle after reaching RTC_ALRBBINR.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_ALRBBINR, and so can also be read or written through RTC_ALRBBINR.

Note: SS[3:0] must be 0000 when SSCLR is set with ATCKSEL[3] = 1 in TAMP_ATCR1.

36.6.20 RTC status register (RTC_SR)

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 36.3.4: 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 36.3.5: 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	Res.	TSOVF	TSF	WUTF	ALRBF	ALRAF
									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 is reloaded with 0xFFFF FFFF after reaching 0. SSRUF is not set when SSCLR = 1.

Note: SSRUF is not an error event as SSR counter is a free-running down-counter with automatic reload.

Bit 5 Reserved, must be kept at reset value.

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.

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**: Wake-up timer flag

This flag is set by hardware when the wake-up auto-reload counter reaches 0.
If WUTOCLR[15:0] is different from 0x0000, WUTF is cleared by hardware when the wake-up 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 few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

36.6.21 RTC nonsecure 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 36.3.5: 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.	SSRUMF	Res.	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
									r		r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUMF**: SSR underflow nonsecure masked flag

This flag is set by hardware when the SSR underflow nonsecure interrupt occurs.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **TSOVMF**: Timestamp overflow nonsecure 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 nonsecure masked flag

This flag is set by hardware when a timestamp nonsecure interrupt occurs.

Bit 2 **WUTMF**: Wake-up timer nonsecure masked flag

This flag is set by hardware when the wake-up timer nonsecure 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 nonsecure masked flag

This flag is set by hardware when the alarm B nonsecure interrupt occurs.

Bit 0 **ALRAMF**: Alarm A masked flag

This flag is set by hardware when the alarm A nonsecure interrupt occurs.

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

36.6.22 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 nonsecure access. Refer to *Section 36.3.4: 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 36.3.5: 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	Res.	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
									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 Reserved, must be kept at reset value.

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.

Bit 2 **WUTMF**: Wake-up timer interrupt secure masked flag

This flag is set by hardware when the wake-up 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.

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

36.6.23 RTC status clear register (RTC_SCR)

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 36.3.4: 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 36.3.5: 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	Res.	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
									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 Reserved, must be kept at reset value.

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.

Bit 2 **CWUTF**: Clear wake-up 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.

36.6.24 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 nonsecure access. Refer to *Section 36.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 36.3.5: 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.

36.6.25 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 nonsecure access. Refer to *Section 36.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 36.3.5: 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.

36.6.26 RTC register map

Table 341. RTC 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	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	Res.	COE	OSEL [1:0]		POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	SSRUIE	FMT	BYPSHAD	REFCKON	TSEDGE	WUCKSEL[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
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 341. RTC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	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	Res.	TSOVF	TSF	WUTF	ALRBF	ALRAF
	Reset value																										0		0	0	0	0	0
0x54	RTC_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	Res.	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
	Reset value																										0		0	0	0	0	0
0x58	RTC_SMISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	Res.	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
	Reset value																										0		0	0	0	0	0
0x5C	RTC_SCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSRUF	Res.	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
	Reset value																										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: Memory organization* for the register boundary addresses.

37 Tamper and backup registers (TAMP)

37.1 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. The backup registers, as well as other secrets in the device, are protected by this anti-tamper detection circuit with 6 tamper pins and 9 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.

37.2 TAMP main features

- A tamper detection can optionally erase the backup registers, SRAM2, cache, and cryptographic peripherals. The device resources protected by tamper are named “device secrets”.
- 32 32-bit backup registers
- Up to 6 tamper pins for 6 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 up to 3 meshes (each input associated to its own exclusive output) to up to 5 meshes (single output shared for up to 5 tamper inputs)
  - Passive tampers: ultra-low power edge or level detection with internal pull-up hardware management.
  - Configurable digital filter.
- 9 internal tamper events to protect against transient or environmental perturbation attacks
- Each tamper can be configured in two modes:
  - Confirmed mode: immediate erase of secrets on tamper detection, including backup registers erase
  - Potential mode: most of the secrets erase following a tamper detection are launched by software
- Any tamper detection can generate a RTC timestamp event.
- TrustZone support:
  - Tamper secure or nonsecure configuration.
  - Backup registers configuration in 3 configurable-size areas:

    1 read/write secure area.

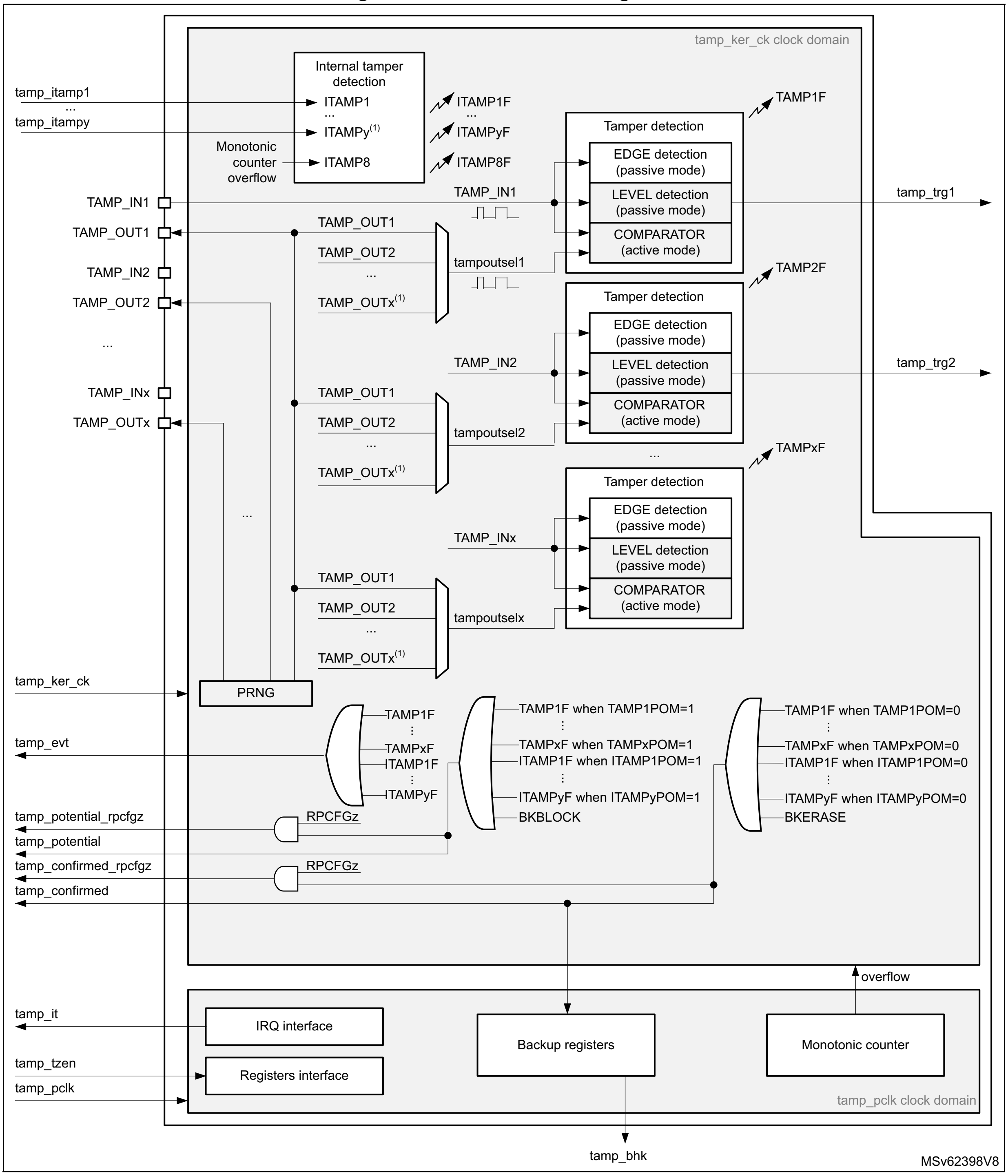
    1 write secure/read nonsecure area.

    1 read/write nonsecure 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.

37.3 TAMP functional description

37.3.1 TAMP block diagram

Figure 383. TAMP block diagram



1. The number of external and internal tampers depends on products.

37.3.2 TAMP pins and internal signals

Table 342. 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 343. 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 confirmed or potential mode 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 potential mode. – or a software request is done by writing BKBLOCK to 1
tamp_confirmed	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 confirmed mode. – or a software request is done by writing BKERASE to 1
tamp_potential_rpcfgz (z = signal index)	Output	Potential tamper detection signal generated only when RPCFGz = 1. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in potential mode. – or a software request is done by writing BKBLOCK to 1

Table 343. TAMP internal input/output signals (continued)

Internal signal name	Signal type	Description
tamp_confirmed_rpcfgz (z = signal index)	Output	Confirmed tamper detection signal generated only when RPCFGz = 1. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in confirmed mode. – or a software request is done by writing BKERASE to 1
tamp_it	Output	TAMP interrupt (refer to *Section 37.5: 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 344: 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). The TAMP kernel clock is required only for external tamper inputs level with filtering, and for external active tamper detection modes. Internal tampers detection and external tampers inputs edge detection are functional without requiring any kernel clock.

Read and write access to backup registers and all other TAMP registers are also functional without any kernel clock (only APB clock is needed).

Some detections modes are not available in some low-power modes depending on the selected clock (refer to *Section 37.4: TAMP low-power modes* for more details.

Table 344. 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
tamp_confirmed	The tamp_confirmed signal is used to erase the device secrets listed hereafter: – backup registers The device secrets access is blocked when erase is ongoing.
tamp_potential_rpcfg1	When the bit RPCFG1 is set in the TAMP_RPCFGR, the tamp_potential_rpcfg1 signal is used to block the read and write accesses to the device secrets listed hereafter: – SRAM2

Table 344. TAMP interconnection (continued)

Signal name	Source/Destination
tamp_confirmed_rpcfg1	When the bit RPCFG1 is set in the TAMP_RPCFGR, the tamp_erase_rpcfg1 signal is used to erase the device secrets listed hereafter: – SRAM2 The device secrets access is blocked when erase is on-going
tamp_potential_rpcfg2	When the bit RPCFG2 is set in the TAMP_RPCFGR, the tamp_potential_rpcfg2 signal is used to block the device secrets listed hereafter: – RHUK
tamp_confirmed_rpcfg2	When the bit RPCFG2 is set in the TAMP_RPCFGR, the tamp_erase_rpcfg2 signal is used to block the device secrets listed hereafter: – RHUK
tamp_potential_rpcfgz (z = 3 to 5)	When the bit RPCFGz is set in the TAMP_RPCFGR, the tamp_potential_rpcfgz signal is used to erase the device secrets listed hereafter: – z = 3: ICACHE – z = 4: SAES, AES and HASH – z = 5: PKA SRAM The device secrets access is blocked when erase is on-going.
tamp_confirmed_rpcfgz (z = 3 to 5)	When the bit RPCFGz is set in the TAMP_RPCFGR, the tamp_erase_rpcfgz signal is used to erase the device secrets listed hereafter: – z = 3: ICACHE – z = 4: SAES, AES and HASH – z = 5: PKA SRAM The device secrets access is blocked when erase is on-going.
tamp_itamp3	LSE monitoring (LSECSS)
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[1]	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 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 nonsecure.

37.3.3 GPIOs controlled by the RTC and TAMP

Refer to *Section 36.3.3: GPIOs controlled by the RTC and TAMP*.

37.3.4 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.

37.3.5 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 nonsecure 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 nonsecure 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 nonsecure access to a secure-protected register is denied:

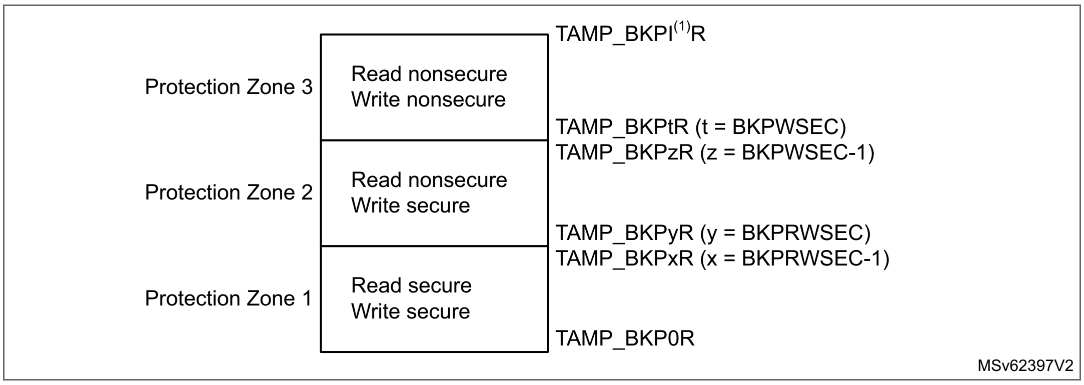
- There is no bus error generated.
- A notification is generated through a flag/interrupt in the TZIC (TrustZone illegal access controller).
- When write protected, the bits are not written.
- When read protected they are read as 0.

As soon as at least one function is configured to be secured, the TAMP reset and clock control is also secured in the RCC.

37.3.6 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 384. 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.

37.3.7 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 counter 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 counter 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 384*). 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.

37.3.8 Boot hardware key (BHK)

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. It is also cleared with BKERASE command (in all 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.

37.3.9 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 = pin index) 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 other device secrets stored in SRAMs or peripherals listed in *Table 344: TAMP interconnection*. The device secrets list is configurable thanks to *TAMP resources protection configuration register (TAMP_RPCFGR)*.
- block the read/write access to the backup registers and other device secrets stored in SRAMs or peripherals listed in *Table 344: TAMP interconnection*. The device secrets list is configurable thanks to TAMP_RPCFGR.
- generate an interrupt, capable to wake-up 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 = pin index) is continuously compared with TAMP_OUTy (y = pin index) 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 in potential mode, 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.

37.3.10 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.

Tamper detection – confirmed mode

The confirmed mode is selected for TAMPx (external tamper x) when TAMPxPOM = 0 in the TAMP_CR2 register. The confirmed mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 0 in the TAMP_CR3 register. The effects of a tamper detection in confirmed mode are described with tamp_confirmed and tamp_confirmed_rpcfgx signals in the *Table 344: TAMP interconnection*.

This mode is selected to erase automatically the device secrets when the tamper is detected.

Tamper detection – potential mode

The potential mode is selected for TAMPx (external tamper x) when TAMPxPOM = 1 in the TAMP_CR2 register. The potential tamper mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 1 in the TAMP_CR3 register. The effects of a tamper detection in potential mode are described with tamp_potential and tamp_potential_rpcfgx signals in the *Table 344: TAMP interconnection*.

This mode is selected to avoid irreversible erasure of some device secrets when the tamper is detected. In this mode, some device secrets are not erased when the corresponding tamper event is detected. In addition, the read and write accesses to these device secrets are blocked as soon as the tamper detection flag is set in potential mode, until this flag is cleared by setting the corresponding clear flag in the TAMP_SCR register. 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 device secrets are erased by software by setting the BKERASE bit in the TAMP_CR2 register.

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 344: TAMP interconnection*). The IWDG tamper must be configured with ITAMPxPOM = 0. This permits the erasure of device secrets to be forced by hardware after a timeout, in case the previous tamper event was in potential mode. 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 resources protection configuration

Some device resources can be configured in order to be included to the list of the device secrets protected by tamper detection.

When RPCFGz = 0 in the TAMP_RPCFGR, the device resource associated to RPCFGz is not protected by the TAMP peripheral:

- It is not affected by tamper detection (whatever confirmed or potential mode)
- It is not affected by BKERASE software command
- It is not affected by BKBLOCK software command

When RPCFGz = 1 in the TAMP_RPCFGR, the device resource associated to RPCFGz is protected by the TAMP peripheral:

- It is affected by confirmed tamper detection and BKERASE software command, as described with tamp_confirmed_rpcfgz signal in *Table 344: TAMP interconnection*
- It is affected by potential tamper detection and BKBLOCK software command, as described with tamp_potential_rpcfgz signal in *Table 344: TAMP interconnection*

Table 345. Device resource x tamper protection

-	Potential tamper or BKBLOCK	Confirmed tamper or BKERASE
RPCFGx = 0	No effect on device resource x	No effect on device resource x
RPCFGx = 1	Device secret x protected as described by tamp_potential_rpcfgx(1)	Device secret x protected as described by tamp_confirmed_rpcfgx(1)

1. Refer to *Table 344: TAMP interconnection*.

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.

37.3.11 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 is 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 wake-up 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 level detection with filtering mode (TAMPFLT ? 00 and active mode is not selected). Refer to *Section : Level detection with filtering on tamper inputs (passive mode)*.

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 case a timestamp occurs due to tamper event, 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.

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. The TAMP_IN I/O schmitt trigger is enabled only during the precharge duration to avoid any extra consumption if the tamper switch is open (floating state).

Figure 385. Tamper sampling with precharge pulse

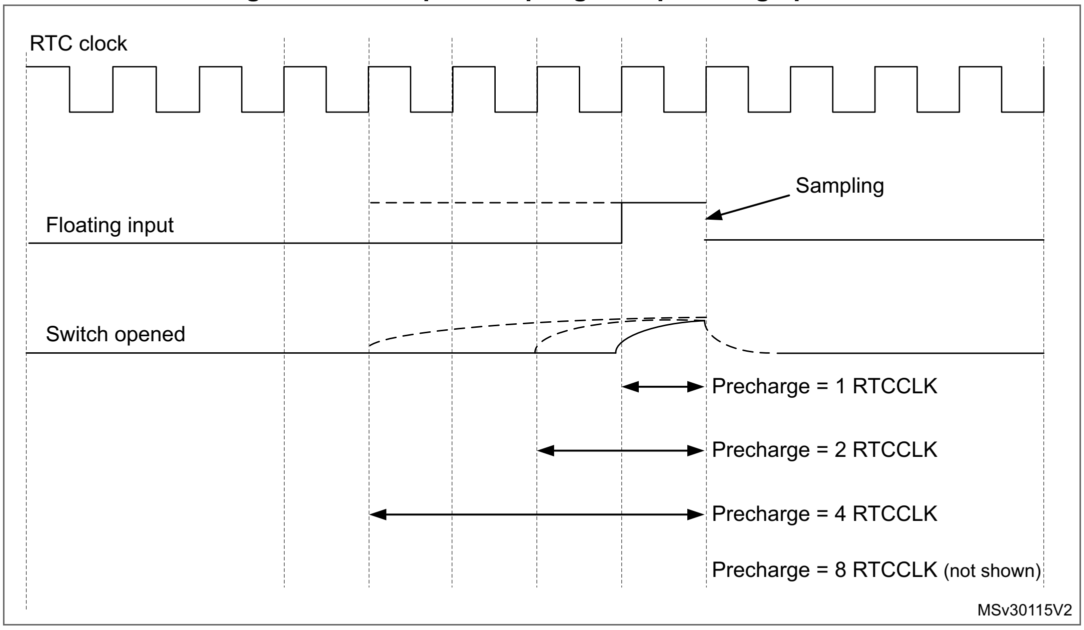
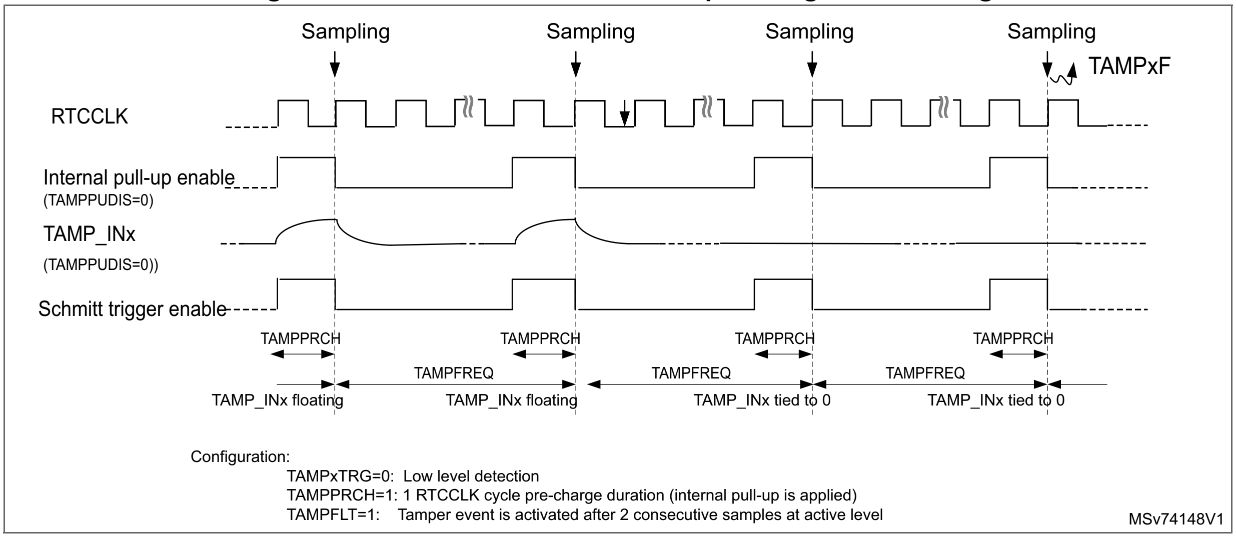


Figure 386. Low level detection with precharge and filtering



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. The TAMP_OUTy function is enabled on the I/O as soon as it is selected for comparison with an active tamper input TAMP_INx (TAMPxEN = TAMPxAM = 1), thanks to

ATOSHARE and ATOSELx bits. Refer to ATOSHARE and ATOSEL bits descriptions in the TAMP_ATCRx (x = 1, 2) registers.

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 346. 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
...	...	...	...	...
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	$f_{RTCCLK}/2048$(2)	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. 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 347. Minimum ATPER value

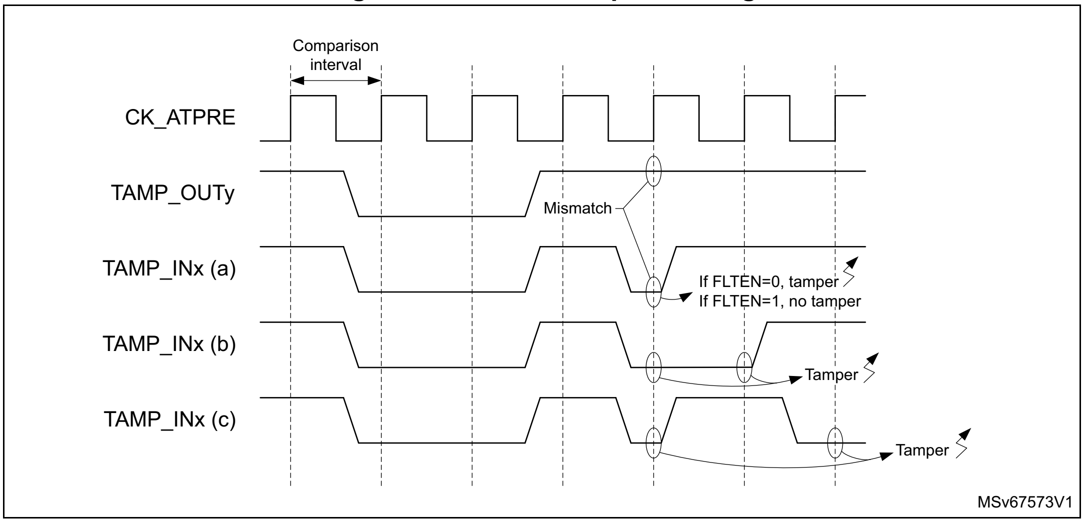
Number of selected outputs	Minimum ATPER
1	0
2	1
3 or 4	2
5 or more	3

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 = 000 and ATOSEL2 = 000.

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 387. Active tamper filtering



As illustrated in *Figure 387*, 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 348* for each possible setting of ATCKSEL, assuming f_{RTCCLK} = 32.768 kHz.

Table 348. 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	$f_{RTCCLK}/2048$	62.500(1)

1. 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: 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 initialized with a 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.

37.4 TAMP low-power modes

Table 349. 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. TAMP interrupts 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. TAMP interrupts cause the device to exit the Standby mode.

Table 350. TAMP pins functionality over modes (STM32WBA52xx only)

Pin name	Functional in all low-power modes
TAMP_IN[5:1]	Yes
TAMP_OUT[6:4, 2:1]	Yes

Table 351. TAMP pins functionality over modes (STM32WBA54/55xx only)

Pin name	Functional in all low-power modes
TAMP_IN[6:1]	Yes
TAMP_OUT[6:1]	Yes

37.5 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 352. Interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit	Interrupt clear method	Exit from low-power modes
TAMP	Tamper x(2)	TAMPxF	TAMPxIE	Write 1 in CTAMPxF	Yes(3)
	Internal tamper y(2)	ITAMPyF	ITAMPyIE	Write 1 in CITAMPyF	Yes(3)

1. The event flags are in the TAMP_SR register.
2. The number of tampers and internal tampers events depend on products.
3. Refer to *Table 349: Effect of low-power modes on TAMP* for more details about available features in the low-power modes.

37.6 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).

37.6.1 TAMP control register 1 (TAMP_CR1)

This register can be protected against nonsecure access. Refer to *Section 37.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 37.3.7:*

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	Res.	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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP6E	TAMP5E	TAMP4E	TAMP3E	TAMP2E	TAMP1E
										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 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 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 **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.

37.6.2 TAMP control register 2 (TAMP_CR2)

This register can be protected against nonsecure access. Refer to *Section 37.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 37.3.7:*

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
Res.	Res.	TAMP6TRG	TAMP5TRG	TAMP4TRG	TAMP3TRG	TAMP2TRG	TAMP1TRG	BKERASE	BKBLOCK	Res.	Res.	Res.	TAMP3MSK	TAMP2MSK	TAMP1MSK
		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.	Res.	Res.	TAMP6POM	TAMP5POM	TAMP4POM	TAMP3POM	TAMP2POM	TAMP1POM
										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 **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 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **TAMP6POM**: Tamper 6 potential mode

0: Tamper 6 event detection is in confirmed mode[(1)].
1: Tamper 6 event detection is in potential mode[(2)].

Bit 4 **TAMP5POM**: Tamper 5 potential mode

0: Tamper 5 event detection is in confirmed mode[(1)].
1: Tamper 5 event detection is in potential mode[(2)].

Bit 3 **TAMP4POM**: Tamper 4 potential mode

0: Tamper 4 event detection is in confirmed mode[(1)].
1: Tamper 4 event detection is in potential mode[(2)].

Bit 2 **TAMP3POM**: Tamper 3 potential mode
0: Tamper 3 event detection is in confirmed mode(1).
1: Tamper 3 event detection is in potential mode(2).

Bit 1 **TAMP2POM**: Tamper 2 potential mode
0: Tamper 2 event detection is in confirmed mode(1).
1: Tamper 2 event detection is in potential mode(2).

Bit 0 **TAMP1POM**: Tamper 1 potential mode
0: Tamper 1 event detection is in confirmed mode(1).
1: Tamper 1 event detection is in potential mode(2).

1. The effects of tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in *Table 344: TAMP interconnection*.
2. The effects of tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in *Table 344: TAMP interconnection*.

37.6.3 TAMP control register 3 (TAMP_CR3)

This register can be protected against nonsecure access. Refer to *Section 37.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 37.3.7: 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.	ITAMP13POM	ITAMP12POM	ITAMP11POM	Res.	ITAMP9POM	ITAMP8POM	ITAMP7POM	ITAMP6POM	ITAMP5POM	Res.	ITAMP3POM	Res.	Res.
			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 **ITAMP13POM**: Internal tamper 13 potential mode
0: Internal tamper 13 event detection is in confirmed mode(1).
1: Internal tamper 13 event detection is in potential mode(2).

Bit 11 **ITAMP12POM**: Internal tamper 12 potential mode
0: Internal tamper 12 event detection is in confirmed mode(1).
1: Internal tamper 12 event detection is in potential mode(2).

Bit 10 **ITAMP11POM**: Internal tamper 11 potential mode
0: Internal tamper 11 event detection is in confirmed mode(1).
1: Internal tamper 11 event detection is in potential mode(2).

Bit 9 Reserved, must be kept at reset value.

Bit 8 **ITAMP9POM**: Internal tamper 9 potential mode
0: Internal tamper 9 event detection is in confirmed mode(1).
1: Internal tamper 9 event detection is in potential mode(2).

Bit 7 **ITAMP8POM**: Internal tamper 8 potential mode
0: Internal tamper 8 event detection is in confirmed mode(1).
1: Internal tamper 8 event detection is in potential mode(2).

Bit 6 **ITAMP7POM**: Internal tamper 7 potential mode
0: Internal tamper 7 event detection is in confirmed mode(1).
1: Internal tamper 7 event detection is in potential mode(2).

Bit 5 **ITAMP6POM**: Internal tamper 6 potential mode
0: Internal tamper 6 event detection is in confirmed mode(1).
1: Internal tamper 6 event detection is in potential mode(2).

Bit 4 **ITAMP5POM**: Internal tamper 5 potential mode
0: Internal tamper 5 event detection is in confirmed mode(1).
1: Internal tamper 5 event detection is in potential mode(2).

Bit 3 Reserved, must be kept at reset value.

Bit 2 **ITAMP3POM**: Internal tamper 3 potential mode
0: Internal tamper 3 event detection is in confirmed mode(1).
1: Internal tamper 3 event detection is in potential mode(2).

Bit 1 Reserved, must be kept at reset value.

Bit 0 Reserved, must be kept at reset value.

1. The effects of internal tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in *Table 344: TAMP interconnection*
2. The effects of internal tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in *Table 344: TAMP interconnection*.

37.6.4 TAMP filter control register (TAMP_FLTCR)

This register can be protected against nonsecure access. Refer to *Section 37.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 37.3.7: 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.	TAMPPUDIS	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.

37.6.5 TAMP active tamper control register 1 (TAMP_ATCR1)

This register can be protected against nonsecure access. Refer to *Section 37.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 37.3.7: 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]		Res.	Res.	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

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 TAMPOUTSELi defined by ATOSELi bits.

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 347: 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.

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.
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
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

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
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

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
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

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
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

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.

37.6.6 TAMP active tamper seed register (TAMP_ATSEEDR)

This register can be protected against nonsecure access. Refer to *Section 37.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 37.3.7: 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.

37.6.7 TAMP active tamper output register (TAMP_ATOR)

This register can be protected against nonsecure access. Refer to *Section 37.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 37.3.7: 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 clock 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.

37.6.8 TAMP active tamper control register 2 (TAMP_ATCR2)

This register can be protected against nonsecure access. Refer to *Section 37.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 37.3.7: 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
Res.	Res.	Res.	Res.	Res.	Res.	ATOSEL6[2:0]			ATOSEL5[2:0]			ATOSEL4[2:0]			ATO SEL3 [2]
						rw	rw	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 31:26 Reserved, must be kept at reset value.

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
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

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
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

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
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.
Bits 18:17 are the mirror of ATOSEL4[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
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.
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
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.
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
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.
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.

37.6.9 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 nonsecure 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 37.3.7: 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
TAMP SEC	BHK LOCK	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
CNT1 SEC	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 nonsecure.

1: Tamper configuration and interrupt can be written only when the APB access is secure.

Note: Refer to Section 37.3.5: 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 nonsecure 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 nonsecure access.

If BKPWSEC = 32: there is no protection zone 3.

Refer to *Figure 384: Backup registers protection zones*.

Note: If TZEN=0: the protection zone 2 can be read and written with nonsecure 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 nonsecure.

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 384: Backup registers protection zones*.

Note: If TZEN=0: the protection zone 1 can be read and written with nonsecure access.

Note: If BKPRWPRIV is set, BKPRWSEC[7:0] can be written only in privileged mode.

37.6.10 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 nonsecure access depending on the TAMP_SECCFGR configuration (refer to *Section 37.3.5: 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 37.3.7: 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.

37.6.11 TAMP interrupt enable register (TAMP_IER)

This register can be protected against nonsecure access. Refer to *Section 37.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 37.3.7:*

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	Res.	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**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP6IE	TAMP5IE	TAMP4IE	TAMP3IE	TAMP2IE	TAMP1IE
										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 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 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 **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.

37.6.12 TAMP status register (TAMP_SR)

This register can be protected against nonsecure access. Refer to *Section 37.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 37.3.7: 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	Res.	Res.
			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.	Res.	Res.	TAMP6F	TAMP5F	TAMP4F	TAMP3F	TAMP2F	TAMP1F
										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 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 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 **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.

37.6.13 TAMP nonsecure masked interrupt status register (TAMP_MISR)

This register can be protected against non-privileged access. Refer to *Section 37.3.7: 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	Res.	Res.
			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.	Res.	Res.	TAMP6MF	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
										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 nonsecure interrupt masked flag

This flag is set by hardware when the internal tamper 13 nonsecure interrupt is raised.

Bit 27 **ITAMP12MF**: internal tamper 12 nonsecure interrupt masked flag

This flag is set by hardware when the internal tamper 12 nonsecure interrupt is raised.

Bit 26 **ITAMP11MF**: internal tamper 11 nonsecure interrupt masked flag

This flag is set by hardware when the internal tamper 11 nonsecure interrupt is raised.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9MF**: internal tamper 9 nonsecure interrupt masked flag

This flag is set by hardware when the internal tamper 9 nonsecure interrupt is raised.

Bit 23 **ITAMP8MF**: Internal tamper 8 nonsecure interrupt masked flag

This flag is set by hardware when the internal tamper 8 nonsecure interrupt is raised.

Bit 22 **ITAMP7MF**: Internal tamper 7 tamper nonsecure interrupt masked flag

This flag is set by hardware when the internal tamper 7 nonsecure interrupt is raised.

Bit 21 **ITAMP6MF**: Internal tamper 6 nonsecure interrupt masked flag

This flag is set by hardware when the internal tamper 6 nonsecure interrupt is raised.

Bit 20 **ITAMP5MF**: Internal tamper 5 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 5 nonsecure interrupt is raised.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **ITAMP3MF**: Internal tamper 3 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 3 nonsecure interrupt is raised.

Bit 17 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 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 **TAMP6MF**: TAMP6 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 6 nonsecure interrupt is raised.

Bit 4 **TAMP5MF**: TAMP5 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 5 nonsecure interrupt is raised.

Bit 3 **TAMP4MF**: TAMP4 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 4 nonsecure interrupt is raised.

Bit 2 **TAMP3MF**: TAMP3 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 3 nonsecure interrupt is raised.

Bit 1 **TAMP2MF**: TAMP2 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 2 nonsecure interrupt is raised.

Bit 0 **TAMP1MF**: TAMP1 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 1 nonsecure interrupt is raised.

37.6.14 TAMP secure masked interrupt status register (TAMP_SMISR)

This register can be protected against nonsecure access. Refer to *Section 37.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 37.3.7: 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	Res.	Res.
			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.	Res.	Res.	TAMP6MF	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
										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 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 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 **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.

37.6.15 TAMP status clear register (TAMP_SCR)

This register can be protected against nonsecure access. Refer to *Section 37.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 37.3.7: 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	Res.	Res.
			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.	Res.	Res.	CTAMP6F	CTAMP5F	CTAMP4F	CTAMP3F	CTAMP2F	CTAMP1F
										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 Reserved, must be kept at reset value.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 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 **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.

37.6.16 TAMP monotonic counter 1 register (TAMP_COUNT1R)

This register can be protected against nonsecure access. Refer to *Section 37.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 37.3.7: 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.

37.6.17 TAMP resources protection configuration register (TAMP_RPCFGR)

This register can be protected against nonsecure access. Refer to *Section 37.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 37.3.7: 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.	RPCFG5	RPCFG4	RPCFG3	RPCFG2	RPCFG1	Res.
										rw	rw	rw	rw	rw	

Bit 31 Reserved, must be kept at reset value.

Bits 30:8 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 **RPCFG5**: Configurable resource 5 protection(1)
  0: Resource 5 is not included in the device secrets protected by TAMP peripheral
  1: Resource 5 is included in the device secrets protected by TAMP peripheral

Bit 4 **RPCFG4**: Configurable resource 4 protection(2)
  0: Resource 4 is not included in the device secrets protected by TAMP peripheral
  1: Resource 4 is included in the device secrets protected by TAMP peripheral

Bit 3 **RPCFG3**: Configurable resource 3 protection(3)
  0: Resource 3 is not included in the device secrets protected by TAMP peripheral
  1: Resource 3 is included in the device secrets protected by TAMP peripheral

Bit 2 **RPCFG2**: Configurable resource 2 protection(4)
  0: Resource 2 is not included in the device secrets protected by TAMP peripheral
  1: Resource 2 is included in the device secrets protected by TAMP peripheral

Bit 1 **RPCFG1**: Configurable resource 1 protection(5)
  0: Resource 1 is not included in the device secrets protected by TAMP peripheral
  1: Resource 1 is included in the device secrets protected by TAMP peripheral

Bit 0 Reserved, must be kept at reset value.

1. Refer to tamp_confirmed_rpcfg5 and tamp_potential_rpcfg5 signals in *Table 342: TAMP input/output pins* and *Table 344: TAMP interconnection*.
2. Refer to tamp_confirmed_rpcfg4 and tamp_potential_rpcfg4 signals in *Table 342: TAMP input/output pins* and *Table 344: TAMP interconnection*.
3. Refer to tamp_confirmed_rpcfg3 and tamp_potential_rpcfg3 signals in *Table 342: TAMP input/output pins* and *Table 344: TAMP interconnection*.
4. Refer to tamp_confirmed_rpcfg2 and tamp_potential_rpcfg2 signals in *Table 342: TAMP input/output pins* and *Table 344: TAMP interconnection*.
5. Refer to tamp_confirmed_rpcfg1 and tamp_potential_rpcfg1 signals in *Table 342: TAMP input/output pins* and *Table 344: TAMP interconnection*.

37.6.18 TAMP backup x register (TAMP_BKPxR)

This register can be protected against nonsecure access. Refer to *Section 37.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 37.3.7: 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.

37.6.19 TAMP register map

Table 353. TAMP 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	TAMP_CR1	Res.	Res.	Res.	ITAMP13E	ITAMP12E	ITAMP11E	Res.	ITAMP9E	ITAMP8E	ITAMP7E	ITAMP6E	ITAMP5E	Res.	ITAMP3E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP6E	TAMP5E	TAMP4E	TAMP3E	TAMP2E	TAMP1E
	Reset value				0	0	0		0	0	0	0	0		0													0	0	0	0	0	0
0x04	TAMP_CR2	Res.	Res.	TAMP6TRG	TAMP5TRG	TAMP4TRG	TAMP3TRG	TAMP2TRG	TAMP1TRG	BKERASE	BKBLOCK	Res.	Res.	Res.	TAMP3MSK	TAMP2MSK	TAMP1MSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP6POM	TAMP5POM	TAMP4POM	TAMP3POM	TAMP2POM	TAMP1POM
	Reset value			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.	ITAMP13POM	ITAMP12POM	ITAMP11POM	Res.	ITAMP9POM	ITAMP8POM	ITAMP7POM	ITAMP6POM	ITAMP5POM	Res.	ITAMP3POM	Res.	Res.
	Reset value																				0	0	0		0	0	0	0	0		0		
0x0C	TAMP_FLTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]		
	Reset value																									0	0	0	0	0	0	0	0
0x10	TAMP_ATCR1	FLTEN	ATOSHARE	Res.	Res.	Res.	ATPER[2:0]			Res.	Res.	Res.	Res.	Res.	ATCKSEL[2:0]			ATOSEL4[1:0]		ATOSEL3[1:0]		ATOSEL2[1:0]		ATOSEL1[1:0]		Res.	Res.	TAMP6AM	TAMP5AM	TAMP4AM	TAMP3AM	TAMP2AM	TAMP1AM
	Reset value	0	0				0	0	0						1	1	1	0	0	0	0	0	0	0	0			0	0	0	0	0	0
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	Res.	Res.	Res.	Res.	Res.	Res.	ATOSEL6[2:0]			ATOSEL5[2:0]			ATOSEL4[2:0]			ATOSEL3[2:0]			ATOSEL2[2:0]			ATOSEL1[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								
0x20	TAMP_SECCFGR	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
0x24	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 353. TAMP 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	TAMP_IER	Res.	Res.	Res.	ITAMP13IE	ITAMP12IE	ITAMP11IE	Res.	ITAMP9IE	ITAMP8IE	ITAMP7IE	ITAMP6IE	ITAMP5IE	Res.	ITAMP3IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP6IE	TAMP5IE	TAMP4IE	TAMP3IE	TAMP2IE	TAMP1IE
	Reset value				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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP6F	TAMP5F	TAMP4F	TAMP3F	TAMP2F	TAMP1F
	Reset value				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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP6MF	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
	Reset value				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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP6MF	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
	Reset value				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	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP6F	CTAMP5F	CTAMP4F	CTAMP3F	CTAMP2F	CTAMP1F
	Reset value				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_ RPCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPCFG5	RPCFG4	RPCFG3	RPCFG2	RPCFG1	Res.
	Reset value																											0	0	0	0	0	
0x100 + 0x04*x, (x= 0 to 31)	TAMP_BKPxR	BKP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

38 Inter-integrated circuit interface (I2C)

38.1 I2C introduction

The I2C peripheral handles the interface between the device and the serial I²C (inter-integrated circuit) bus. It provides multicontroller capability, and controls all I²C-bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

The I2C peripheral is also SMBus (system management bus) and PMBus® (power management bus) compatible.

It can use DMA to reduce the CPU load.

38.2 I2C main features

- I²C-bus specification rev03 compatibility:
  - Target and controller modes
  - Multicontroller 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 target 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
  - Clock stretching (optional)
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters
- SMBus specification rev 3.0 compatibility[(a)]:
  - 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. To check the compliance of the GPIOs selected for SMBus with the specified logical levels, refer to the product datasheet.

- Wake-up from Stop mode
- Autonomous functionality in Stop mode

For information on I2C instantiation, refer to *Section 38.3: I2C implementation*.

38.3 I2C implementation

This section provides an implementation overview with respect to the I2C instantiation.

Table 354. I2C implementation

I2C features(1)	I2C1(2)	I2C3
7-bit addressing mode	X	X
10-bit addressing mode	X	X
Standard-mode (up to 100 kbit/s)	X	X
Fast-mode (up to 400 kbit/s)	X	X
Fast-mode Plus with 20 mA output drive I/Os (up to 1 Mbit/s)	X	X
Independent clock	X	X
Autonomous mode	X	X
Wake-up from Stop mode only (no autonomous mode)	X	X
SMBus/PMBus	X	X

1. X = supported.
2. Only available on STM32WBA52/54/55xx.

38.4 I2C functional description

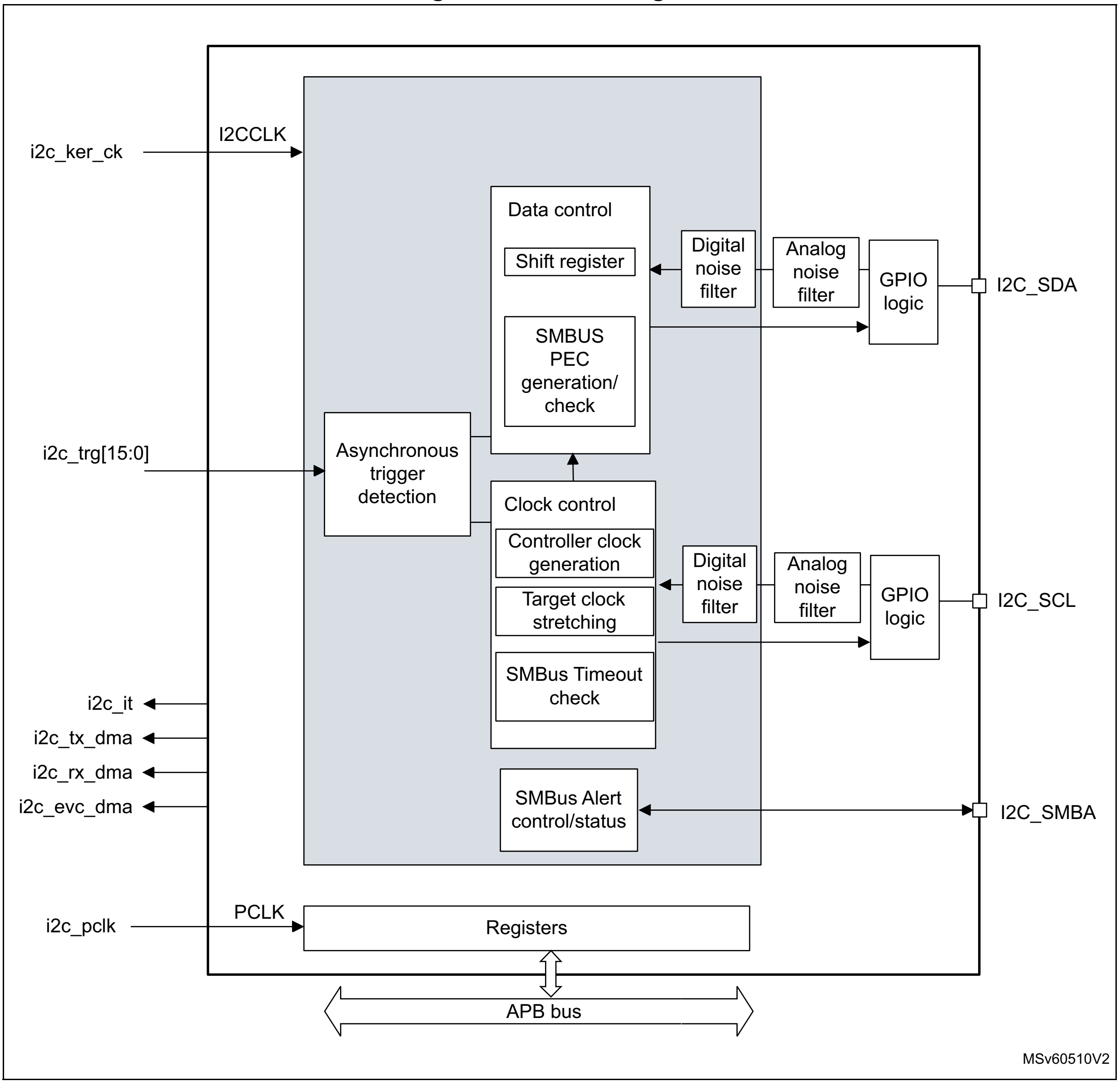
In addition to receiving and transmitting data, the peripheral converts them from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The peripheral is connected to the I²C-bus through a data pin (SDA) and a clock pin (SCL). It supports Standard-mode (up to 100 kHz), Fast-mode (up to 400 kHz), and Fast-mode Plus (up to 1 MHz) I²C-bus.

The peripheral can also be connected to an SMBus, through the data pin (SDA), the clock pin (SCL), and an optional SMBus alert pin (SMBA).

The independent clock function allows the I2C communication speed to be independent of the i2c_pclk frequency.

38.4.1 I2C block diagram

Figure 388. Block diagram



38.4.2 I2C pins and internal signals

Table 355. I2C input/output pins

Pin name	Signal type	Description
I2C_SDA	Bidirectional	I²C-bus data
I2C_SCL	Bidirectional	I²C-bus clock
I2C_SMBA	Bidirectional	SMBus alert

Table 356. 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 371* for the 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 357. I2C1 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(1)
i2c_trg9	COMP2_OUT(1)
i2c_trg10	rtc_alra_trg
i2c_trg11	rtc_wut_trg
i2c_trg12..15	Reserved

1. Available only on STM32WBA54/55xx devices.

Table 358. I2C3 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	exti8
i2c_trg6	lptim1_ch1

Table 358. I2C3 interconnection (continued)

Signal name	Source/destination
i2c_trg7	Reserved
i2c_trg8	COMP1_OUT(1)
i2c_trg9	COMP2_OUT(1)
i2c_trg10	rtc_alra_trg
i2c_trg11	rtc_wut_trg
i2c_trg12..15	Reserved

1. Available only on STM32WBA54/55xx devices.

38.4.3 I2C clock requirements

The I2C kernel is clocked by i2c_ker_ck.

The i2c_ker_ck period t_{I2CCLK} must respect the following conditions:

$t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4$

$t_{I2CCLK} < t_{HIGH}$

where t_{LOW} is the SCL low time, t_{HIGH} is the SCL high time, and $t_{filters}$ is the sum of the analog and digital filter delays (when enabled).

The digital filter delay is DNF[3:0] x t_{I2CCLK}.

The i2c_pclk clock period t_{PCLK} must respect the condition $t_{PCLK} < 4/3\ t_{SCL}$, where t_{SCL} is the SCL period.

Caution: When the I2C kernel is clocked by i2c_pclk, this clock must respect the conditions for t_{I2CCLK}.

38.4.4 I2C mode selection

The peripheral can operate as:

- Target transmitter
- Target receiver
- Controller transmitter
- Controller receiver

By default, the peripheral operates in target mode. It automatically switches from target to controller mode upon generating START condition, and from controller to target mode upon arbitration loss or upon generating STOP condition. This allows the use of the I2C peripheral in a multicontroller I²C-bus environment.

Communication flow

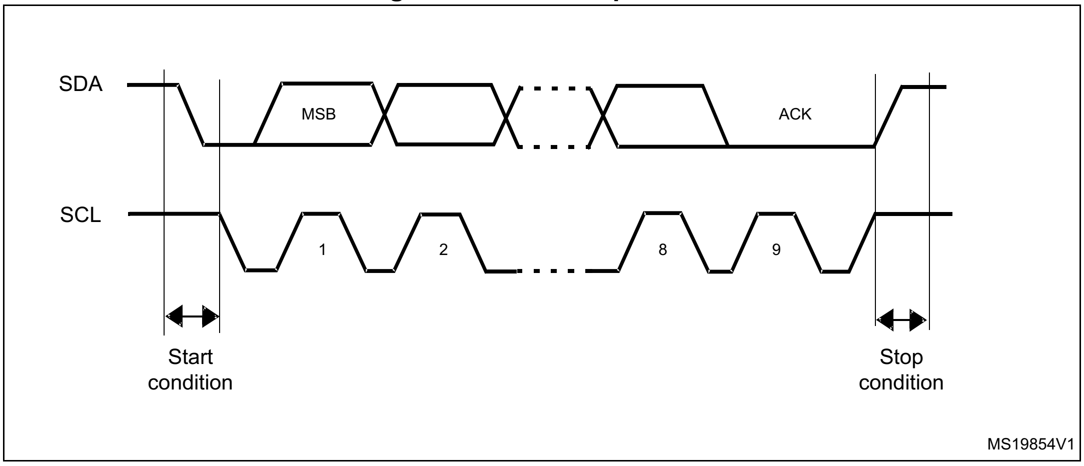
In controller mode, the I2C peripheral initiates a data transfer and generates the clock signal. Serial data transfers always begin with a START condition and end with a STOP condition. Both START and STOP conditions are generated in controller mode by software.

In target mode, the peripheral recognizes its own 7-bit or 10-bit address, 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 address is contained in the first byte (7-bit addressing) or in the first two bytes (10-bit addressing) following the START condition. The address is always transmitted in controller mode.

The following figure shows the transmission of a single byte. The controller generates nine SCL pulses. The transmitter sends the eight data bits to the receiver with the SCL pulses 1 to 8. Then the receiver sends the acknowledge bit to the transmitter with the ninth SCL pulse.

Figure 389. I²C-bus protocol



The acknowledge can be enabled or disabled by software. The own addresses of the I2C peripheral can be selected by software.

38.4.5 I2C initialization

Enabling and disabling the peripheral

Before enabling the I2C peripheral, configure and enable its clock through the RCC, and initialize its control registers.

The I2C peripheral can then be enabled by setting the PE bit of the I2C_CR1 register.

Disabling the I2C peripheral by clearing the PE bit resets the I2C peripheral. Refer to *Section 38.4.6* for more details.

Noise filters

Before enabling the I2C peripheral by setting the PE bit of the I2C_CR1 register, the user must configure the analog and/or digital noise filters, as required.

The analog noise filter on the SDA and SCL inputs complies with the I²C-bus specification which requires, in Fast-mode and Fast-mode Plus, the suppression of spikes shorter than 50 ns. Enabled by default, it can be disabled by setting the ANFOFF bit.

The digital filter is controlled through the DNF[3:0] bitfield of the I2C_CR1 register. When it is enabled, the internal SCL and SDA signals only take the level of their corresponding I²C-bus line when remaining stable for more than DNF[3:0] periods of i2c_ker_ck. This allows suppressing spikes shorter than the filtering capacity period programmable from one to fifteen i2c_ker_ck periods.

The following table compares the two filters.

Table 359. Comparison of analog and digital filters

Item	**Analog filter**	**Digital filter**
Filtering capacity(1)	≥ 50 ns	One to fifteen i2c_ker_ck periods
Benefits	Available in Stop mode	– Programmable filtering capacity – Extra filtering capability versus I²C-bus specification requirements – Stable filtering capacity
Drawbacks	Filtering capacity variation with temperature, voltage, and silicon process	Functionality in Stop mode not supported when the digital filter is enabled

1. Maximum duration of spikes that the filter can suppress

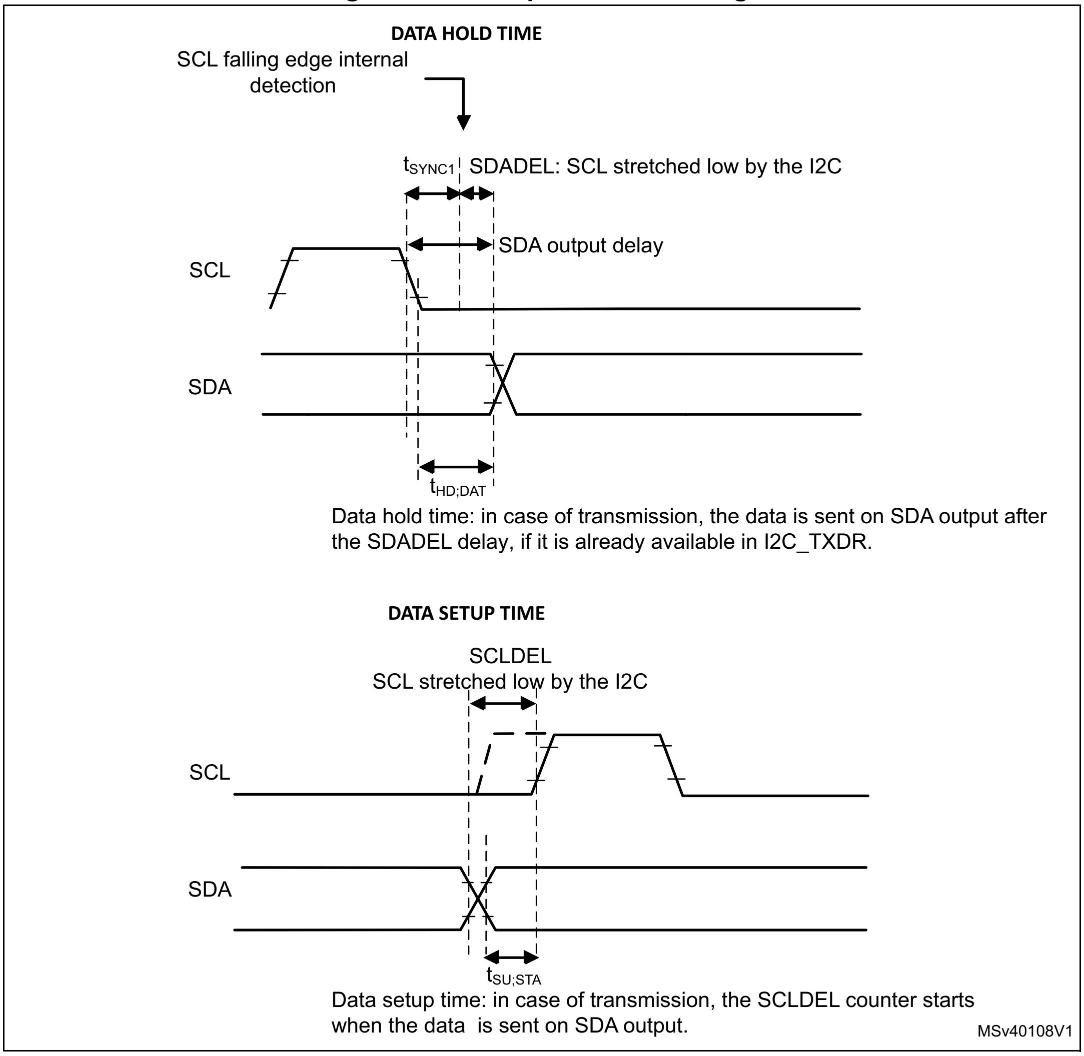
Caution: The filter configuration cannot be changed when the I2C peripheral is enabled.

I2C timings

To ensure correct data hold and setup times, the corresponding timings must be configured through the PRESC[3:0], SCLDEL[3:0], and SDADEL[3:0] bitfields of the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the *I2C configuration* window.

Figure 390. Setup and hold timings



When the SCL falling edge is internally detected, the delay t_{SDADEL} (impacting the hold time $t_{HD;DAT}$) is inserted before sending SDA output:

t_{SDADEL} = SDADEL x t_{PRESC} + t_{I2CCLK}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}.

The total SDA output delay is:

t_{SYNC1} + {[SDADEL x (PRESC + 1) + 1] x t_{I2CCLK}}

The t_{SYNC1} duration depends upon:

- SCL falling slope
- input delay $t_{AF(min)} < t_{AF} < t_{AF(max)}$ introduced by the analog filter (if enabled)
- input delay t_{DNF} = DNF x t_{I2CCLK} introduced by the digital filter (if enabled)
- delay due to SCL synchronization to i2c_ker_ck clock (two to three i2c_ker_ck periods)

To bridge the undefined region of the SCL falling edge, the user must set SDADEL[3:0] so as to fulfill the following condition:

{$t_{f(max)}$ + $t_{HD;DAT(min)}$ - $t_{AF(min)}$ - [(DNF + 3) x t_{I2CCLK}]} / {(PRESC + 1) x t_{I2CCLK}} ≤ SDADEL

SDADEL ≤ {$t_{HD;DAT\ (max)}$ - $t_{AF(max)}$ - [(DNF + 4) x t_{I2CCLK}]} / {(PRESC + 1) x t_{I2CCLK}}

Note: $t_{AF(min)}$ and $t_{AF(max)}$ are only part of the condition when the analog filter is enabled. Refer to the device datasheet for t_{AF} values.

The $t_{HD;DAT}$ time can at maximum be 3.45 µs for Standard-mode, 0.9 µs for Fast-mode, and 0.45 µs for Fast-mode Plus. It must be lower 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. When it stretches 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. The previous condition then becomes:

SDADEL ≤ $\{t_{VD;DAT\ (max)} - t_{r\ (max)} - t_{AF\ (max)} - [(DNF + 4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$

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[3:0] value.

After t_{SDADEL}, or after sending SDA output when the target had to stretch the clock because the data was not yet written in I2C_TXDR register, the SCL line is kept at low level during the setup time. This setup time is $t_{SCLDEL} = (SCLDEL + 1) \times t_{PRESC}$, where $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$. t_{SCLDEL} impacts the setup time $t_{SU;DAT}$.

To bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program SCLDEL[3:0] so as to fulfill the following condition:

$\{[t_{r\ (max)} + t_{SU;DAT\ (min)}] / [(PRESC + 1) \times t_{I2CCLK}]\} - 1 \leq$ SCLDEL

Refer to the following table for t_f, t_r, $t_{HD;DAT}$, $t_{VD;DAT}$, and $t_{SU;DAT}$ standard values.

Use the SDA and SCL real transition time values measured in the application to widen the scope of allowed SDADEL[3:0] and SCLDEL[3:0] values. Use the maximum SDA and SCL transition time values defined in the standard to make the device work reliably regardless of the application.

Note: At every clock pulse, after SCL falling edge detection, I2C operating as controller or target 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 SDA delay elapses, the I2C peripheral keeps 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.

When the NOSTRETCH bit is set in target mode, the SCL is not stretched. The SDADEL[3:0] must then be programmed so that it ensures a sufficient setup time.

Table 360. I²C-bus and 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 controller mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0], and SCLL[7:0] bitfields of the I2C_TIMINGR register.

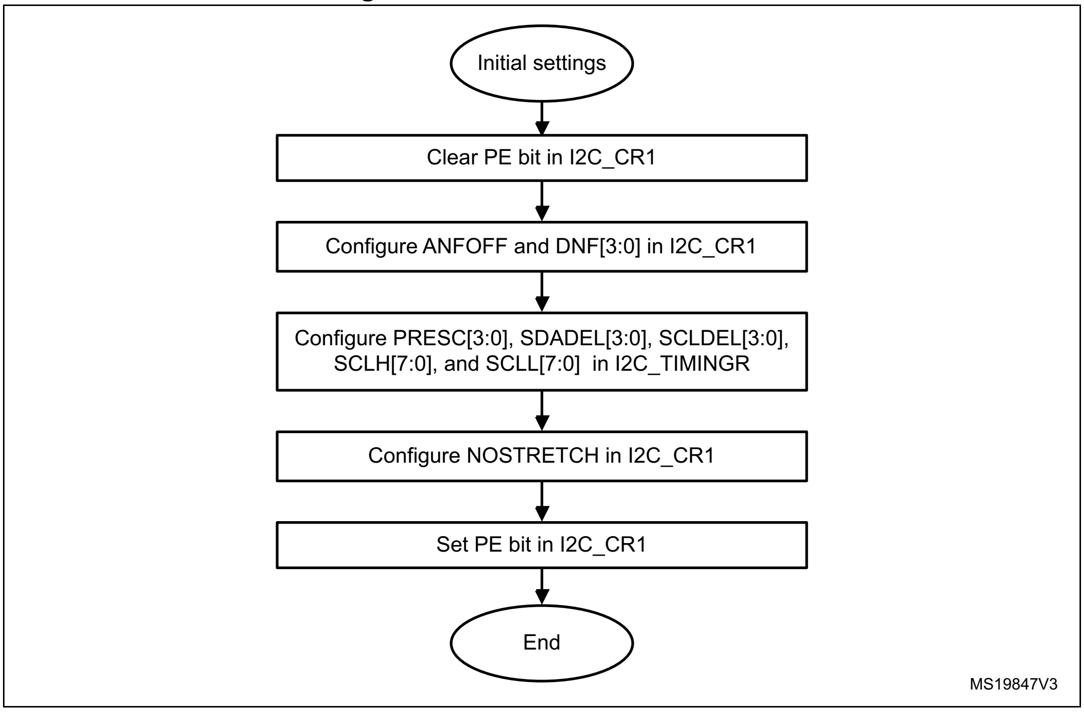
When the SCL falling edge is internally detected, the I2C peripheral releasing the SCL output after the delay t_{SCLL} = (SCLL + 1) x t_{PRESC}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}. The t_{SCLL} delay impacts the SCL low time t_{LOW}.

When the SCL rising edge is internally detected, the I2C peripheral forces the SCL output to low level after the delay t_{SCLH} = (SCLH + 1) x t_{PRESC}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}. The t_{SCLH} impacts the SCL high time t_{HIGH}.

Refer to *I2C controller initialization* for more details.

Caution: Changing the timing configuration and the NOSTRETCH configuration is not allowed when the I2C peripheral is enabled. Like the timing settings, the target NOSTRETCH settings must also be done before enabling the peripheral. Refer to *I2C target initialization* for more details.

Figure 391. I2C initialization flow



38.4.6 I2C reset

The reset of the I2C peripheral is performed by clearing the PE bit of the I2C_CR1 register. It has the effect of releasing the SCL and SDA lines. Internal state machines are reset and the communication control bits and the status bits revert to their reset values. This reset does not impact the configuration registers.

The impacted register bits are:

1. I2C_CR2 register: START, STOP, PECBYTE, and NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, PECERR, TIMEOUT, ALERT, and OVR

PE must be kept low during at least three APB clock cycles to perform the I2C reset. To ensure this, perform the following software sequence:

1. Write PE = 0
2. Check PE = 0
3. Write PE = 1

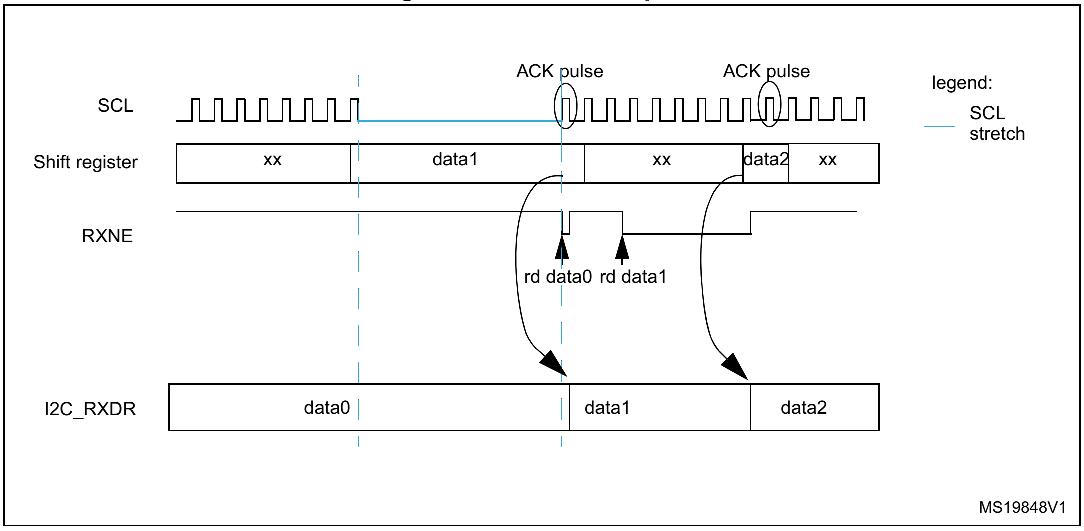
38.4.7 I2C data transfer

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 the I2C_RXDR register if it is empty (RXNE = 0). If RXNE = 1, which means that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch occurs between the eighth and the ninth SCL pulse (before the acknowledge pulse).

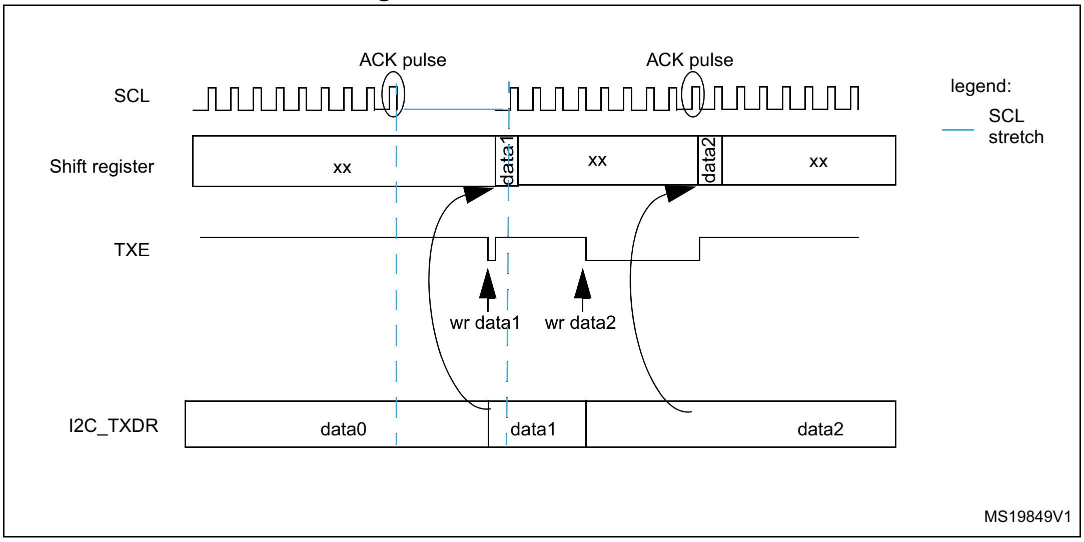
Figure 392. Data reception



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 the SDA line. If TXE = 1, which means that no data is written yet in I2C_TXDR, the SCL line is stretched low until I2C_TXDR is written. The stretch starts after the ninth SCL pulse.

Figure 393. Data transmission



Hardware transfer management

The I2C features an embedded byte counter to manage byte transfer and to close the communication in various modes, such as:

- – NACK, STOP and ReSTART generation in controller mode
- – ACK control in target receiver mode
- – PEC generation/checking

In controller mode, the byte counter is always used. By default, it is disabled in target mode. It can be enabled by software, by setting the SBC (target byte control) bit of the I2C_CR1 register.

The number of bytes to transfer is programmed in the NBYTES[7:0] bitfield of the I2C_CR2 register. If this number 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 of the I2C_CR2 register. In this mode, the TCR flag is set when the number of bytes programmed in NBYTES[7:0] is transferred (when the associated counter reaches zero), and an interrupt is generated if TCIE is set. SCL is stretched as long as the TCR flag is set. TCR is cleared by software when NBYTES[7:0] is written to a non-zero value.

When NBYTES[7:0] is reloaded with the last number of bytes to transfer, the RELOAD bit must be cleared.

When RELOAD = 0 in controller mode, the counter can be used in two modes:

- **Automatic end** (AUTOEND = 1 in the I2C_CR2 register). In this mode, the controller automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bitfield is transferred.
- **Software end** (AUTOEND = 0 in the I2C_CR2 register). In this mode, a software action is expected once the number of bytes programmed in the NBYTES[7:0] bitfield 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 of the I2C_CR2 register is set. This mode must be used when the controller wants to send a RESTART condition.

Caution: The AUTOEND bit has no effect when the RELOAD bit is set.

Table 361. I2C configuration

Function	SBC bit	RELOAD bit	AUTOEND bit
Controller Tx/Rx NBYTES + STOP	X	0	1
Controller Tx/Rx + NBYTES + RESTART	X	0	0
Target Tx/Rx, all received bytes ACKed	0	X	X
Target Rx with ACK control	1	1	X

38.4.8 I2C target mode

I2C target initialization

To work in target mode, the user must enable at least one target address. The I2C_OAR1 and I2C_OAR2 registers are available to program the target own addresses OA1 and OA2, respectively.

OA1 can be configured either in 7-bit (default) or in 10-bit addressing mode, by setting the OA1MODE bit of the I2C_OAR1 register.

OA1 is enabled by setting the OA1EN bit of the I2C_OAR1 register.

If an additional target addresses are required, the second target address OA2 can be configured. Up to seven OA2 LSBs can be masked, by configuring the OA2MSK[2:0] bitfield of the I2C_OAR2 register. Therefore, for OA2MSK[2:0] 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. When OA2MSK[2:0] is other than 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX) and they are not acknowledged. If OA2MSK[2:0] = 7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

When enabled through the specific bit, the reserved addresses can be acknowledged if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK[2:0] = 0.

