RM0038
Reference manual

STM32L100xx, STM32L151xx, STM32L152xx and STM32L162xx advanced ARM-based 32-bit MCUs

Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32L151xx, STM32L152xx and STM32L162xx and STM32L100xx microcontroller memory and peripherals. The STM32L151xx, STM32L152xx and STM32L162xx and STM32L100xx value line will be referred to as STM32L1xxxx throughout the document, unless otherwise specified.

The STM32L1xxxx is a family of microcontrollers with different memory sizes, packages and peripherals.

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

For information on programming, erasing and protection of the internal Flash memory please refer to the *STM32L15xxx Flash programming manual.*

For information on the ARM Cortex™-M3 core, please refer to the *Cortex™-M3 Technical Reference Manual*.

Related documents

Available from *www.arm.com*:

- Cortex™-M3 Technical Reference Manual, available from:
- http://infocenter.arm.com/help/topic/com.arm.doc.ddi0337g/DDI0337G_cortex_m3_r2p0_trm.pdf

Available from *www.st.com*:

- STM32L151xx STM32L152xx datasheets
- STM32L162xx datasheet
- STM32L100xx datasheet
- STM32L15xxx Flash programming manual
- STM32F10xxx/20xxx/21xxx/L1xxxx Cortex-M3 programming manual

Table 1. Applicable products

Type	Part numbers
Microcontroller	STM32L100xx, STM32L151xx, STM32L152xx and STM32L162xx

Contents

List of tables

List of figures

1 Documentation conventions

1.1 List of abbreviations for registers

The following abbreviations are used in register descriptions:

read/write (rw)	Software can read and write to these bits.
read-only (r)	Software can only read these bits.
write-only (w)	Software can only write to this bit. Reading the bit returns the reset value.
read/clear (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 (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 by read (rc_r)	Software can read this bit. Reading this bit automatically clears it to ‘0. Writing ‘0 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-only write trigger (rt_w)	Software can read this bit. Writing ‘0 or ‘1 triggers an event but has no effect on the bit value.
toggle (t)	Software can only toggle this bit by writing ‘1. Writing ‘0 has no effect.
Reserved (Res.)	Reserved bit, must be kept at reset value.

1.2 Peripheral availability

For the peripherals available, and their number, across all STM32L1xxxx sales types, please refer to the STM32L1xxxx datasheet.

2 Memory and bus architecture

2.1 System architecture

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

- Up to five masters:
  - Cortex™-M3 I-bus, D-bus and S-bus
  - DMA1 and DMA2
- Up to five slaves:
  - Internal Flash memory ICode
  - Internal Flash memory DCode
  - Internal SRAM
  - AHB to APBx (APB1 or APB2), which connect all the APB peripherals
  - Flexible Static Memory Controller

These are interconnected using the multilayer AHB bus architecture shown in *Figure 1*:

Figure 1. System architecture (low and medium density devices)

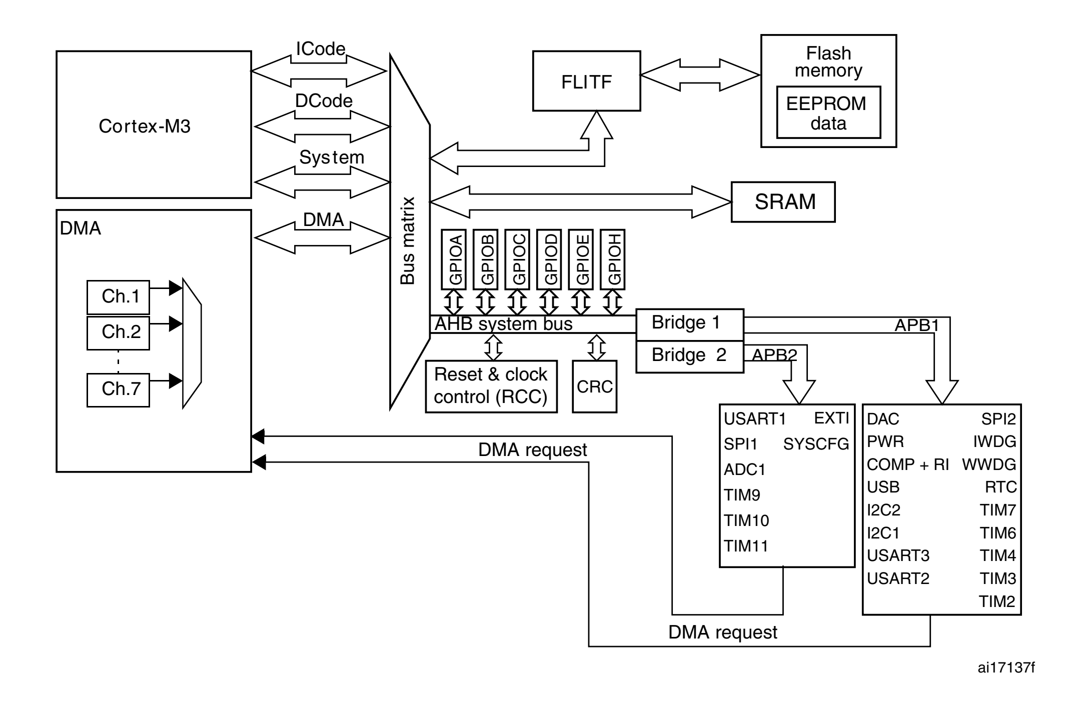


Figure 2. System architecture (medium+ density devices)

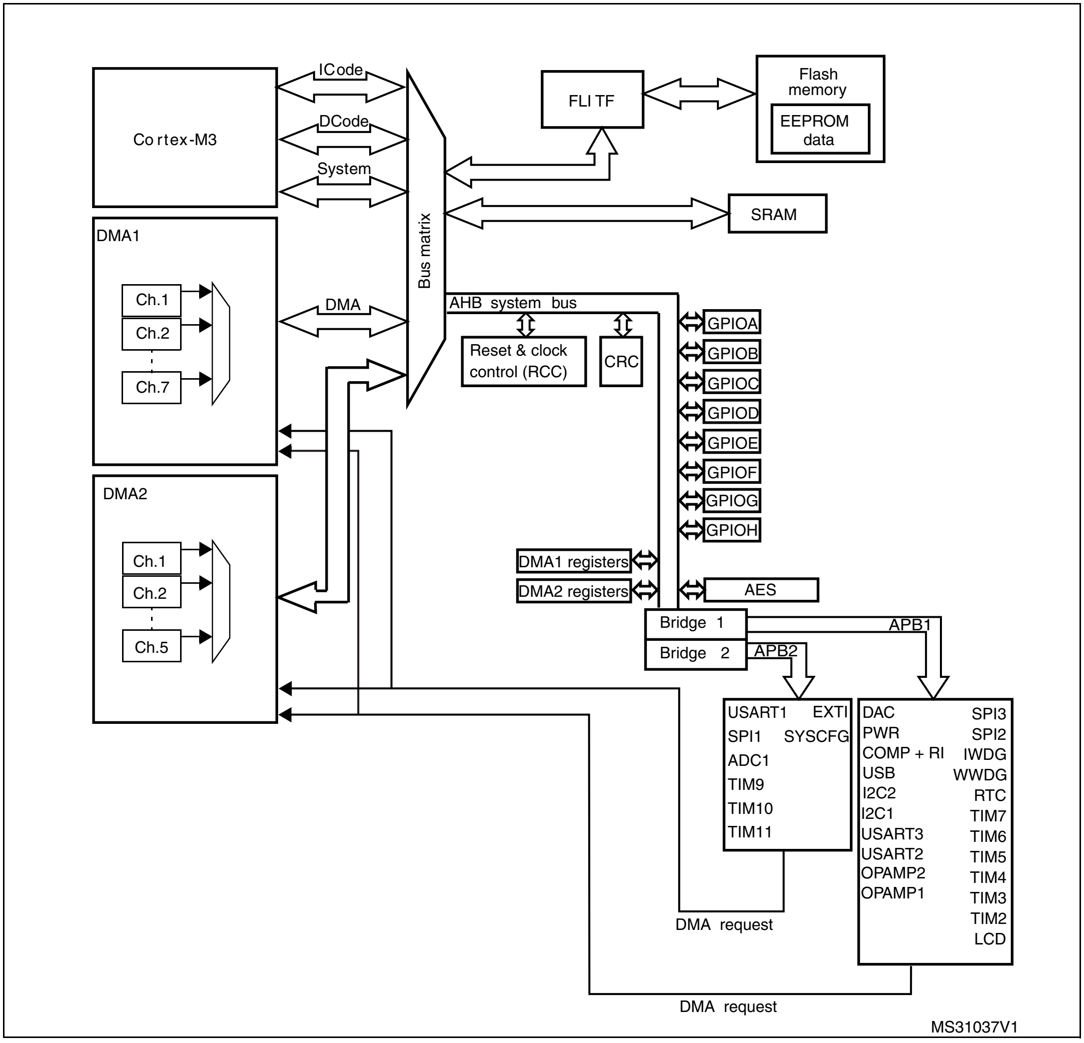
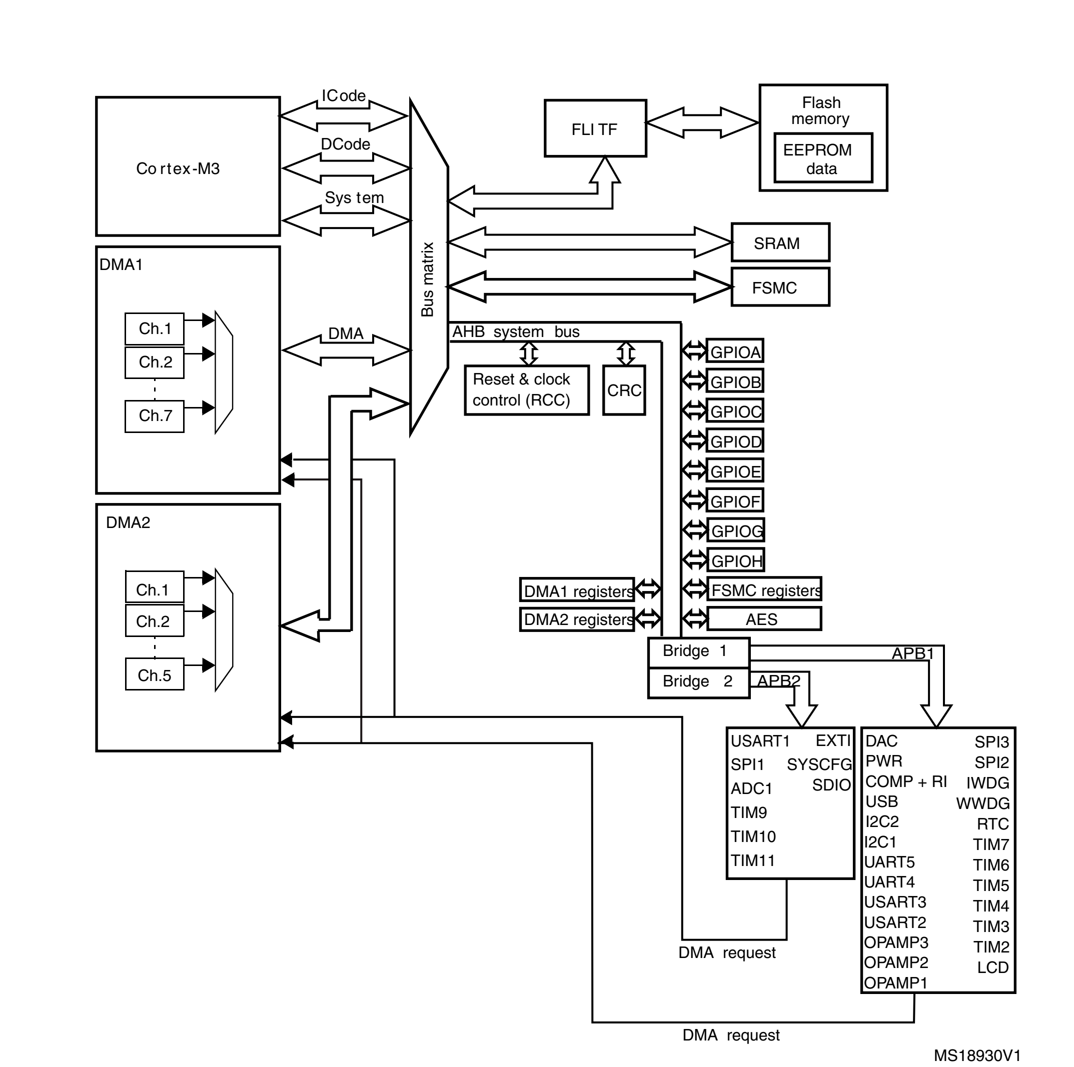


Figure 3. System architecture (high density devices)



ICode bus

This bus connects the Instruction bus of the Cortex™-M3 core to the BusMatrix. This bus is used by the core to fetch instructions. The target of this bus is a memory containing code (internal Flash memory or SRAM).

DCode bus

This bus connects the databus of the Cortex™-M3 to the BusMatrix. This bus is used by the core for literal load and debug access. The target of this bus is a memory containing code or data (internal Flash memory or SRAM).

System bus

This bus connects the system bus of the Cortex™-M3 core to a BusMatrix. This bus is used to access data located in a peripheral or in SRAM. Instructions may also be fetched on this bus (less efficient than ICode). The targets of this bus are the internal SRAM and the AHB/APB bridges.

DMA bus

This bus connects the AHB master interface of the DMA to the bus matrix which manages the access of the CPU DCode and DMA to the SRAM, Flash memory and peripherals.

Bus matrix

The bus matrix manages the access arbitration between the core system bus and the DMA master bus. The arbitration uses a round robin algorithm. The bus matrix is composed of five masters (ICode, DCode, System bus, DMA1 bus, DMA2 bus) and five slaves (Flash ICode interface, Flash DCode interface, SRAM, FSMC, and AHB2APB bridges).

AHB peripherals are connected on the system bus through the bus matrix to allow DMA access.

AHB/APB bridges (APB)

The two AHB/APB bridges provide full synchronous connections between the AHB and the 2 APB buses. The two APB buses operates at full speed (up to 32 MHz).

Refer to *Table 2 on page 42* for the address mapping of the AHB and APB peripherals.

After each device reset, all peripheral clocks are disabled (except for the SRAM and Flash interface). Before using a peripheral, its clock should be enabled in the RCC_AHBENR, RCC_APB1ENR or RCC_APB2ENR register.

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

2.2 Memory organization

Flash program memory, EEPROM data memory, SRAM 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.

For the detailed mapping of peripheral registers, please refer to the related sections.

The addressable memory space is divided into 8 main blocks, each of 512 Mbytes.

All the memory areas that are not allocated to on-chip memories and peripherals are considered “Reserved”. Refer to the memory map figure in the STM32L1xxxx datasheet.

2.3 Memory map

See the STM32L1xxxx datasheet for a comprehensive diagram of the memory map. *Table 2* gives the boundary addresses of the peripherals available in STM32L1xxxx devices.

Table 2. Register boundary addresses

Boundary address	Peripheral	Bus	Register map
0xA000 0000 - 0xA000 0FFF	FSMC	AHB	*Section 24.5.7: FSMC register map on page 616*
0x5006 0000 - 0x5006 03FF	AES		*Section 22.12.13: AES register map on page 544*
0x4002 6400 - 0x4002 67FF	DMA2		*Section 10.4.7: DMA register map on page 221*
0x4002 6000 - 0x4002 63FF	DMA1		*Section 10.4.7: DMA register map on page 221*
0x4002 3C00 - 0x4002 3FFF	Flash memory interface		See Flash programming manual
0x4002 3800 - 0x4002 3BFF	RCC		*Section 5.3.15: RCC register map on page 128*
0x4002 3000 - 0x4002 33FF	CRC		*Section 3.4.4: CRC register map on page 57*
0x4002 1C00 - 0x4002 1FFF	GPIOG		*Section 6.4.11: GPIO register map on page 148*
0x4002 1800 - 0x4002 1BFF	GPIOF		
0x4002 1400 - 0x4002 17FF	GPIOH		
0x4002 1000 - 0x4002 13FF	GPIOE		
0x4002 0C00 - 0x4002 0FFF	GPIOD		
0x4002 0800 - 0x4002 0BFF	GPIOC		
0x4002 0400 - 0x4002 07FF	GPIOB		
0x4002 0000 - 0x4002 03FF	GPIOA		
0x4001 3800 - 0x4001 3BFF	USART1	APB2	*Section 26.6.8: USART register map on page 700*
0x4001 3000 - 0x4001 33FF	SPI1		*Section 27.5.10: SPI register map on page 754*
0x4001 2C00 - 0x4001 2FFF	SDIO		*Section 28.9.16: SDIO register map on page 810*
0x4001 2400 - 0x4001 27FF	ADC		*Section 11.15.21: ADC register map on page 265*
0x4001 1000 - 0x4001 13FF	TIM11		*Section 14.4.17: TIMx register map on page 368*
0x4001 0C00 - 0x4001 0FFF	TIM10		*Section 14.4.17: TIMx register map on page 368*
0x4001 0800 - 0x4001 0BFF	TIM9		*Section 14.4.17: TIMx register map on page 368*
0x4001 0400 - 0x4001 07FF	EXTI		*Section 9.3.7: EXTI register map on page 203*
0x4001 0000 - 0x4001 03FF	SYSCFG		*Section 7.5.7: SYSCFG register map on page 181*

Table 2. Register boundary addresses (continued)

Boundary address	Peripheral	Bus	Register map
0x4000 7C00 - 0x4000 7C03	COMP	APB1	Section 13.9.2: COMP register map on page 300
0x4000 7C04 - 0x4000 7C5B	RI		Section 7.5.7: SYSCFG register map on page 181
0x4000 7C5C - 0x4000 7FFF	OPAMP		Section 14.4.4: OPAMP register map on page 310
0x4000 7400 - 0x4000 77FF	DAC		Section 12.5.15: DAC register map on page 288
0x4000 7000 - 0x4000 73FF	PWR		Section 4.4.3: PWR register map on page 85
0x4000 6000 - 0x4000 63FF	USB device FS SRAM 512 bytes		Section 23.5.4: USB register map on page 575
0x4000 5C00 - 0x4000 5FFF	USB device FS		
0x4000 5800 - 0x4000 5BFF	I2C2		Section 25.6.10: I2C register map on page 648
0x4000 5400 - 0x4000 57FF	I2C1		
0x4000 5000 - 0x4000 53FF	USART5		Section 26.6.8: USART register map on page 700
0x4000 4C00 - 0x4000 4FFF	USART4		
0x4000 4800 - 0x4000 4BFF	USART3		
0x4000 4400 - 0x4000 47FF	USART2		
0x4000 3C00 - 0x4000 3FFF	SPI3		Section 27.5.10: SPI register map on page 754
0x4000 3800 - 0x4000 3BFF	SPI2		
0x4000 3000 - 0x4000 33FF	IWDG		Section 20.4.5: IWDG register map on page 511
0x4000 2C00 - 0x4000 2FFF	WWDG		Section 21.6.4: WWDG register map on page 518
0x4000 2800 - 0x4000 2BFF	RTC		Section 19.6.21: RTC register map on page 503
0x4000 2400 - 0x4000 27FF	LCD		Section 15.5.6: LCD register map on page 337
0x4000 1400 - 0x4000 17FF	TIM7		Section 18.4.9: TIM6&TIM7 register map on page 462
0x4000 1000 - 0x4000 13FF	TIM6		
0x4000 0C00 - 0x4000 0FFF	TIM5 (32-bits)		Section 16.4.21: TIMx register map on page 396
0x4000 0800 - 0x4000 0BFF	TIM4		
0x4000 0400 - 0x4000 07FF	TIM3		
0x4000 0000 - 0x4000 03FF	TIM2		

2.3.1 Embedded SRAM

The STM32L1xxxx features up to 48 Kbytes of SRAM. It can be accessed as bytes, half-words (16 bits) or full words (32 bits). The SRAM start address is 0x2000 0000.

Read and write access at CPU speed with 0 wait states.

The CPU can access the SRAM through the system bus or through the I-Code/D-Code bus when boot in SRAM is selected or when physical remap is selected (see *Section 7.5.1: SYSCFG memory remap register (SYSCFG_MEMRMP)* register in the SYSCFG controller). To get the best SRAM execution performance, physical remap must be selected (boot or software selection).

2.3.2 Bit banding

The Cortex™-M3 memory map includes two bit-band regions. These regions map each word in an alias region of memory to a bit in a bit-band region of memory. Writing to a word in the alias region has the same effect as a read-modify-write operation on the targeted bit in the bit-band region.

In the STM32L1xxxx both the peripheral registers and the SRAM are mapped in a bit-band region. This allows single bit-band write and read operations to be performed. These operations are only available for Cortex-M3 accesses, not from other bus masters (e.g. DMA).

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

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

where:

bit_word_addr is the address of the word in the alias memory region that maps to the targeted bit

bit_band_base is the start address of the alias region

byte_offset is the number of the byte in the bit-band region, that contains the targeted bit

bit_number is the bit position (0-7) of the targeted bit

Example:

The following example shows how to map bit 2 of the byte located at SRAM address 0x2000 0300 in the alias region:

0x2200 6008 = 0x2200 0000 + (0x300 × 32) + (2 × 4)

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

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

For more information on bit-banding, please refer to the *Cortex-M3 Technical Reference Manual*.

2.3.3 Embedded Flash memory

- Up to 384 Kbytes of Flash memory
- Memory organization (dual bank; for high density device only):
  - Up to 384 Kbytes of Flash program memory and up to 12 Kbytes of data EEPROM
  - Up to 8 Kbytes of system memory and up to 64 bytes of option bytes

Each bank is organised as follows:

- 192 Kbytes of program memory and 6 Kbytes of data
- 4 Kbytes of system memory, 32 bytes of option bytes

Flash memory interface (FLITF) features:

- Flash memory read operations: read access is performed by 64 or 32 bits
- Flash memory program/erase operations
- Read while write (RWW) from one bank to the other
- Read/write protection
- Write access is performed by 32 bits
- Option byte loader reset
- Low power mode:
  - Flash memory in Power down mode when the STM32L1xxxx is in the Standby mode or the Stop mode
  - Flash memory can be placed in Power down or Idle mode when the STM32L1xxxx is in the Sleep mode
  - Flash memory can be placed in Power down or Idle mode when the STM32L1xxxx is in the Run mode

Flash module organization

The memory is organized as a main program memory block, a data memory block of 1536 double words, and an information block. *Table 5* shows the Flash memory organization.

The program memory block is divided into 96 sectors of 4 Kbytes each, and each sector is further split up into 16 pages of 256 bytes each. The sector is the write protection granularity. In total, the program memory block contains 1536 pages.

Table 3. Flash module organization (low and medium density devices)

Block	Flash memory addresses	Size	Name	Description
Program memory	0x0800 0000 - 0x0800 00FF	256 bytes	Page 0	Sector 0
	0x0800 0100 - 0x0800 01FF	256 bytes	Page 1	
	0x0800 0200 - 0x0800 02FF	256 bytes	Page 2	
	0x0800 0300 - 0x0800 03FF	256 bytes	Page 3	
	0x0800 0400 - 0x0800 07FF	1 Kbyte	Page 4 to 7	
	0x0800 0800 - 0x0800 0BFF	1 Kbyte	Page 8 to 11	
	0x0800 0C00 - 0x0800 0FFF	1 Kbyte	Page 12 to 15	
	0x0800 1000 - 0x0800 1FFF	4 Kbytes	Page 16 to 31	Sector 1
	0x0800 2000 - 0x0800 2FFF	4 Kbytes	Page 32 to47	Sector 2
	0x0800 3000 - 0x0800 3FFF	4 Kbytes	Page 48 to 63	Sector 3
	. . .	. . .	. . .	. . .
	0x0801 E000 - 0x0801 EFFF	4 Kbytes	Page 478 to 495	Sector 30
	0x0801 F000 - 0x0801 FFFF	4 Kbytes	Page 496 to 511	Sector 31
Data memory	0x0808 0000 - 0x0808 0FFF	4096 bytes	DATA	Data memory
Information Block	0x1FF0 0000 - 0x1FF0 00FF	256 bytes	Page 0	System memory
	0x1FF0 0100 - 0x1FF0 01FF	256 bytes	Page 1	
	0x1FF0 0200 - 0x1FF0 02FF	256 bytes	Page 2	
	0x1FF0 0300 - 0x1FF0 03FF	256 bytes	Page 3	
	. . .	. . .	. . .	
	0x1FF0 0F00 - 0x1FF0 0FFF	256 bytes	Page 15	
	0x1FF8 0000 - 0x1FF8 000F	16 bytes	OPTB	Option bytes block

Table 4. Flash module organization (medium+ density devices)

Block	Flash memory addresses	Size	Name	Description
Program memory	0x0800 0000 - 0x0800 00FF	256 bytes	Page 0	Sector 0
	0x0800 0100 - 0x0800 01FF	256 bytes	Page 1	
	0x0800 0200 - 0x0800 02FF	256 bytes	Page 2	
	0x0800 0300 - 0x0800 03FF	256 bytes	Page 3	
	0x0800 0400 - 0x0800 07FF	1 Kbytes	Page 4 to 7	
	0x0800 0800 - 0x0800 0BFF	1 Kbytes	Page 8 to 11	
	0x0800 0C00 - 0x0800 0FFF	1 Kbytes	Page 12 to 15	
	0x0800 1000 - 0x0800 1FFF	4 Kbytes	Page 16 to 31	Sector 1
	0x0800 2000 - 0x0800 2FFF	4 Kbytes	Page 32 to47	Sector 2
	0x0800 3000 - 0x0800 3FFF	4 Kbytes	Page 48 to 63	Sector 3
	. . .	. . .	. . .	. . .
	0x0801 E000 - 0x0801 EFFF	4 Kbytes	Page 478 to 495	Sector 30
	0x0801 F000 - 0x0801 FFFF	4 Kbytes	Page 496 to 511	Sector 31
	0x0802 0000 - 0x0802 FFFF	64 Kbytes	Page 512 to 767	Sector 32 to Sector 47
	0x0803 0000 - 0x0803 FFFF	64 Kbytes	Page 768 to 1024	Sector 48 to Sector 63
Data memory	0x0808 0000 - 0x0808 1FFF	8 Kbytes	DATA	Data memory
Information Block	0x1FF0 0000 - 0x1FF0 00FF	256 bytes	Page 0	System memory
	0x1FF0 0100 - 0x1FF0 01FF	256 bytes	Page 1	
	0x1FF0 0200 - 0x1FF0 02FF	256 bytes	Page 2	
	0x1FF0 0300 - 0x1FF0 03FF	256 bytes	Page 3	
	. . .	. . .	. . .	
	0x1FF0 0F00 - 0x1FF0 0FFF	256 bytes	Page 15	
	0x1FF0 1000 - 0x1FF0 1FFF	4 Kbytes	Page 16 to 31	
	0x1FF8 0000 - 0x1FF8 001F	32 bytes	OPTB	Option bytes

Table 5. Flash module organization (high density devices)

Block	Flash memory addresses	Size	Name	Description
Program memory	0x0800 0000 - 0x0800 00FF	256 bytes	Page 0	Sector 0
	0x0800 0100 - 0x0800 01FF	256 bytes	Page 1	
	0x0800 0200 - 0x0800 02FF	256 bytes	Page 2	
	0x0800 0300 - 0x0800 03FF	256 bytes	Page 3	
	0x0800 0400 - 0x0800 07FF	1 Kbytes	Page 4 to 7	
	0x0800 0800 - 0x0800 0BFF	1 Kbytes	Page 8 to 11	
	0x0800 0C00 - 0x0800 0FFF	1 Kbytes	Page 12 to 15	
	0x0800 1000 - 0x0800 1FFF	4 Kbytes	Page 16 to 31	Sector 1
	0x0800 2000 - 0x0800 2FFF	4 Kbytes	Page 32 to47	Sector 2
	0x0800 3000 - 0x0800 3FFF	4 Kbytes	Page 48 to 63	Sector 3
	. . .	. . .	. . .	. . .
	0x0801 E000 - 0x0801 EFFF	4 Kbytes	Page 478 to 495	Sector 30
	0x0801 F000 - 0x0801 FFFF	4 Kbytes	Page 496 to 511	Sector 31
	0x0802 0000 - 0x0802 FFFF	64 Kbytes	Page 512 to 767	Sector 32 to Sector 47
	0x0803 0000 - 0x0804 FFFF	128 Kbytes	Page 768 to 1279	Sector 48 to Sector 79
	0x0805 0000 - 0x0805 FFFF	64 Kbytes	Page 1278 to 1535	Sector 80 to Sector 95
Data memory	0x0808 0000 - 0x0808 17FF	6 Kbytes	DATA Bank 1	Data EEprom Bank 1
	0x0808 1800 - 0x0808 2FFF	6 Kbytes	DATA Bank 2	Data EEprom Bank 2
Information Block	0x1FF0 0000 - 0x1FF0 00FF	256 bytes	Page 0	System memory Bank 1
	0x1FF0 0100 - 0x1FF0 01FF	256 bytes	Page 1	
	0x1FF0 0200 - 0x1FF0 02FF	256 bytes	Page 2	
	0x1FF0 0300 - 0x1FF0 03FF	256 bytes	Page 3	
	. . .	. . .	. . .	
	0x1FF0 0F00 - 0x1FF0 0FFF	256 bytes	Page 15	
	0x1FF0 1000 - 0x1FF0 1FFF	4 Kbytes	Page 16 to 31	System memory Bank 2
	0x1FF8 0000 - 0x1FF8 001F	32 bytes	OPTB	Option bytes Bank 1
	0x1FF8 0080 - 0x1FF8 009F	32 bytes	OPTB	Option bytes Bank 2

Reading the Flash memory

Relation between CPU clock frequency and Flash memory read time

The Flash memory is read by 64 bits or 32 bits.

64-bit access is configured by setting the ACC64 bit in the Flash access control register (FLASH_ACR). This access mode accelerates the execution of program operations. Prefetch is useful when the Flash memory cannot be accessed for a CPU cycle. In this case, the number of wait states (LATENCY) must be correctly programmed in the Flash access control register (FLASH_ACR) according to the frequency of the Cortex-M3 clock and the supply voltage of the device. *Table 6* shows the correspondence between wait states and core clock frequency.

Table 6. Number of wait states (WS) according to CPU clock (HCLK) frequency

HCLK frequency (MHz)			Wait states (LATENCY)
Voltage range 1.65 V to 3.6 V		Voltage range 2.0 V to 3.6 V	
V_{CORE} = 1.2 V	V_{CORE} = 1.5 V	V_{CORE} = 1.8 V	
$0 < f_{HCLK} \leq 2.1$ MHz	$0 < f_{HCLK} \leq 8$ MHz	$0 < f_{HCLK} \leq 16$ MHz	0 WS (1 HCLK cycle)
$2 < f_{HCLK} \leq 4.2$ MHz	$8 < f_{HCLK} \leq 16$ MHz	$16 < f_{HCLK} \leq 32$ MHz	1 WS (2 HCLK cycles)

It is also possible to access the Flash memory by 32 bits. This is done by clearing the ACC64 bit in FLASH_ACR. In this case, prefetch has to be disabled. 32-bit access reduces the consumption, so it is used when the CPU frequency is low. In this case, the number of wait states must be 0.

After reset, the used clock is the MSI (2 MHz) with 0 WS configured in the FLASH_ACR register. 32-bit access is enabled and prefetch is disabled.

ST strongly recommends to use the following software sequences to tune the number of wait states needed to access the Flash memory with the CPU frequency.

Increasing the CPU frequency (in the same voltage range)

- Program the 64-bit access by setting the ACC64 bit in *Flash access control register (FLASH_ACR)*
- Check that 64-bit access is taken into account by reading FLASH_ACR
- Program 1 WS to the LATENCY bit in FLASH_ACR
- Check that the new number of WS is taken into account by reading FLASH_ACR
- Modify the CPU clock source by writing to the SW bits in the *Clock configuration register (RCC_CFGR)*
- If needed, modify the CPU clock prescaler by writing to the HPRE bits in RCC_CFGR
- Check that the new CPU clock source or/and the new CPU clock prescaler value is/are taken into account by reading the clock source status (SWS bits) or/and the AHB prescaler value (HPRE bits), respectively, in the RCC_CFGR register

Decreasing the CPU frequency (in the same voltage range)

- Modify the CPU clock source by writing to the SW bits in the *Clock configuration register (RCC_CFGR)*
- If needed, modify the CPU clock prescaler by writing to the HPRE bits in RCC_CFGR
- Check that the new CPU clock source or/and the new CPU clock prescaler value is/are taken into account by reading the clock source status (SWS bits) or/and the AHB prescaler value (HPRE bits), respectively, in the RCC_CFGR register
- Program the new number of WS to the LATENCY bit in *Flash access control register (FLASH_ACR)*
- Check that the new number of WS is taken into account by reading FLASH_ACR
- Program the 32-bit access by clearing ACC64 in FLASH_ACR
- Check that 32-bit access is taken into account by reading FLASH_ACR

Instruction prefetch when Flash access is 64 bits

Each Flash memory read operation provides 64 bits from either two 32-bit instructions or four 16-bit instructions. So, in case of a sequential code, at least 2 CPU cycles are needed to read the previous instruction line. Prefetch on the ICode bus can be used to read the next sequential instruction line from the Flash memory while the current instruction line is being requested by the CPU. Prefetch is enabled by setting the PRFTEN bit in the FLASH_ACR register. This feature is useful if at least one wait state is needed to access the Flash memory.

Table 7 shows the supported ACC64, LATENCY and PRFTEN configurations.

Table 7. Allowed configuration in FLASH_ACR

LATENCY	ACC64 = 0		ACC64 = 1	
	PRFTEN = 0	PRFTEN = 1	PRFTEN = 0	PRFTEN = 1
0	OK	-	OK	OK
1	-	-	OK	OK

Note: The Flash access control register (FLASH_ACR) is used to control the Flash state in run/sleep modes, the prefetch status and access time depending on the CPU frequency. The table above provides the bit map for this register.

For complete information on Flash memory operations, please refer to the STM32L1xxxx Flash programming manual (PM0076). For details on register configurations, refer to the following section (Flash access control register (FLASH_ACR) on page 52).

Flash access control register (FLASH_ACR)

Address offset: 0x00

Reset value: 0x0000 0000

<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><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 colspan="27" rowspan="2">Reserved</td><td>RUN_PD</td><td>SLEEP_PD</td><td>ACC64</td><td>PRFTEN</td><td>LATENCY</td></tr>
<tr><td>rw</td><td>rw</td><td>rw</td><td>rw</td><td>rw</td></tr>
</table>

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RUN_PD:** Flash mode during Run

This bit can be written only when it is unlocked by writing to FLASH_PDKEYR.

This bit determines whether the Flash memory is in Power down mode or Idle mode when the STM32L1xxxx is in Run mode.

The Flash memory can be placed in Power down mode only when the code is executed from RAM).

0: Flash in Idle mode

1: Flash in Power down mode

Bit 3 **SLEEP_PD:** Flash mode during Sleep

This bit is used to have the Flash memory in Power down mode or Idle mode when the STM32L1xxxx is in Sleep mode.

0: Flash in Idle mode

1: Flash in Power down mode

Bit 2 **ACC64**: 64-bit access

This bit is used to read data from the Flash memory 64 bits or 32 bits at a time. 32-bit access is used to decreases the Flash memory consumption. On the contrary, 64-bit access is used to improve the performance. In this case it is useful to enable prefetch.

0: 32-bit access

1: 64-bit access

Note: 32-bit access is a low power mode. It is used only at low frequencies, that is with 0 wait state of latency and prefetch off.

Note: This bit cannot be written at the same time as the LATENCY and PRFTEN bits.

Bit 1 **PRFTEN**: Prefetch enable

0: prefetch disabled

1: prefetch enabled

Note: Prefetch can be enabled only when ACC64 is set.
This bit can be set or cleared only if ACC64 is set.

Bit 0 **LATENCY**: Latency

This bit represents the ratio of the CPU clock period to the Flash access time.

0: zero wait state

1: one wait state

Note: Latency can be set only when ACC64 is set.
This bit can be set or cleared only if ACC64 is set.

Flash interface register map

The following table summarizes the Flash memory registers.

Table 8. Flash interface register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	FLASH_ACR	Reserved																											RUN_PD	SLEEP_PD	Acc64	PRFTEN	LATENCY0
	Reset value: 0x0000 0000																												0	0	0	0	0

2.4 Boot configuration

Due to its fixed memory map, the code area starts from address 0x0000 0000 (accessed through the ICode/DCode buses) while the data area (SRAM) starts from address 0x2000 0000 (accessed through the system bus). The Cortex™-M3 CPU always fetches the reset vector from the ICode bus, which implies to have the boot space available only in the code area (typically, Flash memory). STM32L1xxxx microcontrollers implement a special mechanism to be able to boot from other memory than the Flash (like internal SRAM).

In the STM32L1xxxx, 3 different boot modes can be selected through the BOOT[1:0] pins as shown in *Table 9*.

Table 9. Boot modes

Boot mode selection pins		Boot mode	Aliasing
BOOT1	BOOT0		
x	0	Main Flash memory	Main Flash memory is selected as the boot space
0	1	System memory	System memory is selected as the boot space
1	1	Embedded SRAM	Embedded SRAM is selected as the boot space

The values on the BOOT pins are latched on the 4th rising edge of SYSCLK after a reset. It is up to the user to set the BOOT1 and BOOT0 pins after reset to select the required boot mode.

BOOT0 is a dedicated pin while BOOT1 is shared with a GPIO pin. Once BOOT1 has been sampled, the corresponding GPIO pin is free and can be used by the application.

The BOOT pins are also resampled when exiting the Standby mode. Consequently they must be kept in the required Boot mode configuration in Standby mode. After this startup delay has elapsed, the CPU fetches the top-of-stack value from address 0x0000 0000, then starts code execution from the boot memory starting from 0x0000 0004.

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

Physical remap

When the boot pins are configured as desired, the application software can modify the memory accessible in the code area (code can thus be executed through the ICode/DCode in place of the System bus). This modification is performed by programming the *SYSCFG memory remap register (SYSCFG_MEMRMP)* in the SYSCFG controller.

The following memory can then be remapped:

- Main Flash memory
- System memory
- Embedded SRAM

Note: Depending on the memory protection programmed by option byte, some boot/remap configurations may not be available (refer to the readout protection section in programming manual PM0076 for details).

Table 10. Memory mapping vs. boot mode/physical remap

Addresses	Boot/Remap in main Flash memory	Boot/Remap in embedded SRAM	Boot/Remap in System memory
0x2000 0000 - 0x2000 BFFF	SRAM	SRAM	SRAM
0x1FF0 0000 - 0x1FF0 1FFF	System memory	System memory	System memory
0x0802 0000 - 0x0FFF FFFF	Reserved	Reserved	Reserved
0x0800 0000 - 0x0805 FFFF	Flash memory	Flash memory	Flash memory
0x0002 0000 - 0x07FF FFFF	Reserved	Reserved	Reserved
0x0000 0000 - 0x0005 FFFF	Flash (up to 384 KB) Aliased	SRAM Aliased	System memory (8 KB) Aliased

Note: Even when aliased in the boot memory space, the related memory is still accessible at its original memory space.

Embedded boot loader

The embedded boot loader is used to reprogram the Flash memory through one of the following interfaces:

- In low and medium density devices: USART1 or USART2.
- In high and medium+ density devices: USART1, USART2 or USB

This program is located in the system memory and is programmed by ST during production.

3 CRC calculation unit

This section applies to the whole STM32L1xxxx family, unless otherwise specified.

3.1 CRC introduction

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from a 32-bit data word and a fixed generator polynomial.

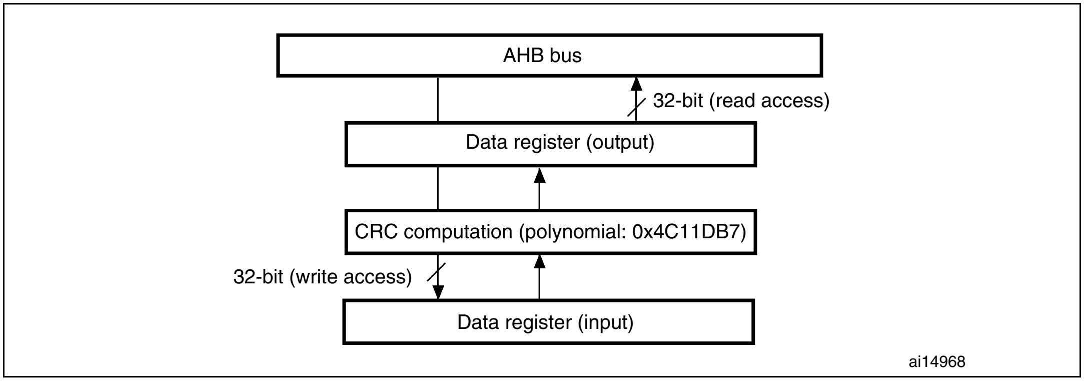
Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the EN/IEC 60335-1 standard, they offer a means of verifying the Flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link-time and stored at a given memory location.

3.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
  - $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Single input/output 32-bit data register
- CRC computation done in 4 AHB clock cycles (HCLK)
- General-purpose 8-bit register (can be used for temporary storage)

The block diagram is shown in *Figure 4*.

Figure 4. CRC calculation unit block diagram



3.3 CRC functional description

The CRC calculation unit mainly consists of a single 32-bit data register, which:

- is used as an input register to enter new data in the CRC calculator (when writing into the register)
- holds the result of the previous CRC calculation (when reading the register)

Each write operation into the data register creates a combination of the previous CRC value and the new one (CRC computation is done on the whole 32-bit data word, and not byte per byte).

The write operation is stalled until the end of the CRC computation, thus allowing back-to-back write accesses or consecutive write and read accesses.

The CRC calculator can be reset to 0xFFFF FFFF with the RESET control bit in the CRC_CR register. This operation does not affect the contents of the CRC_IDR register.

3.4 CRC registers

The CRC calculation unit contains two data registers and a control register.The peripheral The CRC registers have to be accessed by words (32 bits).

3.4.1 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 **Data register bits**

Used as an input register when writing new data into the CRC calculator.
Holds the previous CRC calculation result when it is read.

3.4.2 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								IDR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **General-purpose 8-bit data register bits**

Can be used as a temporary storage location for one byte.
This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register.

3.4.3 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
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															RESET
															w

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

Bit 0 **RESET bit**

Resets the CRC calculation unit and sets the data register to 0xFFFF FFFF.
This bit can only be set, it is automatically cleared by hardware.

3.4.4 CRC register map

The following table provides the CRC register map and reset values.

Table 11. CRC calculation unit register map and reset values

Offset	Register	31-24	23-16	15-8	7	6	5	4	3	2	1	0
0x00	**CRC_DR**	Data register										
	Reset value	0xFFFF FFFF										
0x04	**CRC_IDR**	Reserved			Independent data register							
	Reset value				0x00							
0x08	**CRC_CR**	Reserved										RESET
	Reset value											0

4 Power control (PWR)

4.1 Power supplies

The device requires a 1.8-to-3.6 V V_{DD} operating voltage supply (down to 1.65 V at power down) when the BOR is available. The device requires a 1.65-to-3.6 V V_{DD} operating voltage supply when the BOR is not available.

An embedded linear voltage regulator is used to supply the internal digital power, ranging from 1.2 to 1.8 V.

- V_{DD} = 1.8 V (at power on) or 1.65 V (at power down) to 3.6 V when the BOR is available. V_{DD} = 1.65 V to 3.6 V, when BOR is not available

  V_{DD} is the external power supply for I/Os and internal regulator. It is provided externally through V_{DD} pins
- V_{CORE} = 1.2 to 1.8 V

  V_{CORE} is the power supply for digital peripherals, SRAM and Flash memory. It is generated by a internal voltage regulator. Three V_{CORE} ranges can be selected by software depending on V_{DD} (refer *Figure 6*).
- V_{SSA}, V_{DDA} = 1.8 V (at power on) or 1.65 V (at power down) to 3.6 V, when BOR is available and V_{SSA}, V_{DDA} = 1.65 to 3.6 V, when BOR is not available.

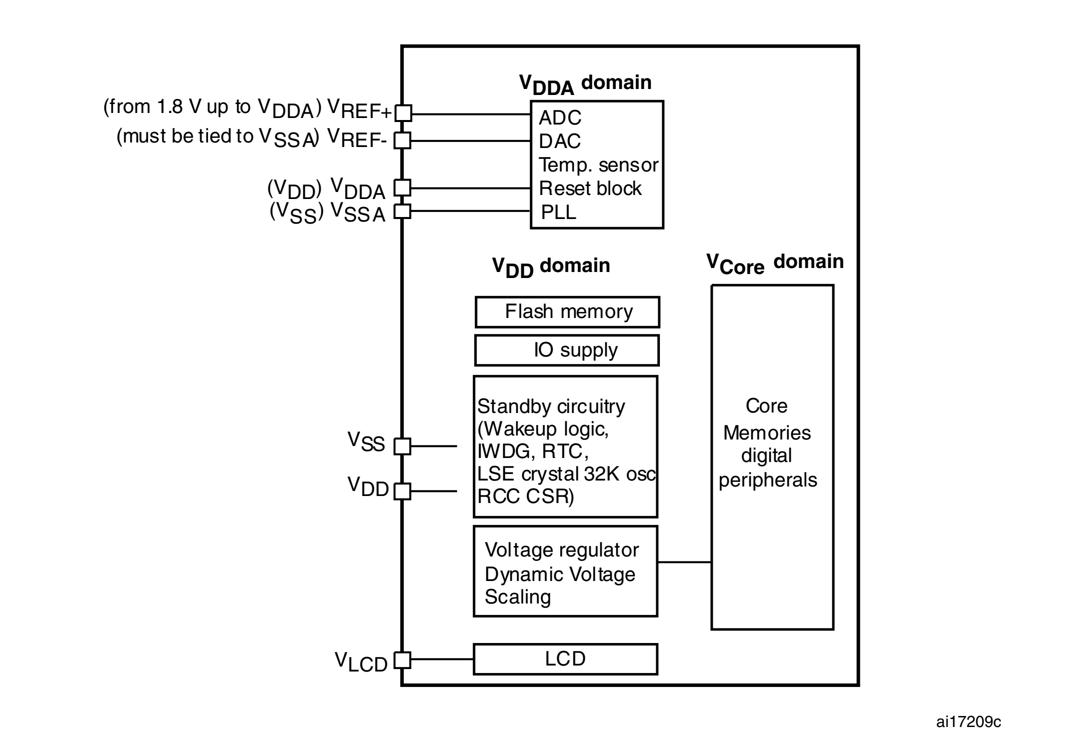
  V_{DDA} is the external analog power supply for ADC, DAC, reset blocks, RC oscillators and PLL. The minimum voltage to be applied to V_{DDA} is 1.8 V when the ADC is used.
- V_{REF-}, V_{REF+}

  V_{REF+} is the input reference voltage.

  V_{REF-} and V_{REF+} are only available as external pins on LQFP144, UFBGA132, LQFP100, UFBGA100, and TFBGA64 packages, otherwise they are bonded to V_{SSA} and V_{DDA}, respectively.
- V_{LCD} = 2.5 to 3.6 V

  The LCD controller can be powered either externally through V_{LCD} pin, or internally from an internal voltage generated by the embedded step-up converter.

Figure 5. Power supply overview



1. V_{DDA} and V_{SSA} must be connected to V_{DD} and V_{SS}, respectively.
2. When available (depending on packages), V_{REF-} must be tied to V_{SSA}.
3. Depending on the operating power supply range used, some peripherals may be used with limited functionalities or performance. For more details, please refer to section "General operating conditions" in STM32L1xxxx datasheets.

4.1.1 Independent A/D and DAC converter supply and reference voltage

To improve conversion accuracy, the ADC and the DAC have an independent power supply that can be filtered separately, and shielded from noise on the PCB.

- The ADC voltage supply input is available on a separate V_{DDA} pin
- An isolated supply ground connection is provided on the V_{SSA} pin

On BGA 64-pin and all packages with 100 pins or more

To ensure a better accuracy on low-voltage inputs and outputs, the user can connect to V_{REF+} a separate external reference voltage lower than V_{DD}. V_{REF+} is the highest voltage, represented by the full scale value, for an analog input (ADC) or output (DAC) signal.

- For ADC
  - $2.4\ V \leq V_{REF+} = V_{DDA}$ for full speed (ADCCLK = 16 MHz, 1 Msps)
  - $1.8\ V \leq V_{REF+} = V_{DDA}$ for medium speed (ADCCLK = 8 MHz, 500 Ksps)
  - $2.4\ V \leq V_{REF+} \neq V_{DDA}$ for medium speed (ADCCLK = 8 MHz, 500 Ksps)
  - $1.8\ V \leq V_{REF+} < V_{DDA}$ for low speed (ADCCLK = 4 MHz, 250 Ksps)
  - When Product voltage range 3 is selected (VCore = 1.2 V) the ADC is low speed (ADCCLK = 4 MHz, 250 Ksps)
- For DAC
  - $1.8\ V \leq V_{REF+} < V_{DDA}$
- When V_{DDA} is higher than 2.4 V, the voltage on V_{REF+} may range from 2.4 V to V_{DDA}.
- When V_{DDA} is below 2.4 V, V_{REF+} must be equal to V_{DDA}.

On packages with 64 pins or less (except BGA package)

V_{REF+} and V_{REF-} pins are not available. They are internally connected to the ADC voltage supply (V_{DDA}) and ground (V_{SSA}).

4.1.2 Independent LCD supply

The V_{LCD} pin is provided to control the contrast of the glass LCD. This pin can be used in two ways:

- It can receive from an external circuitry the desired maximum voltage that is provided on segment and common lines to the glass LCD by the microcontroller.
- It can also be used to connect an external capacitor that is used by the microcontroller for its voltage step-up converter. This step-up converter is controlled by software to provide the desired voltage to segment and common lines of the glass LCD.

The voltage provided to segment and common lines defines the contrast of the glass LCD pixels. This contrast can be reduced when you configure the dead time between frames.

- When an external power supply is provided to the V_{LCD} pin, it should range from 2.5 V to 3.6 V. It does not depend on V_{DD}.
- When the LCD is based on the internal step-up converter, the V_{LCD} pin should be connected to a capacitor (see the product datasheets for further information).

4.1.3 RTC and RTC backup registers

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar, two programmable alarm interrupts, and a periodic programmable wakeup flag with interrupt capability. The RTC contains 20 backup data registers (80 bytes) in low and medium density devices, 32 backup data registers (128 bytes) in high and medium+ density devices and 5 backup data registers (20 bytes) for value line devices. These backup registers are reset when a tamper detection event occurs. For more details refer to *Real-time clock (RTC)* section.