OA2 is enabled by setting the OA2EN bit of the I2C_OAR2 register.

The general call address is enabled by setting the GCEN bit of the I2C_CR1 register.

When the I2C peripheral 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 target uses its clock stretching capability, which means that it stretches the SCL signal at low level when required, to perform software actions. If the controller does not support clock stretching, 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] bitfield of the I2C_ISR register to check which address matched. The DIR flag must also be checked to know the transfer direction.

Target with clock stretching

As long as the NOSTRETCH bit of the I2C_CR1 register is zero (default), the I2C peripheral operating as an I²C-bus target stretches the SCL signal in the following situations:

- The ADDR flag is set and the received address matches with one of the enabled target addresses.
  The stretch is released when the software clears the ADDR flag by setting the ADDRCF bit.
- In transmission, the previous data transmission is completed and no new data is written in I2C_TXDR register, or the first data byte is not written when the ADDR flag is cleared (TXE = 1).
  The stretch is released when the data is written to the I2C_TXDR register.
- In reception, the I2C_RXDR register is not read yet and a new data reception is completed.
  The stretch is released when I2C_RXDR is read.
- In target byte control mode (SBC bit set) with reload (RELOAD bit set), the last data byte transfer is finished (TCR bit set).
  The stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] bitfield.
- After SCL falling edge detection.
  The stretch is released after [(SDADEL + SCLDEL + 1) x (PRESC+ 1) + 1] x t_{I2CCLK} period.

Target without clock stretching

As long as the NOSTRETCH bit of the I2C_CR1 register is set, the I2C peripheral operating as an I²C-bus target 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 of the I2C_CR1 register is set. 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, it 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 of the I2C_CR1 register is set.

Target byte control mode

To allow byte ACK control in target reception mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. This is required to comply with SMBus standards.

The reload mode must be selected to allow byte ACK control in target reception mode (RELOAD = 1). To get control of each byte, NBYTES[7:0] 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 the ninth SCL pulse. The user can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit of 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 the next byte can be received.

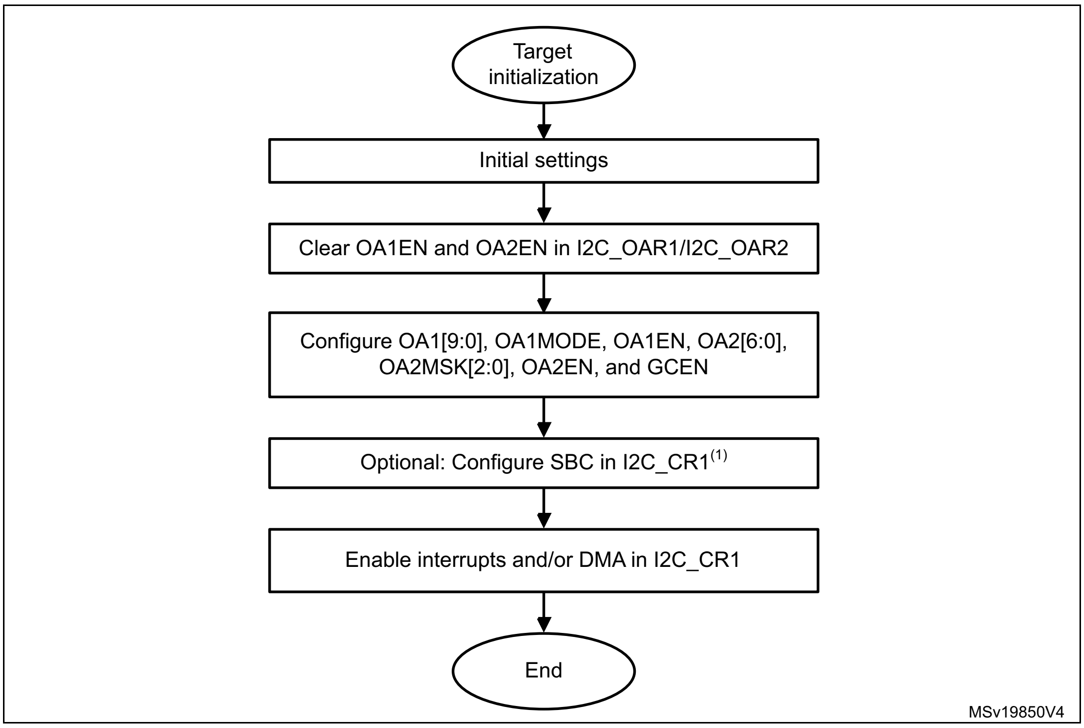
NBYTES[7:0] can be loaded with a value greater than 0x1. Receiving then continues until the corresponding number of bytes are received.

Note: The SBC bit must be configured when the I2C peripheral is disabled, when the target is not addressed, or when ADDR = 1.

The RELOAD bit value can be changed when ADDR = 1, or when TCR = 1.

Caution: The target byte control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH = 1 is not allowed.

Figure 394. Target initialization flow



1. SBC must be set to support SMBus features.

Target transmitter

A transmit interrupt status (TXIS) flag is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit of the I2C_CR1 register is set.

The TXIS flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

When NACK is received, the NACKF flag is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit of the I2C_CR1 register is set. The target automatically releases the SCL and SDA lines to let the controller perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When STOP is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag of the I2C_ISR register is set and an interrupt is generated. In most applications, the SBC bit is usually programmed to 0. In this case, if TXE = 0 when the target 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 target byte control mode (SBC = 1), the number of bytes to transmit must be programmed in NBYTES[7:0] 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[7:0].

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 to program the first data byte. The first data byte to send must be previously programmed in the I2C_TXDR register:

- This data can be the one written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to send, the I2C_TXDR register can be flushed, by setting the TXE bit, to program a new data byte. The STOPF bit must be cleared only after these actions. This guarantees 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 (transmit interrupt or transmit DMA request) is required, the user must set the TXIS bit in addition to the TXE bit, to generate the event.

Figure 395. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 0

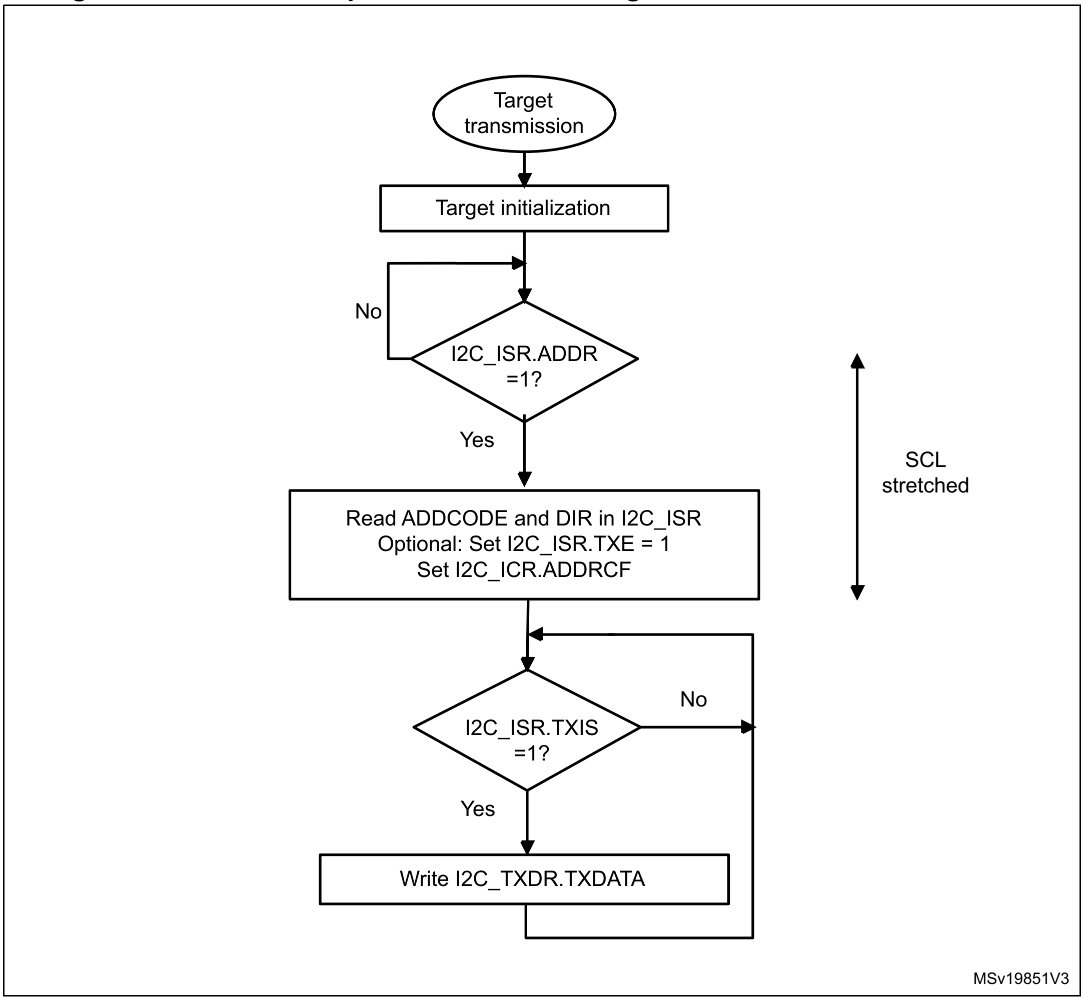


Figure 396. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 1

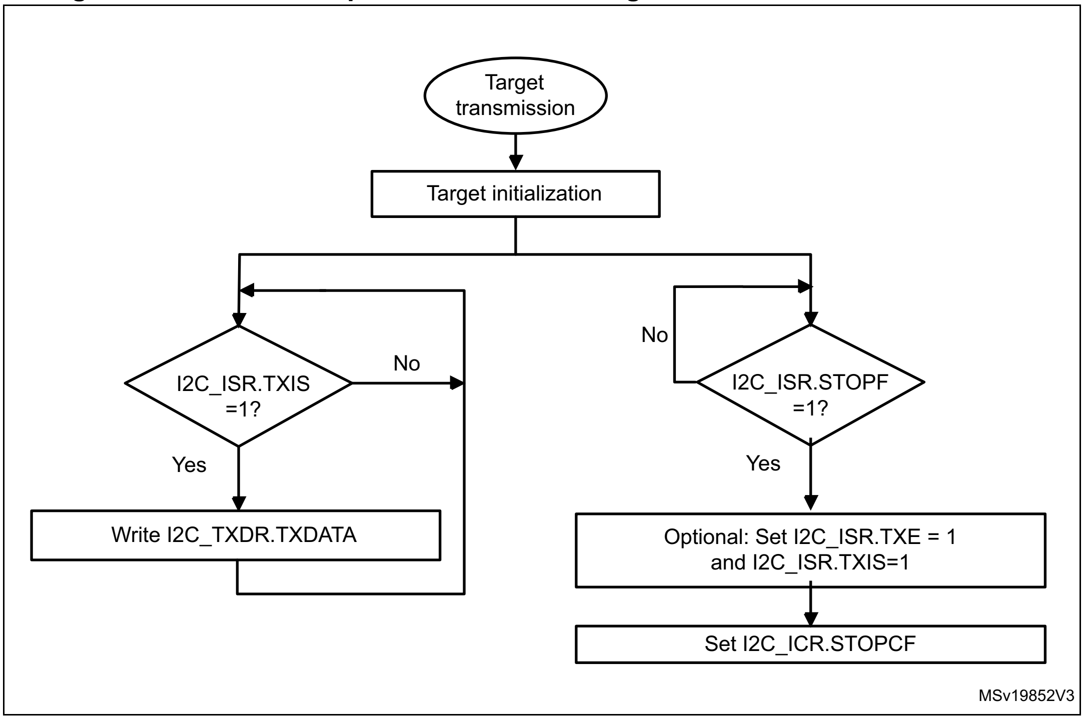
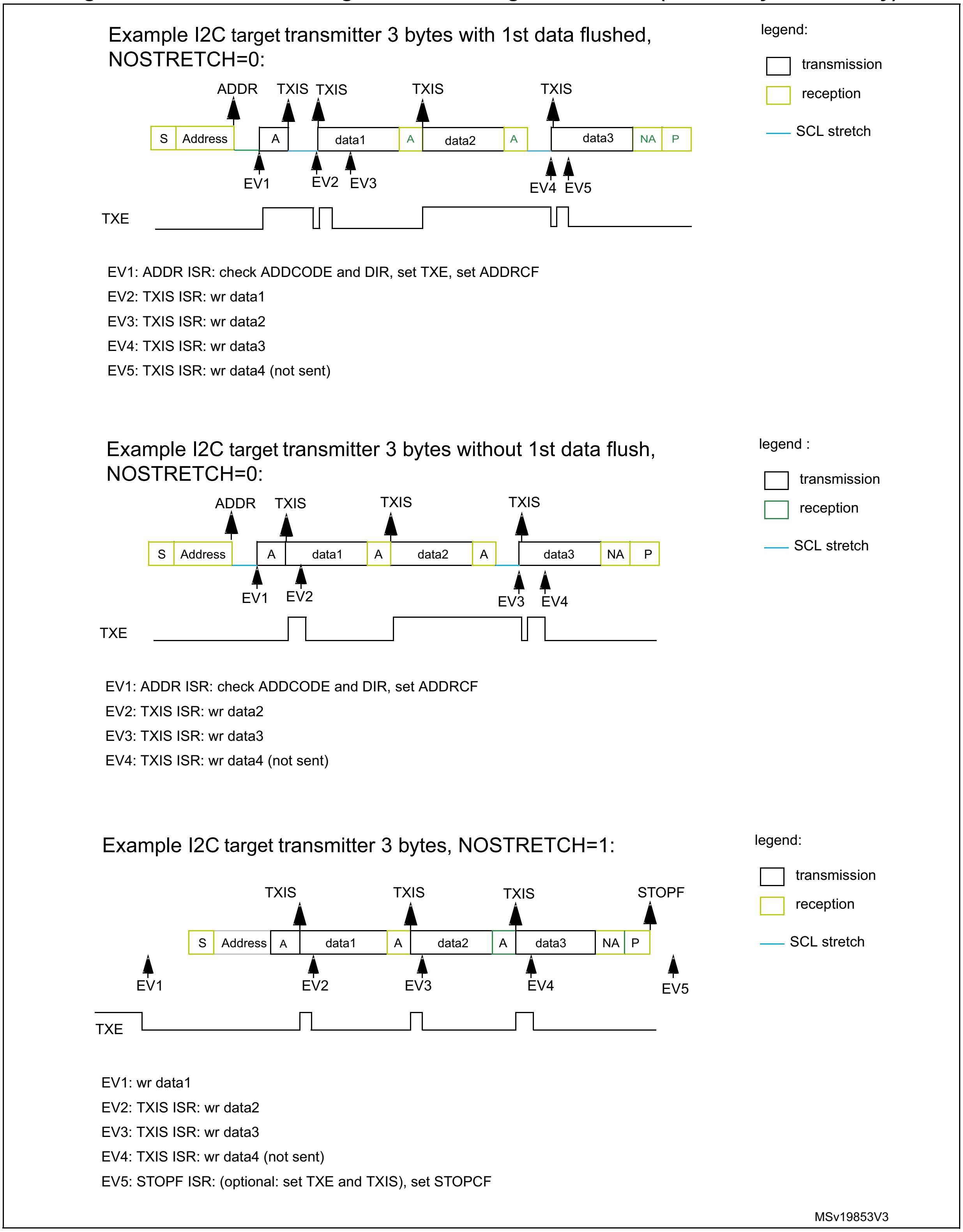


Figure 397. Transfer bus diagrams for I2C target transmitter (mandatory events only)



Target receiver

The RXNE bit of the I2C_ISR register is set when the I2C_RXDR is full, which generates an interrupt if the RXIE bit of the I2C_CR1 register is set. RXNE is cleared when I2C_RXDR is read.

When STOP condition is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag in the I2C_ISR register is set and an interrupt is generated.

Figure 398. Transfer sequence flow for I2C target receiver, NOSTRETCH = 0

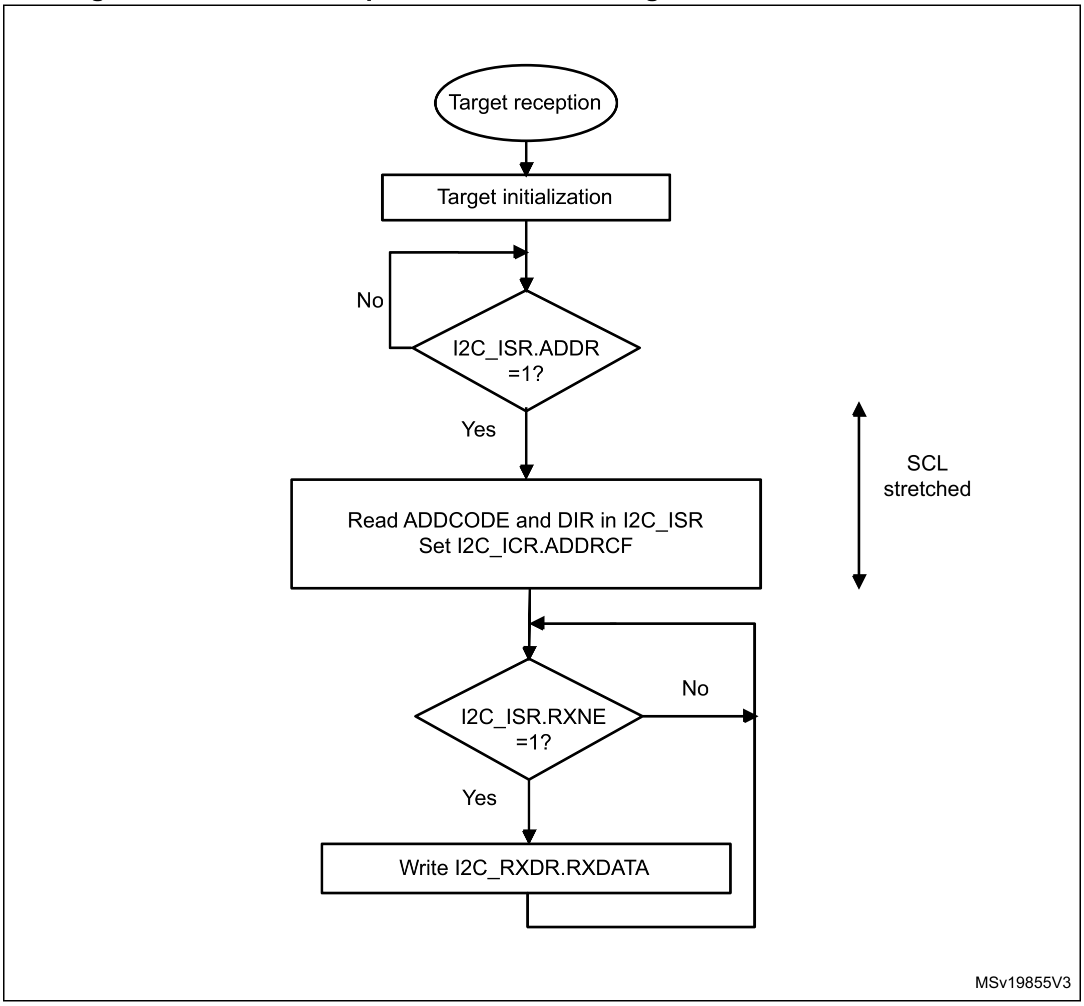


Figure 399. Transfer sequence flow for I2C target receiver, NOSTRETCH = 1

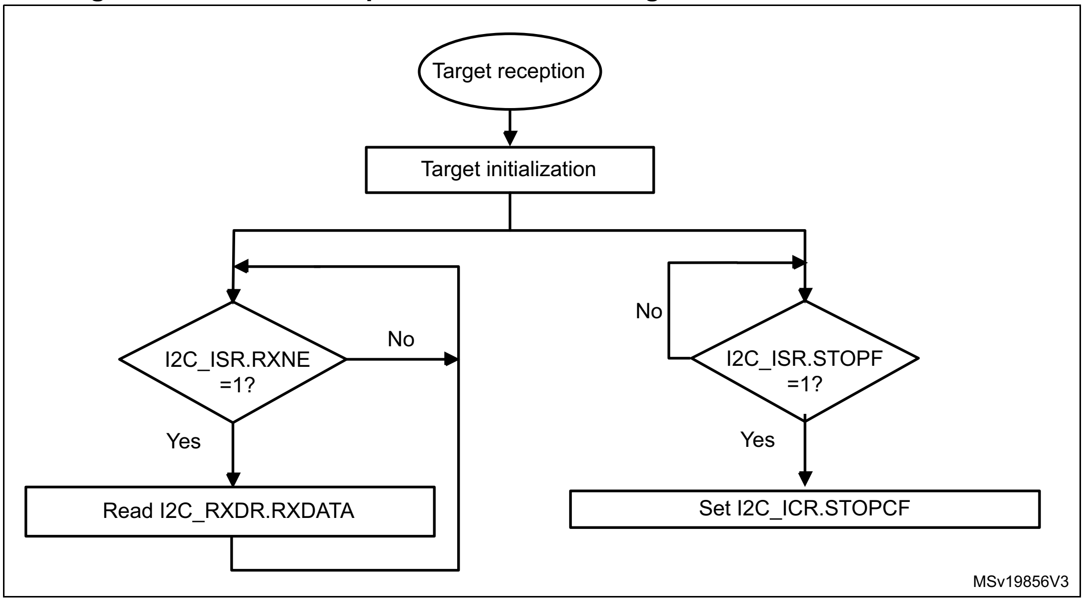
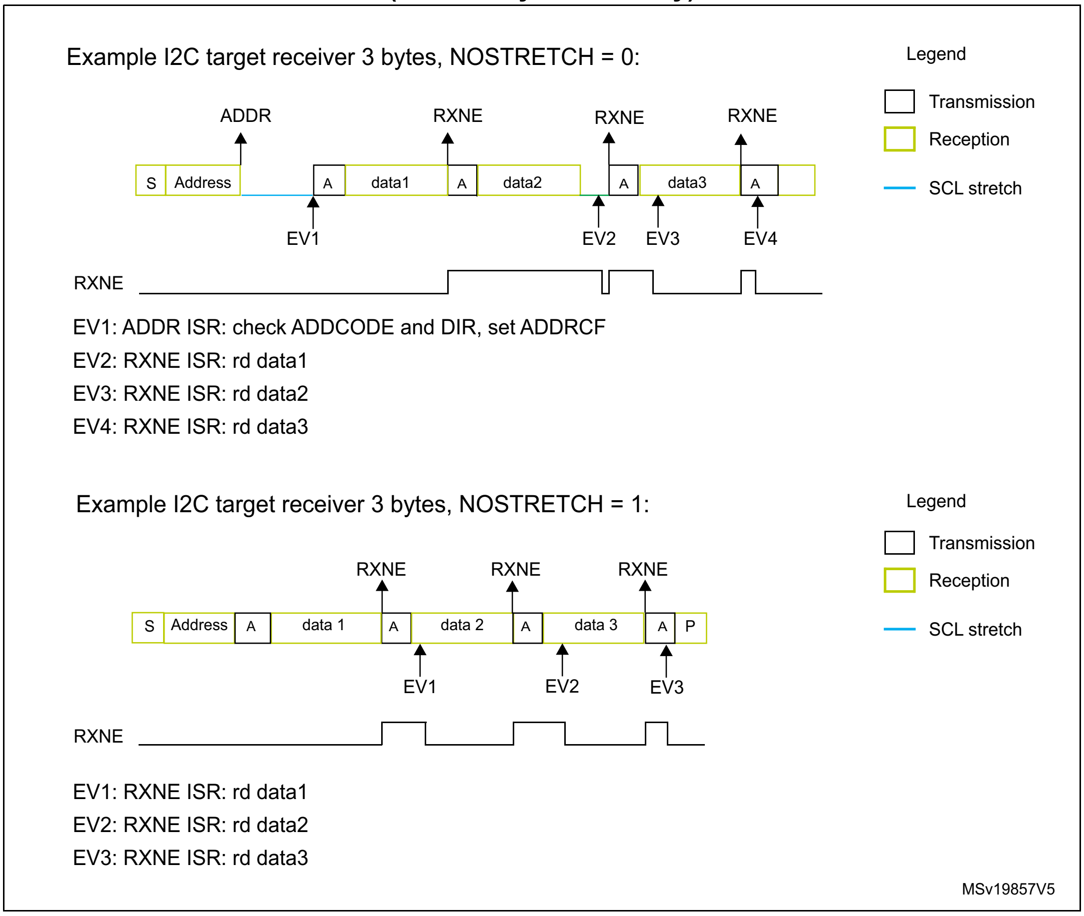


Figure 400. Transfer bus diagrams for I2C target receiver (mandatory events only)



38.4.9 I2C controller mode

I2C controller initialization

Before enabling the peripheral, the I2C controller 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 multicontroller environment and target 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.

I2C detects its own SCL low level after a t_{SYNC1} delay depending on the SCL falling edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bitfield of the I2C_TIMINGR register.

I2C detects its own SCL high level after a t_{SYNC2} delay depending on the SCL rising edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C ties SCL to low level once the SCLH counter reaches the value programmed in the SCLH[7:0] bitfield of the I2C_TIMINGR register.

Consequently the controller clock period is:

$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+ 1) + (SCLL+ 1)] \times (PRESC+ 1) \times t_{I2CCLK}\}$

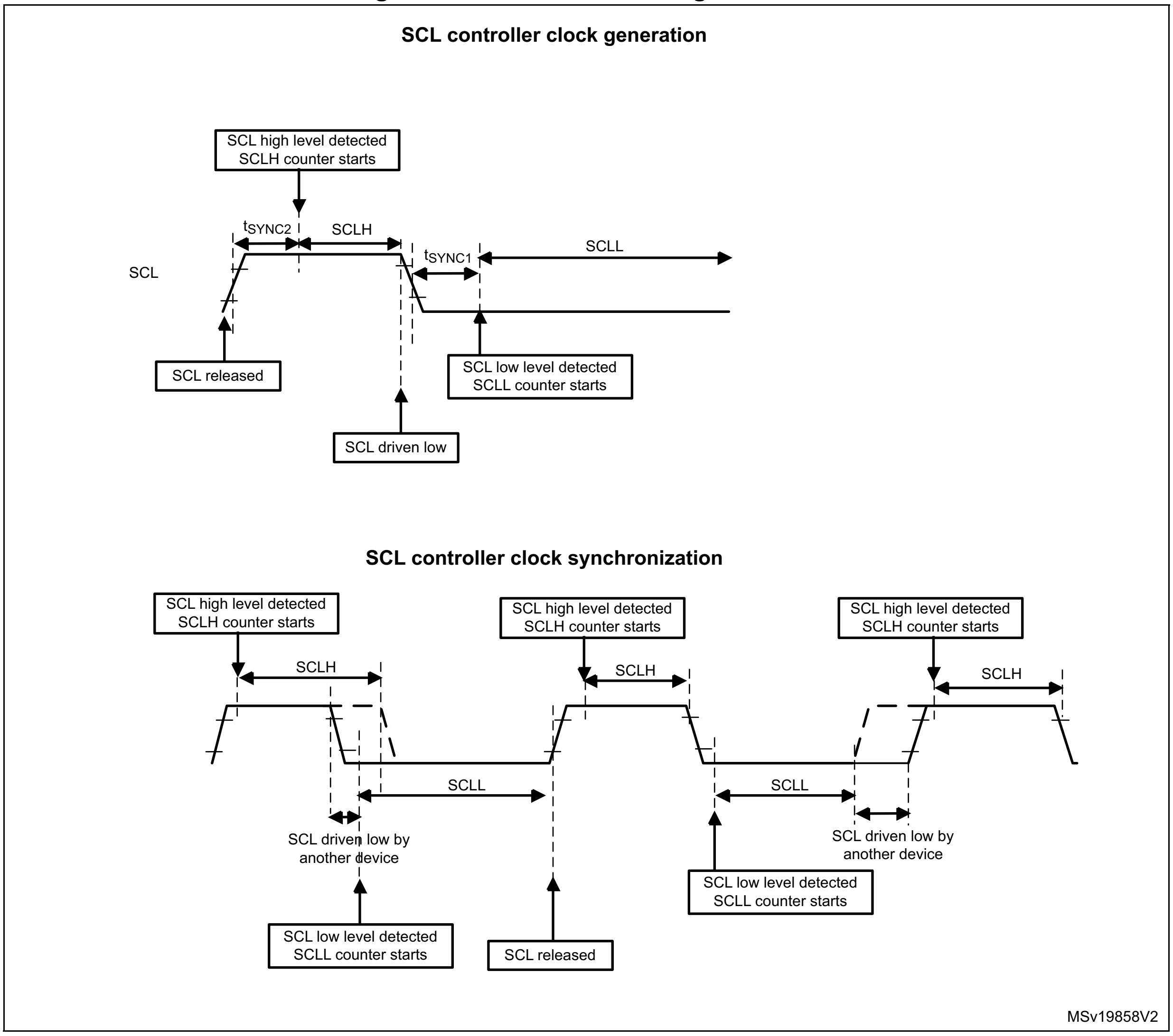
The duration of t_{SYNC1} depends upon:

- SCL falling slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): DNF[3:0] x t_{I2CCLK}
- delay due to SCL synchronization with the i2c_ker_ck clock (two to three i2c_ker_ck periods)

The duration of t_{SYNC2} depends upon:

- SCL rising slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): DNF[3:0] x t_{I2CCLK}
- delay due to SCL synchronization with the i2c_ker_ck clock (two to three i2c_ker_ck periods)

Figure 401. Controller clock generation



Caution: For compliance with the I²C-bus or SMBus specification, the controller clock must respect the timings in the following table.

Table 362. I²C-bus and 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}	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: The SCLL[7:0] bitfield also determines the t_{BUF} and $t_{SU:STA}$ timings and SCLH[7:0] the $t_{HD:STA}$ and $t_{SU:STO}$ timings.

Refer to Section 38.4.10 for examples of I2C_TIMINGR settings versus the i2c_ker_ck frequency.

Controller communication initialization (address phase)

To initiate the communication with a target to address, set the following bitfields of the I2C_CR2 register:

- ADD10: addressing mode (7-bit or 10-bit)
- SADD[9:0]: target address to send
- RD_WRN: transfer direction
- HEAD10R: in case of 10-bit address read, this bit determines whether the header only (for direction change) or the complete address sequence is sent.
- NBYTES[7:0]: the number of bytes to transfer; if equal to or greater than 255 bytes, the bitfield must initially be set to 0xFF.

Note: Changing these bitfields is not allowed as long as the START bit is set.

Before launching the communication, make sure that the I²C-bus is idle. This can be checked using the bus idle detection function or by verifying that the IDR bits of the GPIOs selected as SDA and SCL are set. Any low-level incident on the I²C-bus lines that coincides with the START condition asserted by the I2C peripheral may cause its deadlock if not filtered out by the input filters. If such incidents cannot be prevented, design the software so that it restores the normal operation of the I2C peripheral in case of a deadlock, by toggling the PE bit of the I2C_CR1 register.

To launch the communication, set the START bit of the I2C_CR2 register. The controller then automatically sends a START condition followed by the target address, either immediately if the BUSY flag is low, or t_{BUF} time after the BUSY flag transits from high to low state. The BUSY flag is set upon sending the START condition.

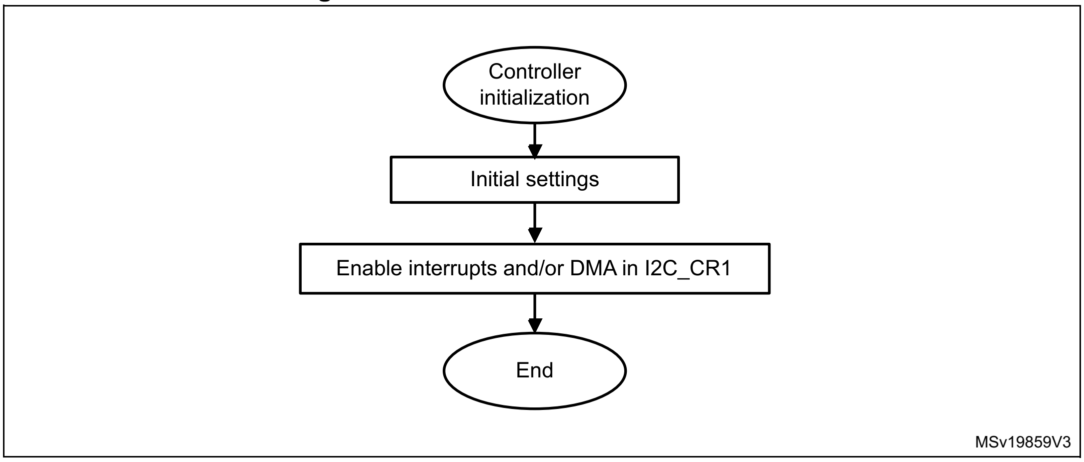
In case of an arbitration loss, the controller automatically switches back to target mode and can acknowledge its own address if it is addressed as a target.

Note: The START bit is reset by hardware when the target address is sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware upon arbitration loss.

In 10-bit addressing mode, the controller automatically keeps resending the target address in a loop until the first address byte (first seven address bits) is acknowledged by the target. Setting the ADDRCF bit makes I2C quit that loop.
If the I2C peripheral is addressed as a target (ADDR = 1) while the START bit is set, the I2C peripheral switches to target mode and the START bit is cleared.

Note: The same procedure is applied for a repeated START condition. In this case, BUSY = 1.

Figure 402. Controller initialization flow

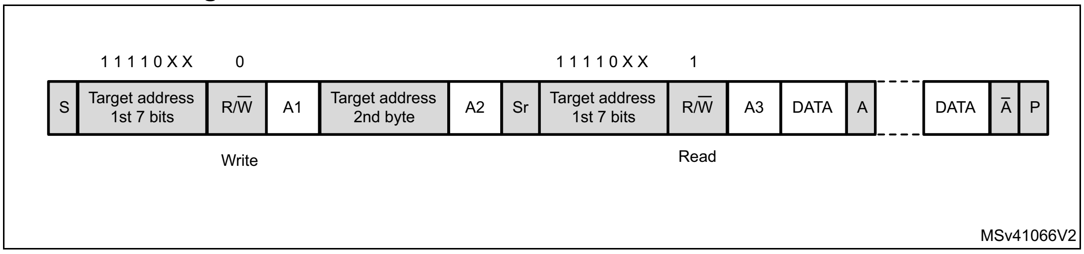


Initialization of a controller receiver addressing a 10-bit address target

If the target address is in 10-bit format, the user can choose to send the complete read sequence, by clearing the HEAD10R bit of the I2C_CR2 register. In this case, the controller automatically sends the following complete sequence after the START bit is set:

(RE)START + Target address 10-bit header Write + Target address second byte + (RE)START + Target address 10-bit header Read.

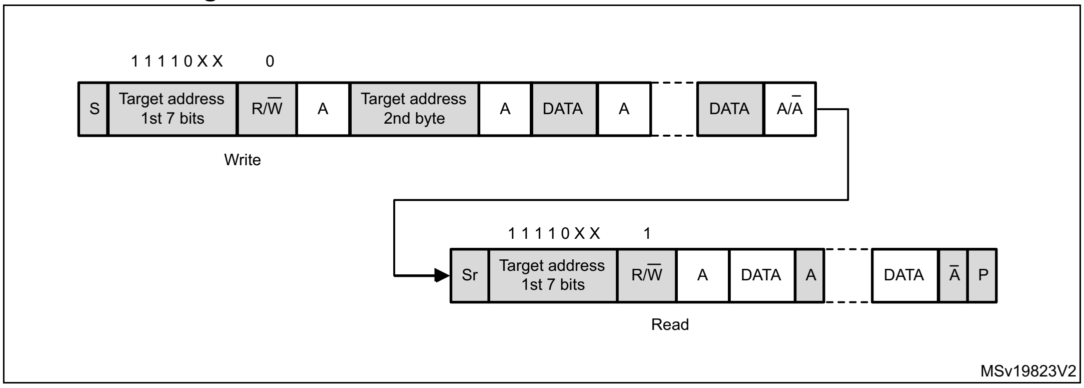
Figure 403. 10-bit address read access with HEAD10R = 0



If the controller addresses a 10-bit address target, transmits data to this target and then reads data from the same target, a controller transmission flow must be done first. Then a repeated START is set with the 10-bit target address configured with HEAD10R = 1. In this case, the controller sends this sequence:

RESTART + Target address 10-bit header Read.

Figure 404. 10-bit address read access with HEAD10R = 1



Controller 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 of the I2C_CR1 register is set. The flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

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 transmit is greater than 255, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and the number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a STOP condition is automatically sent.
- In software end mode (AUTOEND = 0), the TC flag is set and the SCL line is stretched low, to perform software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C_CR2 register with the proper target address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

When a NACK is received, the TXIS flag is not set and a STOP condition is automatically sent. The NACKF flag of the I2C_ISR register is set. An interrupt is generated if the NACKIE bit is set.

Figure 405. Transfer sequence flow for I2C controller transmitter, N ≤ 255 bytes

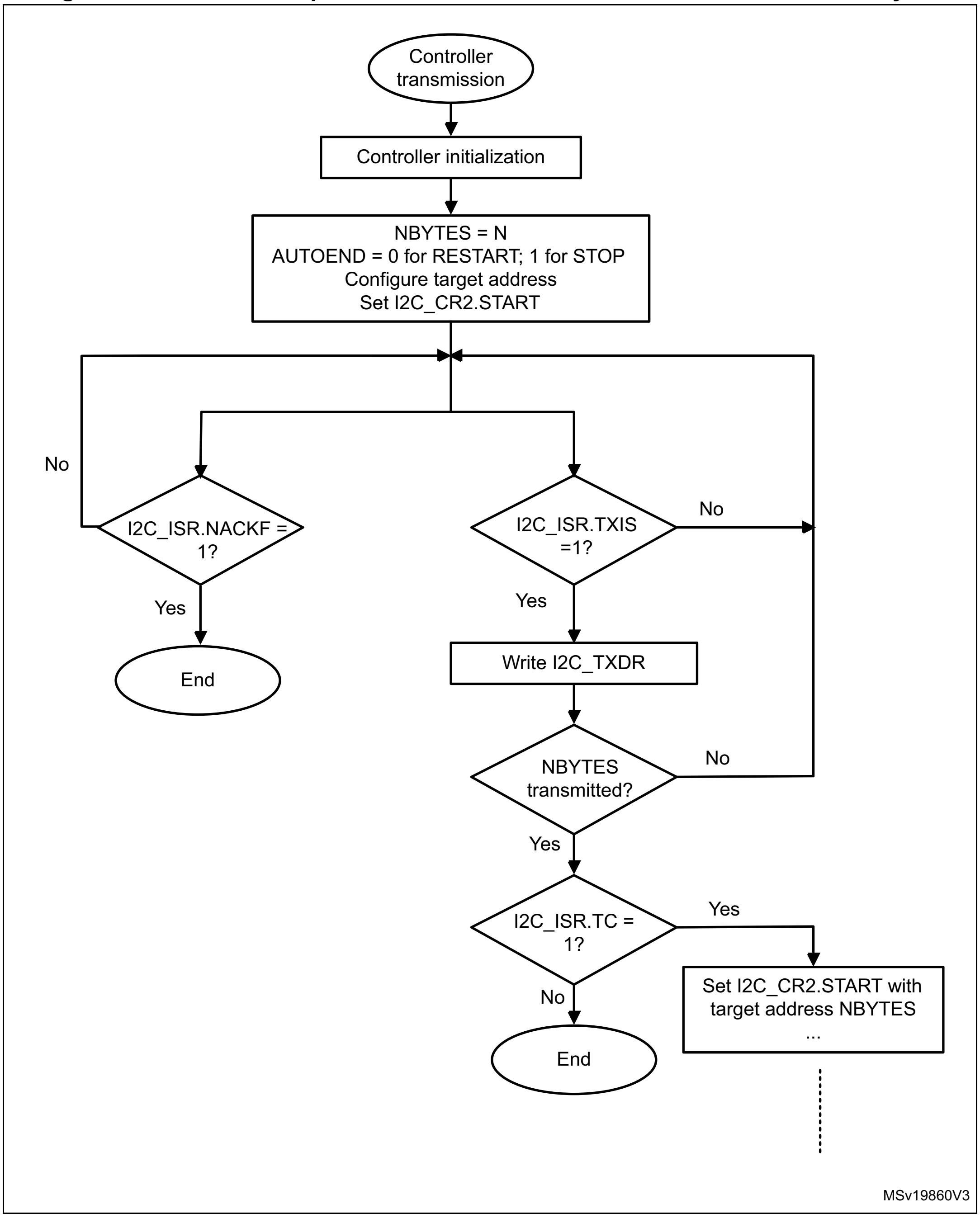


Figure 406. Transfer sequence flow for I2C controller transmitter, N > 255 bytes

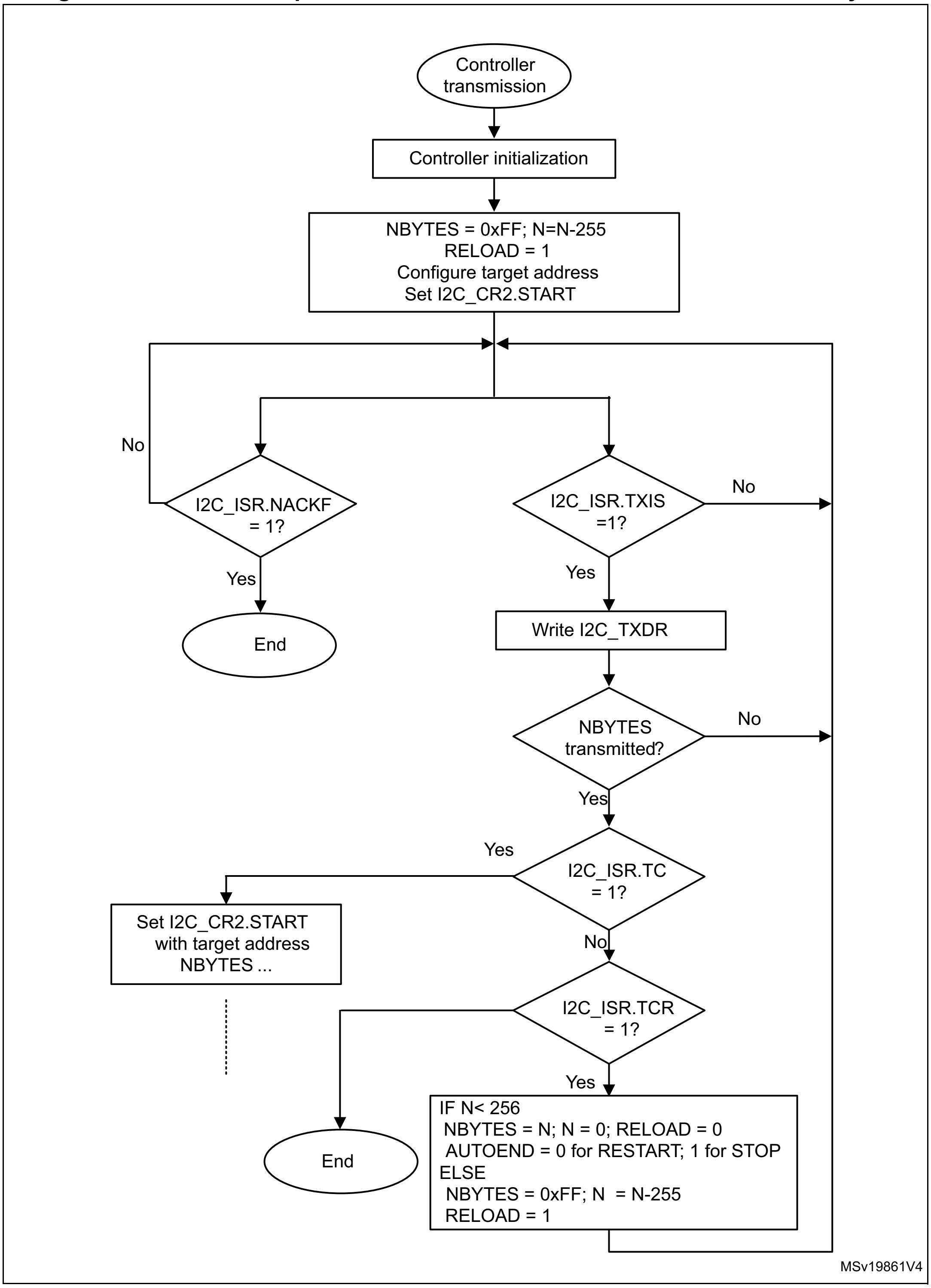
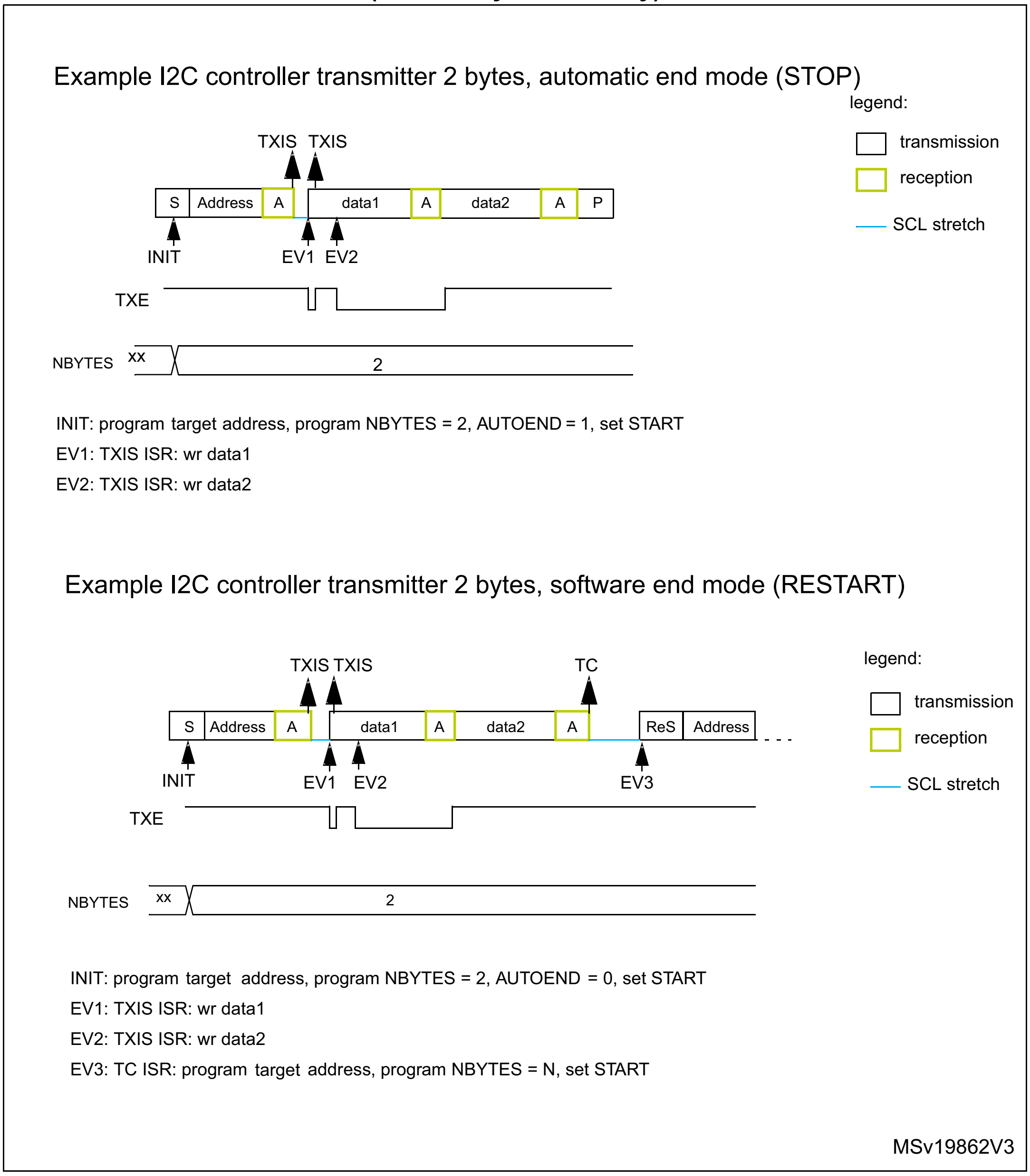


Figure 407. Transfer bus diagrams for I2C controller transmitter (mandatory events only)



Controller 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 of the I2C_CR1 register is set. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to receive is greater than 255, select the reload mode, by setting the RELOAD bit of the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and he number of data bytes defined in NBYTES[7:0] is 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 of the I2C_CR2 register, with the proper target address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition and the target address on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

Figure 408. Transfer sequence flow for I2C controller receiver, N ≤ 255 bytes

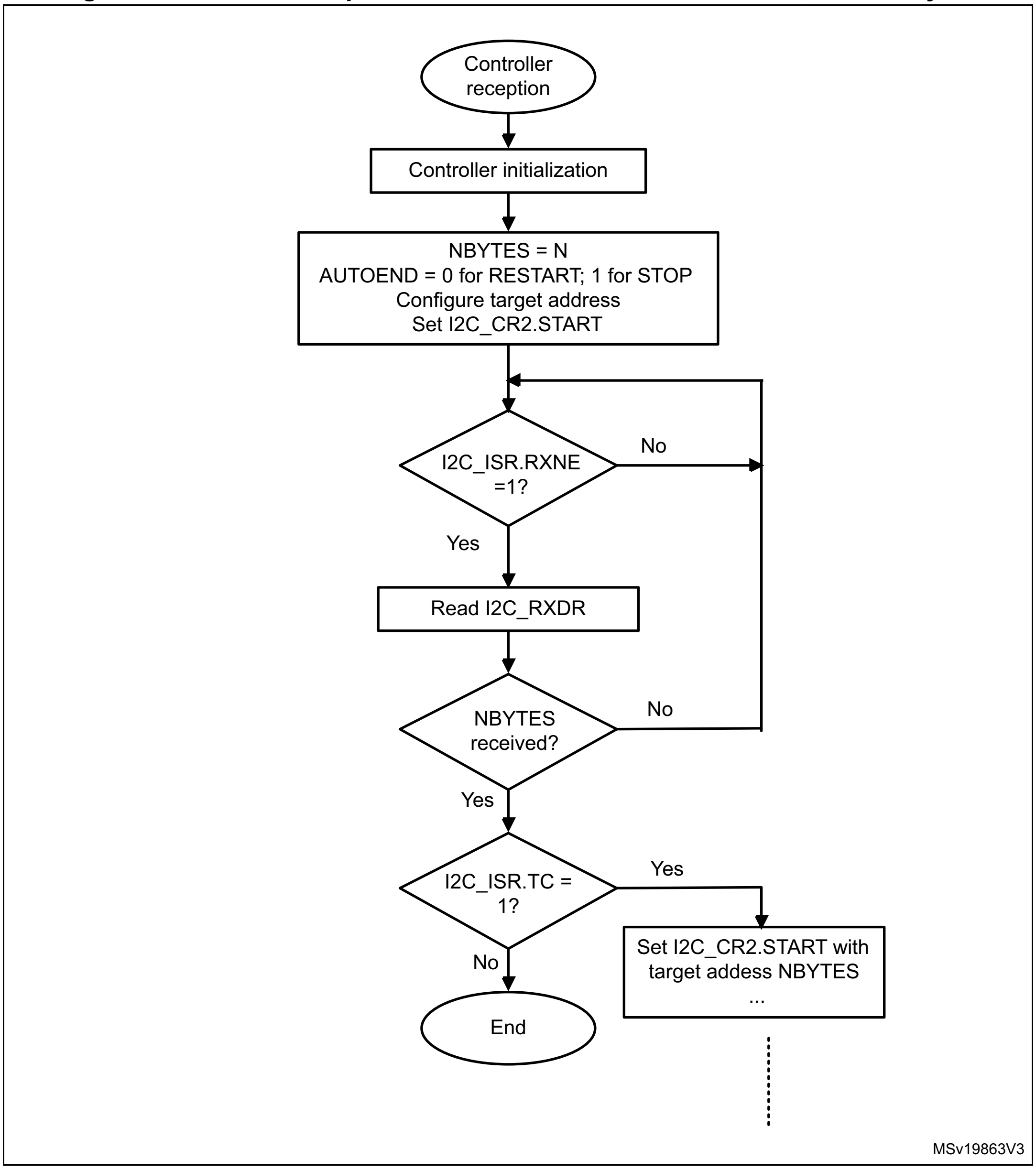


Figure 409. Transfer sequence flow for I2C controller receiver, N > 255 bytes

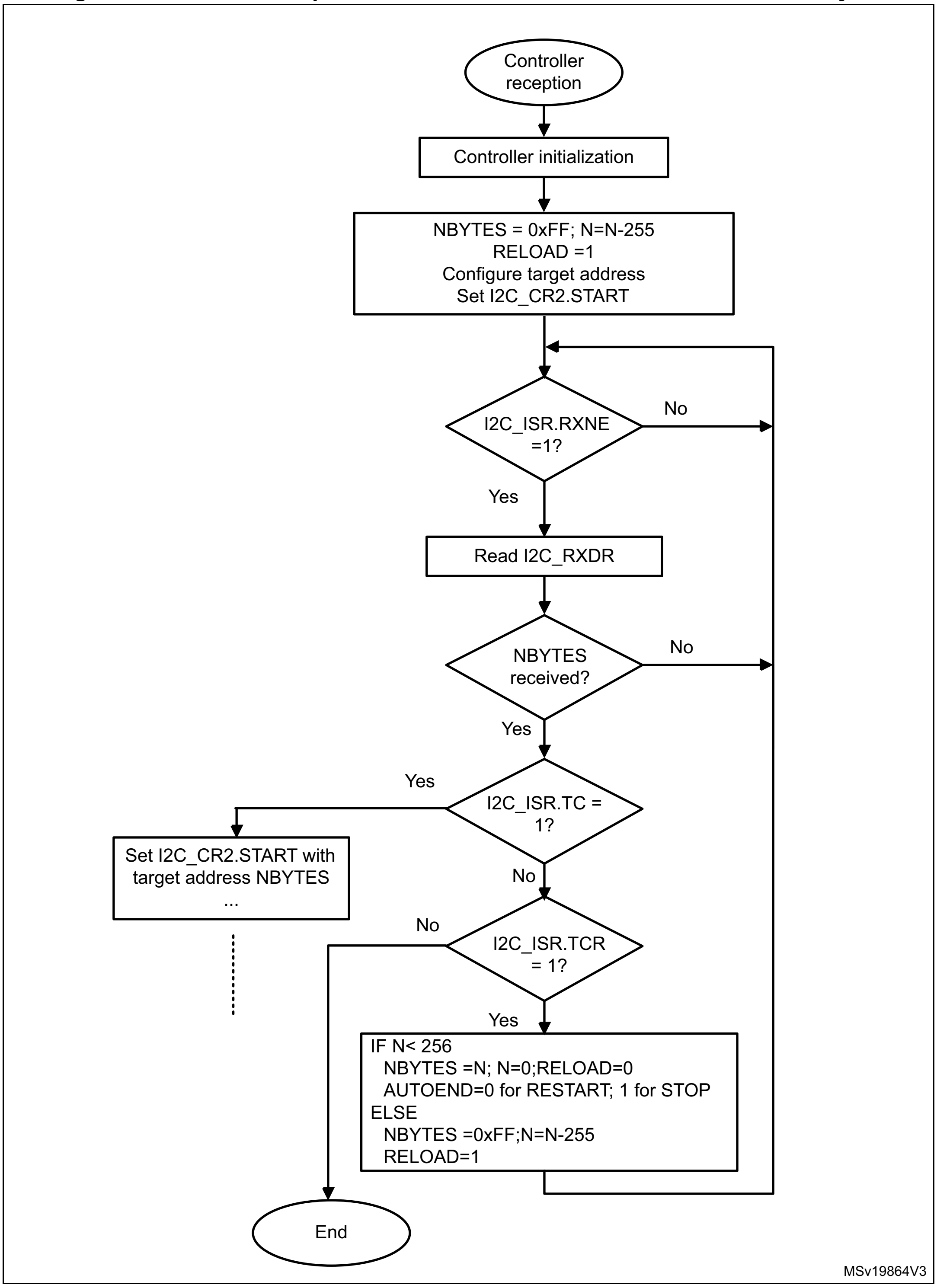
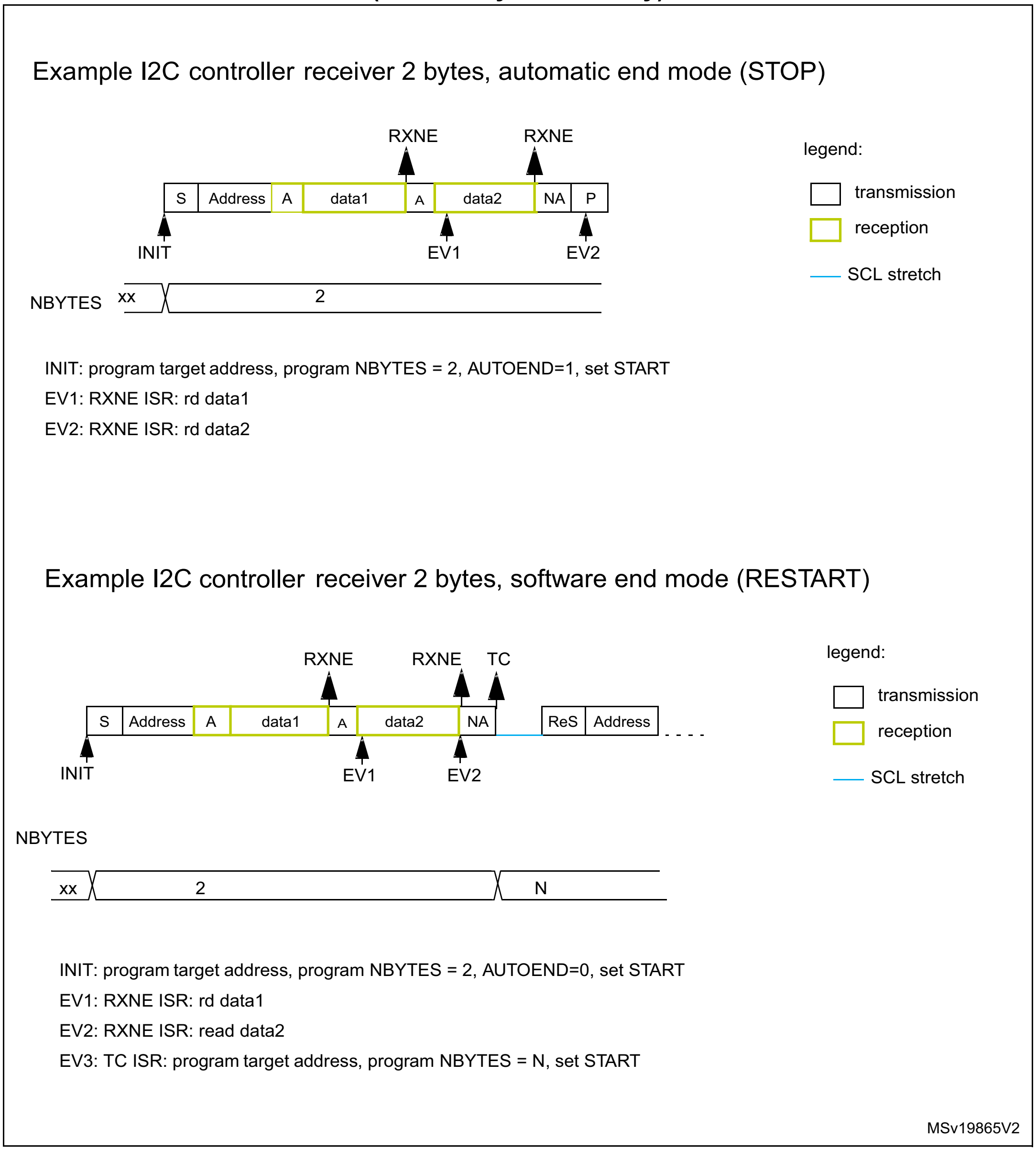


Figure 410. Transfer bus diagrams for I2C controller receiver (mandatory events only)



38.4.10 I2C_TIMINGR register configuration examples

The following tables provide examples of how to program the I2C_TIMINGR register to obtain timings compliant with the I²C-bus specification. To get more accurate configuration values, use the STM32CubeMX tool (*I2C Configuration* window).

Table 363. Timing settings for f_{I2CCLK} of 8 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	500 kHz
PRESC[3:0]	0x1	0x1	0x0	0x0
SCLL[7:0]	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[7:0]	0xC3	0xF	0x3	0x3
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	4 x 125 ns = 500 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2.5 µs(3)	~2.0 µs(4)
SDADEL[3:0]	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[3:0]	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. 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 364. Timing settings for f_{I2CCLK} of 16 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC[3:0]	0x3	0x3	0x1	0x0
SCLL[7:0]	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[7:0]	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)	~2.5 µs(3)	~1.0 µs(4)
SDADEL[3:0]	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[3:0]	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. 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.

38.4.11 SMBus specific features

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 operation principles of the I²C-bus. The SMBus provides a control bus for system and power management related tasks.

The I2C peripheral is compatible with the SMBus specification (http://smbus.org).

The system management bus specification refers to three types of devices:

- **Target** is a device that receives or responds to a command.
- **Controller** is a device that issues commands, generates clocks, and terminates the transfer.
- **Host** is a specialized controller that provides the main interface to the system CPU. A host must be a controller-target and must support the SMBus *host notify* protocol. Only one host is allowed in a system.

The I2C peripheral can be configured as a controller or a target device, and also as a host.

Bus protocols

There are eleven possible command protocols for any given device. The device can use any or all of them to communicate. These 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*. The protocols must be implemented by the user software.

For more details on these protocols, refer to the SMBus specification (http://smbus.org).

STM32CubeMX implements an SMBus stack thanks to X-CUBE-SMBUS, a downloadable software pack that allows basic SMBus configuration per I2C instance.

Address resolution protocol (ARP)

SMBus target address conflicts can be resolved by dynamically assigning a new unique address to each target device. To provide a mechanism to isolate each device for the purpose of address assignment, each device must implement a unique 128-bit device identifier (UDID). In the I2C peripheral, it is implemented by software.

The I2C peripheral supports the Address resolution protocol (ARP). The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register. The ARP commands must be implemented by the user software.

Arbitration is also performed in target mode for ARP support.

For more details on the SMBus address resolution protocol, refer to the SMBus specification (http://smbus.org).

Received command and data acknowledge control

An SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in target mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. Refer to *Target byte control mode* for more details.

Host notify protocol

To enable the host notify protocol, set the SMBHEN bit of the I2C_CR1 register. The I2C peripheral then acknowledges the SMBus host address (0b0001 000).

When this protocol is used, the device acts as a controller and the host as a target.

SMBus alert

The I2C peripheral supports the SMBALERT# optional signal through the SMBA pin. With the SMBALERT# signal, an SMBus target device can signal to the SMBus host 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/devices which pulled SMBALERT# low acknowledges/acknowledge the alert response address.

When the I2C peripheral is configured as an SMBus target device (SMBHEN = 0), the SMBA pin is pulled low by setting the ALERTEN bit of the I2C_CR1 register. The alert response address is enabled at the same time.

When the I2C peripheral is configured as an SMBus host (SMBHEN = 1), the ALERT flag of the I2C_ISR register is set when a falling edge is detected on the SMBA pin and ALERTEN = 1. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set. When ALERTEN = 0, the alert line is considered high even if the external SMBA pin is low.

Note: If the SMBus alert pin is not required, keep the ALERTEN bit cleared. The SMBA pin can then be used as a standard GPIO.

Packet error checking

A packet error checking mechanism introduced in the SMBus specification improves 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 I2C peripheral embeds a hardware PEC calculator and allows a not acknowledge to be sent automatically when the received byte does not match the hardware calculated PEC.

Timeouts

To comply with the SMBus timeout specifications, the I2C peripheral embeds hardware timers.

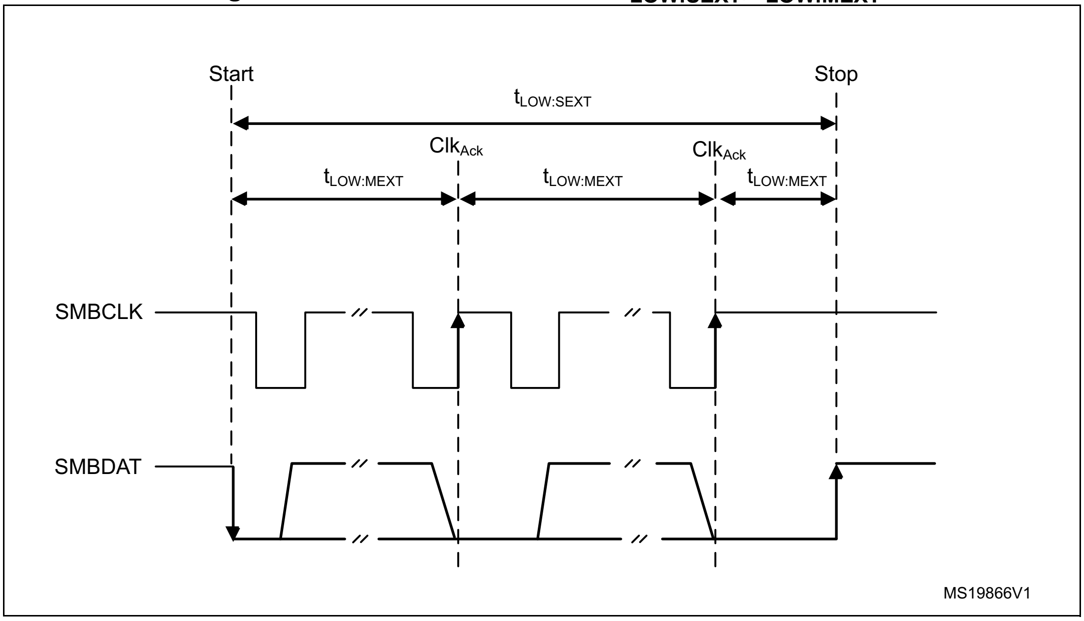
Table 365. SMBus timeout specifications

Symbol	Parameter	Limits		Unit
		Min	Max	
$t_{TIMEOUT}$	Detect clock low timeout	25	35	ms
$t_{LOW:SEXT}$(1)	Cumulative clock low extend time (target device)	-	25	
$t_{LOW:MEXT}$(2)	Cumulative clock low extend time (controller device)	-	10	

1. $t_{LOW:SEXT}$ is the cumulative time a given target device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that another target device or the controller also extends the clock causing the combined clock low extend time to be greater than $t_{LOW:SEXT}$. The value provided applies to a single target device connected to a full-target controller.

2. $t_{LOW:MEXT}$ is the cumulative time a controller 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 target device or another controller also extends the clock, causing the combined clock low time to be greater than $t_{LOW:MEXT}$ on a given byte. The value provided applies to a single target device connected to a full-target controller.

Figure 411. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$



Bus idle detection

A controller can assume that the bus is free if it detects that the clock and data signals have been high for $t_{IDLE} > t_{HIGH}$(max) (refer to the table in *Section 38.4.9*).

This timing parameter covers the condition where a controller is dynamically added to the bus, and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the controller must wait long enough to ensure that a transfer is not currently in progress. The I2C peripheral supports a hardware bus idle detection.

38.4.12 SMBus initialization

In addition to the I2C initialization for the I²C-bus, the use of the peripheral for the SMBus communication requires some extra initialization steps.

Received command and data acknowledge control (target mode)

An SMBus receiver must be able to NACK each received command or data. To allow ACK control in target mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. Refer to *Target byte control mode* for more details.

Specific addresses (target mode)

The specific SMBus addresses must be enabled if required. Refer to *Bus idle detection* for more details.

The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register.

The SMBus host address (0b0001 000) is enabled by setting the SMBHEN bit of the I2C_CR1 register.

The alert response address (0b0001100) is enabled by setting the ALERTEN bit of the I2C_CR1 register.

Packet error checking

PEC calculation is enabled by setting the PECEN bit of the I2C_CR1 register. Then the PEC transfer is managed with the help of the hardware byte counter associated with the NBYTES[7:0] bitfield of 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 target mode. The PEC is transferred after transferring NBYTES[7:0] - 1 data bytes, if 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 peripheral is enabled.

Table 366. SMBus with PEC configuration

Mode	SBC bit	RELOAD bit	AUTOEND bit	PECBYTE bit
Controller Tx/Rx NBYTES + PEC+ STOP	X	0	1	1
Controller Tx/Rx NBYTES + PEC + ReSTART	X	0	0	1
Target Tx/Rx with PEC	1	0	X	1

Timeout detection

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits of 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

To check the $t_{TIMEOUT}$ parameter, load the 12-bit TIMEOUTA[11:0] bitfield with the timer reload value. Keep the TIDLE bit at 0 to detect the SCL low level timeout.

Then set the TIMOUTEN bit of the I2C_TIMEOUTR register, to enable the timer.

If SCL is tied low for longer than the (TIMEOUTA + 1) x 2048 x t_{I2CCLK} period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 367*.

Caution: Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

$t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ check

A 12-bit timer associated with the TIMEOUTB[11:0] bitfield allows checking $t_{LOW:SEXT}$ for the I2C peripheral operating as a target, or $t_{LOW:MEXT}$ when it operates as a controller. As the standard only specifies a maximum, the user can choose the same value for both. The timer is then enabled by setting the TEXTEN bit in the I2C_TIMEOUTR register.

If the SMBus peripheral performs a cumulative SCL stretch for longer than the (TIMEOUTB + 1) x 2048 x t_{I2CCLK} period, and within the timeout interval described in *Bus idle detection* section, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 368*.

Caution: Changing the TIMEOUTB[11:0] bitfield value is not allowed when the TEXTEN bit is set.

Bus idle detection

To check the t_{IDLE} period, the TIMEOUTA[11:0] bitfield associated with 12-bit timer must be loaded with the timer reload value. Keep the TIDLE bit at 1 to detect both SCL and SDA high level timeout. Then set the TIMOUTEN bit of the I2C_TIMEOUTR register to enable the timer.

If both the SCL and SDA lines remain high for longer than the (TIMEOUTA + 1) x 4 x t_{I2CCLK} period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 369*.