RTC registers access

After reset, the RTC Registers (RTC registers and RTC backup registers) are protected against possible stray write accesses. To enable access to the RTC Registers, proceed as follows:

1. Enable the power interface clock by setting the PWREN bits in the RCC_APB1ENR register.
2. Set the DBP bit in the PWR_CR register (see *Section 4.4.1*).
3. Select the RTC clock source through RTCSEL[1:0] bits in RCC_CSR register.
4. Enable the RTC clock by programming the RTCEN bit in the RCC_CSR register.

4.1.4 Voltage regulator

An embedded linear voltage regulator supplies all the digital circuitries except for the Standby circuitry. The regulator output voltage (V_{CORE}) can be programmed by software to three different ranges within 1.2 - 1.8 V (typical) (see *Section 4.1.5*).

The voltage regulator is always enabled after Reset. It works in three different modes: main (MR), low power (LPR) and power down, depending on the application modes.

- In Run mode, the regulator is main (MR) mode and supplies full power to the V_{CORE} domain (core, memories and digital peripherals).
- In Low power run mode, the regulator is in low power (LPR) mode and supplies low power to the V_{CORE} domain, preserving the contents of the registers and internal SRAM.
- In Sleep mode, the regulator is main (MR) mode and supplies full power to the V_{CORE} domain, preserving the contents of the registers and internal SRAM.
- In low power sleep mode, the regulator is in low power (LPR) mode and supplies low power to the V_{CORE} domain, preserving the contents of the registers and internal SRAM.
- In Stop mode the regulator supplies low power to the V_{CORE} domain, preserving the content of registers and internal SRAM.
- In Standby mode, the regulator is powered off. The content of the registers and SRAM are lost except for the Standby circuitry.

4.1.5 Dynamic voltage scaling management

The dynamic voltage scaling is a power management technique which consists in increasing or decreasing the voltage used for the digital peripherals (V_{CORE}), according to the circumstances.

Dynamic voltage scaling to increase V_{CORE} is known as overvolting. It allows to improve the device performance. Refer to *Figure 6* for a description of the STM32L1xxxx operating conditions versus performance.

Dynamic voltage scaling to decrease V_{CORE} is known as undervolting. It is performed to save power, particularly in laptops and other mobile devices where the energy comes from a battery and is thus limited.

Range 1

Range 1 is the “high performance” range.

The voltage regulator outputs a 1.8 V voltage (typical) as long as the V_{DD} input voltage is above 2.0 V. Flash program and erase operations can be performed in this range.

Range 2 and 3

The regulator can also be programmed to output a regulated 1.5 V (typical, range 2) or a 1.2 V (typical, range 3) without any limitations on V_{DD} (1.65 to 3.6 V).

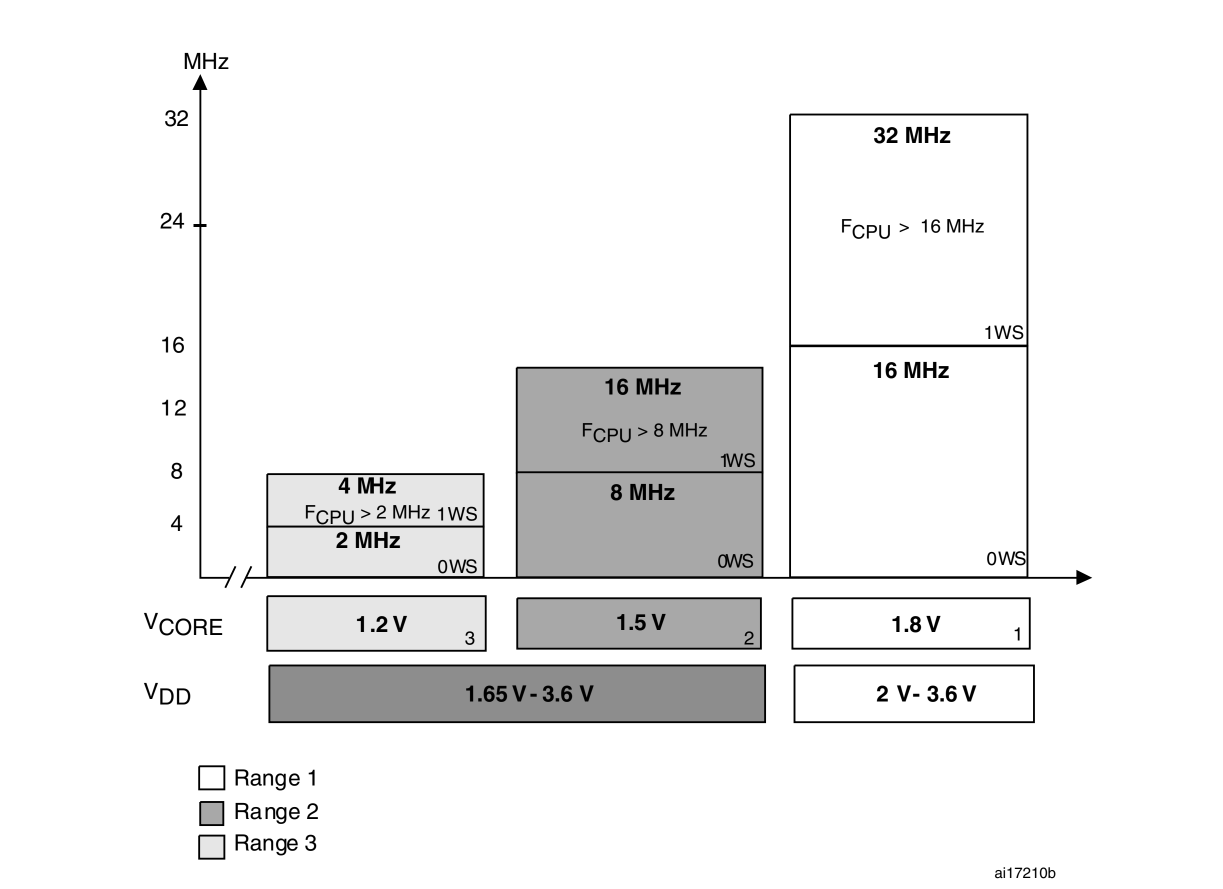
- At 1.5 V, the Flash memory is still functional but with medium read access time. This is the “medium performance” range. Program and erase operations on the Flash memory are still possible.
- At 1.2 V, the Flash memory is still functional but with slow read access time. This is the “low performance” range. Program and erase operations on the Flash memory are not possible under these conditions.

Refer to *Table 12* for details on the performance for each range.

Table 12. Performance versus V_{CORE} ranges

CPU performance	Power performance	V_{CORE} range	Typical Value (V)	Max frequency (MHz)		V_{DD} range
				1 WS	0 WS	
High	Low	1	1.8	32	16	2.0 - 3.6
Medium	Medium	2	1.5	16	8	1.65 - 3.6
Low	High	3	1.2	4	2	

Figure 6. STM32L1xxxx performance versus V_{DD} and V_{CORE} range



4.1.6 Dynamic voltage scaling configuration

The following sequence is required to program the voltage regulator ranges:

1. Check V_{DD} to identify which ranges are allowed (see *Figure 6: STM32L1xxxx performance versus VDD and VCORE range*).
2. Poll VOSF bit of in PWR_CSR. Wait until it is reset to 0.
3. Configure the voltage scaling range by setting the VOS[12:11] bits in the PWR_CR register.
4. Poll VOSF bit of in PWR_CSR register. Wait until it is reset to 0.

Note: During voltage scaling configuration, the system clock is stopped until the regulator is stabilized (VOSF=0). This must be taken into account during application development, in case a critical reaction time to interrupt is needed, and depending on peripheral used (timer, communication,...).

4.1.7 Voltage regulator and clock management when V_{DD} drops below 2.0 V

When V_{CORE} range 1 is selected and V_{DD} drops below 2.0 V, the application must reconfigure the system.

A three-step sequence is required to reconfigure the system:

1. Detect that V_{DD} drops below 2.0 V:

   Use the PVD to monitor the V_{DD} voltage and to generate an interrupt when the voltage goes under the selected level. To detect the 2.0 V voltage limit, the application can select by software PVD threshold 2 (2.26 V typical). For more details on the PVD, refer to *Section 4.2.3*.
2. Adapt the clock frequency to the voltage range that will be selected at next step:

   Below 2.0 V, the system clock frequency is limited to 16 MHz for range 2 and 4.2 MHz for range 3.
3. Select the required voltage range:

   Note that when V_{DD} is below 2.0 V, only range 2 or range 3 can be selected.

Note: When V_{CORE} range 2 or range 3 is selected and V_{DD} drops below 2.0 V, no system reconfiguration is required.

4.1.8 Voltage regulator and clock management when modifying the V_{CORE} range

When V_{DD} is above 2.0 V, any of the 3 voltage ranges can be selected:

- When the voltage range is above the targeted voltage range (e.g. from range 1 to 2):
  a) Adapt the clock frequency to the lower voltage range that will be selected at next step.
  b) Select the required voltage range.
- When the voltage range is below the targeted voltage range (e.g. from range 3 to 1):
  a) Select the required voltage range.
  b) Tune the clock frequency if needed.

When V_{DD} is below 2.0 V, only range 2 and 3 can be selected:

- From range 2 to range 3
  a) Adapt the clock frequency to voltage range 3.
  b) Select voltage range 3.
- From range 3 to range 2
  a) Select the voltage range 2.
  b) Tune the clock frequency if needed.

4.2 Power supply supervisor

The device has an integrated zeropower power on reset (POR)/power down reset (PDR), coupled with a brown out reset (BOR) circuitry. For devices operating between 1.8 and 3.6 V, the BOR is always active at power-on and ensures proper operation starting from 1.8 V. After the 1.8 V BOR threshold is reached, the option byte loading process starts, either to confirm or modify default thresholds, or to disable BOR permanently (in which case, the V_{DD}

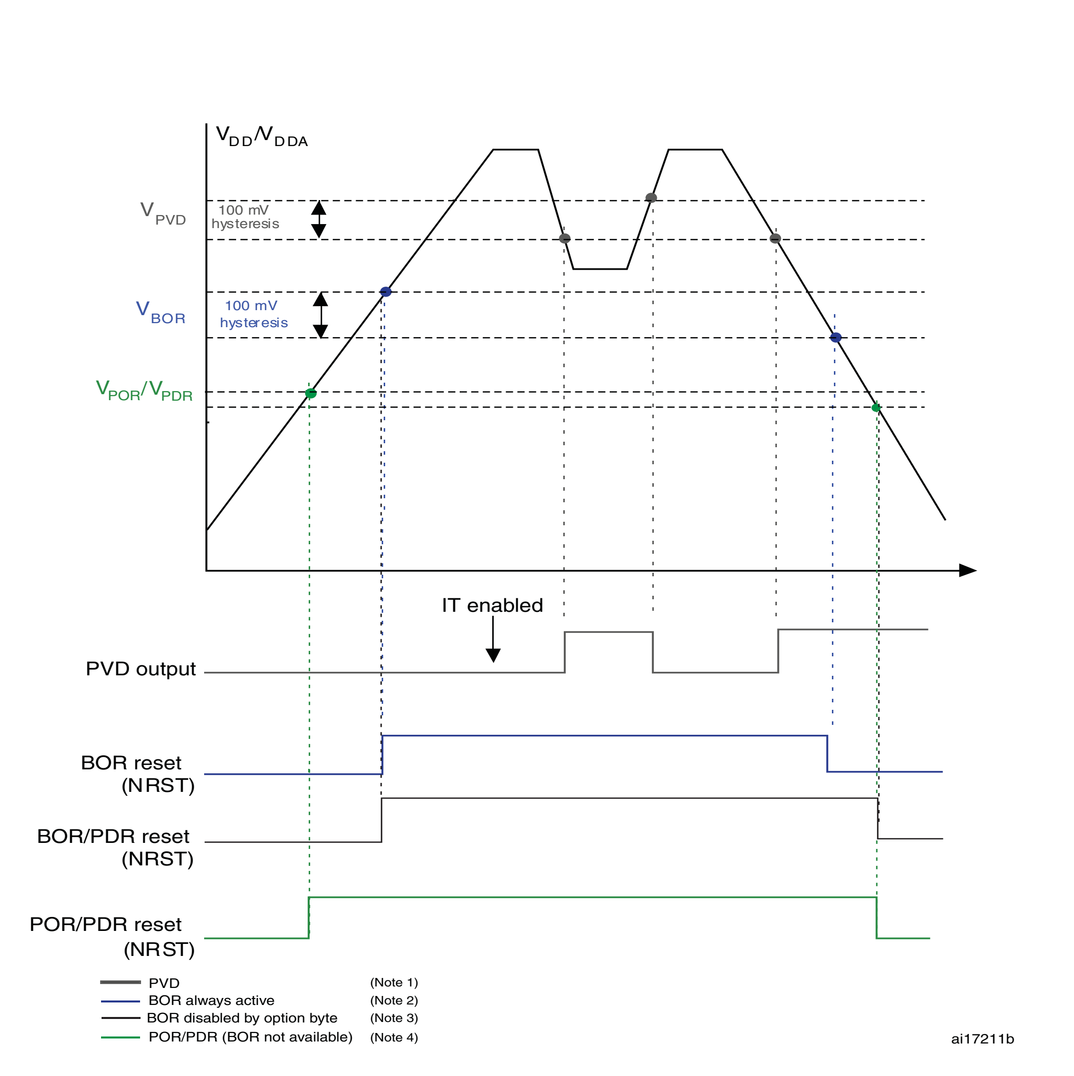
min value at power down is 1.65 V). For devices operating between 1.65 V and 3.6 V, the BOR is permanently disabled. Consequently, the start-up time at power-on can be decreased down to 1 ms typically.

Five BOR thresholds can be configured by option bytes, starting from 1.65 to 3 V. To reduce the power consumption in Stop mode, the internal voltage reference, V_{REFINT}, can be automatically switch off. The device remains in reset mode when V_{DD} is below a specified threshold, V_{POR}, V_{PDR} or V_{BOR}, without the need for any external reset circuit.

The device features an embedded programmable voltage detector (PVD) that monitors the V_{DD}/V_{DDA} power supply and compares it to the V_{PVD} threshold. 7 different PVD levels can be selected by software between 1.85 and 3.05 V, with a 200 mV step. An interrupt can be generated when V_{DD}/V_{DDA} drops below the V_{PVD} threshold and/or when V_{DD}/V_{DDA} is higher than the V_{PVD} threshold. The interrupt service routine then generates a warning message and/or put the MCU into a safe state. The PVD is enabled by software.

The different power supply supervisor (POR, PDR, BOR, PVD) are illustrated in *Figure 7*.

Figure 7. Power supply supervisors



1. The PVD is available on all STM32L devices and it is enabled or disabled by software.
2. The BOR is available only on devices operating from 1.8 to 3.6 V, and unless disabled by option byte it will mask the POR/PDR threshold.
3. When the BOR is disabled by option byte, the reset is asserted when V_{DD} goes below PDR level
4. For devices operating from 1.65 to 3.6 V, there is no BOR and the reset is released when V_{DD} goes above POR level and asserted when V_{DD} goes below PDR level

4.2.1 Power on reset (POR)/power down reset (PDR)

The device has an integrated POR/PDR circuitry that allows operation down to 1.5 V.

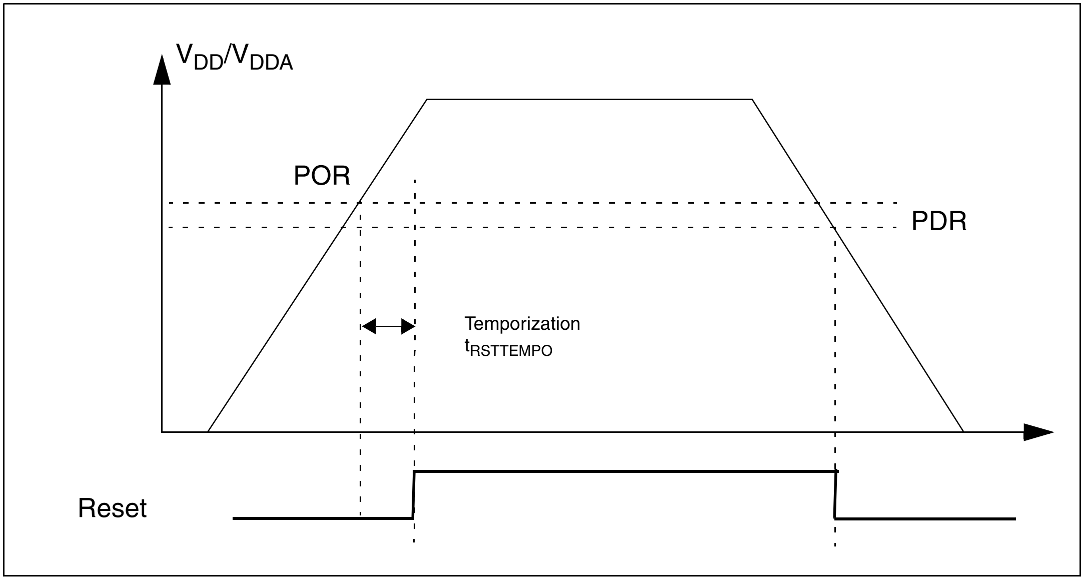
During power on, the device remains in Reset mode when V_{DD}/V_{DDA} is below a specified threshold, V_{POR}, without the need for an external reset circuit. The POR feature is always enabled and the POR threshold is 1.5 V.

During power down, the PDR keeps the device under reset when the supply voltage (V_{DD}) drops below the V_{PDR} threshold. The PDR feature is always enabled and the PDR threshold is 1.5 V.

The POR and PDR are used only when the BOR is disabled (see *Section 4.2.2: Brown out reset (BOR))*). To insure the minimum operating voltage (1.65 V), the BOR should be configured to BOR Level 0. When the BOR is disabled, a “grey zone” exist between the minimum operating voltage (1.65 V) and the V_{POR}/V_{PDR} threshold. This means that V_{DD} can be lower than 1.65 V without device reset until the V_{PDR} threshold is reached.

For more details concerning the power on/power down reset threshold, refer to the electrical characteristics of the datasheet.

Figure 8. Power on reset/power down reset waveform



4.2.2 Brown out reset (BOR)

During power on, the Brown out reset (BOR) keeps the device under reset until the supply voltage reaches the specified V_{BOR} threshold.

For devices operating from 1.65 to 3.6 V, the BOR option is not available and the power supply is monitored by the POR/PDR. As the POR/PDR thresholds are at 1.5 V, a "grey zone" exists between the V_{POR}/V_{PDR} thresholds and the minimum product operating voltage 1.65 V.

For devices operating from 1.8 to 3.6 V, the BOR is always active at power on and it's threshold is 1.8 V.

Then when the system reset is released, the BOR level can be reconfigured or disabled by option byte loading.

If the BOR level is kept at the lowest level, 1.8 V at power on and 1.65 V at power down, the system reset is fully managed by the BOR and the product operating voltages are within safe ranges.

And when the BOR option is disabled by option byte, the power down reset is controlled by the PDR and a "grey zone" exists between the 1.65 V and V_{PDR}.

V_{BOR} is configured through device option bytes. By default, the Level 4 threshold is activated. 5 programmable V_{BOR} thresholds can be selected.

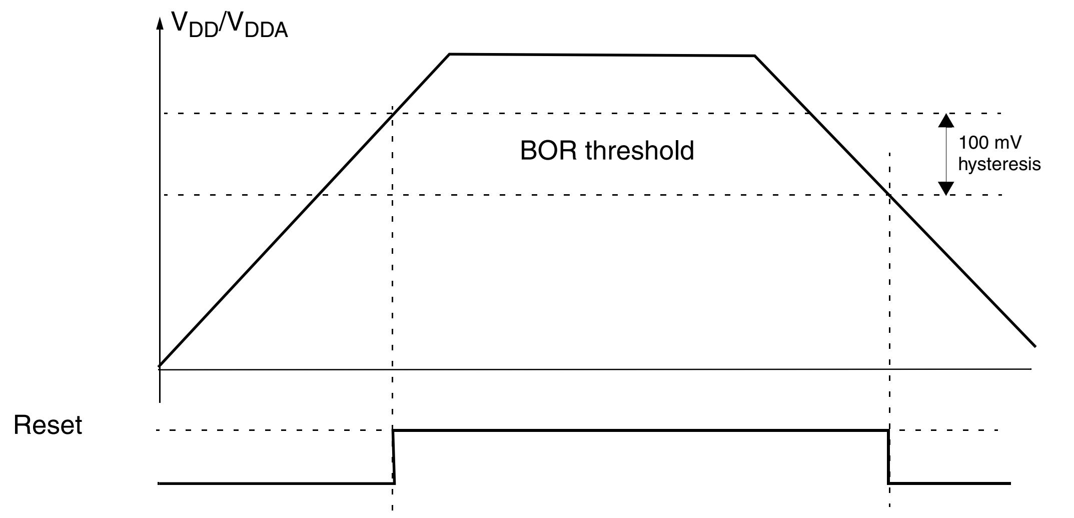
- BOR Level 0 (V_{BOR0}): reset threshold level for 1.69 to 1.80 V voltage range
- BOR Level 1 (V_{BOR1}): reset threshold level for 1.94 to 2.1 V voltage range
- BOR Level 2 (V_{BOR2}): reset threshold level for 2.3 to 2.49 V voltage range
- BOR Level 3 (V_{BOR3}): reset threshold level for 2.54 to 2.74 V voltage range
- BOR Level 4 (V_{BOR4}): reset threshold level for 2.77 to 3.0 V voltage range

When the supply voltage (V_{DD}) drops below the selected V_{BOR} threshold, a device reset is generated. When the V_{DD} is above the V_{BOR} upper limit the device reset is released and the system can start.

BOR can be disabled by programming the device option bytes. To disable the BOR function, V_{DD} must have been higher than V_{BOR0} to start the device option byte programming sequence. The power on and power down is then monitored by the POR and PDR (see *Section 4.2.1: Power on reset (POR)/power down reset (PDR)*)

The BOR threshold hysteresis is ~100 mV (between the rising and the falling edge of the supply voltage).

Figure 9. BOR thresholds



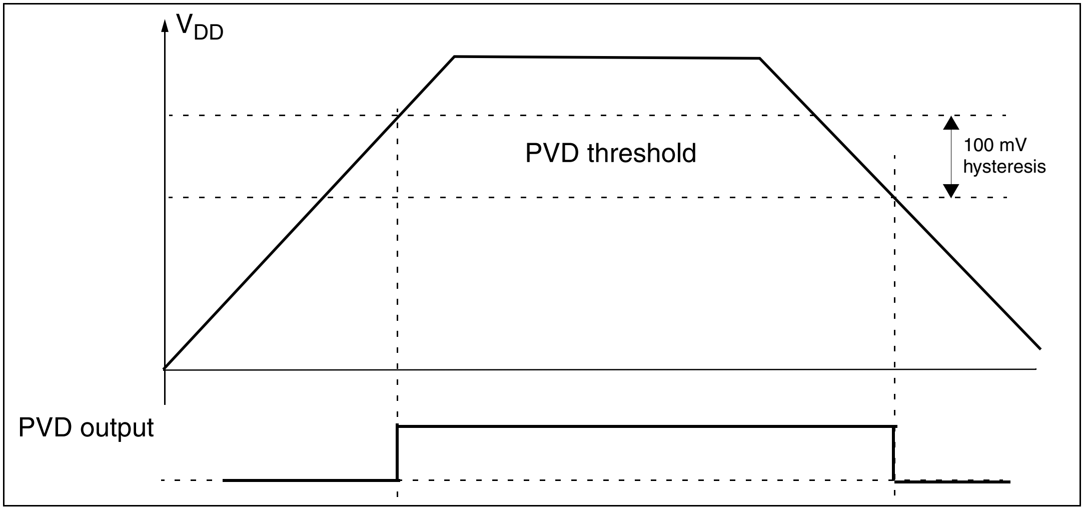
4.2.3 Programmable voltage detector (PVD)

You can use the PVD to monitor the V_{DD} power supply by comparing it to a threshold selected by the PLS[2:0] bits in the PWR_CR (see *Section 4.4.1*).

The PVD can use an external input analog voltage (PVD_IN) which is compared internally to VREFINT. The PVD_IN (PB7) has to be configured in Analog mode when PLS[2:0] = 111. The PVD is enabled by setting the PVDE bit.

A PVDO flag is available, in the PWR_CSR (see *Section 4.4.2*), to indicate if V_{DD} is higher or lower than the PVD threshold. This event is internally connected to the EXTI line16 and can generate an interrupt if enabled through the EXTI registers. The PVD output interrupt can be generated when V_{DD} drops below the PVD threshold and/or when V_{DD} rises above the PVD threshold depending on EXTI line16 rising/falling edge configuration. As an example the service routine could perform emergency shutdown tasks.

Figure 10. PVD thresholds



4.2.4 Internal voltage reference (V_{REFINT})

The functions managed through the internal voltage reference (V_{REFINT}) are BOR, PVD, ADC, LCD and comparators. The internal voltage reference (V_{REFINT}) is always enabled.

The internal voltage reference consumption is not negligible, in particular in Stop and Standby mode. To reduce power consumption, the ULP bit (Ultra low power) in the PWR_CR register can be set to disable the internal voltage reference. However, in this case, when exiting from the Stop/Standby mode, the functions managed through the internal voltage reference are not reliable during the internal voltage reference startup time (up to 3 ms).

To reduce the wakeup time, the device can exit from Stop/Standby mode without waiting for the internal voltage reference startup time. This is performed by setting the FWU bit (Fast wakeup) in the PWR_CR register before entering Stop/Standby mode.

If the ULP bit is set, the functions that were enabled before entering the Stop/Standby mode will be disabled during these modes, and enabled again only after the end of the internal voltage reference startup time whatever FWU value. The VREFINTRDYF flag in the PWR_CSR register indicates that the internal voltage reference is ready.

4.3 Low-power modes

By default, the microcontroller is in Run mode after a system or a power on reset. In Run mode the CPU is clocked by HCLK and the program code is executed. Several low-power modes are available to save power when the CPU does not need to be kept running, for example when waiting for an external event. It is up to the user to select the mode that gives the best compromise between low-power consumption, performance, short startup time and available wakeup sources.

The devices feature five low-power modes:

- Low power run mode: regulator in low power mode, limited clock frequency, limited number of peripherals running
- Sleep mode: Cortex-M3 core stopped, peripherals kept running
- Low power sleep mode: Cortex-M3 core stopped, limited clock frequency, limited number of peripherals running, regulator in low power mode, RAM in power down, Flash stopped.
- Stop mode (all clocks are stopped, regulator running, regulator in low power mode
- Standby mode: V_{CORE} domain powered off

In addition, the power consumption in Run mode can be reduced by one of the following means:

- Slowing down the system clocks
- Gating the clocks to the APBx and AHBx peripherals when they are unused.

Table 13. Summary of low-power modes

Mode name	Entry	Wakeup	Effect on V_{CORE} domain clocks	Effect on V_{DD} domain clocks	Voltage regulator
Low power run	LPSDSR and LPRUN bits + Clock setting	The regulator is forced in Main regulator (1.8 V)	None	None	In low power mode
Sleep (Sleep now or Sleep-on-exit)	WFI	Any interrupt	CPU CLK OFF no effect on other clocks or analog clock sources	None	ON
	WFE	Wakeup event			
Low power sleep (Sleep now or Sleep-on-exit)	LPSDSR bits + WFI	Any interrupt	CPU CLK OFF no effect on other clocks or analog clock sources, Flash CLK OFF	None	In low power mode
	LPSDSR bits + WFE	Wakeup event			
Stop	PDDS, LPSDSR bits + SLEEPDEEP bit + WFI or WFE	Any EXTI line (configured in the EXTI registers, internal and external lines)	All V_{CORE} domain clocks OFF	HSI and HSE and MSI oscillators OFF	ON, in low power mode (depending on PWR_CR)
Standby	PDDS bit + SLEEPDEEP bit + WFI or WFE	WKUP pin rising edge, RTC alarm (Alarm A or Alarm B), RTC Wakeup event, RTC tamper event, RTC timestamp event, external reset in NRST pin, IWDG reset			OFF

4.3.1 Behavior of clocks in low power modes

APB peripheral and DMA clocks can be disabled by software.

Sleep and Low power sleep modes

The CPU clock is stopped in Sleep and Low power sleep mode. The memory interface clocks (FLITF and RAM interfaces) and all peripherals clocks can be stopped by software during Sleep. The memory interface (FLITF) clock is stopped and the RAM is in power-down when in Low power sleep mode. The AHB to APB bridge clocks are disabled by hardware during Sleep/Low power sleep mode when all the clocks of the peripherals connected to them are disabled.

Stop and Standby modes

The system clock and all high speed clocks are stopped in Stop and Standby modes:

- PLL is disabled
- Internal RC 16 MHz (HSI) oscillator is disabled
- External 1-24 MHz (HSE) oscillator is disabled
- Internal 65 kHz - 4.2 MHz (MSI) oscillator is disabled

When exiting this mode by interrupt (Stop mode) or by reset (Standby mode), the internal MSI oscillator is selected as system clock. When the device exits Stop mode, the previous MSI configuration (range and trimming value) is kept. When exiting Standby mode, the range and trimming value are reset to the default 2 MHz values.

If a Flash program operation or an access to APB domain is ongoing, the Stop/Standby mode entry is delayed until the Flash memory or the APB access has completed.

4.3.2 Slowing down system clocks

In Run mode the speed of the system clocks (SYSCLK, HCLK, PCLK1, PCLK2) can be reduced by programming the prescaler registers. These prescalers can also be used to slow down peripherals before entering Sleep mode.

For more details refer to *Section 5.3.3: Clock configuration register (RCC_CFGR)*.

4.3.3 Peripheral clock gating

In Run mode, the HCLK and PCLKx for individual peripherals and memories can be stopped at any time to reduce power consumption.

To further reduce power consumption in Sleep mode the peripheral clocks can be disabled prior to executing the WFI or WFE instructions.

Peripheral clock gating is controlled by the AHB peripheral clock enable register (RCC_AHBENR), APB2 peripheral clock enable register (RCC_APB2ENR), APB1 peripheral clock enable register (RCC_APB1ENR) (see *Section 5.3.8: AHB peripheral clock enable register (RCC_AHBENR)*, *Section 5.3.10: APB1 peripheral clock enable register (RCC_APB1ENR)* and *Section 5.3.9: APB2 peripheral clock enable register (RCC_APB2ENR)*).

Disabling the peripherals clocks in Sleep mode can be performed automatically by resetting the corresponding bit in RCC_AHBLPENR and RCC_APBxLPENR registers (x can 1 or 2).

4.3.4 Low power run mode (LP run)

To further reduce the consumption when the system is in Run mode, the regulator can be configured in low power mode. In this mode, the system frequency should not exceed f_MSI range1.

Please refer to the product datasheet for more details on voltage regulator and peripherals operating conditions.

Note: *To be able to read the RTC calendar register when the APB1 clock frequency is less than seven times the RTC clock frequency (7*RTCLCK), 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.

Low power run mode can only be entered when V_{CORE} is in range 2. In addition, the dynamic voltage scaling must not be used when Low power run mode is selected. Only Stop and Sleep modes with regulator configured in Low power mode is allowed when Low power run mode is selected.

Note: *In Low power run mode, all I/O pins keep the same state as in Run mode.*

Entering Low power run mode

To enter Low power run mode proceed as follows:

- Each digital IP clock must be enabled or disabled by using the RCC_APBxENR and RCC_AHBENR registers.
- The frequency of the system clock must be decreased to not exceed the frequency of f_MSI range1.
- The regulator is forced in low power mode by software (LPRUN and LPSDSR bits set)

Exiting Low power run mode

To exit Low power run mode proceed as follows:

- The regulator is forced in Main regulator mode by software.
- The Flash memory is switched on, if needed.
- The frequency of the clock system can be increased.

4.3.5 Sleep mode

Entering Sleep mode

The Sleep mode is entered by executing the WFI (Wait For Interrupt) or WFE (Wait for Event) instructions. Two options are available to select the Sleep mode entry mechanism, depending on the SLEEPONEXIT bit in the Cortex-M3 System Control register:

- Sleep-now: if the SLEEPONEXIT bit is cleared, the MCU enters Sleep mode as soon as WFI or WFE instruction is executed.
- Sleep-on-exit: if the SLEEPONEXIT bit is set, the MCU enters Sleep mode as soon as it exits the lowest priority ISR.

Note: *In Sleep mode, all I/O pins keep the same state as in Run mode.*

Refer to *Table 14: Sleep-now* and *Table 15: Sleep-on-exit* for details on how to enter Sleep mode.

Exiting Sleep mode

If the WFI instruction is used to enter Sleep mode, any peripheral interrupt acknowledged by the nested vectored interrupt controller (NVIC) can wake up the device from Sleep mode.

If the WFE instruction is used to enter Sleep mode, the MCU exits Sleep mode as soon as an event occurs. The wakeup event can be generated either by:

- Enabling an interrupt in the peripheral control register but not in the NVIC, and enabling the SEVONPEND bit in the Cortex-M3 System Control register. When the MCU resumes from WFE, the peripheral interrupt pending bit and the peripheral NVIC IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.
- Or configuring an external or internal EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bit corresponding to the event line is not set.

This mode offers the lowest wakeup time as no time is wasted in interrupt entry/exit.

Refer to *Table 14: Sleep-now* and *Table 15: Sleep-on-exit* for more details on how to exit Sleep mode.

Table 14. Sleep-now

Sleep-now mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 0 Refer to the Cortex-M3 System Control register.
Mode exit	If WFI was used for entry: Interrupt: Refer to *Table 35: Vector table (low and medium density devices)* If WFE was used for entry Wakeup event: Refer to *Section 9.2.3: Wakeup event management*
Wakeup latency	None

Table 15. Sleep-on-exit

Sleep-on-exit	Description
Mode entry	WFI (wait for interrupt) while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 Refer to the Cortex-M3 System Control register.
Mode exit	Interrupt: refer to *Table 35: Vector table (low and medium density devices)*, *Table 36: Vector table (medium+ density devices)* and *Table 37: Vector table (high-density devices)*.
Wakeup latency	None

4.3.6 Low power sleep mode (LP sleep)

Entering Low power sleep mode

The Low power sleep mode is entered by configuring the voltage regulator in low power mode, and by executing the WFI (wait for interrupt) or WFE (wait for event) instructions. In this mode, the Flash memory is not available but the RAM memory remains available.

In this mode, the system frequency should not exceed f_MSI range1.

Please refer to product datasheet for more details on voltage regulator and peripherals operating conditions.

Low power sleep mode can only be entered when V_{CORE} is in range 2.

*Note: To be able to read the RTC calendar register when the APB1 clock frequency is less than seven times the RTC clock frequency (7*RTCLCK), 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.

Two options are available to select the Sleep low power mode entry mechanism, depending on the SLEEPONEXIT bit in the Cortex-M3 System Control register:

- Sleep-now: if the SLEEPONEXIT bit is cleared, the MCU enters Sleep mode as soon as WFI or WFE instruction is executed.
- Sleep-on-exit: if the SLEEPONEXIT bit is set, the MCU enters Sleep mode as soon as it exits the lowest priority ISR.

To enter Low power sleep mode, proceed as follows:

- The Flash memory can be switched off by using the control bits (SLEEP_PD in the FLASH_ACR register. For more details refer to PM0062). This reduces power consumption but increases the wake-up time.
- Each digital IP clock must be enabled or disabled by using the RCC_APBxENR and RCC_AHBENR registers.
- The frequency of the system clock must be decreased.
- The regulator is forced in low power mode by software (LPSDSR bits set).
- A WFI/WFE instruction must be executed to enter in Sleep mode.

Note: In Low power sleep mode, all I/O pins keep the same state as in Run mode.

Refer to *Table 16: Sleep-now* and *Table 17: Sleep-on-exit* for details on how to enter Low power sleep mode.

Exiting Low power sleep mode

If the WFI instruction was used to enter Low power sleep mode, any peripheral interrupt acknowledged by the nested vectored interrupt controller (NVIC) can wake up the device from Low power sleep mode.

If the WFE instruction was used to enter Low power sleep mode, the MCU exits Sleep mode as soon as an event occurs. The wakeup event can be generated:

- By enabling an interrupt in the peripheral control register but not in the NVIC, and by enabling the SEVONPEND bit in the Cortex-M3 System Control register. When the

MCU resumes from WFE, the peripheral interrupt pending bit and the peripheral NVIC IRQ channel pending bit in the NVIC interrupt clear pending register must be cleared.

- Or by configuring an external or internal EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bit corresponding to the event line is not set.

When exiting Low power sleep mode by issuing an interrupt or a wakeup event, the regulator is configured in Main regulator mode, the Flash memory is switched on (if necessary), and the system clock can be increased.

When the voltage regulator operates in low power mode, an additional startup delay is incurred when waking up from Low power sleep mode.

Refer to *Table 16: Sleep-now* and *Table 17: Sleep-on-exit* for more details on how to exit Sleep low power mode.

Table 16. Sleep-now

Sleep-now mode	Description
Mode entry	Voltage regulator in low power mode and the Flash memory switched off WFI (Wait for Interrupt) or WFE (wait for event) while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 0 Refer to the Cortex™-M3 System Control register.
Mode exit	Voltage regulator in Main regulator mode and the Flash memory switched on If WFI was used for entry: Interrupt: Refer to *Table 35: Vector table (low and medium density devices)* If WFE was used for entry Wakeup event: Refer to *Section 9.2.3: Wakeup event management*
Wakeup latency	Regulator wakeup time from low power mode

Table 17. Sleep-on-exit

Sleep-on-exit	Description
Mode entry	Voltage regulator in low power mode and the Flash memory switched off WFI (wait for interrupt) while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 Refer to the Cortex™-M3 System Control register.
Mode exit	Interrupt: refer to *Table 35: Vector table (low and medium density devices)*, *Table 36: Vector table (medium+ density devices)* and *Table 37: Vector table (high-density devices)*.
Wakeup latency	regulator wakeup time from low power mode

4.3.7 Stop mode

The Stop mode is based on the Cortex™-M3 deepsleep mode combined with peripheral clock gating. The voltage regulator can be configured either in normal or low-power mode.

In Stop mode, all clocks in the V_{CORE} domain are stopped, the PLL, the MSI, the HSI and the HSE RC oscillators are disabled. Internal SRAM and register contents are preserved.

To get the lowest consumption in Stop mode, the internal Flash memory also enters low power mode. When the Flash memory is in power down mode, an additional startup delay is incurred when waking up from Stop mode.

To minimize the consumption In Stop mode, V_{REFINT}, the BOR, PVD, and temperature sensor can be switched off before entering the Stop mode. They can be switched on again by software after exiting the Stop mode using the ULP bit in the PWR_CR register.

Note: In Stop mode, all I/O pins keep the same state as in Run mode.

Entering the Stop mode

Refer to *Table 18* for details on how to enter the Stop mode.

To further reduce power consumption in Stop mode, the internal voltage regulator can be put in low power mode. This is configured by the LPSDSR bit in the PWR_CR register (see *Section 4.4.1*).

If Flash memory programming or an access to the APB domain is ongoing, the Stop mode entry is delayed until the memory or APB access has completed.

In Stop mode, the following features can be selected by programming individual control bits:

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by a Reset. Refer to *Section 20.3* in *Section 20: Independent watchdog (IWDG)*.
- Real-time clock (RTC): this is configured by the RTCEN bit in the RCC_CSR register (see *Section 5.3.14*).
- Internal RC oscillator (LSI RC): this is configured by the LSION bit in the RCC_CSR register.
- External 32.768 kHz oscillator (LSE OSC): this is configured by the LSEON bit in the RCC_CSR register.

The ADC, DAC or LCD can also consume power in Stop mode, unless they are disabled before entering it. To disable them, the ADON bit in the ADC_CR2 register and the ENx bit in the DAC_CR register must both be written to 0.

Exiting the Stop mode

Refer to *Table 18* for more details on how to exit Stop mode.

When exiting Stop mode by issuing an interrupt or a wakeup event, the MSI RC oscillator is selected as system clock.

When the voltage regulator operates in low power mode, an additional startup delay is incurred when waking up from Stop mode. By keeping the internal regulator ON during Stop mode, the consumption is higher although the startup time is reduced.

Table 18. Stop mode

Stop mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – Set SLEEPDEEP bit in Cortex™-M3 System Control register – Clear PDDS bit in Power Control register (PWR_CR) – Clear WUF bit in Power Control/Status register (PWR_CSR) – Select the voltage regulator mode by configuring LPSDSR bit in PWR_CR **Note:** To enter the Stop mode, all EXTI Line pending bits (in *EXTI pending register (EXTI_PR)*), the RTC Alarm (Alarm A and Alarm B), RTC wakeup, RTC tamper, and RTC time-stamp flags, must be reset. Otherwise, the Stop mode entry procedure is ignored and program execution continues.
Mode exit	If WFI was used for entry: Any EXTI Line configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). Refer to *Table 35: Vector table (low and medium density devices), Table 36: Vector table (medium+ density devices)* and *Table 37: Vector table (high-density devices)*. If WFE was used for entry: Any EXTI Line configured in event mode. Refer to *Section 9.2.3: Wakeup event management on page 197*
Wakeup latency	MSI RC wakeup time + regulator wakeup time from Low-power mode + FLASH wakeup time

4.3.8 Standby mode

The Standby mode allows to achieve the lowest power consumption. It is based on the Cortex™-M3 deepsleep mode, with the voltage regulator disabled. The V_{CORE} domain is consequently powered off. The PLL, the MSI, the HSI oscillator and the HSE oscillator are also switched off. SRAM and register contents are lost except for the RTC registers, RTC backup registers and Standby circuitry (see *Figure 5*).

Entering the Standby mode

Refer to *Table 19* for more details on how to enter Standby mode.

In Standby mode, the following features can be selected by programming individual control bits:

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by a reset. Refer to *Section 20.3: IWDG functional description on page 506*.
- Real-time clock (RTC): this is configured by the RTCEN bit in the RCC_CSR register (see *Section 5.3.14*).
- Internal RC oscillator (LSI RC): this is configured by the LSION bit in the RCC_CSR register.
- External 32.768 kHz oscillator (LSE OSC): this is configured by the LSEON bit in the RCC_CSR register.

Exiting the Standby mode

The microcontroller exits Standby mode when an external Reset (NRST pin), an IWDG Reset, a rising edge on WKUP pins (WUKP1, WKUP2 or WKUP3), an RTC alarm, a tamper event, or a time-stamp event is detected. All registers are reset after wakeup from Standby except for *PWR power control/status register (PWR_CSR)*.

After waking up from Standby mode, program execution restarts in the same way as after a Reset (boot pins sampling, vector reset is fetched, etc.). The SBF status flag in the PWR_CSR register (see *Section 4.4.2*) indicates that the MCU was in Standby mode.

Refer to *Table 19* for more details on how to exit Standby mode.

Table 19. Standby mode

Standby mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – Set SLEEPDEEP in Cortex™-M3 System Control register – Set PDDS bit in Power Control register (PWR_CR) – Clear WUF bit in Power Control/Status register (PWR_CSR) – Clear the RTC flag corresponding to the chosen wakeup source (RTC Alarm A, RTC Alarm B, RTC wakeup, Tamper or Time-stamp flags)
Mode exit	WKUP pin rising edge, RTC alarm (Alarm A and Alarm B), RTC wakeup, tamper event, time-stamp event, external reset in NRST pin, IWDG reset.
Wakeup latency	Reset phase

I/O states in Standby mode

In Standby mode, all I/O pins are high impedance except for:

- Reset pad (still available)
- RTC_AF1 pin (PC13) if configured for Wakeup pin 2 (WKUP2), tamper, time-stamp, RTC Alarm out, or RTC clock calibration out.
- WKUP pin 1 (PA0) and WKUP pin 3 (PE6), if enabled.

Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop or Standby mode while the debug features are used. This is due to the fact that the Cortex™-M3 core is no longer clocked.

However, by setting some configuration bits in the DBGMCU_CR register, the software can be debugged even when using the low-power modes extensively. For more details, refer to *Section 29.16.1: Debug support for low-power modes*.

4.3.9 Waking up the device from Stop and Standby modes using the RTC and comparators

The MCU can be woken up from low-power mode by an RTC Alarm event, an RTC Wakeup event, a tamper event, a time-stamp event, or a comparator event, without depending on an external interrupt (Auto-wakeup mode).

These RTC alternate functions can wake up the system from Stop and Standby low power modes while the comparator events can only wake up the system from Stop mode.

The system can also wake up from low power modes without depending on an external interrupt (Auto-wakeup mode) by using the RTC alarm or the RTC wakeup events.

The RTC provides a programmable time base for waking up from Stop or Standby mode at regular intervals. For this purpose, two of the three alternative RTC clock sources can be selected by programming the RTCSEL[1:0] bits in the RCC_CSR register (see *Section 5.3.14*):

- Low-power 32.768 kHz external crystal oscillator (LSE OSC).
  This clock source provides a precise time base with very low-power consumption (less than 1 µA added consumption in typical conditions)
- Low-power internal RC oscillator (LSI RC)

  This clock source has the advantage of saving the cost of the 32.768 kHz crystal. This internal RC Oscillator is designed to use minimum power consumption.

RTC auto-wakeup (AWU) from the Stop mode

- To wake up from the Stop mode with an RTC alarm event, it is necessary to:
  a) Configure the EXTI Line 17 to be sensitive to rising edges (Interrupt or Event modes)
  b) Enable the RTC Alarm interrupt in the RTC_CR register
  c) Configure the RTC to generate the RTC alarm
- To wake up from the Stop mode with an RTC Tamper or time stamp event, it is necessary to:
  a) Configure the EXTI Line 19 to be sensitive to rising edges (Interrupt or Event modes)
  b) Enable the RTC TimeStamp Interrupt in the RTC_CR register or the RTC Tamper Interrupt in the RTC_TCR register
  c) Configure the RTC to detect the tamper or time stamp event
- To wake up from the Stop mode with an RTC Wakeup event, it is necessary to:
  a) Configure the EXTI Line 20 to be sensitive to rising edges (Interrupt or Event modes)
  b) Enable the RTC Wakeup Interrupt in the RTC_CR register
  c) Configure the RTC to generate the RTC Wakeup event

RTC auto-wakeup (AWU) from the Standby mode

- To wake up from the Standby mode with an RTC alarm event, it is necessary to:
  a) Enable the RTC Alarm interrupt in the RTC_CR register
  b) Configure the RTC to generate the RTC alarm
- To wake up from the Stop mode with an RTC Tamper or time stamp event, it is necessary to:
  a) Enable the RTC TimeStamp Interrupt in the RTC_CR register or the RTC Tamper Interrupt in the RTC_TCR register
  b) Configure the RTC to detect the tamper or time stamp event
- To wake up from the Stop mode with an RTC Wakeup event, it is necessary to:
  a) Enable the RTC Wakeup Interrupt in the RTC_CR register
  b) Configure the RTC to generate the RTC Wakeup event

Comparator auto-wakeup (AWU) from the Stop mode

- To wake up from the Stop mode with a comparator 1 or comparator 2 wakeup event, it is necessary to:
  a) Configure the EXTI Line 21 for comparator 1 or EXTI Line 22 for comparator 2 (Interrupt or Event mode) to be sensitive to the selected edges (falling, rising or falling and rising)
  b) Configure the comparator to generate the event

4.4 Power control registers

The peripheral registers have to be accessed by half-words (16-bit) or words (32-bit).

4.4.1 PWR power control register (PWR_CR)

Address offset: 0x00

Reset value: 0x0000 1000 (reset by wakeup from Standby mode)

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LPRUN	Res.	VOS[1:0]		FWU	ULP	DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPSDSR
	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rc_w1	rc_w1	rw	rw

Bits 31:15 Reserved, always read as 0.

Bit 14 **LPRUN:** Low power run mode

When LPRUN bit is set together with the LPSDSR bit, the regulator is switched from main mode to low power mode. Otherwise, it remains in main mode. The regulator goes back to operate in main mode when LPRUN is reset.

It is forbidden to reset LPSDSR when the MCU is in Low power run mode. LPSDSR is used as a prepositioning for the entry into low power mode, indicating to the system which configuration of the regulator will be selected when entering Low power mode. The LPSDSR bit must be set before the LPRUN bit is set. LPSDSR can be reset only when LPRUN bit=0.

0: Voltage regulator in main mode in Low power run mode

1: Voltage regulator in low power mode in Low power run mode

Bits 13 Reserved, must be kept at reset value.

Bits 12:11 **VOS[1:0]:** Voltage scaling range selection

These bits are used to select the internal regulator voltage range.
Before resetting the power interface by resetting the PWRRST bit in the RCC_APB1RSTR register, these bits have to be set to "10" and the frequency of the system has to be configured accordingly.

00: forbidden (range1 will be automatically selected)

01: 1.8 V (range 1)

10: 1.5 V (range 2)

11: 1.2 V (range 3)

Bit 10 **FWU**: Fast wakeup

This bit works in conjunction with ULP bit.

If ULP = 0, FWU is ignored

If ULP = 1 and FWU = 1: V_{REFINT} startup time is ignored when exiting from low power mode. The VREFINTRDYF flag in the PWR_CSR register indicates when the V_{REFINT} is ready again.

If ULP=1 and FWU = 0: Exiting from low power mode occurs only when the V_{REFINT} is ready (after its startup time). This bit is not reset by resetting the PWRRST bit in the RCC_APB1RSTR register.

0: Low power modes exit occurs only when $V_{REFTINT}$ is ready

1: $V_{REFTINT}$ start up time is ignored when exiting low power modes

Bit 9 **ULP**: Ultralow power mode

When set, the V_{REFINT} is switched off in low power mode. This bit is not reset by resetting the PWRRST bit in the RCC_APB1RSTR register.

0: $V_{REFTINT}$ is on in low power mode
1: $V_{REFTINT}$ is off in low power mode

Bit 8 **DBP**: Disable backup write protection

In reset state, the RTC, RTC backup registers and RCC CSR register are protected against parasitic write access. This bit must be set to enable write access to these registers.

0: Access to RTC, RTC Backup and RCC CSR registers disabled
1: Access to RTC, RTC Backup and RCC CSR registers enabled

Note: If the HSE divided by 2, 4, 8 or 16 is used as the RTC clock, this bit must remain set to 1.

Bits 7:5 **PLS[2:0]:** PVD level selection

These bits are written by software to select the voltage threshold detected by the power voltage detector:

000: 1.9 V
001: 2.1 V
010: 2.3 V
011: 2.5 V
100: 2.7 V
101: 2.9 V
110: 3.1 V
111: External input analog voltage (Compare internally to V_{REFINT})

PVD_IN input (PB7) has to be configured as analog input when PLS[2:0] = 111.

Note: Refer to the electrical characteristics of the datasheet for more details.

Bit 4 **PVDE:** Power voltage detector enable

This bit is set and cleared by software.

0: PVD disabled
1: PVD enabled

Bit 3 **CSBF**: Clear standby flag

This bit is always read as 0.

0: No effect
1: Clear the SBF Standby flag (write).

Bit 2 **CWUF:** Clear wakeup flag

This bit is always read as 0.

0: No effect

1: Clear the WUF Wakeup flag after 2 system clock cycles

Bit 1 **PDDS**: Power down deepsleep

This bit is set and cleared by software.

0: Enter Stop mode when the CPU enters deepsleep. The regulator is in low-power mode.

1: Enter Standby mode when the CPU enters deepsleep.

Bit 0 **LPSDSR:** Low-power deepsleep/sleep/low power run

– DeepSleep/Sleep modes

When this bit is set, the regulator switches in low power mode when the CPU enters sleep or deepsleep mode. The regulator goes back to main mode when the CPU exits from these modes.

– Low power run mode

When this bit is set, the regulator switches in low power mode when the bit LPRUN is set. The regulator goes back to main mode when the bit LPRUN is reset.

This bit is set and cleared by software.

0: Voltage regulator on during deepsleep/Sleep/Low power run mode

1: Voltage regulator in low power mode during deepsleep/Sleep/Low power run mode

4.4.2 PWR power control/status register (PWR_CSR)

Address offset: 0x04

Reset value: 0x0000 0008 (not reset by wakeup from Standby mode)

Additional APB cycles are needed to read this register versus a standard APB read.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					EWUP3	EWUP2	EWUP1	Reserved		REGLPF	VOSF	VREFINTRDYF	PVDO	SBF	WUF
					rw	rw	rw			r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **EWUP3**: Enable WKUP pin 3

This bit is set and cleared by software.

0: WKUP pin 3 is used for general purpose I/Os. An event on the WKUP pin 3 does not wakeup the device from Standby mode.

1: WKUP pin 3 is used for wakeup from Standby mode and forced in input pull down configuration (rising edge on WKUP pin 3 wakes-up the system from Standby mode).

Note: This bit is reset by a system reset.

Bit 9 **EWUP2**: Enable WKUP pin 2

This bit is set and cleared by software.

0: WKUP pin 2 is used for general purpose I/Os. An event on the WKUP pin 2 does not wakeup the device from Standby mode.

1: WKUP pin 2 is used for wakeup from Standby mode and forced in input pull down configuration (rising edge on WKUP pin 2 wakes-up the system from Standby mode).

Note: This bit is reset by a system reset.

Bit 8 **EWUP1**: Enable WKUP pin 1

This bit is set and cleared by software.

0: WKUP pin 1 is used for general purpose I/Os. An event on the WKUP pin 1 does not wakeup the device from Standby mode.

1: WKUP pin 1 is used for wakeup from Standby mode and forced in input pull down configuration (rising edge on WKUP pin 1 wakes-up the system from Standby mode).

Note: This bit is reset by a system reset.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **REGLPF**: Regulator LP flag

This bit is set by hardware when the MCU is in Low power run mode.

When the MCU exits from Low power run mode, this bit stays at 1 until the regulator is ready in main mode. A polling on this bit is recommended to wait for the regulator main mode. This bit is reset by hardware when the regulator is ready.

0: Regulator is ready in main mode

1: Regulator voltage is in low power mode

Bit 4 **VOSF**: Voltage Scaling select flag

A delay is required for the internal regulator to be ready after the voltage range is changed. The VOSF bit indicates that the regulator has reached the voltage level defined with bits VOS of PWR_CR register.

This bit is reset when VOS[1:0] in PWR_CR register change.
It is set once the regulator is ready.

0: Regulator is ready in the selected voltage range
1: Regulator voltage output is changing to the required VOS level.

Bit 3 **VREFINTRDYF**: Internal voltage reference (V_{REFINT}) ready flag

This bit indicates the state of the internal voltage reference, V_{REFINT}.

0: V_{REFINT} is OFF
1: V_{REFINT} is ready

Bit 2 **PVDO:** PVD output

This bit is set and cleared by hardware. It is valid only if PVD is enabled by the PVDE bit.

0: V_{DD} is higher than the PVD threshold selected with the PLS[2:0] bits.
1: V_{DD} is lower than the PVD threshold selected with the PLS[2:0] bits.

Note: The PVD is stopped by Standby mode. For this reason, this bit is equal to 0 after Standby or reset until the PVDE bit is set.

Bit 1 **SBF:** Standby flag

This bit is set by hardware and cleared only by a POR/PDR (power on reset/power down reset) or by setting the CSBF bit in the *PWR power control register (PWR_CR)*

0: Device has not been in Standby mode
1: Device has been in Standby mode

Bit 0 **WUF:** Wakeup flag

This bit is set by hardware and cleared only by a POR/PDR (power on reset/power down reset) or by setting the CWUF bit in the *PWR power control register (PWR_CR)*

0: No wakeup event occurred
1: A wakeup event was received from the WKUP pin or from the RTC alarm (Alarm A or Alarm B), RTC Tamper event, RTC TimeStamp event or RTC Wakeup).

Note: An additional wakeup event is detected if the WKUP pins are enabled (by setting the EWUPx (x=1, 2, 3) bits) when the WKUP pin levels are already high.

4.4.3 PWR register map

The following table summarizes the PWR registers.

Table 20. PWR - register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PWR_CR	Reserved																	LPRUN	Reserved	VOS [1:0]		FWU	ULP	DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPSDSR
	Reset value																		0		1	0	0	0	0	0	0	0	0	0	0	0	0

Table 20. PWR - register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	PWR_CSR	Reserved																					EWUP3	EWUP2	EWUP1	Reserved		REGLPF	VOSF	VREFINTRDYF	PVDO	SBF	WUF
	Reset value																						0	0	0			0	0	1	0	0	0

Refer to *Table 2 on page 42* for the register boundary addresses.

5 Reset and clock control (RCC)

5.1 Reset

There are three types of reset, defined as system reset, power reset and RTC domain reset.

5.1.1 System reset

A system reset sets all registers to their reset values except for the RTC, RTC backup registers and control/status register, RCC_CSR.

A system reset is generated when one of the following events occurs:

1. A low level on the NRST pin (external reset)
2. Window watchdog end-of-count condition (WWDG reset)
3. Independent watchdog end-of-count condition (IWDG reset)
4. A software reset (SW reset) (see *Software reset*)
5. Low-power management reset (see *Low-power management reset*)
6. Option byte loader reset (see *Option byte loader reset*)
7. Exit from Standby mode

The reset source can be identified by checking the reset flags in the control/status register, RCC_CSR (see *Section 5.3.14*).

Software reset

The SYSRESETREQ bit in Cortex™-M3 Application Interrupt and Reset Control Register must be set to force a software reset on the device. Refer to the Cortex™-M3 technical reference manual for more details.

Low-power management reset

There are two ways to generate a low-power management reset:

1. Reset generated when entering Standby mode:

   This type of reset is enabled by resetting nRST_STDBY bit in user option bytes. In this case, whenever a Standby mode entry sequence is successfully executed, the device is reset instead of entering Standby mode.
2. Reset when 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.

Option byte loader reset

The Option byte loader reset is generated when the OBL_LAUNCH bit (bit 18) is set in the FLASH_PECR register. This bit is used to launch by software the option byte loading.

For further information on the user option bytes, refer to the STM32L1xxxx Flash programming manual (PM0062).

5.1.2 Power reset

A power reset is generated when one of the following events occurs:

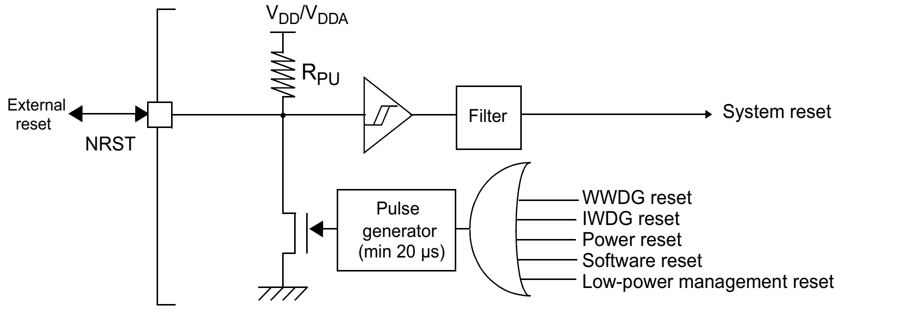
1. Power-on/power-down reset (POR/PDR reset)
2. BOR reset

A power reset sets all registers to their reset values including for the RTC domain (see *Figure 11*)

These sources act on the NRST pin and it is always kept low during the delay phase. The RESET service routine vector is fixed at address 0x0000_0004 in the memory map. For more details, refer to *Table 35: Vector table (low and medium density devices)*, *Table 36: Vector table (medium+ density devices)* and *Table 37: Vector table (high-density devices).*

The system reset signal provided to the device is output on the NRST pin. The pulse generator guarantees a minimum reset pulse duration of 20 µs for each internal reset source. In case of an external reset, the reset pulse is generated while the NRST pin is asserted low.

Figure 11. Simplified diagram of the reset circuit



5.1.3 RTC and backup registers reset

The RTC peripheral, RTC clock source selection (in RCC_CSR) and the backup registers are reset only when one of the following events occurs:

1. A software reset, triggered by setting the RTCRST bit in the RCC_CSR register (see *Section 5.3.14*)
2. Power reset (BOR/POR/PDR)

5.2 Clocks

Four different clock sources can be used to drive the system clock (SYSCLK):

- HSI ((high-speed internal) oscillator clock
- HSE (high-speed external) oscillator clock
- PLL clock
- MSI (multispeed internal) oscillator clock

  The MSI is used as system clock source after startup from Reset, wake-up from Stop or Standby low power modes.

The devices have the following two secondary clock sources:

- 37 kHz low speed internal RC (LSI RC) which drives the independent watchdog and optionally the RTC used for Auto-wakeup from Stop/Standby mode.
- 32.768 kHz low speed external crystal (LSE crystal) which optionally drives the real-time clock (RTCCLK)

Each clock source can be switched on or off independently when it is not used, to optimize power consumption.

Several prescalers allow the configuration of the AHB frequency, the high speed APB (APB2) and the low speed APB (APB1) domains. The maximum frequency of the AHB, APB1 and the APB2 domains is 32 MHz. It may depend on the device voltage range, for more details please refer to the Dynamic voltage scaling management section in the PWR chapter.

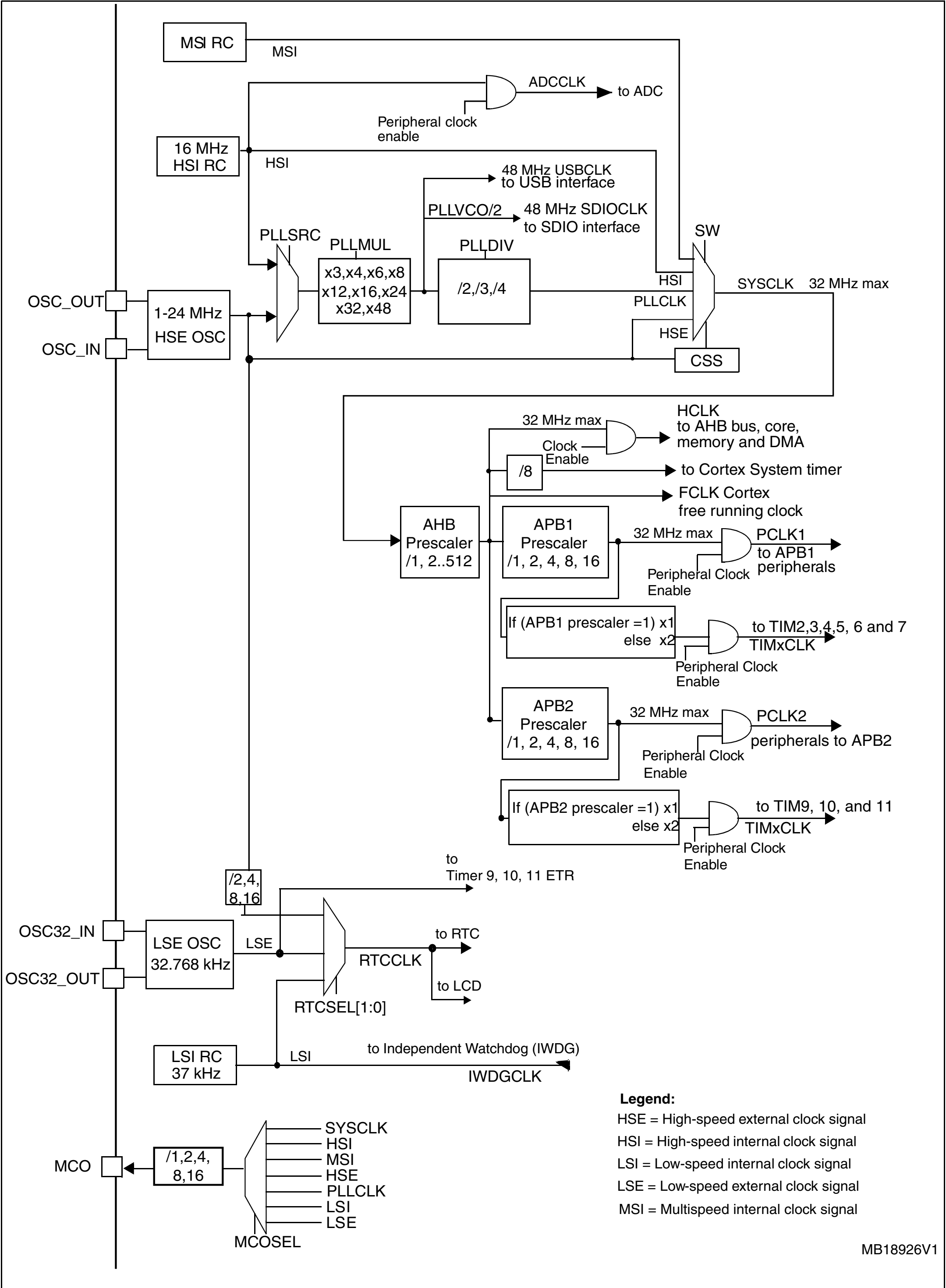
All the peripheral clocks are derived from the system clock (SYSCLK) except:

- The 48 MHz clock USB and SDIO clocks which are derived from the PLL VCO clock.
- The ADC clock which is always the HSI clock. A divider by 1, 2 or 4 allows to adapt the clock frequency to the device operating conditions. For more details please refer to the Operating Power Supply Range section in the PWR chapter.
- The RTC/LCD clock which is derived from the LSE, LSI or 1 MHz HSE_RTC (HSE divided by a programmable prescaler).
- IWDG clock which is always the LSI clock.

The system clock (SYSCLK) frequency must be higher or equal to the RTC/LCD clock frequency.

The RCC feeds the Cortex System Timer (SysTick) external clock with the AHB clock (HCLK) divided by 8. The SysTick can work either with this clock or with the Cortex clock (HCLK), configurable in the SysTick Control and Status Register.

Figure 12. Clock tree



1. For full details about the internal and external clock source characteristics, please refer to the “Electrical characteristics” section in your device datasheet.

The timer clock frequencies are automatically fixed by hardware. There are two cases:

1. If the APB prescaler is 1, the timer clock frequencies are set to the same frequency as that of the APB domain to which the timers are connected.
2. Otherwise, they are set to twice (×2) the frequency of the APB domain to which the timers are connected.

FCLK acts as Cortex™-M3 free running clock. For more details refer to the ARM Cortex™-M3 Technical Reference Manual.

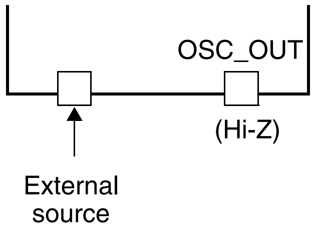
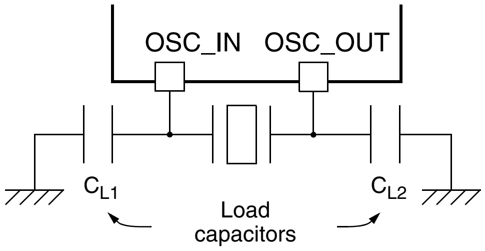
5.2.1 HSE clock

The high speed external clock signal (HSE) can be generated from two possible clock sources:

- HSE external crystal/ceramic resonator
- HSE user external clock

The resonator and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected oscillator.

Figure 13. HSE/ LSE clock sources

Clock source	Hardware configuration
External clock	
Crystal/Ceramic resonators	 OSC_IN OSC_OUT C_{L1} C_{L2} Load capacitors

External source (HSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 32 MHz. This mode is selected by setting the HSEBYP and HSEON bits in the *Clock control register, RCC_CR (see Section 5.3.1)*. The external clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC_IN pin while the OSC_OUT pin should be left hi-Z (see *Figure 13*).

External crystal/ceramic resonator (HSE crystal)

The 1 to 24 MHz external oscillator has the advantage of producing a very accurate rate on the main clock.

The associated hardware configuration is shown in *Figure 13*. Refer to the electrical characteristics section of the *datasheet* for more details.

The HSERDY flag of the *RCC_CR register (see Section 5.3.1)* indicates wether the HSE oscillator is stable or not. At startup, the HSE clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *RCC_CR register*.

The HSE Crystal can be switched on and off using the HSEON bit in the *RCC_CR register*.

5.2.2 HSI clock

The HSI clock signal is generated from an internal 16 MHz RC oscillator. It can be used directly as a system clock or as PLL input.

The HSI RC oscillator has the advantage of providing a clock source at low cost (no external components). It also has a faster startup time than the HSE crystal oscillator however, even with calibration the frequency is less accurate than an external crystal oscillator or ceramic resonator.

Calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations, this is why each device is factory calibrated by ST for 1% accuracy at an ambient temperature, T_A, of 25 °C.

After reset, the factory calibration value is loaded in the HSICAL[7:0] bits in the *Internal Clock Sources Calibration Register (RCC_ICSCR) (see Section 5.3.2).*

If the application is subject to voltage or temperature variations, this may affect the RC oscillator speed. You can trim the HSI frequency in the application by using the HSITRIM[4:0] bits in the *RCC_ICSCR register. For more details on how to measure the HSI frequency variation please refer to Section 5.2.14: Internal/external clock measurement with TIM9/TIM10/TIM11.*

The HSIRDY flag in the *RCC_CR* indicates wether the HSI oscillator is stable or not. At startup, the HSI RC output clock is not released until this bit is set by hardware.

The HSI RC oscillator can be switched on and off using the HSION bit in the *RCC_CR register*.

5.2.3 MSI clock

The MSI clock signal is generated from an internal RC oscillator. Its frequency range can be adjusted by software by using the MSIRANGE[2:0] bits in the RCC_ICSCR register (see *Section 5.3.2: Internal clock sources calibration register (RCC_ICSCR)*). Seven frequency ranges are available: 65.536 kHz, 131.072 kHz, 262.144 kHz, 524.288 kHz, 1.048 MHz, 2.097 MHz (default value) and 4.194 MHz.

The MSI clock is used as system clock after restart from Reset, wake-up from Stop, and Standby low power mode. After restart from Reset or wake-up from Standby, the MSI frequency is set to its default value. The MSI frequency does not change after waking up from Stop.

The MSI RC oscillator has the advantage of providing a low-cost (no external components) low-power clock source. It is used as wake-up clock in low power modes to reduce power consumption and wake-up time.

The MSIRDY flag in the *RCC_CR* register indicates wether the MSI RC is stable or not. At startup, the MSI RC output clock is not released until this bit is set by hardware.

The MSI RC can be switched on and off by using the MSION bit in the *RCC_CR register (see Section 5.3.1)*.

It can also be used as a backup clock source (auxiliary clock) if the HSE crystal oscillator fails. Refer to *Section 5.2.9: Clock security system (CSS) on page 95*.

Calibration

The MSI RC oscillator frequency can vary from one chip to another due to manufacturing process variations, this is why each device is factory calibrated by ST for 1% accuracy at an ambient temperature, T_A, of 25 °C.

After reset, the factory calibration value is loaded in the MSICAL[7:0] bits in the RCC_ICSCR register. If the application is subject to voltage or temperature variations, this may affect the RC oscillator speed. You can trim the MSI frequency in the application by using the MSITRIM[7:0] bits in the RCC_ICSCR register. For more details on how to measure the MSI frequency variation please refer to *Section 5.2.14: Internal/external clock measurement with TIM9/TIM10/TIM11*.

5.2.4 PLL

The internal PLL can be clocked by the HSI RC or HSE crystal. It is used to drive the system clock and to generate the 48 MHz clock for the USB peripheral (refer to *Figure 12* and *Section 5.3.1: Clock control register (RCC_CR)*.

The PLL input clock frequency must be between 2 and 24 MHz.

The desired frequency is obtained by using the multiplication factor and output division embedded in the PLL:

- If the USB or SDIO interface is used in the application, the PLL VCO clock (defined by the PLL multiplication factor) must be programmed to output a 96 MHz frequency. This is required to provide a 48 MHz clock to the USB or SDIO (SDIOCLK or USBCLK = PLLVCO/2).
- The system clock is derived from the PLL VCO divided by the output division factor.

Note: The application software must set correctly the PLL multiplication factor to avoid exceeding
96 MHz as PLLVCO when the product is in range 1,
48 MHz as PLLVCO when the product is in range 2,
24 MHz when the product is in range 3.
It must also set correctly the output division to avoid exceeding 32 MHz as SYSCLK.

The minimum input clock frequency for PLL is 2 MHz (when using HSE as PLL source).

The PLL configuration (selection of the source clock, multiplication factor and output division factor) must be performed before enabling the PLL. Once the PLL is enabled, these parameters cannot be changed.

To modify the PLL configuration, proceed as follows:

1. Disable the PLL by setting PLLON to 0.
2. Wait until PLLRDY is cleared. PLLRDY. The PLL is now fully stopped.
3. Change the desired parameter.
4. Enable the PLL again by setting PLLON to 1.

An interrupt can be generated when the PLL is ready if enabled in the RCC_CIR register (see *Section 5.3.4*).

5.2.5 LSE clock

The LSE crystal is a 32.768 kHz low speed external crystal or ceramic resonator. It has the advantage of providing a low-power but highly accurate clock source to the real-time clock peripheral (RTC) for clock/calendar or other timing functions.

The LSE crystal is switched on and off using the LSEON bit in the RCC_CSR register (see *Section 5.3.14*).

The LSERDY flag in the RCC_CSR register indicates wether 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_CIR register (see *Section 5.3.4*).

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_CR (see Section 5.3.1*). The external clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC32_IN pin while the OSC32_OUT pin should be left Hi-Z (see *Figure 13*).

5.2.6 LSI clock

The LSI RC acts as an low-power clock source that can be kept running in Stop and Standby mode for the independent watchdog (IWDG). The clock frequency is around 37 kHz.

The LSI RC oscillator can be switched on and off using the LSION bit in the RCC_CSR register (see *Section 5.3.14*).

The LSIRDY flag in RCC_CSR indicates wether the low-speed internal oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the RCC_CIR (see *Section 5.3.4*).

LSI measurement

The frequency dispersion of the LSI oscillator can be measured to have accurate RTC time base and/or IWDG timeout (when LSI is used as clock source for these peripherals) with an acceptable accuracy. For more details, refer to the electrical characteristics section of the datasheets. For more details on how to measure the LSI frequency, please refer to *Section 5.2.14: Internal/external clock measurement with TIM9/TIM10/TIM11*.

5.2.7 System clock (SYSCLK) selection

Four different clock sources can be used to drive the system clock (SYSCLK):

- The HSI oscillator
- The HSE oscillator
- The PLL
- The MSI oscillator clock (default after reset)

When a clock source is used directly or through the PLL as system clock, it is not possible to stop it.

A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source which is not yet ready is selected, the switch will occur when the clock source will be ready. Status bits in the RCC_CR register indicate which clock(s) is (are) ready and which clock is currently used as system clock.

5.2.8 System clock source frequency versus voltage range

The following table gives the different clock source frequencies depending on the product voltage range.

Table 21. System clock source frequency

Product voltage range	Clock frequency			
	MSI	HSI	HSE	PLL
Range 1 (1.8 V)	4.2 MHz	16 MHz	HSE 32 MHz (external clock) or 24 MHz (crystal)	32 MHz (PLLVCO max = 96 MHz)
Range 2 (1.5 V)	4.2 MHz	16 MHz	16 MHz	16 MHz (PLLVCO max = 48 MHz)
Range 3 (1.2 V)	4.2 MHz	NA	8 MHz	4 MHz (PLLVCO max = 24 MHz)

5.2.9 Clock security system (CSS)

The Clock security system can be activated by software. In this case, the clock detector is enabled after the HSE oscillator startup delay, and disabled when this oscillator is stopped.

If a failure is detected on the HSE clock, this oscillator is automatically disabled and an interrupt is generated to inform the software about the failure (Clock Security System Interrupt, CSSI), allowing the MCU to perform rescue operations. The CSSI is linked to the Cortex™-M3 NMI (Non-Maskable Interrupt) exception vector.

Note: Once the CSS is enabled and if the HSE clock fails, the CSS interrupt occurs and an NMI is automatically generated. The NMI will be executed indefinitely unless the CSS interrupt pending bit is cleared. As a consequence, in the NMI ISR must clear the CSS interrupt by setting the CSSC bit in the RCC_CIR register.

If the HSE oscillator is used directly or indirectly as the system clock (indirectly means: it is used as PLL input clock, and the PLL clock is used as system clock), a detected failure causes a switch of the system clock to the MSI oscillator and the disabling of the HSE oscillator. If the HSE oscillator clock is the clock entry of the PLL used as system clock when the failure occurs, the PLL is disabled too.

5.2.10 Clock Security System on LSE

In high density and medium+ devices, a Clock Security System on LSE can be activated by software writing the LSECSSON bit in the RCC_CSR register. This bit can be disabled only by a hardware reset or RTC software reset, or after a failure detection on LSE. LSECSSON must be written after LSE and LSI are enabled (LSEON and LSION enabled) and ready (LSERDY and LSIRDY set by hardware), and after the RTC clock has been selected by RTCSEL.

The CSS on LSE is working in all modes: run, Sleep, Stop and Standby.

If a failure is detected on the external 32 kHz oscillator, the LSE clock is no longer supplied to the RTC but no hardware action is made to the registers.

In Standby mode a wakeup is generated. In other modes an interrupt can be sent to wake-up the software (see *Section 5.3.4: Clock interrupt register (RCC_CIR) on page 104*).

The software MUST then disable the LSECSSON bit, stop the defective 32 kHz oscillator (disabling LSEON), and can change the RTC clock source (no clock or LSI or HSE, with RTCSEL), or take any required action to secure the application.

5.2.11 RTC and LCD clock

The RTC and LCD have the same clock source which can be either the LSE, the LSI, or the HSE 1 MHz clock (HSE divided by a programmable prescaler). It is selected by programming the RTCSEL[1:0] bits in the RCC_CSR register (see *Section 5.3.14*) and the RTCPRE[1:0] bits in the RCC_CR register (see *Section 5.3.1*).

Once the RTC and LCD clock source have been selected, the only possible way of modifying the selection is to set the RTCRST bit in the RCC_CSR register, or by a POR.

If the LSE or LSI is used as RTC clock source, the RTC continues to work in Stop and Standby low power modes, and can be used as wakeup source. However, when the HSE is the RTC clock source, the RTC cannot be used in the Stop and Standby low power modes. The LCD can however be used in the Stop low power mode if the LSE or LSI is used as the RTC clock source.

Note: *To be able to read the RTC calendar register when the APB1 clock frequency is less than seven times the RTC clock frequency (7*RTCLCK), 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.

5.2.12 Watchdog clock

If the Independent watchdog (IDG) is started by either hardware option or software access, the LSI oscillator is forced ON and cannot be disabled. After the LSI oscillator temporization, the clock is provided to the IWDG.

5.2.13 Clock-out capability

The microcontroller clock output (MCO) capability allows the clock to be output onto the external MCO pin (PA8) using a configurable prescaler (1, 2, 4, 8, or 16). The configuration registers of the corresponding GPIO port must be programmed in alternate function mode. One of 7 clock signals can be selected as the MCO clock:

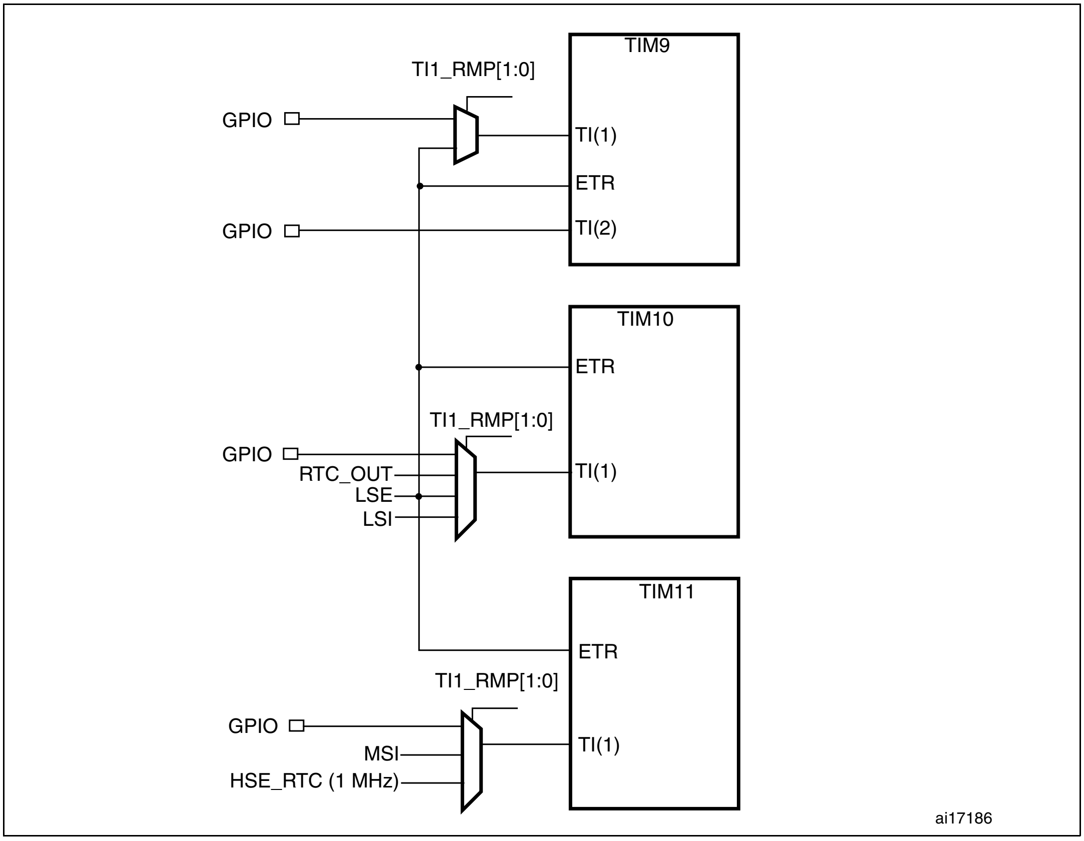
- SYSCLK
- HSI
- MSI
- HSE
- PLL
- LSI
- LSE

The selection is controlled by the MCOSEL[2:0] bits of the RCC_CFGR register (see *Section 5.3.3*).

5.2.14 Internal/external clock measurement with TIM9/TIM10/TIM11

It is possible to indirectly measure the frequency of all on-board clock source generators by means of the TIM9/TIM10/TIM11 channel 1 input capture, as represented on *Figure 14*.

Figure 14. Using the TIM9/TIM10/TIM11 channel 1 input capture to measure frequencies



Each timer has an input multiplexer that selects which of the I/O or the internal clock is to trigger the input capture. This selection is performed through the TI1_RMP [1:0] bits in the TIMx_OR register.

For TIM9 and TIM10, the primary purpose of connecting the LSE to the channel 1 input capture is to be able to precisely measure the HSI and MSI system clocks (for this, either the HSI or MSI should be used as the system clock source). The number of HSI (MSI, respectively) clock counts between consecutive edges of the LSE signal provides a measure of the internal clock period. Taking advantage of the high precision of LSE crystals (typically a few tens of ppm’s), it is possible to determine the internal clock frequency with the same resolution, and trim the source to compensate for manufacturing-process- and/or temperature- and voltage-related frequency deviations.

The MSI and HSI oscillators both have dedicated user-accessible calibration bits for this purpose.

The basic concept consists in providing a relative measurement (e.g. the HSI/LSE ratio): the precision is therefore closely related to the ratio between the two clock sources. The higher the ratio, the better the measurement.

It is however not possible to have a good enough resolution when the MSI clock is low (typically below 1 MHz). In this case, it is advised to:

- accumulate the results of several captures in a row
- use the timer’s input capture prescaler (up to 1 capture every 8 periods)
- use the RTC_OUT signal at 512 Hz (when the RTC is clocked by the LSE) as the input for the channel1 input capture. This improves the measurement precision

TIM10 can also be used to measure the LSI: this is useful for applications with no crystal. The ultralow power LSI oscillator has a wide manufacturing process deviation: by measuring it as a function of the HSI clock source, it is possible to determine its frequency with the precision of the HSI.

Finally, TIM11 has two other sources. TIM11 can use the MSI just like TIM10 uses the LSI for crystal-less applications. The HSE_RTC frequency (HSE divided by a programmable prescaler) being relatively high (1MHz), the relative frequency measurement is not very precise, so its main purpose is to have a rough indication of the external crystal frequency. This is useful for instance to meet the requirements of the IEC 60730/IEC 61335 standards, which requires to be able to determine harmonic or subharmonic frequencies (–50/+100% deviations).

5.2.15 Clock-independent system clock sources for TIM9/TIM10/TIM11

In a number of applications using the 32.768 kHz clock as a time base for the RTC, it is interesting to have time bases that work completely independently of the system clock. This allows the scheduling of tasks without having to take into account the processor state (the processor may be stopped or executing at low, medium or full speed).

For this purpose, the LSE clock is internally redirected to the 3 timers’ ETR inputs, which are used as additional clock sources, as shown in *Figure 14 on page 97*. This gives up to three independent time bases (using the auto-reload feature) with 1 or 2 compare additional channels for fractional events. For instance, the TIM9’s auto-reload interrupt can be programmed for a 1 second tick interrupt with an additional interrupt occurring 250 ms after the main tick.

Note: In this configuration, make sure that you have at least a ratio of 2 between the external clock (LSE) and the APB clock. If the application uses an APB clock frequency lower than twice the LSE clock frequency (typically LSE = 32.768 kHz, so twice LSE = 65.536 kHz), it is mandatory to use the external trigger prescaler feature of the timer: it can divide the ETR clock by up to 8.

5.3 RCC registers

Refer to *Section 1.1* for a list of abbreviations used in register descriptions.

5.3.1 Clock control register (RCC_CR)

Address offset: 0x00

Reset value: 0x0000 0300

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.	RTCPRE[1:0]		CSS ON	Reserved		PLL RDY	PLLON	Reserved					HSE BYP	HSE RDY	HSE ON
	rw	rw	rw			r	rw						rw	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						MSI RDY	MSION	Reserved						HSI RDY	HSION
						r	rw							r	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **RTCPRE[1:0]** RTC/LCD prescaler

These bits are set and reset by software to obtain a 1 MHz clock from HSE. This prescaler cannot be modified if HSE is enabled (HSEON = 1).

00: HSE is divided by 2 for RTC/LCD clock
01: HSE is divided by 4 for RTC/LCD clock
10: HSE is divided by 8 for RTC/LCD clock
11: HSE is divided by 16 for RTC/LCD clock

Bit 28 **CSSON**: Clock security system enable

This bit is set and cleared by software to enable the clock security system (CSS). When CSSON is set, the clock detector is enabled by hardware when the HSE oscillator is ready, and disabled by hardware if an oscillator failure is detected.

0: Clock security system OFF (clock detector OFF)
1: Clock security system ON (clock detector ON if HSE oscillator is stable, OFF otherwise)

Bits 27:26 Reserved, must be kept at reset value.

Bit 25 **PLLRDY:** PLL clock ready flag

This bit is set by hardware to indicate that the PLL is locked.

0: PLL unlocked
1: PLL locked

Bit 24 **PLLON:** PLL enable

This bit is set and cleared by software to enable PLL.

Cleared by hardware when entering Stop or Standby mode. This bit can not be reset if the PLL clock is used as system clock or is selected to become the system clock.

0: PLL OFF
1: PLL ON

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

Bit 18 **HSEBYP**: HSE clock bypass

This bit is set and cleared by software to bypass the oscillator with an external clock. The external clock must be enabled with the HSEON bit, to be used by the device.
The HSEBYP bit can be written only if the HSE oscillator is disabled.

0: HSE oscillator not bypassed
1: HSE oscillator bypassed with an external clock

Bit 17 **HSERDY**: HSE clock ready flag

This bit is set by hardware to indicate that the HSE oscillator is stable. After the HSEON bit is cleared, HSERDY goes low after 6 HSE oscillator clock cycles.

0: HSE oscillator not ready
1: HSE oscillator ready

Bit 16 **HSEON**: HSE clock enable

This bit is set and cleared by software.

Cleared by hardware to stop the HSE oscillator when entering Stop or Standby mode. This bit cannot be reset if the HSE oscillator is used directly or indirectly as the system clock.

0: HSE oscillator OFF
1: HSE oscillator ON

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

Bit 9 **MSIRDY**: MSI clock ready flag

This bit is set by hardware to indicate that the MSI oscillator is stable.

0: MSI oscillator not ready
1: MSI oscillator ready

Note: Once the MSION bit is cleared, MSIRDY goes low after 6 MSI clock cycles.

Bit 8 **MSION**: MSI clock enable

This bit is set and cleared by software.

Set by hardware to force the MSI oscillator ON when exiting from Stop or Standby mode, or in case of a failure of the HSE oscillator used directly or indirectly as system clock. This bit cannot be cleared if the MSI is used as system clock.

0: MSI oscillator OFF
1: MSI oscillator ON

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

Bit 1 **HSIRDY**: Internal high-speed clock ready flag

This bit is set by hardware to indicate that the HSI oscillator is stable. After the HSION bit is cleared, HSIRDY goes low after 6 HSI clock cycles.

0: HSI oscillator not ready
1: HSI oscillator ready

Bit 0 **HSION**: Internal high-speed clock enable

This bit is set and cleared by software.

This bit cannot be cleared if the HSI is used directly or indirectly as the system clock.

0: HSI oscillator OFF
1: HSI oscillator ON

5.3.2 Internal clock sources calibration register (RCC_ICSCR)

Address offset: 0x04

Reset value: 0x00XX B0XX where X is undefined.

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
MSITRIM[7:0]								MSICAL[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSIRANGE[2:0]			HSITRIM[4:0]					HSICAL[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r

Bits 31:24 **MSITRIM[7:0]**: MSI clock trimming

These bits are set by software to adjust MSI calibration.

These bits provide an additional user-programmable trimming value that is added to the MSICAL[7:0] bits. They can be programmed to compensate for the variations in voltage and temperature that influence the frequency of the internal MSI RC.

Bits 23:16 **MSICAL[7:0]**: MSI clock calibration

These bits are automatically initialized at startup.

Bits 15:13 **MSIRANGE[2:0]**: MSI clock ranges

These bits are set by software to choose the frequency range of MSI.7 frequency ranges are available:

000: range 0 around 65.536 kHz
001: range 1 around 131.072 kHz
010: range 2 around 262.144 kHz
011: range 3 around 524.288 kHz
100: range 4 around 1.048 MHz
101: range 5 around 2.097 MHz (reset value)
110: range 6 around 4.194 MHz
111: not allowed

Bits 12:8 **HSITRIM[4:0]**: High speed internal clock trimming

These bits provide an additional user-programmable trimming value that is added to the HSICAL[7:0] bits. They can be programmed to compensated for the variations in voltage and temperature that influence the frequency of the internal HSI RC.

Bits 7:0 **HSICAL[7:0]** Internal high speed clock calibration

These bits are initialized automatically at startup.

5.3.3 Clock configuration register (RCC_CFGR)

Address offset: 0x08

Reset value: 0x0000 0000

Access: 0 ≤wait state ≤2, word, half-word and byte access

1 or 2 wait states inserted only if the access occurs during clock source switch.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MCOPRE[2:0]			Res.	MCOSEL[2:0]			PLLDIV[1:0]		PLLMUL[3:0]				Res.	PLL SRC
	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw		rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

Bits 31 Reserved, must be kept at reset value.

Bits 30:28 **MCOPRE[2:0]:** Microcontroller clock output prescaler

These bits are set and cleared by software.

It is highly recommended to change this prescaler before MCO output is enabled.

000: MCO is divided by 1
001: MCO is divided by 2
010: MCO is divided by 4
011: MCO is divided by 8
100: MCO is divided by 16
Others: not allowed

Bits 27 Reserved, must be kept at reset value.

Bits 26:24 **MCOSEL[2:0]:** Microcontroller clock output selection

These bits are set and cleared by software.

000: MCO output disabled, no clock on MCO
001: SYSCLK clock selected
010: HSI oscillator clock selected
011: MSI oscillator clock selected
100: HSE oscillator clock selected
101: PLL clock selected
110: LSI oscillator clock selected
111:LSE oscillator clock selected

Note: This clock output may have some truncated cycles at startup or during MCO clock source switching.

Bits 23:22 **PLLDIV[1:0]**: PLL output division

These bits are set and cleared by software to control PLL output clock division from PLL VCO clock. These bits can be written only when the PLL is disabled.

00: not allowed
01: PLL clock output = PLLVCO / 2
10: PLL clock output = PLLVCO / 3
11: PLL clock output = PLLVCO / 4

Bits 21:18 **PLLMUL[3:0]:** PLL multiplication factor

These bits are written by software to define the PLL multiplication factor to generate the PLL VCO clock. These bits can be written only when the PLL is disabled.

0000: PLLVCO = PLL clock entry x 3
0001: PLLVCO = PLL clock entry x 4
0010: PLLVCO = PLL clock entry x 6
0011: PLLVCO = PLL clock entry x 8
0100: PLLVCO = PLL clock entry x 12
0101: PLLVCO = PLL clock entry x 16
0110: PLLVCO = PLL clock entry x 24
0111: PLLVCO = PLL clock entry x 32
1000: PLLVCO = PLL clock entry x 48
others: not allowed

Caution: The PLL VCO clock frequency must not exceed 96 MHz when the product is in Range 1, 48 MHz when the product is in Range 2 and 24 MHz when the product is in Range 3.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **PLLSRC:** PLL entry clock source

This bit is set and cleared by software to select PLL clock source. This bit can be written only when PLL is disabled.

0: HSI oscillator clock selected as PLL input clock
1: HSE oscillator clock selected as PLL input clock

Note: The PLL minimum input clock frequency is 2 MHz.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:11 **PPRE2[2:0]**: APB high-speed prescaler (APB2)

These bits are set and cleared by software to control the division factor of the APB high-speed clock (PCLK2).

0xx: HCLK not divided
100: HCLK divided by 2
101: HCLK divided by 4
110: HCLK divided by 8
111: HCLK divided by 16

Bits 10:8 **PPRE1[2:0]**: APB low-speed prescaler (APB1)

These bits are set and cleared by software to control the division factor of the APB low-speed clock (PCLK1).

0xx: HCLK not divided
100: HCLK divided by 2
101: HCLK divided by 4
110: HCLK divided by 8
111: HCLK divided by 16

Bits 7:4 **HPRE[3:0]:** AHB prescaler

These bits are set and cleared by software to control the division factor of the AHB clock.

Caution: Depending on the device voltage range, the software has to set correctly these bits to ensure that the system frequency does not exceed the maximum allowed frequency (for more details please refer to the Dynamic voltage scaling management section in the PWR chapter.) After a write operation to these bits and before decreasing the voltage range, this register must be read to be sure that the new value has been taken into account.

0xxx: SYSCLK not divided
1000: SYSCLK divided by 2
1001: SYSCLK divided by 4
1010: SYSCLK divided by 8
1011: SYSCLK divided by 16
1100: SYSCLK divided by 64
1101: SYSCLK divided by 128
1110: SYSCLK divided by 256
1111: SYSCLK divided by 512

Bits 3:2 **SWS[1:0]**: System clock switch status

These bits are set and cleared by hardware to indicate which clock source is used as system clock.

00: MSI oscillator used as system clock
01: HSI oscillator used as system clock
10: HSE oscillator used as system clock
11: PLL used as system clock

Bits 1:0 **SW[1:0]**: System clock switch

These bits are set and cleared by software to select SYSCLK source.

Set by hardware to force MSI selection when leaving Stop and Standby mode or in case of failure of the HSE oscillator used directly or indirectly as system clock (if the Clock Security System is enabled).

00: MSI oscillator used as system clock
01: HSI oscillator used as system clock
10: HSE oscillator used as system clock
11: PLL used as system clock

5.3.4 Clock interrupt register (RCC_CIR)

Address offset: 0x0C

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
Reserved								CSSC	LSECSSC	MSIRDYC	PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC
								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.	LSECSSIE	MSIRDYIE	PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	LSECSSF	MSIRDYF	PLLRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r

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

Bit 23 **CSSC:** Clock security system interrupt clear

This bit is set by software to clear the CSSF flag.
0: No effect
1: Clear CSSF flag

Bit 22 **LSECSSC:** LSE CSS interrupt clear

Set by software to clear LSECSSF. Reset by hardware when clear done.
0: LSECSSF not cleared
1: LSECSSF cleared

Note: This bit is available in high and medium+ density devices only.

Bit 21 **MSIRDYC:** MSI ready interrupt clear

This bit is set by software to clear the MSIRDYF flag.
0: No effect
1: MSIRDYF cleared

Bit 20 **PLLRDYC:** PLL ready interrupt clear

This bit is set by software to clear the PLLRDYF flag.
0: No effect
1: PLLRDYF cleared

Bit 19 **HSERDYC:** HSE ready interrupt clear

This bit is set by software to clear the HSERDYF flag.
0: No effect
1: HSERDYF cleared

Bit 18 **HSIRDYC:** HSI ready interrupt clear

This bit is set software to clear the HSIRDYF flag.
0: No effect
1: HSIRDYF cleared

Bit 17 **LSERDYC:** LSE ready interrupt clear

This bit is set by software to clear the LSERDYF flag.
0: No effect
1: LSERDYF cleared

Bit 16 **LSIRDYC:** LSI ready interrupt clear

This bit is set by software to clear the LSIRDYF flag.
0: No effect
1: LSIRDYF cleared

Bit 15 Reserved, must be kept at reset value.

Bit 14 **LSECSSIE:** LSE CSS interrupt enable

Set and reset by software to enable/disable interrupts from the Clock Security System on external 32 kHz oscillator (LSE).
0: LSE CSS interrupt disabled
1: LSE CSS interrupt enabled

Note: This bit is available in high and medium+ density devices only.

Bit 13 **MSIRDYIE:** MSI ready interrupt enable

This bit is set and cleared by software to enable/disable interrupt caused by the MSI oscillator stabilization.
0: MSI ready interrupt disabled
1: MSI ready interrupt enabled

Bit 12 **PLLRDYIE:** PLL ready interrupt enable

This bit is set and cleared by software to enable/disable interrupt caused by PLL lock.
0: PLL lock interrupt disabled
1: PLL lock interrupt enabled

Bit 11 **HSERDYIE:** HSE ready interrupt enable

This bit is set and cleared by software to enable/disable interrupt caused by the HSE oscillator stabilization.
0: HSE ready interrupt disabled
1: HSE ready interrupt enabled

Bit 10 **HSIRDYIE:** HSI ready interrupt enable

This bit is set and cleared by software to enable/disable interrupt caused by the HSI oscillator stabilization.
0: HSI ready interrupt disabled
1: HSI ready interrupt enabled

Bit 9 **LSERDYIE:** LSE ready interrupt enable

This bit is set and cleared by software to enable/disable interrupt caused by the LSE oscillator stabilization.
0: LSE ready interrupt disabled
1: LSE ready interrupt enabled

Bit 8 **LSIRDYIE:** LSI ready interrupt enable

This bit is set and cleared by software to enable/disable interrupt caused by LSI oscillator stabilization.
0: LSI ready interrupt disabled
1: LSI ready interrupt enabled

Bit 7 **CSSF:** Clock security system interrupt flag

This bit is set by hardware when a failure is detected in the HSE oscillator.
It is cleared by software by setting the CSSC bit.
0: No clock security interrupt caused by HSE clock failure
1: Clock security interrupt caused by HSE clock failure

Bit 6 **LSECSSF** LSE CSS Interrupt flag

Reset by software by writing to the LSECSSC bit. Set by hardware when a failure is detected on the external 32 KHz oscillator and the LSECSSIE bit is set.
0: No failure detected on the external 32 KHz oscillator (LSE)
1: A failure is detected on the external 32 kHz oscillator (LSE)

Note: This bit is available in high and medium+ density devices only.

Bit 5 **MSIRDYF:** MSI ready interrupt flag

This bit is set by hardware when the MSI becomes stable and MSIRDYDIE is set.
It is cleared by software setting the MSIRDYC bit.
0: No clock ready interrupt caused by the MSI
1: Clock ready interrupt caused by the MSI

Bit 4 **PLLRDYF:** PLL ready interrupt flag

This bit is set by hardware when the PLL locks and PLLRDYDIE is set.
It is cleared by software setting the PLLRDYC bit.
0: No clock ready interrupt caused by PLL lock
1: Clock ready interrupt caused by PLL lock

Bit3 **HSERDYF:** HSE ready interrupt flag

This bit is set by hardware when HSE becomes stable and HSERDYDIE is set.
It is cleared by software setting the HSERDYC bit.
0: No clock ready interrupt caused by the HSE
1: Clock ready interrupt caused by the HSE

Bit 2 **HSIRDYF:** HSI ready interrupt flag

This bit is set by hardware when the HSI becomes stable and HSIRDYDIE is set.
It is cleared by software setting the HSIRDYC bit.
0: No clock ready interrupt caused by the HSI
1: Clock ready interrupt caused by the HSI

Bit 1 **LSERDYF:** LSE ready interrupt flag

This bit is set by hardware when the LSE becomes stable and LSERDYDIE is set.
It is cleared by software setting the LSERDYC bit.
0: No clock ready interrupt caused by the LSE
1: Clock ready interrupt caused by the LSE

Bit 0 **LSIRDYF:** LSI ready interrupt flag

This bit is set by hardware when the LSI becomes stable and LSIRDYDIE is set.
It is cleared by software setting the LSIRDYC bit.
0: No clock ready interrupt caused by the LSI
1: Clock ready interrupt caused by the LSI

5.3.5 AHB peripheral reset register (RCC_AHBRSTR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	FSMC RST	Reserved		AES RST	Res.	DMA2RST	DMA1 RST	Reserved							
	rw			rw		rw	rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLITF RST	Reserved		CRC RST	Reserved				GPIOG RST	GPIOF RST	GPIOH RST	GPIOE RST	GPIOD RST	GPIOC RST	GPIOB RST	GPIOA RST
rw			rw					rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **FSMCRST:** FSMC reset

This bit is set and cleared by software.
0: No effect
1: Reset FSMC

Note: This bit is available in high density devices only.