Caution: Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

38.4.13 SMBus I2C_TIMEOUTR register configuration examples

The following tables provide examples of settings to reach desired $t_{TIMEOUT}$, $t_{LOW:SEXT}$, $t_{LOW:MEXT}$, and t_{IDLE} timings at different f_{I2CCLK} frequencies.

Table 367. TIMEOUTA[11:0] for maximum $t_{TIMEOUT}$ of 25 ms

f_{I2CCLK}	TIMEOUTA[11:0]	TIDLE	TIMEOUTEN	$t_{TIMEOUT}$
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

Table 368. TIMEOUTB[11:0] for maximum $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ of 8 ms

f_{I2CCLK}	TIMEOUTB[11:0]	TEXTEN	$t_{LOW:SEXT}$ $t_{LOW:MEXT}$
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

Table 369. TIMEOUTA[11:0] for maximum t_{IDLE} of 50 µs

f_{I2CCLK}	TIMEOUTA[11:0]	TIDLE	TIMEOUTEN	t_{IDLE}
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

38.4.14 SMBus target mode

In addition to I2C target transfer management (refer to *Section 38.4.8: I2C target mode*), this section provides extra software flowcharts to support SMBus.

SMBus target transmitter

When using the I2C peripheral in SMBus mode, set the SBC bit to enable 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[7:0] - 1, and the content of the I2C_PECR register is automatically transmitted if the controller requests an extra byte after the transfer of the NBYTES[7:0] - 1 data bytes.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 412. Transfer sequence flow for SMBus target transmitter N bytes + PEC

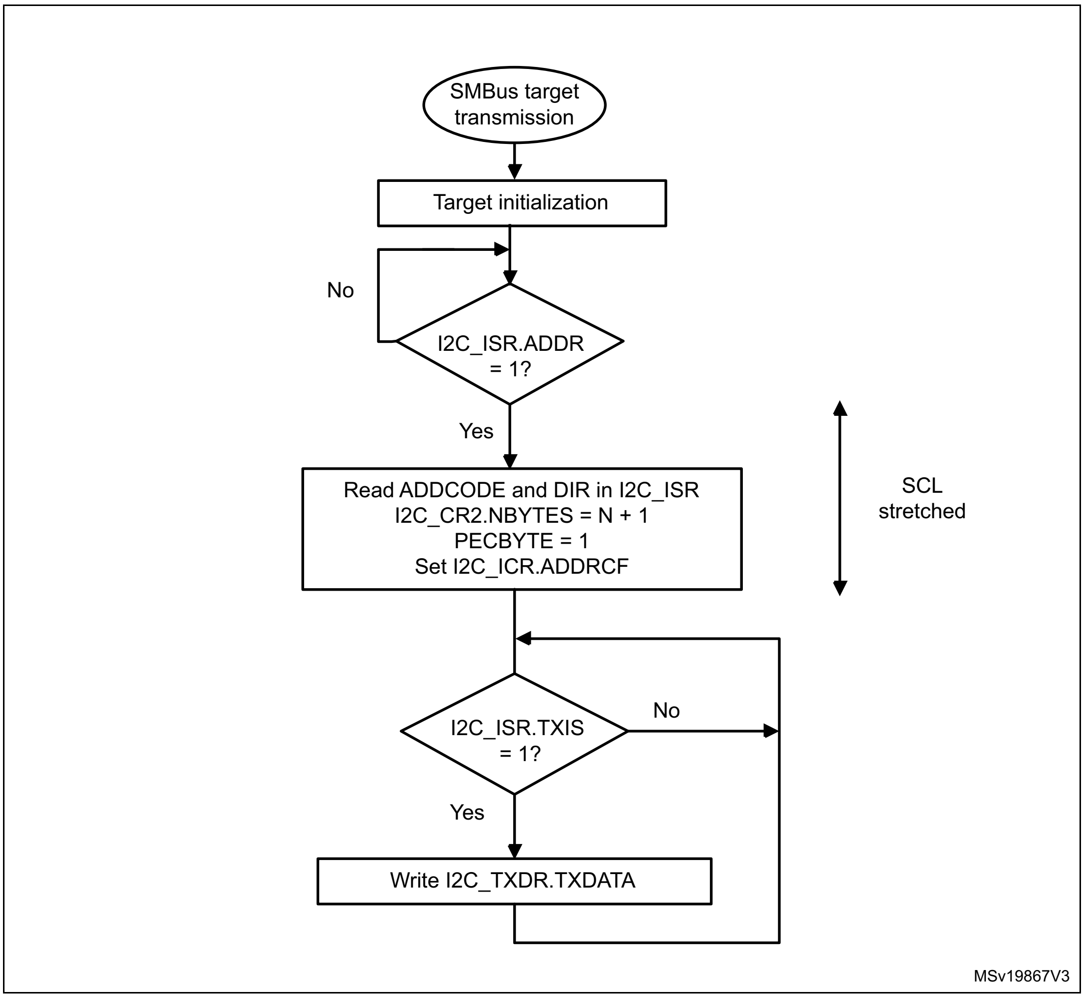
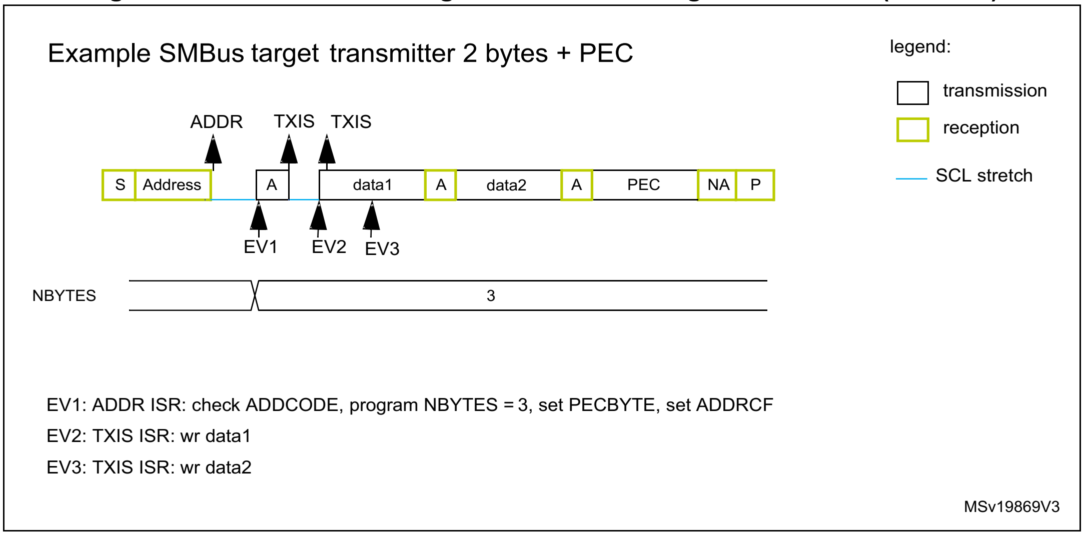


Figure 413. Transfer bus diagram for SMBus target transmitter (SBC = 1)



SMBus target receiver

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC checking at the end of the programmed number of data bytes. To allow the ACK control of each byte, the reload mode must be selected (RELOAD = 1). Refer to *Target byte control mode* for more details.

To check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, 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.

Upon a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

If no ACK software control is required, the user can set the PECBYTE bit and, in the same write operation, load NBYTES[7:0] with the number of bytes to receive in a continuous flow. After the receipt of NBYTES[7:0] - 1 bytes, the next received byte is checked as being the PEC.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 414. Transfer sequence flow for SMBus target receiver N bytes + PEC

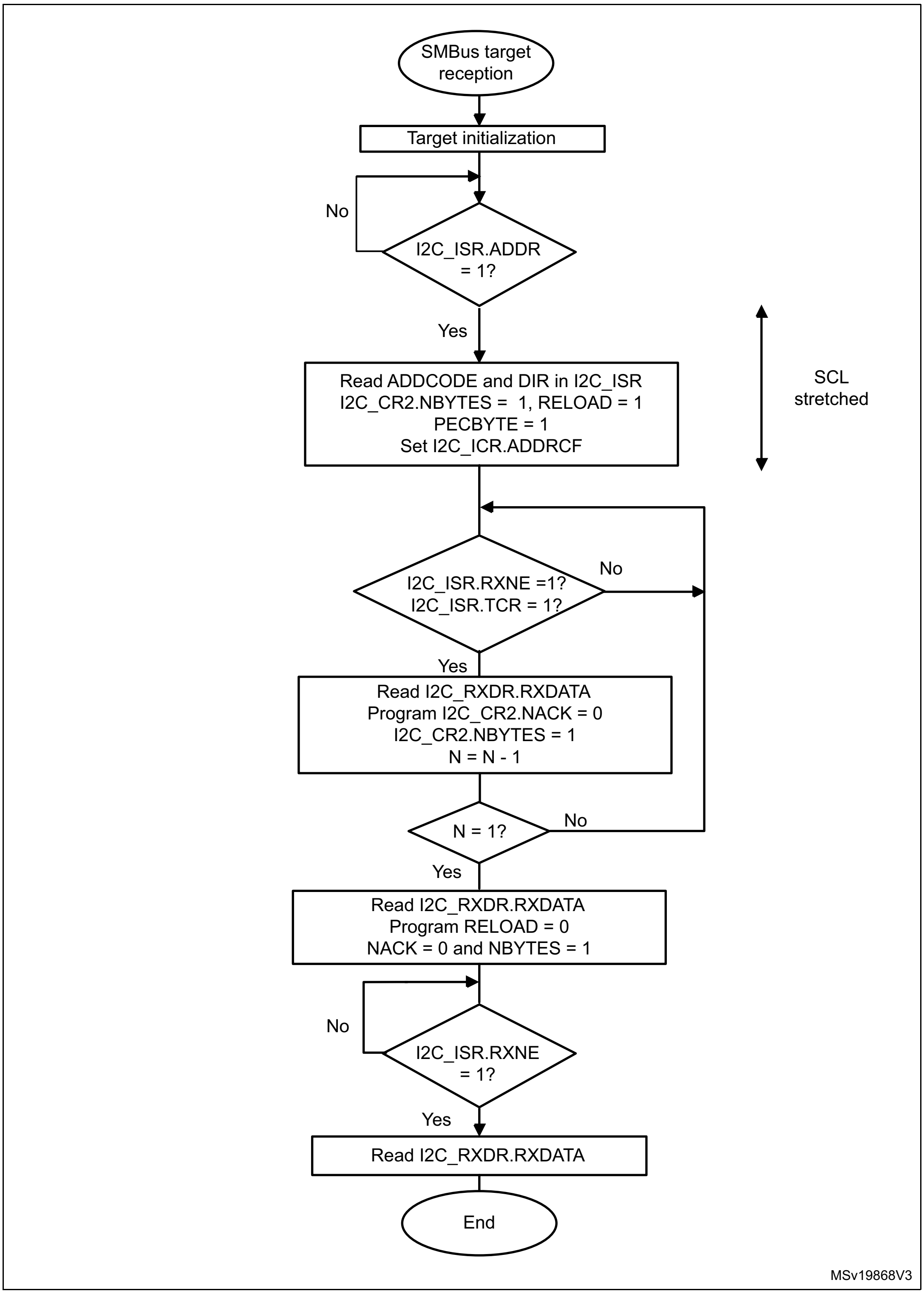
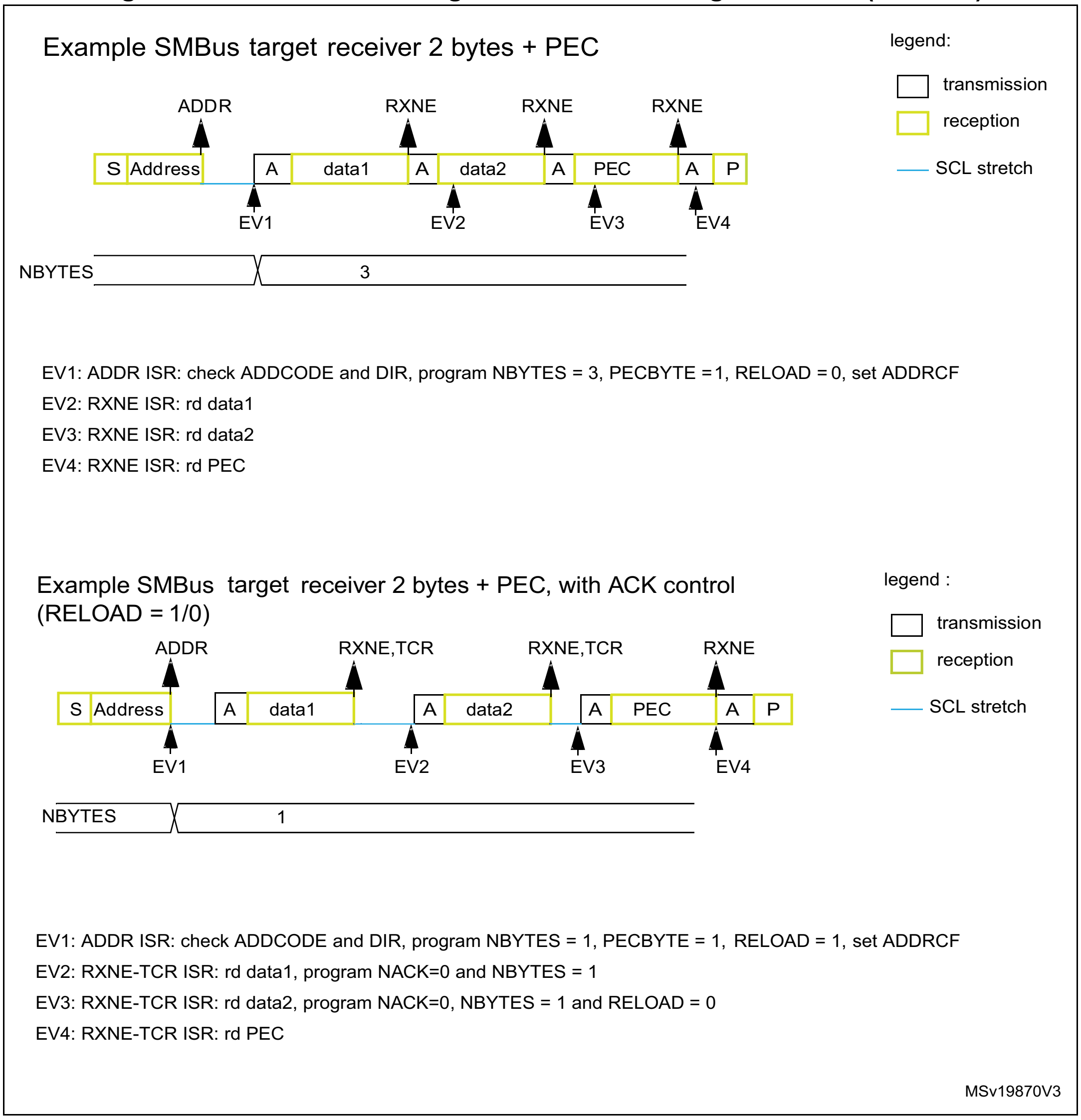


Figure 415. Bus transfer diagrams for SMBus target receiver (SBC = 1)



38.4.15 SMBus controller mode

In addition to I2C controller transfer management (refer to *Section 38.4.9: I2C controller mode*), this section provides extra software flowcharts to support SMBus.

SMBus controller transmitter

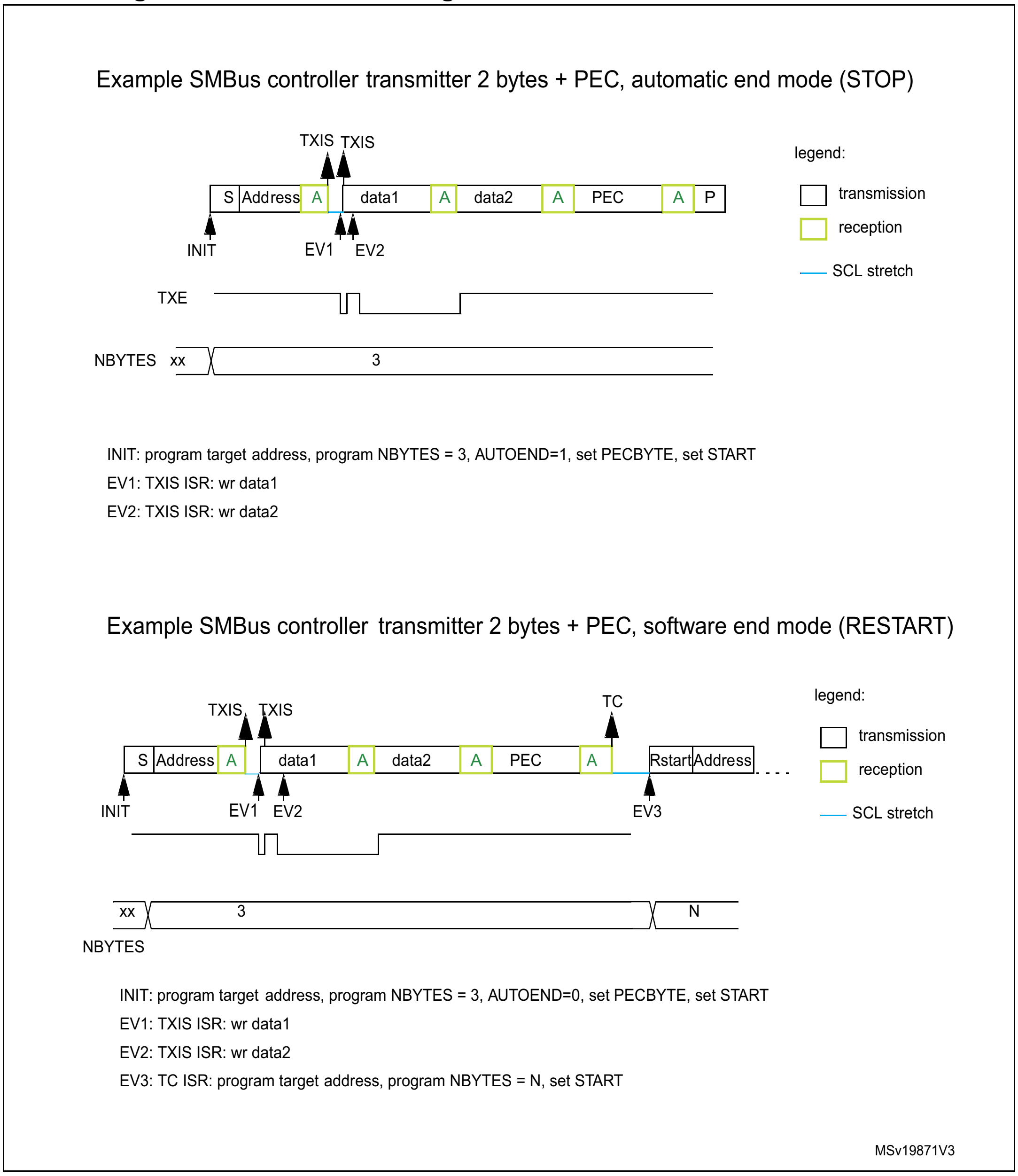
When the SMBus controller wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be loaded in the NBYTES[7:0] bitfield, before setting the START bit. In this case, the total number of TXIS interrupts is NBYTES[7:0] - 1. So if the PECBYTE bit is set when NBYTES[7:0] = 0x1, the content of the I2C_PECR register is automatically transmitted.

If the SMBus controller wants to send a STOP condition after the PEC, the automatic end mode must be selected (AUTOEND = 1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus controller wants to send a RESTART condition after the PEC, the software mode must be selected (AUTOEND = 0). In this case, once NBYTES[7:0] - 1 are transmitted, the I2C_PECR register content is transmitted. 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 416. Bus transfer diagrams for SMBus controller transmitter



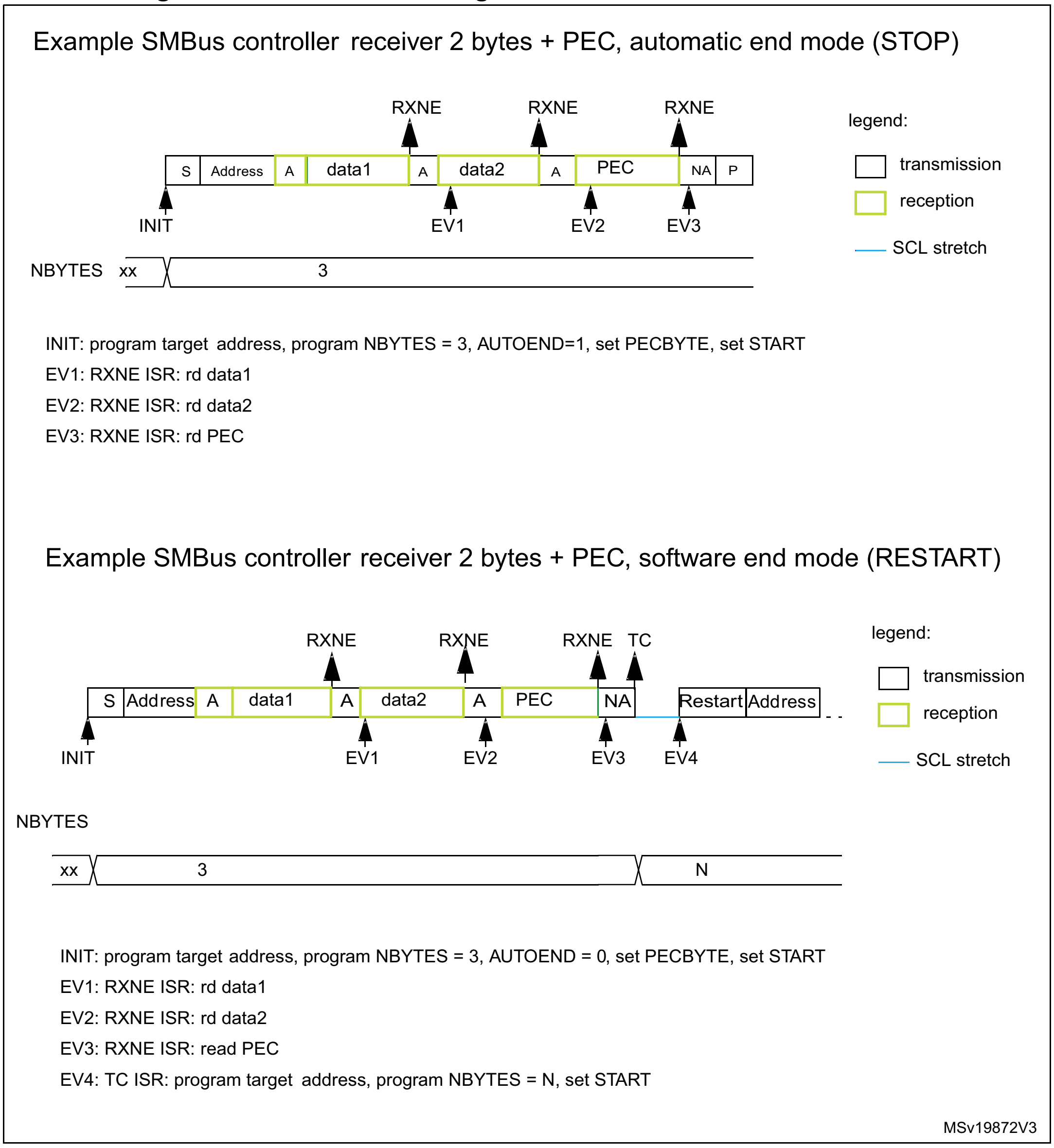
SMBus controller receiver

When the SMBus controller wants to receive, at the end of the transfer, the PEC followed by a STOP condition, the automatic end mode can be selected (AUTOEND = 1). The PECBYTE bit must be set and the target address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, 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 controller receiver wants to receive, at the end of the transfer, the PEC byte followed by a RESTART condition, the software mode must be selected (AUTOEND = 0). The PECBYTE bit must be set and the target address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, 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 417. Bus transfer diagrams for SMBus controller receiver



38.4.16 Autonomous mode

The I2C peripheral can autonomously operate as controller or target in Stop mode as long as NOSTRETCH = 0. The autonomous mode is not supported when NOSTRETCH = 1. Besides Stop mode, the autonomous mode can also be used in Run and Sleep modes.

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.

If an interrupt is generated, the device wakes up from Stop mode.

If there is no interrupt, the device remains in Stop mode, but the kernel and AHB/APB clocks are available for the I2C peripheral and all the autonomous peripherals enabled in the RCC.

If the DMA requests are enabled, the data are DMA-transferred to or from the SRAM, while the device remains in Stop mode.

Target mode

In target 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 target is addressed. If the target is not addressed, the kernel clock request is released after the address phase.

To optimize the functionality in target 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 target mode.

Controller mode

In controller 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.

To avoid waking up the CPU too often, it is possible to replace some interrupts by DMA requests, to make some control register write actions. In controller mode, *transfer complete* (TC) and *transfer complete reload* (TCR) events can generate a DMA request when the corresponding TCDMAEN or TCRDMAEN bits of the I2C_AUTOCR register are set. Consequently, the I2C_CR2 register can be written thanks to a DMA transfer, 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 the control back 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 register
2. Clear the TRIGEN bit of the I2C_AUTOCR register
3. Wait for longer than t_{BUF} (bus free time between a STOP and a START condition)
4. Wait for BUSY = 0 in the I2C_ISR register

Caution: When the device is in Stop mode, the I2C peripheral receives its kernel clock only when it is implicated in the transfer. Consequently, features such as timeouts and bus idle detection are not reliable in Stop mode.

Caution: The digital filter is not compatible with the functionality in Stop mode. Before entering Stop mode with the WUPEN bit set, deactivate the digital filter, by writing zero to the DNF[3:0] bitfield.

Caution: Clock stretching must be enabled (NOSTRETCH = 0) to ensure proper operation of the wake-up from Stop mode feature.

Caution: If wake-up from Stop mode is disabled (WUPEN = 0), the I2C peripheral must be disabled before entering Stop mode (PE = 0).

38.4.17 Error conditions

The following errors are the conditions that can cause the 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 nine SCL clock pulses. START or STOP condition is detected when an SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C peripheral is involved in the transfer as controller or addressed target (that is, not during the address phase in target mode).

In case of a misplaced START or RESTART detection in target mode, the I2C peripheral enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag of the I2C_ISR register is set, and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Arbitration loss (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 controller 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 controller switches automatically to target mode.

In target 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 of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Overrun/underrun error (OVR)

An overrun or underrun error is detected in target 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 must 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 of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Packet error checking error (PECERR)

A PEC error is detected when the received PEC byte does not match the I2C_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Timeout error (TIMEOUT)

A timeout error occurs for any of these conditions:

- TIDLE = 0 and SCL remains low for the time defined in the TIMEOUTA[11:0] bitfield: this is used to detect an SMBus timeout.
- TIDLE = 1 and both SDA and SCL remains high for the time defined in the TIMEOUTA [11:0] bitfield: this is used to detect a bus idle condition.
- Controller cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus $t_{LOW:MEXT}$ parameter).
- Target cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus $t_{LOW:SEXT}$ parameter).

When a timeout violation is detected in controller mode, a STOP condition is automatically sent.

When a timeout violation is detected in target mode, the 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 of the I2C_CR1 register is set.

Alert (ALERT)

The ALERT flag is set when the I2C peripheral is configured as a host (SMBHEN = 1), the SMBALERT# signal detection is enabled (ALERTEN = 1), and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

38.5 I2C in low-power modes

Table 370. Effect of low-power modes to I2C

Mode	Description
Sleep	No effect. I2C interrupts cause the device to exit the Sleep mode.
Stop	The contents of I2C registers are kept. If the autonomous mode is enabled and I2C is clocked by an internal oscillator available in Stop mode: transfers in controller and in target 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. It must be reinitialized after exiting Standby mode.

38.6 I2C interrupts

The following table gives the list of I2C interrupt requests.

Table 371. I2C interrupt requests

Interrupt acronym		Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit the Sleep mode	Exit the Stop mode	Exit the Standby 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_ERR	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 the I2C implementation table for information about wake-up from Stop mode support per instance.

38.7 I2C DMA requests

38.7.1 Transmission using DMA

DMA (direct memory access) can be enabled for transmission by setting the TXDMAEN bit of the I2C_CR1 register. Data is loaded from an SRAM area configured through the DMA peripheral (see *Section 17: General purpose direct memory access controller (GPDMA)*) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software (the transmitted target address cannot be transferred with DMA). When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to *Controller transmitter*.

In target mode:

- With NOSTRETCH = 0, when all data are transferred using DMA, 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.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to *SMBus target transmitter* and *SMBus controller transmitter*.

Note: If DMA is used for transmission, it is not required to set the TXIE bit.

38.7.2 Reception using DMA

DMA (direct memory access) can be enabled for reception by setting the RXDMAEN bit of the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured through the DMA peripheral (refer to *Section 17: General purpose direct memory access controller (GPDMA)*) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.

In target mode with NOSTRETCH = 0, when all data are transferred using DMA, DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to *SMBus target receiver* and *SMBus controller receiver*.

Note: If DMA is used for reception, it is not required to set the RXIE bit.

38.7.3 Controller event control using DMA

In controller mode, the transfer can be automatically managed while the device is in Run, Sleep or Stop mode, thanks to TC and TCR DMA requests.

If the TCDMAEN bit of the I2C_AUTOCR register is set, the I2C_EVC (I2C control event) DMA request is generated when the TC bit of the I2C_ISR register is set. The DMA controller must be programmed to write the next command in the I2C_CR2 register. If both STOP and START bits 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 the TCRDMAEN bit of the I2C_AUTOCR register is set, the I2C_EVC (I2C control event) DMA request is generated when TCR bit of the I2C_ISR register is set. The DMA controller must be programmed to write the remaining number of bytes to transfer in the I2C_CR2 register.

38.8 I2C debug modes

When the device enters debug mode (core halted), the SMBus timeout either continues working normally or stops, depending on the DBG_I2Cx_STOP bits in the DBG block.

38.9 I2C registers

Refer to *Section 1.2* for the list of abbreviations used in register descriptions.

The registers are accessed by words (32-bit).

38.9.1 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 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, cleared by software by setting STOPCF bit.
- 1: STOPF flag remains cleared by hardware. This mode can be used in NOSTRETCH target mode, to avoid the overrun error if the STOPF flag is not cleared before next data transmission. This allows a target 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, cleared by software by setting ADDRCF bit.
- 1: ADDR flag remains cleared by hardware. This mode can be used in target mode, to avoid the ADDR clock stretching if the I2C enables only one target address. This allows a target 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

Bit 22 **ALERTEN**: SMBus alert enable

0: The SMBALERT# signal on SMBA pin 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 SMBALERT# signal on SMBA 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.

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.

Bit 20 **SMBHEN**: SMBus host address enable

0: Host address disabled. Address 0b0001000x is NACKed.
1: Host address enabled. Address 0b0001000x is ACKed.

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**: Wake-up from Stop mode enable

0: Wake-up from Stop mode disabled.
1: Wake-up from Stop mode enabled.

Note: WUPEN can be set only when DNF[3:0] = 0000.

Bit 17 **NOSTRETCH**: Clock stretching disable

This bit is used to disable clock stretching in target mode. It must be kept cleared in controller mode.

0: Clock stretching enabled
1: Clock stretching disabled

Note: This bit can be programmed only when the I2C peripheral is disabled (PE = 0).

Bit 16 **SBC**: Target byte control

This bit is used to enable hardware byte control in target mode.

0: Target byte control disabled
1: Target 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 be programmed only when the I2C peripheral 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 one t_{I2CCLK}

...

1111: digital filter enabled and filtering capability up to fifteen t_{I2CCLK}

Note: If the analog filter is enabled, the digital filter is added to it. This filter can be programmed only when the I2C peripheral 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 generates 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 generates 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 (target 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 disabled

1: Peripheral enabled

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 three APB clock cycles.

38.9.2 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 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.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]							
					rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **PECBYTE**: Packet error checking byte

This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when PE = 0.

0: No PEC transfer

1: PEC transmission/reception is requested

Note: Writing 0 to this bit has no effect.

This bit has no effect when RELOAD is set, and in target mode when SBC = 0.

Bit 25 **AUTOEND**: Automatic end mode (controller 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 target 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 target mode with SBC = 0.

Note: Changing these bits when the START bit is set is not allowed.

Bit 15 **NACK**: NACK generation (target 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 only in target mode: in controller receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.

When an overrun occurs in target 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 condition generation

This bit only pertains to controller mode. It is set by software and cleared by hardware when a STOP condition is detected or when PE = 0.

0: No STOP generation

1: STOP generation after current byte transfer

Note: Writing 0 to this bit has no effect.

Bit 13 **START**: START condition generation

This bit is set by software. It is cleared by hardware after the START condition followed by the address sequence is sent, by an arbitration loss, by an address matched in target mode, by a timeout error detection, or when PE = 0.

0: No START generation

1: RESTART/START generation:

If the I2C is already in controller 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 target mode.

This bit has no effect when RELOAD is set.

Bit 12 **HEAD10R**: 10-bit address header only read direction (controller receiver mode)

0: The controller sends the complete 10-bit target address read sequence: START + 2 bytes 10-bit address in write direction + RESTART + first seven bits of the 10-bit address in read direction.

1: The controller sends only the first seven 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 (controller mode)

0: The controller operates in 7-bit addressing mode

1: The controller 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 (controller mode)

0: Controller requests a write transfer

1: Controller requests a read transfer

Note: Changing this bit when the START bit is set is not allowed.

Bits 9:0 **SADD[9:0]**: Target address (controller mode)

Condition: In 7-bit addressing mode (ADD10 = 0):

SADD[7:1] must be written with the 7-bit target address to be sent. Bits SADD[9], SADD[8] and SADD[0] are don't care.

Condition: In 10-bit addressing mode (ADD10 = 1):

SADD[9:0] must be written with the 10-bit target address to be sent.

Note: Changing these bits when the START bit is set is not allowed.

38.9.3 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 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.	OA1MODE	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 target address OA1 is NACKed.

1: Own address 1 enabled. The received target 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 target address

7-bit addressing mode: OA1[7:1] contains the 7-bit own target address. Bits OA1[9], OA1[8] and OA1[0] are don't care.

10-bit addressing mode: OA1[9:0] contains the 10-bit own target address.

Note: These bits can be written only when OA1EN = 0.

38.9.4 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 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 2x 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 target address OA2 is NACKed.
1: Own address 2 enabled. The received target 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 ≠ 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.

38.9.5 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 to generate the clock period t_{PRESC} used for data setup and hold counters (refer to section *I2C timings*), and for SCL high and low level counters (refer to section *I2C controller initialization*).

$t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **SCLDEL[3:0]**: Data setup time

This field is used to generate a delay $t_{SCLDEL} = (SCLDEL + 1) \times t_{PRESC}$ between SDA edge and SCL rising edge. In controller and in target modes with NOSTRETCH = 0, the SCL line is stretched low during t_{SCLDEL}.

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 controller and in target modes with NOSTRETCH = 0, the SCL line is stretched low during t_{SDADEL}.

$t_{SDADEL} = SDADEL \times t_{PRESC}$

Note: SDADEL is used to generate $t_{HD:DAT}$ timing.

Bits 15:8 **SCLH[7:0]**: SCL high period (controller mode)

This field is used to generate the SCL high period in controller mode.

$t_{SCLH} = (SCLH + 1) \times t_{PRESC}$

Note: SCLH is also used to generate $t_{SU:STO}$ and $t_{HD:STA}$ timing.

Bits 7:0 **SCLL[7:0]**: SCL low period (controller mode)

This field is used to generate the SCL low period in controller mode.

$t_{SCLL} = (SCLL + 1) \times t_{PRESC}$

Note: SCLL is also used to generate t_{BUF} and $t_{SU:STA}$ timings.

Note: This register must be configured when the I2C peripheral is disabled (PE = 0).

Note: The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.

38.9.6 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 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:

- Controller mode: the controller cumulative clock low extend time ($t_{LOW:MEXT}$) is detected
- Target mode: the target cumulative clock low extend time ($t_{LOW:SEXT}$) is detected

$t_{LOW:EXT}$ = (TIMEOUTB + TIDLE = 01) 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.

38.9.7 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	TIMEOUT	PECERR	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 (target 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 two MSBs of the address.

Bit 16 **DIR**: Transfer direction (target mode)

This flag is updated when an address match event occurs (ADDR = 1).

0: Write transfer, target enters receiver mode.

1: Read transfer, target 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, and 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 an 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.

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.

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.

Bit 10 **OVR**: Overrun/underrun (target mode)

This flag is set by hardware in target 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 target mode. It is cleared by software by setting the 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 controller mode, or for target mode when the SBC bit is set.

Bit 6 **TC**: Transfer complete (controller 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:

- as a controller, provided that the STOP condition is generated by the peripheral.
- as a target, 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 (target mode)

This bit is set by hardware as soon as the received target address matched with one of the enabled target 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 only when NOSTRETCH = 1, 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.

38.9.8 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.

Bit 12 **TIMOUTCF**: Timeout detection flag clear

Writing 1 to this bit clears the TIMEOUT flag in the I2C_ISR register.

Bit 11 **PECCF**: PEC error flag clear

Writing 1 to this bit clears the PECERR flag in the I2C_ISR register.

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.

38.9.9 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.

38.9.10 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 I²C-bus.

38.9.11 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 I²C-bus

Note: These bits can be written only when TXE = 1.

38.9.12 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 38.4.2: 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.

38.9.13 I2C register map

The table below provides the I2C register map and the reset values.

Table 372. 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
0x00	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
0x04	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
0x08	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
0x0C	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
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

Table 372. 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
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* for the register boundary addresses.

39 Universal synchronous/asynchronous receiver transmitter (USART/UART)

39.1 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.

39.2 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: wake-up from mute mode by idle line detection or address mark detection
- Wake-up from Stop mode
- Autonomous functionality in Stop mode

39.3 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

39.4 USART implementation

The following tables describe the USART implementation. LPUART is included for comparison.

Table 373. Instance implementation

Instance	Feature set
USART1	Full
USART2	Full
LPUART1	Low-power

Table 374. USART/LPUART features

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	-

Table 374. USART/LPUART features (continued)

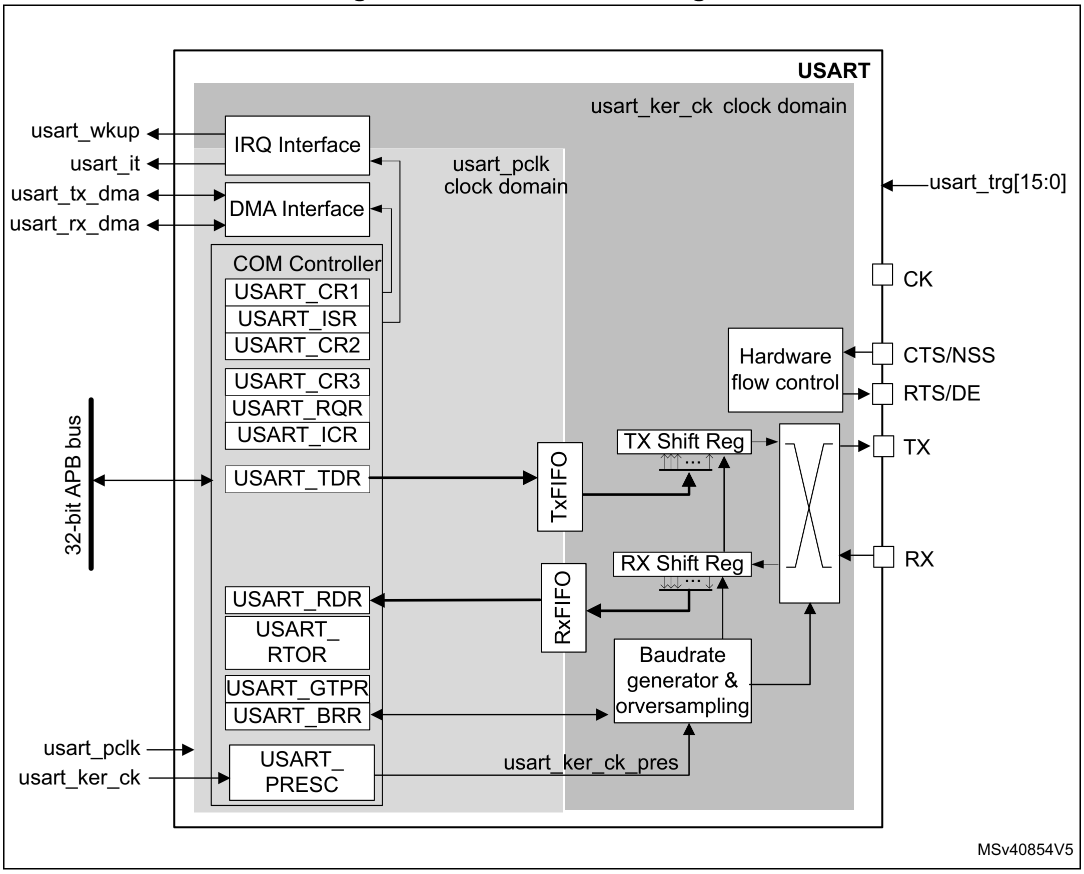
Modes/features[1]	Full feature set	Basic feature set	Low-power feature set
LIN mode	X	X	-
Dual clock domain	X	X	X
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-
Auto baud rate detection	X	X	-
Driver enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size (bytes)	8		
Wake-up from low-power mode	X[2]	X[2]	X[2]
Autonomous mode	X	X	X

1. “X” = supported, “-” = not supported.
2. Wake-up supported from Stop mode.

39.5 USART functional description

39.5.1 USART block diagram

Figure 418. USART block diagram



39.5.2 USART pins and internal signals

Description of 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 SPI 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 SPI master and smartcard modes.

    It acts as clock input in synchronous SPI 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 (such as 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 375* and *Table 376* for the list of USART input/output pins and internal signals.

Table 375. USART/UART input/output pins

Pin name	Signal type	Description
USART_RX/UART_RX	Input	Serial data receive input
USART_TX/UART_TX	Output	Transmit data output
USART_CTS/UART_CTS	Input	Clear to send
USART_RTS/UART_RTS	Output	Request to send
USART_DE(1)/UART_DE(2)	Output	Driver enable
USART_CK	Output	Clock output in synchronous master and smartcard modes.
USART_NSS(3)	Input	Slave select input in synchronous slave mode.

1. USART_DE and USART_RTS share the same pin.
2. UART_DE and UART_RTS share the same pin.
3. USART_NSS and USART_CTS share the same pin.

Description of USART input/output signals

Table 376. USART internal input/output signals

Signal name	Signal type	Description
usart_pclk	Input	APB clock
usart_ker_ck	Input	USART kernel clock
usart_wkup	Output	USART provides a wake-up 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 377. USART interconnection (USART1/2)

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(1)
usart_trg9	comp2_out(1)
usart_trg10	rtc_alra_trg
usart_trg11	rtc_wut_trg
usart_trg12	-
usart_trg13	-
usart_trg14	-
usart_trg15	-

1. Only available for STM32WBA54xx and STM32WBA55xx devices.

39.5.3 USART clocks

The simplified block diagram given in *Section 39.5.1: USART block diagram* 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.

39.5.4 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 419*):

- 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.

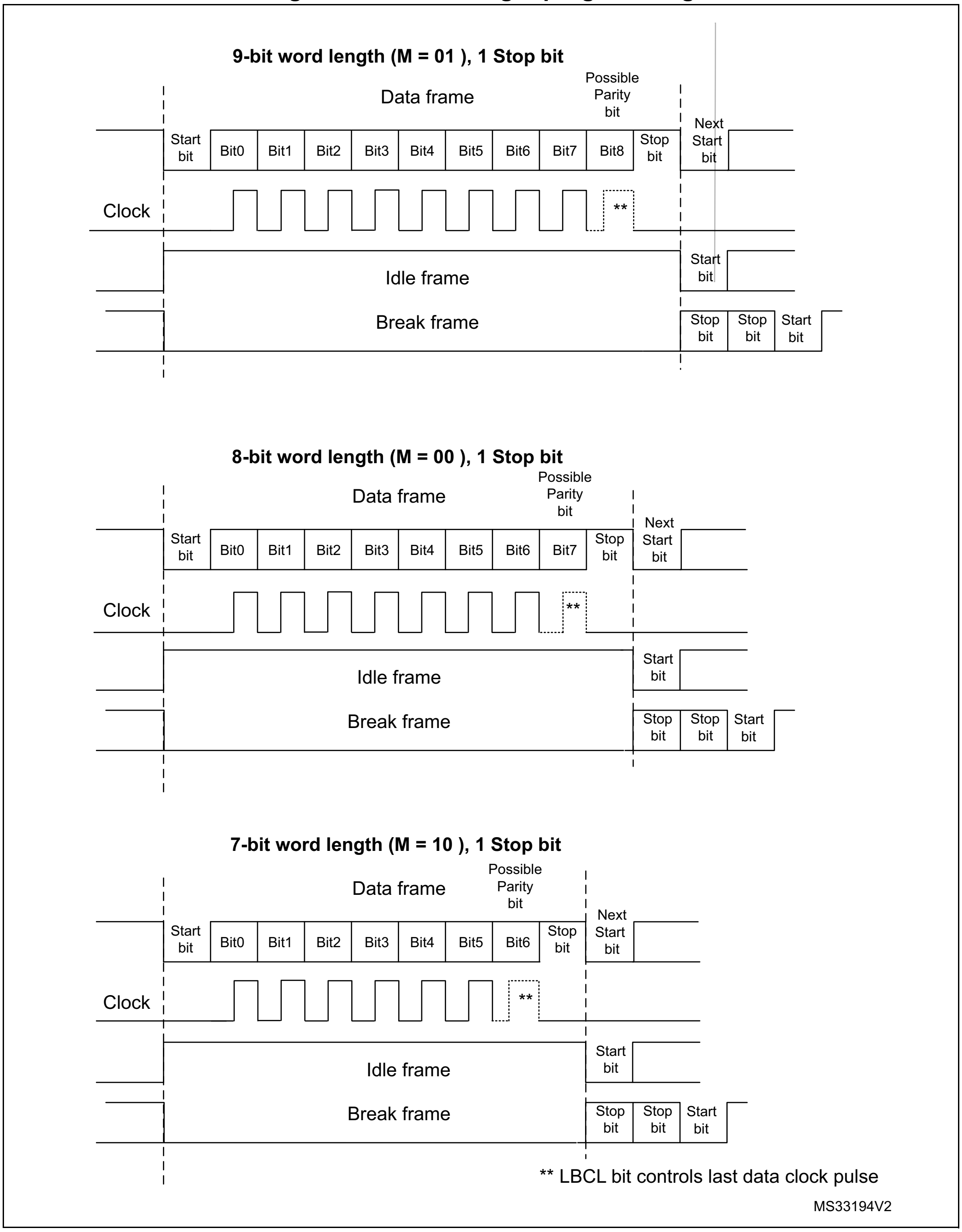
An ***Idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

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 419. Word length programming



39.5.5 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 the RXFTCFG[2:0] and TXFTCFG[2:0] bitfields of the 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[2:0] bitfield.

  In this case, the RXFT flag is set in the USART_ISR register. This means that RXFTCFG[2:0] data have been received: 1 data in USART_RDR and (RXFTCFG[2:0] – 1) data in the RXFIFO. As an example, when RXFTCFG[2:0] 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 is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of the USART_CR3 register.

39.5.6 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 must 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

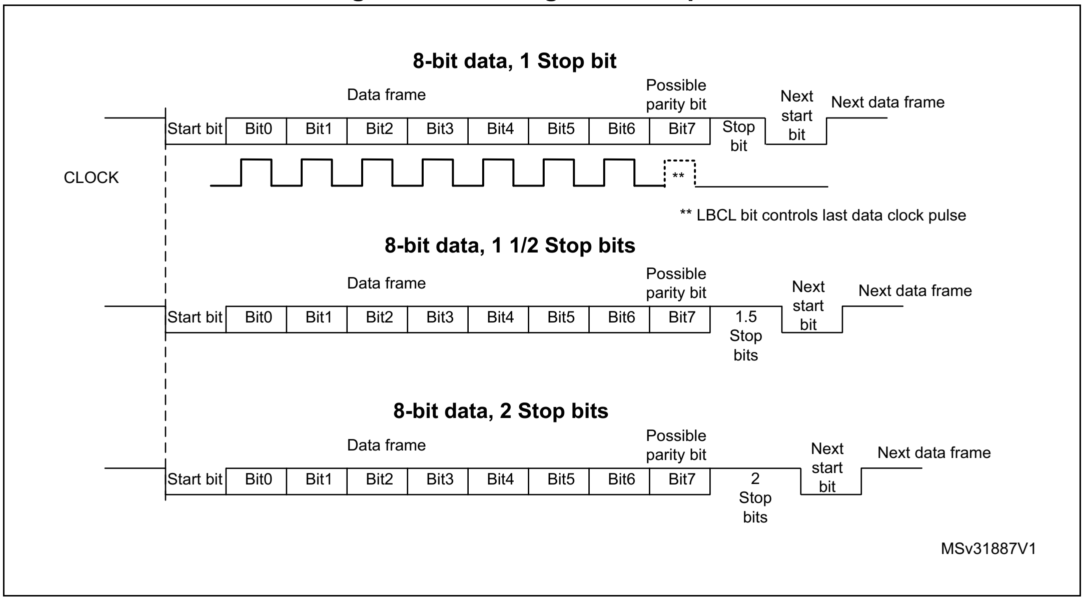
The number of stop bits to be transmitted with every character can be programmed in USART_CR2, bits 13,12.

- ***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 *Section 39.5.1: USART block diagram*). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 420. 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 39.5.20: 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 has completed.
   - 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.

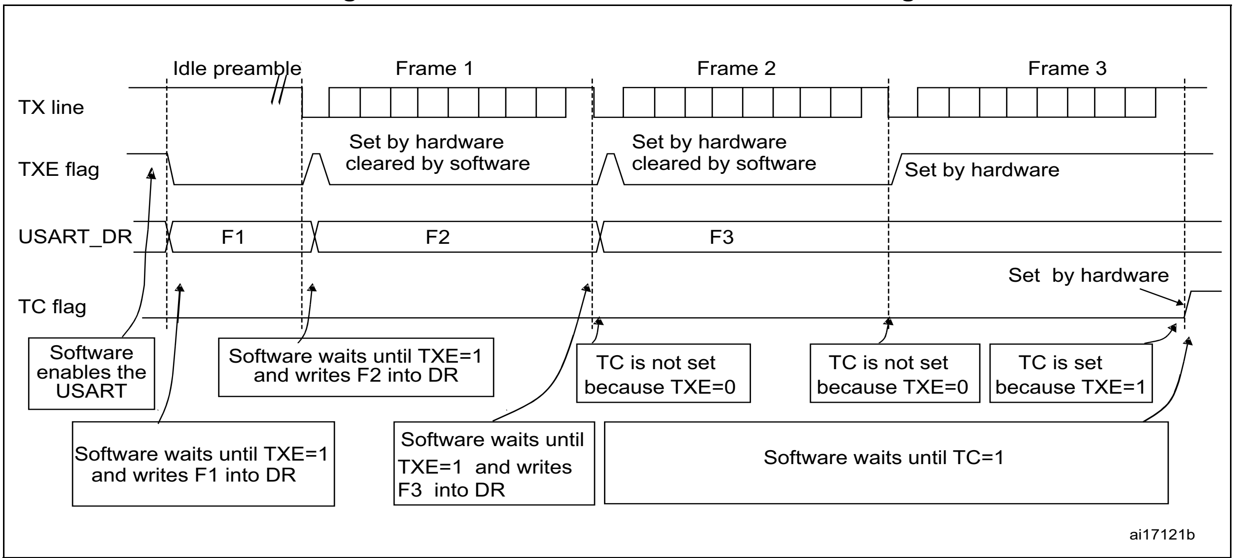
  When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write operation to USART_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.

  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 421*).

Figure 421. 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 419*).

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 complete (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.

39.5.7 USART receiver

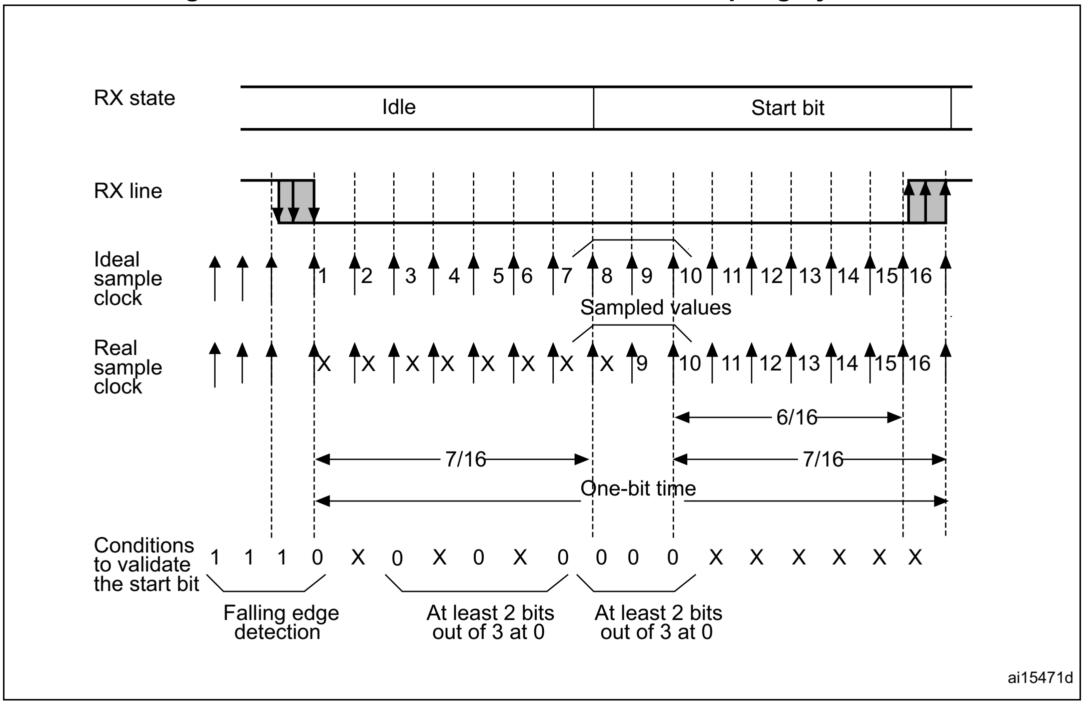
The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

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 422. Start bit detection when oversampling by 16 or 8



Note: If the sequence has not completed, 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,

- 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
- 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 39.5.20: 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, that is 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:

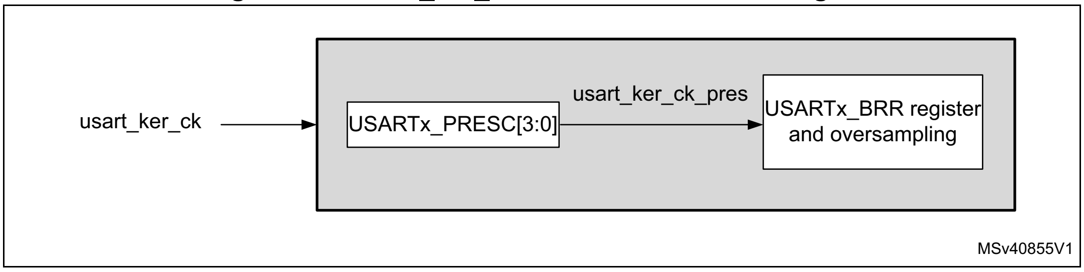
- Possible use of the USART in low-power mode
- Communication speed.

The clock source frequency is usart_ker_ck.

When the dual clock domain and the wake-up 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 423. 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 wake-up 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 424* and *Figure 425*).

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 39.5.9: Tolerance of the USART receiver to clock deviation*)
- 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 378*) 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 39.5.9: Tolerance of the USART receiver to clock deviation*). 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 and IrDA modes.

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 424. Data sampling when oversampling by 16

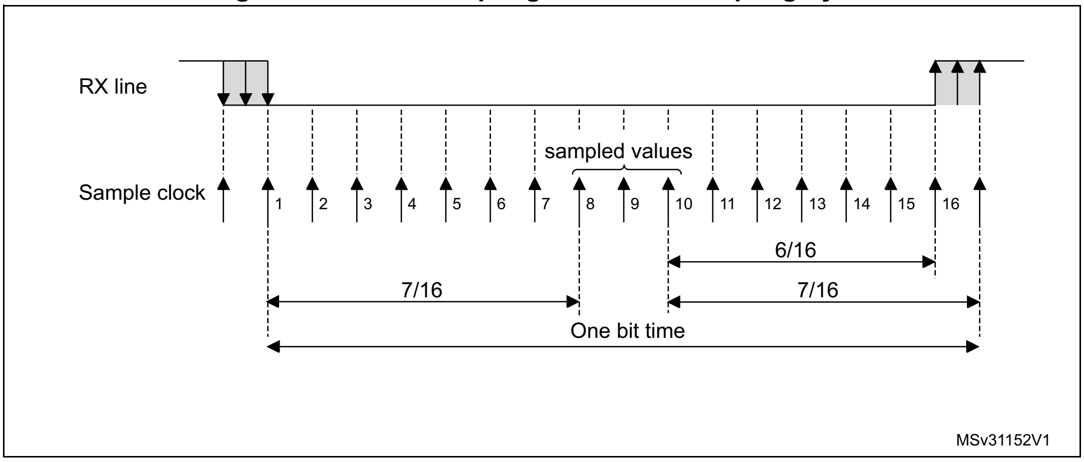


Figure 425. Data sampling when oversampling by 8

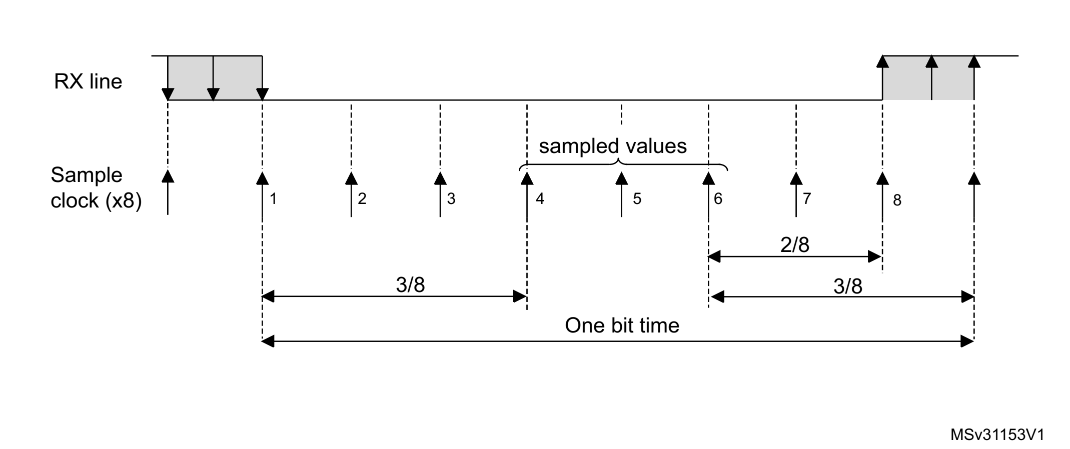


Table 378. Noise detection from sampled data

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 39.5.17: USART receiver timeout* 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.

39.5.8 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 must 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[3:0] = USARTDIV = 0d833 = 0x0341
- In case of oversampling by 8:
  USARTDIV = 2 * 8 000 000/9600
  USARTDIV = 1666,66 (0d1667 = 0x683)
  BRR[3:0] = 0x3 >>1 = 0x1
  BRR[3:0] = 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[3:0] = USARTDIV = 52 = 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[3:0] = 0x64

39.5.9 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)

$$\mathrm{DTRA} + \mathrm{DQUANT} + \mathrm{DREC} + \mathrm{DTCL} + \mathrm{DWU} < \mathrm{USART\ receiver\ tolerance}$$

where

DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

when M[1:0] = 01:

$$\mathrm{DWU} = \frac{t_{\mathrm{WUUSART}}}{11 \times \mathrm{Tbit}}$$

when M[1:0] = 00:

$$\mathrm{DWU} = \frac{t_{\mathrm{WUUSART}}}{10 \times \mathrm{Tbit}}$$

when M[1:0] = 10:

$$\mathrm{DWU} = \frac{t_{\mathrm{WUUSART}}}{9 \times \mathrm{Tbit}}$$

$t_{WUUSART}$ is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 379*, *Table 380*, 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 379. 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 380. 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 379 and Table 380 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).

39.5.10 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/32763 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 must 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.

39.5.11 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:

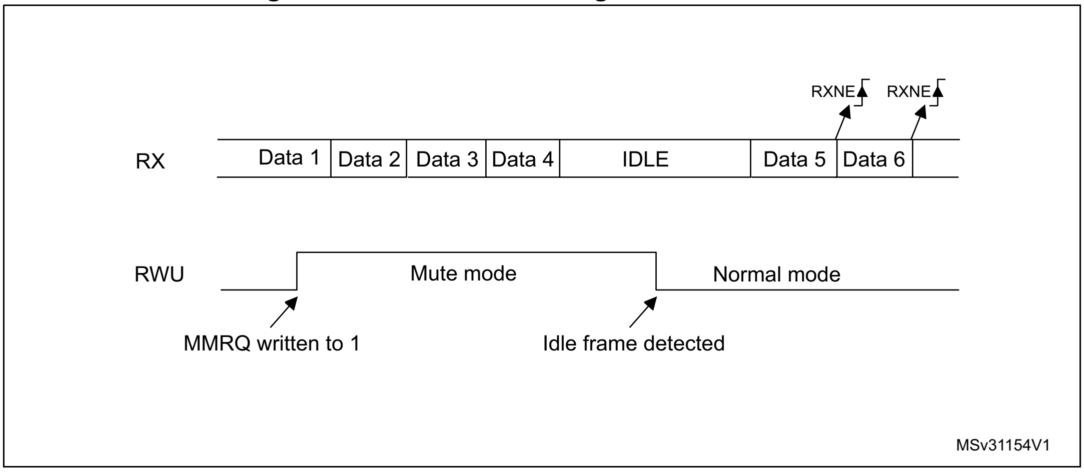
- Idle line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

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 426*.

Figure 426. 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 must ensure that there is at least one empty location in the RXFIFO before entering mute mode.

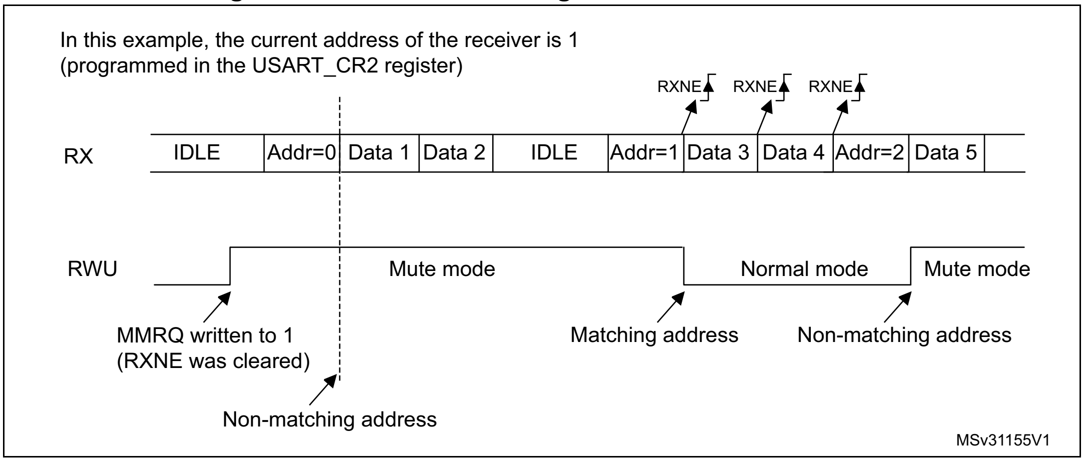
The USART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

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 427*.

Figure 427. Mute mode using address mark detection



39.5.12 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 has not 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.

39.5.13 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 381*.

Table 381. 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)).

39.5.14 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to *Section 39.4: USART implementation*.

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:

- STOP[1:0] and CLKEN in the USART_CR2 register,
- SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The procedure described in *Section 39.5.5* 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 (that is, 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 in *Figure 428*.

Examples of break frames are given in *Figure 429*.

Figure 428. Break detection in LIN mode (11-bit break length - LBDL bit is set)

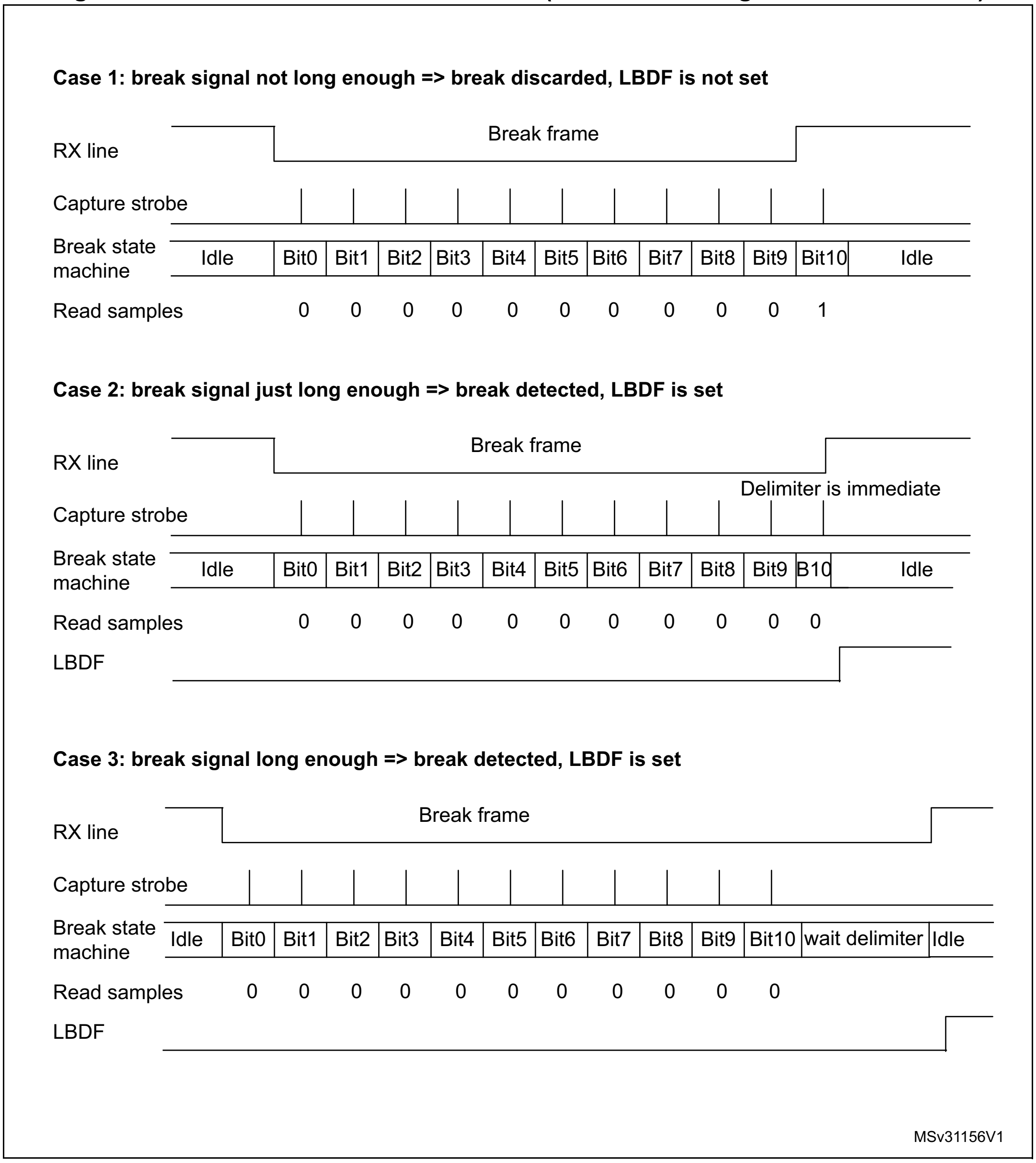
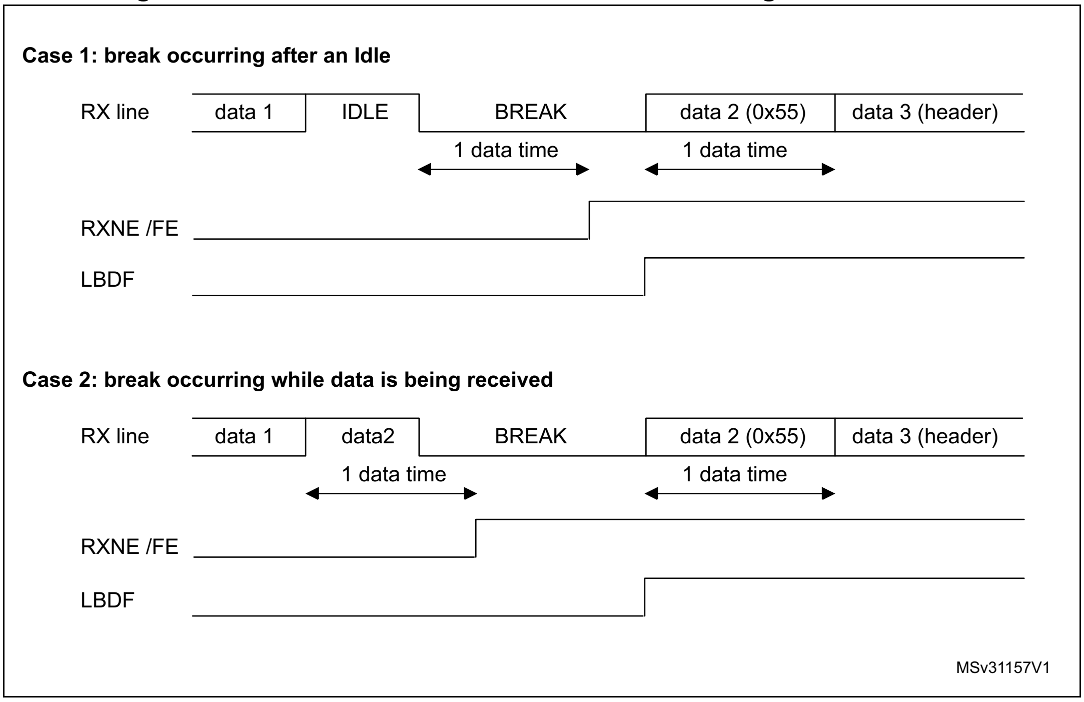


Figure 429. Break detection in LIN mode vs. Framing error detection



39.5.15 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 430*, *Figure 431* and *Figure 432*).