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

Bit 27 **AESRST:** AES reset

This bit is set and cleared by software.

0: No effect

1: Reset AES

Note: This bit is available in STM32L16x devices only.

Bit 26 Reserved, must be kept at reset value.

Bit 25 **DMA2RST:** DMA2 reset

This bit is set and cleared by software.

0: No effect

1: Reset DMA2

Note: This bit is available in high and medium+ density devices only.

Bit 24 **DMA1RST:** DMA1 reset

This bit is set and cleared by software.

0: No effect

1: Reset DMA1

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

Bit 15 **FLITFRST:** FLITF reset

This bit is set and cleared by software. The FLITF reset can be enabled only when the Flash memory is in power down mode.

0: No effect

1: Reset FLITF

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

Bit 12 **CRCRST:** CRC reset

This bit is set and cleared by software.

0: No effect

1: Reset CRC

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **GPIOGRST:** IO port G reset

This bit is set and cleared by software.

0: No effect

1: Reset IO port G

Note: This bit is available in high and medium+ density devices only.

Bit 6 **GPIOFRST:** IO port F reset

This bit is set and cleared by software.

0: No effect

1: Reset IO port F

Note: This bit is available in high and medium+ density devices only.

Bit 5 **GPIOHRST:** IO port H reset

This bit is set and cleared by software.

0: No effect

1: Reset

Bit 4 **GPIOERST:** IO port E reset

This bit is set and cleared by software.

0: No effect

1: Reset IO port E

Bit 3 **GPIODRST:** IO port D reset
This bit is set and cleared by software.
0: No effect
1: Reset IO port D

Bit 2 **GPIOCRST:** IO port C reset
This bit is set and cleared by software.
0: No effect
1: Reset IO port C

Bit 1 **GPIOBRST:** IO port B reset
This bit is set and cleared by software.
0: No effect
1: Reset IO port B

Bit 0 **GPIOARST:** IO port A reset
This bit is set and cleared by software.
0: No effect
1: Reset IO port A

5.3.6 APB2 peripheral reset register (RCC_APB2RSTR)

Address offset: 0x14

Reset value: 0x00000 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
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1RST	Res.	SPI1RST	SDIORST	Res.	ADC1RST	Reserved				TIM11RST	TIM10RST	TIM9RST	Res.	SYSCFGRST
	rw		rw	rw		rw					rw	rw	rw		rw

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

Bit 14 **USART1RST:** USART1 reset
This bit is set and cleared by software.
0: No effect
1: Reset USART1

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1RST:** SPI 1 reset
This bit is set and cleared by software.
0: No effect
1: Reset SPI 1

Bit 11 **SDIORST:** SDIO reset
This bit is set and cleared by software.
0: No effect
1: Reset SDIO
Note: This bit is available in high density devices only.

Bit10 Reserved, must be kept at reset value.

Bit 9 **ADC1RST:** ADC1 interface reset

This bit is set and cleared by software.
0: No effect
1: Reset ADC1 interface

Bits 8:5 Reserved, must be kept at reset value.

Bit 4 **TIM11RST:** TIM11 timer reset

Set and cleared by software.
0: No effect
1: Reset TIM11 timer

Bit 3 **TIM10RST:** TIM10 timer reset

This bit is set and cleared by software.
0: No effect
1: Reset TIM10 timer

Bit 2 **TIM9RST:** TIM9 timer reset

This bit is set and cleared by software.
0: No effect
1: Reset TIM9 timer

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGRST:** System configuration controller reset

This bit is set and cleared by software.
0: No effect
1: Reset System configuration controller

5.3.7 APB1 peripheral reset register (RCC_APB1RSTR)

Address offset: 0x18

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP RST	Res.	DAC RST	PWR RST	Reserved				USB RST	I2C2 RST	I2C1 RST	UART5 RST	UART4 RST	USART 3 RST	USART 2 RST	Res.
rw		rw	rw					rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 RST	SPI2 RST	Reserved		WWDG RST	Res.	LCD RST	Reserved			TIM7 RST	TIM6 RST	TIM5 RST	TIM4 RST	TIM3 RST	TIM2 RST
rw	rw			rw		rw				rw	rw	rw	rw	rw	rw

Bit 31 **COMPRST:** COMP interface reset

This bit is set and cleared by software.
0: No effect
1: Reset COMP interface

Bit 30 Reserved, must be kept at reset value.

Bit 29 **DACRST:** DAC interface reset

This bit is set and cleared by software.
0: No effect
1: Reset DAC interface

Bit 28 **PWRRST:** Power interface reset

This bit is set and cleared by software.
0: No effect
1: Reset power interface

Bits 27:24 Reserved, must be kept at reset value.

Bit 23 **USBRST:** USB reset

This bit is set and cleared by software.
0: No effect
1: Reset USB

Bit 22 **I2C2RST:** I^{2}C 2 reset

This bit is set and cleared by software.
0: No effect
1: Reset I^{2}C 2

Bit 21 **I2C1RST:** I^{2}C 1 reset

This bit is set and cleared by software.
0: No effect
1: Reset I^{2}C 1

Bit 20 **UART5RST:** UART 5 reset

This bit is set and cleared by software.
0: No effect
1: Reset UART 5

Note: This bit is available in high density devices only.

Bit 19 **UART4RST:** UART 4 reset

This bit is set and cleared by software.
0: No effect
1: Reset UART 4

Note: This bit is available in high density devices only.

Bit 18 **USART3RST:** USART 3 reset

This bit is set and cleared by software.
0: No effect
1: Reset USART 3

Bit 17 **USART2RST:** USART 2 reset

This bit is set and cleared by software.
0: No effect
1: Reset USART 2

Bit16 Reserved, must be kept at reset value.

Bit 15 **SPI3RST:** SPI 3 reset

This bit is set and cleared by software.
0: No effect
1: Reset SPI 3

Note: This bit is available in high and medium+ density devices only.

Bit 14 **SPI2RST:** SPI 2 reset

This bit is set and cleared by software.
0: No effect
1: Reset SPI 2

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WWDGRST:** Window watchdog reset

This bit is set and cleared by software.
0: No effect
1: Reset window watchdog

Bits 10 Reserved, must be kept at reset value.

Bit 9 **LCDRST:** LCD reset

This bit is set and cleared by software.
0: No effect
1: Reset LCD

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **TIM7RST:** Timer 7 reset

This bit is set and cleared by software.
0: No effect
1: Reset timer 7

Bit 4 **TIM6RST:** Timer 6reset

Set and cleared by software.
0: No effect
1: Reset timer 6

Bit 3 **TIM5RST:** Timer 5 reset

Set and cleared by software.
0: No effect
1: Reset timer 5

Note: This bit is available in high and medium+ density devices only.

Bit 2 **TIM4RST:** Timer 4 reset

Set and cleared by software.
0: No effect
1: Reset timer 4

Bit 1 **TIM3RST:** Timer 3 reset

Set and cleared by software.
0: No effect
1: Reset timer 3

Bit 0 **TIM2RST:** Timer 2 reset

Set and cleared by software.
0: No effect
1: Reset timer 2

5.3.8 AHB peripheral clock enable register (RCC_AHBENR)

Address offset: 0x1C

Reset value: 0x0000 8000

Access: no wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral register values may not be readable by software and the returned value is always 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	FSMC EN	Reserved		AES EN	Res.	DMA2EN	DMA1EN	Reserved							
	rw			rw		rw	rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLITF EN	Reserved		CRCEN	Reserved				GPIOG EN	GPIOF EN	GPIOH EN	GPIOE EN	GPIOD EN	GPIOC EN	GPIOB EN	GPIOA EN
rw			rw					rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **FSMCEN:** FSMC clock enable

This bit is set and cleared by software.
0: FSMC clock disabled
1: FSMC clock enabled

Note: This bit is available in high density devices only.

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

Bit 27 **AESEN:** AES clock enable

This bit is set and cleared by software.
0: AES clock disabled
1: AES clock enabled

Note: This bit is available in STM32L16x devices only.

Bit 26 Reserved, must be kept at reset value.

Bit 25 **DMA2EN:** DMA2 clock enable

This bit is set and cleared by software.
0: DMA2 clock disabled
1: DMA2 clock enabled

Note: This bit is available in high and medium+ density devices only.

Bit 24 **DMA1EN:** DMA1 clock enable

This bit is set and cleared by software.
0: DMA1 clock disabled
1: DMA1 clock enabled

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

Bit 15 **FLITFEN:** FLITF clock enable

This bit can be written only when the Flash memory is in power down mode.
0: FLITF clock disabled
1: FLITF clock enabled

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

Bit 12 **CRCEN:** CRC clock enable

This bit is set and cleared by software.
0: CRC clock disabled
1: CRC clock enabled

Bits 11:6 Reserved, must be kept at reset value.

Bit 7 **GPIOGEN:** IO port G clock enable

This bit is set and cleared by software.
0: IO port G clock disabled
1: IO port G clock enabled

Note: This bit is available in high and medium+ density devices only.

Bit 6 **GPIOFEN:** IO port F clock enable

This bit is set and cleared by software.
0: IO port F clock disabled
1: IO port F clock enabled

Note: This bit is available in high and medium+ density devices only.

Bit 5 **GPIOHEN:** IO port H clock enable

This bit is set and cleared by software.
0: IO port H clock disabled
1: IO port H clock enabled

Bit 4 **GPIOEEN:** IO port E clock enable

This bit is set and cleared by software.
0: IO port E clock disabled
1: IO port E clock enabled

Bit 3 **GPIODEN:** IO port D clock enable

Set and cleared by software.

0: IO port D clock disabled
1: IO port D clock enabled

Bit 2 **GPIOCEN:** IO port C clock enable

This bit is set and cleared by software.
0: IO port C clock disabled
1: IO port C clock enabled

Bit 1 **GPIOBEN:** IO port B clock enable

This bit is set and cleared by software.
0: IO port B clock disabled
1: IO port B clock enabled

Bit 0 **GPIOAEN:** IO port A clock enable

This bit is set and cleared by software.
0: IO port A clock disabled
1: IO port A clock enabled

5.3.9 APB2 peripheral clock enable register (RCC_APB2ENR)

Address: 0x20

Reset value: 0x0000 0000

Access: word, half-word and byte access

No wait states, except if the access occurs while an access to a peripheral in the APB2 domain is on going. In this case, wait states are inserted until the access to APB2 peripheral is finished.

Note: When the peripheral clock is not active, the peripheral register values may not be readable by software and the returned value is always 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1 EN	Res.	SPI1 EN	SDIO EN	Res.	ADC1 EN	Reserved				TIM11 EN	TIM10 EN	TIM9 EN	Res.	SYSCF GEN
	rw		rw	rw		rw					rw	rw	rw		rw

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

Bit 14 **USART1EN:** USART1 clock enable

This bit is set and cleared by software.
0: USART1 clock disabled
1: USART1 clock enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1EN:** SPI 1 clock enable

This bit is set and cleared by software.
0: SPI 1 clock disabled
1: SPI 1 clock enabled

Bit 11 **SDIOEN:** SDIO clock enable

This bit is set and cleared by software.
0: SDIO clock disabled
1: SDIO clock enabled

Note: This bit is available in high density devices only.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **ADC1EN:** ADC1 interface clock enable

This bit is set and cleared by software.
0: ADC1 interface disabled
1: ADC1 interface clock enabled

Bits 8:5 Reserved, must be kept at reset value.

Bit 4 **TIM11EN:** TIM11 timer clock enable

This bit is set and cleared by software.
0: TIM11 timer clock disabled
1: TIM11 timer clock enabled

Bit 3 **TIM10EN:** TIM10 timer clock enable

This bit is set and cleared by software.

0: TIM10 timer clock disabled

1: TIM10 timer clock enabled

Bit 2 **TIM9EN:** TIM9 timer clock enable

This bit is set and cleared by software.

0: TIM9 timer clock disabled

1: TIM9 timer clock enabled

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGEN:** System configuration controller clock enable

This bit is set and cleared by software.

0: System configuration controller clock disabled

1: System configuration controller clock enabled

5.3.10 APB1 peripheral clock enable register (RCC_APB1ENR)

Address: 0x24

Reset value: 0x0000 0000

Access: word, half-word and byte access

No wait state, except if the access occurs while an access to a peripheral on APB1 domain is on going. In this case, wait states are inserted until this access to APB1 peripheral is finished.

Note: When the peripheral clock is not active, the peripheral register values may not be readable by software and the returned value is always 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP EN	Res.	DAC EN	PWR EN	Reserved				USB EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART3 EN	USART2 EN	Res.
rw		rw	rw					rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 EN	SPI2 EN	Reserved		WWD GEN	Res.	LCD EN	Reserved			TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rw		rw				rw	rw	rw	rw	rw	rw

Bit 31 **COMPEN:** COMP interface clock enable

This bit is set and cleared by software.
0: COMP interface clock disabled
1: COMP interface clock enable

Bits 30 Reserved, must be kept at reset value.

Bit 29 **DACEN:** DAC interface clock enable

This bit is set and cleared by software.
0: DAC interface clock disabled
1: DAC interface clock enable

Bit 28 **PWREN:** Power interface clock enable

This bit is set and cleared by software.
0: Power interface clock disabled
1: Power interface clock enable

Bits 27:24 Reserved, must be kept at reset value.

Bit 23 **USBEN:** USB clock enable

This bit is set and cleared by software.
0: USB clock disabled
1: USB clock enabled

Bit 22 **I2C2EN:** I^{2}C 2 clock enable

This bit is set and cleared by software.
0: I^{2}C 2 clock disabled
1: I^{2}C 2 clock enabled

Bit 21 **I2C1EN:** I2C 1 clock enable

This bit is set and cleared by software.
0: I^{2}C 1 clock disabled
1: I^{2}C 1 clock enabled

Bit 20 **UART5EN:** UART 5 clock enable

This bit is set and cleared by software.
0: UART 5 clock disabled
1: UART 5 clock enabled

Note: This bit is available in high density devices only.

Bit 19 **UART4EN:** UART 4 clock enable

This bit is set and cleared by software.

Note: 0: UART 4 clock disabled
1: UART 4 clock enabled
This bit is available in high density devices only.

Bit 18 **USART3EN:** USART 3 clock enable

This bit is set and cleared by software.
0: USART 3 clock disabled
1: USART 3 clock enabled

Bit 17 **USART2EN:** USART 2 clock enable

This bit is set and cleared by software.
0: USART 2 clock disabled
1: USART 2 clock enabled

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3EN:** SPI 3 clock enable

This bit is set and cleared by software.
0: SPI 3 clock disabled
1: SPI 3 clock enabled

Note: This bit is available in high and medium+ density devices only.

Bit 14 **SPI2EN:** SPI 2 clock enable

This bit is set and cleared by software.
0: SPI 2 clock disabled
1: SPI 2 clock enabled

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WWDGEN:** Window watchdog clock enable

This bit is set and cleared by software.
0: Window watchdog clock disabled
1: Window watchdog clock enabled

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LCDEN:** LCD clock enable

This bit is set and cleared by software.
0: LCD clock disabled
1: LCD clock enabled

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **TIM7EN:** Timer 7 clock enable

This bit is set and cleared by software.
0: Timer 7 clock disabled
1: Timer 7 clock enabled

Bit 4 **TIM6EN:** Timer 6 clock enable

This bit is set and cleared by software.
0: Timer 6 clock disabled
1: Timer 6 clock enabled

Bit 3 **TIM5EN:** Timer 5 clock enable

This bit is set and cleared by software.
0: Timer 5 clock disabled
1: Timer 5 clock enabled

Note: This bit is available in high and medium+ density devices only.

Bit 2 **TIM4EN:** Timer 4 clock enable

This bit is set and cleared by software.
0: Timer 4 clock disabled
1: Timer 4 clock enabled

Bit 1 **TIM3EN:** Timer 3 clock enable

This bit is set and cleared by software.
0: Timer 3 clock disabled
1: Timer 3 clock enabled

Bit 0 **TIM2EN:** Timer 2 clock enable

This bit is set and cleared by software.
0: Timer 2 clock disabled
1: Timer 2 clock enabled

5.3.11 AHB peripheral clock enable in low power mode register (RCC_AHBLPENR)

Address offset: 0x28

Reset value: 0x0101 903F

Access: no wait state, word, half-word and byte access

Note: The peripheral clock is enabled in sleep mode only if it previously has been enabled in AHBENR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	FSMC LPEN	Reserved		AES LPEN	Res.	DMA2 LPEN	DMA1 LPEN	Reserved							SRAM LPEN
	rw			rw		rw	rw								rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLITF LPEN	Reserved		CRC LPEN	Reserved				GPIOG LPEN	GPIOF LPEN	GPIOH LPEN	GPIOE LPEN	GPIOD LPEN	GPIOC LPEN	GPIOB LPEN	GPIOA LPEN
rw			rw					rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **FSMCLPEN:** FSMC clock enable during Sleep mode

This bit is set and cleared by software.
0: FSMC clock disabled during Sleep mode
1: FSMC clock enabled during Sleep mode

Note: This bit is available in high density devices only.

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

Bit 27 **AESLPEN:** AES clock enable during Sleep mode

This bit is set and cleared by software.
0: AES clock disabled during Sleep mode
1: AES clock enabled during Sleep mode

Note: This bit is available in STM32L16x devices only.

Bit 26 Reserved, must be kept at reset value.

Bit 25 **DMA2LPEN:** DMA2 clock enable during Sleep mode

This bit is set and cleared by software.
0: DMA2 clock disabled during Sleep mode
1: DMA2 clock enabled during Sleep mode

Note: This bit is available in high and medium+ density devices only.

Bit 24 **DMA1LPEN:** DMA1 clock enable during Sleep mode

This bit is set and cleared by software.
0: DMA1 clock disabled during Sleep mode
1: DMA1 clock enabled during Sleep mode

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **SRAMLPEN:** SRAM clock enable during Sleep mode

This bit is set and cleared by software.
0: SRAM clock disabled during Sleep mode
1: SRAM clock enabled during Sleep mode

Bit 15 **FLITFLPEN:** FLITF clock enable during Sleep mode

This bit can be written only when the Flash memory is in power down mode.
0: FLITF clock disabled during Sleep mode
1: FLITF clock enabled during Sleep mode

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

Bit 12 **CRCLPEN:** CRC clock enable during Sleep mode

This bit is set and cleared by software.
0: CRC clock disabled during Sleep mode
1: CRC clock enabled during Sleep mode

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **GPIOGLPEN:** IO port G clock enable during Sleep mode

This bit is set and cleared by software.
0: IO port G clock disabled during Sleep mode
1: IO port G clock enabled during Sleep mode

Note: This bit is available in high and medium+ density devices only.

Bit 6 **GPIOFLPEN:** IO port F clock enable during Sleep mode

This bit is set and cleared by software.
0: IO port F clock disabled during Sleep mode
1: IO port F clock enabled during Sleep mode

Note: This bit is available in high and medium+ density devices only.

Bit 5 **GPIOHLPEN:** IO port H clock enable during Sleep mode

This bit is set and cleared by software.
0: IO port H clock disabled during Sleep mode
1: IO port H clock enabled during Sleep mode

Bit 4 **GPIOELPEN:** IO port E clock enable during Sleep mode

This bit is set and cleared by software.
0: IO port E clock disabled during Sleep mode
1: IO port E clock enabled during Sleep mode

Bit 3 **GPIODLPEN:** IO port D clock enable during Sleep mode

This bit is set and cleared by software.
0: IO port D clock disabled during Sleep mode
1: IO port D clock enabled during Sleep mode

Bit 2 **GPIOCLPEN:** IO port C clock enable during Sleep mode

This bit is set and cleared by software.
0: IO port C clock disabled during Sleep mode
1: IO port C clock enabled during Sleep mode

Bit 1 **GPIOBLPEN:** IO port B clock enable during Sleep mode

This bit is set and cleared by software.
0: IO port B clock disabled during Sleep mode
1: IO port B clock enabled during Sleep mode

Bit 0 **GPIOALPEN:** IO port A clock enable during Sleep mode

This bit is set and cleared by software.
0: IO port A clock disabled during Sleep mode
1: IO port A clock enabled during Sleep mode

5.3.12 APB2 peripheral clock enable in low power mode register (RCC_APB2LPENR)

Address: 0x2C

Reset value: 0x0000 521D

Access: no wait states, word, half-word and byte access

Note: The peripheral clock is enabled in sleep mode only if it's previously has been enabled in APB2ENR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1 LPEN	Res.	SPI1 LPEN	SDIO LPEN	Res.	ADC1 LPEN	Reserved				TIM11 LPEN	TIM10 LPEN	TIM9 LPEN	Res.	SYSCF GLPEN
	rw		rw	rw		rw					rw	rw	rw		rw

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

Bit 14 **USART1LPEN:** USART1 clock enable during Sleep mode

This bit is set and cleared by software.
0: USART1 clock disabled during Sleep mode
1: USART1 clock enabled during Sleep mode

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI1LPEN:** SPI 1 clock enable during Sleep mode

This bit is set and cleared by software.
0: SPI 1 clock disabled during Sleep mode
1: SPI 1 clock enabled during Sleep mode

Bit 11 **SDIOLPEN:** SDIO clock enable during Sleep mode

This bit is set and cleared by software.
0: SDIO clock disabled during Sleep mode
1: SDIO clock enabled during Sleep mode

Note: This bit is available in high density devices only.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **ADC1LPEN:** ADC1 interface clock enable during Sleep mode

This bit is set and cleared by software.
0: ADC1 interface disabled during Sleep mode
1: ADC1 interface clock enabled during Sleep mode

Bits 8:5 Reserved, must be kept at reset value.

Bit 4 **TIM11LPEN:** TIM11 timer clock enable during Sleep mode

This bit is set and cleared by software.
0: TIM11 timer clock disabled during Sleep mode
1: TIM11 timer clock enabled during Sleep mode

Bit 3 **TIM10LPEN:** TIM10 timer clock enable during Sleep mode

This bit is set and cleared by software.
0: TIM10 timer clock disabled during Sleep mode
1: TIM10 timer clock enabled during Sleep mode

Bit 2 **TIM9LPEN:** TIM9 timer clock enable during Sleep mode

This bit is set and cleared by software.
0: TIM9 timer clock disabled during Sleep mode
1: TIM9 timer clock enabled during Sleep mode

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGLPEN:** System configuration controller clock enable during Sleep mode

This bit is set and cleared by software.
0: System configuration controller clock disabled during Sleep mode
1: System configuration controller clock enabled during Sleep mode

5.3.13 APB1 peripheral clock enable in low power mode register (RCC_APB1LPENR)

Address: 0x30

Reset value: 0xB0E6 4A37

Access: no wait state, word, half-word and byte access

Note: The peripheral clock is enabled in sleep mode only if it's previously has been enabled in APB1ENR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP LPEN	Res.	DAC LPEN	PWR LPEN	Reserved				USB LPEN	I2C2 LPEN	I2C1 LPEN	UART5 LPEN	UART4 LPEN	USART3 LPEN	USART2 LPEN	Res.
rw		rw	rw					rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 LPEN	SPI2 LPEN	Reserved		WWDGLPEN	Res.	LCD LPEN	Reserved			TIM7 LPEN	TIM6 LPEN	TIM5 LPEN	TIM4 LPEN	TIM3 LPEN	TIM2 LPEN
rw	rw			rw		rw				rw	rw	rw	rw	rw	rw

Bit 31 **COMPLPEN:** COMP interface clock enable during Sleep mode
This bit is set and cleared by software.
0: COMP interface clock disabled during Sleep mode
1: COMP interface clock enable during Sleep mode

Bit 30 Reserved, must be kept at reset value.

Bit 29 **DACLPEN:** DAC interface clock enable during Sleep mode
This bit is set and cleared by software.
0: DAC interface clock disabled during Sleep mode
1: DAC interface clock enable during Sleep mode

Bit 28 **PWRLPEN:** Power interface clock enable during Sleep mode
This bit is set and cleared by software.
0: Power interface clock disabled during Sleep mode
1: Power interface clock enable during Sleep mode

Bits 27:24 Reserved, must be kept at reset value.

Bit 23 **USBLPEN:** USB clock enable during Sleep mode
This bit is set and cleared by software.
0: USB clock disabled during Sleep mode
1: USB clock enabled during Sleep mode

Bit 22 **I2C2LPEN:** I^2C 2 clock enable during Sleep mode
This bit is set and cleared by software.
0: I^2C 2 clock disabled during Sleep mode
1: I^2C 2 clock enabled during Sleep mode

Bit 21 **I2C1LPEN:** I^2C 1 clock enable during Sleep mode
This bit is set and cleared by software.
0: I^2C 1 clock disabled during Sleep mode
1: I^2C 1 clock enabled during Sleep mode

Bit 20 **UART5LPEN:** USART 5 clock enable during Sleep mode

This bit is set and cleared by software.

0: UART 5 clock disabled during Sleep mode

1: UART 5 clock enabled during Sleep mode

Note: This bit is available in high density devices only.

Bit 19 **UART4LPEN:** USART 4 clock enable during Sleep mode

This bit is set and cleared by software.

0: UART 4 clock disabled during Sleep mode

1: UART 4 clock enabled during Sleep mode

Note: This bit is available in high density devices only.

Bit 18 **USART3LPEN:** USART 3 clock enable during Sleep mode

This bit is set and cleared by software.

0: USART 3 clock disabled during Sleep mode

1: USART 3 clock enabled during Sleep mode

Bit 17 **USART2LPEN:** USART 2 clock enable during Sleep mode

This bit is set and cleared by software.

0: USART 2 clock disabled during Sleep mode

1: USART 2 clock enabled during Sleep mode

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3LPEN:** SPI 3 clock enable during Sleep mode

This bit is set and cleared by software.

0: SPI 3 clock disabled during Sleep mode

1: SPI 3 clock enabled during Sleep mode

Note: This bit is available in high and medium+ density devices only.

Bit 14 **SPI2LPEN:** SPI 2 clock enable during Sleep mode

This bit is set and cleared by software.

0: SPI 2 clock disabled during Sleep mode

1: SPI 2 clock enabled during Sleep mode

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WWDGLPEN:** Window watchdog clock enable during Sleep mode

This bit is set and cleared by software.

0: Window watchdog clock disabled during Sleep mode

1: Window watchdog clock enabled during Sleep mode

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LCDLPEN:** LCD clock enable during Sleep mode

This bit is set and cleared by software.

0: LCD clock disabled during Sleep mode

1: LCD clock enabled during Sleep mode

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **TIM7LPEN:** Timer 7 clock enable during Sleep mode

This bit is set and cleared by software.

0: Timer 7 clock disabled during Sleep mode

1: Timer 7 clock enabled during Sleep mode

Bit 4 **TIM6LPEN:** Timer 6 clock enable during Sleep mode

This bit is set and cleared by software.

0: Timer 6 clock disabled during Sleep mode

1: Timer 6 clock enabled during Sleep mode

Bit 3 **TIM5LPEN:** Timer 5 clock enable during Sleep mode

This bit is set and cleared by software.

0: Timer 5 clock disabled during Sleep mode

1: Timer 5 clock enabled during Sleep mode

Note: This bit is available in high density and medium+ devices only.

Bit 2 **TIM4LPEN:** Timer 4 clock enable during Sleep mode

This bit is set and cleared by software.

0: Timer 4 clock disabled during Sleep mode

1: Timer 4 clock enabled during Sleep mode

Bit 1 **TIM3LPEN:** Timer 3 clock enable during Sleep mode

This bit is set and cleared by software.

0: Timer 3 clock disabled during Sleep mode

1: Timer 3 clock enabled during Sleep mode

Bit 0 **TIM2LPEN:** Timer 2 clock enable during Sleep mode

This bit is set and cleared by software.

0: Timer 2 clock disabled during Sleep mode

1: Timer 2 clock enabled during Sleep mode

5.3.14 Control/status register (RCC_CSR)

Address: 0x34

Power-on reset value: 0x0C00 0000,

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 LSEON, LSEBYP, RTCSEL and RTCEN bits in the RCC control and status register (RCC_CSR) are in the RTC domain. As these bits are write protected after reset, the DBP bit in the Power control register (PWR_CR) has to be set to be able to modify them. Refer to Section RTC and RTC backup registers for further information. These bits are only reset after a RTC domain reset (see RTC and backup registers reset). Any internal or external reset does not have any effect on them.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PORRSTF	PINRSTF	OBLRSTF	RMVF	RTCRST	RTCEN	Reserved				RTCSEL[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw					rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			LSECSSD	LSECSSON	LSEBYP	LSERDY	LSEON	Reserved						LSIRDY	LSION
			r	rw	rw	r	rw							r	rw

Bit 31 **LPWRRSTF:** Low-power reset flag

This bit is set by hardware when a Low-power management reset occurs.
It is cleared by writing to the RMVF bit, or by a POR.
0: No Low-power management reset occurred
1: Low-power management reset occurred
For further information on Low-power management reset, refer to *Low-power management reset*.

Bit 30 **WWDGRSTF:** Window watchdog reset flag

This bit is set by hardware when a window watchdog reset occurs.
It is cleared by writing to the RMVF bit, or by a POR.
0: No window watchdog reset occurred
1: Window watchdog reset occurred

Bit 29 **IWDGRSTF**: Independent watchdog reset flag

This bit is set by hardware when an independent watchdog reset from V_{DD} domain occurs.
It is cleared by writing to the RMVF bit, or by a POR.
0: No watchdog reset occurred
1: Watchdog reset occurred

Bit 28 **SFTRSTF:** Software reset flag

This bit is set by hardware when a software reset occurs.
It is cleared by writing to the RMVF bit, or by a POR.
0: No software reset occurred
1: Software reset occurred

Bit 27 **PORRSTF:** POR/PDR reset flag

This bit is set by hardware when a POR/PDR reset occurs.
It is cleared by writing to the RMVF bit.
0: No POR/PDR reset occurred
1: POR/PDR reset occurred

Bit 26 **PINRSTF:** PIN reset flag

This bit is set by hardware when a reset from the NRST pin occurs.
It is cleared by writing to the RMVF bit, or by a POR.
0: No reset from NRST pin occurred
1: Reset from NRST pin occurred

Bit 25 **OBLRSTF** Options bytes loading reset flag

This bit is set by hardware when an OBL reset occurs.
It is cleared by writing to the RMVF bit, or by a POR.
0: No OBL reset occurred
1: OBL reset occurred

Bit 24 **RMVF:** Remove reset flag

This bit is set by software to clear the reset flags.
0: No effect
1: Clear the reset flags

Bit 23 **RTCRST:** RTC software reset

This bit is set and cleared by software.
0: Reset not activated
1: Resets the RTC peripheral, its clock source selection and the backup registers.

Bit 22 **RTCEN:** RTC clock enable

This bit is set and cleared by software.

It is reset by setting the RTCRST bit or by a POR.

0: RTC clock disabled

1: RTC clock enabled

Bits 21:18 Reserved, must be kept at reset value.

Bits 17:16 **RTCSEL[1:0]:** RTC and LCD clock source selection

These bits are set by software to select the clock source for the RTC.

Once the RTC and LCD clock source has been selected it cannot be switched until RTCRST is set or a Power On Reset occurred. The only exception is if the LSE oscillator clock was selected, if the LSE clock stops and it is detected by the CSS, in that case the clock can be switched.

00: No clock

01: LSE oscillator clock used as RTC/LCD clock

10: LSI oscillator clock used as RTC/LCD clock

11: HSE oscillator clock divided by a programmable prescaler (selection through the RTCPRE[1:0] bits in the RCC clock control register (RCC_CR)) used as the RTC/LCD clock

If the LSE or LSI is used as RTC clock source, the RTC continues to work in Stop and Standby low power modes, and can be used as wake-up source. However, when the HSE clock is used as RTC clock source, the RTC cannot be used in Stop and Standby low power modes.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **LSECSSD:** CSS on LSE failure Detection

Set by hardware to indicate when a failure has been detected by the Clock Security System on the external 32 kHz oscillator (LSE).

Reset by power on reset and RTC software reset (RTCRST bit).

0: No failure detected on LSE (32 kHz oscillator)

1: Failure detected on LSE (32 kHz oscillator)

Note: This bit is available in high and medium+ density devices only.

Bit 11 **LSECSSON** CSS on LSE enable

Set by software to enable the Clock Security System on LSE (32 kHz oscillator).

LSECSSON must be enabled after the LSE and LSI oscillators are enabled (LSEON and LSION bits enabled) and ready (LSERDY and LSIRDY flags set by hardware), and after the RTCSEL bit is selected.

Once enabled this bit cannot be disabled, except after an LSE failure detection (LSECSSD =1). In that case the software MUST disable the LSECSSON bit.

Reset by power on reset and RTC software reset (RTCRST bit).

0: CSS on LSE (32 kHz oscillator) OFF

1: CSS on LSE (32 kHz oscillator) ON

Note: This bit is available in high and medium+ density devices only.

Bit 10 **LSEBYP:** External low-speed oscillator bypass

This bit is set and cleared by software to bypass oscillator in debug mode. This bit can be written only when the LSE oscillator is disabled.

It is reset by setting the RTCRST bit or by a POR.

0: LSE oscillator not bypassed

1: LSE oscillator bypassed

Bit 9 **LSERDY:** External low-speed oscillator ready

This bit is set and cleared by hardware to indicate when the LSE oscillator is stable. After the LSEON bit is cleared, LSERDY goes low after 6 LSE oscillator clock cycles.
It is reset by setting the RTCRST bit or by a POR.
0: External 32 kHz oscillator not ready
1: External 32 kHz oscillator ready

Bit 8 **LSEON:** External low-speed oscillator enable

This bit is set and cleared by software.
It is reset by setting the RTCRST bit or by a POR.
0: LSE oscillator OFF
1:LSE oscillator ON

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

Bit 1 **LSIRDY:** Internal low-speed oscillator ready

This bit is set and cleared by hardware to indicate when the LSI oscillator is stable. After the LSION bit is cleared, LSIRDY goes low after 3 LSI oscillator clock cycles.
This bit is reset by system reset.
0: LSI oscillator not ready
1: LSI oscillator ready

Bit 0 **LSION:** Internal low-speed oscillator enable

This bit is set and cleared by software.
It is reset by system reset.
0: LSI oscillator OFF
1: LSI oscillator ON

5.3.15 RCC register map

The following table gives the RCC register map and the reset values.

Table 22. RCC register map and reset values

Off-set	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**RCC_CR**	Reserved	RTCPRE1	RTCPRE0	CSSON	Reserved		PLL RDY	PLL ON	Reserved					HSEBYP	HSERDY	HSEON	Reserved						MSIRDY	MSION	Reserved						HSIRDY	HSION
	Reset value		0	0	0			0	0						0	0	0							1	1							0	0
0x04	**RCC_ICSCR**	MSITRIM[7:0]								MSICAL[7:0]								MSIRANGE[2:0]			HSITRIM[4:0]					HSICAL[7:0]							
	Reset value	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x	1	0	1	1	0	0	0	0	x	x	x	x	x	x	x	x
0x08	**RCC_CFGR**	Reserved	MCOPRE [2:0]			Reserved	MCOSEL [2:0]			PLL DIV [1:0]		PLLMUL[3:0]				Reserved	PLLSRC	Reserved		PPRE2 [2:0]			PPRE1 [2:0]			HPRE[3:0]				SWS [1:0]		SW [1:0]	
	Reset value		0	0	0		0	0	0	0	0	0	0	0	0		0			0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22. RCC register map and reset values (continued)

Off-set	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C	**RCC_CIR**	Reserved								CSSC	LSECSSC	MSIRDYC	PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC	Reserved	LSECSSIE	MSIRDYIE	PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	Reserved	MSIRDYF	PLLRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
	Reset value									0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0	0
0x10	**RCC_AHBRSTR**	Reserved	FSMCRST	Reserved		AESRST	Reserved	DMA2RST	DMA1RST	Reserved								FLITFRST	Reserved		CRCRST	Reserved				GPIOGRST	GPIOFRST	GPIOHRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
	Reset value		0					0	0									0			0					0	0	0	0	0	0	0	0
0x14	**RCC_APB2RSTR**	Reserved																	USART1RST	Reserved	SPI1RST	SDIORST	Reserved	ADC1RST	Reserved				TM11RST	TM10RST	TIM9RST	Reserved	SYSCFGRST
	Reset value																		0		0	0		0					0	0	0		0
0x18	**RCC_APB1RSTR**	COMPRST	Reserved	DACRST	PWRRST	Reserved				USBRST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	Reserved	SPI3RST	SPI2RST	Reserved		WWDRST	Reserved	LCDRST	Reserved			TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
	Reset value	0		0	0					0	0	0	0	0	0	0		0	0			0		0				0	0	0	0	0	0
0x1C	**RCC_AHBENR**	Reserved	FSMCEN	Reserved		AESEN	Reserved	DMA2EN	DMA1EN	Reserved								FLITFEN	Reserved		CRCEN	Reserved				GPIOPGEN	GPIOPFEN	GPIOPHEN	GPIOPEEN	GPIOPDEN	GPIOPCEN	GPIOPBEN	GPIOPAEN
	Reset value		0			0		0	0									1			0					0	0	0	0	0	0	0	0
0x20	**RCC_APB2ENR**	Reserved																	USART1EN	Reserved	SPI1EN	SDIOEN	Reserved	ADC1EN	Reserved				TIM11EN	TIM10EN	TIM9EN	Reserved	SYSCFGEN
	Reset value																		0		0	0		0					0	0	0		0
0x24	**RCC_APB1ENR**	COMPEN	Reserved	DACEN	PWREN	Reserved				USBEN	I2C2EN	I2C1EN	USART5EN	USART4EN	USART3EN	USART2EN	Reserved	SPI3EN	SPI2EN	Reserved		WWDGEN	Reserved	LCDEN	Res.			TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
	Reset value	0		0	0					0	0	0	0	0	0	0		0	0			0		0				0	0	0	0	0	0
0x28	**RCC_AHBLPENR**	Reserved	FSMCLPEN	Reserved		AESLPEN	Reserved	DMA2LPEN	DMA1LPEN	Reserved							SRAMLPEN	FLITFLPEN	Reserved		CRCLPEN	Reserved				GPIOGLPEN	GPIOFLPEN	GPIOHLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
	Reset value		1			1		1	1								1	1			1					1	1	1	1	1	1	1	1

Table 22. RCC register map and reset values (continued)

Off-set	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	**RCC_APB2LPENR**	Reserved																	USART1LPEN	Reserved	SPI1LPEN	SDIOLPEN	Reserved	ADC1LPEN	Reserved				TIM11LPEN	TIM10LPEN	TIM9LPEN	Reserved	SYSCFGLPEN
	Reset value																		1		1	1		1					1	1	1		1
0x30	**RCC_APB1LPENR**	COMPLPEN	Reserved	DACLPEN	PWRLPEN	Reserved				USBLPEN	I2C2LPEN	I2C1LPEN	USART5LPEN	USART4LPEN	USART3LPEN	USART2LPEN	Reserved	SPI3LPEN	SPI2LPEN	Reserved		WWDGLPEN	Reserved	LCDLPEN	Res.			TIM7LPEN	TIM6LPEN	TIM5LPEN	TIM4LPEN	TIM3LPEN	TIM2LPEN
	Reset value	1		1	1					1	1	1	1	1	1	1		1	1			1		1				1	1	1	1	1	1
0x034	**RCC_CSR**	LPWRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PORRSTF	PINRSTF	OBLRSTF	RMVF	RTCRST	RTCEN	Reserved				RTC SEL [1:0]		Reserve d			LSECSSD	LSECSSON	LSEBYP	LSERDY	LSEON	Reserved						LSIRDY	LSION
	Reset value	0	0	0	0	1	1	0	0	0	0					0	0				0	0	0	0								0	0

Refer to *Table 2 on page 42* for the register boundary addresses.

6 General-purpose I/Os (GPIO)

6.1 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 32-bit set/reset register (GPIOx_BSRR), a 32-bit locking register (GPIOx_LCKR) and two 32-bit alternate function selection register (GPIOx_AFRH and GPIOx_AFRL).

6.2 GPIO main features

- Up to 16 I/Os under control
- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_BSRR) for bitwise write access to GPIOx_ODR
- Locking mechanism (GPIOx_LCKR) provided to freeze the I/O configuration
- Analog function
- Alternate function input/output selection registers (at most 16 AFs per I/O)
- 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

6.3 GPIO functional description

Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability

Each I/O port bit is freely programmable, however the I/O port registers have to be accessed as 32-bit words, half-words or bytes. The purpose of the GPIOx_BSRR register is to allow atomic read/modify accesses to any of the GPIO registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.

Figure 15 and *Figure 16* show the basic structures of a standard and a 5 V tolerant I/O port bit, respectively. *Table 26* gives the possible port bit configurations.

Figure 15. Basic structure of a standard I/O port bit

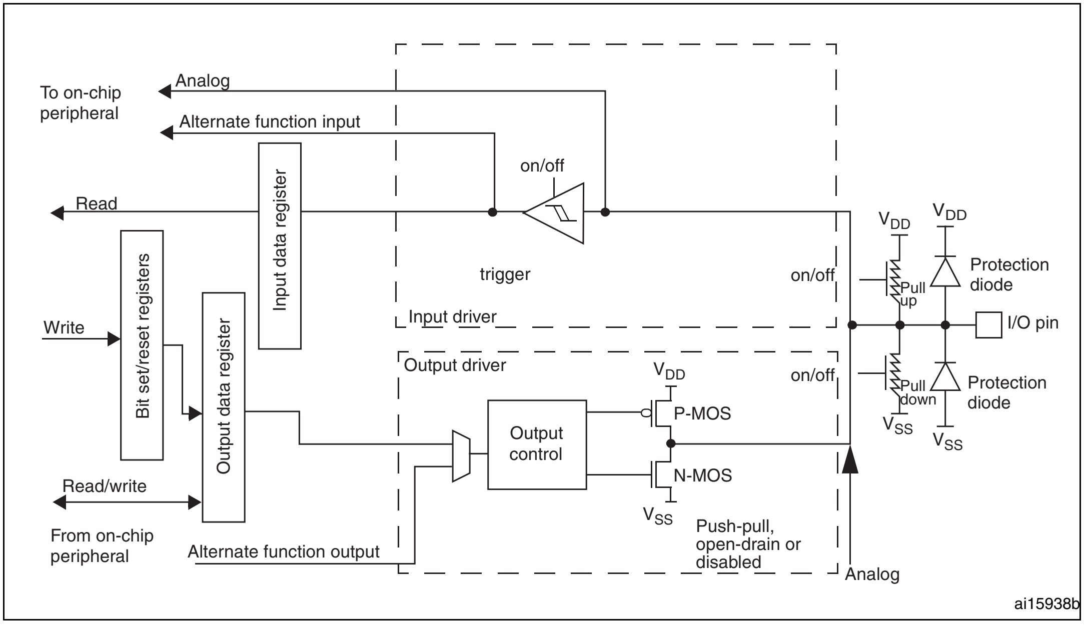
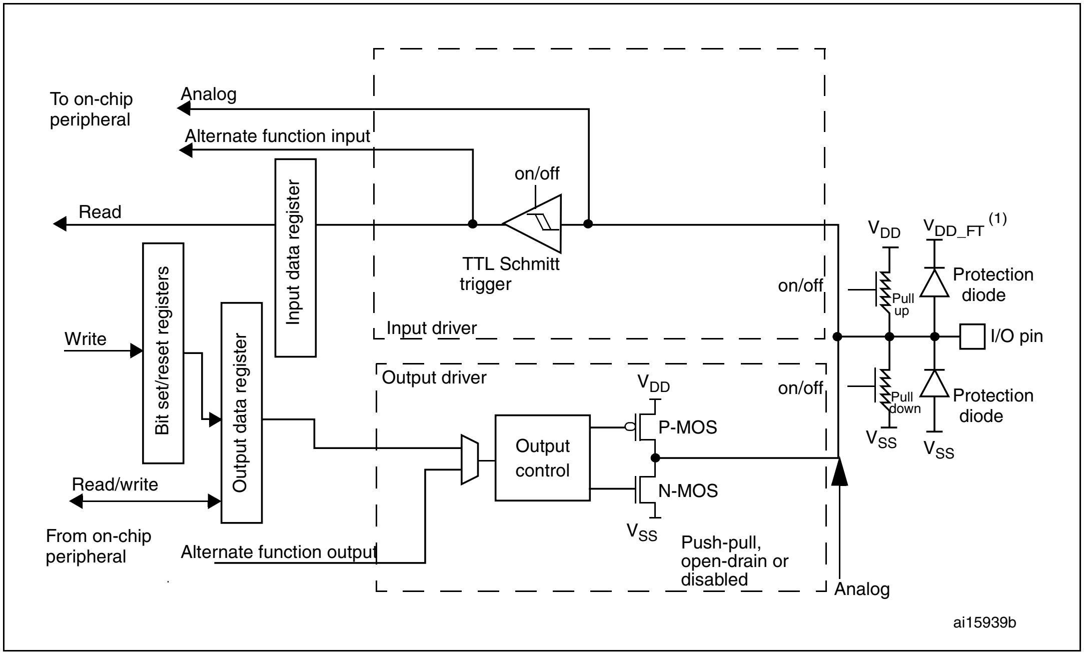


Figure 16. Basic structure of a five-volt tolerant I/O port bit



1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

Table 23. Port bit configuration table[1]

MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [B:A]		PUPDR(i) [1:0]		I/O configuration	
01	0	SPEED [B:A]		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 [B:A]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0		
	x	x	x	1	1		

[1] GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

6.3.1 General-purpose I/O (GPIO)

During and just after reset, the alternate functions are not active and the I/O ports are configured in input floating mode.

The debug pins are in AF pull-up/pull-down after reset:

- PA15: JTDI in pull-up
- PA14: JTCK/SWCLK in pull-down
- PA13: JTMS/SWDAT in pull-up
- PB4: NJTRST in pull-up
- PB3: JTDO in floating state

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 N-MOS is activated when 0 is output).

The input data register (GPIOx_IDR) captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors, which can be activated or not depending on the value in the GPIOx_PUPDR register.

6.3.2 I/O pin multiplexer and mapping

The microcontroller I/O pins are connected to onboard peripherals/modules through a multiplexer that allows only one peripheral's alternate function (AF) connected to an I/O pin at a time. In this way, there can be no conflict between peripherals sharing the same I/O pin.

Each I/O pin has a multiplexer with sixteen alternate function inputs (AF0 to AF15) that can be configured through the GPIOx_AFRL (for pin 0 to 7) and GPIOx_AFRH (for pin 8 to 15) registers:

- After reset all I/Os are connected to the system's alternate function 0 (AF0)
- The peripherals' alternate functions are mapped from AF1 to AF14
- Cortex™-M3 EVENTOUT is mapped on AF15

This structure is shown in *Figure 17* below.

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages.

To use an I/O in a given configuration, proceed as follows:

- **System function**

  Connect the I/O to AF0 and configure it depending on the function used:
  - JTAG/SWD, after each device reset these pins are assigned as dedicated pins immediately usable by the debugger host (not controlled by the GPIO controller)
  - RTC_AF1: refer to *Table 25: RTC_AF1 pin* for more details about this pin configuration
  - RTC_50Hz: this pin should be configured in Input floating mode
  - MCO: this pin has to be configured in alternate function mode.

Note: You can disable some or all of the JTAG/SWD pins and so release the associated pins for GPIO usage.

For more details please refer to *Section 5.2.13: Clock-out capability*.

Table 24. Flexible SWJ-DP pin assignment

Available debug ports	SWJ I/O pin assigned				
	PA13 / JTMS/ SWDIO	PA14 / JTCK/ SWCLK	PA15 / JTDI	PB3 / JTDO	PB4/ NJTRST
Full SWJ (JTAG-DP + SW-DP) - Reset state	X	X	X	X	X
Full SWJ (JTAG-DP + SW-DP) but without NJTRST	X	X	X	X	
JTAG-DP Disabled and SW-DP Enabled	X	X			
JTAG-DP Disabled and SW-DP Disabled	Released				

- **GPIO**

  Configure the desired I/O as output, input or analog in the GPIOx_MODER register.
- **Peripheral alternate function**

  For the ADC and DAC, configure the desired I/O as analog in the GPIOx_MODER register.

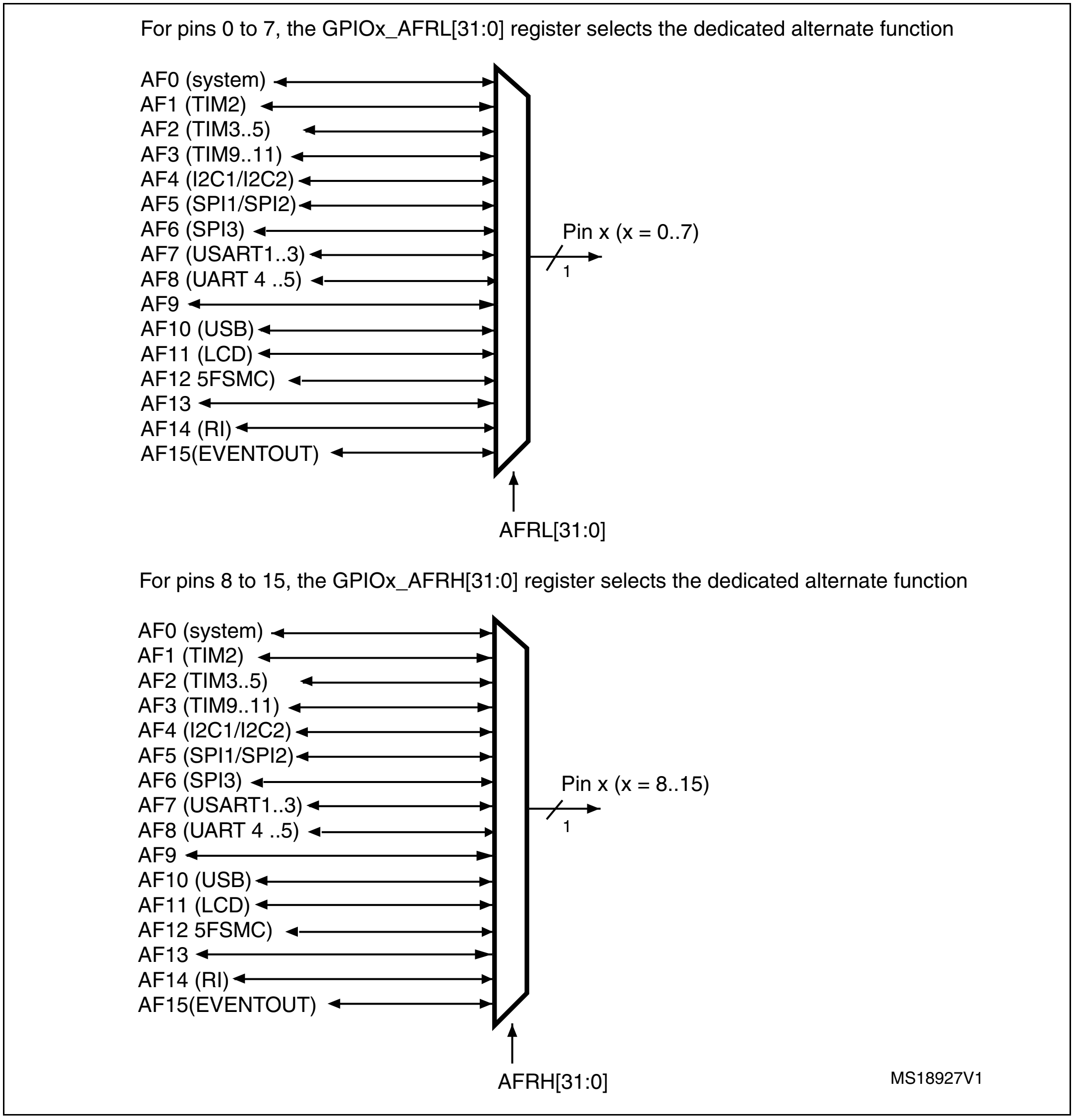
  For other peripherals:
  - Configure the desired I/O as an alternate function in the GPIOx_MODER register
  - Select the type, pull-up/pull-down and output speed via the GPIOx_OTYPER, GPIOx_PUPDR and GPIOx_OSPEEDER registers, respectively
  - Connect the I/O to the desired AFx in the GPIOx_AFRL or GPIOx_AFRH register
- **EVENTOUT**

  Configure the I/O pin used to output the Cortex™-M3 EVENTOUT signal by connecting it to AF15

Note: EVENTOUT is not mapped onto the following I/O pins: PH0, PH1 and PH2.