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 430. USART example of synchronous master transmission

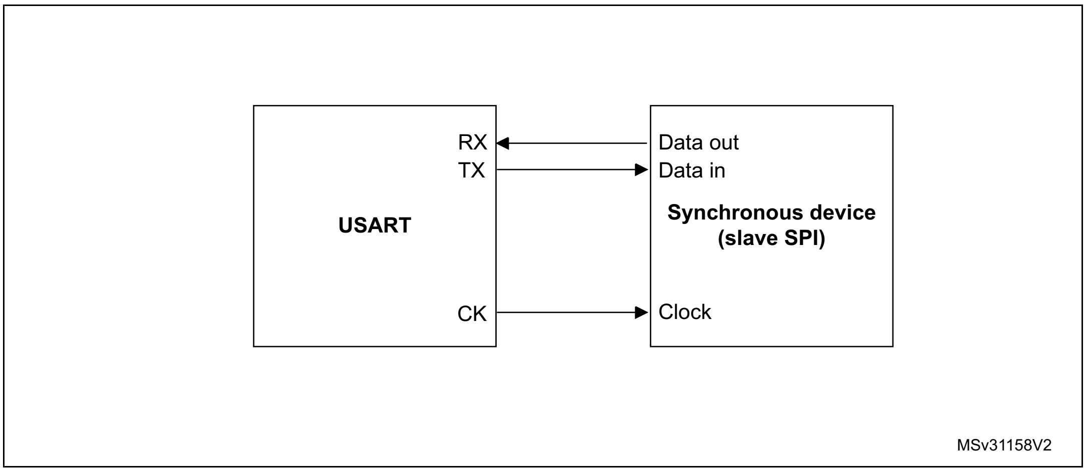


Figure 431. USART data clock timing diagram in synchronous master mode (M bits = 00)

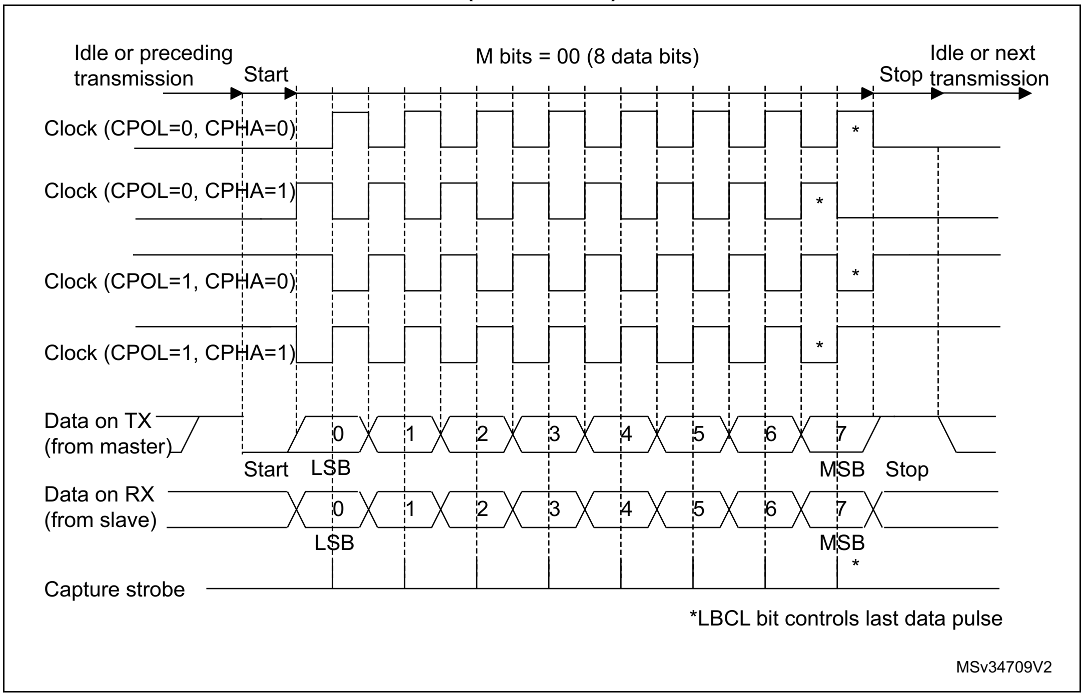
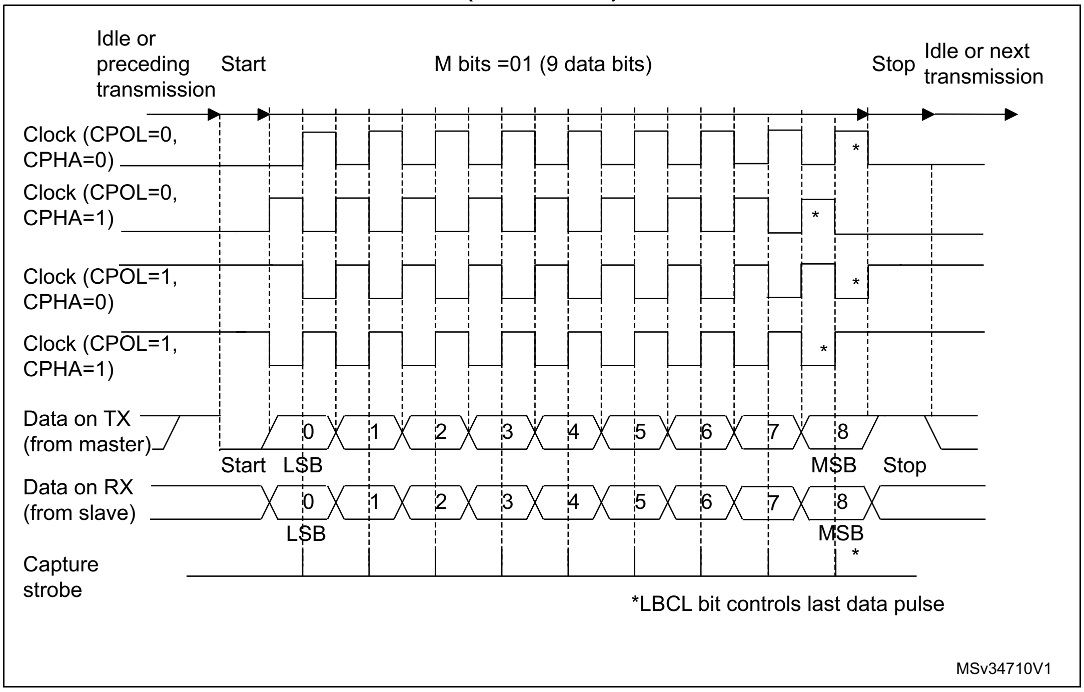


Figure 432. USART data clock timing diagram in synchronous master mode (M bits = 01)



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.

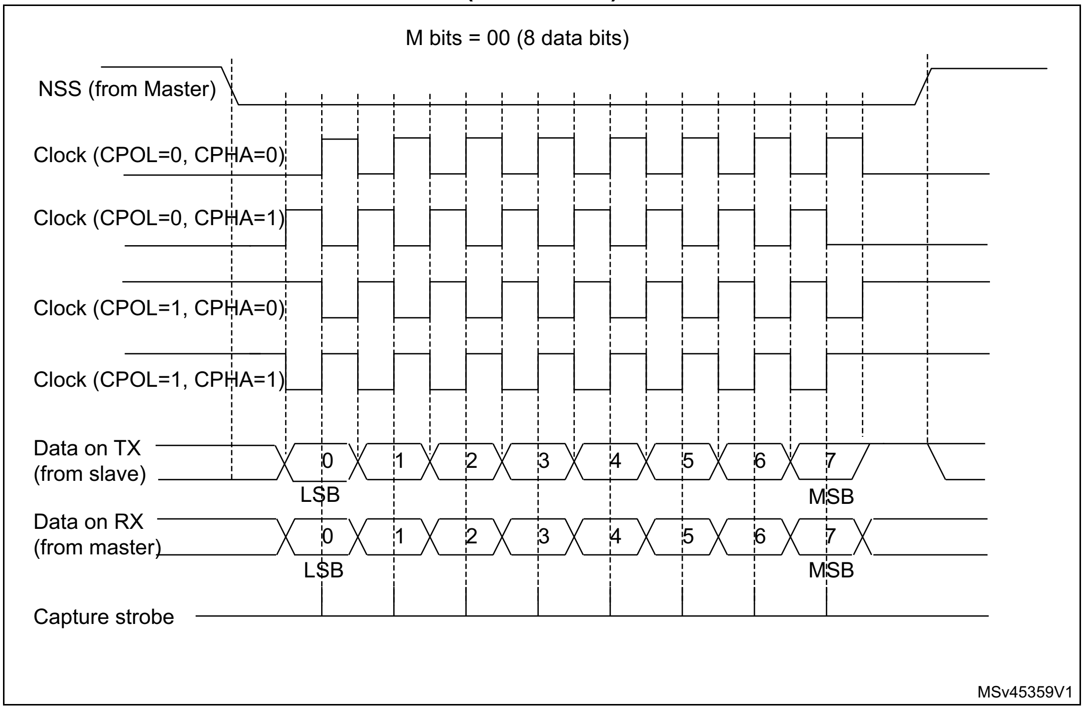
*Note: When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (*usart_ker_ck_pres*) must be greater than 3 times the CK input frequency.*

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 433*).

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 433. 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 must be written 3 usart_ker_ck *cycles before the first CK edge.*

39.5.16 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.

39.5.17 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.

39.5.18 USART smartcard mode

This section is relevant only when smartcard mode is supported. Refer to *Section 39.4: USART implementation*.

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.

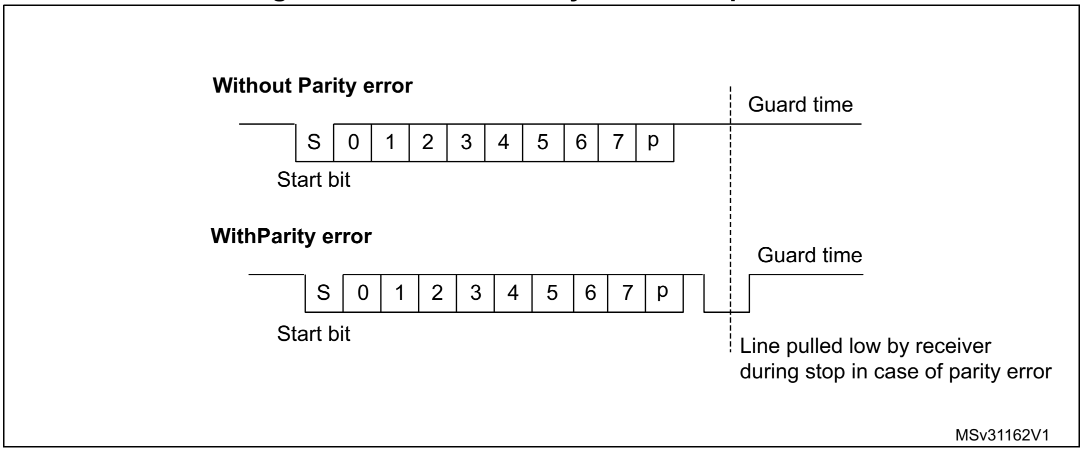
The USART must be configured as:

- 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 434 shows examples of what can be seen on the data line with and without parity error.

Figure 434. 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.

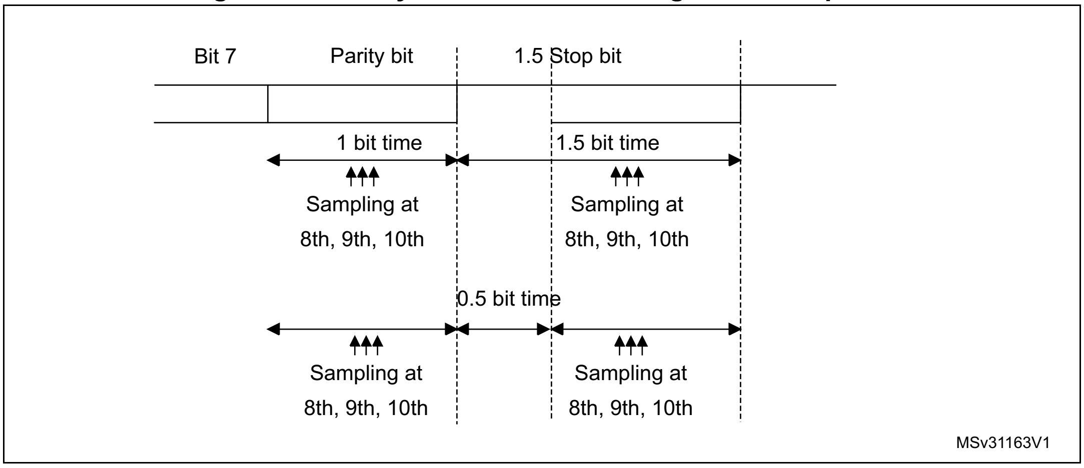
- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guard time). If the software wants to repeat it again, it must ensure 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 435 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 435. Parity error detection using the 1.5 stop bits



The USART can provide a clock to the smartcard through the CK output. In smartcard mode, CK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the USART_GTPR register. CK frequency can be programmed from usart_ker_ck_pres/2 to usart_ker_ck_pres/62, where usart_ker_ck_pres is the peripheral input clock divided by a programmed prescaler.

Block mode (T = 1)

In T = 1 (block) mode, the parity error transmission can be deactivated by clearing the NACK bit in the USART_CR3 register.

When requesting a read from the smartcard, in block mode, the software must program the RTOR register to the BWT (block wait time) – 11 value. If no answer is received from the card before the expiration of this period, a timeout interrupt is generated. If the first character is received before the expiration of the period, it is signaled by the RXNE/RXFNE interrupt.

Note: *The RXNE/RXFNE interrupt must be enabled even when using the USART in DMA mode to read from the smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.*

After the reception of the first character (RXNE/RXFNE interrupt), the RTO register must be programmed to the CWT (character wait time – 11 value), in order to enable the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baud time units. If the smartcard does not send a new character in less than the CWT period after the end of the previous character, the USART signals it to the software through the RTOF flag and interrupt (when RTOIE bit is set).

Note: *As in the smartcard protocol definition, the BWT/CWT values must 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.

39.5.19 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to *Section 39.4: USART implementation*.

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 the use of a return-to-zero, inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see *Figure 436*).

The encoding and decoding of data in IrDA mode is handled by hardware blocks.

Even if IrDA is a half-duplex protocol, it requires two pins for transmission and reception (USART_TX and USART_RX).

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 must be avoided as the data to be transmitted may 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 437*).
- 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 two periods, where the period equals PSC / usart_ker_ck_pres (PSC being the prescaler value programmed in the PSC[7:0] bitfield of the USART_GTPR register). Pulses of width less than one period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than two periods are accepted as a pulse. The IrDA encoder/decoder does not 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 three times the IrDA low-power period. The IrDA low-power frequency minimum value is 1.42 MHz. Generally, this value is 1.8432 MHz (1.42 MHz < IrDA low-power frequency < 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 must discard pulses of duration shorter than one IrDA low-power period. A valid low is accepted only if its duration is greater than two IrDA low-power periods.

Note: IrDA low-power period = PSC / usart_ker_ck_pres, where PSC corresponds to the value programmed in the PSC[7:0] bitfield of *the USART_GTPR register.*

A pulse of width less than two and greater than one IrDA low-power period may or may not be rejected.

The receiver set-up time must be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception.

Figure 436. IrDA SIR ENDEC block diagram

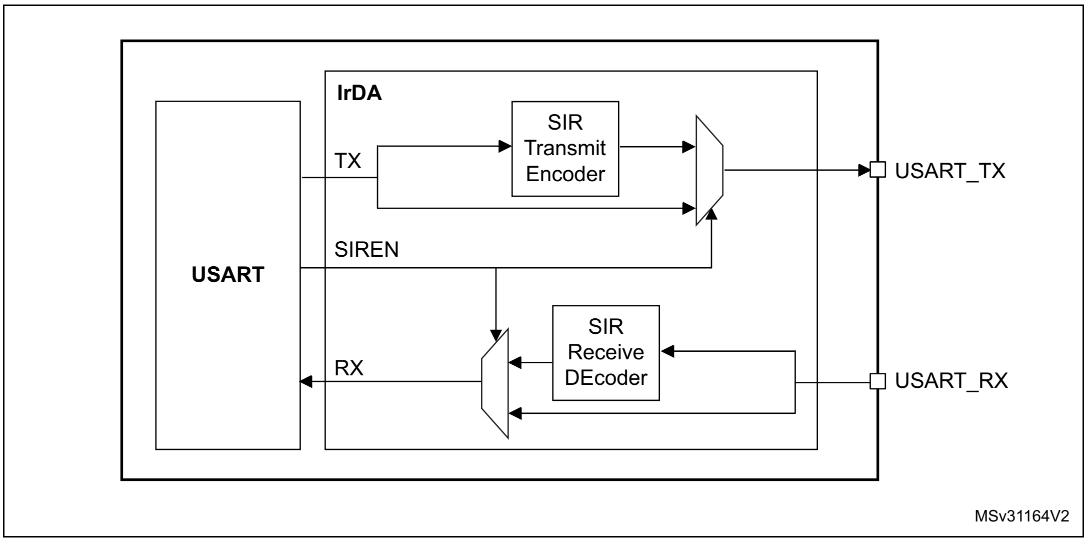
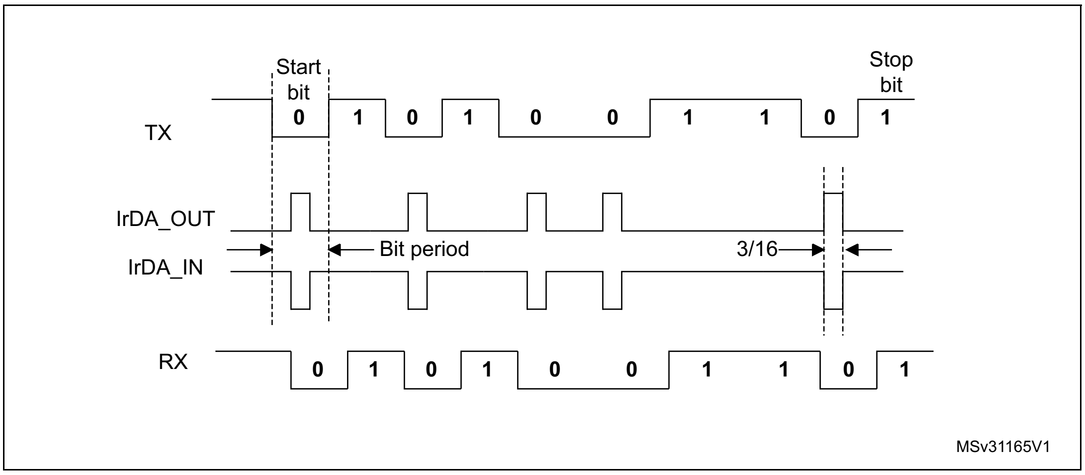


Figure 437. IrDA data modulation (3/16) - normal mode



39.5.20 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 39.4: USART implementation to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in Section 39.5.7. 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 the 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):

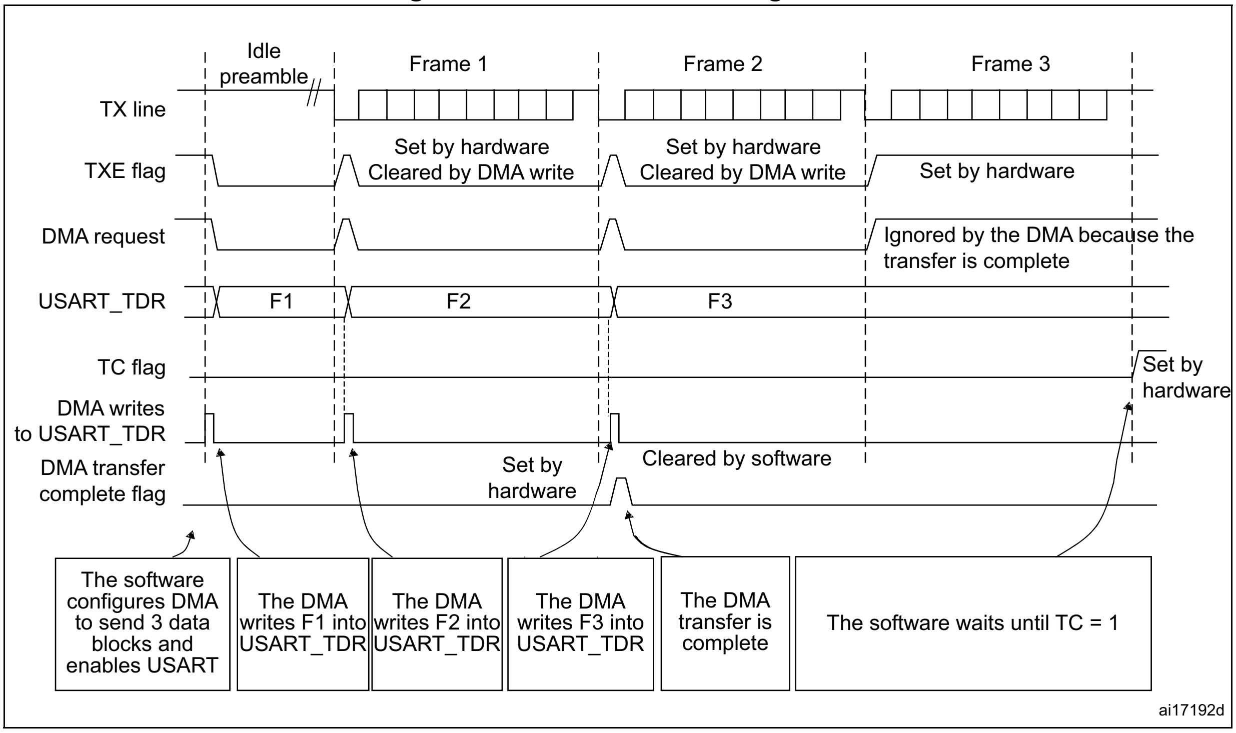
1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the 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 (DMA transfer complete), the TC flag can be monitored to make sure that the USART communication has completed. 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 438. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by transmit FIFO not full (that is, TXFNF = 1).

Reception using DMA

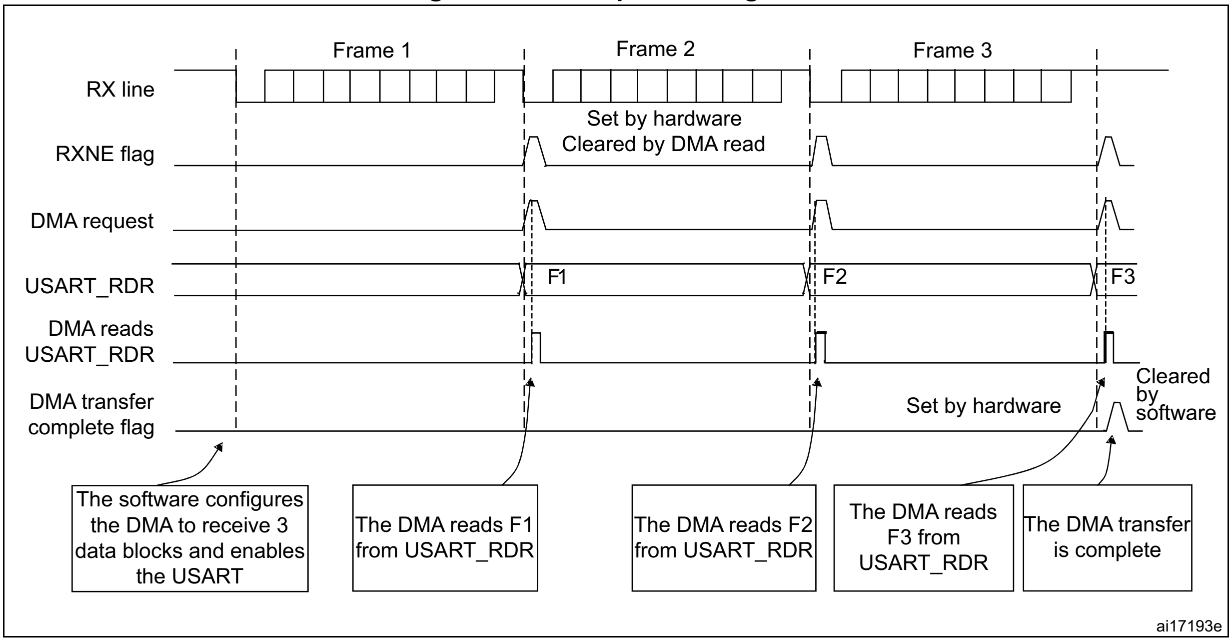
DMA mode can be enabled for reception by setting the DMAR bit in the USART_CR3 register. Data are loaded from the USART_RDR register to an SRAM area configured using the DMA peripheral (refer to *section direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for 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 439. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by receive FIFO not empty (that is, RXFNE = 1).

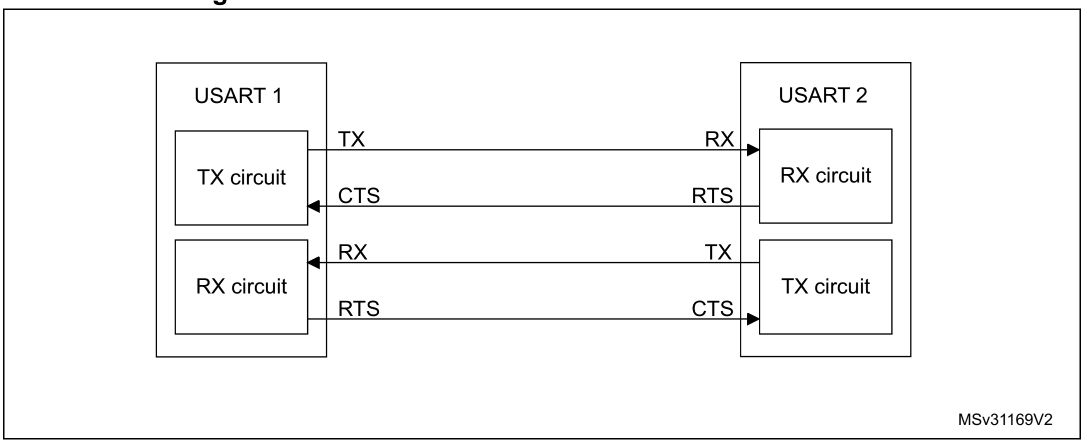
Error flagging and interrupt generation in multibuffer communication

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.

39.5.21 RS232 hardware flow control and RS485 driver enable

It is possible to control the serial data flow between two devices by using the CTS input and the RTS output. The *Figure 440* shows how to connect two devices in this mode:

Figure 440. Hardware flow control between two USARTs

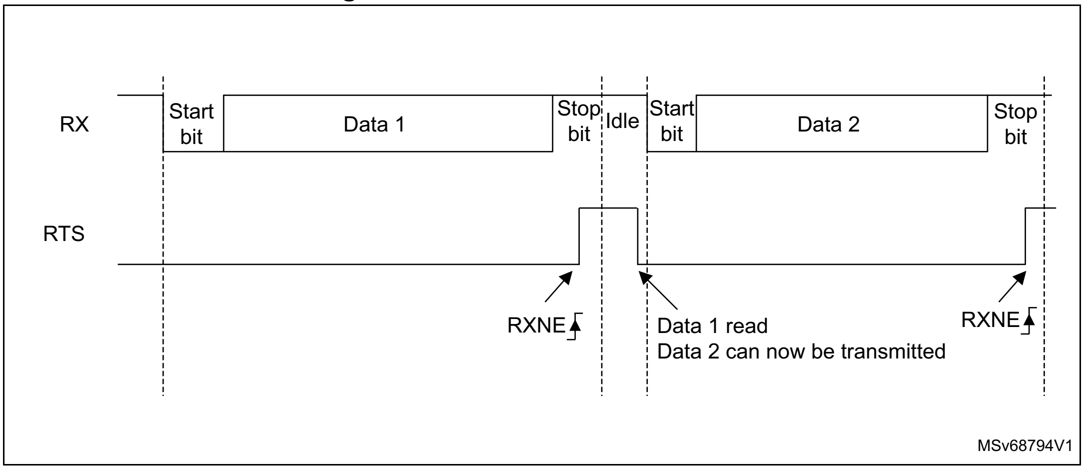


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to 1 in the USART_CR3 register.

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 441* shows an example of communication with RTS flow control enabled.

Figure 441. RS232 RTS flow control



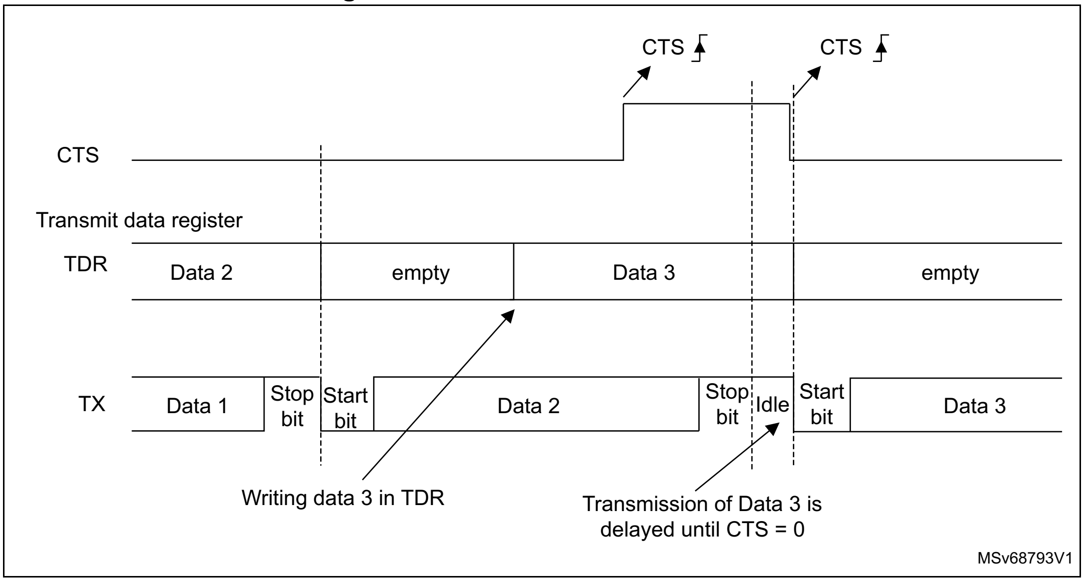
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

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 completes 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 442* shows an example of communication with CTS flow control enabled.

Figure 442. RS232 CTS flow control



Note: For correct behavior, CTS must be deasserted at least three USART clock source periods before the end of the current character. In addition, it must 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 deassertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the USART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the USART_CR3 control register.

In USART, the DEAT and DEDT are expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

39.5.22 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.

USART 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 bitfield 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.

USART 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 wake-up time parameter (refer to the device datasheet) and on the USART receiver tolerance (see *Section 39.5.9: 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 379: 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 wake-up 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 wake up correctly from low-power mode is: 1/11.32 µs = 88.36 kbauds.

39.6 USART in low-power modes

Table 382. 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 39.4* to know if the wake-up 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.

39.7 USART interrupts

Refer to *Table 383* for a detailed description of all USART interrupt requests.

Table 383. 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[(2)]		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 383. 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(3)	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(4)	
	LIN break	LBDF	LBDIE	Write 1 in LBDCF		No	
	Noise error in multibuffer communication.	NE	EIE	Write 1 in NFCF		Yes(4)	
	Overrun error in multibuffer communication.	ORE(5)		Write 1 in ORECF		Yes	
	Framing error in multibuffer communication.	FE		Write 1 in FECF		Yes(4)	
	Character match	CMF	CMIE	Write 1 in CMCF		Yes(6)	
	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 wake-up from Stop mode feature. Refer to *Section 39.4: USART implementation* for the list of supported Stop modes.
2. Writing to TDR clears the TXFT flag only if the number of empty locations is less than the value programmed in TXFTCFG[2:0].
3. 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).
4. Parity/Noise/Framing error interrupts enable waking up from Stop modes when the DMA is used.
5. When OVRDIS = 0.
6. The DMA must be used when the FIFO mode is enabled.

39.8 USART registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

39.8.1 USART control register 1 (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enable, FIFOEN = 1

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 bits

M[1:0] = 01: 1 start bit, 9 Data bits, n stop bits

M[1:0] = 10: 1 start bit, 7 Data bits, n stop bits

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 39.4: USART implementation.

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 39.4: USART implementation.

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 39.4: USART implementation.

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 39.4: USART implementation.

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 wake-up method

This bit determines the USART wake-up 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: When the USART acts as a transmitter, 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 wake-up from low-power mode, this bit is reserved and must be kept at reset value. Refer to Section 39.4: USART implementation.

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.

39.8.2 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, FIFOEN = 0

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 bits
M[1:0] = 01: 1 start bit, 9 Data bits, n stop bits
M[1:0] = 10: 1 start bit, 7 Data bits, n stop bits
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 39.4: USART implementation.

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 39.4: USART implementation.

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 39.4: USART implementation.

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 39.4: USART implementation.

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 wake-up method

This bit determines the USART wake-up 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: When the USART acts as a transmitted, 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 wake-up from low-power mode, this bit is reserved and must be kept at reset value. Refer to Section 39.4: USART implementation.

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.

39.8.3 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 wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter must 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 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 39.4: USART implementation.

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 39.4: USART implementation.

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 39.4: USART implementation.

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 39.4: USART implementation.

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

Note: If neither synchronous mode nor smartcard mode is supported, this bit is reserved and must be kept at reset value. Refer to Section 39.4: USART implementation.

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 39.4: USART implementation.

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 424* and *Figure 425*)

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 39.4: USART implementation.

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 39.4: USART implementation.

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 39.4: USART implementation.

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 39.4: USART implementation.

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.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 39.4: USART implementation.

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.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 39.4: USART implementation.

Note: The CPOL, CPHA, and LBCL bits must not be written while the transmitter is enabled.

39.8.4 USART control register 3 (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXFTIE	RXFTCFG[2:0]			TCBGTIE	TXFTIE	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	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration

000:TXFIFO reaches 1/8 of its depth
001:TXFIFO reaches 1/4 of its depth
010:TXFIFO reaches 1/2 of its depth
011:TXFIFO reaches 3/4 of its depth
100:TXFIFO reaches 7/8 of its depth
101:TXFIFO becomes empty
Others: Reserved, must not be used
This bitfield can only be written when the USART is disabled (UE = 0).

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[2:0].

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
Others: Reserved, must not be used
This bitfield can only be written when the USART is disabled (UE = 0).

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 39.4: USART implementation.

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[2:0].

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 39.4: USART implementation.

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 39.4: USART implementation on page 1452.*

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 39.4: USART implementation on page 1452.*

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 39.4: USART implementation.*

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 completes 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 39.4: USART implementation.

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 39.4: USART implementation.

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 39.4: USART implementation.

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 39.4: USART implementation.

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 39.4: USART implementation.

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 39.4: USART implementation.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE = 1 or ORE = 1 or NE = 1 or UDR = 1 in the USART_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE = 1 or ORE = 1 or NE = 1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

39.8.5 USART control register 3 [alternate] (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCBG TIE	Res.	Res.	Res.	Res.	SCARCNT[2:0]			Res.
							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 31:25 Reserved, must be kept at reset value.

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 39.4: USART implementation.

Bit 23 Reserved, must be kept at reset value.

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 39.4: USART implementation.

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 39.4: USART implementation.

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 39.4: USART implementation.

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 39.4: USART implementation.

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 completes 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 39.4: USART implementation.

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 39.4: USART implementation.

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 39.4: USART implementation.

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 39.4: USART implementation.

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 39.4: USART implementation.

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 39.4: USART implementation.

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.

39.8.6 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.

39.8.7 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 39.4: USART implementation.

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 39.4: USART implementation.

39.8.8 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 39.4: USART implementation.

39.8.9 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, that is 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 must 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 39.4: USART implementation.

39.8.10 USART interrupt and status register (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 28 if FIFO/smartcard mode supported

XX = 08 if FIFO supported and smartcard mode not supported

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

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 number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of USART_CR3 register.

An interrupt is generated if the TXFTIE bit (bit 31) is set 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 the RXFTCFG[2:0] bitfield of the USART_CR3 register is reached. This means that there are (RXFTCFG[2:0] – 1) data in the Receive FIFO and one data in the USART_RDR register. An interrupt is generated if the RXFTIE bit = 1 (bit 28) 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[2:0] threshold is configured to 101, the RXFT flag is set if RXFIFO size data are available, that is, (RXFIFO size – 1) data in the RXFIFO and 1 data in the USART_RDR. Consequently, the (RXFIFO size + 1) th received data does not cause an overrun error. The overrun error occurs after receiving the (RXFIFO size + 2) th 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 has completed 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 has not completed, or transmission has completed unsuccessfully (that is, a NACK is received from the card)

1: Transmission has completed 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 39.4: USART implementation.

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 wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 39.4: USART implementation.

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 wake-up from mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up 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 wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 39.4: USART implementation.

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 ongoing

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 has 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 39.4: USART implementation.

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 39.4: USART implementation.

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 39.4: USART implementation.

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 must 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 has completed 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.

The TC bit is cleared by software, by writing 1 to the TCCF of 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 (that is, 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 39.5.9: Tolerance of the USART receiver to clock deviation).

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.

39.8.11 USART interrupt and status register [alternate] (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 28 if FIFO/smartcard mode supported

XX = 08 if FIFO supported and smartcard mode not supported)

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

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 has completed, 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 has not completed or transmission has completed unsuccessfully (that is, a NACK is received from the card)

1: Transmission has not completed 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 39.4: USART implementation.

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 wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 39.4: USART implementation.

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 wake-up from mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up 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 wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 39.4: USART implementation.

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 = 1 in 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 ongoing

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 has 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 39.4: USART implementation.

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 39.4: USART implementation.

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 39.4: USART implementation.

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 empty

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 has completed 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 (that is, 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 39.5.9: Tolerance of the USART receiver to clock deviation).

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.

39.8.12 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 39.4: USART implementation

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 39.4: USART implementation.

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 39.4: USART implementation.

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 39.4: USART implementation.

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 39.4: USART implementation.

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.

39.8.13 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 *Section 39.5.1: USART block diagram*).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

39.8.14 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 *Section 39.5.1: USART block diagram*).

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.

39.8.15 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
Others: Reserved, must not be used

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is equal to 1011 that is, input clock divided by 256.

If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to Section 39.4: USART implementation.

39.8.16 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.

39.8.17 USART register map

Table 384. 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
0x08	USART_CR3 FIFO mode enabled	TXFTCFG[2:0]			RXFTIE	RXFTCFG2:0]			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
0x08	USART_CR3 FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCBGTIE	Res.	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
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

Table 384. 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
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: Memory organization* for the register boundary addresses.

40 Low-power universal asynchronous receiver transmitter (LPUART)

40.1 Introduction

The LPUART is an UART, which enables bidirectional UART communications with a limited power consumption. Only a 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.

40.2 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 RS485 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: wake-up from mute mode by idle line detection or address mark detection
- Wake-up from Stop mode
- Autonomous functionality in Stop mode

40.3 LPUART implementation

The tables below describe the LPUART implementation. USARTs and UARTs are included for comparison.

Table 385. Instance implementation

Instance	Feature set
USART1	Full
USART2	Full
LPUART1	Low-power

Table 386. USART/LPUART features

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

Table 386. USART/LPUART features (continued)

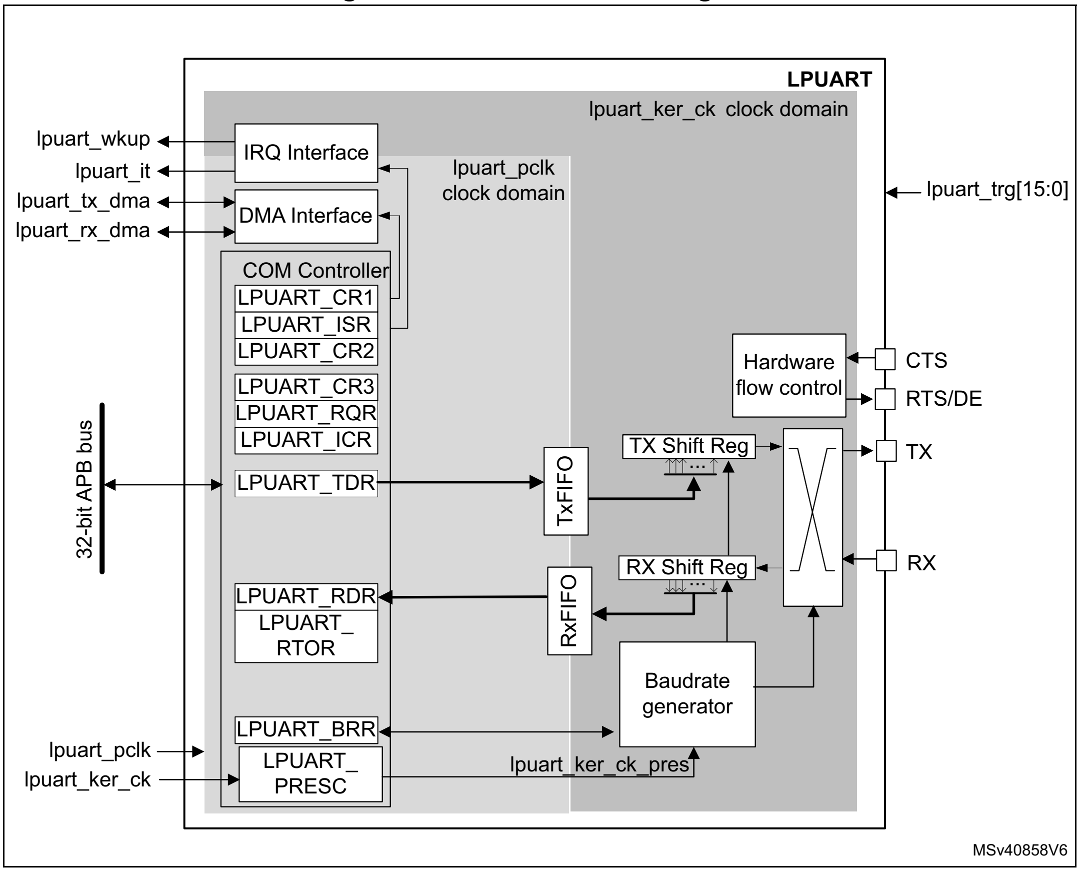
Modes/features[(1)]	Full feature set	Basic feature set	Low-power feature set
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size (bytes)	8		
Wake-up from low-power mode	X[(2)]	X[(2)]	X[(2)]
Autonomous mode	X	X	X

1. X = supported.
2. Wake-up supported from Stop mode.

40.4 LPUART functional description

40.4.1 LPUART block diagram

Figure 443. LPUART block diagram



40.4.2 LPUART pins and internal signals

Description LPUART input/output pins

- LPUART bidirectional communications

  LPUART 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.
  - **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 RS232 hardware flow control mode requires the following pins:
  - **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 LPUART 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 387* and *Table 388* for the list of LPUART input/output pins and internal signals.

Table 387. 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 388. LPUART internal input/output signals

Signal name	Signal type	Description
lpuart_pclk	Input	APB clock
lpuart_ker_ck	Input	LPUART kernel clock
lpuart_wkup	Output	LPUART provides a wake-up interrupt

Table 388. LPUART internal input/output signals (continued)

Signal 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 389. LPUART interconnections (LPUART1)

Signal name	Source
lpuart_trg0	gpdma1_ch0_tc
lpuart_trg1	gpdma1_ch1_tc
lpuart_trg2	gpdma1_ch2_tc
lpuart_trg3	gpdma1_ch3_tc
lpuart_trg4	exti6
lpuart_trg5	exti8
lpuart_trg6	lptim1_ch1
lpuart_trg7	-
lpuart_trg8	comp1_out(1)
lpuart_trg9	comp2_out(1)
lpuart_trg10	rtc_alra_trg
lpuart_trg11	rtc_wut_trg
lpuart_trg12	-
lpuart_trg13	-
lpuart_trg14	-
lpuart_trg15	-

1. Available only on STM32WBA54/55xx devices.

40.4.3 LPUART clocks

The simplified block diagram given in *Section 40.4.1: LPUART block diagram* 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.

40.4.4 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 419*).

- 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.

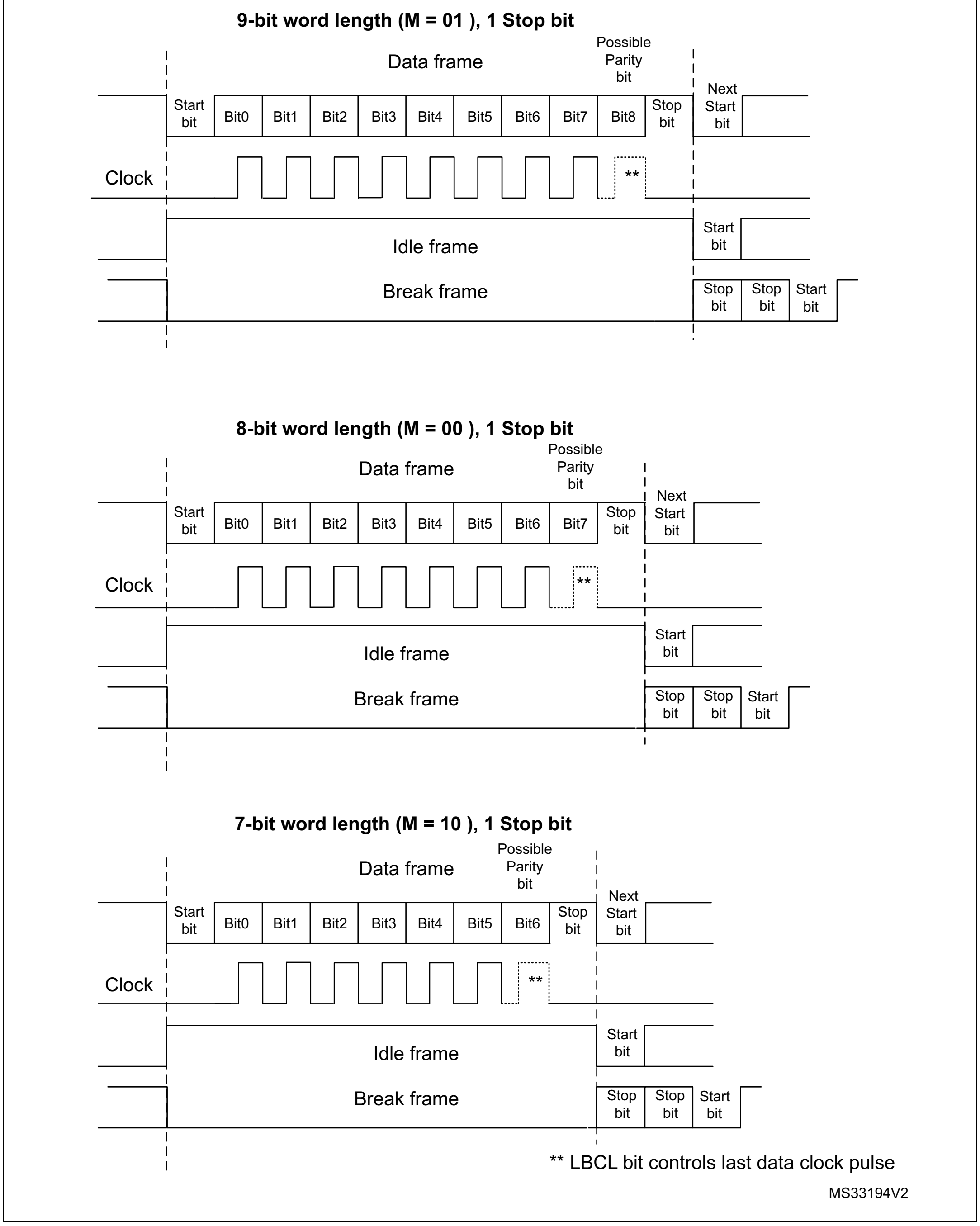
An ***idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

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 444. LPUART word length programming



40.4.5 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 the LPUART_CR1 register.

Since 9 bits the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However, the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the LPUART_ISR register.

It is possible to define the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through the RXFTCFG[2:0] and TXFTCFG[2:0] bitfields of the 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[2:0] bitfield.

  In this case, the RXFT flag is set in the LPUART_ISR register. This means that RXFTCFG[2:0] data have been received: 1 data in LPUART_RDR and (RXFTCFG[2:0] – 1) data in the RXFIFO. As an example, when the RXFTCFG[2:0] 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 is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of the LSART_CR3 register.

40.4.6 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 *Section 40.4.1: LPUART block diagram*).

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 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 must 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

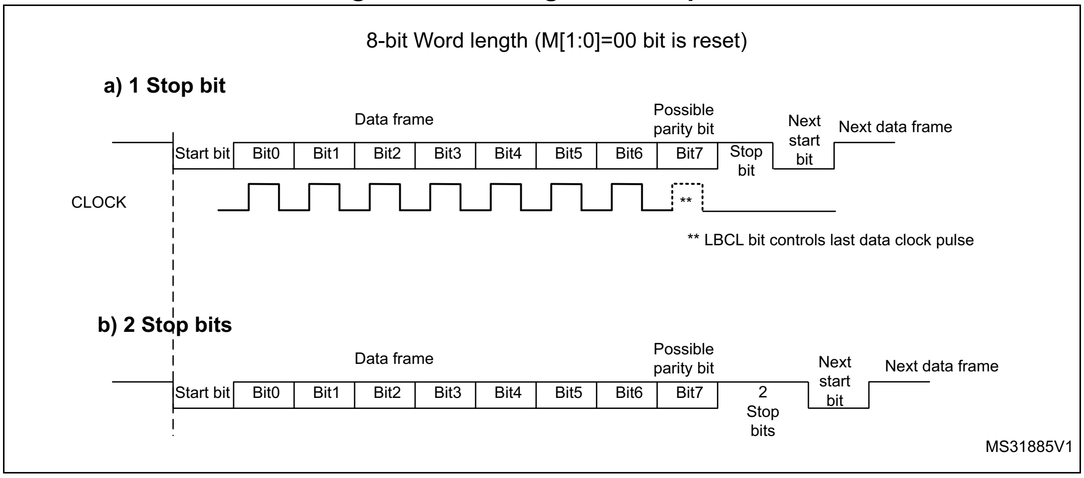
The number of stop bits to be transmitted with every character can be programmed in LPUART_CR2 (bits 13, 12).

- ***1 stop bit***: This is the default value of the 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 445. 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 multibuffer communication is to take place. Configure the DMA register as explained in *Section 40.4.13: 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 the 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 has been completed.
   – When FIFO mode is disabled, this indicates that the transmission of the last frame has been completed.
   – 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.

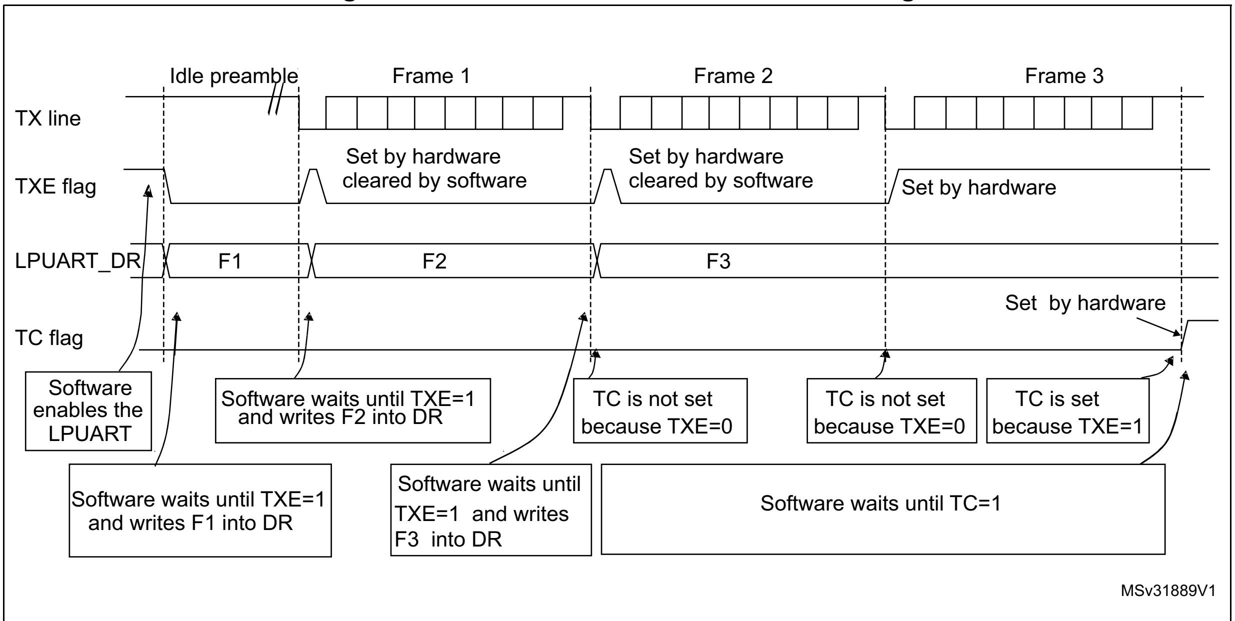
When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write in LPUART_TDR. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNEIE bit is set.

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 in 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 446: TC/TXE behavior when transmitting*).

Figure 446. 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 444*).

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 complete (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.

40.4.7 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 with the 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 40.4.13: 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, that is, there is a data in the RXFIFO to be read.
- In single-buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the LPUART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every read operation from the LPUART_RDR register, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by writing 1 to the RXFRQ bit in the LPUART_RQR register. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the LPUART handles it as a framing error.

Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled

  An overrun error occurs when a character is received when RXNE has not been reset.

  Data cannot 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 the 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 cannot 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:

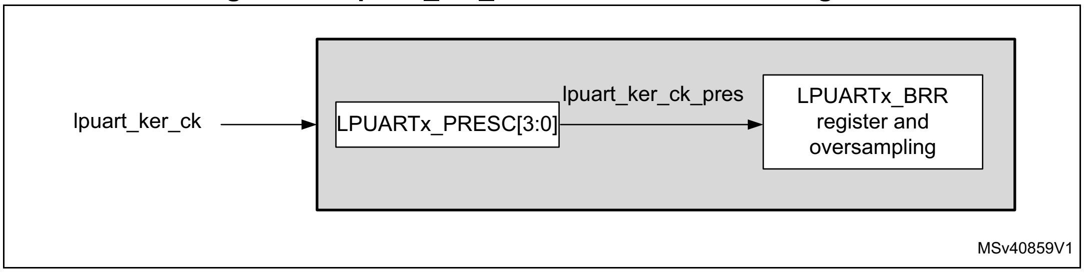
- Possible use of the LPUART in low-power mode
- Communication speed.

The clock source frequency is lpuart_ker_ck.

When the dual clock domain and the wake-up 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 447. 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 wake-up 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 desynchronization 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, that is, during the second stop bit. The first stop bit is not checked for framing error.

40.4.8 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 must not be changed during communication.

It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

lpuart_ker_ck_pres 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.

Table 390. 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	300 baud	0x6D3A	0
2	0.6 kbaud	600 baud	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

40.4.9 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 that contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers, which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$\mathrm{DTRA} + \mathrm{DQUANT} + \mathrm{DREC} + \mathrm{DTCL} + \mathrm{DWU} < \text{LPUART receiver tolerance}$$

where

DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

when M[1:0] = 01:

$$\mathrm{DWU} = \frac{t_{WULPUART}}{11 \times \mathrm{Tbit}}$$

when M[1:0] = 00:

$$\mathrm{DWU} = \frac{t_{WULPUART}}{10 \times \mathrm{Tbit}}$$

when M[1:0] = 10:

$$\mathrm{DWU} = \frac{t_{WULPUART}}{9 \times \mathrm{Tbit}}$$

$t_{WULPUART}$ 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 391*:

- Number of stop bits defined through STOP[1:0] bits in the LPUART_CR2 register
- LPUART_BRR register value.

Table 391. 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 bits	2.08%	2.86%	4.35%	4.42%
9 bits (M = 01), 2 stop bits	1.82%	2.56%	3.90%	4.42%
7 bits (M = 10), 2 stop bits	2.34%	3.23%	4.92%	4.42%

Note: The data specified in Table 391 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).

40.4.10 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 nonaddressed receivers.

The nonaddressed 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 the 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:

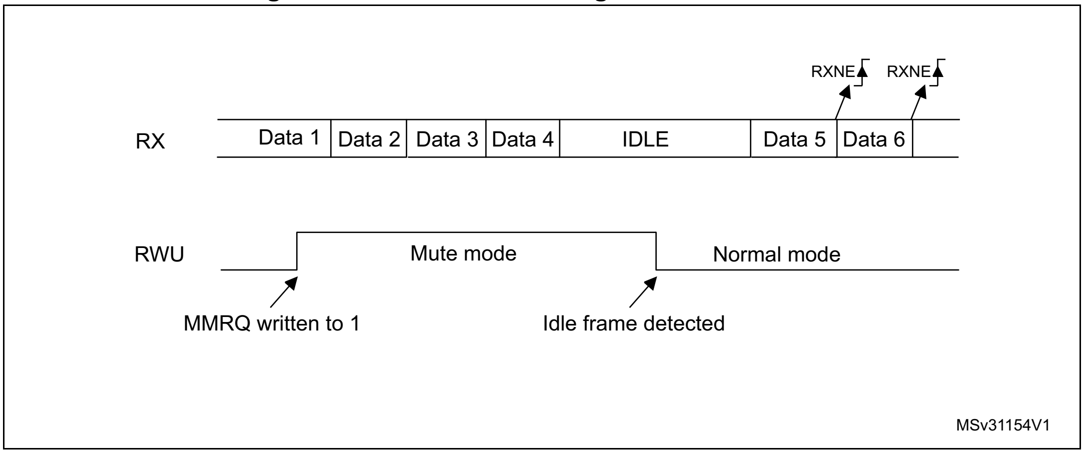
- Idle Line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

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 448.

Figure 448. 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.

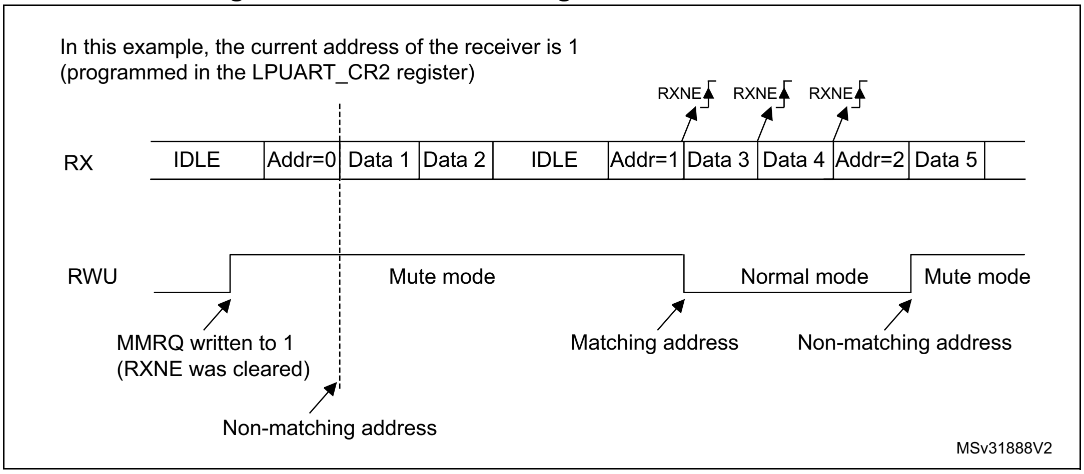
The LPUART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

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 449*.

Figure 449. Mute mode using address mark detection



40.4.11 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 392*.

Table 392: 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)).

40.4.12 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.

40.4.13 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 40.3: LPUART implementation to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in Section 40.4.7. 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 the 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):

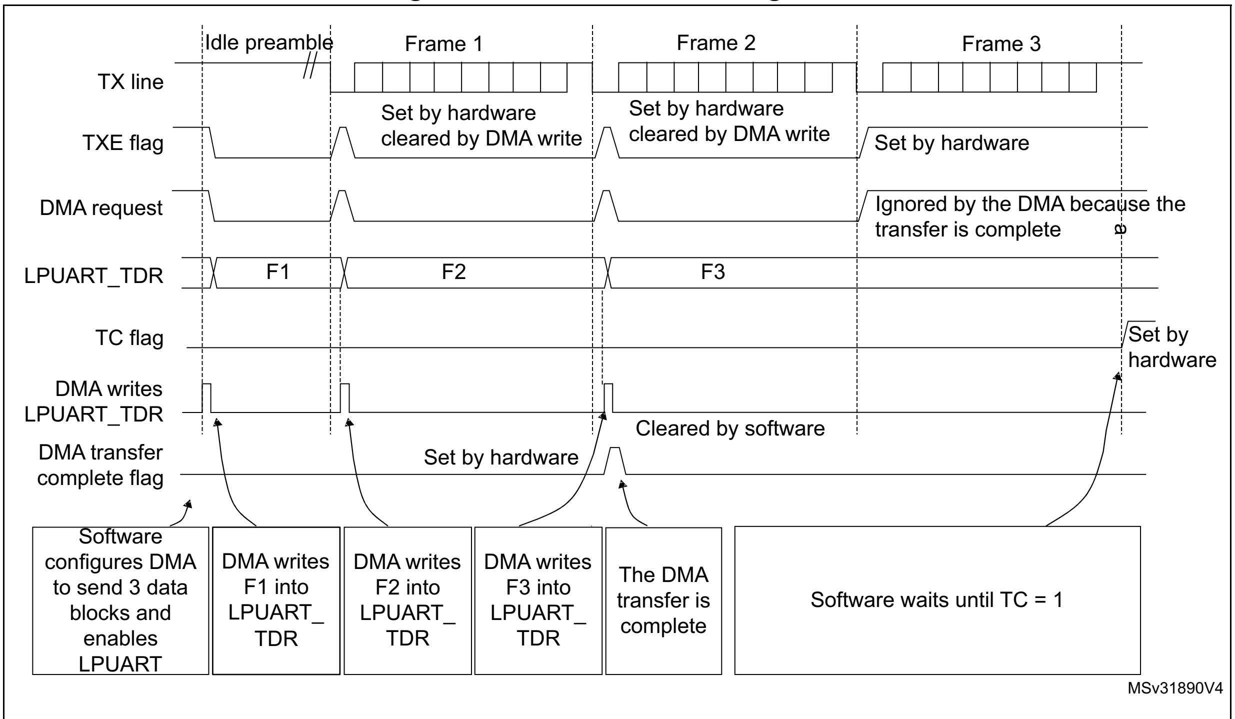
1. Write the LPUART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the 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 (DMA transfer complete), the TC flag can be monitored to make sure that the LPUART communication has completed. 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 must not be cleared before the DMA end of transfer.

Figure 450. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by transmit FIFO not full (that is, TXFNF = 1).

Reception using DMA

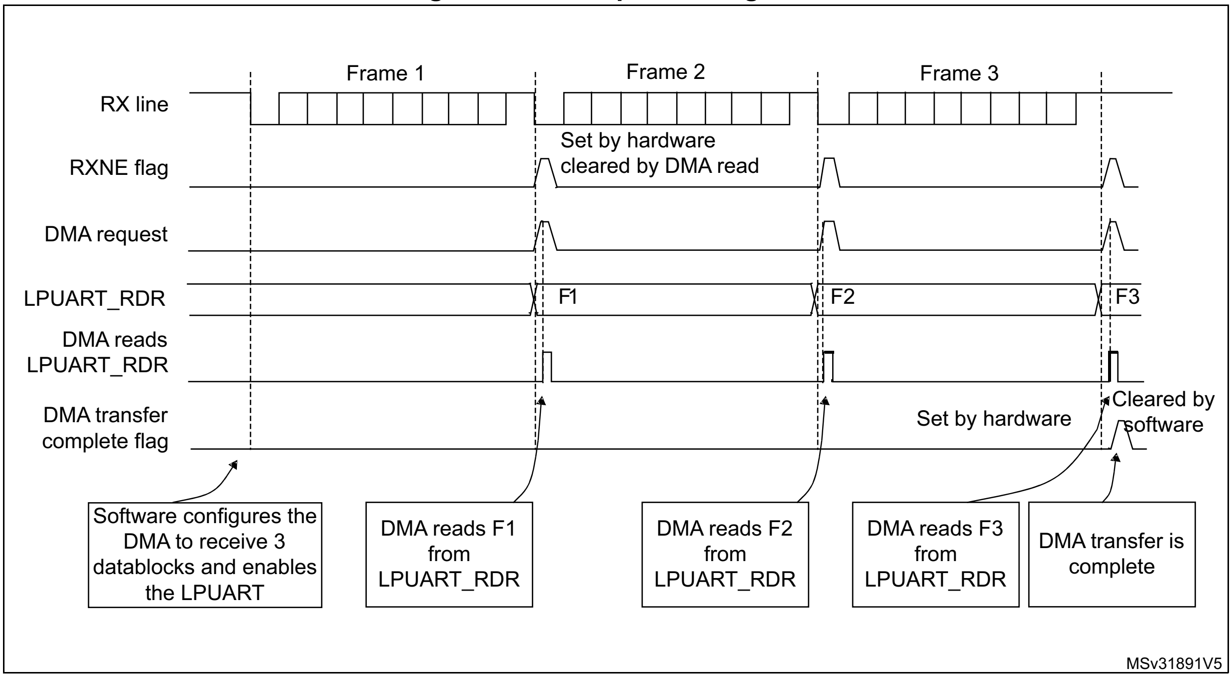
DMA mode can be enabled for reception by setting the DMAR bit in the LPUART_CR3 register. Data are loaded from the LPUART_RDR register to a SRAM area configured using the DMA peripheral (refer to section *direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for 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 must not be cleared before the DMA end of transfer.

Figure 451. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by receive FIFO not empty (that is, RXFNE = 1).

Error flagging and interrupt generation in multibuffer communication

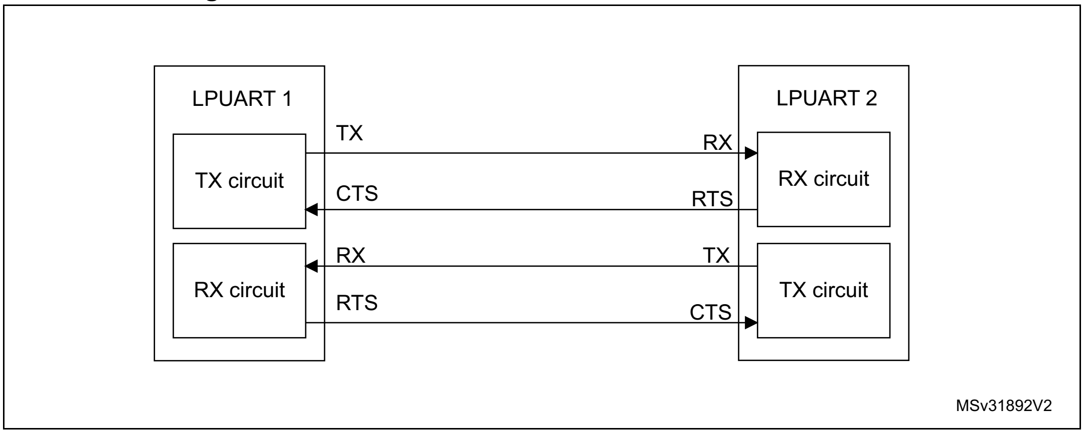
If any error occurs during a transaction In multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set.

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.

40.4.14 RS232 hardware flow control and RS485 driver enable

It is possible to control the serial data flow between two devices by using the CTS input and the RTS output. *Figure 452* shows how to connect two devices in this mode.

Figure 452. Hardware flow control between two LPUARTs

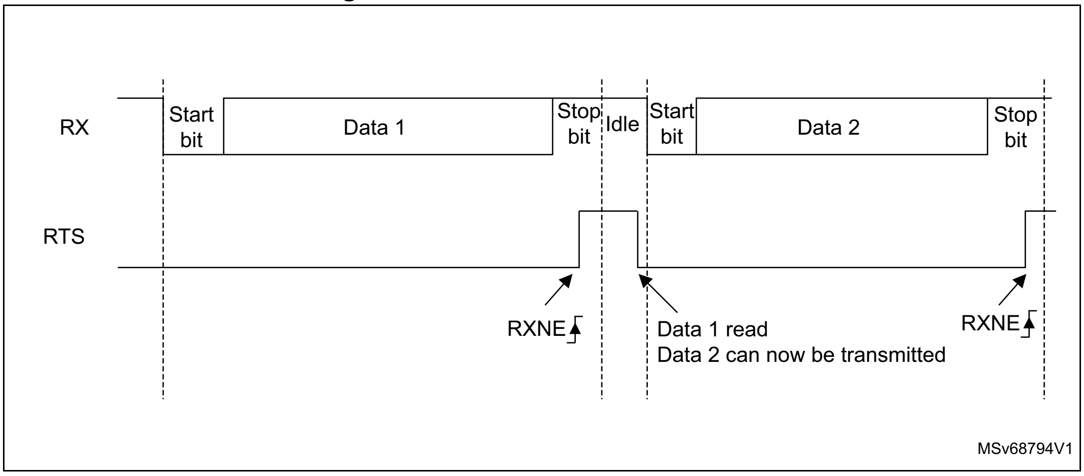


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART_CR3 register).

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 453* shows an example of communication with RTS flow control enabled.

Figure 453. RS232 RTS flow control



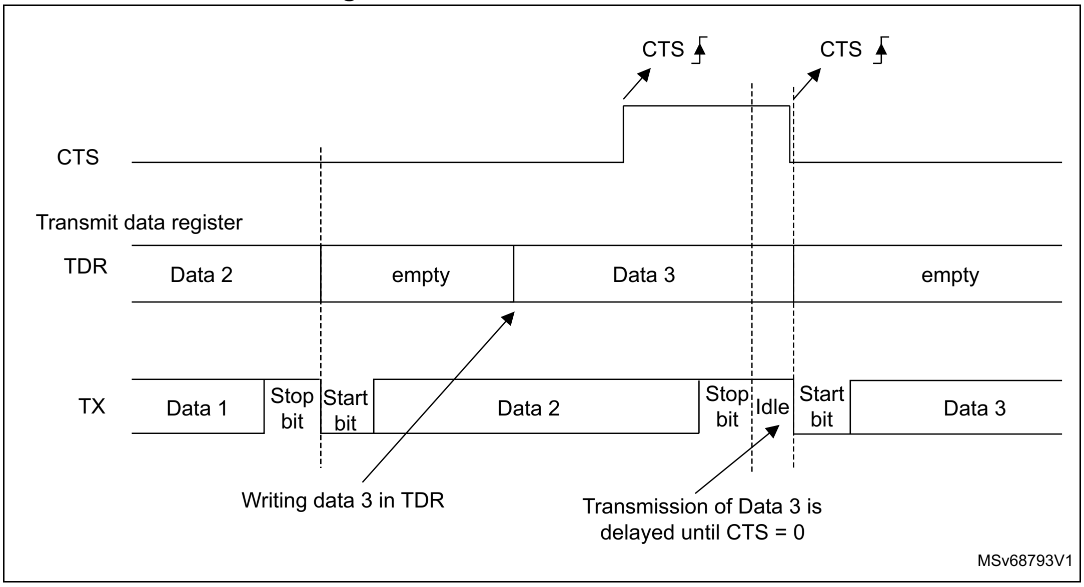
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

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 completes 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 454* shows an example of communication with CTS flow control enabled.

Figure 454. 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 must be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the LPUART_CR3 control register. This enables activating the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the LPUART_CR1 control register. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the deactivation 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 ($f_{lpuart\_ker\_ck\_pres}$) cycles:

- The driver enable assertion time equals
  – (1 + (DEAT x P)) x lpuart_ker_ck_pres, if P # 0
  – (1 + DEAT) x lpuart_ker_ck_pres, if P = 0
- The driver enable deassertion time equals
  – (1 + (DEDT x P)) x lpuart_ker_ck_pres, if P # 0
  – (1 + DEDT) x lpuart_ker_ck_pres, if P = 0

where P = BRR[20:11]

40.4.15 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.

LPUART 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 bitfield 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.

LPUART 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 wake-up time parameter (refer to the device datasheet) and on the LPUART receiver tolerance (see *Section 40.4.9: 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 391: 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 wake-up 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 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 wake up correctly from low-power mode is: 1/11.32 µs = 88.36 kbauds.

40.5 LPUART in low-power modes

Table 393. 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 and SPI master modes are functional. DMA requests are functional, and the interrupts cause the device to exit Stop mode.
Standby	The peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 40.3: LPUART implementation* to know if the wake-up 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.

40.6 LPUART interrupts

Refer to *Table 394* for a detailed description of all LPUART interrupt requests.

Table 394. 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/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)	
	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 wake-up from Stop mode feature. Refer to *Section 40.3: 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.

40.7 LPUART registers

Refer to *Section 1.2: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

40.7.1 LPUART control register 1 (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

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 bits

M[1:0] = 01: 1 Start bit, 9 Data bits, n stop bits

M[1:0] = '10: 1 Start bit, 7 Data bits, n stop bits

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 39.5.21: 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 40.4.14: 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 wake-up method

This bit determines the LPUART wake-up 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: When the LPUART acts as a transmitter, 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 wake-up from low-power mode, this bit is reserved and must be kept at reset value. Refer to Section 40.3: LPUART implementation.*

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.

40.7.2 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, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFOEN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
		rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 29 **FIFOEN:**FIFO mode enable

This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 Start bit, 8 Data bits, n stop bits

M[1:0] = 01: 1 Start bit, 9 Data bits, n stop bits

M[1:0] = '10: 1 Start bit, 7 Data bits, n stop bits

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 39.5.21: 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 40.4.14: 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 wake-up method

This bit determines the LPUART wake-up 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: When the LPUART acts as a transmitted, 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 wake-up from low-power mode, this bit is reserved and must be kept at reset value. Refer to Section 40.3: LPUART implementation.

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.

40.7.3 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 wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter must 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 LPUART is disabled (UE = 0).

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

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.

1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)

1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)

1: TX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).

This enables the use of an external inverter on the TX line.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.

0: RX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0/mark)

1: RX pin signal values are inverted. ((V_{DD} = 0/mark, Gnd = 1 / idle).

This enables the use of an external inverter on the RX line.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.

0: TX/RX pins are used as defined in standard pinout

1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **STOP[1:0]**: Stop bits
These bits are used for programming the stop bits.
00: 1 stop bit
01: Reserved.
10: 2 stop bits
11: Reserved
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 **ADDM7**:7-bit Address Detection/4-bit Address Detection
This bit is for selection between 4-bit address detection or 7-bit address detection.
0: 4-bit address detection
1: 7-bit address detection (in 8-bit data mode)
This bit can only be written when the LPUART is disabled (UE = 0)

Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bits 3:0 Reserved, must be kept at reset value.

40.7.4 LPUART control register 3 (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode enabled, FIFOEN = 1

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.
Others: Reserved, must not be used.
This bit can only be written when the LPUART is disabled (UE = 0).

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[2:0].

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.
Others: Reserved, must not be used.
This bit can only be written when the LPUART is disabled (UE = 0).

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[2:0].

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 completes 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.

40.7.5 LPUART control register 3 [alternate] (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode disabled, FIFOEN = 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**
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 31:23 Reserved, must be kept at reset value.

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 completes 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.

40.7.6 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 must be taken when generating high baud rates using high lpuart_ker_ck_pres values. lpuart_ker_ck_pres must be in the range [3 x baud rate..4096 x baud rate].

40.7.7 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 must wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 Reserved, must be kept at reset value.

40.7.8 LPUART interrupt and status register (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x0080 00C0

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

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 number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of LPUART_CR3 register.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 the RXFTCFG[2:0] bitfield of the LPUART_CR3 register, that is, the Receive FIFO contains RXFTCFG[2:0] data. An interrupt is generated if the RXFTIE bit = 1 (bit 28) in the LPUART_CR3 register.

0: Receive FIFO does not reach the programmed threshold.

1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG[2:0] threshold is configured to 101, the RXFT flag is set if RXFIFO size data are available, that is, (RXFIFO size – 1) data in the RXFIFO and 1 data in the LPUART_RDR. Consequently, the (RXFIFO size + 1) th received data does not cause an overrun error. The overrun error occurs after receiving the (RXFIFO size + 2) th data.

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 wake-up 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 wake-up from mute mode

This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.

When wake-up 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 wake-up 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 ongoing

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 must 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 has completed 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 LPUART_CR1 register.

The TC bit is cleared by software, by writing 1 to the TCCF of 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 (that is, 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.

40.7.9 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, FIFOEN = 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.	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 wake-up 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 wake-up from mute mode

This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.

When wake-up 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 wake-up 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 ongoing

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 full

1: Data register empty

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 is set when the transmission of a frame containing data has completed 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 LPUART_ICR register or by writing to the LPUART_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 (that is, 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.

40.7.10 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.

40.7.11 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 *Section 40.4.1: LPUART block diagram*).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

40.7.12 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 *Section 40.4.1: LPUART block diagram*).

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.

40.7.13 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
Others: Reserved, must not be used.

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is equal to 1011, that is, input clock divided by 256.

If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to Section 40.3: LPUART implementation.

40.7.14 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 *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.

40.7.15 LPUART register map

Table 395. 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_CR1FIFO 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

Table 395. 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
0x00	LPUART_CR1FIFO 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 FIFO mode enabled	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
0x08	LPUART_CR3 FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
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	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_ISRFIFO 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
0x1C	LPUART_ISRFIFO 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

Table 395. 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
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: Memory organization* for the register boundary addresses.

41 Serial peripheral interface (SPI)

41.1 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 can operate in multislave or multimaster configurations. The device configured as master provides a communication clock (SCK) to the slave device. The slave select (NSS) and ready (RDY) signals can be applied optionally just to set up communication with a concrete slave and to ensure it handles the data flow properly. The Motorola data format is used by default, but some other specific modes are supported as well.

41.2 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-bit up to 32-bit data size selection
- Multimaster or multislave 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
- Slave select hardware or software management for both master and slave
- Adjustable minimum delays between data and between NSS and data flow
- Configurable NSS 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 transfer to control NSS and CRC
- Dedicated transmission and reception flags with interrupt capability
- SPI Motorola and TI format support
- Hardware CRC can verify the integrity of the communication at the end of a transfer 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 on the operating mode
- Two 8-bit width embedded Rx and Tx FIFOs (FIFO size depends on instances)
- Configurable FIFO thresholds (to handle the data packets)
- Capability to handle data streams by system DMA controller
- Configurable behavior at slave underrun condition (support of cascaded circular buffers)

- Autonomous functionality in Stop modes (handling of the transfer flow and required clock distribution) with wake-up from Stop capability
- Optional status pin RDY signalizing that the slave device is ready to handle the data flow

41.3 SPI implementation

The table below describes the SPI implementation.

Table 396. SPI features

SPI feature	SPI1[(1)] (full-featured set instance)	SPI3 (limited-featured 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 configurable from 9 to 17 bits
FIFO size	16x8 bits	8x8 bits
FIFO threshold	1 to 16 data	1 to 4 data
Number of data control	Up to 65535	Up to 1023, no data counter
I2S feature	No	No
Autonomous in Stop modes with wake-up capability	Yes	Yes
Autonomous in Standby mode	No	No

1. Available only on STM32WBA52/54/55xx devices.

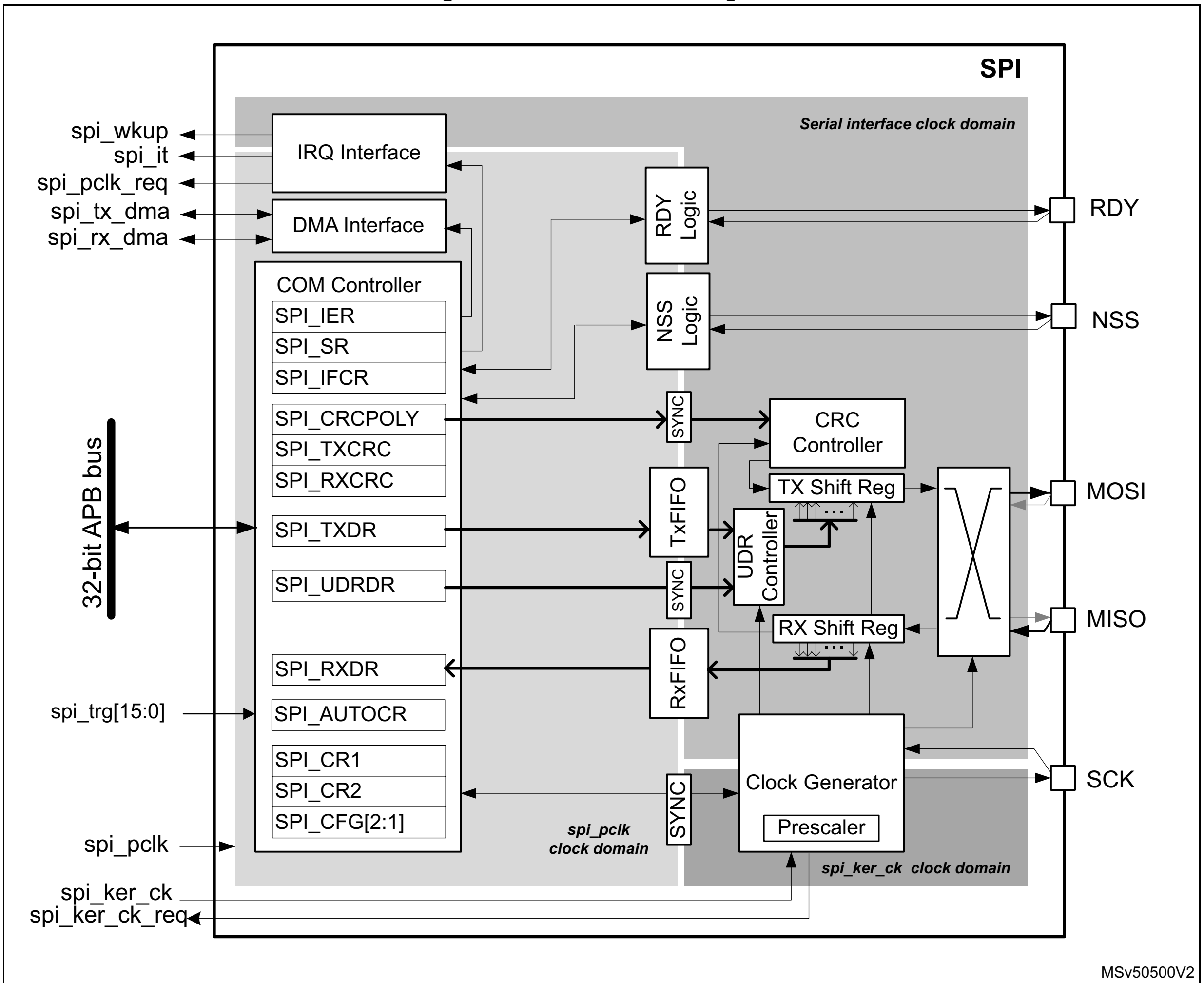
Note: For detailed information about instance capabilities to exit from Stop and Standby modes, refer to Table 402: SPI wake-up and interrupt requests and Table 89: Functionalities depending on the working mode.

41.4 SPI functional description

41.4.1 SPI block diagram

The SPI enables synchronous, serial communications 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 455*.

Figure 455. SPI block diagram



The simplified scheme of *Figure 455* 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. That 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 set up correctly (for example when underrun or overrun is evaluated, see *Section 41.5.2* for details). This cannot be done when the bus becomes idle. In some specific cases, the slave even requires the clock generator working (see *Section 41.5.1*).

When the SPI works as a master, the RCC must provide the **spi_ker_ck** kernel clock to the peripheral during communication to feed the serial interface clock via the clock generator where it can be divided by prescaler or bypassed. The signal is then provided to the slaves via the SCK pin and internally to the serial interface domain of the master.

41.4.2 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.
- **NSS:** slave select pin. Depending on the SPI and NSS settings, this pin can be used to either:
  - Select an individual slave device for communication
  - Synchronize the data frame, or
  - Detect a conflict between multiple masters

  See *Section 41.4.7* for details.
- **RDY:** optional status pin signaling slave FIFO occupancies and so the slave availability to continue the transfer without any risk of data flow corruption. It can be checked by the master to control the 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 397* and *Table 398* for the list of SPI input / output pins and internal signals.

Table 397. 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
NSS	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 398. 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 wake-up 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 399. SPI interconnection (SPI1)

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(1)
spi_trg9	comp2_out(1)
spi_trg10	rtc_alra_trg
spi_trg11	rtc_wut_trg
spi_trg12...15	Reserved

1. Only available for STM32WBA54xx and STM32WBA55xx devices.

Table 400. SPI interconnection (SPI3)

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	exti8
spi_trg6	lptim1_ch1
spi_trg7	Reserved
spi_trg8	comp1_out(1)
spi_trg9	Reserved
spi_trg10	rtc_alra_trg
spi_trg11	rtc_wut_trg
spi_trg12...15	Reserved

1. Only available for STM32WBA54xx and STM32WBA55xx devices.

41.4.3 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 two or three wires (with software slave select management) or three or four wires (with hardware slave select 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 slaves for communication by NSS line when it is managed by hardware.

The data between the master and the slave flow synchronously on the MOSI and/or MISO lines.

41.4.4 Communications between one master and one slave

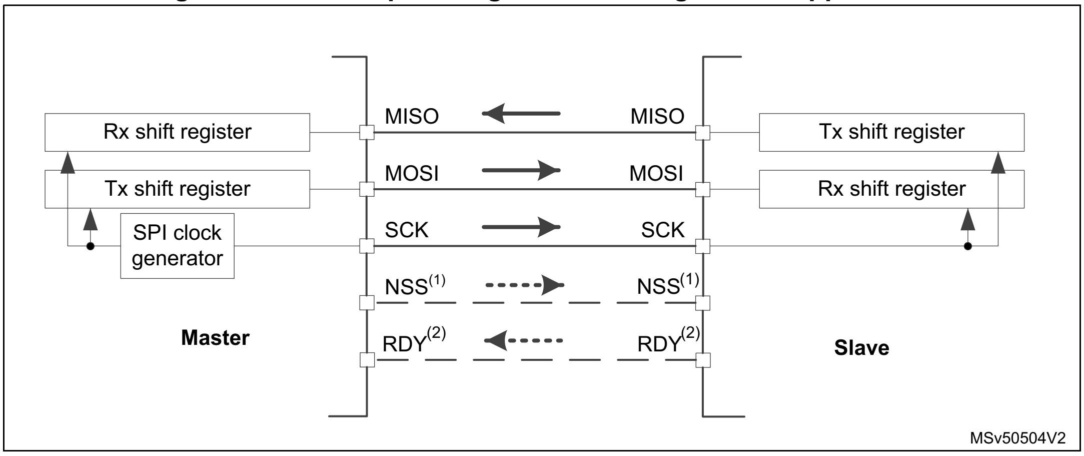
The communication flow can use one of three possible modes: the full-duplex (three wires) mode, half-duplex (two wires) mode, or the simplex (two wires) mode. The NSS signal is optional in single master-slave configuration and is often not connected between the two communication nodes. Nevertheless, the NSS 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 ensure the correct management of all the transferred 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 456. Full-duplex single master/ single slave application

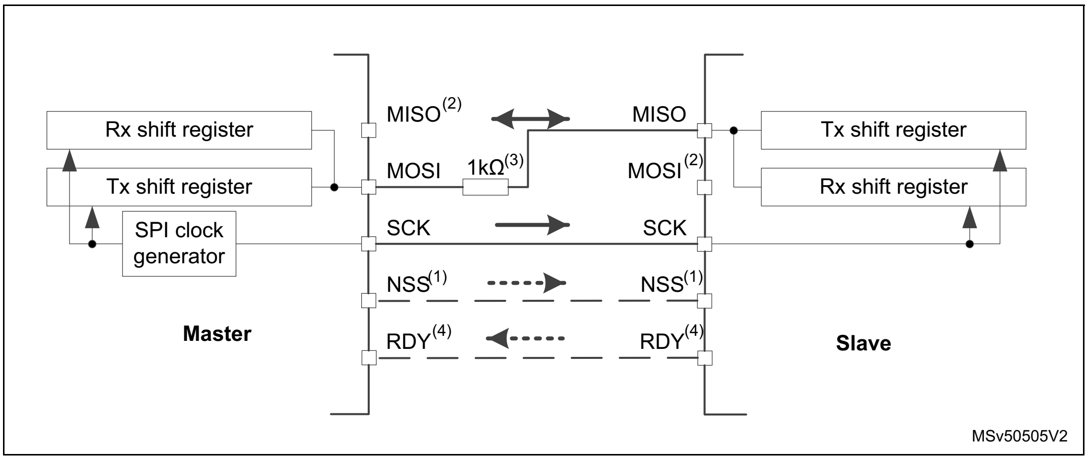


1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 41.4.7*).
2. The RDY signal provided by the slave can be read by the master optionally.

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 the 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 457. Half-duplex single master/ single slave application



1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to

operate in a single master-slave pair (see Section 41.4.7).

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 temporarily 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. The RDY signal provided by the slave can be read by the 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 pin 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 NSS pin (or software managed internal signal) is active (see Section 41.4.7).

- **Receive-only mode: COMM[1:0] = 10**

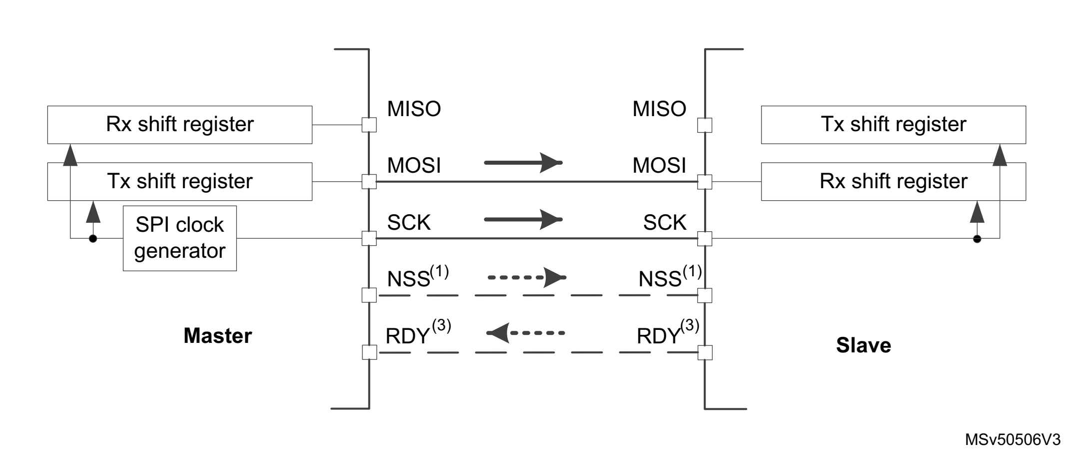
  In master mode, the MOSI output is disabled and can be used as a GPIO. The clock signal is generated continuously as long as the SPI is enabled and the CSTART bit in the SPI_CR1 register is set. The clock is stopped either by software explicitly requesting this by setting the CSUSP bit in the SPI_CR1 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 41.4.7).

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 can 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 transfer 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 458. Simplex single master / single slave application (master in transmit-only / slave in receive-only mode)



1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 41.4.7*).
2. In this configuration, both the MISO pins can be used as GPIOs.
3. The RDY signal provided by the slave can be read by the master optionally.

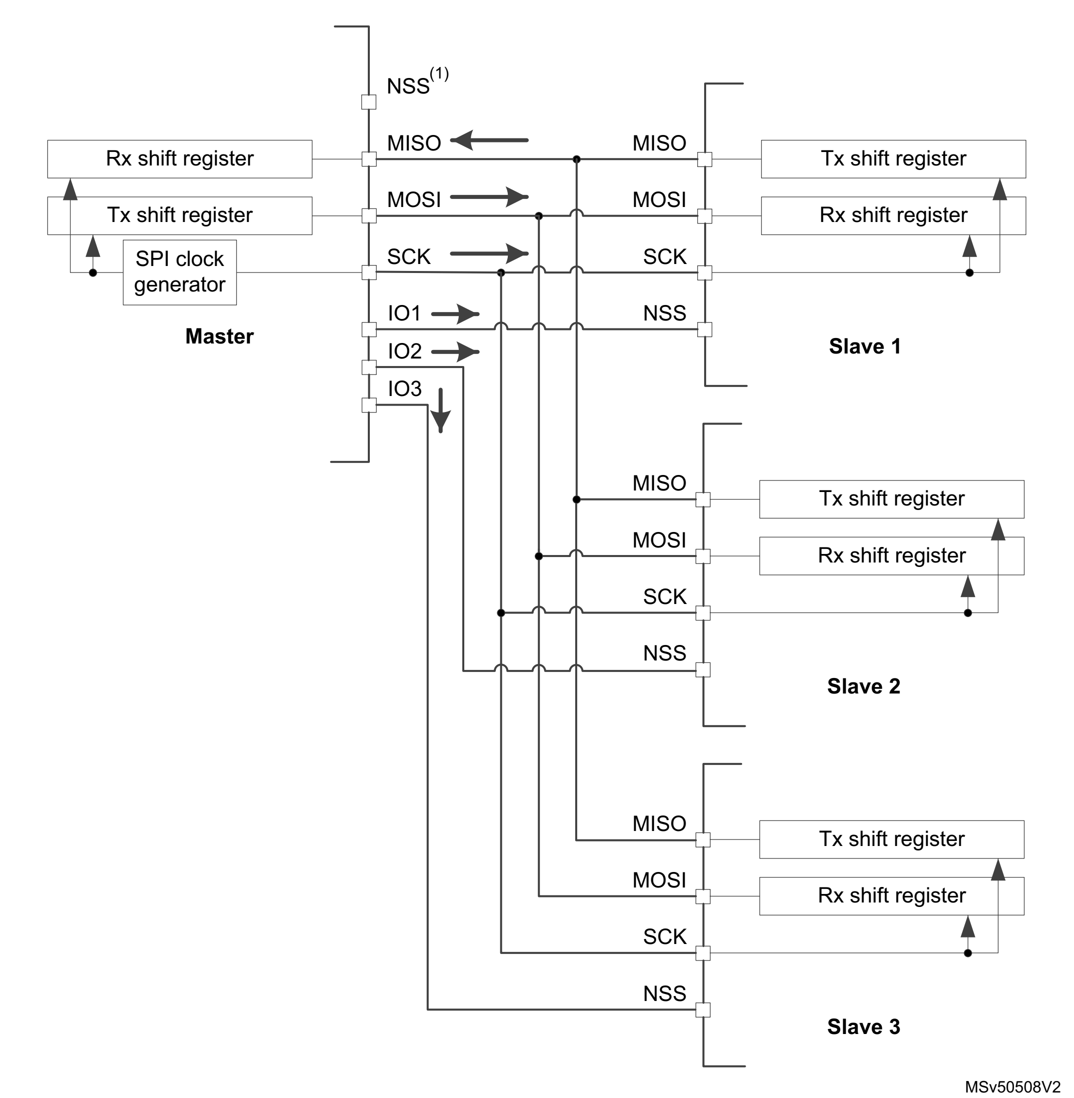
41.4.5 Standard multislave communication

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 459.*).

The master must select one of the slaves individually by pulling low the GPIO connected to the slave NSS input (only one slave can control data on a common MISO line at a given time).

When this is done, a communication between the master and the selected slave is established. In addition to 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 a 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 459. Master and three independent slaves connected in star topology

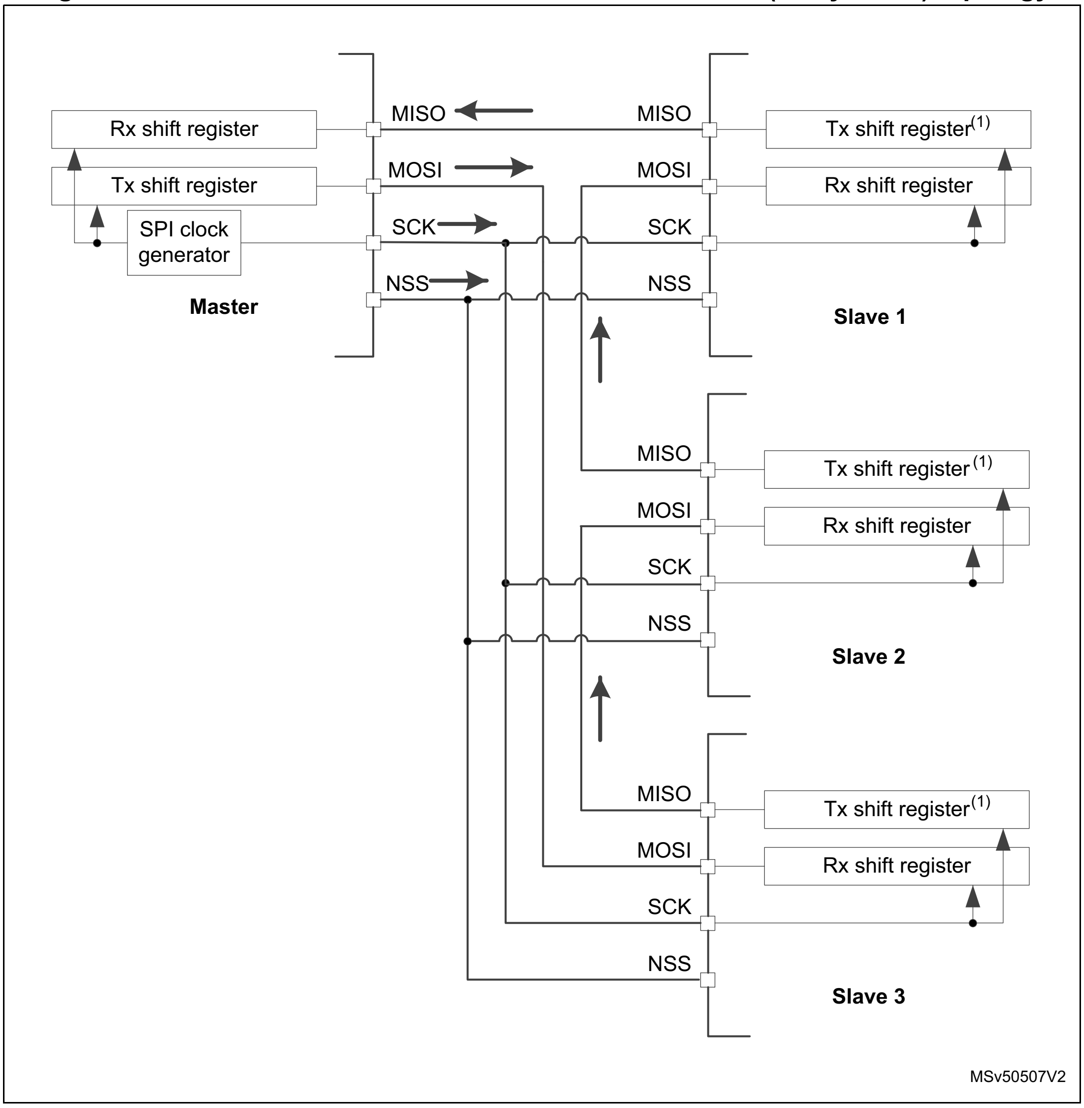


1. The master single NSS pin hardware output functionality cannot support this topology (to be replaced by a set of GPIOs under software control). The NSS pin is free for other application uses (such as GPIO or other alternate functions). Refer to *Section 41.4.7* for details.
2. If the application cannot ensure that no more than a single NSS active signal is provided by the master at a given time, it is better to configure the MISO pins in an open-drain configuration with an external pull-up on the MISO line to prevent conflicts between the interconnected outputs of the slaves. Else, a push-pull configuration can be applied without an extra resistor (see I/O alternate function input/output (GPIO) section).
3. The RDY signals can be read by the master from the slaves optionally.

The master can handle the SPI communication with all the slaves in time when a circular topology is applied (see *Figure 460*). All the slaves behave like simple shift registers applied in serial chain under control of common slave select (NSS) and clock (SCK) signals. All the information is shifted simultaneously around the circle while returning back to the master. Sessions have fixed the length where the number of data frames transferred by the master is equal to the number of slaves.
Then when a first data frame is transferred in the chain, the master just sends the information dedicated for the last slave node in the chain, via the first slave node input, while the first information received by the master comes from the last node output at this time.

Correspondingly, the last transferred data finishing the session is dedicated for the first slave node while its first outgoing data just reaches the master input, after its circling around the chain passing through all the other slaves during the session.
The data format configuration and clock setting must be the same for all the nodes in the chain in this topology. As the receive and transmit shift registers are separated internally, a trick with intentional underrun must be applied to the TxFIFO slaves when the information is transferred between the receiver and the transmitter by hardware.
In this case, the transmission underrun feature is configured in a mode repeating the last received data frame (UDRCFG=1). A session can start optionally with a single data pattern written into the TxFIFO by each slave (usually slave status information is applied) before the session starts. In this case, the underrun happens in fact after this first data frame is transferred. To be able to clear the internal underrun condition immediately and restart the session by the TxFIFO content again, the user must disable and enable the SPI between the sessions and must fill the TxFIFO by a new single data pattern (to overcome the propagating delay of the clearing raised in case the underrun is cleared in a standard way by the UDRC bit).

Figure 460. Master and three slaves connected in circular (daisy chain) topology



1. The underrun feature is used by the slaves in this configuration when the slaves are able to transmit data received previously into the Rx shift register once their TxFIFOs become empty.
2. The RDY signals can be read by the master optionally, either separately or configured as open drain outputs (while RDIOP = 0) and connected with a pull-up resistor as a common chain ready status overdriven by the slowest device.

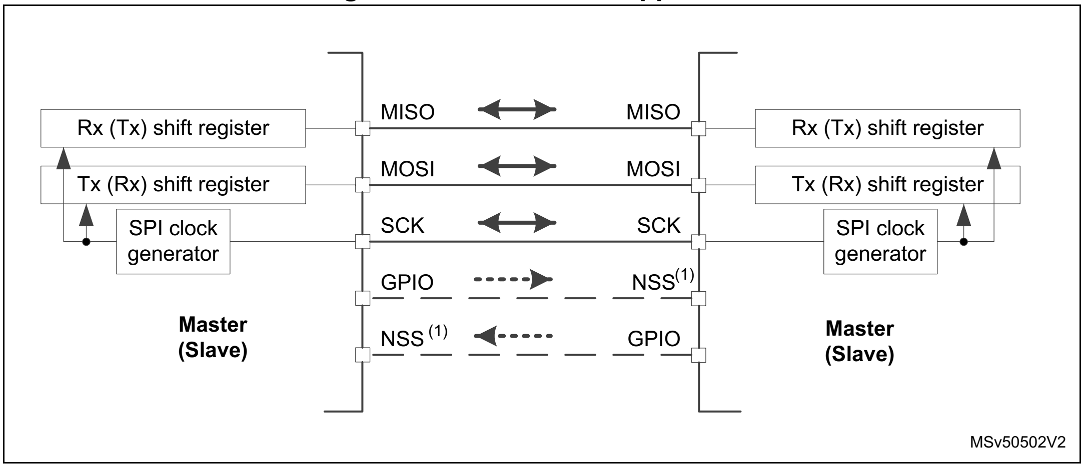
41.4.6 Multimaster communication

Unless the SPI bus is not designed primarily for a multimaster capability, the user can 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 NSS pin is 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. Once a node wants to overtake control on the bus, it switches itself into master mode and applies active level on the slave select input of the other node via the dedicated GPIO pin. After the session is complete, the active slave select signal is released and the node mastering the bus temporarily returns back to passive slave mode waiting for the next session to 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 (for example postpone the next attempt by different predefined timeouts applied to both nodes).

Figure 461. Multimaster application



1. The NSS pin is configured at hardware input mode at both nodes. Its active level enables the MISO line output control as the passive node is configured as a slave.
2. The RDY signal is not used in this communication.

41.4.7 Slave select (NSS pin) management

In slave mode, the NSS works as a standard ‘chip select’ input and lets the slave communicate with the master. In master mode, the NSS can be used either as an output or an input. As an input it can prevent a multimaster bus collision, and as an output it can drive a slave select signal of a single slave. The NSS signal can be managed internally (software management of the NSS input) or externally when both the NSS input and output are associated with the NSS pin (hardware slave select management). The user can configure which level of this input/output external signal (present on the NSS pin) is considered as active by the SSIOP bit setting. NSS 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 slave select management (SSM = 1)**: in this configuration, slave select information is driven internally by the SSI bit value in the SPI_CR1 register. The external NSS pin is free for other application uses (such as GPIO or other alternate functions).
- **Hardware slave select management (SSM = 0)**: in this case, there are two possible configurations. The configuration used depends on the NSS output configuration (SSOE bit in the SPI_CFG2 register).
  - **NSS output enable (SSOE = 1)**: this configuration is only used when the MCU is set as master. The NSS pin is managed by the hardware. The functionality is tied to CSTART and EOT control. As a consequence, the master must apply the proper TSIZE > 0 setting to control the NSS output correctly. Even if SPI AF is not applied at the NSS pin (it can be used as a standard GPIO then), keep anyway SSOE = 1 to ensure the default NSS input level and prevent any mode fault

evaluation at the input of the master NSS internal logic applicable at a multimaster topology exclusively.

a) When SSOM = 0 and SP = 000, the NSS 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 NSS is pulsed inactive between data frames, and kept inactive for a number of SPI clock periods defined by the MIDI value decremented by one (from 1 to 14).

d) NSS input is forced to nonactive state internally at master to prevent any mode fault.

– **NSS output disable (SSM = 0, SSOE = 0)**:

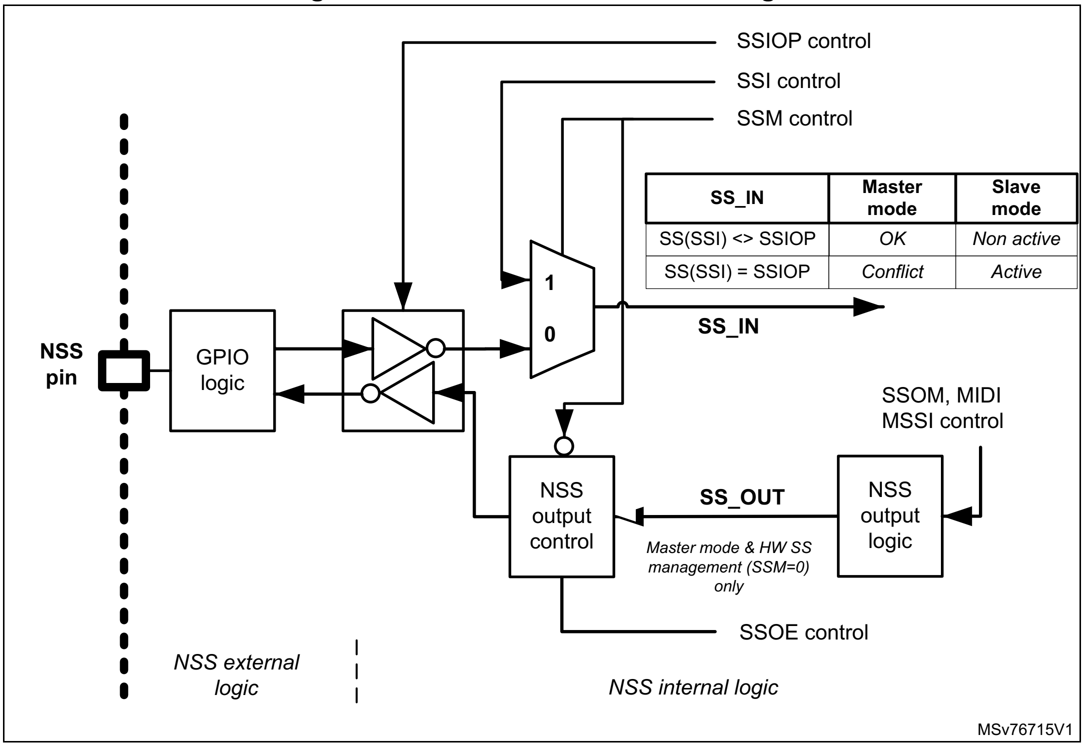
a) If the microcontroller is acting as the master on the bus, this configuration allows multimaster capability. If the NSS pin is pulled into an active level in this mode, the SPI enters master mode fault state and the SPI device is automatically reconfigured in slave mode (MASTER = 0).

b) In slave mode, the NSS pin works as a standard ‘chip select’ input and the slave is selected while the NSS line is at its active level.

Note: The purpose of automatic switching into slave mode when a mode fault occurs is to avoid the possible conflicts on data and clock lines. As the SPE is automatically reset in this condition, both Rx and Tx FIFOs are flushed and current data is lost.

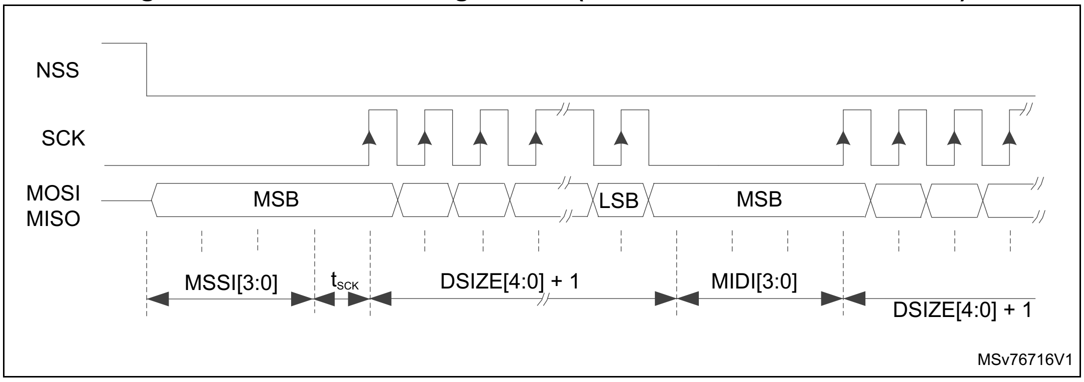
When the SPI slave is enabled in the hardware slave select management mode, all the transfers are ignored even in case of the NSS is found at active level. They are ignored until the slave detects a start of the NSS signal (transition from nonactive to active level) just synchronizing the slave with the master. This is because the hardware management mode cannot be used when the external NSS pin is fixed. There is no such protection in the slave select software management. Then the SSI bit must be changed when there is no traffic on the bus and the SCK signal is in idle state level between transfers exclusively in this case.

Figure 462. Scheme of NSS control logic



When the hardware output NSS control is applied (SSM = 0, SSOE = 1), by configuration of the MIDI[3:0] and MSSI[3:0] bitfields, the user can control the timing of the NSS signal between data frames and can insert an extra delay at the beginning of every transfer (to separate the NSS 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 463*).

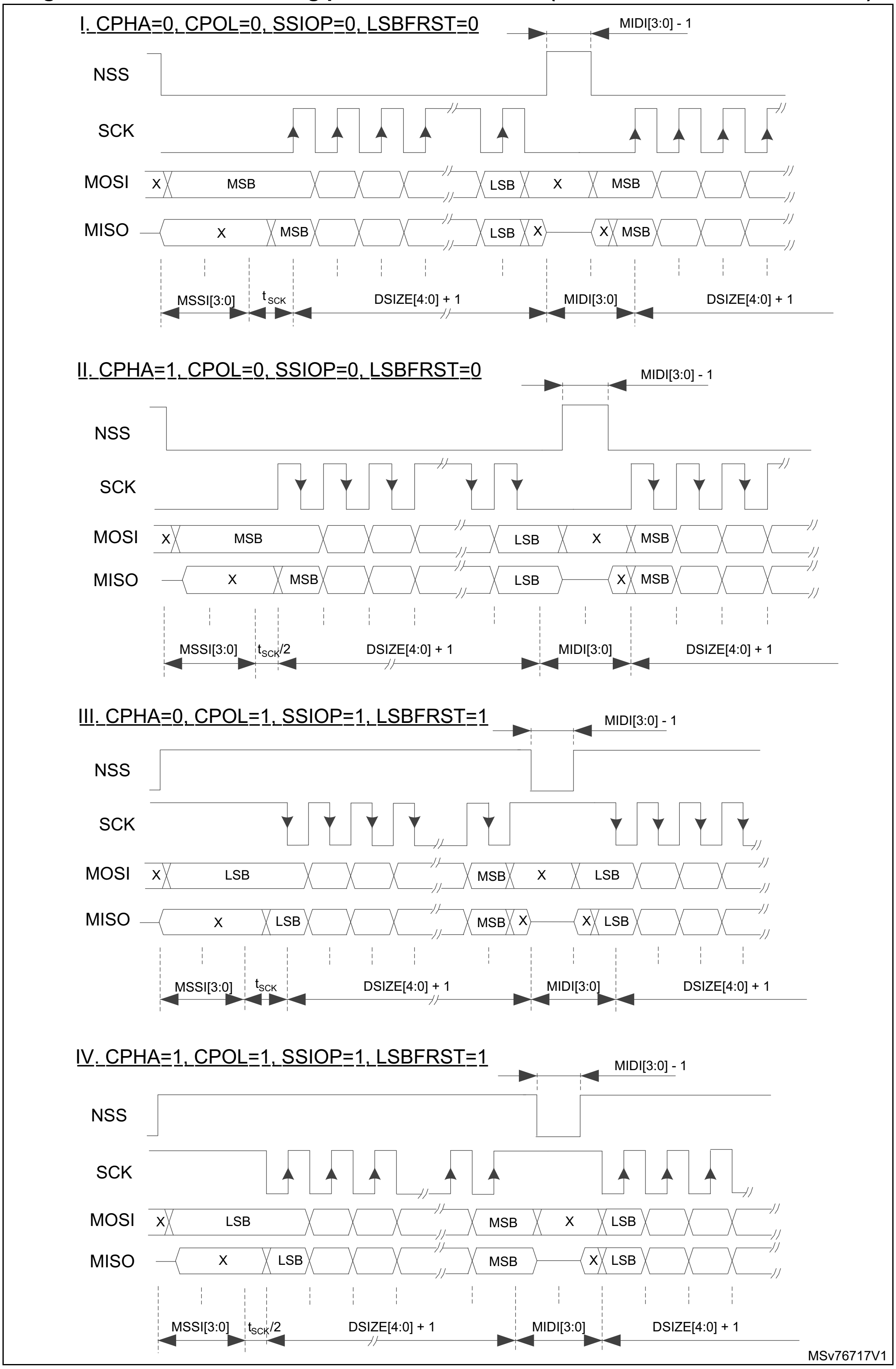
Figure 463. Data flow timing control (SSOE = 1, SSOM = 0, SSM = 0)



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, setting the SSOM bit invokes a specific mode, which interleaves pulses between data frames if there is a sufficient space to provide them (MIDI[3:0] must be set greater than one SPI period). Some configuration examples are shown in *Figure 464*.

Figure 464. NSS interleaving pulses between data (SSOE = 1, SSOM = 1, SSM = 0)



1. MSSI[3:0] = 0010, MIDI[3:0] = 0010.
2. NSS interleaves between data when MIDI[3:0] > 1 wide of the interleaving pulse is always one SCK period less than the gap provided between the frames (defined by the MIDI parameter). If MIDI is set, the frames are separated by a single SCK period but no interleaving pulse appears on NSS.

41.4.8 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 transfer. The reason can be that the slave 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, it is highly probable that the transfer fails.

The logic to control the RDY output is rather complex, tied closely with the 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 transfer. This pessimistic logic is suppressed at the end of a transfer 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 the master in time that it is necessary to suspend the transfer temporarily until the next transferred data can be processed safely again. When the RDY signal input is enabled at master side, the master suspends the communication once the slave indicates not ready status. This prevents the master to complete the transfer of an ongoing frame, which just empties the slave TxFIFO or full fills its RxFIFO until a next data is written and/or read there (despite the frame still can be completed without any constraint). It can make a problem if the TSIZE = 0 configuration is applied at slave because slave then never evaluates the end of the transfer (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 bits.

41.4.9 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 can 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 is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transferred (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 transferred (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.

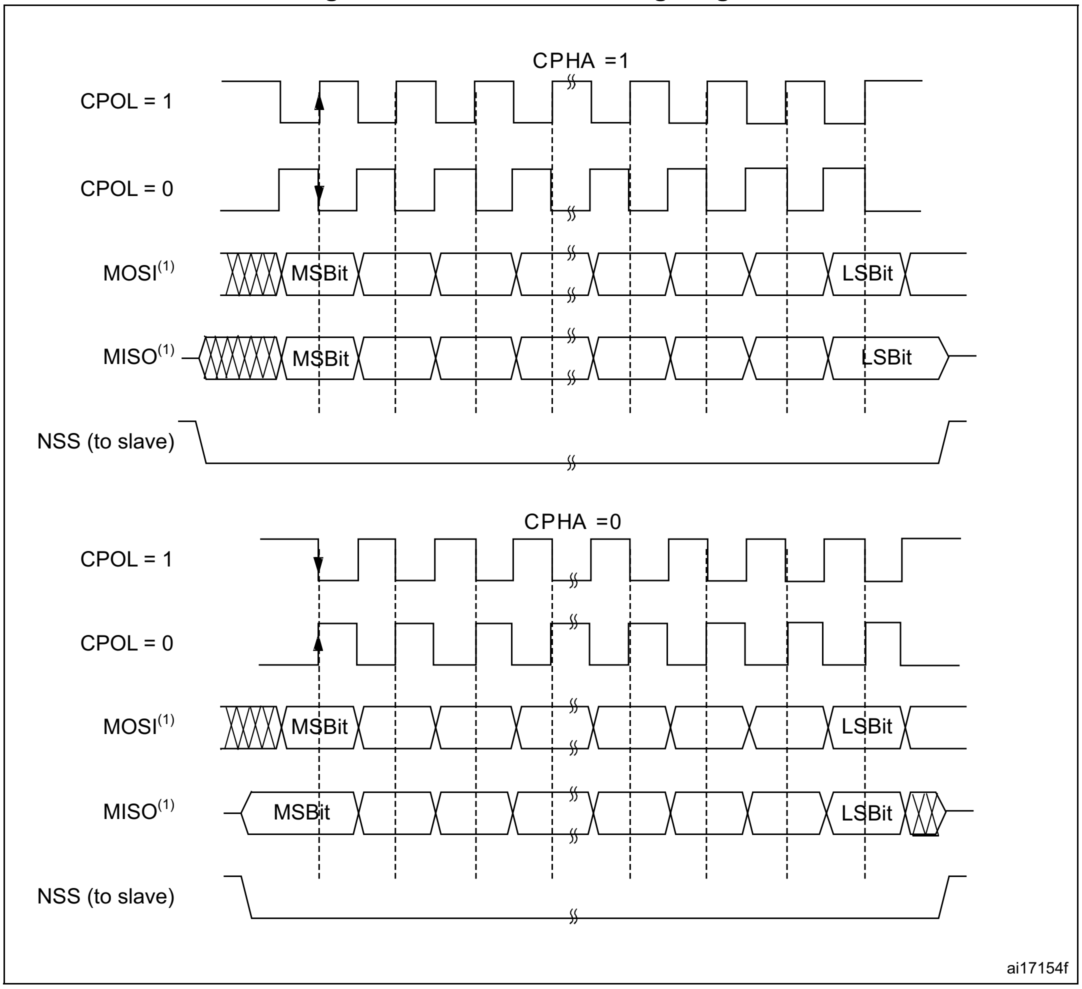
The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edges (dotted lines in *Figure 465*).

Figure 465 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 465. Data clock timing diagram



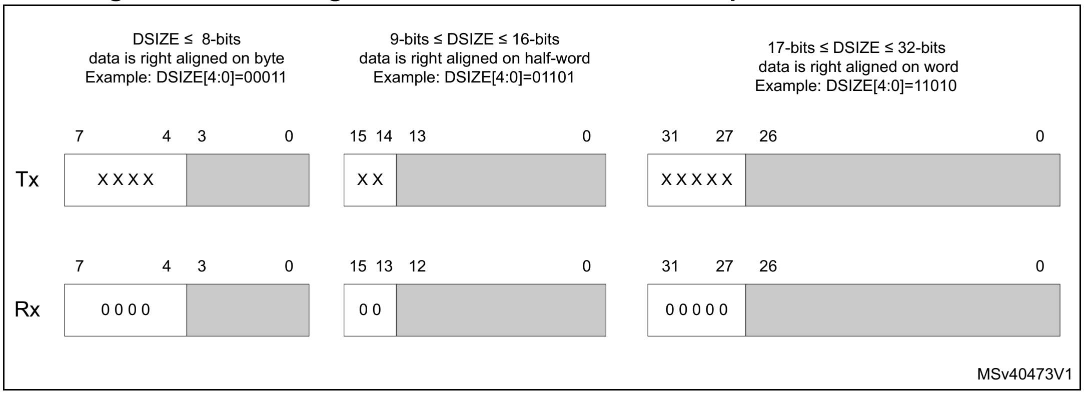
1. The order of data bits depends on the LSBFRST bit setting.

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 of the SPI_CFG2 register.

The data frame size is configured through the DSIZE bitfield of the SPI_CFG1 register to a range that depends on the SPI instance (see *Section 41.3: SPI implementation*). The setting applies to both transmission and reception. The data bits in the SPI_TXDR/SPI_RXDR registers are right-aligned with 8, 16, or 32 bits, depending on the data size (see *Figure 466*). These registers can consequently be accessed by bytes, half-words, or words. FIFO accesses smaller than a single data are forbidden. When the FIFO occupancy flag is raised, a FIFO access greater than the FIFO threshold can lead to spurious data being read or written data being lost. When the access to/from the SPI_TXDR/SPI_RXDR registers is a multiple of the configured data size, data packing is applied automatically, while the lowest significant bits/bytes are communicated first. For more details, see *Section 41.4.12: SPI data transmission and reception procedures*.

Figure 466. Data alignment when data size is not equal to 8, 16 or 32 bits



41.4.10 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, NSS, and RDY pins if applied.
2. Write into the SPI_CFG1 and SPI_CFG2 registers and set up the proper values of all ‘not reserved’ bits and bitfields, prior to enabling the SPI, 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 the slave is configured in TI mode, the MBR[2:0] setting is also considered.
   b) UDRCFG is taken into account in slave mode only.
   c) CRCSIZE[4:0] bitfield must be configured 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 of 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 the SPI_CRCPOLY and into the TCRCINI, RCRCINI, and CRC33_17 bits of 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 *Section 41.4.14: Communication using DMA (direct memory addressing)*).
6. Configure SSI, HDDIR, and MASRX in the SPI_CR1 register if required.
7. Program the IOLOCK bit in the SPI_CR1 register if the configuration protection is required (for safety).

41.4.11 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 traffic is ongoing on the bus, assuming that the NSS signal is managed by hardware at slave or kept inactive by the slave software when the software management of the NSS signal is applied (see *Section 41.4.7*). To prevent any risk of any data underrun, all the data to be sent have to be written to the slave transmitter data register 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 NSS, else the following transfer may be desynchronized.

When the SPI slave is enabled at the hardware slave select management mode, all the transfers are ignored even in case of the NSS is found at active level. They are ignored until the slave detects a start of the NSS signal (its transition from nonactive to active level) just synchronizing the slave with the master. That is why the hardware management mode cannot be used when the external NSS pin is fixed. There is no such protection at the slave select 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 41.4.14*.

41.4.12 SPI data transmission and reception procedures

The SPI can transfer data at a very high communication speed. Even though the data is FIFO buffered, handling a continuous data flow by servicing data frames individually leads to an enormous CPU or DMA load, in particular when the data size is small. This potentially leads to a significant limitation of the overall system performance, as well as communication errors such as data overrun or underrun that may be raised when the system latencies in servicing requests become comparable to single data frame transfer time.

The SPI offers advanced hardware control features to prevent these issues from occurring:

- Decreasing the number of events requiring service by collecting the data frames into larger **data packets**

  The number of data frames per packet (FIFO threshold) can be configured. The complete data packet generates a FIFO occupancy event, which is then handled within a single, compact, service.
- Decreasing the number of CPU execution cycles required for the service, by applying the appropriate **data access**:
  - The widest access to the SPI data registers must be applied. This allows cumulative data to be handled by a single register read or write operation (data packing).
  - The number of read/write accesses necessary to complete the data packet must be aligned with the data frame size and the FIFO threshold.
- **Concurrent read and write services** to handle full-duplex data flow based on common FIFO occupancy events.
- Embedded **hardware data counters** to define the exact number of data involved in transfers.

Note: Using these features is optional. Nothing prevents the application from handling the data flow frame by frame.

The following sections give more details and describe specific cases for using these advanced hardware control features to handle data transfers.

Data packet control

The data frame size (number of bits in the frame) is defined through the DSIZE bitfield of the SPI_CFG1 register. The number of data frames per packet can be configured by selecting the FIFO threshold through the FTHLV bitfield of the SPI_CFG1 register. If the threshold value is set to zero, each completed data frame raises the FIFO occupancy event. The data packet occupancy must not exceed half of the FIFO size, which depends on the SPI instance of the product. The FIFO capacity (multiple of 8 bits) is consumed according to the following rules, depending on the data frame size:

- Data frames from 4 to 8 bits occupy one byte of the FIFO.
- Data frames from 9 to 16 bits occupy two bytes of the FIFO.
- Data frames from 17 to 24 bits occupy three bytes of the FIFO.
- Data frames from 25 to 32 bits occupy four bytes of the FIFO.

For example, if the FIFO capacity is 16 bytes, it accommodates up to five 24-bit frames. In this case, the data packet configuration must not exceed two data frames.

The SPI features two separate FIFOs to handle the reception and transmission data flow, the RxFIFO, and the TxFIFO. Their content can be handled by monitoring the occupancy flags RXP, TXP, and DXP of the SPI_SR register according to the SPI mode (duplex or simplex):

- If RXP is set, at least one complete data packet can be read from the RxFIFO.
- If TXP is set, at least one complete data packet can be written to the TxFIFO.
- If DXP is set, at least one complete data packet can be read from the RxFIFO and written to the TxFIFO at full-duplex mode.

These flags can be polled by software, or they can trigger an interrupt and/or a DMA request (if enabled through the RXPIE, TXPIE, and DXPIE bits of the SPI_EIR register or the RXDMAEN and TXDMAEN bits of the SPI_CFG1 register).

Once an occupancy flag is set, the software or the DMA must read and/or write a complete data packet before checking the flag again to verify if the next packet can be handled. This cycle can be repeated until the corresponding flag is read at zero.
Both RxFIFO and TxFIFO contents are flushed and cannot be accessed when the SPI is disabled (SPE cleared in the SPI_CR1 register).

Data packing versus data register access control

The content of the RxFIFO and TxFIFO can be accessed by reading/writing from/to the SPI data registers SPI_RXDR and SPI_TXDR, respectively. A read access from the SPI_RXDR register returns the oldest values stored in the RxFIFO that have not been read yet. A write access to the SPI_TXDR register stores the data written at the end of the send queue in the TxFIFO.

These data registers can be accessed by 8-, 16-, or 32-bit read and write CPU instructions, forced by the register address casting applied by the software. Data packing is performed automatically by hardware if the data access applied by the software is a multiple of the data size. It allows handling more than one data in parallel in a single data register access. Then the SPI operates using the lowest significant byte or half-word first. FIFO data accesses of less than the configured data size are forbidden. To avoid spurious data being read or written data being lost, a complete data packet (configured through the FIFO threshold) must be serviced when the corresponding FIFO occupancy flag is set. If the data pattern is not byte-, half-word, or word-aligned, only the valid data bits are stored right-aligned to the FIFO and transferred on the bus. Unused bits are discarded on the transmitter side and padded with zeros on the receiver side.

For example, if the data frame size fits into one byte, the data packing is used automatically when a 16-bit or 32-bit read/write access is performed by software from/to the SPI_RXDR/SPI_TXDR register. In this case, two respectively four data are handled by a single 16-bit respectively 32-bit access. Additionally, if such data frames are grouped into packets of four via the FIFO threshold setting, the packet to be serviced is signalized by raising the corresponding FIFO occupancy flag (TXP, RXP, or DXP). If the instance FIFO threshold is sufficient, a packet containing eight 8-bit data frames can be handled by two consecutive 32-bit accesses upon the same single threshold event. The most efficient data handling is achieved when the data packet size (defined by DSIZE and FTHLV bitfields) is aligned with the read/write access from/to the data registers.

Concurrent read and write services

In full-duplex mode, both TxFIFO and RxFIFO packet occupancies can be monitored through a common FIFO flag (DXP). When the DXP flag is set, the application performs the specified number of writes to SPI_TXDR to upload the content of one entire data packet for transmission, followed by the same number of reads from SPI_RXDR to download the content of one received data packet. Once one data packet is uploaded and one packet is downloaded, the application software or the DMA checks again the DXP flag and repeats the data packet read and write operation sequence until DXP is read as zero.

The drawback of services based on DXP exclusively is that servicing the TxFIFO is delayed on purpose due to the SPI nature since the TXP events precedes the RXP ones. To allow continuous SCK clock flow on the master side and prevent underrun on the slave side, it is recommended to prefill a few data ahead to the TxFIFO at the transfer start, and wait until the transfer is complete to read the last received data.

Hardware data counter

If a hardware data counter is used (TSIZE bitfield of the SPI_CR2 register set to a nonzero value), the application software does not need to calculate the remaining number of data to be handled. The end of transfer automatically controls the CRC, as well as the hardware slave select management, when used. The user application does not need to ensure that the overall number of data is a multiple of the packet size. If the last data packet is incomplete, it is serviced in the same way as any full packet. The unused part of this last packet is not handled by the peripheral. Only the valid data are written into the TxFIFO and/or read from the RxFIFO, and the redundant writes and reads are discarded. When the hardware counter reaches zero, the EOT flag is raised, and the RXP flag is not set for the last data packet. If the last packet is not aligned with the packet size, the TXP and EOT occupancy events are not related to the configured packet size but to the number of remaining data calculated by hardware.

If TSIZE is kept at zero (for example due to an unpredictable number of data), only the number of transferred data corresponding to the FIFO thresholds is supported. If some data are not aligned with the configured packet size, they may remain pending and available in the RxFIFO. In this case, the FTHLV level is not reached and the RXP flag is not set. Then the number of remaining received data frames in the RxFIFO is indicated by the RXWNE and RXPLVL bitfields of the SPI_SR register. Nevertheless, the application software can still read the complete data packet from the RxFIFO and the redundant data are read as zero. To prevent such an unaligned data reception, the user must configure the FIFO threshold to a single data (FTHLV = 0). In this case, each data frame is serviced by its own RXP occupancy event.

Data transfer handling

Data are transferred using MOSI and MISO lines, depending on the SPI communication mode and associated configuration. The mode is configured through the COMM[1:0] bitfield of the SPI_CFG2 register. The HDDIR bit of the SPI_CR1 register controls the data flow direction in half-duplex mode. The communication flow is handled by the master via the NSS and SCK signals. The slave selected for the communication is fully subordinated to the master communication activity, no matter if it handles the data flow on time or not. The slave can only temporarily suspend the master communication by using the RDY signal. The active levels of RDY and NSS signals can be changed, or MISO and MOSI functionality swapped (via the RDIOP, SSIOP and IOSWP bits of the SPI_CFG2 register).

The SPI master transmitter can operate in full-duplex, simplex, or half-duplex mode. In full-duplex mode, data are received synchronously. The master starts the data transfer once the CSTART bit of the SPI_CR1 register is set, provided the SPI is enabled and the TxFIFO content is not empty. The master then provides the serial clock signal continuously on the SCK pin until:

- The total number of required data programmed in TSIZE is transferred, or
- The transfer is suspended.

An automatic temporary suspension of the master transmission occurs when:

- The slave does not assert the RDY signal, if it is used, or
- The TxFIFO becomes empty, or
- In full-duplex mode, the RxFIFO becomes full while the automatic suspension is enabled (MASRX set in the SPI_CR1 register).

When an automatic suspension occurs, the master stops providing the clock, and the transfer proceeds depending on the cause of the suspension, when:

- The slave asserts RDY.
- The master software or the DMA writes additional data to TxFIFO.
- In full-duplex mode, the master software or the DMA releases the RxFIFO to enable it to accommodate new data.

The SPI master can receive data only when it operates in simplex receiver or half-duplex receiver mode. In these modes, the master starts the data transfer when the SPI is enabled, and the transfer is released by setting the CSTART bit. The serial clock signal is then provided continuously on the SCK pin by the master until:

- The total number of required data programmed in TSIZE is received, or
- The transfer is suspended by the master.

An automatic temporary suspension of the reception occurs when:

- The slave does not assert the RDY signal, if it is used, or
- The RxFIFO becomes full while the automatic suspension is enabled (MASRX bit set in the SPI_CR1 register).

When an automatic suspension occurs, the master stops providing the clock and the transfer proceeds depending on the cause of the suspension, when:

- The slave asserts RDY.
- The master software or the DMA releases the RxFIFO to enable it in order to accommodate new data.

A preferable way to terminate a transfer is to program the TSIZE bitfield to generate an EOT event. Hardware NSS signal or CRC handshake can then be controlled. To restart the internal state machine properly, it is recommended to disable the SPI and enable it again when the transfer is complete, even if the SPI configuration is not changed.

A transfer can be suspended at any time by setting the CSUSP bit of the SPI_CR1 register, which clears the CSTART bit. This software suspension control ensures the completion of any ongoing data frame. To restart the internal state machine properly, the SPI must be disabled and enabled again before the next transfer starts.

Oppositely, a temporary automatic suspension controlled by hardware typically results in a frozen and incomplete data frame transfer. This depends on baud rate setting, but usually, due to internal synchronization delays, the SCK signal stops when a few bits from the next data frame are already transferred on the bus. Once the suspension is released, the data frame transmission completes by transferring the remaining bits. That is why this automatic suspension is not quite reliable when the data frame size is less than eight bits. When shorter data frames are used, to prevent any data loss and assure proper RDY and/or

MASRX operation, the user can interleave and so extend the transfer time by inserting additional dummy clock cycles so that the period for a single data frame is higher or equal to eight SCK duration. This is done by setting the MIDI[3:0] bitfield of the SPI_CFG2 register.

Caution: If the SPE bit is cleared in master mode while the transfer is ongoing without any suspension, the clock is stopped even if the current frame transmission is not complete, and the content of the FIFOs is flushed and lost.

41.4.13 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 transfers 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 transfers when it stops providing data for transmission. In this case, the clock stops after the last data transfer. 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, to stop the peripheral, the SPI communication must first be suspended, by setting the CSUSP bit.

The data received but not read remain stored in RxFIFO when the SPI is suspended.

After such a software suspension, SPI must always be 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 transfers, for example:

- When the master handles the NSS signal by a GPIO not related to SPI (for example at case of multislave star topology) and it has to provide a proper end-of-NSS pulse for the slave, or
- When transfers from DMA or FIFO are completed while the last data frame or CRC frame transfer 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 the end of the bus activity. When TSIZE = 0, the user has to check TXC, SUSP, or FIFO occupancy flags according to 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 the CRC frame is complete in this case. When a transmission is suspended the software has to wait until the 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 the 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 ongoing 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 bit of the SPI_CFG2 register. When active output control is applied (AFCNTR = 1) and SPI has just been disabled (SPE = 0), the enabled outputs associated with SPI control signals (like NSS and SCK at master and RDY at slave) can immediately toggle to inactive level (according to SSIOP and CPOL settings at master and RDIOP at slave respectively). Instead, the data line output (MOSI at master and MISO at slave) can immediately change its level depending on the actual TxFIFO content, with the effect of potentially making invalid and no more guaranteed the value of the latest transferred 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 transfer is complete.

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 a small software delay between EOT event occurrence and SPI disable action.

41.4.14 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 of the SPI_CFG1 register are set. Separate requests must be issued to the Tx and Rx buffers to fulfill the service of the defined packet.

- In transmission, a series of DMA requests is triggered each time TXP is set. The DMA then performs a series of writes to the SPI_TXDR register.
- In reception, a series of DMA requests is triggered each time RXP is set. The DMA then performs a series of reads from the SPI_RXDR register. When EOT is set at the end of the transfer and the last data packet is incomplete, then DMA request is activated automatically according to RXWNE and RXPLVL[1:0] setting to read the 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 in the SPI_SR register), the EOT (or TXC at case TSIZE = 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 the DMA Rx buffer in the RXDMAEN bit of 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 the DMA Tx buffer in the TXDMAEN bit of 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 requests 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 of 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.

The packing mode is enabled if the DMA channel PSIZE value is a multiple of the data size. Then the DMA automatically manages the sequences of write and read operations to/from the SPI data registers, based on FIFO occupancy flags, and depending on the FIFO threshold and data size configurations.

The DMA completes the transfer automatically according to the TSIZE field setting, whatever the data packing mode used, and even if the number of data to transfer is not a multiple of the DMA data size (16 bits or 32 bits) while the frame size is smaller.

Alternatively, the last data frames can be written by software, in the single/unpacked mode.

Configuring any DMA data access to less than the configured data size is forbidden. One complete data frame must be always accessed at minimum.

41.4.15 Autonomous mode

The SPI is capable of handling and initializing transfers autonomously. It requires no specific system execution interaction until the ongoing transfer ends.

Such autonomous transfers can be handled not only in Run or Sleep mode 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 until the flag with enabled interrupt is 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 the 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 the 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 the slave never initializes a transfer, there is no need to synchronize any transfer start in this mode.

Note: The peripheral clock request stays pending until the 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 to achieve the lowest power consumption, and the application must acknowledge all pending interrupt 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 transfer is complete. The SCK signal is derived from the SPI clock generator running within the kernel clock domain fed from the RCC upon kernel clock request provided by the SPI when the device is in Stop mode. Temporal requests for the APB clock are then based upon specific traffic events, depending on the SPI configuration. The SPI master always initializes a transfer.

To minimize consumption in Stop mode, it is suggested to combine communication starts triggered by hardware (the TRIGEN bit is 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.

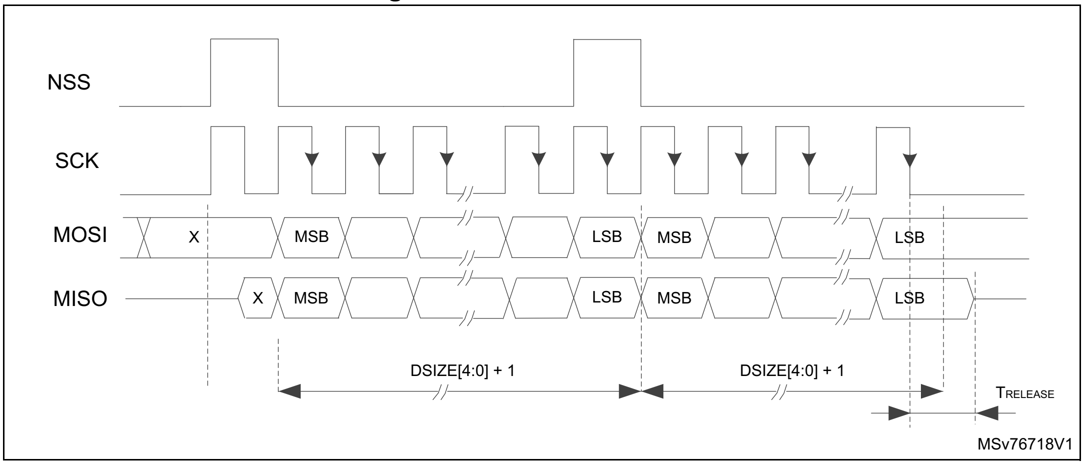
A transfer starts once the CSTART bit is set. In the case of master transmitter, the TxFIFO must be filled by data, too. The CSTART bit can be written either by software or by hardware when a synchronous trigger is detected at Run, Sleep, or Stop mode. The trigger source is selected by the TRIGSEL bits and enabled by the TRIGEN bit of the SPI_AUTOCR register. When the enabled trigger is detected, the transfer starts and continues by handling data until the EOT event or the transfer 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.

41.5 SPI specific modes and control

41.5.1 TI mode

With a specific SP[2:0] bitfield setting of the SPI_CFG2 register, the SPI can be configured compliant with the TI protocol. The SCK and NSS signals polarity, phase and flow, as well as the bit 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 NSS signal synchronizes the protocol by pulses over the LSB data bit as it is shown in *Figure 467*.

Figure 467. TI mode transfer



In slave mode, the clock generator is used to define the time when the slave output at MISO pin becomes high-Z when the current transfer finishes. The master baud rate setting (MBR[2:0] of 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 high-Z (TRELEASE) depends on internal resynchronization, too, which takes the next additional 2-4 periods of the clock signal feeding the generator. It is given by the following formula:

$$\frac{Tbaud}{2} + 2 \times Tspi\_ker\_ck \leq Trelease \leq \frac{Tbaud}{2} + 4 \times Tspi\_ker\_ck$$

If the slave detects a misplaced NSS pulse during a data transfer the TIFRE flag is set.