Please refer to the “Alternate function mapping” table in the datasheets for the detailed mapping of the system and peripherals’ alternate function I/O pins.

Figure 17. Selecting an alternate function



6.3.3 I/O port control registers

Each of the GPIOs 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 direction (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 I/O speed pins are directly connected to the corresponding GPIOx_OSPEEDR register bits whatever the I/O direction). The GPIOx_PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.

6.3.4 I/O port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (GPIOx_IDR and GPIOx_ODR). GPIOx_ODR stores the data to be output, it is read/write accessible. The data input through the I/O are stored into the input data register (GPIOx_IDR), a read-only register.

See *Section 6.4.5: GPIO port input data register (GPIOx_IDR) (x = A..H)* and *Section 6.4.6: GPIO port output data register (GPIOx_ODR) (x = A..H)* for the register descriptions.

6.3.5 I/O data bitwise handling

The bit set reset register (GPIOx_BSRR) is a 32-bit register which allows the application to set and reset each individual bit in the output data register (GPIOx_ODR). The bit set reset register has twice the size of GPIOx_ODR.

To each bit in GPIOx_ODR, correspond two control bits in GPIOx_BSRR: BSRR(i) and BSRR(i+SIZE). When written to 1, bit BSRR(i) **sets** the corresponding ODR(i) bit. When written to 1, bit BSRR(i+SIZE) **resets** the ODR(i) corresponding bit.

Writing any bit to 0 in GPIOx_BSRR does not have any effect on the corresponding bit in GPIOx_ODR. If there is an attempt to both set and reset a bit in GPIOx_BSRR, the set action takes priority.

Using the GPIOx_BSRR register to change the values of individual bits in GPIOx_ODR is a “one-shot” effect that does not lock the GPIOx_ODR bits. The GPIOx_ODR bits can always be accessed directly. The GPIOx_BSRR register provides a way of performing atomic bitwise handling.

There is no need for the software to disable interrupts when programming the GPIOx_ODR at bit level: it is possible to modify one or more bits in a single atomic AHB write access.

6.3.6 GPIO locking mechanism

It is possible to freeze the GPIO control registers by applying a specific write sequence to the GPIOx_LCKR register. The frozen registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

To write the GPIOx_LCKR register, a specific write / read sequence has to be applied. When the right LOCK sequence is applied to bit 16 in this register, the value of LCKR[15:0] is used to lock the configuration of the I/Os (during the write sequence the LCKR[15:0] value must be the same). When the LOCK sequence has been applied to a port bit, the value of the port bit can no longer be modified until the next reset. Each GPIOx_LCKR bit freezes the corresponding bit in the control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH).

The LOCK sequence (refer to *Section 6.4.8: GPIO port configuration lock register (GPIOx_LCKR) (x = A..H)*) can only be performed using a word (32-bit long) access to the GPIOx_LCKR register due to the fact that GPIOx_LCKR bit 16 has to be set at the same time as the [15:0] bits.

For more details please refer to LCKR register description in *Section 6.4.8: GPIO port configuration lock register (GPIOx_LCKR) (x = A..H)*.

6.3.7 I/O alternate function input/output

Two registers are provided to select one out of the sixteen alternate function inputs/outputs available for each I/O. With these registers, you can connect an alternate function to some other pin as required by your application.
This means that a number of possible peripheral functions are multiplexed on each GPIO

using the GPIOx_AFRL and GPIOx_AFRH alternate function registers. The application can thus select any one of the possible functions for each I/O. The AF selection signal being common to the alternate function input and alternate function output, a single channel is selected for the alternate function input/output of one I/O.

To know which functions are multiplexed on each GPIO pin, refer to the datasheets.

Note: The application is allowed to select one of the possible peripheral functions for each I/O at a time.

6.3.8 External interrupt/wakeup lines

All ports have external interrupt capability. To use external interrupt lines, the port must be configured in input mode, refer to *Section 9.2: External interrupt/event controller (EXTI)* and *Section 9.2.3: Wakeup event management*.

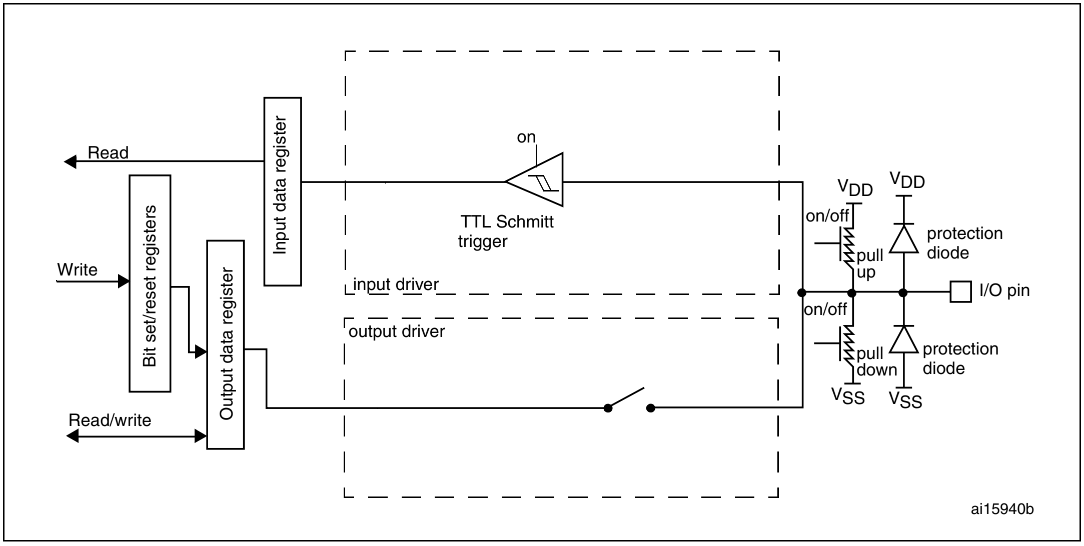
6.3.9 Input configuration

When the I/O port is programmed as Input:

- the output buffer is disabled
- the Schmitt trigger input is activated
- the pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register provides the I/O State

Figure 18 shows the input configuration of the I/O port bit.

Figure 18. Input floating/pull up/pull down configurations



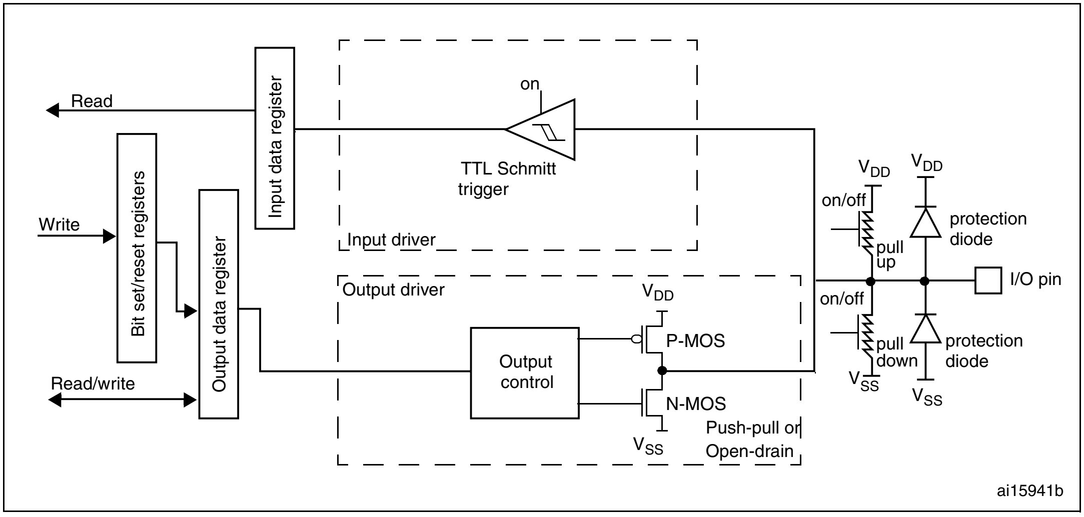
6.3.10 Output configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
  - Open drain mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register leaves the port in Hi-Z (the P-MOS is never activated)
  - Push-pull mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register activates the P-MOS
- The Schmitt trigger input is activated
- The 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
- A read access to the output data register gets the last written value

Figure 19 shows the output configuration of the I/O port bit.

Figure 19. Output configuration



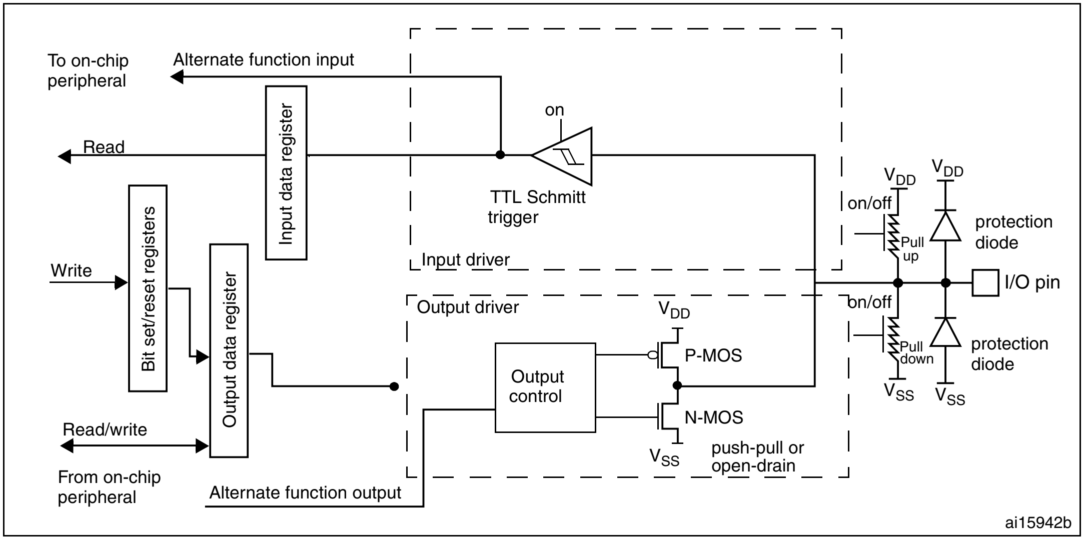
6.3.11 Alternate function configuration

When the I/O port is programmed as alternate function:

- The output buffer can be configured as open-drain or push-pull
- The output buffer is driven by the signal 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 20 shows the Alternate function configuration of the I/O port bit.

Figure 20. Alternate function configuration



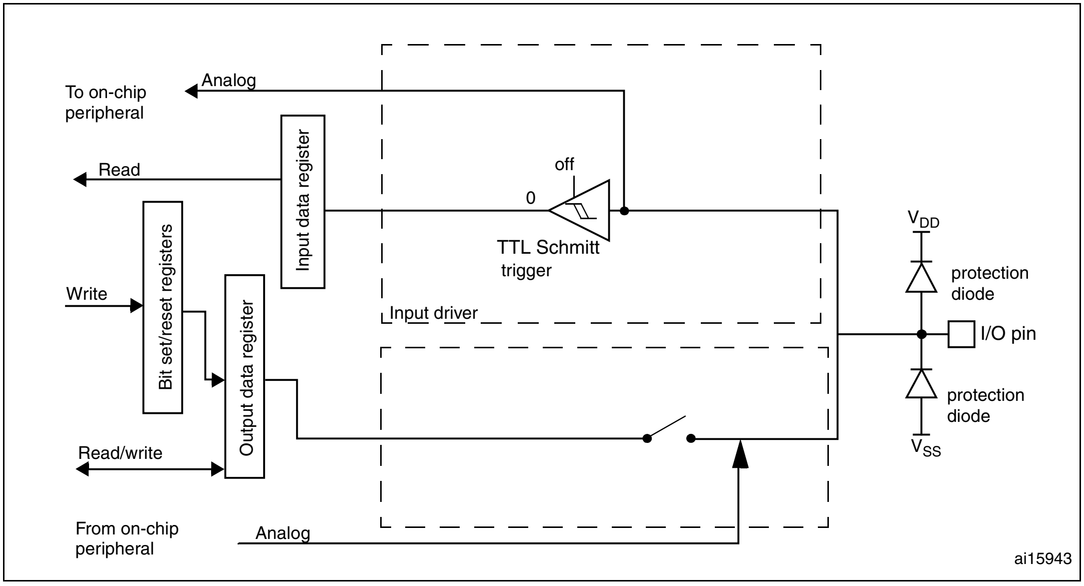
6.3.12 Analog configuration

When the I/O port is programmed as analog configuration:

- The output buffer is disabled
- The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0).
- The weak pull-up and pull-down resistors are disabled
- Read access to the input data register gets the value “0”

Figure 21 shows the high-impedance, analog-input configuration of the I/O port bit.

Figure 21. High impedance-analog configuration



6.3.13 Using the OSC32_IN/OSC32_OUT pins as GPIO PC14/PC15 port pins

The LSE oscillator pins OSC32_IN and OSC32_OUT can be used as general-purpose PC14 and PC15 I/Os, respectively, when the LSE oscillator is off. The PC14 and PC15 I/Os are only configured as LSE oscillator pins OSC32_IN and OSC32_OUT when the LSE oscillator is ON. This is done by setting the LSEON bit in the RCC_BDCR register. The LSE has priority over the GPIO function.

Note: The PC14/PC15 GPIO functionality is lost when the V_{CORE} domain is powered off (by the device entering the standby mode). In this case the I/Os are set in analog input mode.

6.3.14 Using the OSC_IN/OSC_OUT pins as GPIO PH0/PH1 port pins

The HSE oscillator pins OSC_IN/OSC_OUT can be used as general-purpose PH0/PH1 I/Os, respectively, when the HSE oscillator is OFF. (after reset, the HSE oscillator is off). The PH0/PH1 I/Os are only configured as OSC_IN/OSC_OUT HSE oscillator pins when the HSE oscillator is ON. This is done by setting the HSEON bit in the RCC_CR register. The HSE has priority over the GPIO function.

6.3.15 Selection of RTC_AF1 alternate functions

The STM32L1xxxx features:

- Two GPIO pins, which can be used as wakeup pins (WKUP1 and WKUP3).
- One GPIO pin, which can be used as a wakeup pin (WKUP2), for the detection of a tamper or time-stamp event, or to output RTC AFO_ALARM or AFO_CALIB.

The RTC_AF1 pin (PC13) can be used for the following purposes:

- Wakeup pin 2 (WKUP2): this feature is enabled by setting the EWUP2 in the PWR_CSR register.
- RTC AFO_ALARM output: this output can be RTC Alarm A, RTC Alarm B or RTC Wakeup depending on the OSEL[1:0] bits in the RTC_CR register.
- RTC AFO_CALIB output: this feature is enabled by setting the COE[23] bit in the RTC_CR register.
- RTC AFI_TAMPER1: Tamper event detection
- Time-stamp event detection

The selection of the RTC AFO_ALARM output is performed through the RTC_TAFCR register as follows: ALARMOUTTYPE is used to select whether the RTC AFO_ALARM output is configured in push-pull or open-drain mode.

The output mechanism follows the priority order shown in *Table 25*.

Table 25. RTC_AF1 pin [(1)]

Pin configuration and function	AFO_ALARM enabled	AFO_CALIB enabled	Tamper enabled	Time-stamp enabled	EWUP2 enabled	ALARMOUTTYPE AFO_ALARM configuration
Alarm out output OD	1	0	Don’t care	Don’t care	Don’t care	0
Alarm out output PP	1	0	Don’t care	Don’t care	Don’t care	1

Table 25. RTC_AF1 pin (continued)[1]

Pin configuration and function	AFO_ALARM enabled	AFO_CALIB enabled	Tamper enabled	Time-stamp enabled	EWUP2 enabled	ALARMOUTTYPE AFO_ALARM configuration
Calibration out output PP	0	1	Don't care	Don't care	Don't care	Don't care
TAMPER input floating	0	0	1	0	Don't care	Don't care
TIMESTAMP and TAMPER input floating	0	0	1	1	Don't care	Don't care
TIMESTAMP input floating	0	0	0	1	Don't care	Don't care
Wakeup Pin 2	0	0	0	0	1	Don't care
Standard GPIO	0	0	0	0	0	Don't care

[1] OD: open drain; PP: push-pull.

6.4 GPIO registers

This section gives a detailed description of the GPIO registers.
For a summary of register bits, register address offsets and reset values, refer to *Table 26*.

The peripheral registers have to be accessed by words (32-bit).

6.4.1 GPIO port mode register (GPIOx_MODER) (x = A..H)

Address offset: 0x00

Reset values:

- 0xA800 0000 for port A
- 0x0000 0280 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[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
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **MODERy[1:0]:** Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O direction mode.
00: Input (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

6.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A..H)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OTy[1:0]:** Port x configuration bits (y = 0..15)

These bits are written by software to configure the output type of the I/O port.

0: Output push-pull (reset state)

1: Output open-drain

6.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A..H)

Address offset: 0x08

Reset values:

- 0x0000 00C0 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15 [1:0]		OSPEEDR14 [1:0]		OSPEEDR13 [1:0]		OSPEEDR12 [1:0]		OSPEEDR11 [1:0]		OSPEEDR10 [1:0]		OSPEEDR9 [1:0]		OSPEEDR8 [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
OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1 [1:0]		OSPEEDR0 1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **OSPEEDRy[1:0]:** Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O output speed.

00: 400 kHz Very low speed

01: 2 MHz Low speed

10: 10 MHz Medium speed

11: 40 MHz High speed

6.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A..H)

Address offset: 0x0C

Reset values:

- 0x6400 0000 for port A
- 0x0000 0100 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[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
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **PUPDRy[1:0]:** Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O pull-up or pull-down
00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

6.4.5 GPIO port input data register (GPIOx_IDR) (x = A..H)

Address offset: 0x10

Reset value: 0x0000 XXXX (where X means undefined)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
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 **IDRy[15:0]:** Port input data (y = 0..15)

These bits are read-only and can be accessed in word mode only. They contain the input value of the corresponding I/O port.

6.4.6 GPIO port output data register (GPIOx_ODR) (x = A..H)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
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 **ODRy[15:0]:** Port output data (y = 0..15)

These bits can be read and written by software.

Note: For atomic bit set/reset, the ODR bits can be individually set and reset by writing to the GPIOx_BSRR register (x = A..H).

6.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A..H)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port x reset bit y (y = 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Resets the corresponding ODRx bit

Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BSy:** Port x set bit y (y= 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Sets the corresponding ODRx bit

6.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A..H)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the

LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next reset.

Note: A specific write sequence is used to write to the GPIOx_LCKR register. Only word access (32-bit long) is allowed during this write sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

Address offset: 0x1C

Reset value: 0x0000 0000

Access: 32-bit word only, read/write register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															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[16]:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.
0: Port configuration lock key not active
1: Port configuration lock key active. The GPIOx_LCKR register is locked until an MCU reset occurs.

LOCK key write sequence:
WR LCKR[16] = ‘1’ + LCKR[15:0]
WR LCKR[16] = ‘0’ + LCKR[15:0]
WR LCKR[16] = ‘1’ + LCKR[15:0]
RD LCKR
RD LCKR[16] = ‘1’ (this read operation is optional but it confirms that the lock is active)

Note: During the LOCK key write sequence, the value of LCK[15:0] must not change.

Any error in the lock sequence aborts the lock.

After the first lock sequence on any bit of the port, any read access on the LCKK bit will return ‘1’ until the next CPU reset.

Bits 15:0 **LCKy:** Port x lock bit y (y= 0..15)

These bits are read/write but can only be written when the LCKK bit is ‘0.
0: Port configuration not locked
1: Port configuration locked

6.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A..H)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[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
AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFRLy:** Alternate function selection for port x bit y (y = 0..7)

These bits are written by software to configure alternate function I/Os

AFRLy selection:
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

6.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A..H)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRH15[3:0]				AFRH14[3:0]				AFRH13[3:0]				AFRH12[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
AFRH11[3:0]				AFRH10[3:0]				AFRH9[3:0]				AFRH8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFRHy:** Alternate function selection for port x bit y (y = 8..15)

These bits are written by software to configure alternate function I/Os

AFRHy selection:
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

6.4.11 GPIO register map

The following table gives the GPIO register map and the reset values.

Table 26. GPIO register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**GPIOA_MODER**	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	1	0	1	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
0x00	**GPIOB_MODER**	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[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	1	0	1	0	0	0	0	0	0	0
0x00	**GPIOx_MODER** (where x = C..F)	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**GPIOx_OTYPER** (where x = A..E and H)	Reserved																OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**GPIOx_OSPEEDER** (where x = A..E and H except B)	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**GPIOB_OSPEEDER**	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
0x0C	**GPIOA_PUPDR**	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26. GPIO register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C	**GPIOB_PUPDR**	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0x0C	**GPIOx_PUPDR** (where x = C..F)	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**GPIOx_IDR** (where x = A..E and H)	Reserved																IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x14	**GPIOx_ODR** (where x = A..E and H)	Reserved																ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**GPIOx_BSRR** (where x = A..E and H)	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**GPIOx_LCKR** (where x = A..E and H)	Reserved															LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**GPIOx_AFRL** (where x = A..E and H)	AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[3:0]				AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**GPIOx_AFRH** (where x = A..E and H)	AFRH15[3:0]				AFRH14[3:0]				AFRH13[3:0]				AFRH12[3:0]				AFRH11[3:0]				AFRH10[3:0]				AFRH9[3:0]				AFRH8[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

Refer to *Table 2 on page 42* for the register boundary addresses.

7 System configuration controller (SYSCFG) and routing interface (RI)

This section applies to the whole STM32L1xxxx family, unless otherwise specified.

7.1 SYSCFG and RI introduction

The system configuration controller is mainly used to remap the memory accessible in the code area, and manage the external interrupt line connection to the GPIOs.

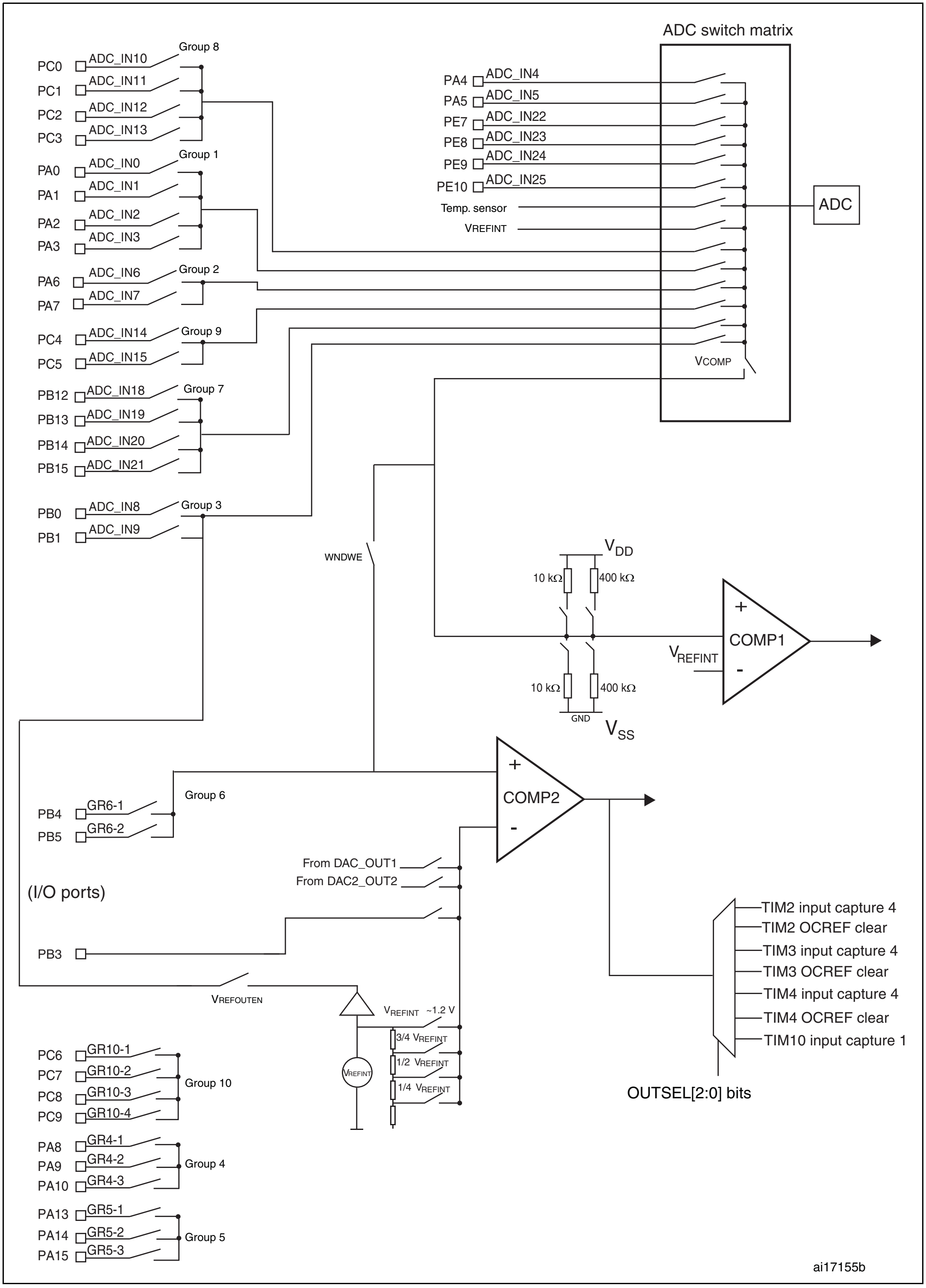
The routing interface provides high flexibility by allowing the software routing of I/Os toward the input captures of the STM32L1xxxx’s three high-end timers (TIM2, TIM3 and TIM4). The STM32L1xxxx’s ADC has an analog input matrix that is usually managed by a specific ADC interface. With the routing interface, it is possible to connect several I/O analog pins to a given channel of the ADC matrix by managing the analog switches of each I/O.

7.2 RI main features

- TIM2/TIM3/TIM4’s input captures 1,2,3 and four routing selections from selectable I/Os
- Routing of internal reference voltage V_{REFINT} to selectable I/Os for all packages
- Up to 40 external I/Os + 3 internal nodes (internal reference voltage + temperature sensor + V_{DD} and $V_{DD/2}$ measurement by V_{COMP}) can be used for data acquisition purposes in conjunction with the ADC interface
- Input and output routing of COMP1 and COMP2

Note: The RI registers can be accessed only when the comparator interface clock is enabled by setting the COMPEN bit in the RCC_APB1ENR register. Refer to Section 5.3.10 on page 117.

Figure 22. Routing interface (RI) block diagram for low and medium density devices



Note: The internal reference voltage and temperature sensor cannot be used as COMP1 non-inverting input.

Figure 23. Routing interface (RI) block diagram for medium+ density devices

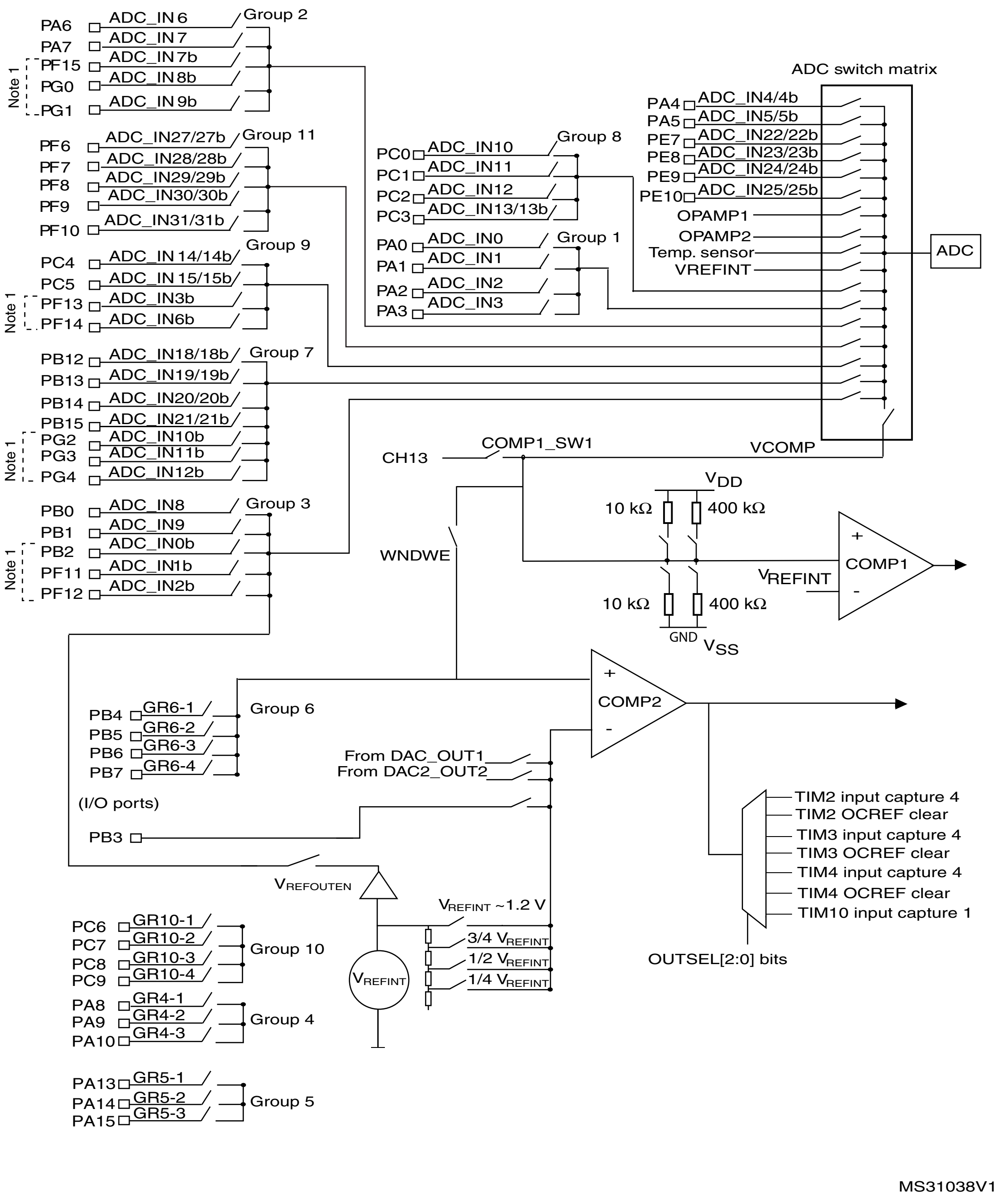
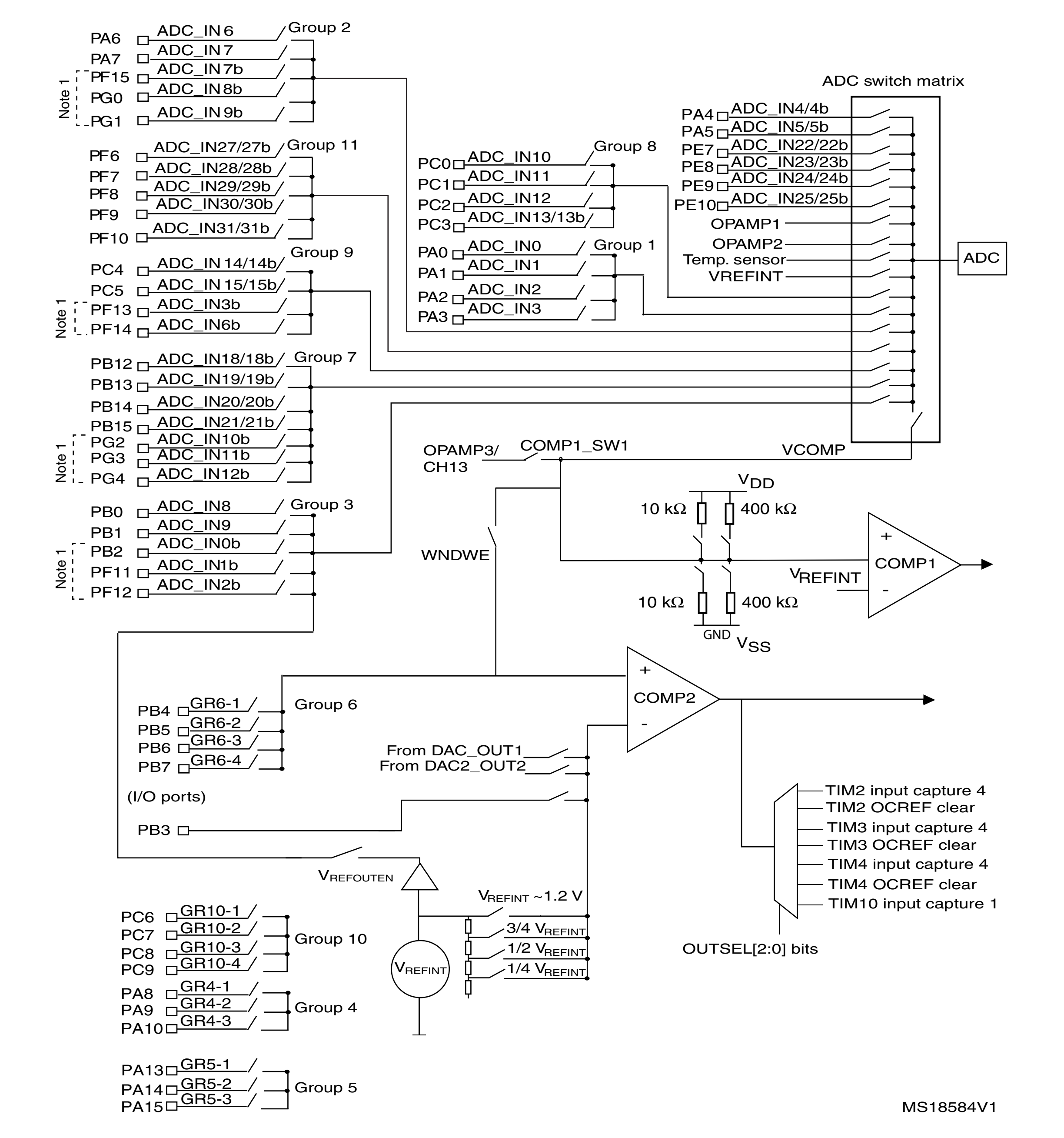


Figure 24. Routing interface (RI) block diagram for high-density devices



Note: *These I/O pins cannot be used as COMP1 inputs.*

The internal reference voltage and temperature sensor cannot be used as COMP1 non-inverting input.

7.3 RI functional description

7.3.1 Special I/O configuration

Two matrices of switches control the routing of I/Os toward analog blocks (that is the ADC or the comparator): I/O switches and ADC switches (refer to *Figure 22: Routing interface (RI) block diagram for low and medium density devices*).

- When I/Os are used for analog purposes other than data acquisition, the I/O and ADC switch matrices have to be controlled by the RI_ASCR1 and RI_ASCR2 registers. These registers are then used to close or open switches by software: closing switches sets the corresponding bits whereas opening switches clears them.
- When I/Os are used as ADC inputs for data acquisition purposes, the I/O and ADC switch matrices are directly controlled by the ADC interface. The corresponding bits in the RI_ASCR1 and RI_ASCR2 registers must be kept cleared (switches open).

High density and medium+ devices:

Up to 6 I/Os are connected directly and independently to the ADC through the pad resistor: these 6 I/Os offer the capability of fast and full 1 Mega sample/s data acquisition (Max ADC acquisition time). All others ADC channels don’t exceed 750 Ksamples. The output of the operational amplifiers 1 and 2 are directly connected to the ADC switches matrix and can be used also as 1 Mega sample/s data acquisition.This is not the case for operational amplifier 3 because its output goes through an additional COMP1_SW1 switch before to be connected to ADC switches matrix, available in high density devices only.

As shown in *Table 27: I/O groups and selection on page 155*, 50 I/Os are grouped within 11 groups to allow the application described into the appendices. When they are programmed in input mode by standard IOPORT registers, the Schmidt trigger and the hysteresis are enable by default. In this mode, registers RI_ASWCRx and RI_HYSCR allow simultaneously to close the corresponding analog switch pad and disable the Schmidt trigger hysteresis. So, It is possible to read the corresponding port with a trigger level of VDDIO/2.

Among these 11 groups only 7 groups (34 I/Os) are multiplexed to the ADC thanks to pad analog switches. With the 6 fast independent channels, 40 I/Os are available for data acquisition but only 29 I/Os are available for COMP1 comparison versus Vref bandgap (see *Figure 61: COMP1 interconnections (low and medium density devices) on page 292*, *Figure 62: COMP1 interconnections (high and medium+ density devices) on page 293* and application described in *Chapter Appendix C: Touch sensing by Stantum multitouch acquisition principle on page 64*).

Specific channels dedicated for each OPAMP use 3 I/Os among the 40 I/Os discussed above when these amplifiers are selected. (see *Chapter 2: Operational Amplifier interface on page 24*)

Low and medium density devices:

Note: For all I/Os used as comparator inputs, the I/O port configuration must be kept in analog mode.

Table 27 shows the grouping of I/Os, the control register bits used to configure them as analog inputs or outputs (irrespective of standard I/O port programming), and the associated ADC channel number.

Table 27. I/O groups and selection

Group numbering		GPIO port	Analog ADC channel	I/O + ADC analog switch	I/O functions
Group 1	GR1-1	PA0	CH0	RI_ASCR1->CH0	COMP1_INP
	GR1-2	PA1	CH1	RI_ASCR1->CH1	
	GR1-3	PA2	CH2	RI_ASCR1->CH2	
	GR1-4	PA3	CH3	RI_ASCR1->CH3	
Group 2	GR2-1	PA6	CH6	RI_ASCR1->CH6	COMP1_INP
	GR2-2	PA7	CH7	RI_ASCR1->CH7	
	GR2-3	PF15	CH7b/37	RI_ASWCR2->CH7b	
	GR2-4(1)	PG0(2)	CH8b/38	RI_ASWCR2->CH8b	
	GR2-5(1)	PG1(2)	CH9b/39	RI_ASWCR2->CH9b	
Group 3	GR3-1	PB0	CH8	RI_ASCR1->CH8	COMP1_INP /VREF_OUT
	GR3-2	PB1	CH9	RI_ASCR1->CH9	
	GR3-3(1)	PB2	CH0b/32	RI_ASWCR2->CH0b	COMP1_INP
	GR3-4(1)	PF11	CH1b/33	RI_ASWCR2->CH1b	
	GR3-5(1)	PF12	CH2b/34	RI_ASWCR2->CH2b	
Group 4	GR4-1	PA8	NA	RI_ASCR2->GR4-1	
	GR4-2	PA9		RI_ASCR2->GR4-2	
	GR4-3	PA10		RI_ASCR2->GR4-3	
Group 5	GR5-1	PA13	NA	RI_ASCR2->GR5-1	
	GR5-2	PA14		RI_ASCR2->GR5-2	
	GR5-3	PA15		RI_ASCR2->GR5-3	
Group 6	GR6-1	PB4	NA	RI_ASCR2->GR6-1	COMP2_INP
	GR6-2	PB5		RI_ASCR2->GR6-2	
	GR6-3(1)	PB6		RI_ASWCR2->GR6-3	
	GR6-4(1)	PB7		RI_ASWCR2->GR6-4	COMP2_INP/PVD_IN
Group 7	GR7-1	PB12	CH18	RI_ASCR1->CH18	COMP1_INP
	GR7-2	PB13	CH19	RI_ASCR1->CH19	
	GR7-3	PB14	CH20	RI_ASCR1->CH20	
	GR7-4	PB15	CH21	RI_ASCR1->CH21	
	GR7-5(1)	PG2(2)	CH10b	RI_ASWCR2->CH10b	
	GR7-6(1)	PG3(2)	CH11b	RI_ASWCR2->CH11b	
	GR7-7(1)	PG4(2)	CH12b	RI_ASWCR2->CH12b	

Table 27. I/O groups and selection (continued)

Group numbering		GPIO port	Analog ADC channel	I/O + ADC analog switch	I/O functions
Group 8	GR8-1	PC0	CH10	RI_ASCR1->CH10	COMP1_INP
	GR8-2	PC1	CH11	RI_ASCR1->CH11	
	GR8-3	PC2	CH12	RI_ASCR1->CH12	
	GR8-4	PC3	CH13	RI_ASCR1->CH13	
Group 9	GR9-1	PC4	CH14	RI_ASCR1->CH14	COMP1_INP
	GR9-2	PC5	CH15	RI_ASCR1->CH15	
	GR9-3(1)	PF13	CH3b	RI_ASWCR2->GR9-3	
	GR9-4(1)	PF14	CH6b	RI_ASWCR2->GR9-4	
Group 10	GR10-1	PC6	NA	RI_ASCR2->GR10-1	
	GR10-2	PC7		RI_ASCR2->GR10-2	
	GR10-3	PC8		RI_ASCR2->GR10-3	
	GR10-4	PC9		RI_ASCR2->GR10-4	
Group 11	GR11-1	PF6	CH27	RI_ASWCR->CH27	COMP1_INP
	GR11-2	PF7	CH28	RI_ASWCR->CH28	
	GR11-3	PF8	CH29	RI_ASWCR->CH29	
	GR11-4	PF9	CH30	RI_ASWCR->CH30	
	GR11-5	PF10	CH31	RI_ASWCR->CH31	
Fast channels		PA4	CH4	RI_ASWCR->CH4	COMP1_INP/DAC1
		PA5	CH5	RI_ASWCR->CH5	COMP1_INP/DAC2
		PE7	CH22	RI_ASWCR->CH22	COMP1_INP
		PE8	CH23	RI_ASWCR->CH23	COMP1_INP
		PE9	CH24	RI_ASWCR->CH24	COMP1_INP
		PE10	CH25	RI_ASWCR->CH25	COMP1_INP
OPAMP1_VOUT			CH3	NA	
Fast channel		PA3	CH3	COMP_CSR->FCH3	
OPAMP2_VOUT			CH8	NA	
Fast channel		PB0	CH8	COMP_CSR->FCH8	
OPAMP3_VOUT			CH13	NA	
		PC3	CH13	COMP_CSR->RCH13	
		V_{TMP}	CH16/CH16b	NA	
		V_{REF}	CH17/CH17b	NA	
		V_{COMP}	CH26/CH26b	NA	
NA		PB3	NA		COMP2_INM

Table 27. I/O groups and selection (continued)

Group numbering	GPIO port	Analog ADC channel	I/O + ADC analog switch	I/O functions
NA	PB6[(1)]	NA		COMP2_INP
NA	PB7	NA		PVD_IN/COMP2_INP

1. Available only in high and medium+ density devices.

2. When used in touch sensing solutions, these GPIOs can only be configured as sampling capacitor I/Os.

7.3.2 Input capture routing

By default (at reset), the four input captures of the three general-purpose timers (TIM2, TIM3, TIM4) are connected to the I/O port specified in the STM32L1xxxx datasheet's "pin descriptions" table.

The I/O routing can be changed by programming register RI_ICR as indicated below:

- The input capture 1 of TIM2, TIM3 and TIM4 can be rerouted from any I/O by configuring the IC1IOS[3:0] bits in RI_ICR.
- The input capture 2 of TIM2, TIM3 and TIM4 can be rerouted from any I/O by configuring the IC2IOS[3:0] bits in RI_ICR.
- The input capture 3 of TIM2, TIM3 and TIM4 can be rerouted from any I/O by configuring the IC3IOS[3:0] bits in RI_ICR.
- The input capture 4 of TIM2, TIM3 and TIM4 can be rerouted from any I/O by configuring the IC4IOS[3:0] bits in RI_ICR.

Refer to the following table for I/O routing to the input capture timers.

This capability can be applied on only one out of the three timers at a time by configuring TIM[1:0] in RI_ICR. When TIM[1:0]= 00 none of the three timers are affected by the I/O routing: the defaults connections are enabled.

Moreover, when a timer is selected, I/O routing can be enabled for one or more input captures by configuring the IC1, IC2, IC3 and IC4 bits in RI_ICR.

Refer to *Table 28* for the I/O correspondence and to *Table 29* for the timer selection.

Table 28. Input capture mapping

IC1IOS / IC2IOS / IC3IOS / IC4IOS	TIMx IC1 / TIMx IC2 / TIMx IC3 / TIMx IC4
0000	PA0 / PA1 / PA2 / PA3
0001	PA4 / PA5 / PA6 / PA7
0010	PA8 / PA9 / PA10 / PA11
0011	PA12 / PA13 / PA14 / PA15
0100	PC0 / PC1 / PC2 / PC3
0101	PC4 / PC5 / PC6 / PC7
0110	PC8 / PC9 / PC10 / PC11
0111	PC12 / PC13 / PC14 / PC15
1000	PD0 / PD1 / PD2 / PD3
1001	PD4 / PD5 / PD6 / PD7
1010	PD8 / PD9 / PD10 / PD11
1011	PD12 / PD13 / PD14 / PD15
1100	PE0 / PE1 / PE2 / PE3
1101	PE4 / PE5 / PE6 / PE7
1110	PE8 / PE9 / PE10 / PE11
1111	PE12 / PE13 / PE14 / PE15

Note: The I/O should be configured in alternate function mode (AF14).

Table 29. Timer selection

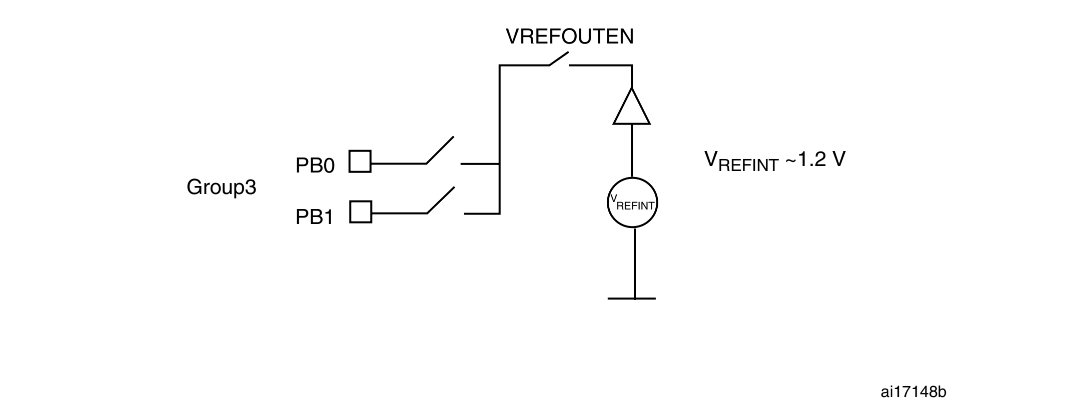
TIM[1:0]	Selected timer
00	No timer selected, default routing on all timers
01	TIM2 selected
10	TIM3 selected
11	TIM4 selected

Table 30. Input capture selection

IC4 / IC3 / IC2 / IC1	Selected input capture
0	IC deselected, default routing on the input capture (AF)
1	Input capture routing follows *Table 29*

7.3.3 Reference voltage routing

Figure 25. Internal reference voltage output



The V_{REFINT} output can be routed to any I/O in group 3 by following this procedure:

1. Set the VREFOUTEN bit in COMP_CSR.
2. Close the analog switch of all I/Os in group 3 by setting CH8 or CH9 in RI_ASCR1.

7.4 RI registers

The peripheral registers have to be accessed by words (32-bit).

7.4.1 RI input capture register (RI_ICR)

The RI_ICR register is used to select the routing of 4 full ports to the input captures of TIM2, TIM3 and TIM4.

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										IC4	IC3	IC2	IC1	TIM[1:0]	
										rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4IOS[3:0]				IC3IOS[3:0]				IC2IOS[3:0]				IC1IOS[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value

Bit 21 **IC4:** This bit is set and cleared by software to select the standard AF or the large routing capability on the input capture 4 of the timer selected by TIM[1:0] (bits 17:16).

0: AF on IC4
1: Multiple port routing capability according to IC4IOS[3:0] (bits 15:12)

Bit 20 **IC3:** This bit is set and cleared by software to select the standard AF or the large routing capability on the input capture 3 of the timer selected by TIM[1:0] (bits 17:16).

0: AF on IC3
1: Multiple port routing capability according to IC3IOS[3:0] (bits 11:8)

Bit 19 **IC2:** This bit is set and cleared by software to select the standard AF or the large routing capability on the input capture 2 of the timer selected by TIM[1:0] (bits 17:16).

0: AF on IC2
1: Multiple port routing capability according to IC2IOS[3:0] (bits 7:4)

Bit 18 **IC1**: This bit is set and cleared by software to select the standard AF or the large routing capability on the input capture 2 of the timer selected by TIM[1:0] (bits 17:16).

0: AF on IC1
1: Multiple port routing capability according to IC1IOS[3:0] (bits 3:0)

Bits 17:16 **TIM[1:0]**: Timer select bits

These bits are set and cleared by software. They are used to select one out of three timers or none.

00: non timer selected
01: TIM2 selected
10: TIM3 selected
11: TIM4 selected

Bits 15:12 **IC4IOS[3:0]**: Input capture 4 select bits

These bits are set and cleared by software. They select the input port to be routed to the IC4 of the selected timer (see bits 16:17).

0000: PA3	1000: PD3
0001: PA7	1001: PD7
0010: PA11	1010: PD11
0011: PA15	1011: PD15
0100: PC3	1100: PE3
0101: PC7	1101: PE7
0110: PC11	1110: PE11
0111: PC15	1111: PE15

Bits 11:8 **IC3IOS[3:0]**: Input capture 3 select bits

These bits are set and cleared by software. They select the input port to be routed toward the IC3 of the selected timer (see bits 16:17).

0000: PA2	1000: PD2
0001: PA6	1001: PD6
0010: PA10	1010: PD10
0011: PA14	1011: PD14
0100: PC2	1100: PE2
0101: PC6	1101: PE6
0110: PC10	1110: PE10
0111: PC14	1111: PE14

Bits 7:4 **IC2IOS[3:0]**: Input capture 2 select bits

These bits are set and cleared by software. They select the input port to be routed toward the IC2 of the selected timer (see bits 16:17).

0000: PA1	1000: PD1
0001: PA5	1001: PD5
0010: PA9	1010: PD9
0011: PA13	1011: PD13
0100: PC	11100: PE1
0101: PC5	1101: PE5
0110: PC9	1110: PE9
0111: PC13	1111: PE13

Bits 3:0 **IC1IOS[3:0]**: Input capture 1 select bits

These bits are set and cleared by software. They select the input port to be routed toward the IC1 of the selected timer (see bits 16:17).

0000: PA0	1000: PD0
0001: PA4	1001: PD4
0010: PA8	1010: PD8
0011: PA12	1011: PD12
0100: PC0	1100: PE0
0101: PC4	1101: PE4
0110: PC8	1110: PE8
0111: PC12	1111: PE12

Note: The standard AFs dedicated to TIM2 are:
IC4-> PA3,PB11 or PE12
IC3-> PA2, PB10 or PE11
IC2-> PA1, PB3 or PE10
IC1-> PA0, PA5, PA15 or PE9

The standard AFs dedicated to TIM3 are:
IC4-> PB1 or PC9

IC3-> PB0 or PC8
IC2-> PA7, PC7, PB5 or PE4
IC1-> PA6, PC6, PB4 or PE3

The standard AFs dedicated to TIM4 are:
IC4-> PD15 or PB9
IC3-> PD14 or PB8
IC2-> PD13 or PB7
IC1-> PD12 or PB6

7.4.2 RI analog switches control register (RI_ASCR1)

The RI_ASCR1 register is used to configure the analog switches of the I/Os linked to the ADC. These I/Os are pointed to by the ADC channel number.

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCM	CH30 GR11-4	CH29 GR11-3	CH28 GR11-2	CH27 GR11-1	Vcomp	CH25	CH24	CH23	CH22	CH21 GR7-4	CH20 GR7-3	CH19 GR7-2	CH18 GR7-1	Res.	CH31 GR11-5
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CH15 GR9-2	CH14 GR9-1	CH13 GR8-4	CH12 GR8-3	CH11 GR8-2	CH10 GR8-1	CH9 GR3-2	CH8 GR3-1	CH7 GR2-2	CH6 GR2-1	CH5	CH4	CH3 GR1-4	CH2 GR1-3	CH1 GR1-2	CH0 GR1-1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SCM**: ADC Switch control mode

This bit is set and cleared by software. When this bit is set, setting a bit in RI_ASCRx that controls an analog I/O switch will also close the corresponding switch of the ADC switch matrix. When this bit is reset, the other bits in RI_ASCRx do not control the switches of the ADC switch matrix.

0: ADC analog switches open or controlled by the ADC interface
1: ADC analog switches closed if the corresponding I/O switch is also closed

Bits 30:27 **CH[30:27]/GR11[4:1]** *I/O Analog switch control*

These bits are set and cleared by software to control the I/O analog switches.

0: Analog switch open or controlled by the ADC interface
1: Analog switch closed

Note: These bits are available in high and medium+ density devices only

Bit 26 **VCOMP**: ADC analog switch selection for internal node to comparator 1

This bit is set and cleared by software to control the VCOMP ADC analog switch. See *Figure 61 on page 292* and *Figure 62 on page 293*.

0: Analog switch open
1: Analog switch closed

Bits 25:22 **CH[25:22]**: Analog I/O switch control of channels CH[25:22]

These bits are set and cleared by software to control the analog switches of the ADC switch matrix. If the I/O is used as an ADC input, the switch must be left open to allow the ADC to control it.

0: Analog switch open
1: Analog switch closed

Bits 21:18 **CH[21:18]/GR7[4:1]** *I/O Analog switch control*

These bits are set and cleared by software to control the I/O analog switches.

0: Analog switch open or controlled by the ADC interface

1: Analog switch closed

Bit 17 Reserved

Bit 16 **CH31/GR11-5** *I/O Analog switch control*

This bit is set and cleared by software to control the I/O analog switch.

0: Analog switch open or controlled by the ADC interface

1: Analog switch closed

Note: This bit is available in high and medium+ density devices only

Bits 15:14 **CH[15:14] GR9[2:1]**: *I/O Analog switch control*

These bits are set and cleared by software to control the I/O analog switches.

0: Analog switch open or controlled by the ADC interface

1: Analog switch closed

Bits 13:10 **CH[13:10] GR8[4:1]**: *I/O Analog switch control*

These bits are set and cleared by software to control the I/O analog switches.

0: Analog switch open or controlled by the ADC interface

1: Analog switch closed

Bits 9:8 **CH[9:8] GR3[2:1]**: *I/O Analog switch control*

These bits are set and cleared by software to control the I/O analog switches.

0: Analog switch open or controlled by the ADC interface

1: Analog switch closed

Bits 7:6 **CH[7:6] GR2[2:1]**: *I/O Analog switch control*

These bits are set and cleared by software to control the I/O analog switches.

0: Analog switch open or controlled by the ADC interface

1: Analog switch closed

Bit 5 **CH5**: Comparator 1 analog switch

This bit is set and cleared by software to control the core analog switch of the ADC switch matrix connecting the positive input of the COMP1 comparator. It can be used to route the ADC matrix or OPAMP3 output to the comparator1 positive input. See *Figure 62 on page 293*.

0: Analog switch open

1: Analog switch closed

Bit 4 **CH4**: *Analog switch control*

This bit is set and cleared by software to control the analog switches of the ADC switch matrix.

0: Analog switch open

1: Analog switch closed

Bits 3:0 **CH[3:0] GR1[4:1]**: *I/O Analog switch control*

These bits are set and cleared by software to control the I/O analog switches.

0: Analog switch open or controlled by the ADC interface

1: Analog switch closed

Note: The ADC_IN16 and ADC_IN17 channels are internal and controlled only by the ADC interface for data acquisition purposes.

The ADC_IN4, ADC_IN5, ADC_IN22, ADC_IN23, ADC_IN24 and ADC_IN25 channels are directly connected to the ADC through a resistor, no need to close external I/O analog switches.

When the SCM bit is low, the CH bits are used to connect groups of I/Os together by analog switches, independently of the ADC.

When the SCM bit is high, the CH bits are used to connect several I/Os together through the ADC switch matrix in order to allow a possible wakeup by COMP1 if the VCOMP bit is high.

7.4.3 RI analog switch control register 2 (RI_ASCR2)

The RI_ASCR2 register is used to configure the analog switches of groups of I/Os not linked to the ADC. In this way, predefined groups of I/Os can be connected together.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			GR6-4	GR6-3	CH12b GR7-7	CH11b GR7-6	CH10b GR7-5	CH9b GR2-5	CH8b GR2-4	CH7b GR2-3	CH6b GR9-4	CH3b GR9-3	CH2b GR3-5	CH1b GR3-4	CH0b GR3-3
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				GR4-3	GR4-2	GR4-1	GR5-3	GR5-2	GR5-1	GR6-2	GR6-1	GR10-4	GR10-3	GR10-2	GR10-1
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value

Bits 28:16 **GRx-x:** GRx-x I/O analog switch control

These bits are set and cleared by software to control the I/O analog switches independently from the ADC interface. Refer to *Table 27: I/O groups and selection on page 155*.

0: Analog switch open or controlled by the ADC interface

1: Analog switch closed

Note: These bits are available in high and medium+ density devices only.

Bits 15:12 Reserved, must be kept at reset value

Bits 11:0 **GRx-x:** GRx-x I/O analog switch control

These bits are set and cleared by software to control the I/O analog switches independently from the ADC interface. Refer to *Table 27: I/O groups and selection on page 155* .

0: Analog switch open or controlled by the ADC interface

1: Analog switch closed

7.4.4 RI hysteresis control register (RI_HYSCR1)

The RI_HYSCR1 register is used to enable/disable the hysteresis of the input Schmitt trigger of ports A and B.

Address offset: 0x10

Reset value: 0x0000 0000

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

Bits 31:16 **PB[15:0]:** Port B hysteresis control on/off

These bits are set and cleared by software to control the Schmitt trigger hysteresis of the Port B[15:0].

0: Hysteresis on

1: Hysteresis off

Bits 15:0 **PA[15:0]:** Port A hysteresis control on/off

These bits are set and cleared by software to control the Schmitt trigger hysteresis of the Port A[15:0].

0: Hysteresis on

1: Hysteresis off

7.4.5 RI Hysteresis control register (RI_HYSCR2)

RI_HYSCR2 register allows to enable/disable hysteresis of input Schmitt trigger of ports C and D.

Address offset: 0x14

Reset value: 0x0000 0000

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

Bits 31:16 **PD[15:0]:** Port D hysteresis control on/off

These bits are set and cleared by software to control the Schmitt trigger hysteresis of the Port D[15:0].

0: Hysteresis on

1: Hysteresis off

Bits 15:0 **PC[15:0]:** Port C hysteresis control on/off

These bits are set and cleared by software to control the Schmitt trigger hysteresis of the Port C[15:0].

0: Hysteresis on

1: Hysteresis off

7.4.6 RI Hysteresis control register (RI_HYSCR3)

The RI_HYSCR3 register is used to enable/disable the hysteresis of the input Schmitt trigger of the entire port E and F.

Address offset: 0x18

Reset value: 0x0000 0000

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

Bits 31:6 **PF[15:0]:** Port F hysteresis control on/off

These bits are set and cleared by software to control the Schmitt trigger hysteresis of the Port F[15:0].

0: Hysteresis on

1: Hysteresis off

Note: These bits are available in high and medium+ density devices only.

Bits 15:0 **PE[15:0]:** Port E hysteresis control on/off

These bits are set and cleared by software to control the Schmitt trigger hysteresis of the Port E[15:0].

0: Hysteresis on

1: Hysteresis off

7.4.7 RI Hysteresis control register (RI_HYSCR4)

The RI_HYSCR4 register is used to enable/disable the hysteresis of the input Schmitt trigger of the entire port G.

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PG[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 **PG[15:0]:** Port G hysteresis control on/off

These bits are set and cleared by software to control the Schmitt trigger hysteresis of the Port G[15:0].

0: Hysteresis on

1: Hysteresis off

Note: These bits are available in high and medium+ density devices only.

7.4.8 Analog switch mode register (RI_ASMR1)

The RI_ASMR1 register is available in high and medium+ density devices only and is used to select if analog switches of port A are to be controlled by the timer OC or through the ADC interface or RI_ASCRx registers.

Address offset: 0x20

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value

Bits 15:0 **PA[15:0]:** *Port A analog switch mode selection*

These bits are set and cleared by software to select the mode of controlling the analog switches for Port A.

0: ADC interface or RI_ASCRx controlled

1: Timer controlled

7.4.9 Channel mask register (RI_CMR1)

RI_CMR1 is available in high and medium+ density devices only and is used to mask a port A channel designated as a timer input capture (after acquisition completion to avoid triggering multiple detections).

Address offset: 0x24

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value

Bits 15:0 **PA[15:0]:** *Port A channel masking*

These bits are set and cleared by software to mask the input of port A during the capacitive sensing acquisition.

0: Masked

1: Not masked

7.4.10 Channel identification for capture register (RI_CICR1)

The RI_CICR1 register is available in high and medium+ density devices only and is used when analog switches are controlled by a timer OC. RI_CICR1 allows a channel to be identified for timer input capture.

Address offset: 0x28

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value

Bits 15:0 **PA[15:0]:** *Port A channel identification for capture*

These bits are set and cleared by software to identify the sampling capacitor I/Os on Port A.

0: Channel I/O

1: Sampling capacitor I/O

7.4.11 Analog switch mode register (RI_ASMR2)

The RI_ASMR2 register is available in high and medium+ density devices only and is used to select if analog switches of port B are to be controlled by the timer OC or through the ADC interface or RI_ASCRx registers.

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PB[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 **PB[15:0]:** *Port B analog switch mode selection*

These bits are set and cleared by software to select the mode of controlling the analog switches for Port B.

0: ADC interface or RI_ASCRx controlled

1: Timer controlled

7.4.12 Channel mask register (RI_CMR2)

RI_CMR2 is available in high and medium+ density devices only and is used to mask a por B channel designated as a timer input capture (after acquisition completion to avoid triggering multiple detections)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PB[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 **PB[15:0]:** *Port B channel masking*

These bits are set and cleared by software to mask the input of port B during the capacitive sensing acquisition.

0: Masked

1: Not masked

7.4.13 Channel identification for capture register (RI_CICR2)

The RI_CICR2 register is available in high and medium+ density devices only and is used when analog switches are controlled by a timer OC. RI_CICR2 allows a port B channel to be identified for timer input capture.

Address offset: 0x34

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PB[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 **PB[15:0]:** *Port B channel identification for capture*

These bits are set and cleared by software to identify the sampling capacitor I/Os on Port B.

0: Channel I/O

1: Sampling capacitor I/O

7.4.14 Analog switch mode register (RI_ASMR3)

The RI_ASMR3 register is available in high and medium+ density devices only and is used to select if analog switches of port C are to be controlled by the timer OC or through the ADC interface or RI_ASCRx registers.

Address offset: 0x38

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC[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 **PC[15:0]:** *Port C analog switch mode selection*

These bits are set and cleared by software to select the mode of controlling the analog switches for Port C.

0: ADC interface or RI_ASCRx controlled

1: Timer controlled

7.4.15 Channel mask register (RI_CMR3)

RI_CMR3 is available in high and medium+ density devices only and is used to mask a port C channel designated as a timer input capture (after acquisition completion to avoid triggering multiple detections)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC[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 **PC[15:0]:** *Port C channel masking*

These bits are set and cleared by software to mask the input of port C during the capacitive sensing acquisition.

0: Masked

1: Not masked

7.4.16 Channel identification for capture register (RI_CICR3)

The RI_CICR3 register is available in high and medium+ density devices only and is used when analog switches are controlled by a timer OC. RI_CICR3 allows a port C channel to be identified for timer input capture.

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC[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 **PC[15:0]:** *Port C channel identification for capture*

These bits are set and cleared by software to identify the sampling capacitor I/Os on Port C.

0: Channel I/O

1: Sampling capacitor I/O

7.4.17 Analog switch mode register (RI_ASMR4)

The RI_ASMR4 register is available in high and medium+ density devices only and is used to select if analog switches of port F are to be controlled by the timer OC or through the ADC interface or RI_ASCRx registers.

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PF[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 **PF[15:0]:** *Port F analog switch mode selection*

These bits are set and cleared by software to select the mode of controlling the analog switches for Port F.

0: ADC interface or RI_ASCRx controlled

1: Timer controlled

Note: These bits are available in high and medium+ density devices only.

7.4.18 Channel mask register (RI_CMR4)

RI_CMR4 is available in high and medium+ density devices only and is used to mask a port F channel designated as a timer input capture (after acquisition completion to avoid triggering multiple detections).

Address offset: 0x48

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PF[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 **PF[15:0]:** *Port F channel masking*

These bits are set and cleared by software to mask the input of port F during the capacitive sensing acquisition.

0: Masked

1: Not masked

Note: These bits are available in high and medium+ density devices only.

7.4.19 Channel identification for capture register (RI_CICR4)

The RI_CICR4 register is available in high and medium+ density devices only and is used when analog switches are controlled by a timer OC. RI_CICR4 allows a port F channel to be identified for timer input capture.

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PF[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 **PF[15:0]:** *Port F channel identification for capture*

These bits are set and cleared by software to identify the sampling capacitor I/Os on Port F.

0: Channel I/O

1: Sampling capacitor I/O

Note: These bits are available in high density and medium+ devices only.

7.4.20 Analog switch mode register (RI_ASMR5)

The RI_ASMR5 register is available in high and medium+ density devices only and is used to select if analog switches of port G are to be controlled by the timer OC or through the ADC interface or RI_ASCRx registers.

Address offset: 0x50

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PG[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 **PG[15:0]:** *Port G analog switch mode selection*

These bits are set and cleared by software to select the mode of controlling the analog switches for Port G.

0: ADC interface or RI_ASCRx controlled

1: Timer controlled

Note: These bits are available in high and medium+ density devices only.

7.4.21 Channel mask register (RI_CMR5)

RI_CMR5 is available in high and medium+ density devices only and is used to mask a port G channel designated as a timer input capture (after acquisition completion to avoid triggering multiple detections).

Address offset: 0x54

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PG[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 **PG[15:0]:** *Port G channel masking*

These bits are set and cleared by software to mask the input of port G during the capacitive sensing acquisition.

0: Masked

1: Not masked

Note: These bits are available in high and medium+ density devices only.

7.4.22 Channel identification for capture register (RI_CICR5)

The RI_CICR5 register is available in high and medium+ density devices only and is used when analog switches are controlled by a timer OC. RI_CICR5 allows a port G channel to be identified for timer input capture.

Address offset: 0x58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PG[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 **PG[15:0]:** *Port G channel identification for capture*

These bits are set and cleared by software to identify the sampling capacitor I/Os on Port G.

0: Channel I/O

1: Sampling capacitor I/O

Note: These bits are available in high and medium+ density devices only.

7.4.23 RI register map

Table 31 summarizes the RI registers.

Table 31. RI 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
0x04	RI_ICR	Reserved										IC 4	IC 3	IC 2	IC 1	TIM[1:0]		IC4IOS[3:0]				IC3IOS[3:0]				IC2IOS[3:0]				IC1IOS[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
0x08	RI_ASCR1	SCM	GR11[4:1]				VCOMP	CH25:22				GR7[4:1]				Reserved	GR11-5	GR9 [2:1]		GR8[4:1]				GR3 [2:1]		GR2 [2:1]		CH5	CH4	GR1[4:1]			
	Reset value	0	0	0	0	0	0	0	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	RI_ASCR2	Reserved			GR6 [4:3]		GR7[7:5]			GR2[5:3]			GR9 [4:3]		GR3[5:3]			Reserved				GR4[3:1]			GR5[3:1]			GR6 [2:1]		GR10[4:1]			
	Reset value				0	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	RI_HYSCR1	PB[15:0]																PA[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	RI_HYSCR2	PD[15:0]																PC[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
0x18	RI_HYSCR3	PF[15:0]																PE[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
0x1C	RI_HYSCR4	Reserved																PG[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	RI_ASMR1	Reserved																PA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	RI_CMR1	Reserved																PA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	RI_CICR1	Reserved																PA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	RI_ASMR2	Reserved																PB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	RI_CMR2	Reserved																PB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	RI_CICR2	Reserved																PB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	RI_ASMR3	Reserved																PC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	RI_CMR3	Reserved																PC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 31. RI register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x40	**RI_CICR3**	Reserved																PC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**RI_ASMR4**	Reserved																PF[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**RI_CMR4**	Reserved																PF[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	**RI_CICR4**	Reserved																PF[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	**RI_ASMR5**	Reserved																PG[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x54	**RI_CMR5**	Reserved																PG[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x58	**RI_CICR5**	Reserved																PG[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

7.5 SYSCFG registers

The peripheral registers have to be accessed by words (32-bit).

7.5.1 SYSCFG memory remap register (SYSCFG_MEMRMP)

This register is used for specific configurations on memory remap:

- Two bits are used to configure the type of memory accessible at address 0x0000 0000. These bits are used to select the physical remap by software and so, bypass the BOOT pins.
- After reset these bits take the value selected by the BOOT pins.

Note: This register is not reset through the SYSCFGRST bit in the RCC_APB2RSTR register.

Address offset: 0x00

Reset value: 0x0000 00XX (X is the memory mode selected by the BOOT pins)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						BOOT_MODE		Reserved						MEM_MODE	
						r	r							rw	rw

Bits 31:10 Reserved

Bits 9:8 **BOOT_MODE:** Boot mode selected by the boot pins

These bits are read only. They indicate the boot mode selected by the boot pins. Bit 9 corresponds to the value sampled on the BOOT1 pin, and bit 8 corresponds to value sampled on the BOOT0 pin. See also *Section 2.4: Boot configuration on page 53*.

00: Main Flash memory boot mode
01: System Flash memory boot mode
10: Reserved
11: Embedded SRAM boot mode

Bits 7:2 Reserved

Bits 1:0 **MEM_MODE:** Memory mapping selection

Set and cleared by software. This bit controls the memory’s internal mapping at address 0x0000 0000. After reset these bits take on the memory mapping selected by the BOOT pins.

00: Main Flash memory mapped at 0x0000 0000
01: System Flash memory mapped at 0x0000 0000
10: FSMC
11: SRAM mapped at 0x0000 0000

7.5.2 SYSCFG peripheral mode configuration register (SYSCFG_PMC)

An internal pull-up resistor (1.5 kΩ) can be connected by software on the USB data + (DP) line. This internal pull-up resistor is enabled if the USB is not in power-down mode and if the USB_PU bit is set.

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										LCD_CAPA					USB_PU
										rw					rw

Bits 31:6 Reserved

Bit 5:1 **LCD_CAPA** *decoupling capacitance connection* (see device datasheet for this device capability)

Bit 1 controls the connection of $V_{LCDrail2}$ on PB2/LCD_VCAP2

0: $V_{LCDrail2}$ not connected to PB2/LCD_VCAP2

1: $V_{LCDrail2}$ connected to PB2/LCD_VCAP2

Bit 2 controls the connection of $V_{LCDrail1}$ on PB12/LCD_VCAP1

Bit 3 controls the connection of $V_{LCDrail3}$ on PB0/LCD_VCAP3

Bit 4 controls the connection of $V_{LCDrail1}$ on PE11/LCD_VCAP1

Bit 5 controls the connection of $V_{LCDrail3}$ on PE12/LCD_VCAP3

Bit 0 **USB_PU** *USB pull-up enable on DP line*

Set and cleared by software. This bit controls the internal pull-up (1.5 kΩ) on the USB DP line.

0: no pull-up on the USB DP line (even if USB is not in power down mode)

1: internal pull-up is connected on USB DP line (only if USB is not in power down mode)

7.5.3 SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1)

Address offset: 0x08

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 0 to 3)

These bits are written by software to select the source input for the EXTIx external interrupt.
0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PH[x] (only PH[2:0], PH3 is not available)
0110: PF[x] pin (medium+ and high density device only)
0111: PG[x] pin (medium+ and high density device only)

7.5.4 SYSCFG external interrupt configuration register 2 (SYSCFG_EXTICR2)

Address offset: 0x0C

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 4 to 7)

These bits are written by software to select the source input for the EXTIx external interrupt.
0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0110: PF[x] pin (medium+ and high density device only)
0111: PG[x] pin (medium+ and high density device only)
PH[7:4] are not available.

7.5.5 SYSCFG external interrupt configuration register 3 (SYSCFG_EXTICR3)

Address offset: 0x10

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 8 to 11)

These bits are written by software to select the source input for the EXTIx external interrupt.
0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0110: PF[x] pin (medium+ and high density device only)
0111: PG[x] pin (medium+ and high density device only)
PH[11:8] are not available.

7.5.6 SYSCFG external interrupt configuration register 4 (SYSCFG_EXTICR4)

Address offset: 0x14

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 12 to 15)

These bits are written by software to select the source input for the EXTIx external interrupt.
0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0110: PF[x] pin (medium+ and high density device only)
0111: PG[x] pin (medium+ and high density device only)
PH[15:12] are not available.

7.5.7 SYSCFG register map

The following table gives the SYSCFG register map and the reset values.

Table 32. SYSCFG register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**SYSCFG_ MEMRMP**	Reserved																						BOOT_MODE		Reserved						MEM_MODE	
	Reset value																							x	x							x	x
0x04	**SYSCFG_PMC**	Reserved																															USB_PU
	Reset value																																0
0x08	**SYSCFG_ EXTICR1**	Reserved																EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 32. SYSCFG register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C	SYSCFG_EXTICR2	Reserved																EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SYSCFG_EXTICR3	Reserved																EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	SYSCFG_EXTICR4	Reserved																EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Table 2: Register boundary addresses on page 42*.

8 Touch sensing I/Os

8.1 Introduction

All STM32L151xx devices except the value line provide a simple solution for adding capacitive sensing functionality to any application. Capacitive sensing technology is able to detect finger presence near an electrode which is protected from direct touch by a dielectric (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. It consists of charging the electrode capacitance and then transferring a part of the accumulated charges into a sampling capacitor until the voltage across this capacitor has reached a specific threshold. In the STM32L1xxxx, this acquisition is managed directly by the GPIOs, timers and analog I/O groups (see *Section 7: System configuration controller (SYSCFG) and routing interface (RI)*).

Reliable touch sensing solution can be quickly and easily implemented using the free STM32L1xx STMTouch firmware library.

8.2 Main features

- Proven and robust surface charge transfer acquisition principle
- Management of the charge transfer acquisition sequence in two modes: software mode or timer mode
- Supports up to 34 capacitive sensing channels
- Up to 11 capacitive sensing channels can be acquired in parallel offering a very good response time
- One sampling capacitor for up to 4 capacitive sensing channels which reduces the system BOM
- Compatible with touchkey, proximity, linear and rotary touch sensors
- Designed to operate with STM32L1xx STMTouch firmware library

8.3 Functional description

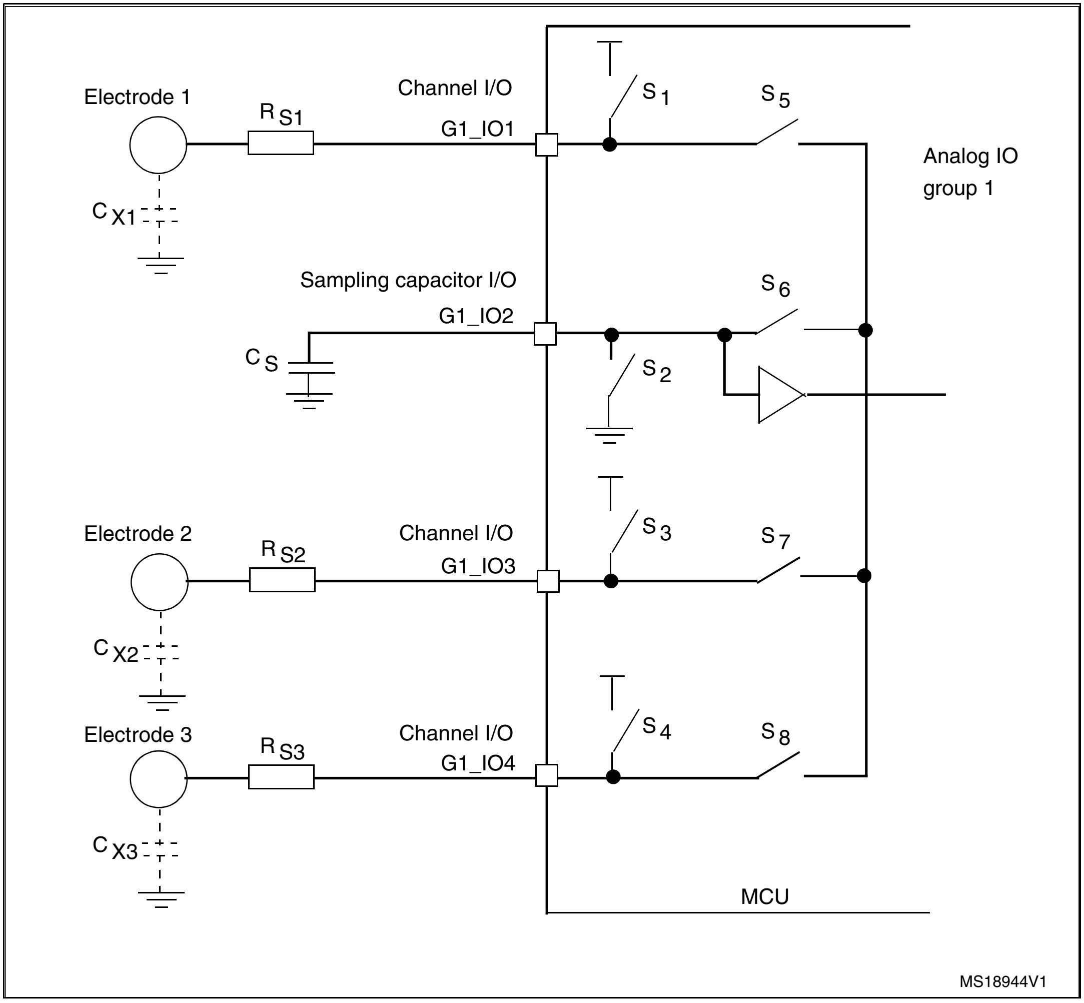
8.3.1 Surface charge transfer acquisition overview

The surface charge transfer acquisition is a proven, robust and efficient way to measure a capacitance. It uses a minimum of external components to operate with single ended electrode type. This acquisition is designed around an analog IO group which is composed of up to seven GPIOs (see *Figure 26*). The device offers several analog IO groups to allow acquiring simultaneously several capacitive sensing channels and to support a larger number of channels. Within a same analog IO group, the capacitive sensing channels acquisition is sequential.

One of the GPIOs is dedicated to the sampling capacitor C_S. Only one sampling capacitor per analog IO group must be enabled at a time.

The remaining GPIOs are dedicated to the electrodes and are commonly called channels.

Figure 26. Surface charge transfer analog IO group structure



Note: Gx_IOy where x is the analog IO group number and y the GPIO number within the selected group.

For some specific needs (for example proximity detection), it is possible to enable, more than one channel per analog IO group simultaneously.

For the table of capacitive sensing I/Os refer to *Table 27: I/O groups and selection on page 155*.

The surface charge transfer acquisition principle consists in 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 (typically V_{IH}). The number of charge transfers required to reach the threshold is a direct representation of the size of the electrode capacitance.

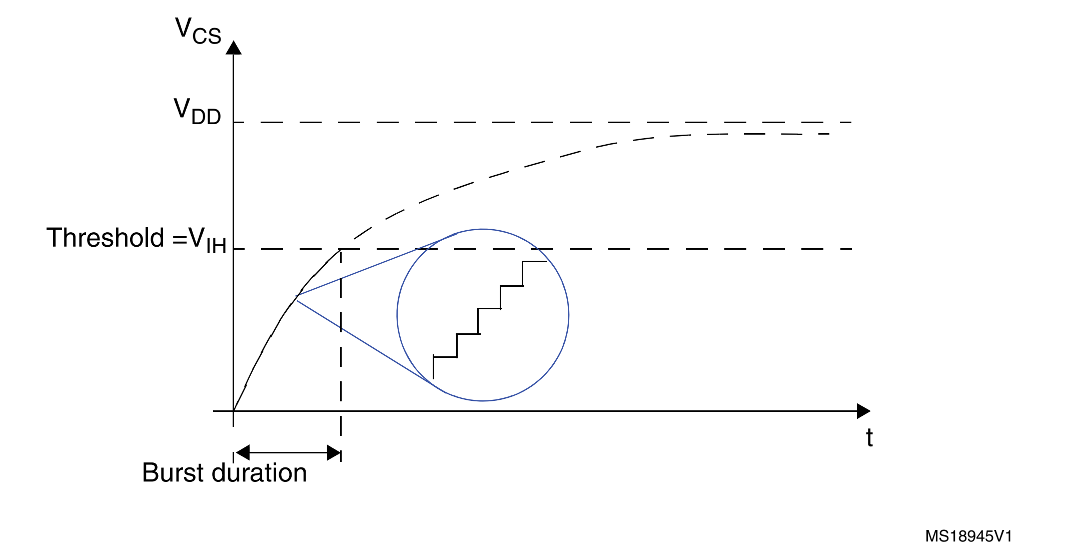
The *Table 33* details the acquisition sequence of the capacitive sensing channel 1.The states 3 to 7 are repeated until the voltage across C_S reaches the given threshold. The same sequence applies to the acquisition of the other channels. The electrode serial resistor R_S allows improving the ESD immunity of the solution.

Table 33. Acquisition switching sequence summary

State	S1	S2	S3	S4	S5	S6	S7	S8	State description
#1	Open	Closed	Open	Open	Closed	Closed	Closed	Closed	Discharge all C_X and C_S
#2	Open	Open	Open	Open	Open	Open	Open	Open	Dead time
#3	Closed	Open	Open	Open	Open	Open	Open	Open	Charge C_{X1}
#4	Open	Open	Open	Open	Open	Open	Open	Open	Dead time
#5	Open	Open	Open	Open	Closed	Closed	Open	Open	Charge transfer from C_{X1} to C_S
#6	Open	Open	Open	Open	Open	Open	Open	Open	Dead time
#7	Open	Open	Open	Open	Open	Open	Open	Open	Measure voltage across C_S

The voltage variation over the time of the sampling capacitor C_S is detailed below:

Figure 27. Sampling capacitor charge overview



8.3.2 Charge transfer acquisition management

In STM32L1xxxx devices, the acquisition can be managed in two modes:

- **Software mode:** the GPIO port toggling and counting of the number of pulses is fully handled by the CPU.
- **Timer mode:** the GPIO port toggling and counting of the number of pulses is handled by the timers (only in high and medium+ density devices).

Software mode acquisition

This mode uses the two following peripherals to control of the channel and sampling capacitor I/Os according to:

- General purpose I/Os (see *Section 6: General-purpose I/Os (GPIO)*)
- Routing interface (see *Section 7: System configuration controller (SYSCFG) and routing interface (RI) on page 150*)

Table 34. Channel and sampling capacitor I/Os configuration summary

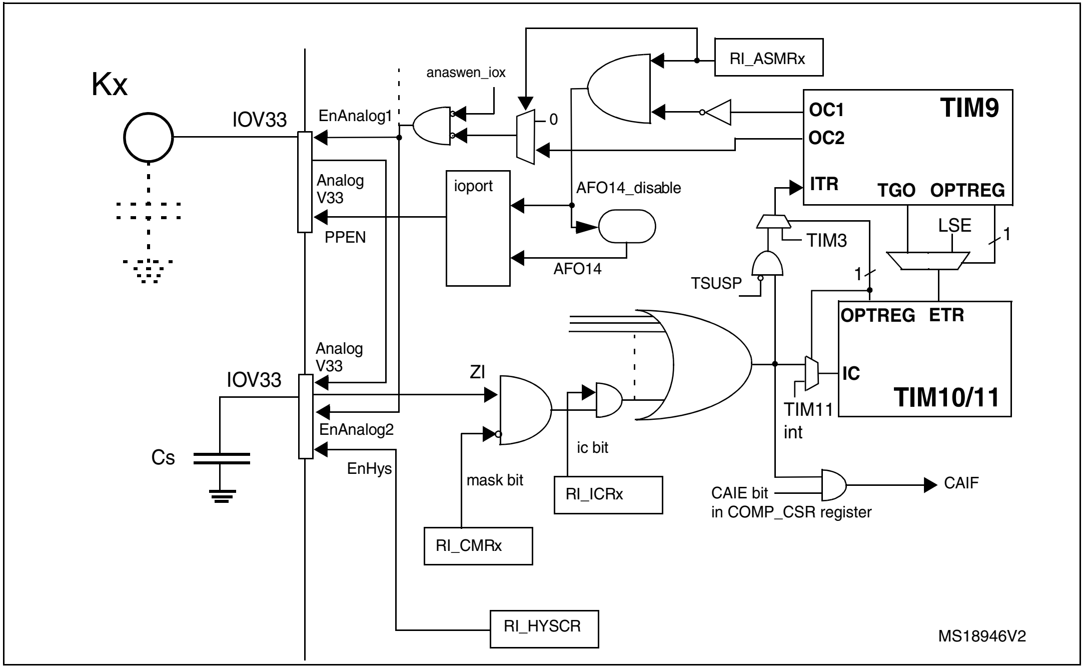
State	Channel I/O configuration	Sampling capacitor I/O configuration
#1	Output push-pull low	Output push-pull low
#2	Input floating	Input floating with hysteresis disabled
#3	Output push-pull high	
#4	Input floating	
#5	Input floating with analog switch closed	Input floating with hysteresis disabled and analog switch closed
#6	Input floating	Input floating with hysteresis disabled
#7		

Timer mode acquisition

This mode requires the use of the following peripherals:

- General purpose I/Os (see *Section 6 on page 131*)
- Routing interface (see *Section 7 on page 150*)
- General-purpose timer 9 (see *Section 17 on page 399*)
- General-purpose timer 10 or 11 (see *Section 17 on page 399*)

Figure 28. Timer mode acquisition logic



Both timers are used to manage the GPIO port toggling which dramatically reduces the CPU load. TIM9 is handles the charge transfer sequence generation by directly controlling the channel and sampling capacitor I/Os. TIM10 or 11 count the number of charge transfer cycles generated before an end of acquisition is detected on one of the enabled analog I/O groups.

The software sequence can be used to configure the different peripherals and to perform the acquisition.

Configuration steps:

1. Configure the I/O ports in alternate push-pull output mode, for the capacitive sensing channel I/Os to be acquired. Note that only one channel per group must be enabled at a time.
2. Configure the sampling capacitor I/Os in floating input mode with hysteresis disabled.
3. Configure TIM9 timer in center-aligned mode and generate PWM signals on OC1 and OC2. The typical frequency of the PWM signal is 250 KHz.
4. Configure TIM10 or 11 in slave mode with the clock signal generated by TIM9. In addition, IC1 must be enabled to capture the counter value on detection of an end of acquisition. Interrupt generation can be optionally enabled.
5. Enable TIM10 or 11.
6. Enable the control of analog switches of the channel and sampling capacitor I/Os by TIM9 using the RI_ASMRx registers (see *Section 7.4: RI registers on page 160*).
7. Identify the sampling capacitor I/Os using the registers RI_CICRx.
8. Start the acquisition by enabling TIM9.

Measurement steps:

At the end of acquisition on one sampling capacitor:

9. Clear the capture interrupt flag of TIM10 or 11 if interrupt generation enabled.
10. Read the IC register and save its contents in RAM.
11. Perform a XOR between RI_CMRx and GPIOx_IDR registers to determine which new channel(s) has (have) triggered an end of acquisition and update the corresponding channel information with the counter value.
12. Update the RI_CMRx registers to mask further end of acquisition detections on already detected channels. This will automatically restart TIM9.

8.4 Touch sensing library

In order to facilitate the development of a touch sensing solution based on STM32L1xxxx devices, STMicroelectronics offers a STM32L1xx STMTouch sensing library that provides a complete robust C source-code solution. This firmware library is available as a free download from www.st.com. For information how to download the STM32L1xx STMTouch sensing library, please contact your local ST representative.

9 Interrupts and events

This section applies to the whole STM32L1xxxx family, unless otherwise specified.

9.1 Nested vectored interrupt controller (NVIC)

Features

- 45 maskable interrupt channels in low and medium density devices (see *Table 35*), 53 maskable interrupt channels in medium+ density devices (see *Table 36*) and 56 channels in high density devices (see *Table 37*). These do not include the 16 interrupt lines of Cortex™-M3.
- 16 programmable priority levels (4 bits of interrupt priority are used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of system control registers

The NVIC and the processor core interface are closely coupled, which enables low-latency interrupt processing and efficient processing of late arriving interrupts.

All interrupts including the core exceptions are managed by the NVIC. For more information on exceptions and NVIC programming, refer to the PM0056 programming manual.

9.1.1 SysTick calibration value register

The SysTick calibration value is fixed to 4000, which gives a reference time base of 1 ms with the SysTick clock set to 4 MHz (max HCLK/8).

9.1.2 Interrupt and exception vectors

Table 35 is the vector table for STM32L1xxxx devices.

Table 35. Vector table (low and medium density devices)

Position	Priority	Type of priority	Acronym	Description	Address
	-	-	-	Reserved	0x0000_0000
	-3	fixed	Reset	Reset	0x0000_0004
	-2	fixed	NMI_Handler	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000_0008
	-1	fixed	HardFault_Handler	All class of fault	0x0000_000C
	0	settable	MemManage_Handler	Memory management	0x0000_0010
	1	settable	BusFault_Handler	Pre-fetch fault, memory access fault	0x0000_0014
	2	settable	UsageFault_Handler	Undefined instruction or illegal state	0x0000_0018
	-	-	-	Reserved	0x0000_001C - 0x0000_002B

Table 35. Vector table (low and medium density devices) (continued)

Position	Priority	Type of priority	Acronym	Description	Address
	3	settable	SVC_Handler	System service call via SWI instruction	0x0000_002C
	4	settable	DebugMon_Handler	Debug Monitor	0x0000_0030
	-	-	-	Reserved	0x0000_0034
	5	settable	PendSV_Handler	Pendable request for system service	0x0000_0038
	6	settable	SysTick_Handler	System tick timer	0x0000_003C
0	7	settable	WWDG	Window Watchdog interrupt	0x0000_0040
1	8	settable	PVD	PVD through EXTI Line detection interrupt	0x0000_0044
2	9	settable	TAMPER_STAMP	Tamper and TimeStamp through EXTI line interrupts	0x0000_0048
3	10	settable	RTC_WKUP	RTC Wakeup through EXTI line interrupt	0x0000_004C
4	11	settable	FLASH	Flash global interrupt	0x0000_0050
5	12	settable	RCC	RCC global interrupt	0x0000_0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000_0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000_005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000_0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000_0064
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000_0068
11	18	settable	DMA1_Channel1	DMA1 Channel1 global interrupt	0x0000_006C
12	19	settable	DMA1_Channel2	DMA1 Channel2 global interrupt	0x0000_0070
13	20	settable	DMA1_Channel3	DMA1 Channel3 global interrupt	0x0000_0074
14	21	settable	DMA1_Channel4	DMA1 Channel4 global interrupt	0x0000_0078
15	22	settable	DMA1_Channel5	DMA1 Channel5 global interrupt	0x0000_007C
16	23	settable	DMA1_Channel6	DMA1 Channel6 global interrupt	0x0000_0080
17	24	settable	DMA1_Channel7	DMA1 Channel7 global interrupt	0x0000_0084
18	25	settable	ADC1	ADC1 global interrupt	0x0000_0088
19	26	settable	USB HP	USB High priority interrupt	0x0000_008C
20	27	settable	USB_LP	USB Low priority interrupt	0x0000_0090
21	28	settable	DAC	DAC interrupt	0x0000_0094
22	29	settable	COMP	Comparator wakeup through EXTI line (21 and 22) interrupt	0x0000_0098
23	30	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000_009C
24	31	settable	LCD	LCD global interrupt	0x0000_00A0
25	32	settable	TIM9	TIM9 global interrupt	0x0000_00A4

Table 35. Vector table (low and medium density devices) (continued)

Position	Priority	Type of priority	Acronym	Description	Address
26	33	settable	TIM10	TIM10 global interrupt	0x0000_00A8
27	34	settable	TIM11	TIM11 global interrupt	0x0000_00AC
28	35	settable	TIM2	TIM2 global interrupt	0x0000_00B0
29	36	settable	TIM3	TIM3 global interrupt	0x0000_00B4
30	37	settable	TIM4	TIM4 global interrupt	0x0000_00B8
31	38	settable	I2C1_EV	I^2C1 event interrupt	0x0000_00BC
32	39	settable	I2C1_ER	I^2C1 error interrupt	0x0000_00C0
33	40	settable	I2C2_EV	I^2C2 event interrupt	0x0000_00C4
34	41	settable	I2C2_ER	I^2C2 error interrupt	0x0000_00C8
35	42	settable	SPI1	SPI1 global interrupt	0x0000_00CC
36	43	settable	SPI2	SPI2 global interrupt	0x0000_00D0
37	44	settable	USART1	USART1 global interrupt	0x0000_00D4
38	45	settable	USART2	USART2 global interrupt	0x0000_00D8
39	46	settable	USART3	USART3 global interrupt	0x0000_00DC
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000_00E0
41	48	settable	RTC_Alarm	RTC Alarms (A and B) through EXTI line interrupt	0x0000_00E4
42	49	settable	USB_FS_WKUP	USB Device FS Wakeup through EXTI line interrupt	0x0000_00E8
43	50	settable	TIM6	TIM6 global interrupt	0x0000_00EC
44	51	settable	TIM7	TIM7 global interrupt	0x0000_00F0

Table 36. Vector table (medium+ density devices)

Position	Priority	Type of priority	Acronym	Description	Address
	-	-	-	Reserved	0x0000_0000
	-3	fixed	Reset	Reset	0x0000_0004
	-2	fixed	NMI_Handler	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000_0008
	-1	fixed	HardFault_Handler	All class of fault	0x0000_000C
	0	settable	MemManage_Handler	Memory management	0x0000_0010
	1	settable	BusFault_Handler	Pre-fetch fault, memory access fault	0x0000_0014
	2	settable	UsageFault_Handler	Undefined instruction or illegal state	0x0000_0018

Table 36. Vector table (medium+ density devices) (continued)

Position	Priority	Type of priority	Acronym	Description	Address
	-	-	-	Reserved	0x0000_001C - 0x0000_002B
	3	settable	SVC_Handler	System service call via SWI instruction	0x0000_002C
	4	settable	DebugMon_Handler	Debug Monitor	0x0000_0030
	-	-	-	Reserved	0x0000_0034
	5	settable	PendSV_Handler	Pendable request for system service	0x0000_0038
	6	settable	SysTick_Handler	System tick timer	0x0000_003C
0	7	settable	WWDG	Window Watchdog interrupt	0x0000_0040
1	8	settable	PVD	PVD through EXTI Line detection interrupt	0x0000_0044
2	9	settable	TAMPER_STAMP	Tamper and TimeStamp through EXTI line interrupts	0x0000_0048
3	10	settable	RTC_WKUP	RTC Wakeup through EXTI line interrupt	0x0000_004C
4	11	settable	FLASH	Flash global interrupt	0x0000_0050
5	12	settable	RCC	RCC global interrupt	0x0000_0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000_0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000_005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000_0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000_0064
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000_0068
11	18	settable	DMA1_Channel1	DMA1 Channel1 global interrupt	0x0000_006C
12	19	settable	DMA1_Channel2	DMA1 Channel2 global interrupt	0x0000_0070
13	20	settable	DMA1_Channel3	DMA1 Channel3 global interrupt	0x0000_0074
14	21	settable	DMA1_Channel4	DMA1 Channel4 global interrupt	0x0000_0078
15	22	settable	DMA1_Channel5	DMA1 Channel5 global interrupt	0x0000_007C
16	23	settable	DMA1_Channel6	DMA1 Channel6 global interrupt	0x0000_0080
17	24	settable	DMA1_Channel7	DMA1 Channel7 global interrupt	0x0000_0084
18	25	settable	ADC1	ADC1 global interrupt	0x0000_0088
19	26	settable	USB HP	USB High priority interrupt	0x0000_008C
20	27	settable	USB_LP	USB Low priority interrupt	0x0000_0090
21	28	settable	DAC	DAC interrupt	0x0000_0094
22	29	settable	COMP/CA	Comparator wakeup through EXTI line (21 and 22) interrupt/Channel acquisition interrupt	0x0000_0098
23	30	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000_009C

Table 36. Vector table (medium+ density devices) (continued)

Position	Priority	Type of priority	Acronym	Description	Address
24	31	settable	LCD	LCD global interrupt	0x0000_00A0
25	32	settable	TIM9	TIM10 global interrupt	0x0000_00A4
26	33	settable	TIM10	TIM10 global interrupt	0x0000_00A8
27	34	settable	TIM11	TIM11 global interrupt	0x0000_00AC
28	35	settable	TIM2	TIM2 global interrupt	0x0000_00B0
29	36	settable	TIM3	TIM3 global interrupt	0x0000_00B4
30	37	settable	TIM4	TIM4 global interrupt	0x0000_00B8
31	38	settable	I2C1_EV	I^{2}C1 event interrupt	0x0000_00BC
32	39	settable	I2C1_ER	I^{2}C1 error interrupt	0x0000_00C0
33	40	settable	I2C2_EV	I^{2}C2 event interrupt	0x0000_00C4
34	41	settable	I2C2_ER	I^{2}C2 error interrupt	0x0000_00C8
35	42	settable	SPI1	SPI1 global interrupt	0x0000_00CC
36	43	settable	SPI2	SPI2 global interrupt	0x0000_00D0
37	44	settable	USART1	USART1 global interrupt	0x0000_00D4
38	45	settable	USART2	USART2 global interrupt	0x0000_00D8
39	46	settable	USART3	USART3 global interrupt	0x0000_00DC
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000_00E0
41	48	settable	RTC_Alarm	RTC Alarms (A and B) through EXTI line interrupt	0x0000_00E4
42	49	settable	USB_FS_WKUP	USB Device FS Wakeup through EXTI line interrupt	0x0000_00E8
43	50	settable	TIM6	TIM6 global interrupt	0x0000_00EC
44	51	settable	TIM7	TIM7 global interrupt	0x0000_00F0
45	53	settable	TIM5	TIM5 Global interrupt	0x0000_00F8
46	54	settable	SPI3	SPI3 Global interrupt	0x0000_00FC
47	57	settable	DMA2_CH1	DMA2 Channel 1 interrupt	0x0000_0108
48	58	settable	DMA2_CH2	DMA2 Channel 2 interrupt	0x0000_010C
49	59	settable	DMA2_CH3	DMA2 Channel 3 interrupt	0x0000_0110
50	60	settable	DMA2_CH4	DMA2 Channel 4 interrupt	0x0000_0114
51	61	settable	DMA2_CH5	DMA2 Channel 5 interrupt	0x0000_0118
52	62	settable	AES	AES global interrupt	0x0000_011C
53	63	settable	COMP_ACQ	Comparator Channel Acquisition Interrupt	0x0000_0120

Table 37. Vector table (high-density devices)

Position	Priority	Type of priority	Acronym	Description	Address
	-	-	-	Reserved	0x0000_0000
	-3	fixed	Reset	Reset	0x0000_0004
	-2	fixed	NMI_Handler	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000_0008
	-1	fixed	HardFault_Handler	All class of fault	0x0000_000C
	0	settable	MemManage_Handler	Memory management	0x0000_0010
	1	settable	BusFault_Handler	Pre-fetch fault, memory access fault	0x0000_0014
	2	settable	UsageFault_Handler	Undefined instruction or illegal state	0x0000_0018
	-	-	-	Reserved	0x0000_001C - 0x0000_002B
	3	settable	SVC_Handler	System service call via SWI instruction	0x0000_002C
	4	settable	DebugMon_Handler	Debug Monitor	0x0000_0030
	-	-	-	Reserved	0x0000_0034
	5	settable	PendSV_Handler	Pendable request for system service	0x0000_0038
	6	settable	SysTick_Handler	System tick timer	0x0000_003C
0	7	settable	WWDG	Window Watchdog interrupt	0x0000_0040
1	8	settable	PVD	PVD through EXTI Line detection interrupt	0x0000_0044
2	9	settable	TAMPER_STAMP	Tamper and TimeStamp through EXTI line interrupts	0x0000_0048
3	10	settable	RTC_WKUP	RTC Wakeup through EXTI line interrupt	0x0000_004C
4	11	settable	FLASH	Flash global interrupt	0x0000_0050
5	12	settable	RCC	RCC global interrupt	0x0000_0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000_0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000_005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000_0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000_0064
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000_0068
11	18	settable	DMA1_Channel1	DMA1 Channel1 global interrupt	0x0000_006C
12	19	settable	DMA1_Channel2	DMA1 Channel2 global interrupt	0x0000_0070
13	20	settable	DMA1_Channel3	DMA1 Channel3 global interrupt	0x0000_0074
14	21	settable	DMA1_Channel4	DMA1 Channel4 global interrupt	0x0000_0078
15	22	settable	DMA1_Channel5	DMA1 Channel5 global interrupt	0x0000_007C

Table 37. Vector table (high-density devices) (continued)

Position	Priority	Type of priority	Acronym	Description	Address
16	23	settable	DMA1_Channel6	DMA1 Channel6 global interrupt	0x0000_0080
17	24	settable	DMA1_Channel7	DMA1 Channel7 global interrupt	0x0000_0084
18	25	settable	ADC1	ADC1 global interrupt	0x0000_0088
19	26	settable	USB HP	USB High priority interrupt	0x0000_008C
20	27	settable	USB_LP	USB Low priority interrupt	0x0000_0090
21	28	settable	DAC	DAC interrupt	0x0000_0094
22	29	settable	COMP/CA	Comparator wakeup through EXTI line (21 and 22) interrupt/Channel acquisition interrupt	0x0000_0098
23	30	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000_009C
24	31	settable	LCD	LCD global interrupt	0x0000_00A0
25	32	settable	TIM9	TIM10 global interrupt	0x0000_00A4
26	33	settable	TIM10	TIM10 global interrupt	0x0000_00A8
27	34	settable	TIM11	TIM11 global interrupt	0x0000_00AC
28	35	settable	TIM2	TIM2 global interrupt	0x0000_00B0
29	36	settable	TIM3	TIM3 global interrupt	0x0000_00B4
30	37	settable	TIM4	TIM4 global interrupt	0x0000_00B8
31	38	settable	I2C1_EV	I^{2}C1 event interrupt	0x0000_00BC
32	39	settable	I2C1_ER	I^{2}C1 error interrupt	0x0000_00C0
33	40	settable	I2C2_EV	I^{2}C2 event interrupt	0x0000_00C4
34	41	settable	I2C2_ER	I^{2}C2 error interrupt	0x0000_00C8
35	42	settable	SPI1	SPI1 global interrupt	0x0000_00CC
36	43	settable	SPI2	SPI2 global interrupt	0x0000_00D0
37	44	settable	USART1	USART1 global interrupt	0x0000_00D4
38	45	settable	USART2	USART2 global interrupt	0x0000_00D8
39	46	settable	USART3	USART3 global interrupt	0x0000_00DC
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000_00E0
41	48	settable	RTC_Alarm	RTC Alarms (A and B) through EXTI line interrupt	0x0000_00E4
42	49	settable	USB_FS_WKUP	USB Device FS Wakeup through EXTI line interrupt	0x0000_00E8
43	50	settable	TIM6	TIM6 global interrupt	0x0000_00EC
44	51	settable	TIM7	TIM7 global interrupt	0x0000_00F0
45	52	settable	SDIO	SDIO Global interrupt	0x0000_00F4
46	53	settable	TIM5	TIM5 Global interrupt	0x0000_00F8

Table 37. Vector table (high-density devices) (continued)

Position	Priority	Type of priority	Acronym	Description	Address
47	54	settable	SPI3	SPI3 Global interrupt	0x0000_00FC
48	55	settable	UART4	UART4 Global interrupt	0x0000_0100
49	56	settable	UART5	UART5 Global interrupt	0x0000_0104
50	57	settable	DMA2_CH1	DMA2 Channel 1 interrupt	0x0000_0108
51	58	settable	DMA2_CH2	DMA2 Channel 2 interrupt	0x0000_010C
52	59	settable	DMA2_CH3	DMA2 Channel 3 interrupt	0x0000_0110
53	60	settable	DMA2_CH4	DMA2 Channel 4 interrupt	0x0000_0114
54	61	settable	DMA2_CH5	DMA2 Channel 5 interrupt	0x0000_0118
55	62	settable	AES	AES global interrupt	0x0000_011C
56	63	settable	COMP_ACQ	Comparator Channel Acquisition Interrupt	0x0000_0120

9.2 External interrupt/event controller (EXTI)

The external interrupt/event controller consists of up to 24 (or 23 for low and medium density devices) edge detectors for generating event/interrupt requests. Each input line can be independently configured to select the type (event or interrupt) and the corresponding trigger event (rising edge, falling edge or both). Each line can also be masked independently. A pending register maintains the status line of the interrupt requests.