41.5.2 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and the 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. The OVR flag triggers an interrupt if the OVRIE bit is set. Clearing the OVR bit is done by setting the OVRC bit of the SPI_IFCR register. Clearing the RxFIFO content by performing software reads before the OVR bit is cleared reduces the risk of any immediate repetition of its next overrun. 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)

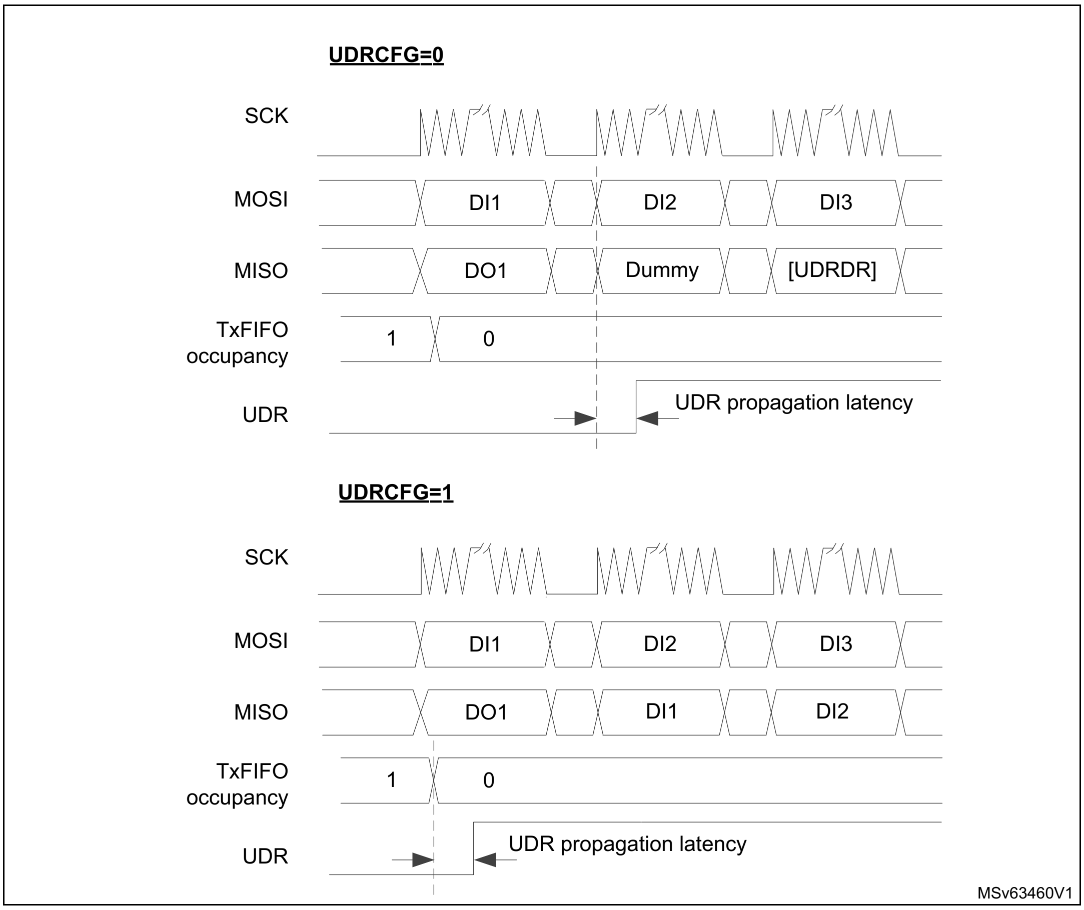
In a slave-transmitting mode, the underrun condition is captured internally by hardware if no data is available for transmission in the slave TxFIFO commonly. The UDR flag setting is then propagated into the status register by hardware (see note below). UDR triggers an interrupt if the UDRIE bit is set.

Underrun detection logic and system behavior depend on the UDRCFG bit. When an underrun is detected by the 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 the master starts to communicate a new data frame while TxFIFO is empty. Then single additional dummy (accidental) data is always inserted between the last valid data and the constant pattern defined at the UDRDR register (see *Figure 468*). The second configuration (UDRCFG=1) can be used in circular topography structures (see *Figure 460*). Assuming that TxFIFO is not empty when the 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 reenabled once the software clears the UDR flag and this clearing is propagated into SPI logic by hardware. Writing some data to the TxFIFO before the UVR bit is cleared reduces the risk of any immediate repetition of its next underrun.

The data transferred by the slave is unpredictable especially when the transfer starts or continues while TxFIFO is empty and the underrun condition is either not yet captured or just cleared. Typically, this is the case when SPI is just enabled or when a transfer with a defined size just starts. First bits can be corrupted in this case, as well, when the slave software writes the first data into the empty TxFIFO too close prior to starting the data transfer (propagation of the data into TxFIFO takes a few APB clock cycles).

Figure 468. Optional configurations of the slave behavior when an underrun condition is detected



Note: The hardware propagation of an UDR event needs additional traffic on the bus. It always takes a 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 the data frame transfer or when the SCK line is at idle in between the frames, the next extra underrun pattern is sent initially by the slave before the valid data from TxFIFO becomes transferred 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 NSS signal (NSS pin in slave select hardware mode, or SSI bit in slave select 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 until the MODF bit is set. This disables the 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 of the SPI_IFCR register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled to its nonactive level before reenabling the SPI, by setting the SPE bit.

As a security, the 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 multimaster conflict.

A correct software procedure when a master overtakes the bus in multimaster systems must be the following one:

1. Switch into master mode while SSOE = 0 (potential conflict can appear when another master occupies the bus. In this case, MODF is raised, which prevents any next node switching into master mode).
2. Put GPIO pin dedicated for another master slave select control into active level.
3. Perform a data transfer.
4. Put GPIO pin dedicated for another master slave select control into nonactive level.
5. Switch back to slave mode.

CRC error (CRCE)

This flag is used to verify the validity of the value received when the CRCEN bit of the SPI_CFG1 register is set. The CRCE flag of 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 the CRCEIE bit is set. Clearing the bit CRCE is done by a writing 1 to the CRCEC bit of the SPI_IFCR register.

TI mode frame format error (TIFRE)

A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the TIFRE flag is set in the SPI_SR register. The SPI is not disabled when an error occurs, the NSS pulse is ignored, and the SPI waits for the next NSS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of a few data frames.

The TIFRE flag is cleared by writing 1 to the TIFREC bit of the SPI_IFCR register. If the TIFREIE bit is set, an interrupt is generated on the NSS error detection. As data consistency is no longer guaranteed, communication must be reinitiated by software between master and slave.

41.5.3 CRC computation

Two separate CRC calculators are implemented to check the reliability of transmitted and received data. The CRC polynomial configuration depends on the instance. Refer to *Section 41.3: SPI implementation* for more information.

The length of the polynomial is defined by the most significant bit of the value stored in the SPI_CRCPOLY register. It must be greater than the data frame size (in bits) defined in the DSIZE[4:0] bitfield of the SPI_CFG1 register. To obtain a full-size polynomial, the polynomial length must exceed the maximum data size of the peripheral instance, and the CRC33_17 bit of the SPI_CR1 register must be set to select the most significant bit of the polynomial string.

For example, to select the standard CRC16-CCITT (XMODEM) polynomial x^16 + x^12 + x^5 + 1:

- For a 32-bit instance: write 0x11021 to the SPI_CRCPOLY register and clear the CRC33_17 bit.
- For a 16-bit instance: to obtain the full size, write 0x1021, and set the CRC33_17 bit.

The CRCSIZE field of the SPI_CFG1 register then defines how many most significant bits from the CRC calculation registers are transferred 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. 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 frame size, else the bits unused at the calculation are transferred and expected all zero at the end of the CRC frame if its size is set greater than the polynomial length.

CRC principle

The CRC calculation is enabled by setting the CRCEN bit of 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 the 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 of 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 has been 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 transferred, the TXCRC is transmitted and the data received on the line are compared to the RXCRC value.

Whatever the CRCSIZE configuration, TSIZE cannot be set to 0xFFFF for a full-featured instance, and to 0x3FF for a limited-featured instance, if CRC is enabled.

In transmission, the CRC computation is frozen during the CRC transfer and the TXCRC is transmitted, in a frame of length equal to the CRCSIZE field value.

In reception, the RXCRC is also frozen when the desired number of data is transferred. 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 complete, 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 of 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 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 of the SPI_CR1 register.

The CRC values are reset when the SPI is disabled.

41.6 SPI in low-power modes

The SPI supports autonomous operation down to Stop mode, refer to *Section 41.4.15: Autonomous mode*.

Table 401. 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 this mode cause the device to exit Stop mode.
Standby	The SPI instance is not functional in this mode. It is powered down, and must be reinitialized after exiting Standby mode.

1. Refer to *Section 41.3: SPI implementation* for information about wake-up 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.

41.7 SPI interrupts

Table 402 gives an overview of the SPI events capable of generating 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 the 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 402. SPI wake-up 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 can wake up the system from Sleep mode at each instance. For detailed information about instances capabilities to exit from concrete Stop and Standby mode refer to the *Functionalities depending on the working mode* table.
2. Refer to *Section 41.3: 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.

41.8 SPI registers

41.8.1 SPI control register 1 (SPI_CR1)

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.	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 can be changed by software only when SPI is disabled (SPE = 0). It is cleared by hardware if a MODF event occurs

When this bit is set, the SPI_CFG2 register content cannot be modified. This bit is write-protected when SPI is enabled (SPE = 1).

0: AF configuration is not locked

1: AF configuration is locked

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**: Full size (33-bit or 17-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 slave select 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 NSS input internally and the I/O value of the NSS 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 transfer.

The master mode communication must be suspended (using this bit or keeping TXDR empty) before going to Low-power mode.

After software suspension, SUSP flag must be cleared and SPI disabled and re-enabled before the next transfer 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 transfer suspend request is accepted.

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.

0: master transfer is at idle

1: master transfer is ongoing or temporary suspended by automatic suspend

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.

When SPI communication is suspended by hardware automatically, it may 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 must 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 transfer without any next constraint. Before 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.

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.

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.

When SPE = 1, SPI data transfer is enabled, the SPI_CFG1 and SPI_CFG2 configuration registers, CRCPOLY, UDRDR, part of the 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 NSS 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.

0: Serial peripheral disabled.

1: Serial peripheral enabled

41.8.2 SPI control register 2 (SPI_CR2)

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**
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 transfer is initialized when CSTART is set while zero value is stored in TSIZE. TSIZE cannot be set to 0xFFFF for a full-featured instance and 0x3FF for a limited-featured instance, when CRC is enabled.

Note: TSIZE[15:10] bits are reserved for limited-featured instances, and must be kept at reset value.

41.8.3 SPI configuration register 1 (SPI_CFG1)

Address offset: 0x008

Reset value: 0x0007 0007

The 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 41.5.1: 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 transferred and compared

Most significant bits are taken into account from polynomial calculation when CRC result is transferred or compared. The length of the polynomial is not affected by this setting.
The value must be equal or a multiple of the data size configured through the DSIZE bitfield.
Its maximum size corresponds to the instance maximum data size.
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: The most significant bit at CRCSIZE bit field is reserved for the peripheral instances where data size is limited to 16 bits.

The CRCSIZE[2:0] bits are fixed to 1 for the peripheral instances with a limited set of features.

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

For more details see *Figure 468: Optional configurations of the slave behavior when an underrun condition is detected*.

0: slave sends a constant pattern defined by the user in 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.

Bits 8:5 **FTHLV[3:0]:** FIFO threshold level

Defines number of data frames in a single data packet. It is recommended that the size of the packet does not exceed 1/2 of FIFO space.

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 bits, better to select FTHLV = 2, 4, 6.

–If SPI data register is accessed as a 32-bit register and DSIZE > 8 bits, better to select FTHLV = 2, 4, 6, while if DSIZE ≤ 8bit, better to select FTHLV = 4, 8, 12.

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

Note: FTHLV[3:2] bits are reserved for instances with a limited set of features.

Bits 4:0 **DSIZE[4:0]:** number of bits in a 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: The most significant bit DSIZE[4] is reserved for instances which data size is limited to 16 bits.

DSIZE[2:0] bits are fixed to 1 for instances with a limited set of features.

41.8.4 SPI configuration register 2 (SPI_CFG2)

Address offset: 0x00C

Reset value: 0x0000 0000

The content of this register is write-protected when SPI is enabled or the IOLOCK bit is set in the 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.

When SPI must be disabled temporary for a specific configuration reason (for example 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.

0: The peripheral takes no control of GPIOs while it is disabled

1: The peripheral keeps always control of all associated GPIOs

Bit 30 **SSOM**: NSS output management in master mode

This bit is taken into account in master mode when SSOE is enabled. It allows the NSS output to be configured between two consecutive data transfers.

0: NSS is kept at active level until data transfer is complete, it becomes inactive with EOT flag

1: SPI data frames are interleaved with NSS nonactive pulses when MIDI[3:0] > 1

Bit 29 **SSOE:** NSS output enable

This bit is taken into account in master mode only

0: NSS output is disabled and the SPI can work in multimaster configuration

1: NSS output is enabled. The SPI cannot work in a multimaster environment. It forces the NSS pin at inactive level after the transfer is complete or SPI is disabled with respect to SSOM, MIDI, MSSI, SSIOP bits setting

Bit 28 **SSIOP**: NSS input/output polarity

0: low level is active for NSS signal

1: high level is active for NSS signal

Bit 27 Reserved, must be kept at reset value.

Bit 26 **SSM:** Software management of internal slave select signal input

0: Input value of the internal slave select signal is determined by the external NSS hardware pin

1: Input value of the internal slave select signal is determined by the SSI bit controlled by software

Note: When master uses hardware NSS output (SSM = 0 and SSOE = 1), the internal slave select 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
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
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.
0: no swap
1: MOSI and MISO are swapped

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 in the SPI_CFG1 register is configured shorter than 8 bits, 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 NSS Idleness

Specifies an extra delay, expressed in number of SPI clock cycle periods, inserted additionally between active edge of NSS 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.

41.8.5 SPI interrupt enable register (SPI_IER)

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.	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:** TXTF 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

41.8.6 SPI status register (SPI_SR)

Address offset: 0x014

Reset value: 0x0000 1002

All the flags of this register are not cleared automatically when the SPI is reenabled. 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 bits, the value gives number of remaining data frames persisting at RxFIFO.

When data size is greater than 16 bits, these bits are always read as 00. As a result, the single data frame received in the FIFO cannot be detected neither by RWNE nor by RXPLVL bits if the data size is set from 17 to 24 bits. The user must then apply other methods to detect the number of data received, such as monitor the EOT event when TSIZE > 0 or RXP events when FTHLV = 0.

00: no next frame is available at RxFIFO

01: 1 frame is available

10: 2 frames are available*

11: 3 frames are available*

Note: (*): Possible value when data size is set up to 8 bits only.

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.

This flag is set when SPI is reset or disabled.

0: current data transfer 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 complete after CSUSP request is done or at master automatic suspend receive mode (the MASRX bit is set in the SPI_CR1 register) on RxFIFO full condition.

SUSP generates an interrupt when EOTIE is set.

This bit must be cleared prior to disabling the SPI. This is done by setting the 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

When MODF is set, SPE and IOLOCK bits of the SPI_CR1 register are reset and setting SPE again is blocked until MODF is cleared.

This bit is cleared by writing 1 to the MODFC bit of the SPI_IFCR exclusively.

0: no mode fault

1: mode fault detected.

Bit 8 **TIFRE**: TI frame format error

This bit is cleared by writing 1 to the TIFREC bit of the SPI_IFCR exclusively.

0: no TI Frame Error

1: TI frame error detected

Bit 7 **CRCE:** CRC error

This bit is cleared when SPI is re-enabled or by writing 1 to the CRCEC bit of the SPI_IFCR optionally.

0: no CRC error

1: CRC error detected

Bit 6 **OVR:** overrun

This bit is cleared when SPI is re-enabled or by writing 1 to the OVRC bit of the SPI_IFCR optionally.

0: no overrun

1: overrun detected

Bit 5 **UDR:** underrun

This bit is cleared when SPI is re-enabled or by writing 1 to the UDRC bit of the SPI_IFCR optionally.

0: no underrun

1: underrun detected

Note: In SPI mode, the UDR flag applies to slave mode only.

Bit 4 **TXTF:** transmission transfer filled

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 the TXTFC bit of the SPI_IFCR exclusively.

The TXTF flag triggers an interrupt if the 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.

0: upload of TxFIFO is ongoing or not started

1: TxFIFO upload is finished

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 the EOTC bit of the SPI_IFCR register optionally.

EOT flag triggers an interrupt if the 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.

In master, EOT event terminates the data transfer and handles NSS output optionally. When CRC is applied, the EOT event is extended over the CRC frame transfer.

To restart the internal state machine properly, SPI is strongly suggested to be disabled and re-enabled before next transfer starts despite its setting is not changed.

0: transfer is ongoing or not started

1: transfer complete

Bit 2 **DXP**: duplex packet

DXP flag is set whenever both TXP and RXP flags are set regardless SPI mode.

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

Bit 1 **TXP:** Tx-packet space available

TXP flag can be changed only by hardware. Its value depends on the physical size of the FIFO and its threshold (FTHLV[3:0]), data frame size (DSIZE[4:0] in SPI mode), and actual communication flow. If the data packet is stored by performing consecutive write operations to SPI_TXDR, TXP flag must be checked again once a complete data packet is stored at TxFIFO. TXP is set despite SPI TxFIFO becomes inaccessible when SPI is reset or disabled.

0: not enough free space at TxFIFO to host next data packet

1: enough free space at TxFIFO to host at least one data packet

Bit 0 **RXP:** Rx-packet available

The flag is changed by hardware. It monitors the total number of data currently available at RxFIFO if SPI is enabled. RXP value depends on the FIFO threshold (FTHLV[3:0]), data frame size (DSIZE[4:0] in SPI mode), and actual communication flow. If the data packet is read by performing consecutive read operations from SPI_RXDR, RXP flag must be checked again once a complete data packet is read out from RxFIFO.

0: RxFIFO is empty or an incomplete data packet is received

1: RxFIFO contains at least one data packet

41.8.7 SPI interrupt/status flags clear register (SPI_IFCR)

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.	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 of 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 of the SPI_SR register

Bit 8 **TIFREC**: TI frame format error flag clear

Writing a 1 into this bit clears TIFRE flag of the SPI_SR register

Bit 7 **CRCEC:** CRC error flag clear

Writing a 1 into this bit clears CRCE flag of the SPI_SR register

Bit 6 **OVRC:** overrun flag clear

Writing a 1 into this bit clears OVR flag of the SPI_SR register

Bit 5 **UDRC:** underrun flag clear

Writing a 1 into this bit clears UDR flag of the SPI_SR register

Bit 4 **TXTFC:** transmission transfer filled flag clear

Writing a 1 into this bit clears TXTF flag of the SPI_SR register

Bit 3 **EOTC:** end of transfer flag clear

Writing a 1 into this bit clears EOT flag of the SPI_SR register

Bits 2:0 Reserved, must be kept at reset value.

41.8.8 SPI autonomous mode control register (SPI_AUTOCR)

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.	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:** Hardware control of CSTART triggering enable
- 0: Hardware control disabled
- 1: Hardware 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.

41.8.9 SPI transmit data register (SPI_TXDR)

Address offset: 0x020

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.

half-word-wise (16 bit access) in this case 2 data-bytes or 1 half-word-data can be written by single access.

word-wise (32 bit access). In this case 4 data-bytes or 2 half-word-data or word-data can be written by single access.

Write access of this register less than the configured data size is forbidden.

41.8.10 SPI receive data register (SPI_RXDR)

Address offset: 0x030

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

half-word-wise (16 bit access) in this case 2 data-bytes or 1 half-word data can be read by single access

word-wise (32 bit access). In this case 4 data-bytes or 2 half-word data or word-data can be read by single access.

Read access of this register less than the configured data size is forbidden.

41.8.11 SPI polynomial register (SPI_CRCPOLY)

Address offset: 0x040

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 data size and CRC is enabled (to keep polynomial length grater than data size).

Note: CRCPOLY[31:16] bits are reserved for instances with data size limited to 16 bits. 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.

41.8.12 SPI transmitter CRC register (SPI_TXCRC)

Address offset: 0x044

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 the SPI_CR1 register is set or when a data block is transferred completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.

The number of bits considered at calculation depends on the SPI_CRCPOLY register and CRCSIZE bits settings in the SPI_CFG1 register.

Note: A read to this register when the communication is ongoing may return an incorrect value.

Note: TXCRC[31-16] bits are reserved for instances with data size limited to 16 bits. 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 the CRCSIZE bitfield 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.

41.8.13 SPI receiver CRC register (SPI_RXCRC)

Address offset: 0x048

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 the SPI_CR1 register is set or when a data block is transferred completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.

The number of bits considered at calculation depends on the SPI_CRCPOLY register and CRCSIZE bits settings in the SPI_CFG1 register.

Note: A read to this register when the communication is ongoing may return an incorrect value.

Note: RXCRC[31-16] bits are reserved for the peripheral instances with data size limited to 16 bits. 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.

41.8.14 SPI underrun data register (SPI_UDRDR)

Address offset: 0x04C

Reset value: 0x0000 0000

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
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 the DSIZE bit setting in the SPI_CFG1 register. Underrun condition handling depends on the UDRCFG bit setting in the SPI_CFG1 register.

Note: UDRDR[31-16] bits are reserved for the peripheral instances with data size limited to 16 bits. 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.

41.8.15 SPI register map

Table 403. 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
0x000	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
0x004	SPI_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSIZE[15:0](1)															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	SPI_CFG1	BPASS	MBR[2:0]			Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	CRCSIZE[4:0](1)					TXDMAEN	RXDMAEN	Res.	Res.	Res.	Res.	UDRCFG	FTHLV[3:0](1)				DSIZE[4:0](1)				
	Reset value	0	0	0	0						0		0	0	1	1	1	0	0					0	0	0	0	0	0	0	1	1	1
0x00C	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
0x010	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
0x014	SPI_SR	CTSIZE[15:0](1)																RXWNE	RXPLVL[1:0]		TXC	SUSP	Res.	MODF	TIFRE	CRCE	OVR	UDR	TXTF	EOT	DXP	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
0x018	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			
0x01C	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																
0x020	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
0x024-0x02C	Reserved	Res.																															
0x030	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
0x034 - 0x03C	Reserved	Res.																															
0x040	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
0x044	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
0x048	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
0x04C	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 configuration of this bitfield depends on the features of the SPI instance. For more details, refer to *Section 41.3: SPI implementation*.
2. The bits 31-16 are reserved for the peripheral instances with data size limited to 16 bits. There is no constraint 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: Memory organization* for the register boundary addresses.

42 Serial audio interface (SAI)

Available only on STM32WBA54/55xx devices.

42.1 Introduction

The SAI interface (serial audio interface) offers a wide set of audio protocols due to its flexibility and wide range of configurations. Many audio protocols can be addressed thanks to SAI “free protocol mode”. The free protocol mode allows the application to define the audio frame shape, number of slots, slot size, data size, and so on. For example, the SAI supports I2S standards, LSB- or MSB-justified, PCM/DSP, TDM, and AC’97 protocols. 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).

42.2 SAI main features

- Two independent audio subblocks, which can be transmitters or receivers with their respective FIFO
- 8-word integrated FIFOs for each audio subblock
- Synchronous or asynchronous mode between the audio subblocks
- Master or slave configuration independent for both audio subblocks
- Clock generator for each audio block to target independent audio frequency sampling when both audio subblocks are configured in master mode
- Data size configurable: 8-, 10-, 16-, 20-, 24-, 32-bit
- Audio protocol: I2S, LSB- or MSB-justified, PCM/DSP, TDM, AC’97
- PDM interface, supporting up to 4 microphone pairs
- SPDIF output available if required
- Up to 16 slots available with configurable size
- Number of bits by frame can be configurable
- Frame synchronization active level configurable (offset, bit length, level)
- First active bit position in the slot is configurable
- LSB first or MSB first for data transfer
- Mute mode
- Stereo/Mono audio frame capability
- Communication clock strobing edge configurable (SCK)
- Error flags with associated interrupts if enabled respectively
  - Overrun and underrun detection
  - Anticipated frame synchronization signal detection in slave mode
  - Late frame synchronization signal detection in slave mode
  - Codec not ready for the AC’97 mode in reception

- Interrupt sources when enabled:
  - Errors
  - FIFO requests
- 2-channel DMA interface

42.3 SAI implementation

Table 404. SAI features (1)

SAI features	SAI1
I2S, LSB or MSB-justified, PCM/DSP, TDM, AC’97	X
FIFO size	8 words
SPDIF	X
PDM	X(2)

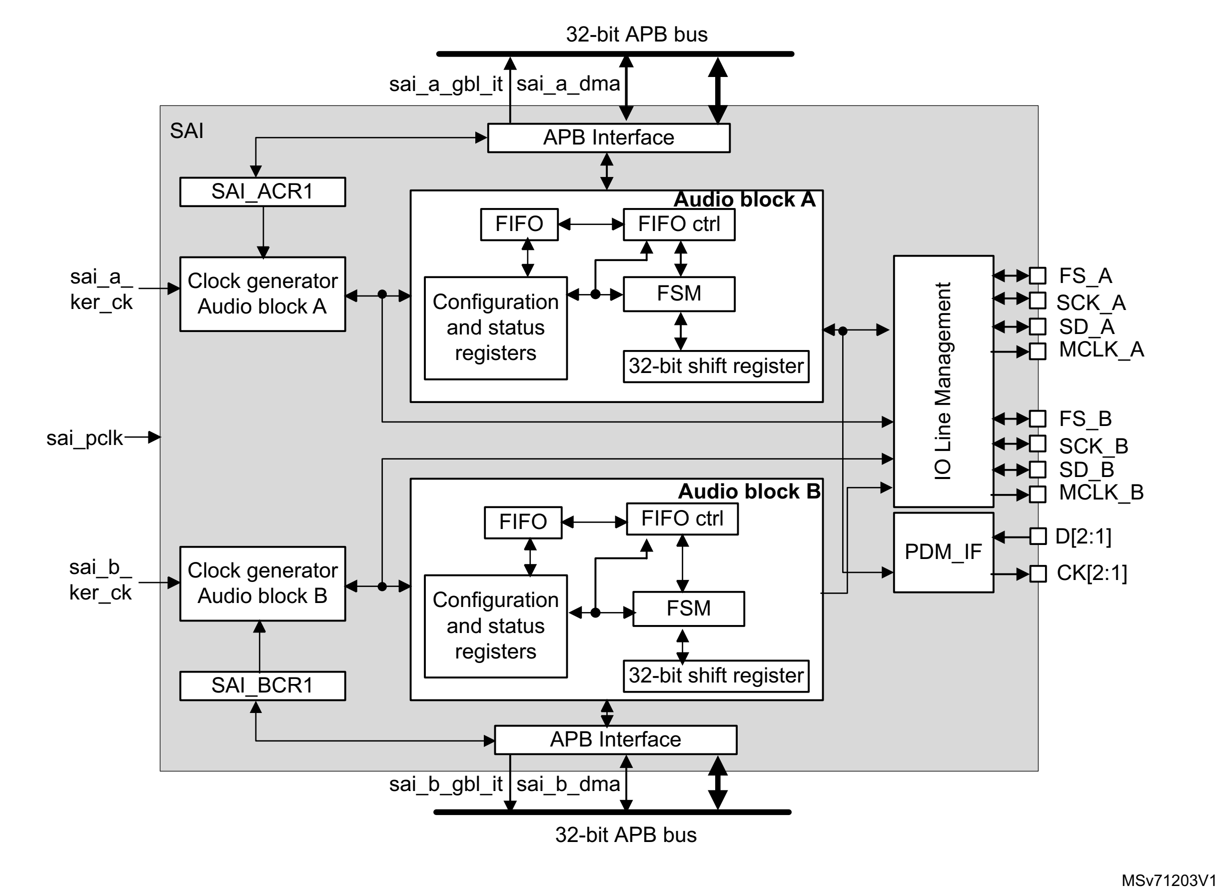
1. ‘X’ = supported, ‘-’ = not supported.
2. Only signals D[2:1], and CK[2:1] are available.

42.4 SAI functional description

42.4.1 SAI block diagram

Figure 469 shows the SAI block diagram while *Table 405* and *Table 406* list SAI internal and external signals.

Figure 469. SAI functional block diagram



The SAI is mainly composed of two audio subblocks with their own clock generator. Each audio block integrates a 32-bit shift register controlled by their own functional state machine. Data are stored or read from the dedicated FIFO. FIFO may be accessed by the CPU, or by DMA 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.

42.4.2 SAI pins and internal signals

Table 405. SAI internal input/output signals

Internal signal name	Signal type	Description
sai_a_gbl_it/ sai_b_gbl_it	Output	Audio block A and B global interrupts
sai_a_dma, sai_b_dma	Input/output	Audio block A and B DMA acknowledges and requests
sai_a_ker_ck/ sai_b_ker_ck	Input	Audio block A/B kernel clock
sai_pclk	Input	APB clock

Table 406. SAI input/output pins

Pin name	Pin 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[2:1]	Output	PDM bitstream clock
SAI_D[2:1]	Input	PDM bitstream data

42.4.3 Main SAI modes

Each audio subblock of the SAI can be configured to be master or slave via the 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 the 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 the SCK_x and FS_x pins are configured as inputs.
- If the SAI subblock is configured to operate synchronously with the second audio subblock, the corresponding SCK_x and FS_x pins are left free to be used as general purpose I/Os.

In slave mode, the 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 block can use a different audio protocol. When PRTCFG[1:0] of the SAI_xCR1 register is set to 0, the free protocol mode is selected and each SAI subblock can emulate I2S standards, LSB- or MSB-justified, PCM/DSP, TDM, or AC’97 protocols.

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, the 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 is enabled by the 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, data, and synchronization lines 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.

42.4.4 SAI synchronization mode

SAI subclock A and B can be synchronized.

Internal synchronization

An audio subblock can be configured to operate synchronously with the second audio subblock in the same SAI. In this case, the bit clock and the frame synchronization signals are shared to reduce the number of external pins used for the communication. The audio block configured in synchronous mode sees its own SCK_x, FS_x, and MCLK_x pins released back as GPIOs while the audio block configured in asynchronous mode is the one for which FS_x and SCK_x ad MCLK_x I/O pins are relevant (if the audio block is considered as master).

Typically, the audio block in synchronous mode can be used to configure the SAI in full duplex mode. One of the two audio blocks can be configured as a master and the other as slave, or both as slaves with one asynchronous block (corresponding SYNCEN[1:0] bits set to 00 in SAI_xCR1) and one synchronous block (corresponding SYNCEN[1:0] bits set to 01 in the SAI_xCR1 register).

Note: Due to internal resynchronization stages, PCLK APB frequency must be higher than twice the bit rate clock frequency.

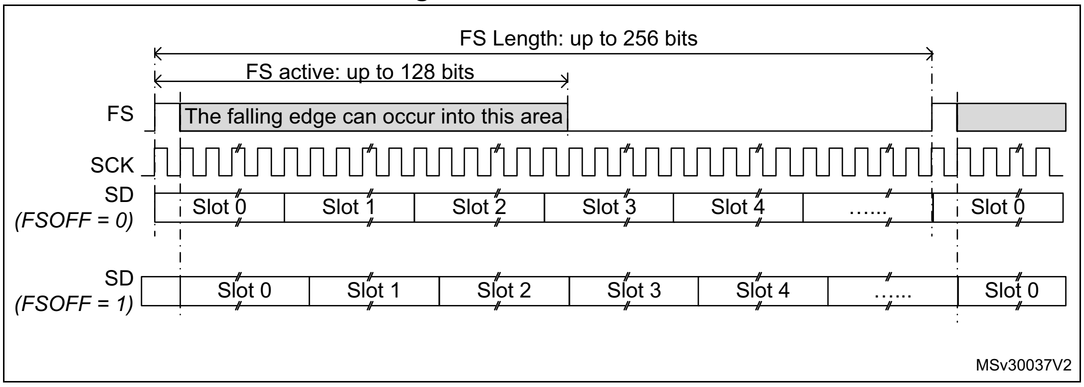
42.4.5 Audio data size

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 is sent first, depending on the configuration of the LSBFIRST bit in the SAI_xCR1 register.

42.4.6 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 to target the different audio protocols with their own specificities concerning this frame synchronization behavior. This reconfigurability is done using the SAI_xFRCR register. *Figure 470* illustrates this flexibility.

Figure 470. 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 configuring the 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 are extended to 0 or the SD line is released to high-Z depending on 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 42.4.8: 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 42.4.14: 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

The FSPOL bit in the SAI_xFRCR register sets the active polarity of the FS pin from which a frame is started. The start of the frame is edge sensitive.

In slave mode, the audio block waits for a valid frame to start transmitting or receiving. The start of the frame is synchronized to this signal. It is effective only if the start of the frame is not detected during an ongoing communication and assimilated to an anticipated start of frame (refer to *Section 42.4.14: 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 42.4.14: 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). The FSOFF bit in the SAI_xFRCR register enables the possibility to choose between two configurations.

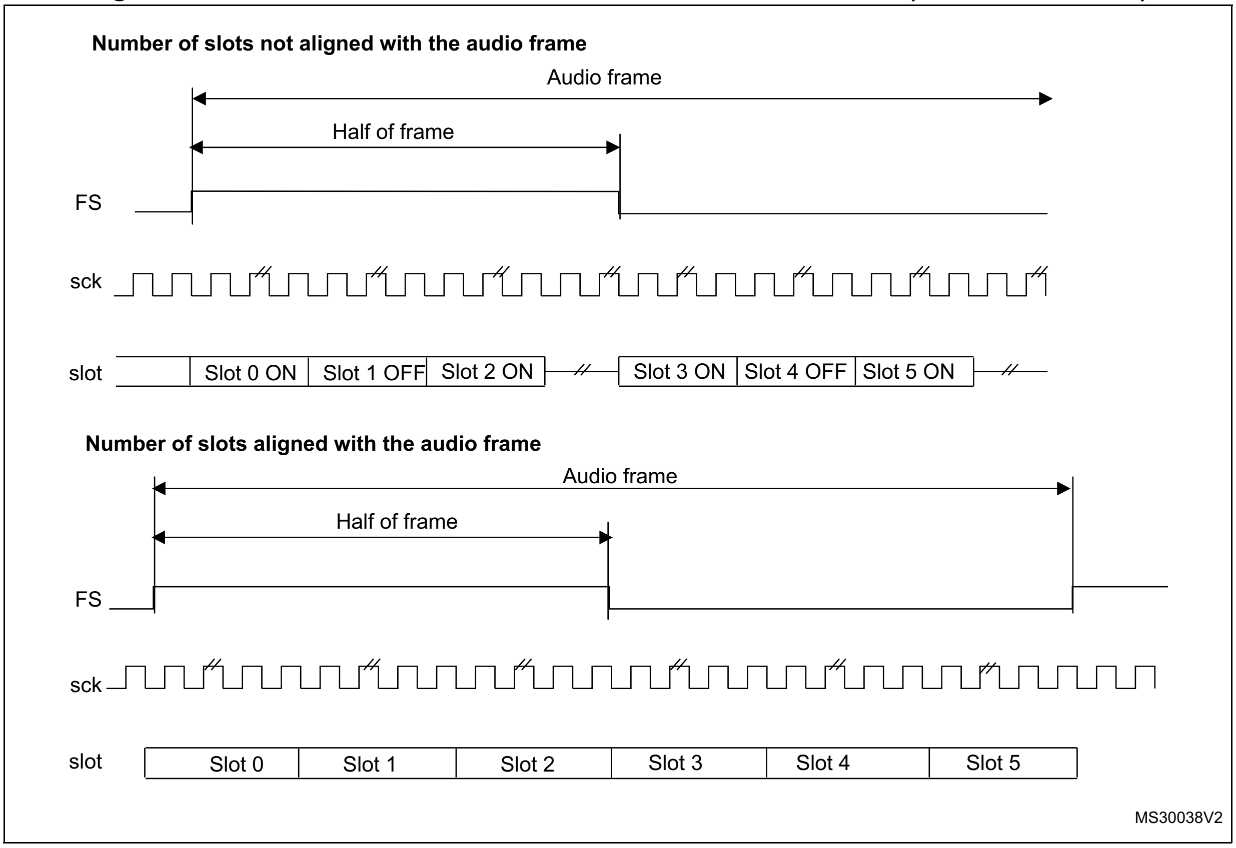
FS signal role

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 or 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 high-Z. In reception mode, the remaining bit clock cycles are not considered until the channel side changes.

Figure 471. FS role is start of frame + channel side identification (FSDEF = TRIS = 1)

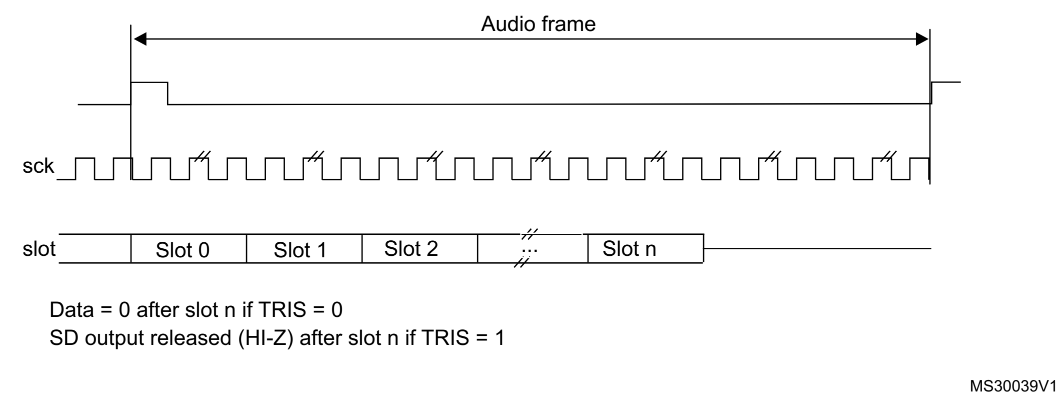


1. The frame length must be even.

If the 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 high-Z during the transfer of these remaining bits. In reception mode, these bits are discarded.

Figure 472. 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.

42.4.7 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.

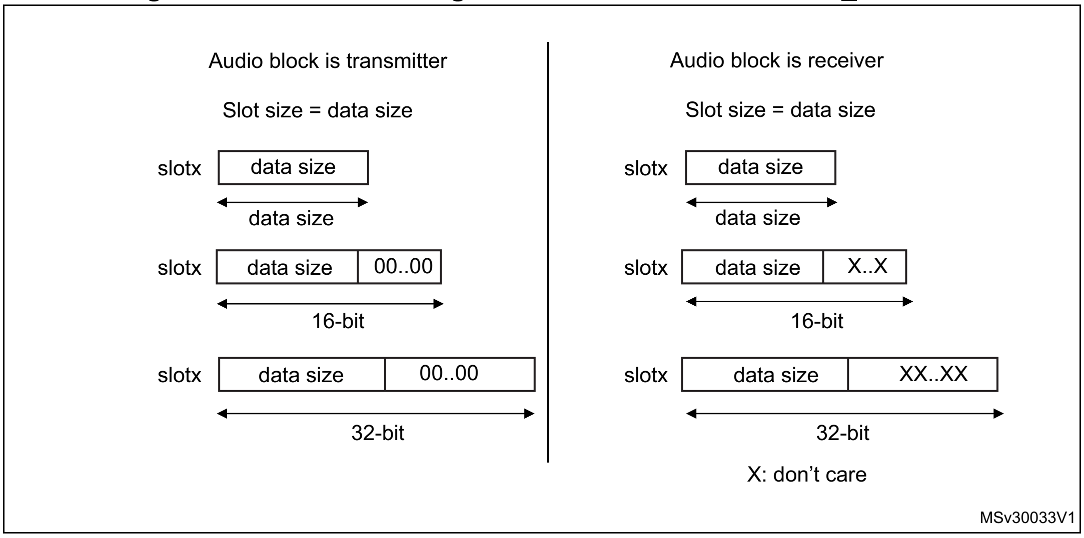
For AC'97 protocol or SPDIF (when bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the number of slots is automatically set to target the protocol specification, and the value of NBSLOT[3:0] is ignored.

Each slot can be defined as a valid slot, or not, by setting SLOTEN[15:0] bits of the SAI_xSLOTR register.

When an invalid slot is transferred, the SD data line is either forced to 0 or released to high-Z depending on the 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.

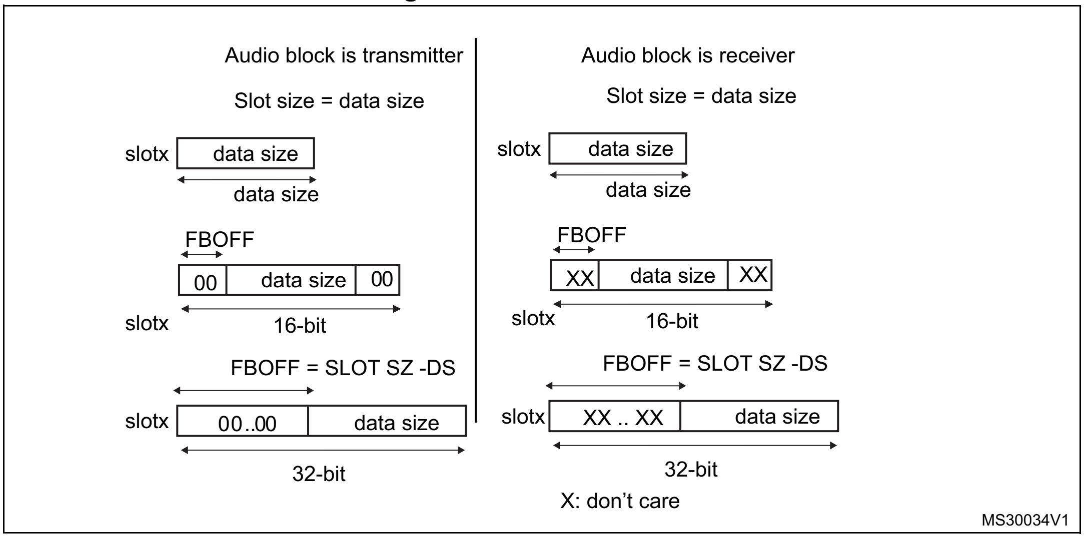
The slot size is also configurable as shown in *Figure 473.* The size of the slots is selected by configuring the SLOTSZ[1:0] bits in the SAI_xSLOTR register. The size is applied identically for each slot in an audio frame.

Figure 473. 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 474. 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 42.4.11: AC’97 link controller*.

42.4.8 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 makes sure that there are no spurs on MCLK_x output when the MCLK_x is switched ON and OFF via the MCKEN bit (with SAIEN = 0).

Table 407 shows MCLK_x activation conditions.

Table 407. MCLK_x activation conditions

<table>
<tr><th>MCLKEN</th><th>NODIV</th><th>SAIEN for block x</th><th>MCLK_x</th></tr>
<tr><td>0</td><td rowspan="2">X</td><td rowspan="2">0</td><td>Disabled</td></tr>
<tr><td>1</td><td>Enabled</td></tr>
<tr><td>0</td><td rowspan="2">1</td><td rowspan="3">1</td><td>Disabled</td></tr>
<tr><td>1</td><td>Enabled</td></tr>
<tr><td>X</td><td>0</td><td>Enabled</td></tr>
</table>

Note: MCLK_x can also be generated in AC'97 mode, when MCLKEN is set to 1.

Generation of the bit clock (SCK_x)

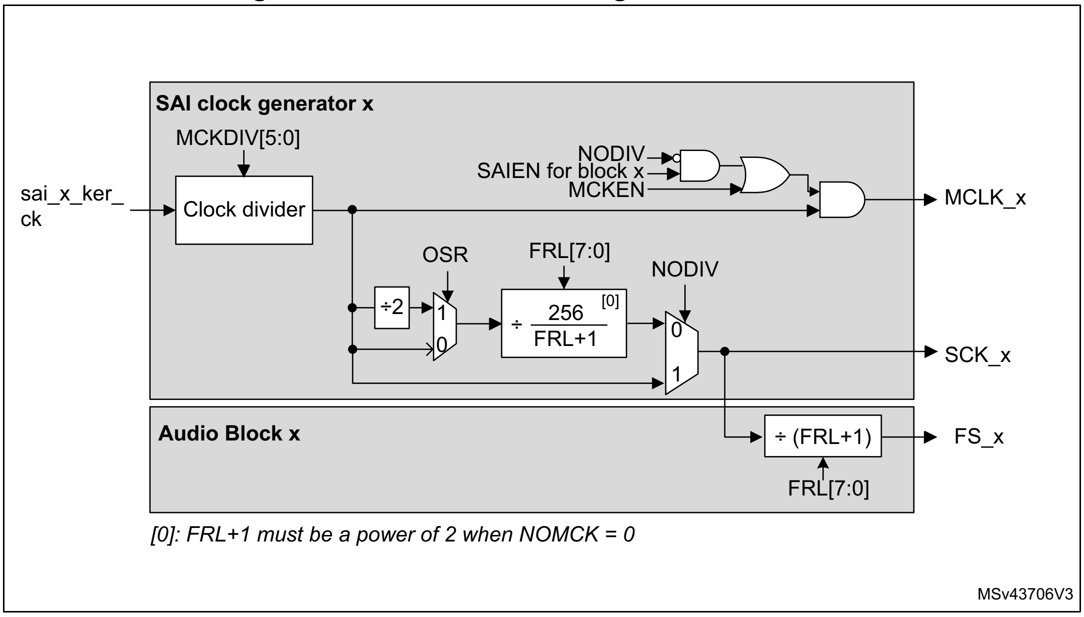
The clock generator provides the bit clock (SCK_x) when the audio block is defined as Master. The frame synchronization (FS_x) is also derived from the signals provided by the clock generator.

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 475 illustrates the architecture of the audio block clock generator.

Figure 475. 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 below.
- 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 (that is, 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, the 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, the 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 the 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 large 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.

MCKDIV = 0 gives the same result as MCKDIV = 1.

Clock generator programming examples

Table 408 gives programming examples for 48, 96, and 192 kHz.

Table 408. 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.

42.4.9 Internal FIFOs

Each audio block in the SAI has its own FIFO. Depending on if the block is defined to be a transmitter or a receiver, the FIFO can be written or read, respectively. Thus, there is only one FIFO request linked to the FREQ bit in the SAI_xSR register.

An interrupt is generated if the FREQIE bit is enabled in the SAI_xIM register. This depends on:

- The 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 the 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 the SAI_xSR register) if no data are available in the SAI_xDR register (FLVL[2:0] bits in SAI_xSR are less than 0b001). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is no longer empty (FLVL[2:0] bits in SAI_xSR are different from 0b000), that is, one or more data are stored in the FIFO.
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO quarter full (FTH[2:0] set to 0b001), an interrupt is generated (FREQ bit set by hardware to 1 in the 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 the 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 than or equal to 0b010).

- When the FIFO threshold bits in the 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 the SAI_xSR register) if less than half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b011). This interrupt (FREQ bit in the 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 than or equal to 0b011).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO three quarter (FTH[2:0] set to 0b011), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if less than three quarters of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b100). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least three quarters of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b100).
- When the FIFO threshold bits in the 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 the SAI_xSR register) if the FIFO is not full (FLVL[2:0] bits in SAI_xSR are less than 0b101). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is full (FLVL[2:0] bits in SAI_xSR are equal to 0b101).

Interrupt generation in reception mode

The interrupt generation depends on the FIFO configuration in reception mode:

- When the FIFO threshold bits in the 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 the SAI_xSR register) if at least one data is available in the SAI_xDR register (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b001). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO becomes empty (FLVL[2:0] bits in the SAI_xSR are equal to 0b000), that is, no data are stored in FIFO.
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO quarter full (FTH[2:0] set to 0b001), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least one quarter of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b010). This interrupt (FREQ bit in the 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 are less than 0b010).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO half full (FTH[2:0] set to 0b010), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least half of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b011). This interrupt (FREQ bit in the 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 are less than 0b011).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO three quarter full (FTH[2:0] set to 0b011), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least three quarters of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b100). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO has less than three quarters of the FIFO data locations available (FLVL[2:0] bits in SAI_xSR is less than 0b100).
- When the FIFO threshold bits in the 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 the SAI_xSR register) if the FIFO is full (FLVL[2:0] bits in SAI_xSR are equal to 0b101). This

interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is not full (FLVL[2:0] bits in SAI_xSR are less than 0b101).

Like interrupt generation, the SAI can use the DMA if the 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 must be right-aligned when written in the SAI_xDR register.

Data received are right-aligned in the SAI_xDR register.

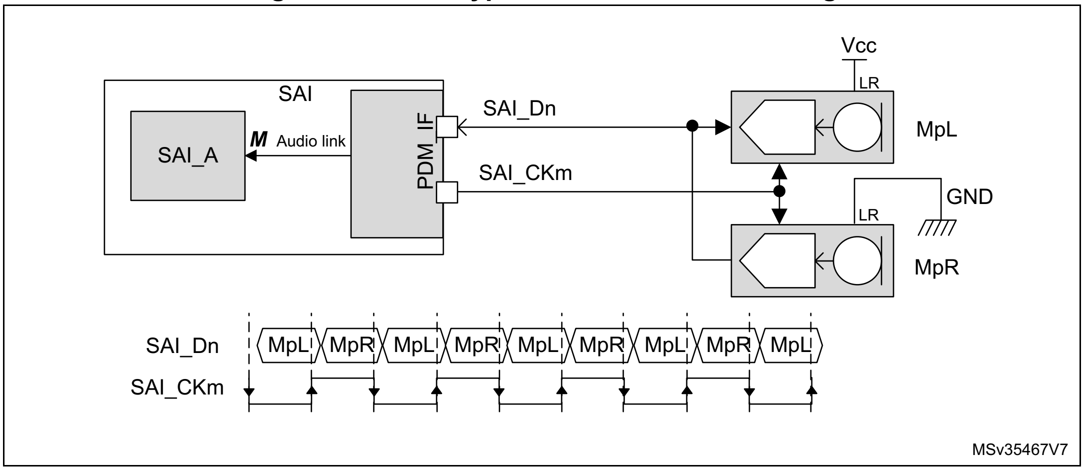
The FIFO pointers can be reinitialized when the SAI is disabled by configuring the FFLUSH bit in the SAI_xCR2 register. If FFLUSH is set when the SAI is enabled, the data present in the FIFO are lost automatically.

42.4.10 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 42.3: SAI implementation*).

Figure 476 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 476. 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 serial audio interface of SAI_A and adapts them to generate a bitstream clock (SAI_CK[m]).

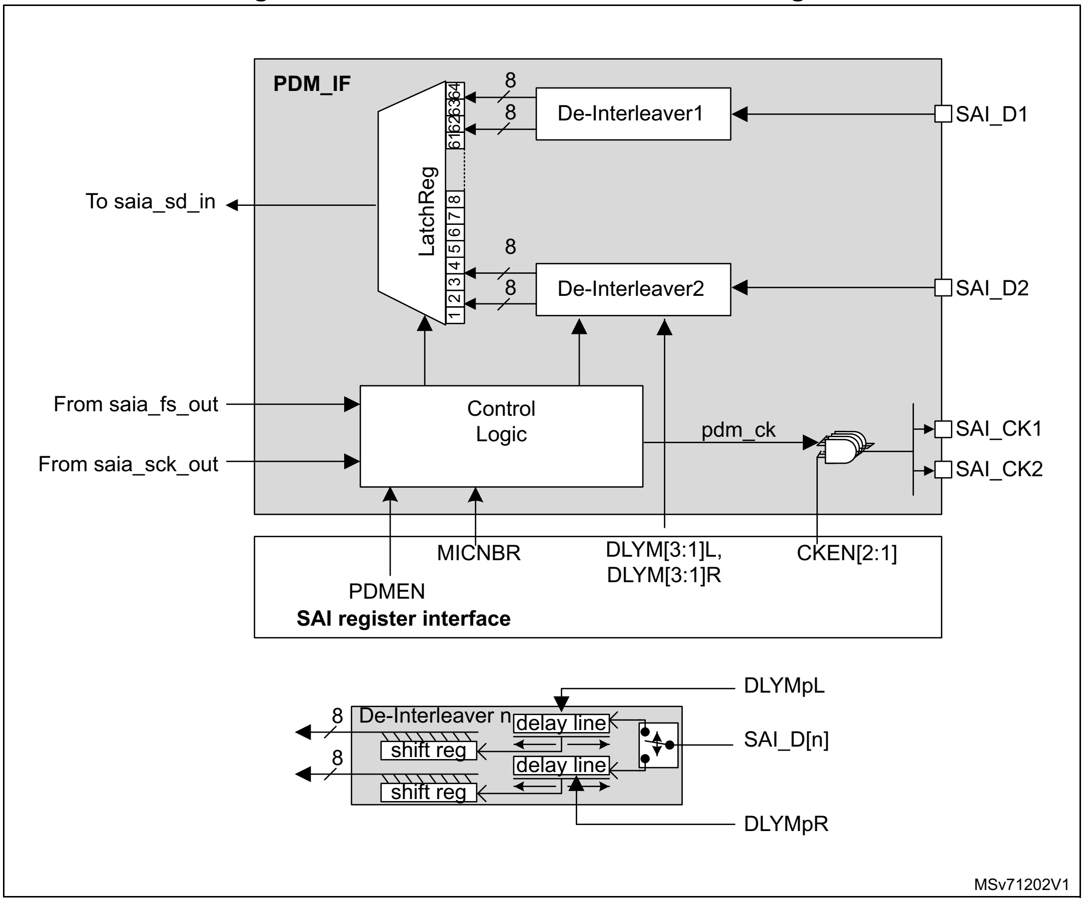
The processing performed into the PDM interface is the following:

1. The PDM interface builds the bitstream clock from the bit clock received from the serial audio 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 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 through the data line of the serial audio interface.

Figure 477 below shows the block diagram of the 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 477. Detailed PDM interface block diagram



1. **n** refers to the number of data lines and **p** to the number of microphone pairs.

The PDM interface can be enabled through the PDMEN bit in the SAI_PDMCR register. However, the PDM interface must be enabled prior to enabling the SAI_A block.

To reduce the memory footprint, the user can select the number of microphones the application needs. This can be done through the MICNBR[1:0] bits. It is possible to choose 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:

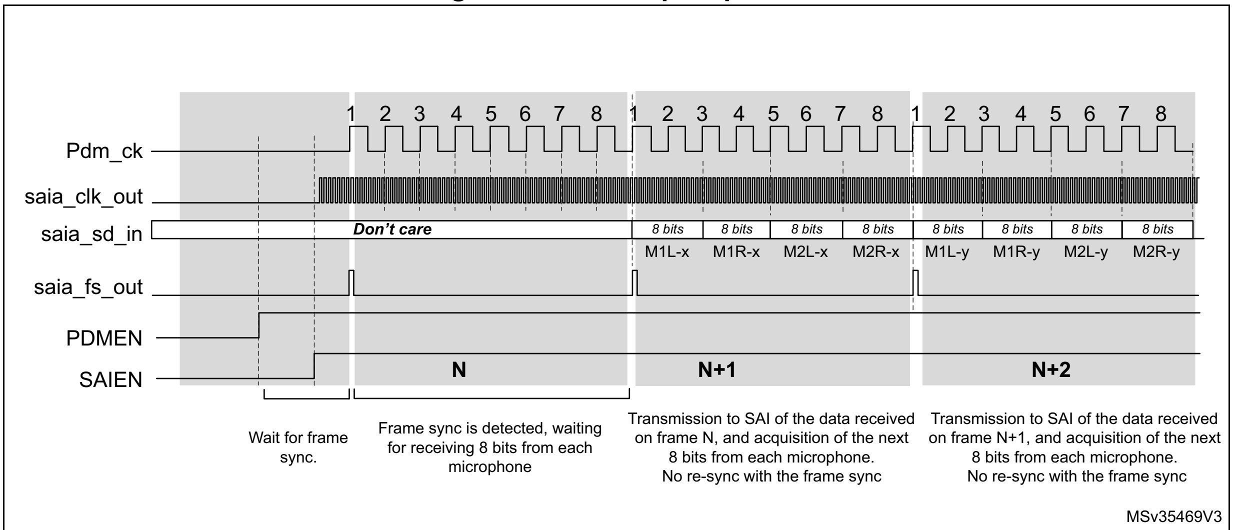
1. Configure SAI_A in TDM master mode (see *Table 409*).
2. Configure the PDM interface as follows:
   a) Define the number of digital microphones via MICNBR.
   b) Enable the bitstream clock needed in the application by setting the corresponding bits on CKEN to 1.
3. Enable the PDM interface, via the PDMEN bit.
4. Enable the SAI_A.

Note: Once the PDM interface and SAI_A are enabled, the first two frames received on SAI_ADR are invalid and must be dropped.

Startup sequence

Figure 478 shows the startup 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 serial audio interface.

Figure 478. 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:

- Number of microphones
- Slot width selected
- LSBFIRST bit

The slot width defines the number 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 number 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.

On the other hand, when the slot width is set to 8 bits, each data available into the SAI_ADR contains 8 useful bits. This increases the number of words stored in the memory. However, it offers the advantage of avoiding extra processing since each word contains information from one microphone.

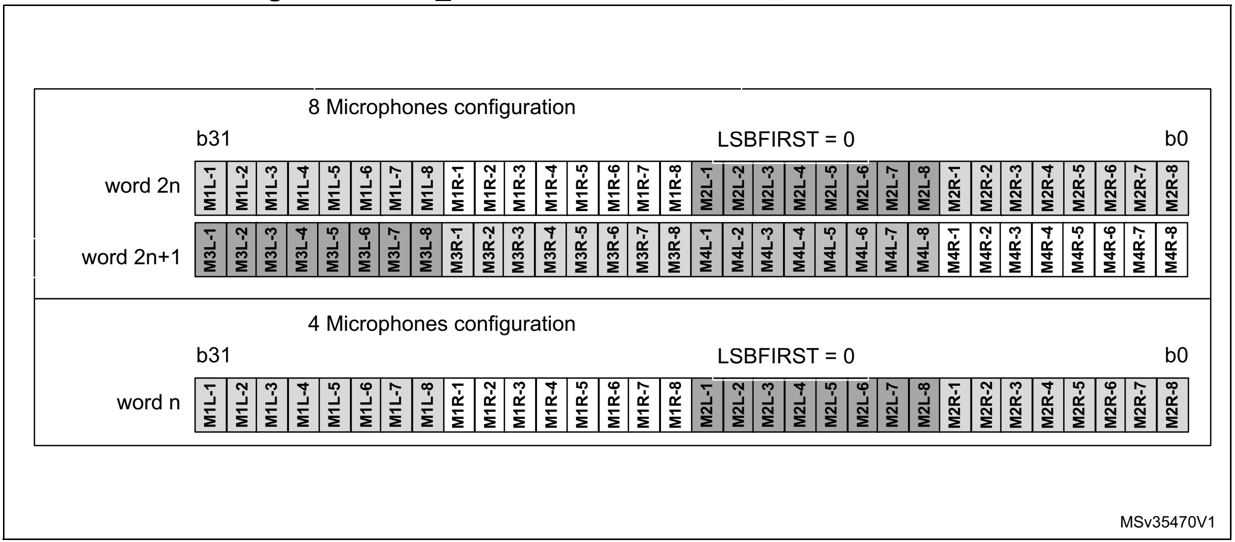
SAI_ADR data format example

- **32-bit slot width** (DS = 0b111 and SLOTSZ = 0). Refer to *Figure 479*.

  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 479. SAI_ADR format in TDM mode, 32-bit slot width

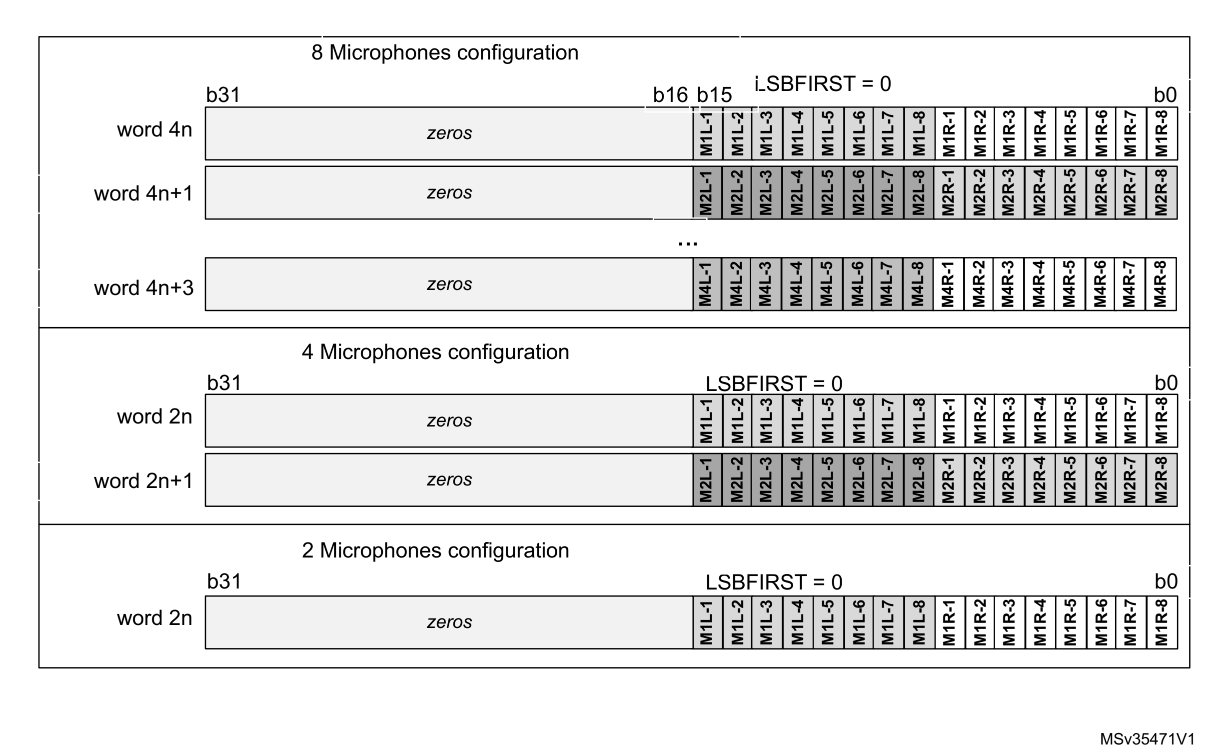


- **16-bit slot width** (DS = 0b100 and SLOTSZ = 0). Refer to *Figure 480*.

  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 a 4- or 2-microphone configuration, the SAI behavior is similar to the 8-microphone configuration. Up to 2 words of 16 bits are required to acquire a byte from 4 microphones and a single word for 2 microphones.

Figure 480. SAI_ADR format in TDM mode, 16-bit slot width

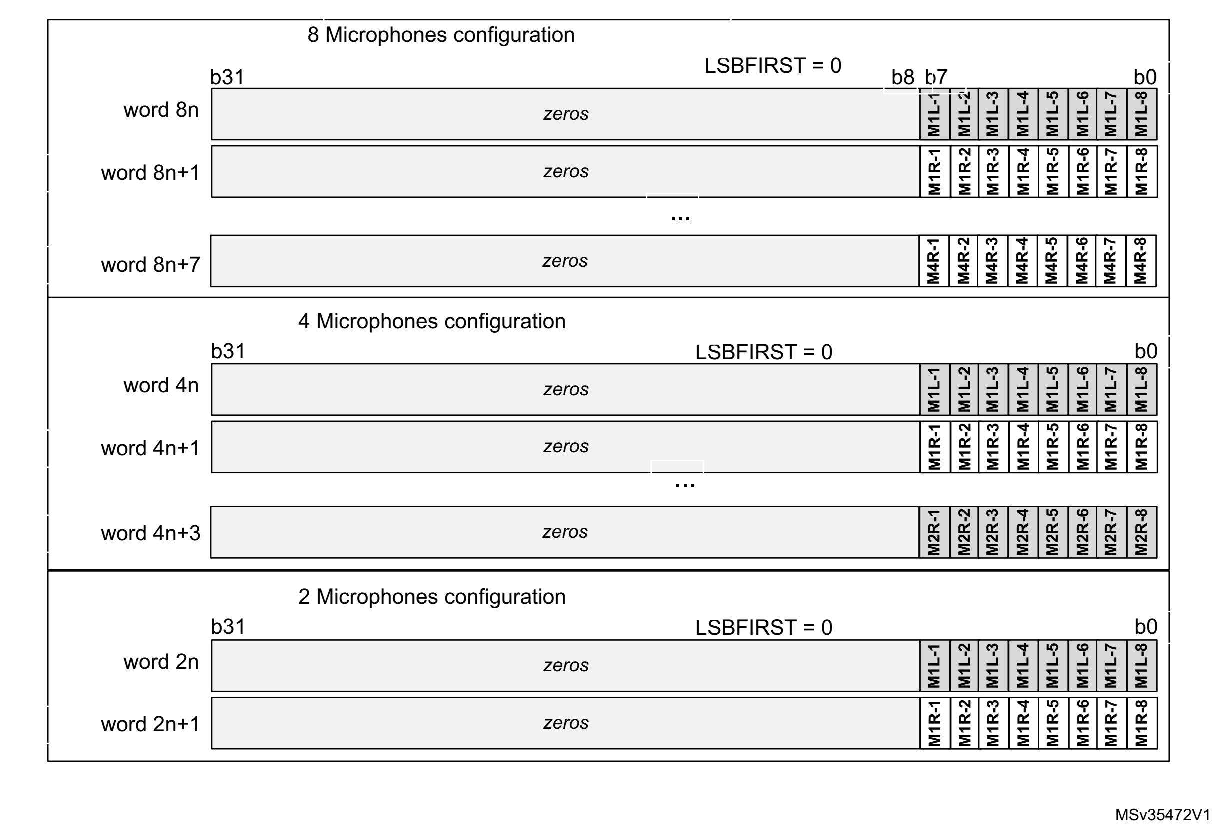


- **Using an 8-bit slot width** (DS = 0b010 and SLOTSZ = 0). Refer to *Figure 481*.

  For an 8-microphone configuration, eight 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 a 4- or 2-microphone configuration, the SAI behavior is similar to the 8-microphone configuration. Up to four words of eight bits are required to acquire a byte from four microphones and two words from two microphones.

Figure 481. SAI_ADR format in TDM mode, 8-bit slot width



TDM mode configuration for PDM interface

The SAI_A serial audio interface is internally connected to the PDM interface to get the microphone samples. The user application must configure the SAI_A serial audio interface as shown in *Table 409* to ensure a good connection with the PDM interface.

Table 409. SAI_A configuration for TDM mode

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 with the frame length and the number of slots (FRL and NBSLOT). See *Table 410*.
LSBFIRST	X	This parameter can be used according to the desired 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 410*.
FSALL	0	Pulse width is one bit clock cycle.

Table 409. SAI_A configuration for TDM mode (continued)

Bit Fields	Values	Comments
FSDEF	0	FS signal is a start of frame.
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 with the slot size, and the frame length (FRL and DS). See *Table 410*.
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 410*.

Adjusting the bitstream clock rate

To program the serial audio interface properly, the user application must take into account the settings given in *Table 409*, and follow the sequence below:

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 42.6.17)*

2. Set the frame length (FRL) using the following formula:

$$FRL = (16 \times (MICNBR + 1)) - 1$$

Table 410. 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 409: SAI_A configuration for TDM mode* 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 must 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 the SAI_PDMDLY register.

The new delay values become effective after two audio frames.

42.4.11 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.

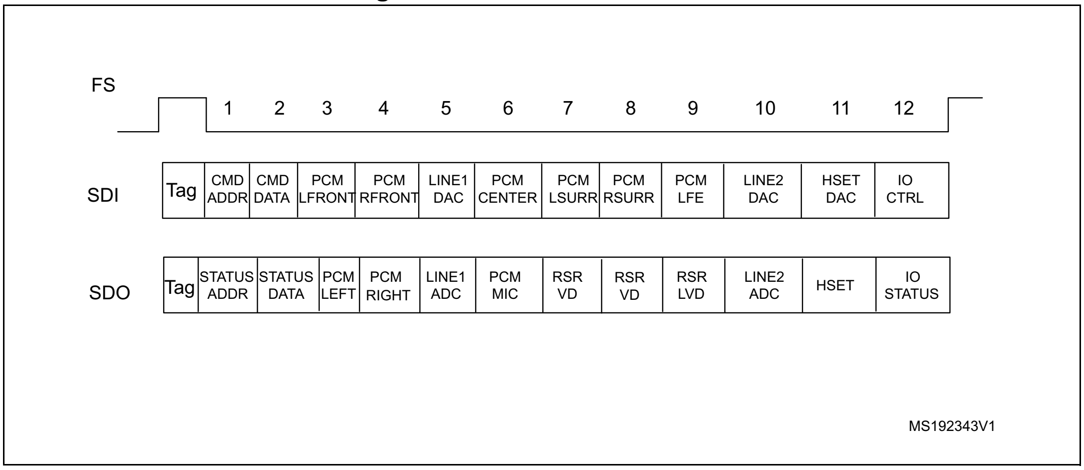
To select this protocol, set the PRTCFG[1:0] bits in the SAI_xCR1 register to 10. When AC’97 mode is selected, only data sizes of 16 or 20 bits can be used, otherwise the SAI behavior is not guaranteed.

- The NBSLOT[3:0] and SLOTSZ[1:0] bits are consequently ignored.
- The number of slots is fixed at 13 slots. The first one is 16 bits wide and all the others are 20 bits wide (data slots).
- The 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 482 shows an AC’97 audio frame structure.

Figure 482. AC’97 audio frame



Note: In the AC’97 protocol, bit 2 of the tag is reserved (always 0), so bit 2 of the TAG is forced to 0 level whatever the value written in the SAI FIFO.

For more details about tag representation, refer to the AC’97 protocol standard.

One SAI can be used to target an AC’97 point-to-point communication.

In receiver mode, the SAI acting as an AC’97 link controller requires no FIFO request and so no data storage in the FIFO when the codec-ready bit in 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 42.4.8: SAI clock generator* must be used with FRL = 255, to generate the proper frame rate (F_{FS_x}).