9.2.1 Main features

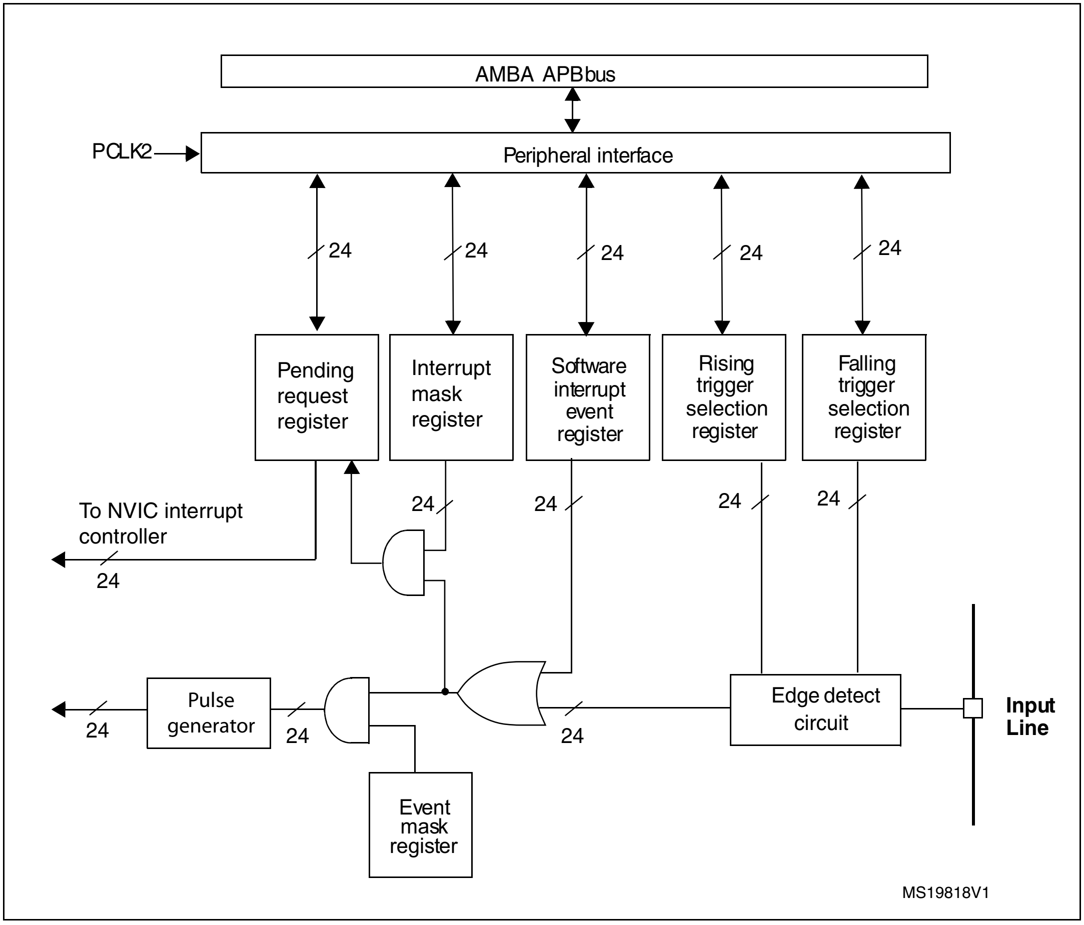
The main features of the EXTI controller are the following:

- Independent trigger and mask on each interrupt/event line
- Dedicated status bit for each interrupt line
- Generation of up to 24 (or 23 for low and medium density devices) software event/interrupt requests
- Detection of external signals with a pulse width lower than the APB2 clock period. Refer to the electrical characteristics section of the STM32L1xxxx datasheet for details on this parameter.

9.2.2 Block diagram

The block diagram is shown in *Figure 29*.

Figure 29. External interrupt/event controller block diagram



9.2.3 Wakeup event management

The STM32L1xxxx is able to handle external or internal events in order to wake up the core (WFE). The wakeup event can be generated by either:

- enabling an interrupt in the peripheral control register but not in the NVIC, and enabling the SEVONPEND bit in the Cortex™-M3 system control register. When the MCU resumes from WFE, the peripheral interrupt pending bit and the peripheral NVIC IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.
- or configuring an external or internal EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bit corresponding to the event line is not set.

To use an external line as a wakeup event, refer to *Section 9.2.4: Functional description*.

9.2.4 Functional description

To generate the interrupt, the interrupt line should be configured and enabled. This is done by programming the two trigger registers with the desired edge detection and by enabling the interrupt request by writing a ‘1 to the corresponding bit in the interrupt mask register. When the selected edge occurs on the external interrupt line, an interrupt request is generated. The pending bit corresponding to the interrupt line is also set. This request is reset by writing a ‘1 into the pending register.

To generate the event, the event line should be configured and enabled. This is done by programming the two trigger registers with the desired edge detection and by enabling the event request by writing a ‘1 to the corresponding bit in the event mask register. When the selected edge occurs on the event line, an event pulse is generated. The pending bit corresponding to the event line is not set

An interrupt/event request can also be generated by software by writing a ‘1 into the software interrupt/event register.

Hardware interrupt selection

To configure the 24 (or 23 for low and medium density devices) lines as interrupt sources, use the following procedure:

- Configure the mask bits of the Interrupt lines (EXTI_IMR)
- Configure the Trigger Selection bits of the Interrupt lines (EXTI_RTSR and EXTI_FTSR)
- Configure the enable and mask bits that control the NVIC IRQ channel mapped to the external interrupt controller (EXTI) so that an interrupt coming from any one of the lines can be correctly acknowledged.

Hardware event selection

To configure the 24 (or 23 for low and medium density devices) lines as event sources, use the following procedure:

- Configure the mask bits of the Event lines (EXTI_EMR)
- Configure the Trigger Selection bits of the Event lines (EXTI_RTSR and EXTI_FTSR)

Software interrupt/event selection

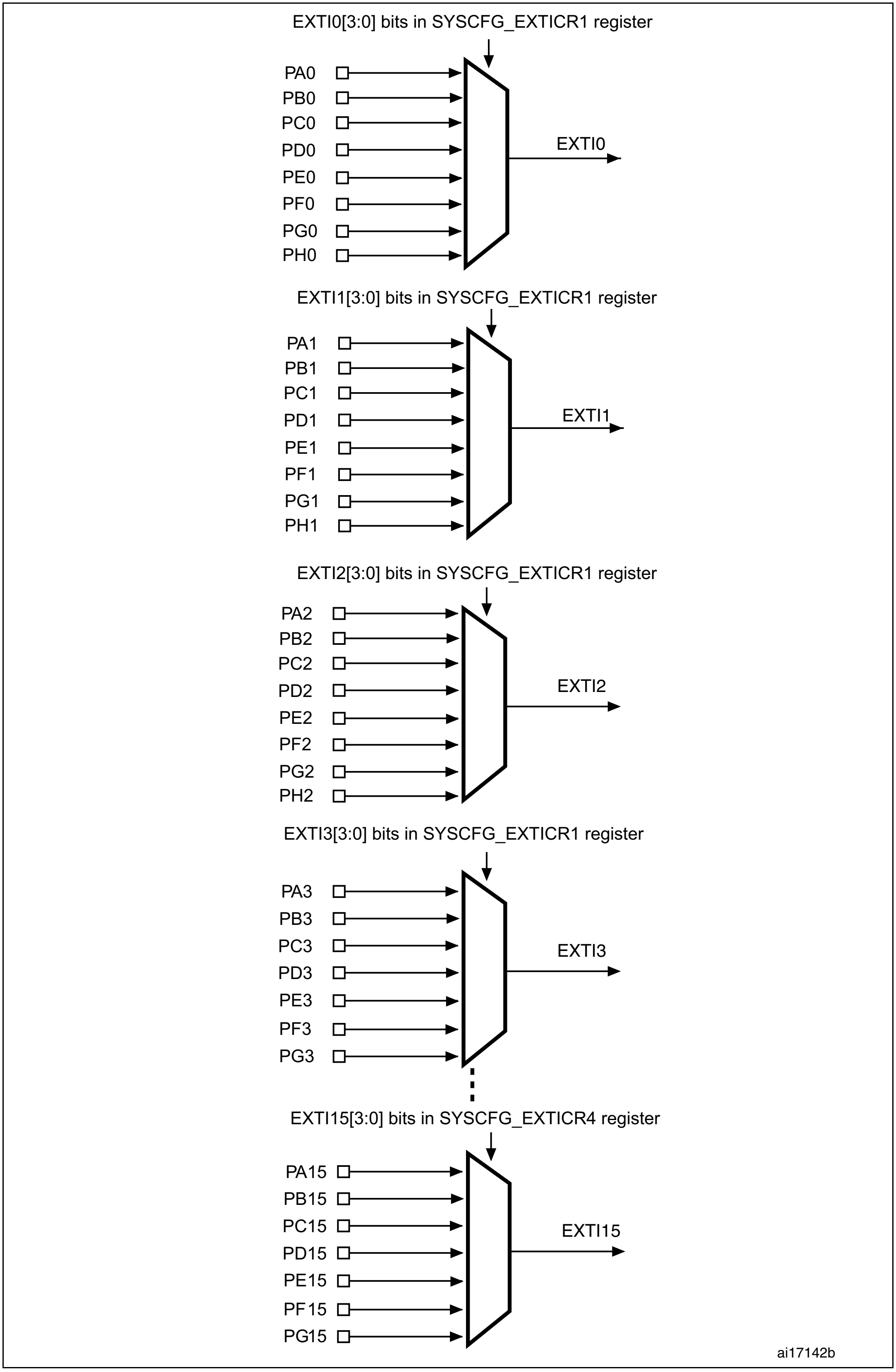
The 24 (or 23 for low and medium density devices) lines can be configured as software interrupt/event lines. The following is the procedure to generate a software interrupt.

- Configure the mask bits of the Interrupt/Event lines (EXTI_IMR, EXTI_EMR)
- Set the required bit in the software interrupt register (EXTI_SWIER)

9.2.5 External interrupt/event line mapping

Up to 116 GPIOs are connected to the 16 external interrupt/event lines in the following manner:

Figure 30. External interrupt/event GPIO mapping


EXTI0[3:0] bits in SYSCFG_EXTICR1 register
PA0
PB0
PC0
PD0
PE0
PF0
PG0
PH0
EXTI0
EXTI1[3:0] bits in SYSCFG_EXTICR1 register
PA1
PB1
PC1
PD1
PE1
PF1
PG1
PH1
EXTI1
EXTI2[3:0] bits in SYSCFG_EXTICR1 register
PA2
PB2
PC2
PD2
PE2
PF2
PG2
PH2
EXTI2
EXTI3[3:0] bits in SYSCFG_EXTICR1 register
PA3
PB3
PC3
PD3
PE3
PF3
PG3
EXTI3
EXTI15[3:0] bits in SYSCFG_EXTICR4 register
PA15
PB15
PC15
PD15
PE15
PF15
PG15
EXTI15
ai17142b

The other EXTI lines are connected as follows:

- EXTI line 16 is connected to the PVD output
- EXTI line 17 is connected to the RTC Alarm event
- EXTI line 18 is connected to the USB Device FS wakeup event
- EXTI line 19 is connected to the RTC Tamper and TimeStamp events
- EXTI line 20 is connected to the RTC Wakeup event
- EXTI line 21 is connected to the Comparator 1 wakeup event
- EXTI line 22 is connected to the Comparator 2 wakeup event
- EXTI line 23 is connected to the channel acquisition interrupt

9.3 EXTI registers

Refer to *Section 1.1* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

9.3.1 EXTI interrupt mask register (EXTI_IMR)

Address offset: 0x00
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								MR23[1]	MR22	MR21	MR20	MR19	MR18	MR17	MR16
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

1. Only in medium+ and high density devices

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

Bits 23:0 **MRx:** Interrupt mask on line x
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is not masked

9.3.2 EXTI event mask register (EXTI_EMR)

Address offset: 0x04
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								MR23[1]	MR22	MR21	MR20	MR19	MR18	MR17	MR16
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

1. Only in medium+ and high density devices

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

Bits 23:0 **MRx:** Event mask on line x
0: Event request from Line x is masked
1: Event request from Line x is not masked

9.3.3 EXTI rising edge trigger selection register (EXTI_RTSR)

Address offset: 0x08
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								TR23[1]	TR22	TR21	TR20	TR19	TR18	TR17	TR16
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

1. Only in medium+ and high density devices

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

Bits 23:0 **TRx:** Rising edge trigger event configuration bit of line x
0: Rising edge trigger disabled (for Event and Interrupt) for input line x
1: Rising edge trigger enabled (for Event and Interrupt) for input line x

Note: *The external wakeup lines are edge triggered, no glitch must be generated on these lines. If a rising edge on the external interrupt line occurs while writing to the EXTI_RTSR register, the pending bit will not be set.*

Rising and falling edge triggers can be set for the same interrupt line. In this configuration, both generate a trigger condition.

9.3.4 Falling edge trigger selection register (EXTI_FTSR)

Address offset: 0x0C
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								TR23[1]	TR22	TR21	TR20	TR19	TR18	TR17	TR16
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

1. Only in medium+ and high density devices

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

Bits 23:0 **TRx:** Falling edge trigger event configuration bit of line x
0: Falling edge trigger disabled (for Event and Interrupt) for input line x
1: Falling edge trigger enabled (for Event and Interrupt) for input line x

Note: The external wakeup lines are edge triggered, no glitch must be generated on these lines. If a falling edge on the external interrupt line occurs while writing to the EXTI_FTSR register, the pending bit will not be set.

Rising and falling edge triggers can be set for the same interrupt line. In this configuration, both generate a trigger condition.

9.3.5 EXTI software interrupt event register (EXTI_SWIER)

Address offset: 0x10
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								SWIER 23[1]	SWIER 22	SWIER 21	SWIER 20	SWIER 19	SWIER 18	SWIER 17	SWIER 16
								rw	rw	rw	rw	rw	rw	rw	rw

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

[1] Only in medium+ and high density devices

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

Bits 22:0 **SWIERx:** Software interrupt on line x
Writing a 1 to this bit when it is at 0 sets the corresponding pending bit in EXTI_PR. If the interrupt is enabled on this line in EXTI_IMR and EXTI_EMR, an interrupt request is generated.
This bit is cleared by clearing the corresponding bit in EXTI_PR (by writing a 1 to this bit).

9.3.6 EXTI pending register (EXTI_PR)

Address offset: 0x14
Reset value: undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								PR23[1]	PR22	PR21	PR20	PR19	PR18	PR17	PR16
								rw	rw	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
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

[1] Only in medium+ and high density devices

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

Bits 23:0 **PRx:** Pending bit
0: No trigger request occurred
1: The selected trigger request occurred
This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing it to 1 or by changing the sensitivity of the edge detector.

9.3.7 EXTI register map

The following table gives the EXTI register map and the reset values.

Table 38. External interrupt/event controller register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	EXTI_IMR	Reserved								MR[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
0x04	EXTI_EMR	Reserved								MR[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
0x08	EXTI_RTSR	Reserved								TR[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
0x0C	EXTI_FTSR	Reserved								TR[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
0x10	EXTI_SWIER	Reserved								SWIER[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

Table 38. External interrupt/event controller register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x14	**EXTI_PR**	Reserved								PR[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

Refer to *Table 2 on page 42* for the register boundary addresses.

10 Direct memory access controller (DMA)

10.1 DMA introduction

Direct memory access (DMA) is used in order to provide high-speed data transfer between peripherals and memory as well as memory to memory. Data can be quickly moved by DMA without any CPU actions. This keeps CPU resources free for other operations.

The two DMA controllers have 12 channels in total (7 for DMA1 and 5 for DMA2), each dedicated to managing memory access requests from one or more peripherals. It has an arbiter for handling the priority between DMA requests.

10.2 DMA main features

- 12 independently configurable channels (requests): 7 for DMA1 and 5 for DMA2
- Each of the 12 channels is connected to dedicated hardware DMA requests, software trigger is also supported on each channel. This configuration is done by software.
- Priorities between requests from channels of one DMA are software programmable (4 levels consisting of *very high*, *high*, *medium*, *low*) or hardware in case of equality (request 1 has priority over request 2, etc.)
- Independent source and destination transfer size (byte, half word, word), emulating packing and unpacking. Source/destination addresses must be aligned on the data size.
- Support for circular buffer management
- 3 event flags (DMA Half Transfer, DMA Transfer complete and DMA Transfer Error) logically ORed together in a single interrupt request for each channel
- Memory-to-memory transfer
- Peripheral-to-memory and memory-to-peripheral, and peripheral-to-peripheral transfers
- Access to Flash, SRAM, APB1, APB2 and AHB peripherals as source and destination
- Programmable number of data to be transferred: up to 65536

The block diagram is shown in *Figure 31*.

Figure 31. DMA block diagram in low and medium density STM32L1xxxx devices

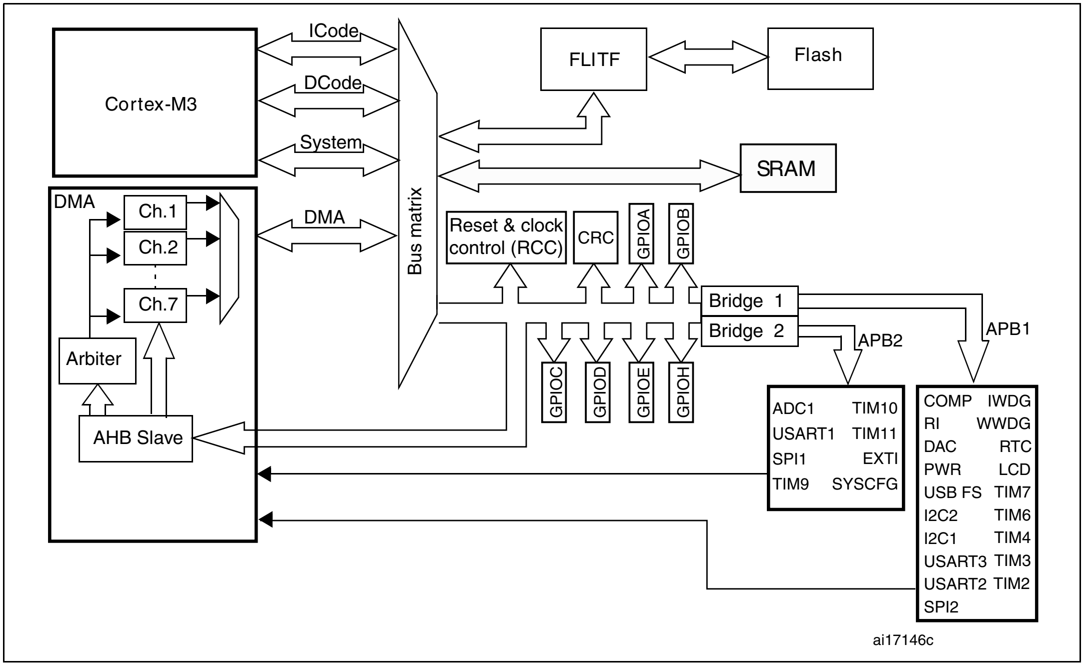


Figure 32. DMA block diagram in medium+ density STM32L1xxxx devices

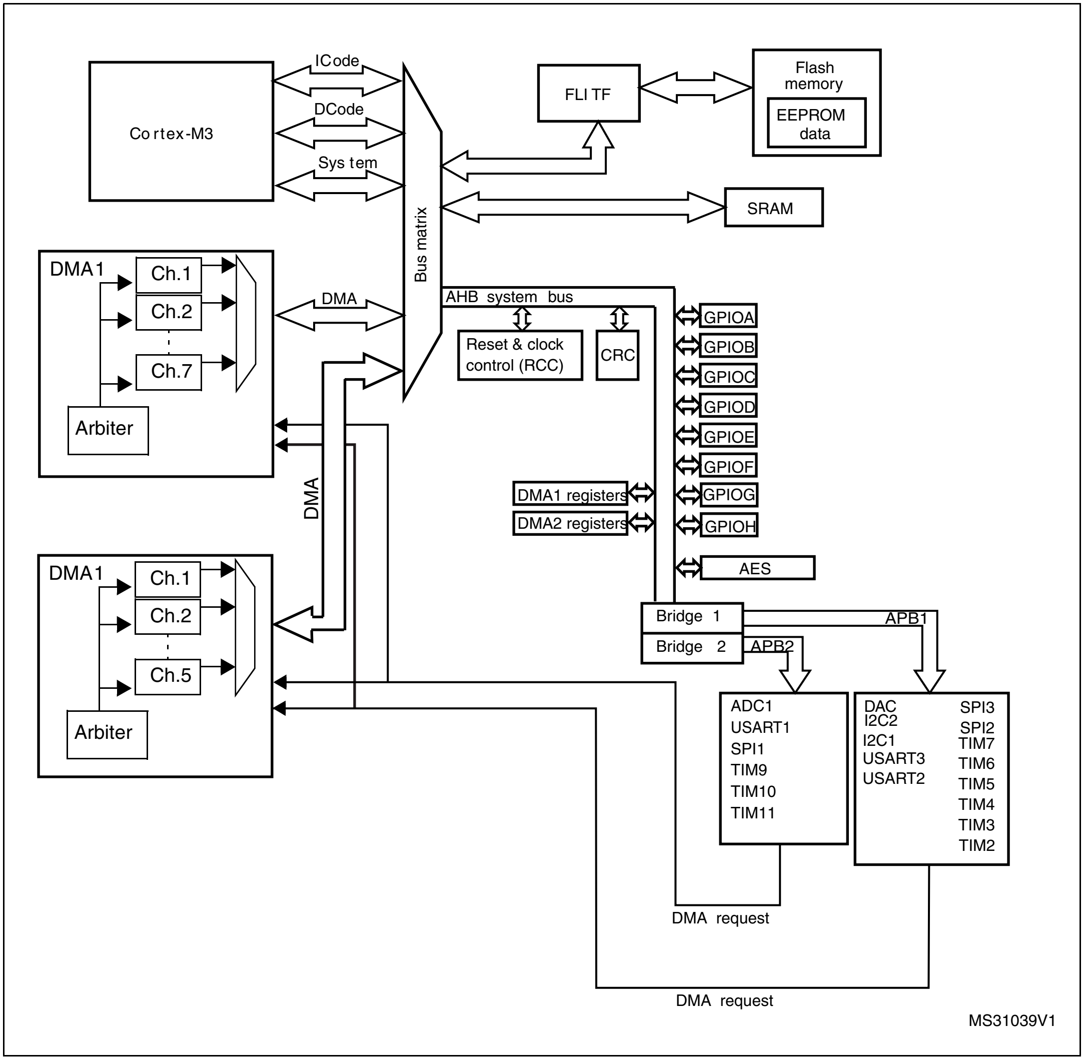
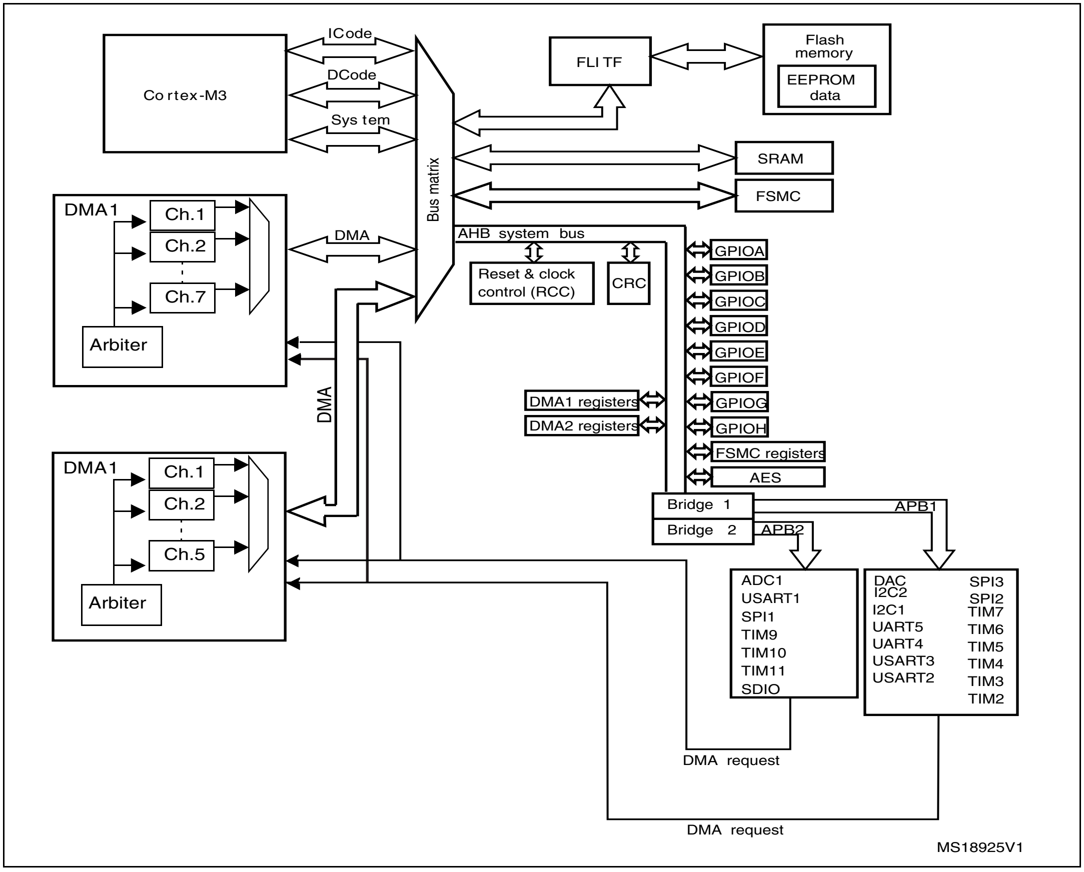


Figure 33. DMA block diagram in high-density STM32L1xxxx devices



Note: The DMA2 controller and its related requests are available only in high and medium+ density devices.

SPI3 and TIM5 DMA requests are available only in high and medium+ density devices.

UART4, UART5 and SDIO are available only in high density devices.

10.3 DMA functional description

The DMA controller performs direct memory transfer by sharing the system bus with the Cortex™-M3 core. The DMA request may stop the CPU access to the system bus for some bus cycles, when the CPU and DMA are targeting the same destination (memory or peripheral). The bus matrix implements round-robin scheduling, thus ensuring at least half of the system bus bandwidth (both to memory and peripheral) for the CPU.

10.3.1 DMA transactions

After an event, the peripheral sends a request signal to the DMA Controller. The DMA controller serves the request depending on the channel priorities. As soon as the DMA Controller accesses the peripheral, an Acknowledge is sent to the peripheral by the DMA Controller. The peripheral releases its request as soon as it gets the Acknowledge from the DMA Controller. Once the request is deasserted by the peripheral, the DMA Controller release the Acknowledge. If there are more requests, the peripheral can initiate the next transaction.

In summary, each DMA transfer consists of three operations:

- The loading of data from the peripheral data register or a location in memory addressed through an internal current peripheral/memory address register. The start address used for the first transfer is the base peripheral/memory address programmed in the DMA_CPARx or DMA_CMARx register
- The storage of the data loaded to the peripheral data register or a location in memory addressed through an internal current peripheral/memory address register. The start address used for the first transfer is the base peripheral/memory address programmed in the DMA_CPARx or DMA_CMARx register
- The post-decrementing of the DMA_CNDTRx register, which contains the number of transactions that have still to be performed.

10.3.2 Arbiter

The arbiter manages the channel requests based on their priority and launches the peripheral/memory access sequences.

The priorities are managed in two stages:

- Software: each channel priority can be configured in the DMA_CCRx register. There are four levels:
  - Very high priority
  - High priority
  - Medium priority
  - Low priority
- Hardware: if 2 requests have the same software priority level, the channel with the lowest number will get priority versus the channel with the highest number. For example, channel 2 gets priority over channel 4.

10.3.3 DMA channels

Each channel can handle DMA transfer between a peripheral register located at a fixed address and a memory address. The amount of data to be transferred (up to 65535) is programmable. The register which contains the amount of data items to be transferred is decremented after each transaction.

Programmable data sizes

Transfer data sizes of the peripheral and memory are fully programmable through the PSIZE and MSIZE bits in the DMA_CCRx register.

Pointer incrementation

Peripheral and memory pointers can optionally be automatically post-incremented after each transaction depending on the PINC and MINC bits in the DMA_CCRx register. If incremented mode is enabled, the address of the next transfer will be the address of the previous one incremented by 1, 2 or 4 depending on the chosen data size. The first transfer address is the one programmed in the DMA_CPARx/DMA_CMARx registers. During transfer operations, these registers keep the initially programmed value. The current transfer addresses (in the current internal peripheral/memory address register) are not accessible by software.

If the channel is configured in noncircular mode, no DMA request is served after the last transfer (that is once the number of data items to be transferred has reached zero). In order to reload a new number of data items to be transferred into the DMA_CNDTRx register, the DMA channel must be disabled.

Note: If a DMA channel is disabled, the DMA registers are not reset. The DMA channel registers (DMA_CCRx, DMA_CPARx and DMA_CMARx) retain the initial values programmed during the channel configuration phase.

In circular mode, after the last transfer, the DMA_CNDTRx register is automatically reloaded with the initially programmed value. The current internal address registers are reloaded with the base address values from the DMA_CPARx/DMA_CMARx registers.

Channel configuration procedure

The following sequence should be followed to configure a DMA channelx (where x is the channel number).

1. Set the peripheral register address in the DMA_CPARx register. The data will be moved from/ to this address to/ from the memory after the peripheral event.
2. Set the memory address in the DMA_CMARx register. The data will be written to or read from this memory after the peripheral event.
3. Configure the total number of data to be transferred in the DMA_CNDTRx register. After each peripheral event, this value will be decremented.
4. Configure the channel priority using the PL[1:0] bits in the DMA_CCRx register
5. Configure data transfer direction, circular mode, peripheral & memory incremented mode, peripheral & memory data size, and interrupt after half and/or full transfer in the DMA_CCRx register
6. Activate the channel by setting the ENABLE bit in the DMA_CCRx register.

As soon as the channel is enabled, it can serve any DMA request from the peripheral connected on the channel.

Once half of the bytes are transferred, the half-transfer flag (HTIF) is set and an interrupt is generated if the Half-Transfer Interrupt Enable bit (HTIE) is set. At the end of the transfer, the Transfer Complete Flag (TCIF) is set and an interrupt is generated if the Transfer Complete Interrupt Enable bit (TCIE) is set.

Circular mode

Circular mode is available to handle circular buffers and continuous data flows (e.g. ADC scan mode). This feature can be enabled using the CIRC bit in the DMA_CCRx register. When circular mode is activated, the number of data to be transferred is automatically reloaded with the initial value programmed during the channel configuration phase, and the DMA requests continue to be served.

Memory-to-memory mode

The DMA channels can also work without being triggered by a request from a peripheral. This mode is called Memory to Memory mode.

If the MEM2MEM bit in the DMA_CCRx register is set, then the channel initiates transfers as soon as it is enabled by software by setting the Enable bit (EN) in the DMA_CCRx register. The transfer stops once the DMA_CNDTRx register reaches zero. Memory to Memory mode may not be used at the same time as Circular mode.

10.3.4 Programmable data width, data alignment and endians

When PSIZE and MSIZE are not equal, the DMA performs some data alignments as described in *Table 39: Programmable data width & endian behavior (when bits PINC = MINC = 1)*.

Table 39. Programmable data width & endian behavior (when bits PINC = MINC = 1)

Source port width	Desti-nation port width	Number of data items to transfer (NDT)	Source content: address / data	Transfer operations	Destination content: address / data
8	8	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: READ B0[7:0] @0x0 then WRITE B0[7:0] @0x0 2: READ B1[7:0] @0x1 then WRITE B1[7:0] @0x1 3: READ B2[7:0] @0x2 then WRITE B2[7:0] @0x2 4: READ B3[7:0] @0x3 then WRITE B3[7:0] @0x3	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3
8	16	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: READ B0[7:0] @0x0 then WRITE 00B0[15:0] @0x0 2: READ B1[7:0] @0x1 then WRITE 00B1[15:0] @0x2 3: READ B3[7:0] @0x2 then WRITE 00B2[15:0] @0x4 4: READ B4[7:0] @0x3 then WRITE 00B3[15:0] @0x6	@0x0 / 00B0 @0x2 / 00B1 @0x4 / 00B2 @0x6 / 00B3
8	32	4	@0x0 / B0 @0x1 / B1 @0x2 / B2 @0x3 / B3	1: READ B0[7:0] @0x0 then WRITE 000000B0[31:0] @0x0 2: READ B1[7:0] @0x1 then WRITE 000000B1[31:0] @0x4 3: READ B3[7:0] @0x2 then WRITE 000000B2[31:0] @0x8 4: READ B4[7:0] @0x3 then WRITE 000000B3[31:0] @0xC	@0x0 / 000000B0 @0x4 / 000000B1 @0x8 / 000000B2 @0xC / 000000B3
16	8	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: READ B1B0[15:0] @0x0 then WRITE B0[7:0] @0x0 2: READ B3B2[15:0] @0x2 then WRITE B2[7:0] @0x1 3: READ B5B4[15:0] @0x4 then WRITE B4[7:0] @0x2 4: READ B7B6[15:0] @0x6 then WRITE B6[7:0] @0x3	@0x0 / B0 @0x1 / B2 @0x2 / B4 @0x3 / B6
16	16	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: READ B1B0[15:0] @0x0 then WRITE B1B0[15:0] @0x0 2: READ B3B2[15:0] @0x2 then WRITE B3B2[15:0] @0x2 3: READ B5B4[15:0] @0x4 then WRITE B5B4[15:0] @0x4 4: READ B7B6[15:0] @0x6 then WRITE B7B6[15:0] @0x6	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6
16	32	4	@0x0 / B1B0 @0x2 / B3B2 @0x4 / B5B4 @0x6 / B7B6	1: READ B1B0[15:0] @0x0 then WRITE 0000B1B0[31:0] @0x0 2: READ B3B2[15:0] @0x2 then WRITE 0000B3B2[31:0] @0x4 3: READ B5B4[15:0] @0x4 then WRITE 0000B5B4[31:0] @0x8 4: READ B7B6[15:0] @0x6 then WRITE 0000B7B6[31:0] @0xC	@0x0 / 0000B1B0 @0x4 / 0000B3B2 @0x8 / 0000B5B4 @0xC / 0000B7B6
32	8	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: READ B3B2B1B0[31:0] @0x0 then WRITE B0[7:0] @0x0 2: READ B7B6B5B4[31:0] @0x4 then WRITE B4[7:0] @0x1 3: READ BBBAB9B8[31:0] @0x8 then WRITE B8[7:0] @0x2 4: READ BFBEBDBC[31:0] @0xC then WRITE BC[7:0] @0x3	@0x0 / B0 @0x1 / B4 @0x2 / B8 @0x3 / BC
32	16	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: READ B3B2B1B0[31:0] @0x0 then WRITE B1B0[7:0] @0x0 2: READ B7B6B5B4[31:0] @0x4 then WRITE B5B4[7:0] @0x1 3: READ BBBAB9B8[31:0] @0x8 then WRITE B9B8[7:0] @0x2 4: READ BFBEBDBC[31:0] @0xC then WRITE BDBC[7:0] @0x3	@0x0 / B1B0 @0x2 / B5B4 @0x4 / B9B8 @0x6 / BDBC
32	32	4	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC	1: READ B3B2B1B0[31:0] @0x0 then WRITE B3B2B1B0[31:0] @0x0 2: READ B7B6B5B4[31:0] @0x4 then WRITE B7B6B5B4[31:0] @0x4 3: READ BBBAB9B8[31:0] @0x8 then WRITE BBBAB9B8[31:0] @0x8 4: READ BFBEBDBC[31:0] @0xC then WRITE BFBEBDBC[31:0] @0xC	@0x0 / B3B2B1B0 @0x4 / B7B6B5B4 @0x8 / BBBAB9B8 @0xC / BFBEBDBC

Addressing an AHB peripheral that does not support byte or halfword write operations

When the DMA initiates an AHB byte or halfword write operation, the data are duplicated on the unused lanes of the HWDATA[31:0] bus. So when the used AHB slave peripheral does not support byte or halfword write operations (when HSIZE is not used by the peripheral) *and* does not generate any error, the DMA writes the 32 HWDATA bits as shown in the two examples below:

- To write the halfword “0xABCD”, the DMA sets the HWDATA bus to “0xABCDABCD” with HSIZE = HalfWord
- To write the byte “0xAB”, the DMA sets the HWDATA bus to “0xABABABAB” with HSIZE = Byte

Assuming that the AHB/APB bridge is an AHB 32-bit slave peripheral that does not take the HSIZE data into account, it will transform any AHB byte or halfword operation into a 32-bit APB operation in the following manner:

- an AHB byte write operation of the data “0xB0” to 0x0 (or to 0x1, 0x2 or 0x3) will be converted to an APB word write operation of the data “0xB0B0B0B0” to 0x0
- an AHB halfword write operation of the data “0xB1B0” to 0x0 (or to 0x2) will be converted to an APB word write operation of the data “0xB1B0B1B0” to 0x0

For instance, if you want to write the APB backup registers (16-bit registers aligned to a 32-bit address boundary), you must configure the memory source size (MSIZE) to “16-bit” and the peripheral destination size (PSIZE) to “32-bit”.

10.3.5 Error management

A DMA transfer error can be generated by reading from or writing to a reserved address space. When a DMA transfer error occurs during a DMA read or a write access, the faulty channel is automatically disabled through a hardware clear of its EN bit in the corresponding Channel configuration register (DMA_CCRx). The channel's transfer error interrupt flag (TEIF) in the DMA_IFR register is set and an interrupt is generated if the transfer error interrupt enable bit (TEIE) in the DMA_CCRx register is set.

10.3.6 Interrupts

An interrupt can be produced on a Half-transfer, Transfer complete or Transfer error for each DMA channel. Separate interrupt enable bits are available for flexibility.

Table 40. DMA interrupt requests

Interrupt event	Event flag	Enable Control bit
Half-transfer	HTIF	HTIE
Transfer complete	TCIF	TCIE
Transfer error	TEIF	TEIE

10.3.7 DMA request mapping

DMA1 controller

The 7 requests from the peripherals (TIMx[2,3,4,6,7], ADC1, SPI[1,2], I2Cx[1,2], USARTx[1,2,3]) and DAC Channelx[1,2] are simply logically ORed before entering the DMA1, this means that only one request must be enabled at a time. Refer to *Figure 34: DMA1 request mapping*.

The peripheral DMA requests can be independently activated/de-activated by programming the DMA control bit in the registers of the corresponding peripheral.

Figure 34. DMA1 request mapping

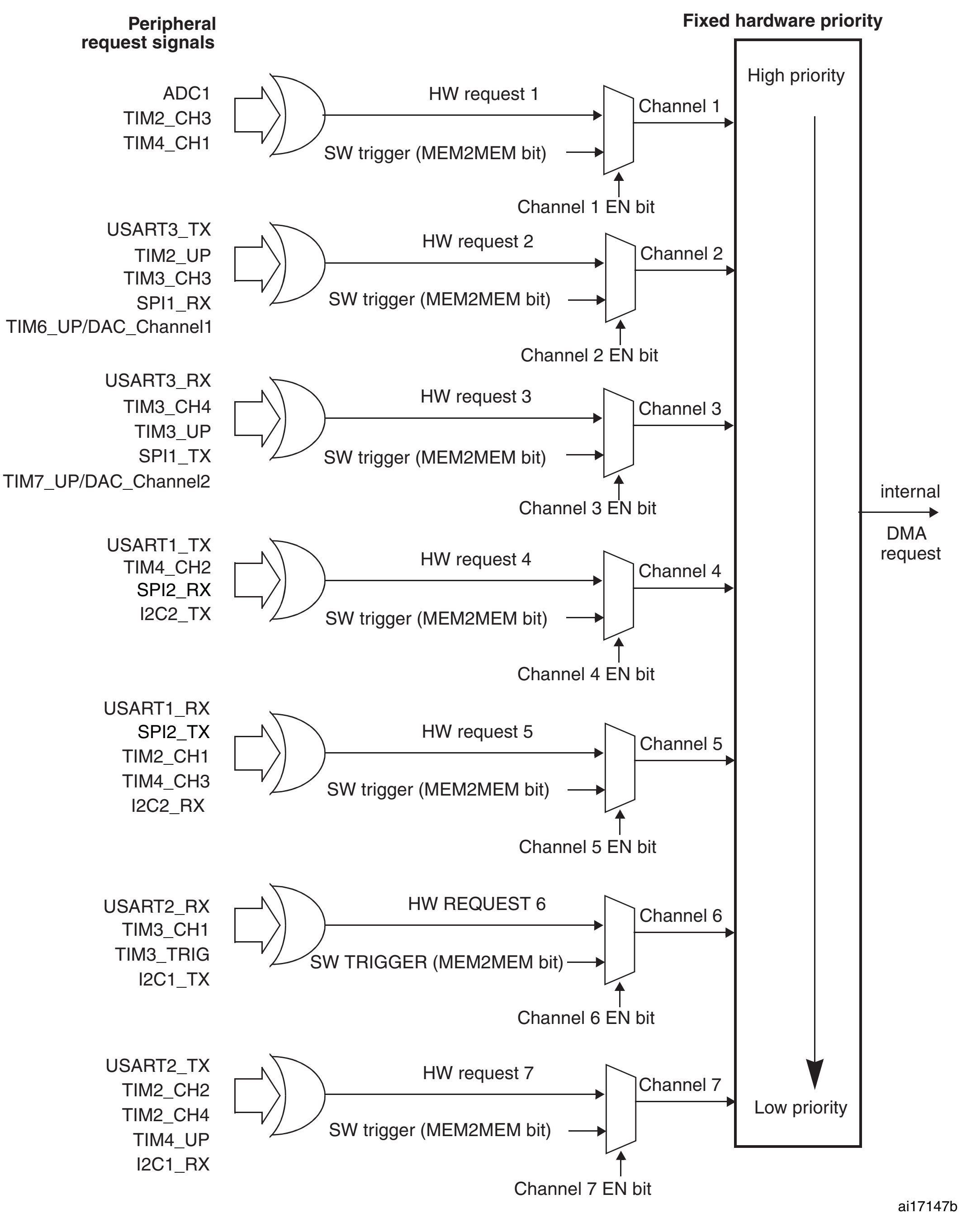


Table 41 lists the DMA requests for each channel.

Table 41. Summary of DMA1 requests for each channel

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7
ADC1	ADC1						
SPI		SPI1_RX	SPI1_TX	SPI2_RX	SPI2_TX		
USART		USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I^{2}C				I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TIM2	TIM2_CH3	TIM2_UP			TIM2_CH1		TIM2_CH2 TIM2_CH4
TIM3		TIM3_CH3	TIM3_CH4 TIM3_UP			TIM3_CH1 TIM3_TRIG	
TIM4	TIM4_CH1			TIM4_CH2	TIM4_CH3		TIM4_UP
TIM6/DAC_Channel1		TIM6_UP/DAC_Channel1					
TIM7/DAC_Channel2			TIM7_UP/DAC_Channel2				

Figure 35. DMA2 request mapping

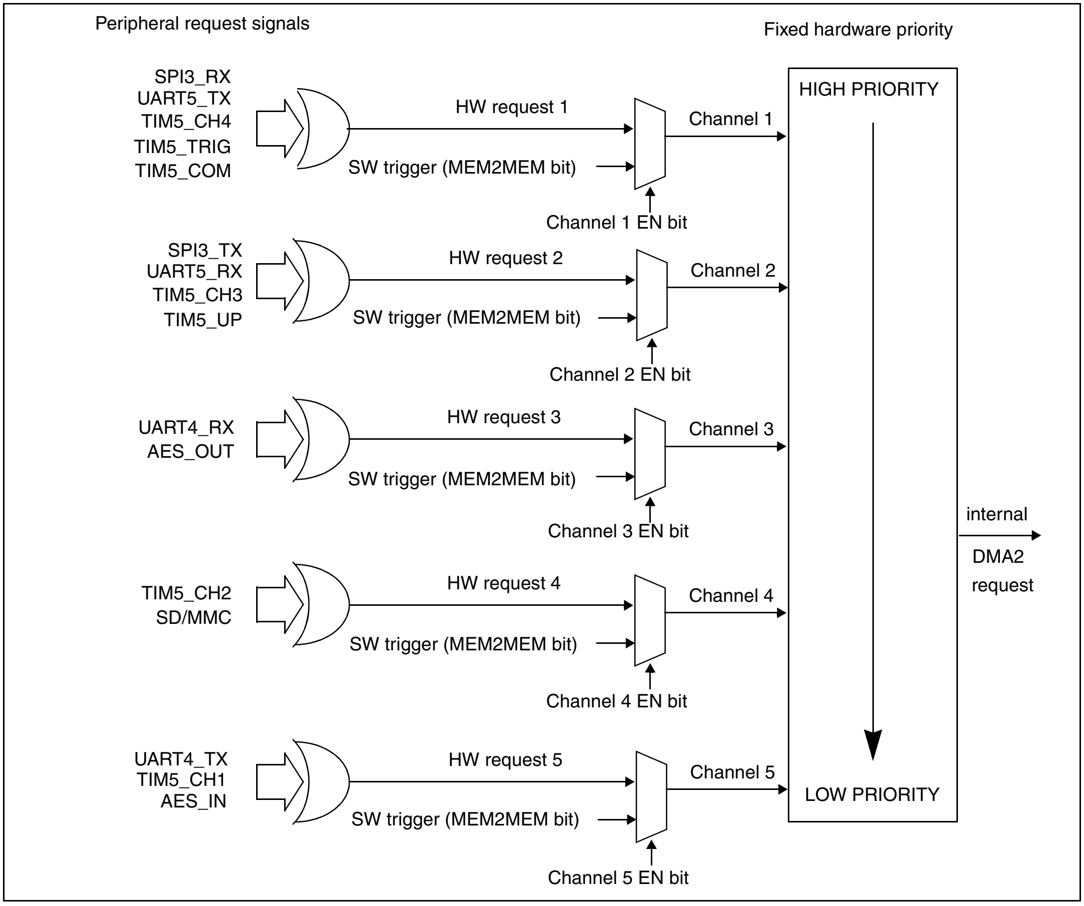


Table 42 lists the DMA2 requests for each channel.