42.4.12 SPDIF output

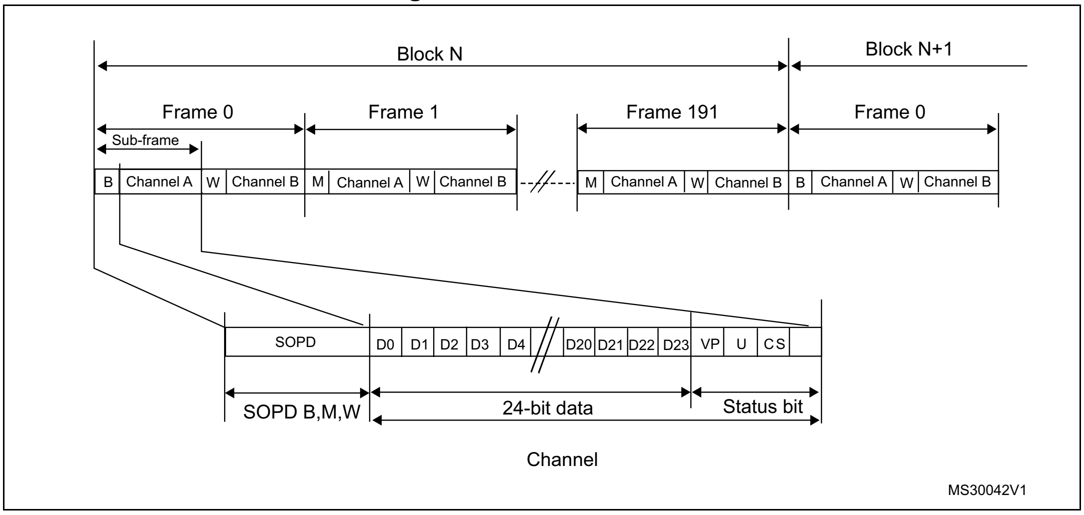
The SPDIF interface is available in transmitter mode only. It supports the audio IEC60958.

To select SPDIF mode, set the PRTCFG[1:0] bits to 01 in the SAI_xCR1 register.

For SPDIF protocol:

- Only the SD data line is enabled.
- The FS, SCK, and MCLK I/Os pins are left free.
- The MODE[1] bit is forced to 0 to select the master mode 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 the 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 must 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 483*.

Figure 483. SPDIF format



An SPDIF block contains 192 frames. Each frame is composed of two 32-bit subframes, generally one for the left channel and one for the right channel. Each subframe is composed of a SOPD pattern (4-bit) to specify if the subframe 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 411*). The next 28 bits of channel information are composed of 24 data bits + 4 status bits.

Table 411. 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:

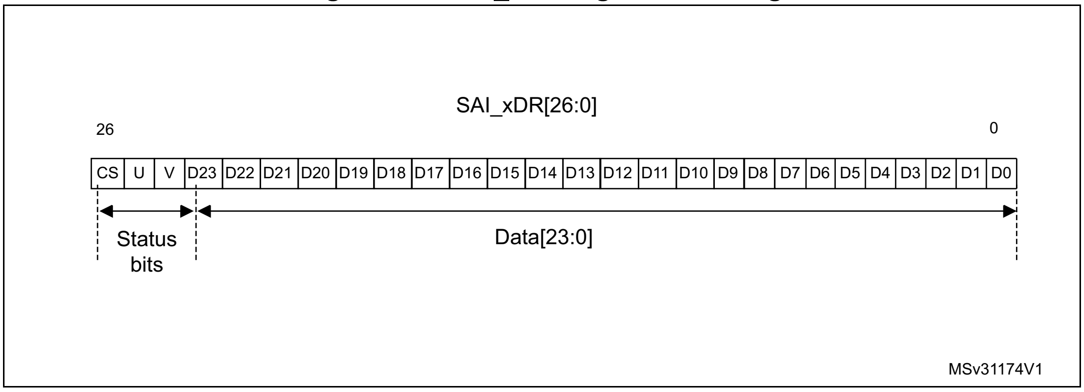
- SAI_xDR[26:24] contain the channel status, user, and validity bits.
- SAI_xDR[23:0] contain the 24-bit data for the considered channel.

If the data size is 20 bits, the data must be mapped on SAI_xDR[23:4].

If the data size is 16 bits, the data must be mapped on SAI_xDR[23:8].

SAI_xDR[23] always represents the MSB.

Figure 484. SAI_xDR register ordering



Note: The transfer is always performed with LSB first.

The SAI first sends the adequate preamble for each subframe in a block. The SAI_xDR is then sent on the SD line (Manchester coded). The SAI ends the subframe by transferring the parity bit calculated as described in *Table 412*.

Table 412. 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 must 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 the 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 the start of a new block (data for preamble B). If the DMA is used, the DMA source base address pointer must be updated accordingly.
5. Enable the DMA stream (DMA peripheral) again if the DMA is used to manage data transfers according to the new source base address.
6. Enable the SAI again by configuring the SAIEN bit in the SAI_xCR1 register.

Clock generator programming in SPDIF generator mode

For the SPDIF generator, the SAI provides a bit clock twice as fast as the symbol-rate. The table below shows examples of symbol rates with respect to the audio sampling rate.

Table 413. 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 the SAI_ACR1 register.

42.4.13 Specific features

The SAI interface embeds specific features that 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 any time. The mute mode is active for entire audio frames. The MUTE bit in the SAI_xCR2 register enables the mute mode when it is configured 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 the 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 the MUTEVAL bit in the SAI_xCR2 register.

If the number of slots set in the NBSLOT[3:0] bits in the SAI_xSLOTR register is greater than 2, the MUTEVAL bit in the SAI_xCR2 register 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 the 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 the 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 the 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, like 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 processing 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 the SD serial output line (compression) or to expand the data after the reception on the SD serial input line (expansion), as illustrated in *Figure 485*. The two companding modes supported are the µ-Law and the A-Law logs, 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).

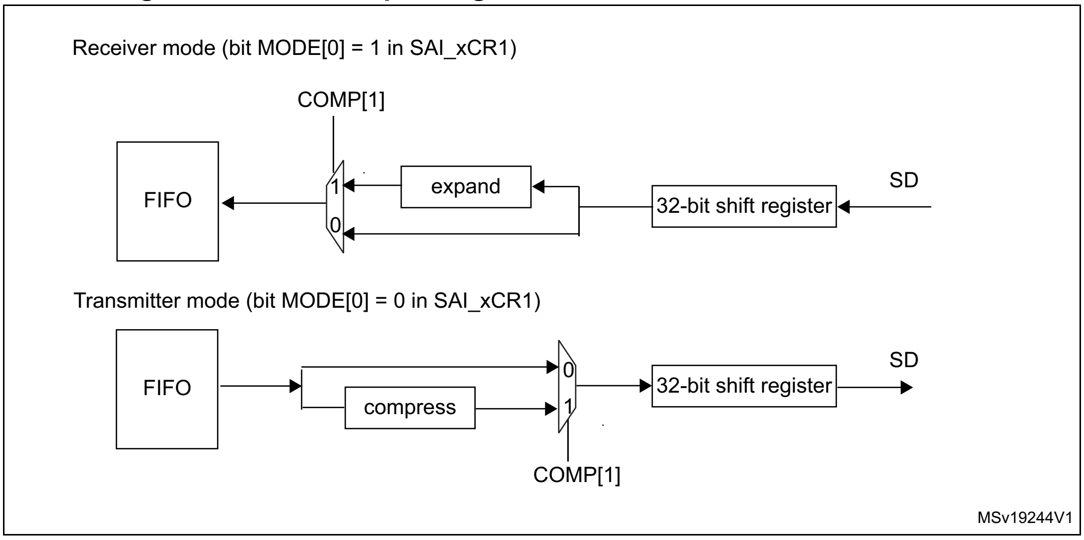
The European companding standard is A-Law and supports 13 bits of dynamic range (COMP[1:0] = 11 in the SAI_xCR2 register).

Both µ-Law and 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 bits wide. For this reason, the DS[2:0] bits in the SAI_xCR1 register are forced to 010 when the SAI audio block is enabled (the SAIEN bit = 1 in the SAI_xCR1 register) and when one of these two companding modes is selected through the COMP[1:0] bits.

If no companding processing is required, the COMP[1:0] bits must be kept clear.

Figure 485. 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 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 the TRIS bit).

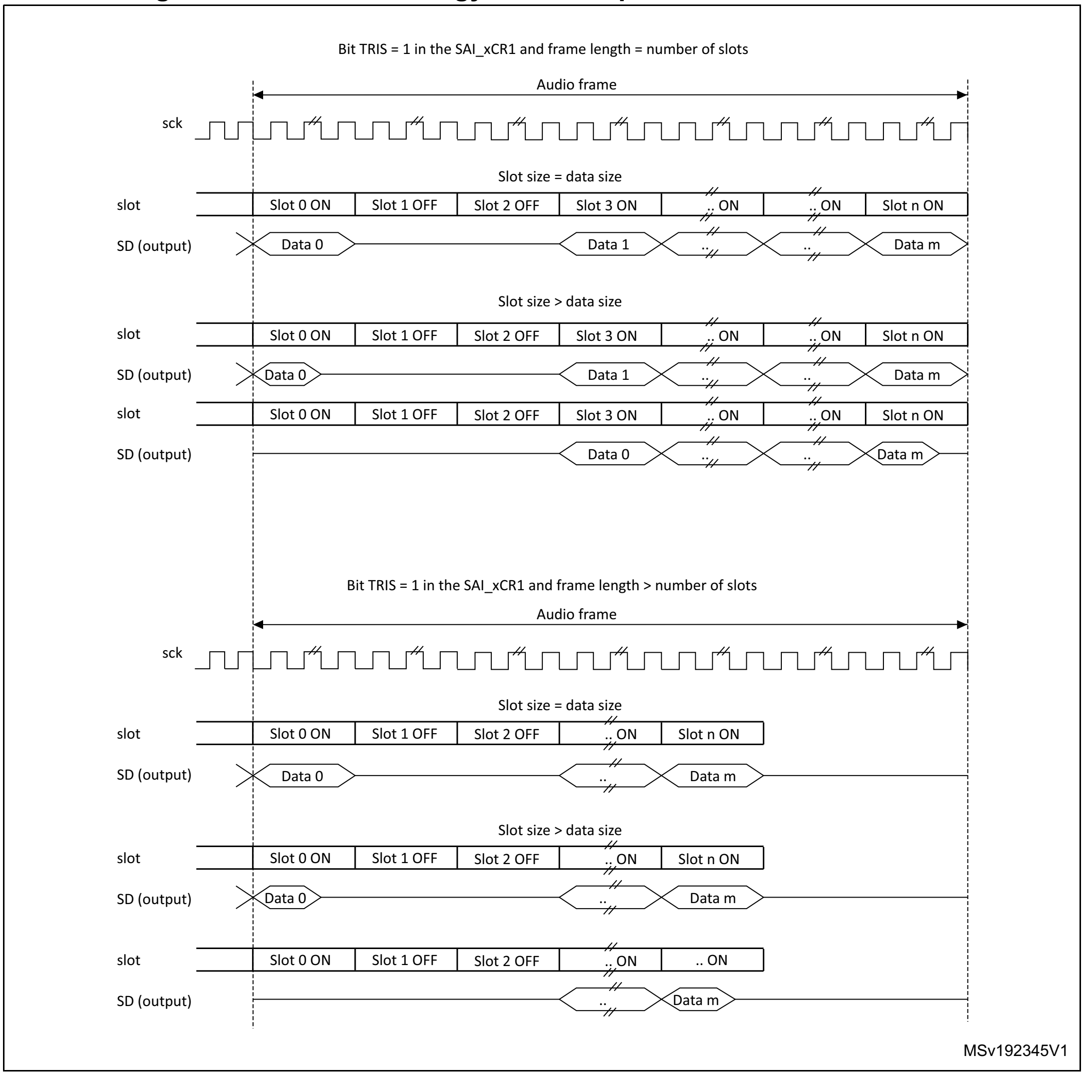
- Either the SAI forces 0 on the SD output line when an inactive slot is transmitted, or
- The line is released in high-Z state at the end of the last bit of data transferred, to release the line for other transmitters connected to this node.

It is important to note that the two transmitters cannot attempt to drive the same SD output pin simultaneously, which may 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 the 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.

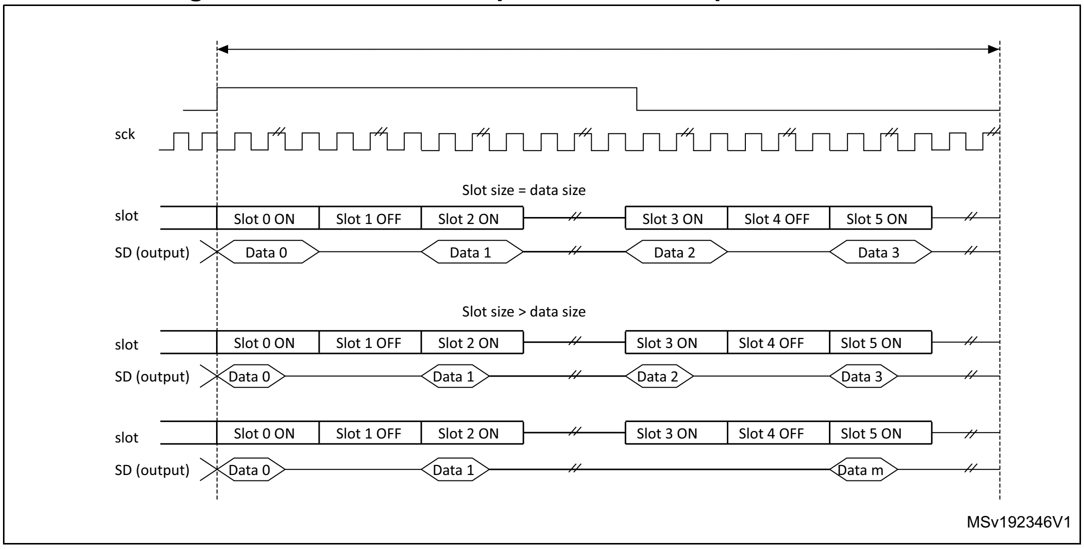
Figure 486 illustrates these behaviors.

Figure 486. 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 487* (where the 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 487. 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 in *Figure 486* and *Figure 487* are replaced by a drive with a value of 0.

42.4.14 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.

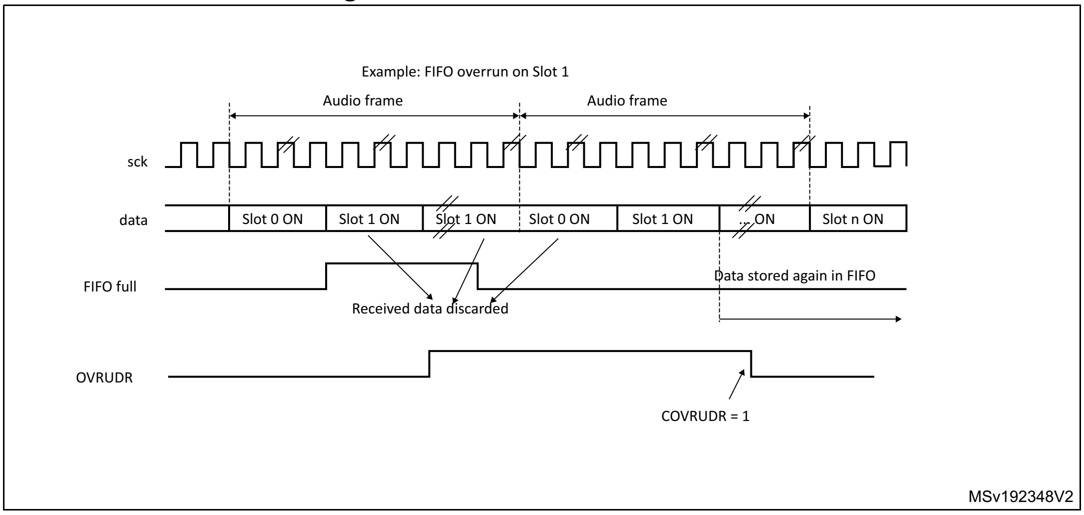
The overrun or underrun errors share the same bit since an audio block can be either receiver or transmitter and each audio block in a given SAI has its own SAI_xSR register.

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 OVRUDR flag in the SAI_xSR register is set, and an interrupt is generated if the 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 a new audio frame) from the slot number that was stored internally when the overrun condition was detected. This avoids data slot dealignment in the destination memory (refer to *Figure 488*).

The OVRUDR flag is cleared when the COVRUDR bit is set in the SAI_xCLRFR register.

Figure 488. Overrun detection error



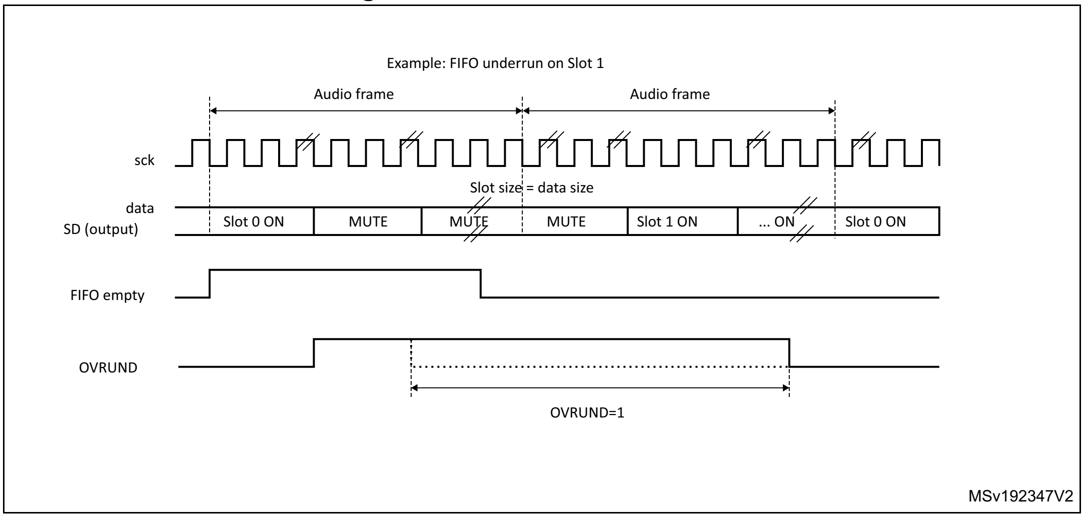
Underrun

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 the 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 489*). 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 the COVRUDR bit in the SAI_xCLRFR register.

The underrun event can occur when the audio subblock is configured as master or slave.

Figure 489. 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, and 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, the 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 the SAIEN bit in the SAI_xCR1 register. To make sure that the SAI is disabled, read back the SAIEN bit and check it is set to 0.
2. Flush the FIFO via the FFLUS bit in the SAI_xCR2 register.
3. Enable the SAI peripheral again (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 expected time, thus generating the signal too late, the LFSDET flag is set and an interrupt is generated if the LFSDETIE bit is set in the SAI_xIM register.

The LFSDET flag is cleared when the 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 must 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 the 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 (slot 0) 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 the CNRDY flag, the 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 the 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.

The MODE, NODIV, and SAIEN bits belong to the SAI_xCR1 register and FRL to the SAI_xFRCR register.

If the WCKCFGIE bit is set, an interrupt is generated when the WCKCFG flag is set in the SAI_xSR register. To clear this flag, set the CWCKCFG bit in the SAI_xCLRFR register.

When the WCKCFG bit is set, the audio block is automatically disabled, thus performing a hardware clear of the SAIEN bit.

42.4.15 Disabling the SAI

The SAI audio block can be disabled at any moment by clearing the SAIEN bit in the SAI_xCR1 register. All the already started frames are automatically completed before the SAI stops working. The 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 that is the master must be disabled first.

42.4.16 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 the basic DMA request/acknowledge protocol.

To configure the audio subblock for DMA transfer, set the 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 42.4.9: Internal FIFOs*). The 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 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 the SAI and FIFO threshold levels to specify when the DMA request is launched.
2. Configure the SAI DMA channel.
3. Enable the DMA.
4. Enable the SAI interface.

42.5 SAI interrupts

The SAI supports 7 interrupt sources, as shown in *Table 414*.

Table 414. 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 42.4.9: 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.

42.6 SAI registers

The peripheral registers have to be accessed by words (32 bits).

42.6.1 SAI configuration register 1 (SAI_ACR1)

Address offset: 0x04

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 42.4.8: 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: Reserved.

11: Reserved

Note: The audio subblock must 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).

42.6.2 SAI configuration register 2 (SAI_ACR2)

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
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 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 must 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 must 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 must 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 must 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

42.6.3 SAI frame configuration register (SAI_AFRCR)

Address offset: 0x0C

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 must 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.

42.6.4 SAI slot register (SAI_ASLOTR)

Address offset: 0x10

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 in the audio frame (including the number of inactive slots). The maximum number of slots is 16.
The number of slots must 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.
0000: Number of slots is 1.
0001: Number of slots is 2.
...
1111: Number of slots is 16.

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.

42.6.5 SAI interrupt mask register (SAI_AIM)

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.	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.

42.6.6 SAI status register (SAI_ASR)

Address offset: 0x18

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 42.4.6: 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.

42.6.7 SAI clear flag register (SAI_ACLRFR)

Address offset: 0x1C

Reset value: 0x0000 0000

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

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

42.6.8 SAI data register (SAI_ADR)

Address offset: 0x20

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.

42.6.9 SAI configuration register 1 (SAI_BCR1)

Address offset: 0x24

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 42.4.8*.
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: Reserved.

11: Reserved

Note: The audio subblock must 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.

42.6.10 SAI configuration register 2 (SAI_BCR2)

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**
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 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 must 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 must 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 must 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 must 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

42.6.11 SAI frame configuration register (SAI_BFRCR)

Address offset: 0x2C

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 must 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.

42.6.12 SAI slot register (SAI_BSLOTR)

Address offset: 0x30

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 in the audio frame (including the number of inactive slots). The maximum number of slots is 16.

The number of slots must 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.

0000: Number of slots is 1.

0001: Number of slots is 2.

...

1111: Number of slots is 16.

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.

42.6.13 SAI interrupt mask register (SAI_BIM)

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.	LFSDET IE	AFSDETI E	CNRDY IE	FREQ IE	WCKCFG IE	MUTEDET IE	OVRUDR IE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LFSDETIE**: Late frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the LFSDET bit is set in the SAI_xSR register.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 5 **AFSDETIE**: Anticipated frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the AFSDET bit in the SAI_xSR register is set.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 4 **CNRDYIE**: Codec not ready interrupt enable (AC'97).
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC'97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR register is set and an interrupt is generated.
This bit has a meaning only if the AC'97 mode is selected through PRTCFG[1:0] bits and the audio block is operates as a receiver.

Bit 3 **FREQIE**: FIFO request interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.
Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interrupt in receiver mode,

Bit 2 **WCKCFGIE**: Wrong clock configuration interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NODIV = 0.
It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.
Note: This bit is used only in Free protocol mode and is meaningless in other modes.

Bit 1 **MUTEDETIE**: Mute detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.
This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 **OVRUDRIE**: Overrun/underrun interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.

42.6.14 SAI status register (SAI_BSR)

Address offset: 0x38

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 42.4.6: 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.

42.6.15 SAI clear flag register (SAI_BCLRFR)

Address offset: 0x3C

Reset value: 0x0000 0000

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

42.6.16 SAI data register (SAI_BDR)

Address offset: 0x40

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.

42.6.17 SAI PDM control register (SAI_PDMCR)

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.	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]		Res.	Res.	Res.	PDMEN
						rw	rw			rw	rw				rw

Bits 31:16 Reserved, must be kept at reset value.

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

Bit 9 **CKEN2**: Clock enable of bitstream clock number 2
This bit is set and cleared by software.
0: SAI_CK2 clock disabled
1: SAI_CK2 clock enabled
Note: It is not recommended to configure this bit when PDMEN = 1.
SAI_CK2 might not be available for all SAI instances. Refer to Section 42.3: 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 42.3: 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 42.3: 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

42.6.18 SAI PDM delay register (SAI_PDMDLY)

Address offset: 0x48

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 bitfield 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 bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D4 line is available.Refer to Section 42.3: 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 bitfield 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 bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D4 line is available.Refer to Section 42.3: 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 bitfield 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 bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D3 line is available.Refer to Section 42.3: 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 bitfield 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 bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D3 line is available.Refer to Section 42.3: 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 bitfield 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 bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D2 line is available.Refer to Section 42.3: 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 bitfield 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 bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D2 line is available.Refer to Section 42.3: 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 bitfield 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 bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D1 line is available.Refer to Section 42.3: 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 bitfield 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 bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D1 line is available.Refer to Section 42.3: SAI implementation to check if it is available.

42.6.19 SAI register map

Table 415. SAI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	Reserved																																
0x04 or 0x24	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
0x08 or 0x28	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
0x0C or 0x2C	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
0x10 or 0x30	SAI_xSLOTR	SLOTEN[15:0]																Res.	Res.	Res.	Res.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0		0	0	0	0	0

Table 415. 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
0x14 or 0x34	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
0x18 or 0x38	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	1	0	0	0
0x1C or 0x3C	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
0x20 or 0x40	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
0x44	SAI_PDMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]		Res.	Res.	Res.	PDMEN
	Reset value																							0	0			0	0				0
0x48	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: Memory organization* for the register boundary addresses.

43 Debug support (DBG)

43.1 DBG introduction and main features

A comprehensive set of debug features is provided to support software development and system integration:

- Breakpoint debugging of the CPU core in the system
- Code execution tracing
- Software instrumentation
- 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 following debug features are based on Arm® CoreSight™ components.

- General features:
  - SWJ-DP: JTAG/Serial-wire debug port
  - AP access ports
  - ROM tables
  - System control space (SCS)
  - Breakpoint unit (BPU)
  - Data watchpoint and trace unit (DWT)
  - Instrumentation trace macrocell (ITM)
  - Trace port interface unit (TPIU)
  - Cross trigger interface (CTI)

The debug features are accessible by the debugger via the debug AP access ports.

Additional information can be found in the Arm® documents referenced in *Section 43.12*.

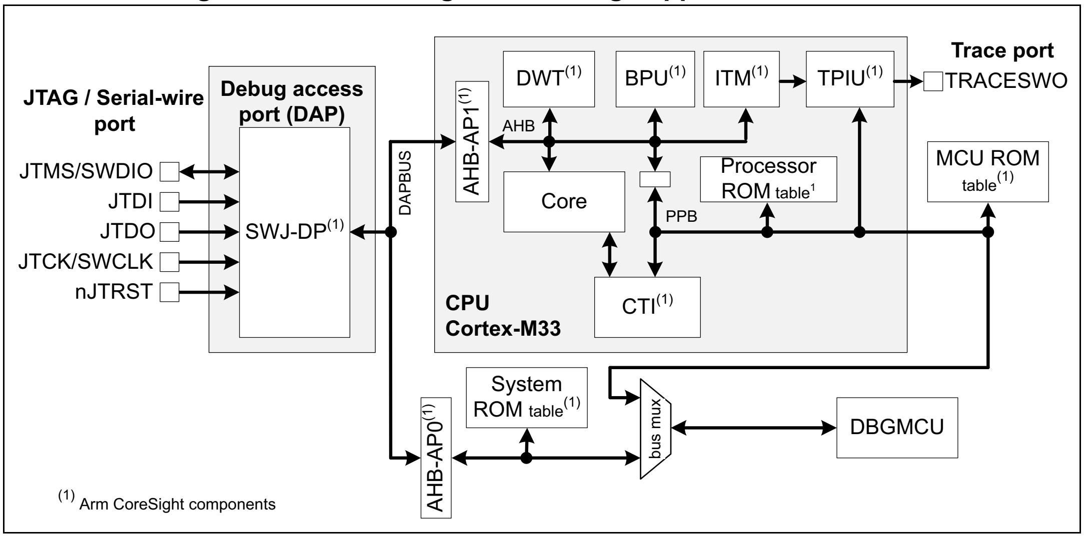
The following debug features are based on STMicroelectronics components

- System debug features:
  - debug MCU controller (DBGMCU).

43.2 DBG functional description

43.2.1 DBG block diagram

Figure 490. Block diagram of debug support infrastructure



43.2.2 DBG pins and internal signals

Table 416. JTAG/Serial-wire debug port pins

Pin name	JTAG debug port		Serial-wire debug port		Pin assignment
	Type	Description	Type	Description	
JTMS/SWDIO	I	JTAG test mode select	IO	Serial wire data in/out	PA13
JTCK/SWCLK	I	JTAG test clock	I	Serial wire clock	PA14
JTDI	I	JTAG test data input	-	-	PA15
JTDO/TRACESWO(1)	O	JTAG test data output	-	-	PB3
nJTRST	I	JTAG test reset	-	-	PB4

1. Debug access port JTDO and Trace port TRACESWO are multiplexed on a single device GPIO pin.

Table 417. 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.

43.2.3 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 DAPCLK domain, which covers the rest of the SWJ-DP.

The DAPCLK is a gated version of the CPU bus clock hclk1.

The DAPCLK domain is enabled by the debugger using the CDBGPWRUPREQ bit in the DP_CTRLSTATR register. The clock must be enabled before the debugger can access any of the debug features on the device. The availability of the clock is reflected in the CDBGPWRUPACK bit in the DP_CTRLSTATR register. DAPCLK is disabled at power up, after OBL, and after a wakeup from Standby. DAPCLK must be disabled when the debugger is disconnected to avoid wasting energy.

The debug components included in the processor are clocked with the processor core clock hclk1.

43.2.4 DBG power domains

The debug components are located in the core power domain. This means that debugger connection is not possible in Standby modes. To avoid losing the connection when the device enters Standby mode, the power to the core can be maintained by setting a bit in the debug unit (DBGMCU). This also keeps the processor clocks active and hold off the reset, so that the debug session is maintained.

43.2.5 DBG low-power modes

The devices include power saving features that allow the core power domain to be switched off or clocks to be stopped when not required. If the power is switched off all debug components are inaccessible to the debugger. When the core is not clocked all debug components except the DBGMCU are inaccessible to the debugger. To avoid this, low-power mode emulation is implemented. If emulation is enabled, the device still enters low-power mode, but the clocks and power are maintained. In other words, the device behaves similar as if it is in low-power mode, but the debugger does not lose the connection to the CPU.

The low-power emulation mode is programmed in the DBGMCU. For more information refer to *Section 43.12.4: Low-power mode emulation*.

43.2.6 DBG 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 device 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 Readout protection (RDP), as shown in the following table.

Table 418. Debug features control

<table>
<tr><th>RDP level</th><th>dbgen</th><th>spiden</th><th>niden</th><th>spniden</th><th>Description</th></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>Debug and trace is enabled whatever the state of the processor. The debugger access to secure memory is permitted.</td></tr>
<tr><td>0.5</td><td rowspan="2">1</td><td rowspan="2">0</td><td rowspan="2">1</td><td rowspan="2">0</td><td>Debug and trace is enabled when the processor is in non-secure state. The debugger access to secure memory is disabled.</td></tr>
<tr><td>1</td><td>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.</td></tr>
<tr><td>2</td><td>0</td><td>0</td><td>0</td><td>0</td><td>Debug and trace is disabled.</td></tr>
</table>

Note: Security features are only relevant when option bit TZEN = 1. If security features are disabled, the authentication signals are still set according to the RDP level, but since the processor and all 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 areas (when permitted) must be performed using secure transactions from the AP, that is with the PROT[6] bit set in the AP_CSWR register.

The debugger access is disabled while the processor is booting from system Flash memory (RSS), whatever the RDP level.

43.2.7 Serial-wire and JTAG debug port

The Serial-wire and JTAG debug port (SWJ-DP) is a CoreSight component that implements an external access port for connecting debugging equipment.

The two following types of interface can be configured:

- a 5-pin standard JTAG interface (JTAG-DP)
- a 2-pin (clock + data) Serial-wire debug port (SW-DP)

The two modes are mutually exclusive since they share the same I/O pins.

By default the JTAG-DP is selected after a system or a power-on reset. The five I/O pins are configured by hardware in debug alternative function mode.

The SWJ-DP incorporates pull-up resistors on JTDI, JTMS/SWDIO and nJTRST, as well as a pull-down resistor on JTCK/SWCLK.

A debugger can select the SW-DP by transmitting the following serial data sequence on JTMS/SWDIO:

... (50 or more ones) ..., 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, ... (50 or more ones) ...

JTCK/SWCLK must be cycled for each data bit.

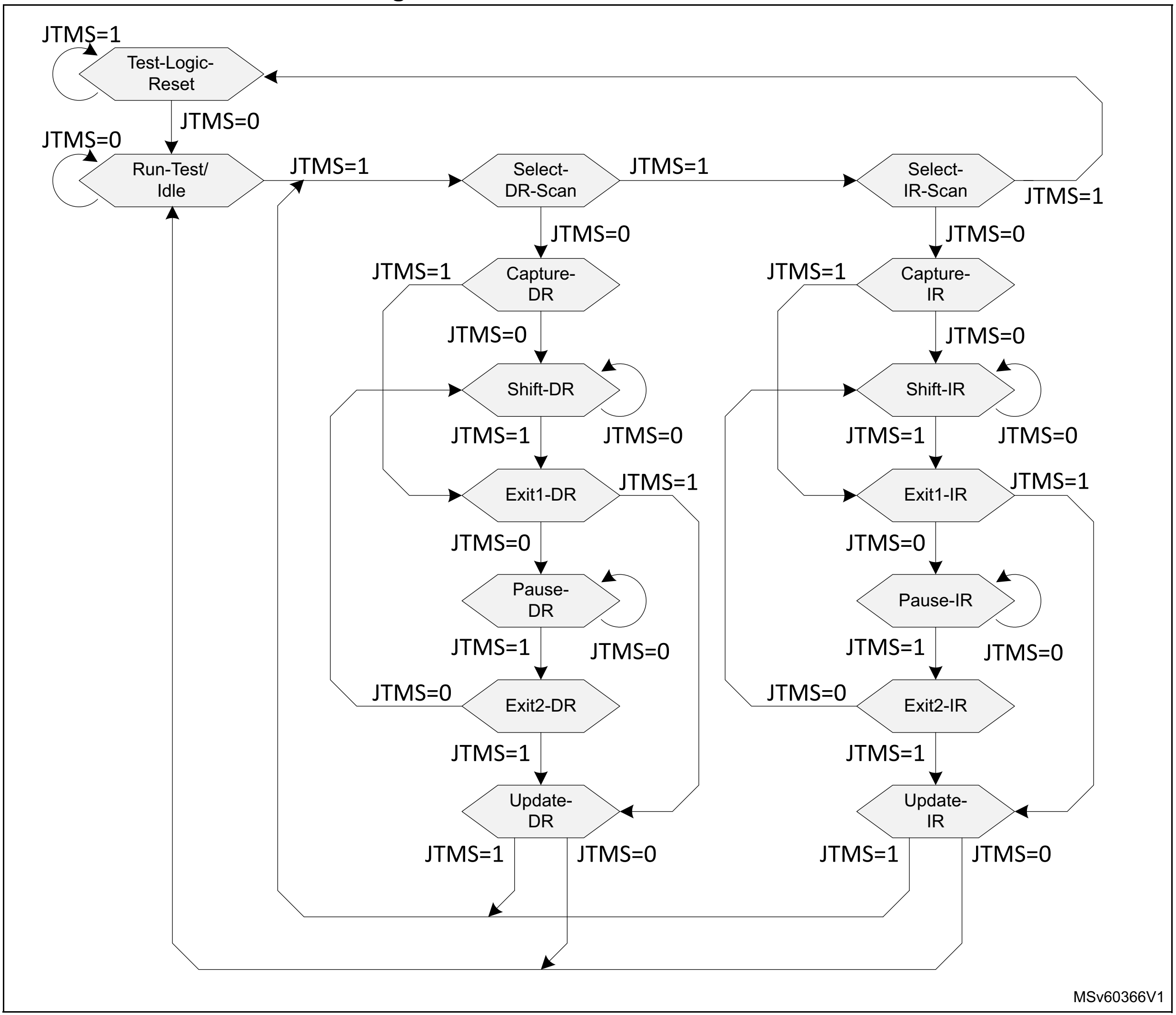
In SW-DP mode, the unused JTAG pins JTDI, JTDO and nJTRST can be used for other functions.

Note: All SWJ port I/Os can be reconfigured to other functions by software but debugging is no longer possible.

43.2.8 JTAG debug port

The JTAG debug port (JTAG-DP) implements a TAP state machine (TAPSM) 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 491. JTAG TAP state machine



The operation of the JTAG-DP is as follows:

1. When the TAPSM goes through the Capture-IR state, 0b0001 is transferred onto the instruction register (IR) scan chain. The IR scan chain is connected between JTDI and JTDO.
2. While the TAPSM is in the Shift-IR state, the IR scan chain shifts one bit for each rising edge of JTCK. This means that, on the first tick:
   – The LSB of the IR scan chain is output on JTDO.
   – Bit[n] of the IR scan chain is transferred to bit[n-1].
   – The value on JTDI is transferred to the MSB of the IR scan chain.
3. When the TAPSM goes through the Update-IR state, the value scanned into the IR scan chain is transferred into the instruction register.
4. When the TAPSM goes through the Capture-DR state, a value is transferred from one of the data registers onto one of the DR scan chains, connected between JTDI and JTDO.
5. The value held in the instruction register determines which data register and associated DR scan chain are selected.
6. This data is then shifted while the TAPSM is in the Shift-DR state, in the same manner as the IR shift in the Shift-IR state.
7. When the TAPSM goes through the Update-DR state, the value scanned into the DR scan chain is transferred into the selected data register.
8. When the TAPSM is in the Run-Test/Idle state, no special actions occur. The IDCODE instruction is loaded in the instruction register.

When active, the nJTRST signal resets the state machine asynchronously to the Test-Logic-Reset state.

The data registers corresponding to the 4-bit IR instructions are listed in *Table 419*.

Table 419. JTAG-DP data registers

IR instruction	Data register	Scan chain length	Description
0000 to 0111	(BYPASS)	1	Not implemented: BYPASS selected
1000	ABORT	35	ABORT register – Bits 34:1 = Reserved – Bit 0 = APABORT: write 1 to generate an AP abort.
1001	(BYPASS)	1	Reserved*:* BYPASS selected
1010	DPACC	35	Debug port access register Initiates the debug port and gives access to a debug port register. – When transferring data IN: Bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request Bits 2:1 = A[3:2] = 2-bit address of a debug port register Bit 0 = RnW = Read request (1) or write request (0) – When transferring data OUT: Bits 34:3 = DATA[31:0] = 32-bit data read following a read request Bits 2:0 = ACK[2:0] = 3-bit Acknowledge 010 = OK/FAULT 001 = WAIT OTHER = reserved

Table 419. JTAG-DP data registers (continued)

IR instruction	Data 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 OTHER = reserved
1100	(BYPASS)	1	Reserved*:* BYPASS selected
1101	(BYPASS)	1	Reserved*:* BYPASS selected
1110	IDCODE	32	ID 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.

Data registers are described in more detail in the Arm® Debug Interface Architecture Specification *[1]*.

43.2.9 Serial-wire debug port

The Serial-wire debug (SWD) protocol uses the two following pins:

- SWCLK: clock from host to target
- SWDIO: bi-directional serial data

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 420*)
2. acknowledge response (3 bits) transmitted by the target (see *Table 421*)
3. data transfer (33 bits) transmitted by the host (in case of a write) or target (in case of a read) (see *Table 422*)

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 420. Packet request

Bit field	Name	Description
0	Start	Must be 1.
1	APnDP	– 0: DP register access - see *Table 419* for a list of DP registers – 1: AP register access - see *Section 43.4: Access port*
2	RnW	– 0: Write request – 1: Read request
4:3	A(3:2)	Address field of the DP or AP registers
5	Parity	Single bit parity of preceding bits
6	Stop	0
7	Park	Not driven by host, must be read as 1 by the target.

Table 421. ACK response

Bit field	Name	Description
2:0	ACK	– 000: FAULT – 010: WAIT – 100: OK

Table 422. 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.

43.3 Debug port registers

Both SW-DP and JTAG-DP access the debug port (DP) registers listed in *Table 423*.

The debugger accesses 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 420*). 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 accesses the register corresponding to the content of the DP_SELECTR register.

43.3.1 DP identification register [alternate] (DP_DPIDR)

Address offset: 0x00

Reset value: 0x0BE0 2477 (SW-DP), 0x0BE0 1477 (JTAG-DP)

Read only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				PARTNO[7:0]								Res.	Res.	Res.	MIN
r	r	r	r	r	r	r	r	r	r	r	r				r

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

Bits 31:28 **REVISION[3:0]:** Revision code
0x0 (JTAG-DP)
0x0 (SW-DP)

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 not implemented (transaction counter and pushed operations are supported)

Bits 15:12 **VERSION[3:0]:** DP architecture version
0x1 (JTAG-DP)
0x2 (SW-DP)

Bits 11:1 **DESIGNER[10:0]:** JEDEC designer identity code
0x23B Arm® JEDEC code

Bit 0 Reserved, must be kept at reset value.

43.3.2 DP abort register [alternate] (DP_ABORTR)

Address offset: 0x00

Reset value: 0x0000 0000

Write only

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ORUNERRCLR	WDERRCLR	STKERRCLR	Res.	DAPABORT
											w	w	w		w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **ORUNERRCLR:** Overrrun error clear
0: No effect
1: Clear STICKYORUN bit in DP_CTRLSTATR

Bit 3 **WDERRCLR:** Write data error clear
0: No effect
1: Clear WDATAERR bit in DP_CTRLSTATR

Bit 2 **STKERRCLR:** Sticky error clear
0: No effect
1: Clear STICKYERR bit in DP_CTRLSTATR

Bit 1 Reserved, must be kept at reset value.

Bit 0 **DAPABORT:** Data AP abort
Aborts current AP transaction if an excessive number of WAIT responses are returned, indicating that the transaction is stalled.
0: No effect
1: Aborts the transaction.

43.3.3 DP control and status register (DP_CTRLSTATR)

Address offset: 0x04

and DP_SELECTR.DPBANKSEL = 0x0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	CDBGPWRUPACK	CDBGPWRUPREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		r	rw												
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATAERR	READOK	STICKYERR	Res.	Res.	Res.	STICKYORUN	ORUNDETECT
								r	r	rw				rw	rw

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

Bit 29 **CDBGPWRUPACK:** See description in *Section 43.2.7*

0 = DAPCLK gated

1 = DAPCLK enabled

Bit 28 **CDBGPWRUPREQ:** Control of DAPCLK enable request signal

0 = Requests DAPCLK gating

1 = Requests DAPCLK enabled

Bits 27:8 Reserved, must be kept at reset value.

Bit 7 **WDATAERR:** Write data error (read only) in SW-DP

There is a parity or framing error on the data phase of a write, or a write that has been accepted by the DP is then discarded without being submitted to the AP.

This bit is reset by writing 1 to the DP_ABORTR.WDERRCLR bit.

0: No error

1: An error occurred.

This bit is reserved in JTAG-DP.

Bit 6 **READOK:** AP read response (read only) in SW-DP

Indicates the response to the last AP read access.

0: Read not OK

1: Read OK

This bit is reserved in JTAG-DP.

Bit 5 **STICKYERR:** Transaction error (read only in SW-DP, R/W in JTAG-DP)

Indicates that an error occurred in an AP transaction.

0: No error

1: An error occurred.

In SW-DP, STICKYERR bit is read only, reset by writing 1 to the DP_ABORTR.STKERRCLR bit.

In JTAG-DP, STICKYERR bit is read, cleared by writing a 1 to it.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **STICKYORUN:** Overrun (read only in SW-DP, R/W in JTAG-DP)

Indicates that an overrun occurred (new transaction received before previous transaction completed). This bit is only set if the ORUNDETECT bit is set.

0: No overrun

1: An overrun occurred.

In SW-DP, STICKYORUN bit is read only, reset by writing 1 to the DP_ABORTR.ORUNERRCLR bit.

In JTAG-DP, STICKYORUN bit is read, cleared by writing a 1 to it.

Bit 0 **ORUNDETECT:** Overrun detection mode enable

0: Overrun detection disabled

1: Overrun detection enabled

In the event of an overrun, the STICKYORUN bit is set and subsequent transactions are blocked until the STICKYORUN bit is cleared.

43.3.4 DP data link control register (DP_DLCR)

Address offset: 0x04

and DP_SELECTR.DPBANKSEL = 0x1

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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
0x1: 2 data bit periods
0x2: 3 data bit periods
0x3: 4 data bit periods

Bits 7:6 **WIREMODE[1:0]:** SW-DP SWDIO
0x0: synchronous
others: reserved

Bits 5:0 Reserved, must be kept at reset value.

43.3.5 DP target identification register (DP_TARGETIDR)

Address offset: 0x04

and DP_SELECTR.DPBANKSEL = 0x2

Reset value: 0x0492 0041

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

Bits 31:28 **TREVISION[3:0]:** Target revision
0x0: revision A

Bits 27:12 **TPARTNO[15:0]:** Target part number
0x4920: STM32WBA5xxx

Bits 11:1 **TDESIGNER[10:0]:** Target designer JEDEC code.
0x020: STMicroelectronics

Bit 0 Reserved, must be kept at reset value.

43.3.6 DP data link protocol identification register (DP_DLPIDR)

Address offset: 0x04

and DP_SELECTR.DPBANKSEL = 0x3

Reset value: 0x0000 0001

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

Bits 31:28 **TINSTANCE[3:0]:** Target instance number
Defines the instance number for this device in a multi-drop system.
0x0: Instance number 0

Bits 27:4 Reserved, must be kept at reset value.

Bits 3:0 **PROTSVN[3:0]:** Serial-wire debug protocol version
0x1: Version 2

43.3.7 DP resend register (DP_EVENSTATR)

Address offset: 0x04

and DP_SELECTR.DPBANKSEL = 0x4

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.	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.

43.3.8 DP resend register (DP_RESENDR)

Address offset: 0x08

Reset value: 0x0000 0000

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

Bits 31:0 **RESEND[31:0]:** Value returned by the last AP read or DP_RDBUFFR read.
Used in the event of a corrupted read transfer.

43.3.9 DP access port select register (DP_SELECTR)

Address offset: 0x08

Reset value: 0xXXXX XXXX

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

Bits 31:28 **APSEL[3:0]:** Access port selection
Selects the access port for the next transaction.
0x0: AP0 - System debug access port
0x1: AP1 - Cortex-M33 debug access port
others: reserved

Bits 27:8 Reserved, must be kept at reset value.

Bits 7:4 **APBANKSEL[3:0]:** AP register bank selection
Selects the 4-word register bank on the active AP for the next transaction.

Bits 3:0 **DPBANKSEL[3:0]:** DP register bank selection
Selects the register at address 0x4 of the debug port.
0x0: DP_CTRLSTATR
0x1: DP_DLCR
0x2: DP_TARGETIDR
0x3: DP_DLPIDR
0x4: DP_EVENTSTAT
others: reserved

43.3.10 DP read buffer register (DP_BUFFR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bits 31:0 **RDBUFF[31:0]:** Contains the value returned by the last AP read access.

The value returned by an AP read access can either be obtained using a second read access to the same address, which initiates a new transaction on the corresponding bus, or else it can be read from this register, in which case no new AP transaction occurs.

43.3.11 DP register map and reset values

These registers are not on the CPU memory bus and are accessed only 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 423. DP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DP_DPIDR	REVISION[3:0]				PARTNO[7:0]								Res.	Res.	Res.	MIN	VERSION[3:0]				DESIGNER[10:0]											Res.
	Reset value	0	0	0	0	1	0	1	1	1	1	1	0				0	X	X	X	X	0	1	0	0	0	1	1	1	0	1	1	
0x00	DP_ABORTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ORUNERRCLR	WDERRCLR	STKERRCLR	Res.	DAPABORT
	Reset value																												0	0	0		0
0x04(1)	DP_CTRLSTATR	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
0x04(2)	DP_DLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TURNROUND[1:0]		WIREMODE[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																							0	0	0	0						

Table 423. DP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x04(3)	DP_TARGETIDR	TREVISION[3:0]				TPARTNO[15:0]																TDESIGNER[10:0]											Res.
	Reset value	0	0	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
0x04(4)	DP_DLPIDR	TINSTANCE[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROTSVN[3:0]			
	Reset value	0	0	0	0																									0	0	0	1
0x04(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																																0
0x08	DP_RESENDR	RESEND[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	DP_SELECTR	APSEL[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBANKSEL[3:0]				DPBANKSEL[3:0]			
	Reset value	X	X	X	X																					X	X	X	X	X	X	X	X
0x0C	DP_BUFFR	RDBUFF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. DP_SELECTR.DPBANKSEL = 0x0.
2. DP_SELECTR.DPBANKSEL = 0x1.
3. DP_SELECTR.DPBANKSEL = 0x2.
4. DP_SELECTR.DPBANKSEL = 0x3.
5. DP_SELECTR.DPBANKSEL = 0x4

Refer to *Section 43.3: Debug port registers* for the register boundary addresses.

43.4 Access port

There are two access ports (AP) attached to the DP.

- AP0, System debug access port: enables access to the system debug features integrated at device level.
- AP1, Cortex-M33 processor core access port: enables access to the debug and trace features integrated in the Cortex-M33 processor core.

43.4.1 Access port registers

The access ports are of MEM-AP type: the debug and trace component registers are mapped in the address space of the associated bus.

An 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 424*.

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_SELECTR register
- Bits [3:2]: content of the A(3:2) field of the APACC data register in the JTAG-DP or of the SW-DP packet request, depending on the debug interface used
- Bits [1:0]: always set to 0

The content of the APSEL[3:0] defines which MEM-AP is being accessed.

The debugger can access the AP registers as follows:

1. Program in the APSEL[3:0] field to choose the AP and the APBANKSEL[3:0] field to select the register bank to be accessed.
2. Program the A(3:2) field in the APACC register, if using JTAG, with the register address within the bank. Program the RnW bit to select a read or write. In the case of a write, program the DATA field with the write data. If using SWD, the A(3:2) and RnW fields are part of the packet request word sent to the SW-DP with the APnDP bit set (see *Table 420*). The write data is sent in the data phase.

The debugger can access the memory mapped debug component registers through the MEM-AP registers (using the above AP register access procedure) as follows:

1. Program the transaction target address in the AP_TAR register.
2. Program the AP_CSWR register, if necessary, with the transfer parameters (AddrInc for example).
3. Write to or read from the AP_DRWR register to initiate a bus transaction at the address held in the AP_TAR register. Alternatively, a read or write to the AP_BDxR register triggers an access to address AP_TAR[31:4] + x (allowing up to four consecutive addresses to be accessed without changing the address in the AP_TAR register).

For more detailed information on the MEM-AP, refer to the Arm® Debug Interface Architecture Specification *[1]*.

43.4.2 AP control/status word register (APx_CSWR) (x = 0 to 1)

Address offset: 0x00

Reset value: 0x0100 00X0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PROT6	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.	DBG STATUS	ADDRINC[1:0]		Res.	SIZE[2:0]		
									r	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **PROT6:** Secure transfer protection bit [6]
This field sets protection attribute HPROT[6] of the AP 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 protection attribute HPROT[3:0] of the AP bus transfer.
0bXXX1: data transfer
0bXX0X: unprivileged transfer
0bXX1X: privileged transfer
0bX0XX: non-bufferable
0bX1XX: bufferable
0b0XXX: non-shareble, no look-up non-modifiable
0b1XXX: sharable, look-up, modifiable

Bits 23:7 Reserved, must be kept at reset value.

Bit 6 **DBGSTATUS:** Device enable (DEVICEEN) status
Indicates whether the Cortex-M33 AP bus can be accessed by the debugger. This bit reflects the state of dbgen.
0: AP AHB bus transfers blocked.
1: AP AHB bus transfers granted.

Bits 5:4 **ADDRINC[1:0]:** Auto-increment mode
Defines whether AP_TAR address is automatically incremented after a transaction.
0x0: No auto-increment
0x1: Address is incremented by the size in bytes of the transaction (SIZE field).
others: 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

43.4.3 AP transfer address register (APx_TAR) (x = 0 to 1)

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

43.4.4 AP data read/write register (APx_DRWR) (x = 0 to 1)

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

43.4.5 AP banked data registers y (APx_BDyR) (x = 0 to 1)

Address offset: 0x10 + 0x4 * y, (y = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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 in AP_TAR
AP_TAR + AP_BDnR address [3:2] + 0b00
The auto address incrementing is not performed on AP_BD[3:0]R.
Banked transfers are only supported for word transfers.

43.4.6 AP configuration register (APx_CFGR) (x = 0 to 1)

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 longer than 32-bits supported.

Bit 1 **LA:** Long address
0: Physical addresses not longer than 32-bits supported.

Bit 0 **BE:** Big-endian
0: only little-endian supported.

43.4.7 AP base address register (APx_BASER) (x = 0 to 1)

Address offset: 0xF8

Reset value: Block 0: 0xE008 0003

Reset value: Block 1: 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	ENTRY PRESENT
r	r	r	r											r	r

Bits 31:12 **BASEADDR[31:12]:** Base address (bits 31 to 12) of ROM table for the AP
The 12 LSBs are zero since the ROM table must be aligned on a 4-Kbyte boundary.
AP0: 0xE0080 (system ROM table)
AP1: 0xE00FE (MCU ROM table)

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 are present

43.4.8 AP identification register (APx_IDR) (x = 0 to 1)

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
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]:** Memory access port
0x8: MEM_AP Standard register map

Bits 12:8 Reserved, must be kept at reset value.

Bits 7:4 **VARIANT[3:0]:** AP variant
0x1: AHB-AP

Bits 3:0 **TYPE[3:0]:** AP type
0x5: AHB5

43.4.9 Access port register map and reset values

These registers are not on the CPU memory bus and are only accessed through SW-DP and JTAG-DP debug interface.

The access port address is 8-bit wide, defined by DP_SELECTR.APBANKSEL[3:0] field and by JTAG-DP register DPACC or SW-DP packet request A[3:2] field.

Table 424. AP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	AP1_CSWR	Res.	SPROT[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	0	0		0	0	0
0x04	APx_TAR	TA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	Reserved	Reserved																															
0x0C	APx_DRWR	TD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	APx_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	APx_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	APx_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	APx_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	APx_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	AP0_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	0	0	0	0	0	0	0											1	1
0xF8	AP1_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	APx_IDR	REViISION [3:0]				JEDECBANK [3:0]				JEDECCODE[6:0]							CLASS[3:0]				Res.	Res.	Res.	Res.	Res.	VARIANT[3:0]				TYPE[3:0]			
	Reset value	0	0	0	1	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0						0	0	0	1	0	1	0	1

Refer to *Section 43.4: Access port* for the register boundary addresses.

43.5 System debug AP0 features

The system debug subsystem integrates the following features accessible through the AP0:

- system ROM table
- debug MCU controller (DBGMCU)

43.5.1 System debug ROM table

The system debug ROM table is a CoreSight component that contains the base addresses of all the debug components accessible via the AP0. This table allows a debugger to discover the topology of the debug system automatically.

There is one ROM table in the system debug sub-system.

1. The system debug ROM table is pointed to by the AP0_BASER register. It contains the base-address pointer for the DBGMCU.

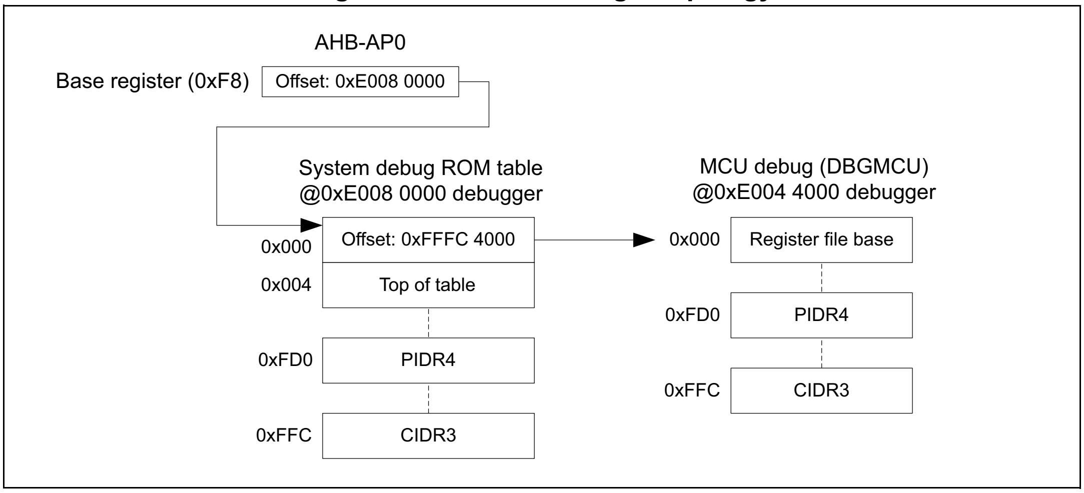
The system debug ROM table (see *Table 425*) occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE008 0000 to 0xE008 0FFC.

Table 425. System debug ROM table

Address offset in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0x000	DBGMCU	0xE004 4000	0xFFFC 4000	4	0xFFFC 4003
0x004	Top of table	-	-	-	0x0000 0000
0x00C to 0xFC8	Reserved	-	-	-	0x0000 0000
0xFCC to 0xFFC	ROM table registers	-	-	-	See *Table 429*

The CoreSight topology for the debug components in the system debug sub-system is shown in *Figure 492*.

Figure 492. AP0: CoreSight topology



43.5.2 System debug memory type register (SYSROM_MEMTYPER)

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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

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

Bit 0 **SYSMEM:** system memory
0: No system memory present on this bus

43.5.3 Sytem debug CoreSight peripheral identity register 4 (SYSROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **F4KCOUNT[3:0]:** Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]:** JEP106 continuation code
0x0: STMicroelectronics JEDEC continuation code

43.5.4 System debug CoreSight peripheral identity register 0 (SYSROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0092

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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]
0x92: STM32WBA5xxx

43.5.5 System debug CoreSight peripheral identity register 1 (SYSROM_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: MCU ROM: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0x4: MCU ROM: STM32WBA5xxx

43.5.6 System debug CoreSight peripheral identity register 2 (SYSROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

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

Bits 7:4 **REVISION[3:0]:** Component revision number
0x0: rev r0p0

Bit 3 **JEDEC:** JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]:** JEP106 identity code bits [6:4]
0x2: STMicroelectronics JEDEC code

43.5.7 System debug CoreSight peripheral identity register 3 (SYSROM_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

43.5.8 System debug CoreSight component identity register 0 (SYSROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

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

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

Bits 7:0 **PREAMBLE[7:0]:** Component ID bits [7:0]
0x0D: Common ID value

43.5.9 System debug CoreSight peripheral identity register 1 (SYSROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

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

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

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

Bits 7:4 **CLASS[3:0]:** Component ID bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]:** Component ID bits [11:8]
0x0: Common ID value

43.5.10 System debug CoreSight component identity register 2 (SYSROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

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

Bits 7:0 **PREAMBLE[19:12]:** Component ID bits [23:16]
0x05: Common ID value

43.5.11 System debug CoreSight component identity register 3 (SYSROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

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

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

Bits 7:0 **PREAMBLE[27:20]:** Component ID bits [31:24]
0xB1: Common ID value

43.5.12 System debug ROM table register map and reset values

Table 426. AP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	ROM_MEMTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																0
0xFD0	SYSROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4-0xFDC	Reserved	Reserved.																															
0xFCC	ROM_MEMTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																0
0xFD0	SYSROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4-0xFDC	Reserved	Reserved.																															
0xFE0	SYSROM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									1	0	0	1	0	0	1	0
0xFE4	SYSROM_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	SYSROM_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	SYSROM_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	SYSROM_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	SYSROM_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	SYSROM_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	SYSROM_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 43.5: System debug AP0 features* for the register boundary addresses.

43.6 Cortex-M33 AP1 features

The Cortex-M33 subsystem integrates the following features accessible through the AP1:

- MCU and processor ROM tables
- system control space (SCS)
- breakpoint unit (FPB)
- data watchpoint and trace unit (DWT)
- instrumental trace macrocell (ITM)
- trace port interface unit (TPIU)
- cross trigger interface (CTI)

43.6.1 CPU ROM tables

The ROM tables are CoreSight components that contain the base addresses of all the debug components accessible via the AP1. These tables allow the debugger to discover the CoreSight topology of the system automatically.

There are two ROM tables on AP1 in the Cortex-M33 sub-system.

1. The MCU ROM table is pointed to by the AP_BASER register in the AP1. It contains the base-address pointers for the processor ROM table and for the TPIU and DBGMCU.
2. 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, ITM and CTI.

The MCU ROM table occupies a 4-Kbyte, 32-bit wide chunk of address space, at base address 0xE00F E000.

Table 427. MCU ROM table

Address offset in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0x000	Processor ROM table	0xE00F F000	0x0000 1000	4	0x0000 1003
0x004	TPIU	0xE004 0000	0xFFF4 2000	4	0xFFF4 2003
0x008	DBGMCU	0xE004 4000	0xFFF4 6000	4	0xFFF4 6003
0x00C	Reserved	-	-	-	0x1FF0 2002
0x010	Top of table	-	-	-	0x0000 0000
0x014 to 0xFC8	Reserved	-	-	-	0x0000 0000
0xFCC to 0xFFC	ROM table registers	-	-	-	See *Table 429*

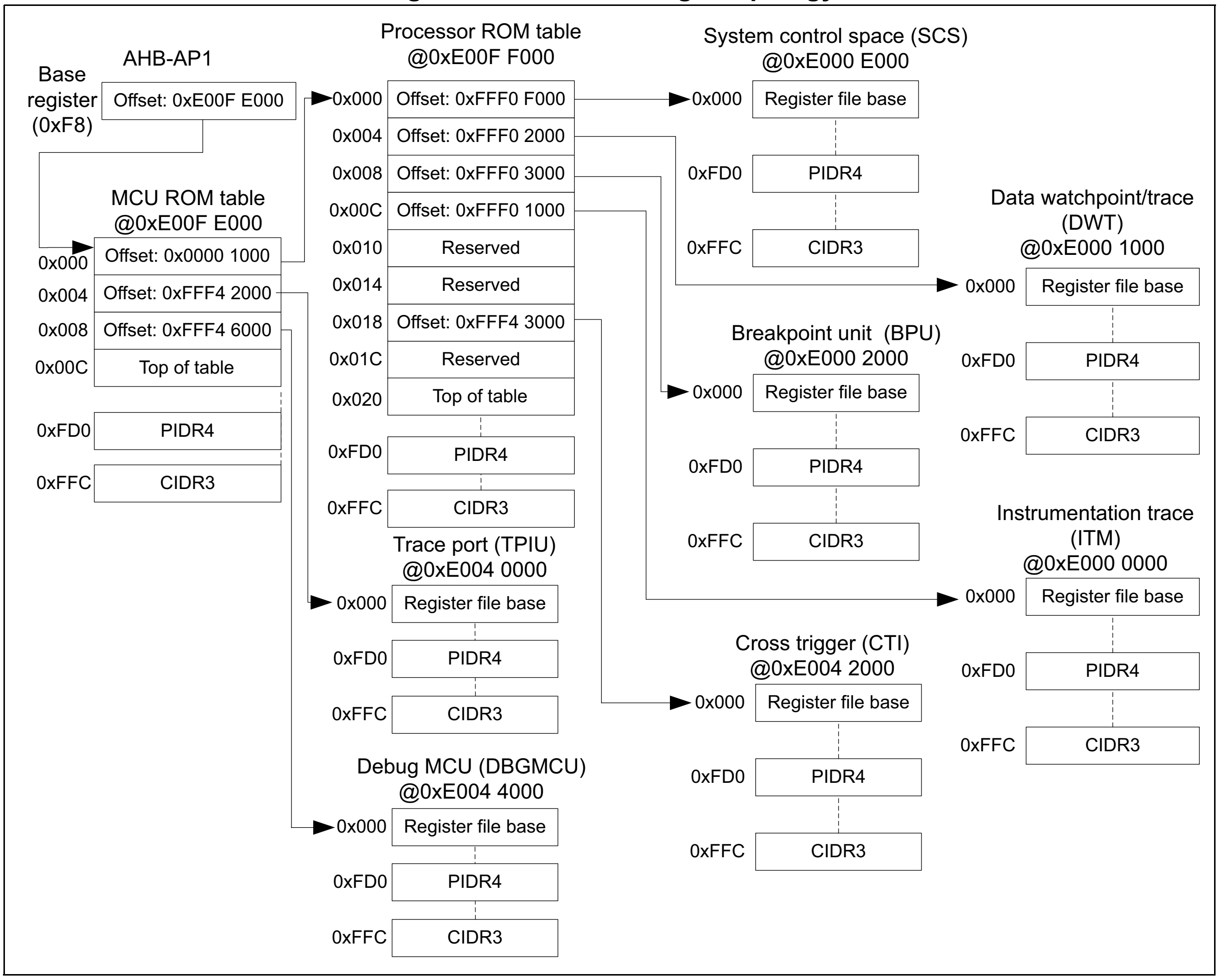
The processor ROM table occupies a 4-Kbyte, 32-bit wide chunk of address space, at base address 0xE00F F000.

Table 428. Processor ROM table

Address in ROM table	Component name	Component base address	Component address offset	Size	Entry
0x000	SCS	0xE000 E000	0xFFF0 F000	4 KB	0xFFF0 F003
0x004	DWT	0xE000 1000	0xFFF0 2000	4 KB	0xFFF0 2003
0x008	BPU	0xE000 2000	0xFFF0 3000	4 KB	0xFFF0 3003
0x00C	ITM	0xE000 0000	0xFFF0 1000	4 KB	0xFFF0 1003
0x010	Reserved	-	-	-	0xFFF4 1002
0x014	Reserved	-	-	-	0xFFF4 2002
0x018	CTI	0xE004 2000	0xFFF4 3000	4 KB	0xFFF4 3003
0x01C	Reserved	-	-	-	0xFFF4 4002
0x020	Top of table	-	-	-	0x0000 0000
0x024 to 0xFC8	Reserved	-	-	-	0x0000 0000
0xFCC to 0xFFC	ROM table registers	-	-	-	See *Table 429*

The CoreSight topology for the debug components in the CPU sub-system is shown in *Figure 493*.

Figure 493. CPU CoreSight topology



43.6.2 MCU and processor ROM memory type register (ROM_MEMTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0001

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

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

Bit 0 **SYSMEM:** System memory
1: System memory present on this bus

43.6.3 MCU and processor ROM CoreSight peripheral identity register 4 (ROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000 (MCU ROM)

Reset value: 0x0000 0004 (processor ROM)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **F4KCOUNT[3:0]:** Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]:** JEP106 continuation code
MCU ROM:
0x0: STMicroelectronics JEDEC continuation code
Processor ROM:
0x4: Arm® JEDEC continuation code

43.6.4 MCU and processor ROM CoreSight peripheral identity register 0 (ROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0092 (MCU ROM)

Reset value: 0x0000 00C9 (processor ROM)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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]
MCU ROM:
0x92: STM32WBA5xxx
processor ROM:
0xC9: Cortex-M33

43.6.5 MCU and processor ROM CoreSight peripheral identity register 1 (ROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 0004 (MCU ROM)

Reset value: 0x0000 00B4 (processor ROM)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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: MCU ROM: STMicroelectronics JEDEC code
0xB: processor ROM: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0x4: MCU ROM: STM32WBA5xxx
0x4: processor ROM: Cortex-M33

43.6.6 MCU and processor ROM CoreSight peripheral identity register 2 (ROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A (MCU ROM)

Reset value: 0x0000 000B (processor ROM)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
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]
MCU ROM:
0x2: STMicroelectronics JEDEC code
processor ROM:
0x3: Arm® JEDEC code

43.6.7 MCU and processor ROM CoreSight peripheral identity register 3 (ROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

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

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

Bits 7:4 **REVAND[3:0]:** Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]:** Customer modified
0x0: No customer modifications

43.6.8 MCU and processor ROM CoreSight component identity register 0 (ROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

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

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

Bits 7:0 **PREAMBLE[7:0]:** Component ID bits [7:0]
0x0D: Common ID value

43.6.9 MCU and processor ROM CoreSight peripheral identity register 1 (ROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

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

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

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

Bits 7:4 **CLASS[3:0]:** Component ID bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]:** Component ID bits [11:8]
0x0: Common ID value

43.6.10 MCU and processor ROM CoreSight component identity register 2 (ROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

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

Bits 7:0 **PREAMBLE[19:12]:** Component ID bits [23:16]
0x05: Common ID value

43.6.11 MCU and processor ROM CoreSight component identity register 3 (ROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

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

Bits 7:0 **PREAMBLE[27:20]:** Component ID bits [31:24]
0xB1: Common ID value

43.6.12 MCU and processor ROM tables register map and reset values

Table 429. ROM table register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	**ROM_MEMTYPER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																1
0xFD0	**ROM_PIDR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT [3:0]				JEP106CON [3:0]			
	MCU Reset value																									0	0	0	0	0	0	0	0
	processor Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved	Reserved.																															
0xFE0	**ROM_PIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	MCU Reset value																									1	0	0	1	0	0	1	0
	processor Reset value																									1	1	0	0	1	0	0	1
0xFE4	**ROM_PIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	MCU Reset value																									0	0	0	0	0	1	0	0
	processor Reset value																									1	0	1	1	0	1	0	0
0xFE8	**ROM_PIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	MCU Reset value																									0	0	0	0	1	0	1	0
	processor Reset value																									0	0	0	0	1	0	1	1
0xFEC	**ROM_PIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	**ROM_CIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	**ROM_CIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	**ROM_CIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	**ROM_CIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 43.6: Cortex-M33 AP1 features* for the register boundary addresses.

43.7 Data watchpoint and trace unit (DWT)

The DWT provides four comparators that can be used as one of the following function:

- watchpoint
- PC sampling trigger
- data address sampling trigger
- data comparator (comparator 1 only)
- clock cycle counter comparator (comparator 0 only)

It also contains counters for:

- clock cycles
- folded instructions
- load store unit (LSU) operations
- sleep cycles
- number of cycles per instruction
- interrupt overhead

A DWT comparator compares the value held in its DWT_COMPxR registers with one of the following:

- a data address
- an instruction address
- a data value
- the cycle count value, for comparator 0 only.

For address matching, the comparator can use a mask, so it matches a range of addresses.

On a successful match, the comparator generates one of the following:

- one or more DWT data trace packets, containing one or more of:
  - the address of the instruction that caused a data access
  - an address offset, bits[15:0] of the data access address
  - the matched data value
- a watchpoint debug event, on either the PC value or the accessed data address
- a CMPMATCH[N] event that signals the match outside the DWT unit

A watchpoint debug event either generates a DebugMonitor exception or causes the processor to halt execution and enter Debug state.

For more details on how to use the DWT, refer to the Armv8-M Architecture Reference Manual *[4]*.

43.7.1 DWT control register (DWT_CTRLR)

Address offset: 0x000

Reset value: 0x4000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCOMP[3:0]				NOTRCPKT	NOEXTTRIG	NOCYCCNT	NOPRFCNT	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]				POSTPRESET[3:0]				CYCCNTENA
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **NUMCOMP[3:0]:** Number of comparators implemented (read only)
0x4: Four comparators

Bit 27 **NOTRCPKT:** Trace sampling and exception tracing support (read only)
0: Supported

Bit 26 **NOEXTTRIG:** External match signal, CMPMATCH support (read only)
0: Supported

Bit 25 **NOCYCCNT:** Cycle counter support (read only)
0: Supported

Bit 24 **NOPRFCNT:** Profiling counter support (read only)
0: Supported

Bit 23 **CYCDISS:** Cycle counter disable secure
Controls whether the cycle counter is disabled in secure mode
0: No effect
1: Disabled incrementing the cycle counter when the processor is in secure mode

Bit 22 **CYCEVTENA:** Enable for POSTCNT underflow event counter packet generation
0: Disabled
1: Enabled

Bit 21 **FOLDEVTENA:** Enable for folded instruction counter overflow event generation
0: Disabled
1: Enabled

Bit 20 **LSUEVTENA:** Enable for LSU counter overflow event generation
0: Disabled
1: Enabled

Bit 19 **SLEEPEVTENA:** Enable for sleep counter overflow event generation
0: Disabled
1: Enabled

Bit 18 **EXCEVTENA:** Enable for exception overhead counter overflow event generation
0: Disabled
1: Enabled

Bit 17 **CPIEVTENA:** Enable for CPI counter overflow event generation
0: Disabled
1: Enabled

Bit 16 **EXCTRCENA:** Enable for exception trace generation
0: Disabled
1: Enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PCSAMPLENA:** Enable for POSTCNT counter used as a timer for periodic PC sample packet generation
0: Disabled
1: Enabled

Bits 11:10 **SYNCTAP[1:0]:** synchronization packet counter tap
Selects the position of the synchronization packet counter tap on the CYCCNT counter. This determines the synchronization packet rate.
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 must be reset prior to writing POSTINIT).

Bits 4:1 **POSTPRESET[3:0]:** Reloads value of the POSTCNT counter.

Bit 0 **CYCCNTENA:** Enable for CYCCNT counter
0: Disabled
1: Enabled

43.7.2 DWT cycle count register (DWT_CYCCNTR)

Address offset: 0x004

Reset value: 0x0000 0000

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

Bits 31:0 **CYCCNT[31:0]:** processor clock cycle counter

43.7.3 DWT CPI count register (DWT_CPICNTR)

Address offset: 0x008

Reset value: 0x0000 0000

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

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

Bits 7:0 **CPICNT[7:0]:** CPI counter
Counts additional cycles required to execute multi-cycle instructions (except those recorded by DWT_LSUCNTR) and counts any instruction fetch stalls.

43.7.4 DWT exception count register (DWT_EXCCNTR)

Address offset: 0x00C

Reset value: 0x0000 0000

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

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

Bits 7:0 **EXCCNT[7:0**]: exception overhead cycle counter
Counts the number of cycles spent in exception processing.

43.7.5 DWT sleep count register (DWT_SLPCNTR)

Address offset: 0x010

Reset value: 0x0000 0000

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

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

Bits 7:0 **SLEEPCNT[7:0]:** sleep cycle counter
Counts the number of cycles spent in sleep mode (WFI, WFE, sleep-on-exit).

43.7.6 DWT LSU count register (DWT_LSUCNTR)

Address offset: 0x014

Reset value: 0x0000 0000

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

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

Bits 7:0 **LSUCNT[7:0]:** load store counter
Counts additional cycles required to execute load and store instructions.

43.7.7 DWT fold count register (DWT_FOLDCNTR)

Address offset: 0x018

Reset value: 0x0000 0000

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

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

Bits 7:0 **FOLDCNT[7:0]:** folded instruction counter
Increments on each instruction that takes 0 cycles.

43.7.8 DWT program counter sample register (DWT_PCSR)

Address offset: 0x01C

Reset value: 0x0000 0000

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

Bits 31:0 **EIASAMPLE[31:0]:** executed Instruction Address sample value
Samples the current value of the program counter.

43.7.9 DWT comparator register x (DWT_COMPxR)

Address offset: 0x020 + 0x10 * x, (x = 0 to 3)

Reset value: 0x0000 0000

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

Bits 31:0 **COMP[31:0]:** reference value for comparison

43.7.10 DWT function register 0(DWT_FUNCTR0)

Address offset: 0x028

Reset value: 0x5800 0000

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

Bits 31:27 **ID[4:0]**: capability identification
Identifies the capability for match for comparator 0.
0b01011: Cycle Counter, Instruction Address, Data Address and Data Address With Value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: comparator match
Indicates if a comparator match has occurred since the register was last read.
0: No match
1: A match occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: Data value size
Defines the size of the object being watched for by Data Value and Data Address comparators.
0x0: 1 byte
0x1: 2 bytes
0x2: 4 bytes
0x3: reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: Action on match
0x0: trigger only
0x1: generate debug event
0x2: For a Cycle Counter, Instruction Address, Data Address, Data Value or Linked Data Value comparator, generate a Data Trace Match packet. For a Data Address With Value comparator, generate a Data Trace Data Value packet.
0x3: For a Data Address Limit comparator, generate a Data Trace Data Address packet. For a Cycle Counter, Instruction Address Limit, or Data Address comparator, generate a Data Trace PC Value packet. For a Data Address With Value comparator, generate both a Data Trace PC Value packet and a Data Trace Data Value packet.

Bits 3:0 **MATCH[3:0]**: Match type
Controls the type of match generated by comparator 0.
For possible values of this field, refer to [4].

DWT function register 1 (DWT_FUNCTR1)

Address offset: 0x038

Reset value: 0xD000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r			r								

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

Bits 31:27 **ID[4:0]**: capability identification
Identifies the capability for match for comparator 1.
0b11010: Instruction Address, Instruction Address Limit, Data Address, Data Address Limit, and Data Address With Value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: Comparator match
Indicates if a comparator match has occurred since the register was last read.
0: no match
1: a match occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: Data value size
Defines the size of the object being watched for by Data Value and Data Address comparators.
0x0: 1 byte
0x1: 2 bytes
0x2: 4 bytes
0x3: reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: Action on match
0x0: trigger only
0x1: generate debug event
0x2: For a Cycle Counter, Instruction Address, Data Address, Data Value or Linked Data Value comparator, generate a Data Trace Match packet. For a Data Address With Value comparator, generate a Data Trace Data Value packet.
0x3: For a Data Address Limit comparator, generate a Data Trace Data Address packet. For a Cycle Counter, Instruction Address Limit, or Data Address comparator, generate a Data Trace PC Value packet. For a Data Address With Value comparator, generate both a Data Trace PC Value packet and a Data Trace Data Value packet.

Bits 3:0 **MATCH[3:0]**: Match type
Controls the type of match generated by comparator 1.
For possible values of this field, refer to *[4]*.

DWT function register 2 (DWT_FUNCTR2)

Address offset: 0x048

Reset value: 0x5000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r			r								

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

Bits 31:27 **ID[4:0]**: capability identification
Identifies the capability for MATCH for comparator 2
0b01010: Instruction Address, Data Address, and Data Address With Value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: comparator match

Indicates if a comparator match has occurred since the register was last read.
0: no match
1: a match occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: Data value size:

Defines the size of the object being watched for by Data Value and Data Address comparators.
0x0: 1 byte
0x1: 2 bytes
0x2: 4 bytes
0x3: reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: Action on match

0x0: trigger only
0x1: Generate debug event
0x2: For a Cycle Counter, Instruction Address, Data Address, Data Value or Linked Data Value comparator, generate a Data Trace Match packet. For a Data Address With Value comparator, generate a Data Trace Data Value packet.
0x3: For a Data Address Limit comparator, generate a Data Trace Data Address packet. For a Cycle Counter, Instruction Address Limit, or Data Address comparator, generate a Data Trace PC Value packet. For a Data Address With Value comparator, generate both a Data Trace PC Value packet and a Data Trace Data Value packet.

Bits 3:0 **MATCH[3:0]**: Match type

Controls the type of match generated by comparator 2.
For possible values of this field, refer to *[4]*.

DWT function register 3 (DWT_FUNCTR3)

Address offset: 0x058

Reset value: 0xF000 0000

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

Bits 31:27 **ID[4:0]**: capability identification

Identifies the capability for MATCH for comparator 2.
0b11110: Instruction Address, Instruction Address Limit, Data Address, Data Address Limit, Data value, Linked Data Value, and Data Address With Value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: comparator match

Indicates if a comparator match has occurred since the register was last read.
0: no match
1: a match occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: Data value size

Defines the size of the object being watched for by Data Value and Data Address comparators.
0x0: 1 byte
0x1: 2 bytes
0x2: 4 bytes
0x3: reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: Action on match

0x0: trigger only
0x1: Generate debug event
0x2: For a Cycle Counter, Instruction Address, Data Address, Data Value or Linked Data Value comparator, generate a Data Trace Match packet. For a Data Address With Value comparator, generate a Data Trace Data Value packet.
0x3: For a Data Address Limit comparator, generate a Data Trace Data Address packet. For a Cycle Counter, Instruction Address Limit, or Data Address comparator, generate a Data Trace PC Value packet. For a Data Address With Value comparator, generate both a Data Trace PC Value packet and a Data Trace Data Value packet.

Bits 3:0 **MATCH[3:0]**: Match type

Controls the type of match generated by comparator 2.
For possible values of this field, refer to *[4]*.

43.7.11 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®.

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

43.7.12 DWT device type register 4 (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

43.7.13 DWT CoreSight peripheral identity register 4 (DWT_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **F4KCOUNT[3:0]:** Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]:** JEP106 continuation code
0x4: Arm® JEDEC code

43.7.14 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: DWT part number

43.7.15 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: DWT part number

43.7.16 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 ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]:** JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

43.7.17 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

43.7.18 DWT CoreSight component identity register 0 (DWT_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

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

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

Bits 7:0 **PREAMBLE[7:0]:** Component ID bits [7:0]
0x0D: Common ID value

43.7.19 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 ID bits [15:12] - component class
0x9: Debug component

Bits 3:0 **PREAMBLE[11:8]:** Component ID bits [11:8]
0x0: Common ID value

43.7.20 DWT CoreSight component identity register 2 (DWT_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

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

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

Bits 7:0 **PREAMBLE[19:12]:** Component ID bits [23:16]
0x05: Common ID value

43.7.21 DWT CoreSight component identity register 3 (DWT_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

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

Bits 7:0 **PREAMBLE[27:20]:** Component ID bits [31:24]
0xB1: Common ID value

43.7.22 DWT register map and reset values

Table 430. 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	POSINIT[3:0]				POSTPRESET [3:0]				CYCCNTENA
	Reset value	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	DWT_CYCCNTR	CYCCNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	DWT_CPICNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPICNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x00C	DWT_EXCCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x010	DWT_SLPCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEPCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x014	DWT_LSUCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSUCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x018	DWT_FOLDCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FOLDCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x01C	DWT_PCSR	EIASAMPLE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	DWT_COMP0R	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	Reserved	Reserved.																															
0x028	DWT_FUNCT0R	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	0	0	0	0	0	0	0	0	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	Reserved	Reserved.																															
0x038	DWT_FUNCT1R	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	0	0	0	0	0	0	0	0	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	Reserved	Reserved.																															

Table 430. DWT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x048	**DWT_FUNCT2R**	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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	Reserved	Reserved.																															
0x058	**DWT_FUNCT3R**	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**	ARCHTITECT[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	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[3:8]			
	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.	F4KCOUNT [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFC4	Reserved	Reserved.																															
0xFE0	**DWT_PIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	**DWT_PIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	**DWT_PIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	**DWT_PIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
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 427: MCU ROM table* for the register boundary addresses.

43.8 Instrumentation trace macrocell (ITM)

The ITM generates trace information as packets. There are three sources that can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The three sources in decreasing order of priority are:

1. Software trace

   Software can write directly to any of 32 x 32-bit ITM stimulus registers to generate packets. The permission level for each port can be programmed. When software writes to an enabled stimulus port, the ITM combines the identity of the port, the size of the write access and the data written, into a packet that it writes to a FIFO. The ITM outputs packets from the FIFO onto the trace bus. Reading a stimulus port register returns the status of the stimulus register (empty or pending) in bit 0.
2. Hardware trace

   The DWT generates trace packets in response to a data trace event, a PC sample or a performance profiling counter wraparound. The ITM outputs these packets on the trace bus.
3. Local timestamping

   The ITM contains a 21-bit counter clocked by the (pre-divided) processor clock. The counter value is output in a timestamp packet on the trace bus. The counter is reset to zero every time a timestamp packet is generated. The timestamps thus indicate the time elapsed since the previous timestamp packet.

For more information on the ITM and how to use it, refer to the Armv8-M Architecture Reference Manual *[4]*.

43.8.1 ITM registers

43.8.2 ITM stimulus register x (ITM_STIMRx)

Address offset: 0x000 + 0x4 * 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[15:2]														STIMULUS1	STIMULUS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	rw	rw

Bits 31:2 **STIMULUS[31:2]:** Trace output data

When writing, written data is output on the trace bus as a software event packet.

Bit 1 **STIMULUS1:** Trace output data bit [1]

When writing, written data is output on the trace bus as a software event packet.
When reading: disable flag:
0: Stimulus port and ITM enabled.
1: Stimulus port and ITM disabled.

Bit 0 **STIMULUS0:** Trace output data bit [0]
When writing, written data is output on the trace bus as a software event packet.
When reading: FIFO ready indicator:
0: Stimulus port buffer is full (or port is disabled).
1: Stimulus port can accept new write data.

43.8.3 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]:** Enable for stimulus port
Each bit x enables the stimulus port associated with the ITM_STIMRx register.
0: Port disabled
1: Port enabled

43.8.4 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]:** Enables unprivileged access to ITM stimulus ports.
Each bit controls eight stimulus ports.
0bXXX0: Unprivileged access permitted on ports 0 to 7
0bXXX1: Only privileged access permitted on ports 0 to 7
0bXX0X: Unprivileged access permitted on ports 8 to 15
0bXX1X: Only privileged access permitted on ports 8 to 15
0bX0XX: Unprivileged access permitted on ports 16 to 23
0bX1XX: Only privileged access permitted on ports 16 to 23
0b0XXX: Unprivileged access permitted on ports 24 to 31
0b1XXX: Only privileged access permitted on ports 24 to 31.

43.8.5 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.	STALL ENA	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 IDs must be used for each trace source in the system.

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

Bits 9:8 **TSPRESCALE[1:0]:** local timestamp prescaler
Used with the trace packet reference clock. The possible values are:
0x0: No prescaling
0x1: Divides by 4.
0x2: Divides by 16.
0x3: Divides by 64.

Bits 7:6 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 the data trace packets.

Bit 4 **SWOENA:** Enable for asynchronous clocking of the timestamp counter (read only)
0: Timestamp counter uses processor clock

Bit 3 **TXENA:** Enables forwarding of hardware event packets from the DWT unit to the trace port.
0: Disabled
1: Enabled

Bit 2 **SYNCENA:** Enable for packet transmission synchronization
Note: The debugger setting this bit must also configure the DWT_CTRLR register SYNCTAP field in the DWT for the correct synchronization speed.
0: Disabled
1: Enabled

Bit 1 **TSENA:** Enable for local timestamp generation
0: Disabled
1: Enabled

Bit 0 **ITMENA:** ITM enable
0: Disabled
1: Enabled

43.8.6 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®.

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
0xA01: DWT architecture

43.8.7 ITM device type register 4 (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

43.8.8 ITM CoreSight peripheral identity register 4 (ITM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **F4KCOUNT[3:0]:** Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]:** JEP106 continuation code
0x4: Arm® JEDEC code

43.8.9 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

43.8.10 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

43.8.11 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 ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]:** JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

43.8.12 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

43.8.13 ITM CoreSight component identity register 0 (ITM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

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

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

Bits 7:0 **PREAMBLE[7:0]:** Component ID bits [7:0]
0x0D: Common ID value

43.8.14 ITM CoreSight peripheral identity register 1 (ITM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

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

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

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

Bits 7:4 **CLASS[3:0]:** Component ID bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]:** Component ID bits [11:8]
0x0: Common ID value

43.8.15 ITM CoreSight component identity register 2 (ITM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

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

Bits 7:0 **PREAMBLE[19:12]:** Component ID bits [23:16]
0x05: Common ID value

43.8.16 ITM CoreSight component identity register 3 (ITM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

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

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

Bits 7:0 **PREAMBLE[27:20]:** Component ID bits [31:24]
0xB1: Common ID value

43.8.17 ITM register map and reset values

Table 431. 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
0x080 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
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	DWT_DEVARCHR	ARCHTITECT[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	0	1
0xFC0 to 0xFC8	Reserved	Reserved.																															
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[3:8]			
	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.	F4KCOUNT [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved.																															
0xFE0	ITM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	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

Table 431. 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
0xFF4	ITM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	1	1	0	0	0	0	0
0xFF8	ITM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	ITM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 43.5: System debug AP0 features* for the register boundary addresses.

43.9 Breakpoint unit (BPU)

The BPU allows hardware breakpoints to be set. 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 BPU and how to use it, refer to the Armv8-M Architecture Reference Manual [4].

43.9.1 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 - least significant bits
0x8: 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 BPU_CTRLR register is ignored if this bit is not set to 1.

Bit 0 **ENABLE**: BPU enable
0: Disabled
1: Enabled

43.9.2 BPU comparator x register (BPU_COMPxR)

Address offset: 0x008 + 0x4 * 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

43.9.3 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®.

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

43.9.4 BPU device type register 4 (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

43.9.5 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.	F4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **F4KCOUNT[3:0]:** register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]:** JEP106 continuation code
0x4: Arm® JEDEC code

43.9.6 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

43.9.7 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

43.9.8 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
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]:** JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

43.9.9 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

43.9.10 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 ID bits [7:0]
0x0D: Common ID value

43.9.11 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 ID bits [15:12] - component class
0x9: debug component

Bits 3:0 **PREAMBLE[11:8]:** Component ID bits [11:8]
0x0: Common ID value

43.9.12 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 ID bits [23:16]
0x05: Common ID value

43.9.13 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 ID bits [31:24]
0xB1: Common ID value

43.9.14 BPU register map and reset values

Table 432. 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	NUM_LIT [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUM_CODE [6:4]			Res.	Res.	Res.	Res.	NUM_CODE [3:0]				Res.	Res.	KEY	ENABLE
	Reset value	0	0	0	1														0	0	0					1	0	0	0			0	0
0x004	Reserved	Reserved.																															
0x008 to 0x024	BPU_COMP0R to 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
0x02C-0xFB8	Reserved	Reserved.																															
0xFBC	DWT_DEVARCHR	ARCHTITECT[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	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[3:8]			
	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.	F4KCOUNT [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	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

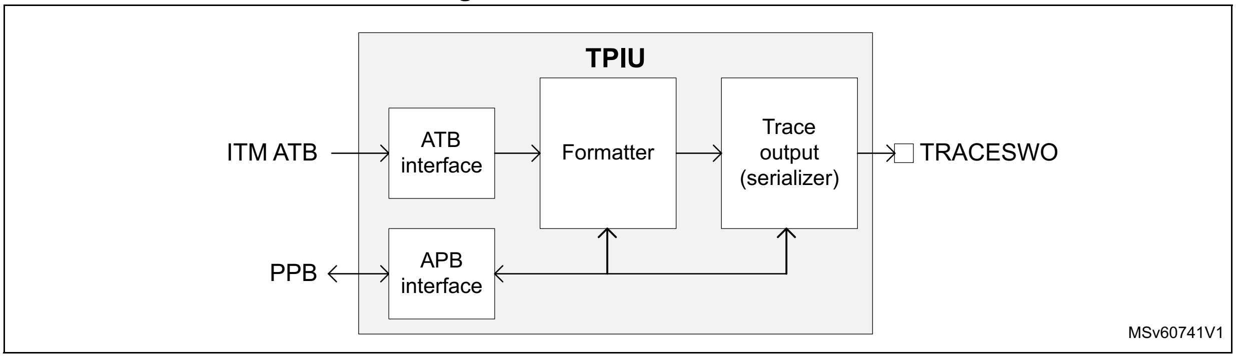
Refer to *Section 43.5: System debug AP0 features* for the register boundary addresses.

43.10 Trace port interface unit (TPIU)

The TPIU formats the trace stream and outputs it on the external trace port signals. The TPIU has one ATB slave ports for incoming trace data from the ITM. The trace port is the serial-wire output, TRACESWO.

Figure 494 shows the TPIU architecture.

Figure 494. TPIU architecture



For more information on the TPIU and how to use it, refer to the Arm® Cortex-M33 Technical Reference Manual *[5]*.

43.10.1 TPIU registers

43.10.2 TPIU supported port size register (TPIU_SSPSR)

Address offset: 0x000

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]:** Supported trace port sizes, from 1 to 32 pins
Bit n-1 when set indicates that port size n is supported.
0x0000 0001: Port size 1 supported

43.10.3 TPIU current port size register (TPIU_CSPSR)

Address offset: 0x004

Reset value: 0x0000 0001

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

Bits 31:0 **PORTSIZE[31:0]:** Current trace port size

Bit n-1 when set indicates that the current port size is n pins. The value of n must be within the range of supported port size (1). Only one bit can be set, or unpredictable behavior may result. This register must be modified only when the formatter is stopped.

43.10.4 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]:** selects the baud rate for the asynchronous output, TRACESWO

The baud rate is given by the TRACELKIN frequency divided by (PRESCALER +1).

43.10.5 TPIU selected pin protocol register (TPIU_SPPR)

Address offset: 0x0F0

Reset value: 0x0000 0001

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

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **TXMODE[1:0]:** selects the protocol used for trace output
0x0: reserved (parallel trace port mode not supported in this device)
0x1: Asynchronous SWO using Manchester encoding
0x2: Asynchronous SWO using NRZ encoding
0x3: reserved

43.10.6 TPIU formatter and flush status register (TPIU_FFSR)

Address offset: 0x300

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTNON STOP	TCPRE SENT	FTSTO PPED	FLINPR OG
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **FTNONSTOP:** Indicates whether formatter can be stopped or not
1: Formatter cannot be stopped.

Bit 2 **TCPRESENT:** Indicates whether the optional TRACECTL output pin is available for use
0: TRACECTL pin is not present in this device.

Bit 1 **FTSTOPPED:** Stop request signal received
The formatter received a stop request signal and all trace data and post-amble is sent. Any additional trace data on the ATB interface is ignored.
0: Formatter not stopped
1: Formatter stopped

Bit 0 **FLINPROG:** Flush in progress
This bit indicates whether a flush on the ATB slave port is in progress and reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in the TPIU_FFCR register.
0: No flush in progress
1: Flush in progress

43.10.7 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: generate a flush

Bits 5:2 Reserved, must be kept at reset value.

Bit 1 **ENFCONT:** Continuous formatting enable
Setting this bit to 0 in SWO mode bypasses the formatter and only ITM/DWT trace is output.
0: Continuous formatting disabled
1: Continuous formatting enabled

Bit 0 Reserved, must be kept at reset value.

43.10.8 TPIU formatter synchronization counter register (TPIU_FSCR)

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.

43.10.9 TPIU claim tag set register (TPIU_CLAIMSETR)

Address offset: 0xFA0

Reset value: 0x0000 000F

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

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]:** Sets claim tag bits

Write:
0000: No effect
xxx1: Sets bit 0.
xx1x: Sets bit 1.
x1xx: Sets bit 2.
1xxx: Sets bit 3.
Read:
0xF: Indicates there are four bits in claim tag.

43.10.10 TPIU claim tag clear register (TPIU_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

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

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]:** resets claim tag bits
Write:
0000: No effect
xxx1: Clears bit 0.
xx1x: Clears bit 1.
x1xx: Clears bit 2.
1xxx: Clears bit 3.
Read: Returns current value of claim tag.

43.10.11 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]			Res.	MAXNUM[4:0]				
				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 supported
1: supported

Bit 10 **SWOMAN:** serial-wire output, Manchester encoded format supported
1: supported

Bit 9 **TCLKDATA:** indicates whether trace clock plus data is supported
0: supported

Bits 8:6 **FIFOSIZE[2:0]:** FIFO size in powers of 2
0x2: FIFO size = 4 bytes

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **MAXNUM[4:0]:** number/type of ATB input port multiplexing
0x0: one input port

43.10.12 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

43.10.13 TPIU CoreSight peripheral identity register 4 (TPIU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **F4KCOUNT[3:0]:** Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]:** JEP106 continuation code
0x4: Arm® JEDEC code

43.10.14 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

43.10.15 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

43.10.16 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 ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]:** JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

43.10.17 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

43.10.18 TPIU CoreSight component identity register 0 (TPIU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

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

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

Bits 7:0 **PREAMBLE[7:0]:** Component ID bits [7:0]
0x0D: Common ID value

43.10.19 TPIU CoreSight peripheral identity register 1 (TPIU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

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

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

Bits 7:4 **CLASS[3:0]:** Component ID bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]:** Component ID bits [11:8]
0x0: Common ID value

43.10.20 TPIU CoreSight component identity register 2 (TPIU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

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

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

Bits 7:0 **PREAMBLE[19:12]:** Component ID bits [23:16]
0x05: Common ID value

43.10.21 TPIU CoreSight component identity register 3 (TPIU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

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

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

Bits 7:0 **PREAMBLE[27:20]:** Component ID bits [31:24]
0xB1: Common ID value

43.10.22 TPIU register map and reset values

Table 433. 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	0	0	0	1
0x004	TPIU_CSPSR	PORTSIZE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x008 to 0x00C	Reserved	Reserved.																															
0x010	TPIU_ACPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[12:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
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_FSCR	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
0x30C to 0xF9C	Reserved	Reserved.																															
0xFA0	TPIU_CLAIMSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET [3:0]			
	Reset value																													1	1	1	1
0xFA4	TPIU_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR [3:0]			
	Reset value																													0	0	0	0
0xFA8 to 0xFC4	Reserved	Reserved.																															
0xFC8	TPIU_DEVIDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWONRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			Res.	MAXNUM[4:0]				
	Reset value																					1	1	0	0	1	0		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

Table 433. TPIU register map and reset values (continued)

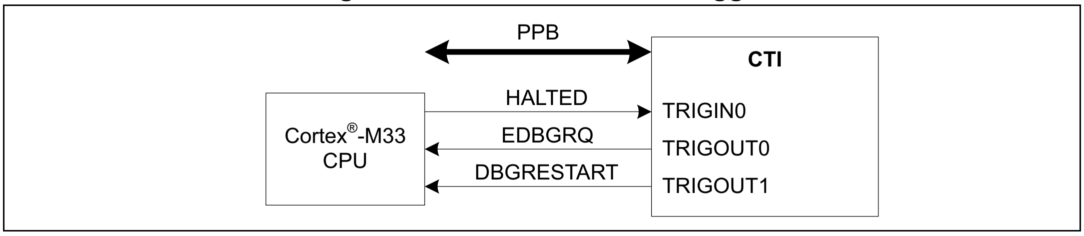
Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFD0	TPIU_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F4KCOUNT [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved.																															
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 *Section 43.5: System debug AP0 features* for the register boundary addresses.

43.11 Cross trigger interface (CTI)

The CTI allows cross triggering on the processor.

Figure 495. Embedded cross trigger



The CTI enables events from various sources to trigger debug activity (e.g. halt the processor core).

The trigger input and output signals for the CTI are listed, respectively, in tables 434 and 435.

Table 434. CTI inputs

No.	Source signal	Source component	Comments
0	HALTED	CPU	CPU halted - indicates CPU is in debug mode.
1	-	-	Not used
2	-	-	Not used
3	-	-	Not used
4	-	-	Not used
5	-	-	Not used
6	-	-	Not used
7	-	-	Not used

Table 435. CTI outputs

No.	Source signal	Source component	Comments
0	EDBGRQ	CPU	CPU halt request - Puts CPU in debug mode.
1	DBGRESTART	CPU	CPU restart request - CPU exits debug mode.
2	-	-	Not used
3	-	-	Not used
4	-	-	Not used
5	-	-	Not used
6	-	-	Not used
7	-	-	Not used

For more information on the CTI and how to use it, refer to the Arm® CoreSight SoC-400 Technical Reference Manual [2].

43.11.1 CTI registers

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 cross-triggering enable
0: disabled
1: enabled

CTI trigger acknowledge register (CTI_INTACKR)

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.	INTACK[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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 CTI TRIGOUT output. When a 1 is written to a bit in this register, the corresponding CTI TRIGOUT 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
XXX1: 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:
0000: No effect
XXX1: Sets event on Channel 0.
XX1X: Sets event on Channel 1.
X1XX: Sets event on Channel 2.
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]			
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPCLEAR[3:0]:** channel event clearing
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: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**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 in x enable register (CTI_INENRx)

Address offset: 0x020 + 0x4 * 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]:** cross trigger event enable/disable
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 any channel.
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 out x enable register (CTI_OUTENRx)

Address offset: 0x0A0 + 0x4 * 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]**: For each channel, defines whether an event on that channel generates a trigger on CTI TRIGOUTx (x = 0 to 7).
0000: Channel events do not generate triggers on any trigger output.
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 in 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 CTITRIGIN 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 out 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 CTI TRIGOUT 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 in 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 out 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
0x4: 4 channels

Bits 15:8 **NUMTRIG[7:0]:** Number of ECT triggers
0x8: 8 trigger inputs and 8 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: Indicates that this component is a cross-triggering one.

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.	F4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **F4KCOUNT[3:0]:** register file size
0x0: 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 identifier 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 ID bits [7:0]
0x0D: Common ID 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 ID bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]:** Component ID bits [11:8]
0x0: Common ID 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 ID bits [23:16]
0x05: Common ID 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 ID bits [31:24]
0xB1: Common ID value

CTI register map and reset values

Table 436. 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																									0	0	0	0	0	0	0	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																													0	0	0	0
0x020 to 0x03C	CTI_INENR0 to CTI_INENR7	Res.	Res.	Res.	Res.	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_OUTENR0 to CTI_OUTENR7	Res.	Res.	Res.	Res.	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_TRIGISTSR	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_TRIGOSTSR	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.	CHIN STATUS[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.	CHOUT STATUS[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.	SUBTYPE [3:0]				MAJORTYPE [3:0]			
	Reset value																									0	0	0	1	0	1	0	0

Table 436. 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
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.	F4KCOUNT [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved	Reserved																															
0xFE0	CTI_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	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 *Section 43.5: System debug AP0 features* for the register boundary addresses.

43.12 Microcontroller debug unit (DBGMCU)

DBGMCU is a component containing a number of registers that control the power and clock behavior in debug mode. It allows the debugger (or the debug software) to perform the following tasks:

- Maintain the clock and power to the processor core and system debug and trace components when in low-power modes.(Stop or Standby)
- Stop the clock to certain peripherals (such as watchdogs, timers, RTC) when the processor core is halt in debug mode.
- Obtain information on the processor core and system operating modes.

43.12.1 DBGMCU access

The DBGMCU can be accessed through both AP0 and AP1, and directly by the Cortex-M33 processor with the following restrictions:

- When there is no debugger connected to AP0:
  - DBGMCU can be accessed through AP1 and by the Cortex-M33 processor.
- When the debugger is connected to AP0:
  - DBGMCU can only be accessed through AP0.
  - DBGMCU accesses by the Cortex-M33 processor generate a bus fault.

43.12.2 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 43.6.1: CPU ROM tables*.

43.12.3 Part number codification

The part number codification can be read from the DBGMCU registers DBGMCU_PNCR.

43.12.4 Low-power mode emulation

When the device enters Stop mode or Standby mode, the debugger can no longer access the CPU debug access port and loses the connection with the device CPU. In Stop mode it remains still possible to access the DBGMCU from the debugger. To gain or keep CPU debug access, the debugger (or software) can set the DBG_STOP and/or DBG_STANDBY bits in the DBGMCU_CR register. This allows to maintain the clock and power to the device CPU and DBGMCU even when it enters Stop mode or Standby mode. When in debug Stop or debug Standby mode, the processor remains in SleepDeep mode and exit this mode in the normal way. In these debug modes the processor can also be woken up by a debugger halt instruction, which also puts the device back to Run mode. When in debug Stop or debug Standby mode all other device peripherals behave as in Stop mode or Standby mode and are not accessible from the debugger. In debug Stop mode autonomous peripheral continue to operate when they request their clock. In debug Standby mode the peripherals are kept under reset.

Note: When using debug Stop mode (DBG_STOP) a SysTick event wakes up the device. It is good practice that software disables the SysTick, before the CPU enters SleepDeep.

In Stop mode, without enabling DBG_STOP, debug access via the DAP is still possible to the DBGMCU. To keep access to the DBGMCU in Standby mode, without enabling

DBG_STANDBY, is possible by enabling CDBGPWRIUPREQ before entering Standby mode. Keeping access to the DBGMCU allows to access the DBGMCU registers.

In debug Stop mode and debug Standby mode the device behavior does not differ from Stop mode and Standby mode, with the exception of SRAMs, which are retained and keep their values. For an overview see *Table 437*.

Table 437. Low power debug overview

Low power mode	System clock	Power supply	DBGMCU and DAP	CPU debug	Autonomous peripherals	Oother peripherals	SRAMs	Low-power mode exit	Debug wakeup
DBG_STOP[1]	HSI16	On	Accessible		As Stop mode, clock on request	As Stop mode, clock off	As Stop mode retained	Stop wakeup	Halt CPU
DBG_STANDBY[1]			Accessible		Under reset		As Standby mode retained	Standby reset	Halt CPU
Stop[2]	Off		Accessible	As Stop mode, clock off	As Stop mode, clock on request	As Stop mode, clock off	As Stop mode retained	Stop wakeup	Via DBG_STOP
Standby with CDBGPWRUPREQ[2]			Accessible	Under reset			As Standby mode retained	Standby reset	Via DBG_ STANDBY
Standby[3]		Off	Power down				Power down when not retained	Standby reset	None

1. In DBG_STOP and DBG_STANDBY mode the debug connection with the CPU and access to the DBGMCU is kept. CPU can be woken up with debugger Halt.
2. In Stop mode and Standby mode with CDBGPWRUPREQ, only debug access to the DBGMCU is kept, the debug connection to the CPU is lost. DBGMCU can be used to activate CPU clock via DBG_STOP or DBG_STANDBY.
3. In Standby mode all debug access is lost.

In Stop debug mode and Standby debug mode both the CPU debug and DBGMCU are accessible from the debugger. A debugger halt to the CPU wakes up the CPU and sets the system to Run mode. Only in Run mode peripherals with their bus clock enabled are accessible from the debugger.

In Stop mode and Standby mode with CDBGPWRUPREQ enabled, only the DBGMCU is accessible from the debugger. Via the DBGMCU DBG_STOP and DBG_STANDBY the debugger can move the device to the debug Stop and debug Standby mode to regain debug access to the CPU.

In Standby mode all debug access is lost.

43.12.5 Low-power mode status

The DBGMCU provide status flags for the processor core and device operating mode in DBGMCU_SCR.

Table 438. Low-power mode status flags

Status flag	Description
CS	CPU in Sleep mode
CDS	CPU in DeepSleep mode
STOPF	Device in Stop mode
SBF	Device in Standby mode (this bit is only available with DBG_STANDBY or CDBGPWRUPREQ)

43.12.6 Peripheral clock freeze

The DBGMCU provides a peripheral clock freeze feature in the DBGMCU_xxxFZR registers. This allows the operation (clock) of certain peripherals to be suspended in debug mode (when the processor is halted). Peripherals supporting this feature are listed in *Table 439*.

Table 439. Peripheral clock freeze control bits

<table>
<tr><th>Bus</th><th>Control</th><th>Peripheral</th><th>Behavior</th></tr>
<tr><td rowspan="5">APB1L</td><td rowspan="5">DBGMCU_APB1LFZR</td><td>I2C1</td><td>Freeze timeout counting</td></tr>
<tr><td>IWDG</td><td rowspan="4">Freeze counting</td></tr>
<tr><td>WWDG</td></tr>
<tr><td>TIM3</td></tr>
<tr><td>TIM2</td></tr>
<tr><td>APB1H</td><td>DBGMCU_APB1HFZR</td><td>LPTIM2</td><td>Freeze counting</td></tr>
<tr><td rowspan="3">APB2</td><td rowspan="3">DBGMCU_APB2FZR</td><td>TIM17</td><td rowspan="3">Freeze counting</td></tr>
<tr><td>TIM16</td></tr>
<tr><td>TIM1</td></tr>
<tr><td rowspan="3">APB7</td><td rowspan="3">DBGMCU_APB7FZR</td><td>RTC</td><td rowspan="2">Freeze counting</td></tr>
<tr><td>LPTIM1</td></tr>
<tr><td>I2C3</td><td>Freeze timeout counting</td></tr>
<tr><td>AHB1</td><td>DBGMCU_AHB1FZR</td><td>GPDMA1</td><td>Freeze channel transfers</td></tr>
</table>

The control bit, when set, causes the corresponding peripheral operation to be suspended when the CPU is stopped in debug (HALTED = 1).

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 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 and 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 440. Access to peripheral clock freeze control

spiden	Peripheral status	Access type	DBG_xxx_STOP access
x	Non-secure	Non-secure	RW
		Secure	
0	Secure	Non-secure	R only
		Secure(1)	
1		Non-secure	R only
		Secure	RW

1. When spiden = 0, secure accesses requested by the debugger are converted to non-secure.

The status (secure or non-secure) of a peripheral or a DMA channel is signaled to the DBGMCU by GTZC or the peripheral.

The CPU access to the DBG_xxx_STOP bits does not depend upon spiden. This access depends only upon the security status of the peripheral or DMA channel. The bits corresponding to a secure peripheral or DMA channel can only be modified by a secure access (when the CPU is in secure state).

43.12.7 DBGMCU registers

The DBGMCU registers are not reset by a system reset, only by a power-on reset.

DBGMCU identity code register (DBGMCU_IDCODE)

Address offset: 0x000

Reset value: 0xXXXX 6492

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 ID
0x1000: Revision A
0x2000: Revision B

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV_ID[11:0]:** Device ID
0x492: STM32WBA5xxx

DBGMCU status and configuration register (DBGMCU_SCR)

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.	CDS	CS	Res.	Res.	Res.	SBF	STOPF	LPMS[2:0]		
						r	r				r	r	r	r	r

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

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **CDS:** CPU DeepSleep
0: CPU not in DeepSleep
1: CPU in DeepSleep

Bit 24 **CS:** CPU Sleep
0: CPU not in Sleep
1: CPU in Sleep

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

Bit 20 **SBF:** Device Standby flag
0: Device not in Standby mode
1: Device in Standby mode

Bit 19 **STOPF:** Device Stop flag
0: device not in Stop mode
1: device in Stop mode

Bits 18:16 **LPMS[2:0]:** Device low power mode selected
000: Stop 0 mode
001: Stop 1 mode
10x: Standby mode
others reserved

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **DBG_STANDBY:** Allows debug in Standby mode
Write access can be protected by PWR_SECCFGR.LPMSEC.
0: Normal Standby mode operation, all clocks are disabled automatically in Stop mode and core is powered down.
1: Core power maintained and CPU debug clock enabled in Standby mode, all other are under reset in Stop mode, except for DBGMCU.
The CPU debug and DBGMCU clocks remain active and the HSI16 oscillator is used as system clock, the supply and SRAM memory content is maintained during Standby debug mode, allowing CPU debug capability. On exit from Standby mode, a standby reset is performed.

Bit 1 **DBG_STOP:** Allows debug in Stop mode
Write access can be protected by PWR_SECCFGR.LPMSEC.
0: Normal Stop mode operation, all clocks are disabled automatically in Stop mode.
1: CPU debug clock enabled in Stop mode, all other peripheral clocks are disabled automatically in Stop mode, except for DBGMCU.
The CPU debug and DBGMCU clocks remain active and the HSI16 oscillators is used as system clock during Stop debug mode, allowing CPU 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: 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.	DBG_I2C1_STOP	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.	DBG_IWDG_STOP	DBG_WWDG_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM3_STOP	DBG_TIM2_STOP
			rw	rw										rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **DBG_I2C1_STOP:** I2C1 SMBUS timeout stop in CPU debug
Write access can be protected by GTZC_TZSC.I2C1SEC.
0: Normal operation. I2C1 SMBUS timeout continues to operate while CPU is in debug mode.
1: Stop in debug. I2C1 SMBUS timeout is frozen while CPU is in debug mode.

Bits 20:13 Reserved, must be kept at reset value.

Bit 12 **DBG_IWDG_STOP:** IWDG stop in CPU debug
Write access can be protected by GTZC_TZSC.IWDGSEC.
0: Normal operation. IWDG continues to operate while CPU is in debug mode.
1: Stop in debug. IWDG is frozen while CPU is in debug mode.

Bit 11 **DBG_WWDG_STOP:** WWDG stop in CPU debug
Write access can be protected by GTZC_TZSC.WWDGSEC
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 10:2 Reserved, must be kept at reset value.

Bit 1 **DBG_TIM3_STOP:** TIM3 stop in CPU debug
Write access can be protected by GTZC_TZSC.TIM3SEC.
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 CPU debug
Write access can be protected by GTZC_TZSC.TIM2SEC.
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: 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.	DBG_ LPTIM2 _STOP	Res.	Res.	Res.	Res.	Res.
										rw					

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **DBG_LPTIM2_STOP**: LPTIM2 stop in CPU debug
Write access can be protected by GTZC_TZSC.LPTIM2SEC.
0: Normal operation. LPTIM2 continues to operate while CPU is in debug mode.
1: Stop in debug. LPTIM2 is frozen while CPU is in debug mode.

Bits 4:0 Reserved, must be kept at reset value.

DBGMCU APB2 peripheral freeze register (DBGMCU_APB2FZR)

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.	DBG_ TIM17 _STOP	DBG_ TIM16 _STOP	Res.
													rw	rw	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	DBG_ TIM1 _STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw											

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

Bit 18 **DBG_TIM17_STOP**: TIM17 stop in CPU debug
Write access can be protected by GTZC_TZSC.TIM17SEC.
0: Normal operation. TIM17 continues to operate while CPU is in debug mode.
1: Stop in debug. TIM17 is frozen while CPU is in debug mode.

Bit 17 **DBG_TIM16_STOP**: TIM16 stop in CPU debug
Write access can be protected by GTZC_TZSC.TIM16SEC.
0: Normal operation. TIM16 continues to operate while CPU is in debug mode.
1: Stop in debug. TIM16 is frozen while CPU is in debug mode.

Bits 16:12 Reserved, must be kept at reset value.

Bit 11 **DBG_TIM1_STOP**: TIM1 stop in CPU debug
Write access can be protected by GTZC_TZSC.TIM1SEC.
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 APB7 peripheral freeze register (DBGMCU_APB7FZR)

Address offset: 0x024

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.	Res.	Res.	DBG_LPTIM1_STOP	Res.
	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 CPU debug
Access can be protected by GTZC_TZSC.TIM17SEC.
Can only be accessed secure when one or more features in the RTC or TAMP is/are secure.
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 29:18 Reserved, must be kept at reset value.

Bit 17 **DBG_LPTIM1_STOP**: LPTIM1 stop in CPU debug
Access can be protected by GTZC_TZSC.LPTIM1SEC.
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 CPU debug
Access can be protected by GTZC_TZSC.I2C3SEC.
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: 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.	DBG_GPDMA1_CH7_STOP	DBG_GPDMA1_CH6_STOP	DBG_GPDMA1_CH5_STOP	DBG_GPDMA1_CH4_STOP	DBG_GPDMA1_CH3_STOP	DBG_GPDMA1_CH2_STOP	DBG_GPDMA1_CH1_STOP	DBG_GPDMA1_CH0_STOP
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 7 **DBG_GPDMA1_CH7_STOP**: GPDMA 1 channel 7 stop in CPU debug
Write access can be protected by GPDMA_SECCFGR.SEC7.
0: Normal operation. GPDMA 1 channel 7 continues to operate while CPU is in debug mode.
1: Stop in debug. GPDMA 1 channel 7 is frozen while CPU is in debug mode.

Bit 6 **DBG_GPDMA1_CH6_STOP**: GPDMA 1 channel 6 stop in CPU debug
Write access can be protected by GPDMA_SECCFGR.SEC6.
0: Normal operation. GPDMA 1 channel 6 continues to operate while CPU is in debug mode.
1: Stop in debug. GPDMA 1 channel 6 is frozen while CPU is in debug mode.

Bit 5 **DBG_GPDMA1_CH5_STOP**: GPDMA 1 channel 5 stop in CPU debug
Write access can be protected by GPDMA_SECCFGR.SEC5.
0: Normal operation. GPDMA 1 channel 5 continues to operate while CPU is in debug mode.
1: Stop in debug. GPDMA 1 channel 5 is frozen while CPU is in debug mode.

Bit 4 **DBG_GPDMA1_CH4_STOP**: GPDMA 1 channel 4 stop in CPU debug
Write access can be protected by GPDMA_SECCFGR.SEC4.
0: Normal operation. GPDMA 1 channel 4 continues to operate while CPU is in debug mode.
1: Stop in debug. GPDMA 1 channel 4 is frozen while CPU is in debug mode.

Bit 3 **DBG_GPDMA1_CH3_STOP**: GPDMA 1 channel 3 stop in CPU debug
Write access can be protected by GPDMA_SECCFGR.SEC3.
0: Normal operation. GPDMA 1 channel 3 continues to operate while CPU is in debug mode.
1: Stop in debug. GPDMA 1 channel 3 is frozen while CPU is in debug mode.

Bit 2 **DBG_GPDMA1_CH2_STOP**: GPDMA 1 channel 2 stop in CPU debug
Write access can be protected by GPDMA_SECCFGR.SEC2.
0: Normal operation. GPDMA 1 channel 2 continues to operate while CPU is in debug mode.
1: Stop in debug. GPDMA 1 channel 2 is frozen while CPU is in debug mode.

Bit 1 **DBG_GPDMA1_CH1_STOP**: GPDMA 1 channel 1 stop in CPU debug
Write access can be protected by GPDMA_SECCFGR.SEC1.
0: Normal operation. GPDMA 1 channel 1 continues to operate while CPU is in debug mode.
1: Stop in debug. GPDMA 1 channel 1 is frozen while CPU is in debug mode.

Bit 0 **DBG_GPDMA1_CH0_STOP**: GPDMA 1 channel 0 stop in CPU debug
Write access can be protected by GPDMA_SECCFGR.SEC0.
0: Normal operation. GPDMA 1 channel 0 continues to operate while CPU is in debug mode.
1: Stop in debug. GPDMA 1 channel 0 is frozen while CPU is in debug mode.

DBGMCU status register (DBGMCU_SR)

Address offset: 0x0FC

Reset value: 0x000X 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AP_ENABLED[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**
AP_PRESENT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **AP_ENABLED[15:0]**: Bit n identifies whether access port APn is open (can be accessed via the debug port) or locked (debug access to the APn is blocked, except for DBGMCU access)
Bit n = 0: APn locked (except for access to DBGMCU)
Bit n = 1: APn enabled

Bits 15:0 **AP_PRESENT[15:0]**: Bit n identifies whether access port APn 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 (OEMn 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 part number codification register (DBGMCU_PNCR)

Address offset: 0x7DC

Reset value: 0xXXXX XXXX

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

Bits 31:0 **CODIFICATION[31:0]:** Part number codification
0x0032 3541: STMicroelectronics STM32WBA52xx part number codification.
0x0034 3541: STMicroelectronics STM32WBA54xx part number codification.
0x0035 3541: STMicroelectronics STM32WBA55xx part number codification.

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.	F4KCOUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

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

Bits 7:4 **F4KCOUNT[3:0]:** Register file size
0x0: 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 identifier 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 ID bits [7:0]
0x0D: Common ID value

DBGMCU CoreSight peripheral 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 ID bits [15:12] - component class
0xF: no CoreSight component

Bits 3:0 **PREAMBLE[11:8]:** Component ID bits [11:8]
0x0: common ID 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 ID bits [23:16]
0x05: Common ID 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 ID bits [31:24]
0xB1: Common ID value

43.12.8 DBGMCU register map and reset values

Table 441. 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					0	1	0	0	1	0	0	1	0	0	1	0
0x004	DBGMCU_SCR	Res.	Res.	Res.	Res.	Res.	Res.	CDS	CS	Res.	Res.	Res.	SBF	STOPF	LPMS[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_STANDBY	DBG_STOP	Res.
	Reset value							0	0				0	0	0	0	0														0	0	
0x008	DBGMCU_ APB1LFZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_IWDG_STOP	DBG_WWDG_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM3_STOP	DBG_TIM2_STOP
	Reset value											0									0	0										0	0
0x00C	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.	Res.	Res.
	Reset value																											0					
0x010	DBGMCU_ APB2FZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP	Res.	Res.	Res.	Res.	Res.	DBG_TIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0						0											
0x014 to 0x020	Reserved	Reserved																															
0x024	DBGMCU_ APB7FZR	Res.	DBG_RTC_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I2C3_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		0													0							0										
0x028	DBGMCU_ AHB1FZR	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_GPDMA1_CH7_STOP	DBG_GPDMA1_CH6_STOP	DBG_GPDMA1_CH5_STOP	DBG_GPDMA1_CH4_STOP	DBG_GPDMA1_CH3_STOP	DBG_GPDMA1_CH2_STOP	DBG_GPDMA1_CH1_STOP	DBG_GPDMA1_CH0_STOP
	Reset value																									0	0	0	0	0	0	0	0

Table 441. 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
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	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
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_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
0x108 to 0x7D8	Reserved	Reserved																															
0x7DC	DBGMCU_PNCR	CODIFICATION[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
0x7E0 to 0xFCC	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.	F4KCOUNT [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 43.5.1: System debug ROM table* and *Section 43.6.1: CPU ROM tables* for the register boundary addresses.

43.13 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 2015
3. DDI 0314H - Arm® CoreSight™ Components Technical Reference Manual, Issue H, 10 July, 2009
4. DDI 0553A (ID092917) - Arm®v8-M Architecture Reference Manual, Issue A.f, 29 September 2017
5. 100230_0002_00_en - Arm® Cortex®-M33 Processor r0p2 Technical Reference Manual, Issue 0002-00, 10 May 2017

44 Device electronic signature (DESIG)

The device electronic signature is stored in the System memory area of the flash memory module, and can be read using the debug interface or by the CPU. It contains factory-programmed identification and calibration data that allow the user firmware or other external devices to automatically match the characteristics of the microcontroller.

44.1 Device electronic signature registers

44.1.1 DESIG package data register (DESIG_PKGR)

Address offset: 0x000

Reset value: 0xXXXX XXXX (X is factory-programmed)

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

15	14	13	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 31:5 Reserved, must be kept at reset value.

Bits 4:0 **PKG[4:0]**: Package type
00000: UFQFPN32
00010: UFQFPN48
01001: WLCSP41 SMPS
01010: UFQFPN48 SMPS
01011: UFBGA59 SMPS
Others: reserved

44.1.2 DESIG 96-bit unique device ID register 1 (DESIG_UIDR1)

Address offset: 0x200

Reset value: 0xXXXX XXXX (X is factory-programmed)

The 96-bit unique device identifier is ideally suited:

- for use as serial number (USB string serial number or other end applications)
- for use as part of the security keys to increase the code security in the flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the memory
- during processes such as secure boot

This unique device identifier provides a reference number, unique for a given device and in any context. These bits cannot be altered by the user.

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

Bits 31:0 **UID[31:0]:** X and Y coordinates on the wafer expressed in BCD format

44.1.3 DESIG 96-bit unique device ID register 2 (DESIG_UIDR2)

Address offset: 0x204

Reset value: 0xXXXX XXXX (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 31:0 **UID[63:32]:** Lot and wafer number
UID[63:40] = LOT_NUM[23:0], lot number lower part (ASCII encoded)
UID[39:32] = WAF_NUM[7:0], wafer number (8-bit unsigned number)

44.1.4 DESIG 96-bit unique device ID register 3 (DESIG_UIDR3)

Address offset: 0x208

Reset value: 0xXXXX XXXX (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 higher part (ASCII encoded)

44.1.5 DESIG temperature calibration 1 register (DESIG_TSCAL1R)

Address offset: 0x210

Reset value: 0xXXXX XXXX (X is factory-programmed)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TS_CAL1[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **TS_CAL1[11:0]:** Factory temperature sensor calibration 1 value for ADC4

44.1.6 DESIG temperature calibration 2 register (DESIG_TSCAL2R)

Address offset: 0x240

Reset value: 0xXXXX XXXX (X is factory-programmed)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TS_CAL2[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.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

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

Bits 27:16 **TS_CAL2[11:0]:** Factory temperature sensor calibration 2 value for ADC4

Bits 15:0 Reserved, must be kept at reset value.

44.1.7 DESIG FLASH size data register (DESIG_FLASHSIZER)

Address offset: 0x2A0

Reset value: 0xXXXX XXXX (X is factory-programmed)

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

Bits 31:16 **RAM_SIZE[15:0]:** RAM memory size

These bits indicates the size of the device total system RAM memory in Kbytes.
As an example, 0x0040 corresponds to 64 Kbytes.

Bits 15:0 **FLASH_SIZE[15:0]:** Flash memory size

These bits indicates the size of the device flash memory in Kbytes.
As an example, 0x0040 corresponds to 64 Kbytes.

44.1.8 DESIG internal voltage reference calibration register (DESIG_VREFINTCALR)

Address offset: 0x2A4

Reset value: 0xXXXX XXXX (X is factory-programmed)

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

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

Bits 19:8 **VREFINT_CAL[11:0]:** Factory internal voltage reference calibration value for ADC4

Bits 7:0 Reserved, must be kept at reset value.

44.1.9 DESIG resistor calibration register (DESIG_RCALR)

Address offset: 0x2F0

Reset value: 0xXXXX XXXX (X is factory-programmed)

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

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

Bits 7:0 **R_CAL[7:0]:** Resistor calibration value

44.1.10 DESIG radio gain calibration register (DESIG_RFGAINCALR)

Address offset: 0x2F4

Reset value: 0xXXXX XXXX (X is factory-programmed)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GAIN4_CAL[7:0]								GAIN3_CAL[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
GAIN2_CAL[7:0]								GAIN1_CAL[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **GAIN4_CAL[7:0]:** Radio gain 4 calibration value

Bits 23:16 **GAIN3_CAL[7:0]:** Radio gain 3 calibration value

Bits 15:8 **GAIN2_CAL[7:0]:** Radio gain 2 calibration value

Bits 7:0 **GAIN1_CAL[7:0]:** Radio gain 1 calibration value

44.1.11 DESIG IEEE 64-bit unique device ID register 1 (DESIG_UID64R1)

Address offset: 0x500

Reset value: 0xXXXX XXXX (X is factory-programmed)

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

Bits 31:0 **DEVNUM[31:0]:** device number
The 32-bit unique device number is a sequential number, different for each individual device.

44.1.12 DESIG IEEE 64-bit unique device ID register 2 (DESIG_UID64R2)

Address offset: 0x504

Reset value: 0x0080 E12A

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

Bits 31:8 **STID[23:0]:** company ID
0x0080E1 for STMicroelectronics

Bits 7:0 **DEVID[7:0]:** device ID
0x2A: STM32WBA5xxx

44.1.13 DESIG register map

Table 442. DESIG 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	DESIG_ PKGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKG[4:0]				
	Reset value																												x	x	x	x	x
0x004-0x1FF	Reserved																																
0x200	DESIG_ UIDR1	UID[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
0x204	DESIG_ UIDR2	UID[63:32]																															
	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
0x208	DESIG_ UIDR3	UID[95:64]																															
	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
0x210	DESIG_ TSCAL1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS_CAL1[11:0]											
	Reset value																					x	x	x	x	x	x	x	x	x	x	x	x
0x214-0x23F	Reserved																																
0x240	DESIG_ TSCAL2R	Res.	Res.	Res.	Res.	TS_CAL2[11:0]												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	x	x	x	x																
0x248-0x2EC	Reserved																																
0x2F0	DESIG_ RCALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	R_CAL[7:0]							
	Reset value																									x	x	x	x	x	x	x	x
0x2F4	DESIG_ RFGAINCALR	GAIN4_CAL[7:0]								GAIN3_CAL[7:0]								GAIN2_CAL[7:0]								GAIN1_CAL[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
0x2F8-0x29F	Reserved																																
0x2A0	DESIG_ FLASHSIZER	RAM_SIZE[15:0]																FLASH_SIZE[15:0]															
	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
0x2A4	DESIG_ VREFINT CALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VREFINT_CAL[11:0]												Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													x	x	x	x	x	x	x	x	x	x	x	x								
0x2A8-0x4FF	Reserved																																
0x500	DESIG_ UID64R1	DEVNUM[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
0x504	DESIG_ UID64R2	STID[23:0]																								DEVID[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.

45 Important security notice

The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST's customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

- ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture (www.psacertified.org) and/or Security Evaluation standard for IoT Platforms (www.trustcb.com). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on *www.st.com* for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers should re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers should not assume it is certified.
- Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.
- Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.
- While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.
- All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.

46 Revision history

Table 443. Document revision history

Date	Revision	Changes
20-Jan-2022	1	Initial release.
23-Nov-2022	2	Updated document title, *Flash programming errors*, *Section 7.9.15: FLASH boot address 0 register (FLASH_NSBOOTADD0R)*, *Section 7.9.16: FLASH boot address 1 register (FLASH_NSBOOTADD1R)*, *Section 7.9.17: FLASH secure boot address 0 register (FLASH_SECBOOTADD0R)*, *I/O states in Standby mode*, *Section 9.1: Introduction*, *Section 9.2: Main features*, *Section 9.4.3: Transmit output power*, *Section 11.7.2: PWR background autonomous mode (BAM)*, *Section 11.7.6: PWR Stop 0 mode*, *Section 11.7.7: PWR Stop 1 mode*, *Section 11.7.9: Power modes output pins*, *Section 11.10.9: PWR disable Backup domain register (PWR_DBPR)*, *Section 11.10.10: PWR security configuration register (PWR_SECCFGR)*, *LSI1 low-power*, *Section 12.4.17: Clock-out capability*, *Section 12.8.13: RCC AHB2 peripheral reset register (RCC_AHB2RSTR)*, *Section 12.8.39: RCC Backup domain control register (RCC_BDCR1)*, *Section 12.8.51: RCC clock configuration register 2 (RCC_CFGR4)*, *Section 14.4.5: GPIO port data registers*, *Section 14.8.1: GPIO port H mode register (GPIOH_MODER)*, *Section 15.3.3: SYSCFG FPU interrupt mask register (SYSCFG_FPUIMR)*, *Section 15.3.5: SYSCFG CPU secure lock register (SYSCFG_CSLCKR)*, *Section 16.3.1: Master to slave interconnection for timers*, *Section 16.3.5: Triggers to low-power timer*, and *Section 44.1.1: DESIG package data register (DESIG_PKGR)*. Removed former *Section 9.4.4: Bluetooth AoA and AoD*. Added *Section 45: Important security notice*. Updated *Table 41: Number of wait states according to CPU clock (hclk1) frequency (LPM = 1)*,*Table 75: Input / output pins*, *Table 78: 2.4 GHz RADIO supply configuration*, *Table 83: PWR internal input/output signals*, *Table 92: Power modes output states versus MCU power modes*, *Table 94: PWR interrupt requests*, *Table 104: RCC register map and reset values*, *Table 119: SYSCFG register map and reset values*, and *Table 132: Vector table*. Updated *Figure 32: Operating modes*. Minor text edits across the whole document.

Table 443. Document revision history (continued)

Date	Revision	Changes
20-Oct-2023	3	Document scope extended to STM32WBA54xx and STM32WBA55xx devices. Removed former *Section 1.4: Availability of peripherals*. Added *Section 10: PTA converter (PTACONV)*, *Section 12.4.19: Audio synchronization*, *Section 12.8.41: RCC Backup domain control register (RCC_BDCR2)*, sections *12.8.44* to *12.8.50*, *Section 16.3.14: From timer (TIM1/TIM2/TIM3) to comparators (COMP1/COMP2)*, *Section 16.3.15: From comparators (COMP1/COMP2) to timers*, *Section 22: Comparator (COMP)*, *Section 39.8.5: USART control register 3 [alternate] (USART_CR3)*, *Section 40.7.5: LPUART control register 3 [alternate] (LPUART_CR3)*, *Section 44.1.9: DESIG resistor calibration register (DESIG_RCALR)*, and *Section 44.1.10: DESIG radio gain calibration register (DESIG_RFGAINCALR)*. Updated *Section 11: Power control (PWR)* and its subsections to add SMPS (switch-mode power supply) support, *Section 12.4: RCC clocks functional description*, *Section 12.4.5: LSI clock*, *Section 12.4.6: System clock (SYSCLK) selection*, *Section 12.8.18: RCC APB2 peripheral reset register (RCC_APB2RSTR)*, *Section 12.8.19: RCC APB7 peripheral reset register (RCC_APB7RSTR)*, *Section 12.8.26: RCC APB2 peripheral clock enable register (RCC_APB2ENR)*, *Section 12.8.27: RCC APB7 peripheral clock enable register (RCC_APB7ENR)*, *Section 12.8.34: RCC APB2 peripheral clocks enable in Sleep and Stop modes register (RCC_APB2SMENR)*, *Section 12.8.35: RCC APB7 peripheral clock enable in Sleep and Stop modes register (RCC_APB7SMENR)*, *Section 12.8.37: RCC peripherals independent clock configuration register 2 (RCC_CCIPR2)*, *Section 12.8.39: RCC Backup domain control register (RCC_BDCR1)*, *Section 14.5.1: GPIO port A mode register (GPIOA_MODER)*, *Section 23.3.4: Charge transfer acquisition sequence*, *Section 23.3.4: Charge transfer acquisition sequence*, *Section 24.7.4: RNG noise source control register (RNG_NSCR)*, *Section 37.2: TAMP main features*, *Section 37.3.9: Tamper detection*, *DBGMCU part number codification register (DBGMCU_PNCR)*, and *Section 44.1.1: DESIG package data register (DESIG_PKGR)*. Updated *Table 99: RCC input/output signals connected to package pins or balls*, *Table 105: RCC security configuration summary*, *Table 106: Interrupt sources and control*, *Table 107: RCC register map and reset values*, *Table 112: GPIO implementation*, *Table 123: Peripherals interconnect matrix*, *Table 135: Vector table*, *Table 138: EXTI line connections*, *Table 145: ADC features*, *Table 180: RNG register map and reset map*, *Table 255: Interconnect to the tim_ti1 input multiplexer*, tables *260* to *264*, tables *279* to *282*, tables *284* to *285*, *Table 298: Timer break interconnect*, *Table 300: Interconnect to the ocref_clr input multiplexer*, tables *311* to *314*, *Table 377: USART interconnection (USART1/2)*, *Table 384: USART register map and reset values*, *Table 389: LPUART interconnections (LPUART1)*, *Table 395: LPUART register map and reset values*, tables in *Description of SPI interconnections*, and *Table 442: DESIG register map and reset values*. Updated *Figure 34: Clock tree*, *Figure 73: Calibration factor forcing*, *Figure 359: LPTIM block diagram(1)*, *Figure 388: I2C initialization flow*, and *Figure 394: Target initialization flow*. Minor text edits across the whole document.

Table 443. Document revision history (continued)

Date	Revision	Changes
05-Jun-2024	4	Updated *Section 8.4: ICACHE functional description*, *Section 9.2: Main features*, *Section 11.10.2: PWR control register 2 (PWR_CR2)*, *Section 15.2.1: I/O compensation cell management*, *Suspend/resume operations in ECB/CBC modes*, *Suspend and resume operations in CCM mode*, *Suspend/resume operations in ECB/CBC modes*, *Suspend and resume operations in CCM mode*, *Section 30.5.10: TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2, 3)*, *Section 36.3.6: Clock and prescalers*, *Section 37.3.2: TAMP pins and internal signals*, *Section 41.4.5: Standard multislave communication*, and *DBGMCU identity code register (DBGMCU_IDCODE)*. Updated figures *72* to *74*, *77*, and *79* in *Section 21: Analog-to-digital converter (ADC4)*. Updated *Figure 245: General-purpose timer block diagram*, *Figure 436: IrDA SIR ENDEC block diagram*, *Figure 438: Transmission using DMA*, *Figure 439: Reception using DMA*, *Figure 450: Transmission using DMA*, and *Figure 451: Reception using DMA*. Added *Figure 385: Tamper sampling with precharge pulse* and *Figure 386: Low level detection with precharge and filtering*. Added *Table 177: RNG initialization times* and *Table 179: Configuration selection*. Removed former *Section 24.6.3: Data collection*. Removed former *Table 384: Error calculation for programmed baud rates at f_{CK} = 100 MHz*, *Table 394: SPI interconnection (SPI8)*, and *Table 395: SPI interconnection (SPI1 to SPI6)*. Added *Initialization time*, *Section 38.7: I2C DMA requests* and its subsections, and *Section 38.8: I2C debug modes*. Updated *Table 363: Timing settings for f_{I2CCLK} of 8 MHz*, *Table 364: Timing settings for f_{I2CCLK} of 16 MHz*, *Table 375: USART/UART input/output pins*, *Table 377: USART interconnection (USART1/2)*, *Table 389: LPUART interconnections (LPUART1)*, and *Table 396: SPI features*. Rearranged sequence of registers in *Section 42: Serial audio interface (SAI)*. Minor text edits across the whole document.

Table 443. Document revision history (continued)

Date	Revision	Changes
27-Sep-2024	5	Document scope extended to STM32WBA50xx devices. Updated *Introduction*, *Section 2.3.2: Memory map and register boundary addresses*, *Section 2.3.3: Embedded SRAM*, *Section 12.8.39: RCC Backup domain control register (RCC_BDCR1)*, *Section 12.8.40: RCC control/status register (RCC_CSR)*, *Section 17.4.17: GPDMA autonomous mode*, *Section 34.4.1: IWDG block diagram*, *Configuring the IWDG when the window option is disabled*, *Section 34.7.4: IWDG status register (IWDG_SR)*, *Section 36.6.24: RTC alarm A binary mode register (RTC_ALRABINR)*, *Section 36.6.24: RTC alarm A binary mode register (RTC_ALRABINR)*, *Active tamper detection*, *Section 37.6.5: TAMP active tamper control register 1 (TAMP_ATCR1)*, *Data frame format*, *Section 41.4.12: SPI data transmission and reception procedures*, and *Section 41.4.14: Communication using DMA (direct memory addressing)*. Updated *Figure 1: System architecture*, *Figure 34: Clock tree*, *Figure 377: Window comparator update*[(1)], and *Figure 379: Early wake-up comparator update*[(1)]. Updated *Table 5: Memory map and peripheral register boundary addresses* and its footnotes, *Table 34: SRAM parity access error*, *Table 75: 2.4 GHz RADIO implementation*, *Table 123: Peripherals interconnect matrix*, *Table 346: Active tamper output change period*, and *Table 348: Active tamper filtered pulse duration*. Minor text edits across the whole document.
16-Jan-2025	6	Added *Section 1.3: Register reset value*. Updated *Section 2.1.4: Bus matrix*, *Section 2.2.1: Default TrustZone security state*, *Section 5.6.5: GTZC1 TZSC privilege configuration register 1 (GTZC1_TZSC_PRIVCFGR1)*, *Section 5.6.6: GTZC1 TZSC privilege configuration register 2 (GTZC1_TZSC_PRIVCFGR2)*, *Section 5.6.7: GTZC1 TZSC privilege configuration register 3 (GTZC1_TZSC_PRIVCFGR3)*, *Section 7.3.2: Error code correction (ECC)*, *Section 9.1: Introduction*, *Section 10.3.3: PTACONV protocols*, *Section 10.3.4: PTACONV interface with the 2.4 GHz RADIO*, *Section 11.4.1: External power supplies*, *Exiting Standby mode*, *Section 14.4: GPIO functional description*, and *Section 14.4.13: GPIO analog configuration*. Replaced master/slave with controller/target in *Section 38: Inter-integrated circuit interface (I2C)*. Added *Table 1: Bus matrix access arbitration*. Updated *Table 5: Memory map and peripheral register boundary addresses*, *Table 81: PTACONV internal input/output signals*, *Table 257: Interconnect to the tim_ti3 input multiplexer*, *Table 273: Interrupt requests*, and *Table 442: DESIG register map and reset values*. Updated *Figure 24: PTACONV block diagram*, *Figure 25: 4-wire PTA grant protocol*, *Figure 26: 4-wire PTA deny protocol*, *Figure 27: 3-wire time-multiplexed PTA_STATUS*, *Figure 47: High-impedance analog configuration*, *Figure 207: tim_ocref_clr input selection multiplexer*, *Figure 285: OCREF_CLR input selection multiplexer*, and *Figure 422: Start bit detection when oversampling by 16 or 8*. Minor text edits across the whole document.

Table 443. Document revision history (continued)

Date	Revision	Changes
11-Mar-2025	7	Replaced Bluetooth® Low Energy with Bluetooth® LE. Updated document *Section 12.8.9: RCC clock interrupt enable register (RCC_CIER)*, *Section 40.7.8: LPUART interrupt and status register (LPUART_ISR)*, *Section 41.8.4: SPI configuration register 2 (SPI_CFG2)*, *Section 42.6.4: SAI slot register (SAI_ASLOTR)*, and *DBGMCU identity code register (DBGMCU_IDCODE)*. Added *Section 3.6.2: RSSLIB functions*, *Section 9.5: Low-power modes*, and *Section 10.4: Low-power modes*. Replaced SS with NSS in *Section 41: Serial peripheral interface (SPI)*. Updated *Table 99: RCC input/output signals connected to package pins or balls*, *Table 135: Vector table*, *Table 377: USART interconnection (USART1/2)*, *Table 389: LPUART interconnections (LPUART1)*, and *Table 441: DBGMCU register map and reset values*. Minor text edits across the whole document.

Index

D

E

F

G

H

I

L

P

R

S

T

U

W