Table 42. Summary of DMA2 requests for each channel

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
SPI3	SPI3_RX	SPI3_TX			
UART4			UART4_RX		UART4_TX
UART5	UART5_TX	UART5_RX			
TIM5	TIM5_CH4 TIM5_TRIG TIM5_COM	TIM5_CH3 TIM5_UP		TIM5_CH2	TIM5_CH1
SDIO				SD/MMC	
AES			AES_OUT		AES_IN

10.4 DMA registers

Refer to *Section 1.1: List of abbreviations for registers on page 37* for a list of abbreviations used in register descriptions.

Note: In the following registers, all bits related to channel6 and channel7 are not relevant for DMA2 since it has only 5 channels.

The peripheral registers can be accessed by bytes (8-bit), half-words (16-bit) or words (32-bit).

10.4.1 DMA interrupt status register (DMA_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 27, 23, 19, 15, 11, 7, 3 **TEIFx:** Channel x transfer error flag (x = 1 ..7)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.

0: No transfer error (TE) on channel x

1: A transfer error (TE) occurred on channel x

Bits 26, 22, 18, 14, 10, 6, 2 **HTIFx:** Channel x half transfer flag (x = 1 ..7)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.

0: No half transfer (HT) event on channel x

1: A half transfer (HT) event occurred on channel x

Bits 25, 21, 17, 13, 9, 5, 1 **TCIFx:** Channel x transfer complete flag (x = 1 ..7)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.

0: No transfer complete (TC) event on channel x

1: A transfer complete (TC) event occurred on channel x

Bits 24, 20, 16, 12, 8, 4, 0 **GIFx:** Channel x global interrupt flag (x = 1 ..7)

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in the DMA_IFCR register.

0: No TE, HT or TC event on channel x

1: A TE, HT or TC event occurred on channel x

10.4.2 DMA interrupt flag clear register (DMA_IFCR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				CTEIF 7	CHTIF 7	CTCIF7	CGIF7	CTEIF6	*CHTIF6*	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5
				w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTEIF 4	CHTIF 4	CTCIF 4	CGIF4	CTEIF 3	CHTIF 3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

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

Bits 27, 23, 19, 15, 11, 7, 3 **CTEIFx:** Channel x transfer error clear (x = 1 ..7)

This bit is set and cleared by software.

0: No effect

1: Clears the corresponding TEIF flag in the DMA_ISR register

Bits 26, 22, 18, 14, 10, 6, 2 **CHTIFx:** Channel x half transfer clear (x = 1 ..7)

This bit is set and cleared by software.

0: No effect

1: Clears the corresponding HTIF flag in the DMA_ISR register

Bits 25, 21, 17, 13, 9, 5, 1 **CTCIFx:** Channel x transfer complete clear (x = 1 ..7)

This bit is set and cleared by software.

0: No effect

1: Clears the corresponding TCIF flag in the DMA_ISR register

Bits 24, 20, 16, 12, 8, 4, 0 **CGIFx:** Channel x global interrupt clear (x = 1 ..7)

This bit is set and cleared by software.

0: No effect

1: Clears the GIF, TEIF, HTIF and TCIF flags in the DMA_ISR register

10.4.3 DMA channel x configuration register (DMA_CCRx) (x = 1..7, where x = channel number)

Address offset: 0x08 + 0d20 × (channel number – 1)

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MEM2 MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 14 **MEM2MEM:** Memory to memory mode

This bit is set and cleared by software.
0: Memory to memory mode disabled
1: Memory to memory mode enabled

Bits 13:12 **PL[1:0]:** Channel priority level

These bits are set and cleared by software.
00: Low
01: Medium
10: High
11: Very high

Bits 11:10 **MSIZE[1:0]:** Memory size

These bits are set and cleared by software.
00: 8-bits
01: 16-bits
10: 32-bits
11: Reserved

Bits 9:8 **PSIZE[1:0]:** Peripheral size

These bits are set and cleared by software.
00: 8-bits
01: 16-bits
10: 32-bits
11: Reserved

Bit 7 **MINC:** Memory increment mode

This bit is set and cleared by software.
0: Memory increment mode disabled
1: Memory increment mode enabled

Bit 6 **PINC:** Peripheral increment mode

This bit is set and cleared by software.
0: Peripheral increment mode disabled
1: Peripheral increment mode enabled

Bit 5 **CIRC:** Circular mode

This bit is set and cleared by software.
0: Circular mode disabled
1: Circular mode enabled

Bit 4 **DIR:** Data transfer direction

This bit is set and cleared by software.
0: Read from peripheral
1: Read from memory

Bit 3 **TEIE:** Transfer error interrupt enable

This bit is set and cleared by software.
0: TE interrupt disabled
1: TE interrupt enabled

Bit 2 **HTIE:** Half transfer interrupt enable

This bit is set and cleared by software.
0: HT interrupt disabled
1: HT interrupt enabled

Bit 1 **TCIE:** Transfer complete interrupt enable

This bit is set and cleared by software.
0: TC interrupt disabled
1: TC interrupt enabled

Bit 0 **EN:** Channel enable

This bit is set and cleared by software.
0: Channel disabled
1: Channel enabled

10.4.4 DMA channel x number of data register (DMA_CNDTRx) (x = 1..7), where x = channel number)

Address offset: 0x0C + 0d20 × (channel number – 1)

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **NDT[15:0]:** Number of data to transfer

Number of data to be transferred (0 up to 65535). This register can only be written when the channel is disabled. Once the channel is enabled, this register is read-only, indicating the remaining bytes to be transmitted. This register decrements after each DMA transfer.
Once the transfer is completed, this register can either stay at zero or be reloaded automatically by the value previously programmed if the channel is configured in auto-reload mode.
If this register is zero, no transaction can be served whether the channel is enabled or not.

10.4.5 DMA channel x peripheral address register (DMA_CPARx) (x = 1..7), where x = channel number)

Address offset: 0x10 + 0d20 × (channel number – 1)

Reset value: 0x0000 0000

This register must *not* be written when the channel is enabled.

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

Bits 31:0 **PA[31:0]:** Peripheral address

Base address of the peripheral data register from/to which the data will be read/written.

When PSIZE is 01 (16-bit), the PA[0] bit is ignored. Access is automatically aligned to a half-word address.

When PSIZE is 10 (32-bit), PA[1:0] are ignored. Access is automatically aligned to a word address.

10.4.6 DMA channel x memory address register (DMA_CMARx) (x = 1..7), where x = channel number)

Address offset: 0x14 + 0d20 × (channel number – 1)

Reset value: 0x0000 0000

This register must *not* be written when the channel is enabled.

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

Bits 31:0 **MA[31:0]:** Memory address

Base address of the memory area from/to which the data will be read/written.

When MSIZE is 01 (16-bit), the MA[0] bit is ignored. Access is automatically aligned to a half-word address.

When MSIZE is 10 (32-bit), MA[1:0] are ignored. Access is automatically aligned to a word address.

10.4.7 DMA register map

The following table gives the DMA register map and the reset values.

Table 43. DMA register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DMA_ISR	Reserved				TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5	TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	DMA_IFCR	Reserved				CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5	CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	DMA_CCR1	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	DMA_CNDTR1	Reserved																NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	DMA_CPAR1	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	DMA_CMAR1	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	Reserved																																
0x01C	DMA_CCR2	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	DMA_CNDTR2	Reserved																NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	DMA_CPAR2	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	DMA_CMAR2	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	Reserved																																
0x030	DMA_CCR3	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43. DMA register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x034	DMA_CNDTR3	Reserved																NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	DMA_CPAR3	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	DMA_CMAR3	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	Reserved																																
0x044	DMA_CCR4	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	DMA_CNDTR4	Reserved																NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	DMA_CPAR4	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	DMA_CMAR4	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	Reserved																																
0x058	DMA_CCR5	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	DMA_CNDTR5	Reserved																NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060	DMA_CPAR5	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	DMA_CMAR5	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	Reserved																																
0x06C	DMA_CCR6	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	DMA_CNDTR6	Reserved																NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 43. DMA register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x074	**DMA_CPAR6**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x078	**DMA_CMAR6**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C	Reserved																																
0x080	**DMA_CCR7**	Reserved																	MEM2MEM	PL [1:0]		M SIZE [1:0]		PSIZE [1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	**DMA_CNDTR7**	Reserved																NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088	**DMA_CPAR7**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	**DMA_CMAR7**	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090	Reserved																																

Refer to *Table 2 on page 42* for the register boundary addresses.

11 Analog-to-digital converter (ADC)

This section applies to the whole STM32L1xxxx family, unless otherwise specified.

11.1 ADC introduction

The 12-bit ADC is a successive approximation analog-to-digital converter. It has up to 42 multiplexed channels allowing it measure signals from up to 40 external and two internal sources. The A/D conversion of the channels can be performed in single, continuous, scan or discontinuous mode. The result of the ADC is stored into a left- or right-aligned 16-bit data register.

The analog watchdog feature allows the application to detect if the input voltage goes beyond the user-defined, higher or lower thresholds.

Conversions are always performed at maximum speed to have the highest possible conversion rate for a given system clock frequency. The automatic power control dramatically reduces the consumption by powering-on the ADC only during conversions.

11.2 ADC main features

- 12-bit, 10-bit, 8-bit or 6-bit configurable resolution
- Interrupt generation at the end of regular conversions, end of injected conversions, and in case of analog watchdog or overrun events (for regular conversions)
- Single and continuous conversion modes
- Scan mode for automatic conversions in a fully programmable order
- Programmable data alignment with in-built data coherency
- Programmable and individual sampling time for each ADC channel
- External trigger option with configurable edge detection for both regular and injected conversions
- Discontinuous mode
- ADC conversion time: 1 µs at full speed (ADC clocked at 16 MHz) down to 4 µs at low speed (ADC clocked at 4 MHz), independent of the APB clock
- Automatic power-up/power-down to reduce the power consumption
- ADC supply requirements:
  - 2.4 V to 3.6 V at full speed or with reference zooming ($V_{REF+} < V_{DDA}$)
  - down to 1.8 V at slower speeds
- ADC input range: $V_{REF-} \leq V_{IN} \leq V_{REF+}$
- Automatic programmable hardware delay insertion between conversions
- DMA request generation during regular channel conversion

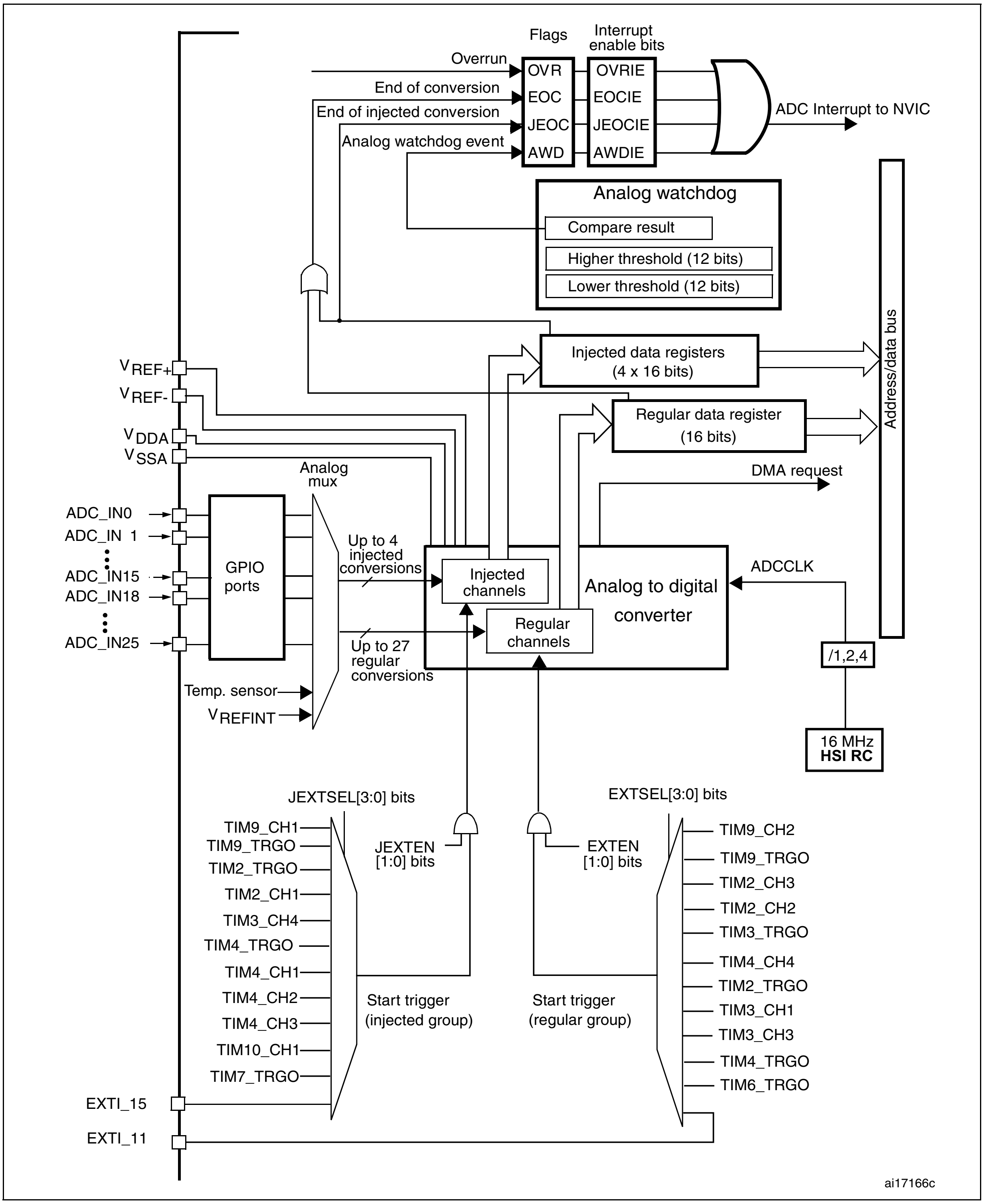
Figure 36 shows the block diagram of the ADC.

Note: V_{REF-}, if available (depending on package), must be tied to V_{SSA}.

11.3 ADC functional description

Figure 36 and *Figure 37* show the ADC block diagram, *Table 44* gives the pin description.

Figure 36. ADC block diagram (low and medium density devices)

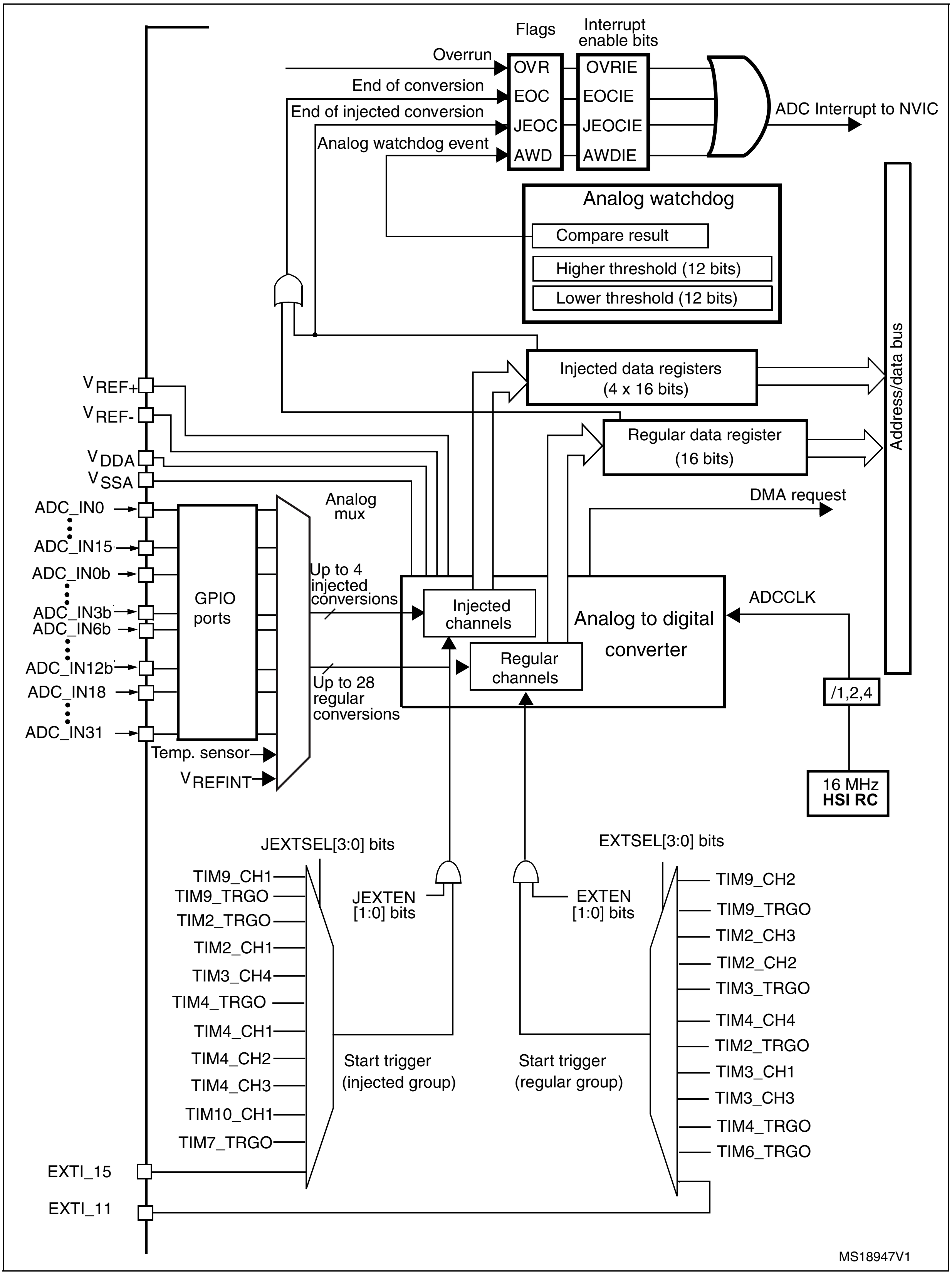


Note: Due to internal connections (ADC multiplexer switches), ADC channels 4, 5, 22, 23, 24 and 25 are direct channels (up to 1 Msample/s) and the other channels are multiplexed (they do not exceed 800 ksamples/s). For more details, refer to Figure 22: Routing interface (RI) block diagram for low and medium density devices on page 151, Figure 23: Routing interface (RI) block diagram for medium+ density devices on page 152 and Figure 24: Routing interface (RI) block diagram for high-density devices on page 153.

Table 44. ADC pins

Name	Signal type	Remarks
V_{REF+}	Input, analog reference positive	The higher/positive reference voltage for the ADC is: 2.4V ≤V_{REF+} = V_{DDA} for full speed (ADCCLK = 16 MHz, 1 Msps) 1.8V ≤V_{REF+} = V_{DDA} for medium speed (ADCCLK = 8 MHz, 500 Ksps) 2.4V ≤V_{REF+} ≠ V_{DDA} for medium speed (ADCCLK = 8 MHz, 500 Ksps) 1.8V ≤V_{REF+} < V_{DDA} for low speed (ADCCLK = 4 MHz, 250 Ksps)
		When product voltage range 3 is selected (V_{CORE} = 1.2 V), the ADC is low speed (ADCCLK = 4 MHz, 250 Ksps)
V_{DDA}	Input, analog supply	Analog power supply equal to V_{DD} and 2.4 V ≤V_{DDA} ≤V_{DD} (3.6 V) for full speed 1.8 V ≤V_{DDA} ≤V_{DD} (3.6 V) for medium and low speed
V_{REF-}	Input, analog reference negative	The lower/negative reference voltage for the ADC, V_{REF-} = V_{SSA}
V_{SSA}	Input, analog supply ground	Ground for analog power supply equal to V_{SS}
ADC_IN[15:0] and ADC_IN[25:18] ADC_IN[31:27], ADC_IN[3:0]b and ADC_IN[12:6]b	Analog input signals	24 analog input channels in low and medium density devices Up to 40 channels in high and medium+ density devices

Figure 37. ADC block diagram (high and medium+ density devices)



11.3.1 ADC power on-off control

The ADC is powered on by setting the ADON bit in the ADC_CR2 register. When the ADON bit is set for the first time, it wakes up the ADC from the Power-down mode.

Conversion starts when either the SWSTART or the JSWSTART bit is set, or in response to an external trigger. These software or hardware triggers must be enabled only when the ADC is ready to convert (ADONS=1).

Resetting the ADON bit stops the conversions and put the ADC in power down mode. In this mode the ADC consumes almost no power. ADONS is cleared after ADON has been synchronized to the ADCCLK clock domain.

Note: Due to the latency introduced by the synchronization between the two clock domains, ADON must be set only when ADONS=0 and it must be cleared only when the ADC is ready to convert (ADONS=1).

Power down configurations (PDI and PDD)

In order to reduce the consumption when the ADC is ready to convert (ADONS=1), the ADC can be automatically powered off when it is not converting, until the next conversion starts depending on the PDI and PDD bits in the ADC_CR1 register. Refer to *Section 11.10: Power saving on page 241* for more details.

Using the PDI bit, the user can determine whether the ADC is powered up or down when it is not converting (waiting for a hardware or software trigger event).

Using the PDD bit, the user can determine whether the ADC is powered up or down between 2 conversions (or sequences of conversions) when a delay is inserted (DELS bits).

When PDI=1, ADONS is the image of ADON (same value) as viewed from the ADCCLK clock.

Conversion starts after the ADC power-up time (t_{STAB}) when either the SWSTART or the JSWSTART bit is set, or in response to an external trigger. These software or hardware triggers must be enabled only when the ADC is ready to convert (ADONS=1).

Resetting the ADON bit stops the conversions and places the ADC in a mode where it is no longer supplied.

Note: Due to the latency introduced by the synchronization between the two clock domains, ADON must be set only when ADONS=0 and it must be cleared only when ADONS=1.

11.3.2 ADC clock

To avoid unnecessary consumption while not converting, the ADC digital interface has been designed to operate in a completely independent manner, at its maximum speed using an internal 16 MHz clock source (HSI), whatever the CPU operating frequency (which can range from a few sub-kHz up to 32 MHz).

Note: When entering Stop mode, the ADC analog and digital interfaces remain inactive as the HSI and PCLK2 are disabled. Since the HSI is still deactived after resuming from Stop mode, the user must enable the HSI as the ADC analog interface clock source and continue using ADC conversions.

The ADCCLK clock is provided by the clock controller. It is generated from the HSI oscillator after a clock divider:

- by 1 for full speed (f_{ADCCLK} = 16 MHz)
- by 2 for medium speed and by 4 for low speed (f_{ADCCLK} = 4 MHz)

Depending on the APB clock (PCLK) frequency, the ADCCLK clock frequency can be higher or lower than PCLK. In particular, when the APB becomes too low, it can become difficult to get the results of conversions at full speed without losing any data (because the data flow is higher than what the CPU or the DMA can handle). This problem can be solved by inserting a delay between 2 conversions or between 2 sequences of conversions in order to give the system enough time to read and save the converted data before the next data arrive. Refer to *Section 11.9: Hardware freeze and delay insertion modes for slow conversions on page 238* for more details.

11.3.3 Channel selection

In low and medium density devices, there are 26 multiplexed channels.

In high and medium+ density devices there are 64 multiplexed channels organized in 2 banks A and B: channel 0a to 31a and 0b to 31b. The bank selection is configured by the ADC_CFG bit in the ADC_CR2 register. Some analog inputs can be converted whatever bank is selected.

It is possible to organize the conversions in two groups: regular and injected. A group consists of a sequence of conversions that can be done on any channel and in any order. For instance, it is possible to implement the conversion sequence in the following order: Ch3, Ch8, Ch2, Ch2, Ch0, Ch2, Ch2, Ch15.

- A **regular group** is composed of up to 28 conversions. The regular channels and their order in the conversion sequence must be selected in the ADC_SQRx registers. The total number of conversions, which can be up to 28 in the regular group must be written in the L[4:0] bits in the ADC_SQR1 register.
- An **injected group** is composed of up to 4 conversions. The injected channels and their order in the conversion sequence must be selected in the ADC_JSQR register. The total number of conversions, which can be up to 4 in the injected group must be written in the L[1:0] bits in the ADC_JSQR register.

Note: If the ADC_SQRx register is modified during a regular conversion or the ADC_JSQR register is modified during an injected conversion, the current conversion is reset and the ADC waits for a new start pulse. If the conversion that is reset is an injected conversion that had interrupted a regular conversion, then the regular conversion is resumed.

Temperature sensor, V_{REFINT} internal channels

The temperature sensor is connected to channel ADCx_IN16 and the internal reference voltage V_{REFINT} is connected to ADCx_IN17. These two internal channels can be selected and converted as injected or regular channels.

11.3.4 Single conversion mode

In Single conversion mode the ADC does one conversion. This mode is started with the CONT bit in the ADC_CR2 at 0 by either:

- setting the SWSTART bit in the ADC_CR2 register (for a regular channel only)
- setting the JSWSTART bit (for an injected channel)
- external trigger (for a regular or injected channel)

Once the conversion of the selected channel is complete:

- If a regular channel was converted (converted channel is selected by the SQ1[4:0] bits in the SQR5 register):
  - The converted data are stored into the 16-bit ADC_DR register
  - The EOC (end of conversion) flag is set
  - An interrupt is generated if the EOCIE bit is set
- If an injected channel was converted (converted channel is selected by the JSQ1[4:0] bits in the JSQR register):
  - The converted data are stored into the 16-bit ADC_JDR1 register
  - The JEOC (end of conversion injected) flag is set
  - An interrupt is generated if the JEOCIE bit is set

Then the ADC stops.

11.3.5 Continuous conversion mode

In continuous conversion mode, the ADC starts a new conversion as soon as it finishes one. This mode is started with the CONT bit at 1 either by external trigger or by setting the SWSTART bit in the ADC_CR2 register (for regular channels only).

After each conversion:

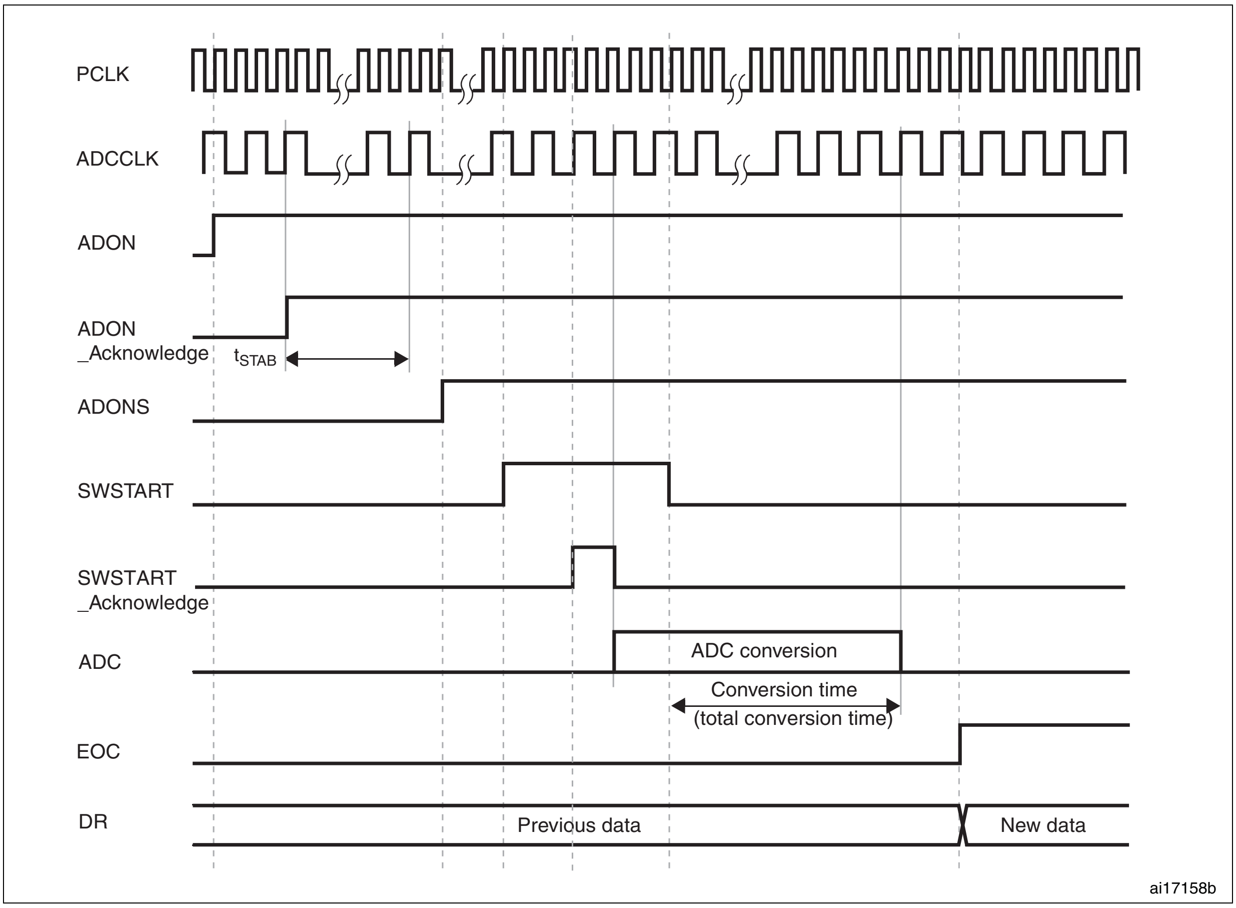
- If a regular channel was converted (converted channel is selected by the SQ1[4:0] bits in the SQR5 register):
  - The last converted data are stored into the 16-bit ADC_DR register
  - The EOC (end of conversion) flag is set
  - An interrupt is generated if the EOCIE bit is set

Note: Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in continuous mode (using JAUTO bit), refer to Auto-injected conversion section).

11.3.6 Timing diagram

As shown in *Figure 38*, the ADC needs a stabilization time (t_{STAB}) before it can actually convert. The ADONS bit is set when a conversion can be triggered. A conversion is launched when the SWSTART bit is set (or when an external trigger is detected). After the conversion time (programmable sampling time + 12 ADCCLK clock cycles for 12-bit data), the EOC flag is set and the ADC data register contains the result of the conversion. Note that some delays are needed to resynchronize the different signals from one clock domain to the other.

Figure 38. Timing diagram (normal mode, PDI=0)



11.3.7 Analog watchdog

The AWD analog watchdog status bit is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold. These thresholds are programmed in the 12 least significant bits of the ADC_HTR and ADC_LTR 16-bit registers. An interrupt can be enabled by using the AWDIE bit in the ADC_CR1 register.

The threshold value is independent of the alignment selected by the ALIGN bit in the ADC_CR2 register. The analog voltage is compared to the lower and higher thresholds before alignment.

Table 45 shows how the ADC_CR1 register should be configured to enable the analog watchdog on one or more channels.

Figure 39. Analog watchdog's guarded area

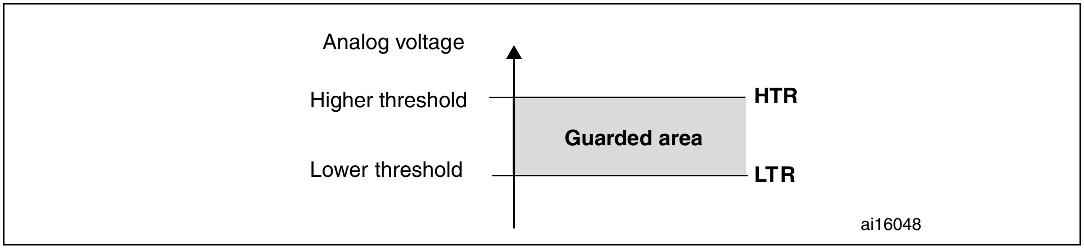


Table 45. Analog watchdog channel selection

Channels guarded by the analog watchdog	ADC_CR1 register control bits (x = don't care)		
	AWDSGL bit	AWDEN bit	JAWDEN bit
None	x	0	0
All injected channels	0	0	1
All regular channels	0	1	0
All regular and injected channels	0	1	1
Single[1] injected channel	1	0	1
Single[1] regular channel	1	1	0
Single [1] regular or injected channel	1	1	1

[1] Selected by the AWDCH[4:0] bits

11.3.8 Scan mode

This mode is used to scan a group of analog channels.

The Scan mode is selected by setting the SCAN bit in the ADC_CR1 register. Once this bit has been set, the ADC scans all the channels selected in the ADC_SQRx registers (for regular channels) or in the ADC_JSQR register (for injected channels). All the channels to be converted must be located in the same bank as the ADC_CFG bit is stable during the scan. A single conversion is performed for each channel of the group. After each end of conversion, the next channel in the group is converted automatically. If the CONT bit in the ADC_CR2 register is set, regular channel conversion does not stop at the last selected channel in the group but continues again from the first selected channel.

If the DMA bit is set, the direct memory access (DMA) controller is used to transfer the data converted from the regular group of channels (stored in the ADC_DR register) to memory after each regular channel conversion.

The EOC bit is set in the ADC_SR register if:

- At the end of each regular group sequence the EOCS bit is cleared to 0
- At the end of each regular channel conversion the EOCS bit is set to 1

The data converted from an injected channel is always stored into the ADC_JDRx registers.

11.3.9 Injected channel management

Triggered injected conversion

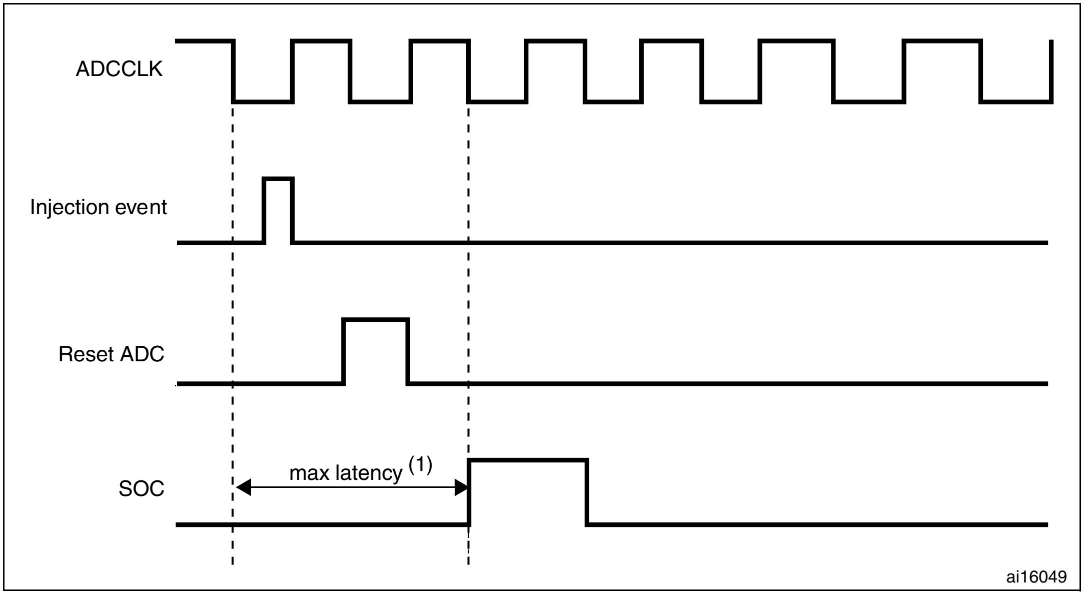
To use triggered injection, the JAUTO bit must be cleared in the ADC_CR1 register.

1. Start the conversion of a group of injected channels either by external trigger or by setting the JSWSTART bit in the ADC_CR2 register.
2. If an external injected trigger occurs or if the JSWSTART bit is set during the conversion of a regular group of channels, the current conversion is reset and the injected channel sequence switches to Scan-once mode.
3. Then, the regular conversion of the regular group of channels is resumed from the last interrupted regular conversion.
   If a regular event occurs during an injected conversion, the injected conversion is not

interrupted but the regular sequence is executed at the end of the injected sequence. *Figure 40* shows the corresponding timing diagram.

Note: When using triggered injection, one must ensure that the interval between trigger events is longer than the injection sequence. For instance, if the sequence length is 30 ADC clock cycles (that is two conversions with a sampling time of 3 clock periods), the minimum interval between triggers must be 31 ADC clock cycles.

Figure 40. Injected conversion latency



1. The maximum latency value can be found in the electrical characteristics of the STM32L1xxxx datasheet.

Auto-injected conversion

If the JAUTO bit is set, then the channels in the injected group are automatically converted after the regular group of channels. This can be used to convert a sequence of up to 31 conversions programmed in the ADC_SQRx and ADC_JSQR registers.

In this mode, external trigger on injected channels must be disabled.

If the CONT bit is also set in addition to the JAUTO bit, regular channels followed by injected channels are continuously converted.

Note: It is not possible to use both the auto-injected and discontinuous modes simultaneously.

11.3.10 Discontinuous mode

Regular group

This mode is enabled by setting the DISCEN bit in the ADC_CR1 register. It can be used to convert a short sequence of n conversions (n ≤8) that is part of the sequence of conversions selected in the ADC_SQRx registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADC_CR1 register.

When an external trigger occurs, it starts the next n conversions selected in the ADC_SQRx registers until all the conversions in the sequence are done. The total sequence length is defined by the L[4:0] bits in the ADC_SQR1 register.

Example:

n = 3, regular channels to be converted = 0, 1, 2, 3, 6, 7, 9, 10
1st trigger: sequence converted 0, 1, 2
2nd trigger: sequence converted 3, 6, 7
3rd trigger: sequence converted 9, 10 and an EOC event generated
4th trigger: sequence converted 0, 1, 2

Note: When a regular group is converted in discontinuous mode, no rollover occurs.

When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the example above, the 4th trigger reconverts the channels 0, 1 and 2 in the 1st subgroup.

Injected group

This mode is enabled by setting the JDISCEN bit in the ADC_CR1 register. It can be used to convert a short sequence of n conversions (n ≤3) part of the sequence of conversions selected in the ADC_JSQR registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADC_CR1 register.

When an external trigger occurs, it starts the next channel conversions selected in the ADC_JSQR registers until all the conversions in the sequence are done. The total sequence length is defined by the JL[1:0] bits in the ADC_JSQR register.

Example:

n = 1, injected channels to be converted = 1, 2, 3
1st trigger: channel 1 converted
2nd trigger: channel 2 converted
3rd trigger: channel 3 converted and EOC and JEOC events generated
4th trigger: channel 1

Note: When all injected channels are converted, the next trigger starts the conversion of the first injected channel. In the example above, the 4th trigger reconverts the 1st injected channel 1.

It is not possible to use both the auto-injected and discontinuous modes simultaneously.

Discontinuous mode must not be set for regular and injected groups at the same time.

11.4 Data alignment

The ALIGN bit in the ADC_CR2 register selects the alignment of the data stored after conversion. Data can be right- or left-aligned as shown in *Figure 41* and *Figure 42*.

The converted data value from the injected group of channels is decreased by the user-defined offset written in the ADC_JOFRx registers so the result can be a negative value. The SEXT bit represents the extended sign value.

For channels in a regular group, no offset is subtracted so only twelve bits are significant.

Figure 41. Right alignment of 12-bit data

Injected group

SEXT	SEXT	SEXT	SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Regular group

0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

ai16050

Figure 42. Left alignment of 12-bit data

Injected group

SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0

Regular group

D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0

ai16051

Special case: when left-aligned, the data are aligned on a half-word basis except when the resolution is set to 6-bit. in that case, the data are aligned on a byte basis as shown in *Figure 43*.

Figure 43. Left alignment of 6-bit data

Injected group

SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	SEXT	D5	D4	D3	D2	D1	D0	0

Regular group

0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0	0	0

ai16052

11.5 Channel-wise programmable sampling time

The ADC samples the input voltage for a number of ADCCLK cycles that can be modified using the SMP[2:0] bits in the ADC_SMPRx registers (x =1 to 3). Each channel of a given bank can be sampled with a different sampling time. Nevertheless, the sampling time selection is shared between the 2 banks.

The total conversion time is calculated as follows:

T_{conv} = Sampling time + channel conversion time

Example:

With ADCCLK = 16 MHz and sampling time = 4 cycles:

$T_{conv} = 4 + 12 = 16$ cycles = 1 µs (for 12-bit conversion)

$T_{conv} = 4 + 7 = 11$ cycles = 685 ns (for 6-bit conversion)

11.6 Conversion on external trigger

Conversion can be triggered by an external event (e.g. timer capture, EXTI line). If the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from “0b00”, then external events are able to trigger a conversion with the selected edge. Table 46 provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger edge.

Table 46. Configuring the trigger edge detection

Source	EXTEN[1:0] / JEXTEN[1:0]
Trigger detection disabled	00
Detection on the rising edge	01
Detection on the falling edge	10
Detection on both the rising and falling edges	11

Note: The edge detection of the external trigger can be changed on the fly.

The EXTSEL[3:0] and JEXTSEL[3:0] control bits are used to select which out of 16 possible events can trigger conversion for the regular and injected groups.

Table 47 gives the possible external trigger for regular conversion.

Table 47. External trigger for regular channels

Source	Type	EXTSEL[3:0]
TIM9_CC2 event	Internal signal from on-chip timers	0000
TIM9_TRGO event		0001
TIM2_CC3 event		0010
TIM2_CC2 event		0011
TIM3_TRGO event		0100
TIM4_CC4 event		0101
TIM2_TRGO event		0110
TIM3_CC1 event		0111
TIM3_CC3 event		1000
TIM4_TRGO event		1001
TIM6_TRGO event		1010
Reserved	NA	1011
Reserved		1100
Reserved		1101
Reserved		1110
EXTI line11	External pin	1111

Table 48 gives the possible external trigger for injected conversion.

Table 48. External trigger for injected channels

Source	Type	EXTSEL[3:0]
TIM9_CC1 event	Internal signal from on-chip timers	0000
TIM9_TRGO event		0001
TIM2_TRGO event		0010
TIM2_CC1 event		0011
TIM3_CC4 event		0100
TIM4_TRGO event		0101
TIM4_CC1 event		0110
TIM4_CC2 event		0111
TIM4_CC3 event		1000
TIM10_CC1 event		1001
TIM7_TRGO event		1010
Reserved	NA	1011
Reserved		1100
Reserved		1101
Reserved		1110
EXTI line15	External pin	1111

A regular group conversion can be interrupted by an injected trigger.

Note: The trigger selection can be changed on the fly. When this is done, however, trigger detection is disabled for a period of 2 PCLK cycles. This is to avoid spurious detections during the transition.

The interval between trigger events must be longer than:

- the sequence for regular conversions
- the sequence + 1 ADCCLK cycle for injected conversions

For instance, if the sequence length is 32 ADC clock cycles (that is two conversions with a 4 clock-period sampling time), the minimum interval between regular triggers must be greater than 32 ADC clock cycles and the interval between injected triggers must be greater than 33 ADC clock cycles.

11.7 Aborting a conversion

11.7.1 Injected channels

An injected conversion or a sequence of conversions can be stopped by writing to the JSQR register (the JL[1:0] bitfield has to be written with its current value). Then any ongoing injected conversion aborts and any pending trigger is reset. A new injected conversion can start when a new hardware or software trigger occurs.

After aborting an injected conversion, the system requires a few clock cycles before a new injected conversion can start (3 to 5 ADC clock cycles + 2 to 5 APB clock cycles). To meet this requirement, JSWSTART should not be set before JCNR=0.

11.7.2 Regular channels

A regular conversion or a sequence of conversions can be stopped by writing to any of the SQR1 to SQR5 registers (if SQR1 is written, the L[4:0] bitfield has to be written with its current value). The ADC then behaves in the same way as in the case of injected conversions (see *Section 11.7.2: Regular channels*).

If several of the SQRi registers have to be written in order to configure a new sequence, no conversion should be launched between the different write accesses. In this case, the following sequence must be applied:

1. Disable the external triggers by writing the EXTEN bits to 00 (when external triggers are used)
2. Change the sequence configuration (by writing to the SQRi registers)
3. Wait for RCNR=0 in the ADC_SR register
4. Enable the external trigger or set the SWSTART bit

11.8 Conversion resolution

It is possible to perform faster conversion by reducing the ADC resolution. The RES[1:0] bits are used to select the number of bits available in the data register. The minimal conversion time for each resolution, when the sampling time is 4 cycles, is then as follows:

- for 12-bit resolution : 12 + 4 = 16 cycles
- for 10-bit resolution : 11 + 4 = 15 cycles
- for 8-bit resolution : 9 + 4 = 13 cycles
- for 6-bit resolution : 7 + 4 = 11 cycles

11.9 Hardware freeze and delay insertion modes for slow conversions

When the APB clock is not fast enough to manage the data rate, a delay can be introduced between conversions to reduce this data rate. The delay is inserted after each regular conversion and after each sequence of injected conversions as, during conversion, a trigger event (for the same group of conversions) occurring during this delay is ignored.

No delay is inserted between conversions of different groups (a regular conversion followed by an injected conversion or conversely):

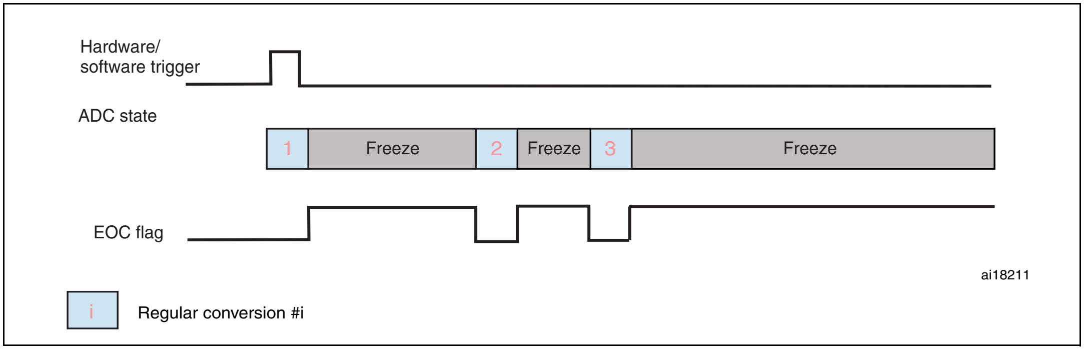
- If an injected trigger occurs during the delay of a regular conversion, the injected conversion starts immediately.
- If a regular conversion is to be resumed after being interrupted by an injected sequence, it starts as soon as the delay of the previous regular conversion is finished.

The behavior is slightly different in auto-injected mode where a new regular conversion can start only when the delay of the previous injected conversion has ended. This is to ensure that the software can read all the data of a given sequence before starting a new sequence. In this mode, a regular trigger is ignored if it occurs during the delay that follows a regular conversion. It is however considered pending if it occurs after this delay, even if it occurs during an injected sequence or the delay that follows it. The conversion then starts at the end of the delay of the injected sequence.

The length of the delay is configured using the DELS[2:0] bits in the ADC_CR2 register. Two cases should be considered:

- ADC freeze mode:
  When DELS[2:0]=001, a new conversion can start only if all the previous data of the same group have been treated:
  - for a regular conversion: once the ADC_DR register has been read or if the EOC bit has been cleared
  - for an injected conversion: when the JEOC bit has been cleared
- ADC delay insertion mode:
  When DELS[2:0]>001, a new conversion can start only after a given number of APB clock cycles after the end of the previous conversion in the same group.

Figure 44. ADC freeze mode

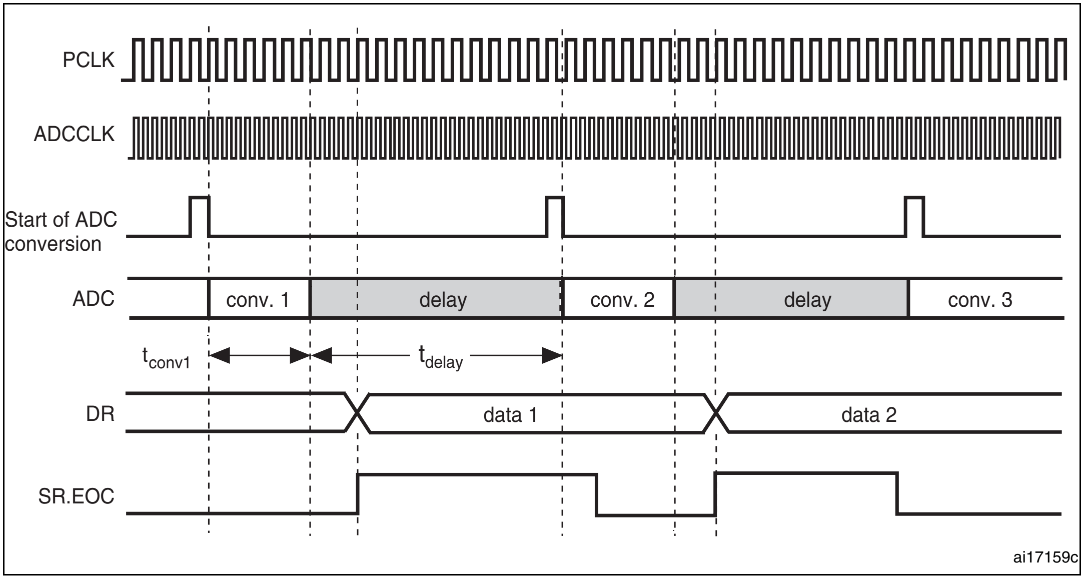


11.9.1 Inserting a delay after each regular conversion

When enabled, a delay is inserted at the end of each regular conversion before a new regular conversion can start. It gives time to read the converted data in the ADC_DR register before a new regular conversion is completed. The length of the delay is configured by the DELS[2:0] bits. *Figure 45* shows an example of continuous regular conversions where a 10 PCLK cycle delay is inserted after each conversion.

Note: When ADC_CR2_EOCS = 1, the delay is inserted after each sequence of regular group conversions.

Figure 45. Continuous regular conversions with a delay

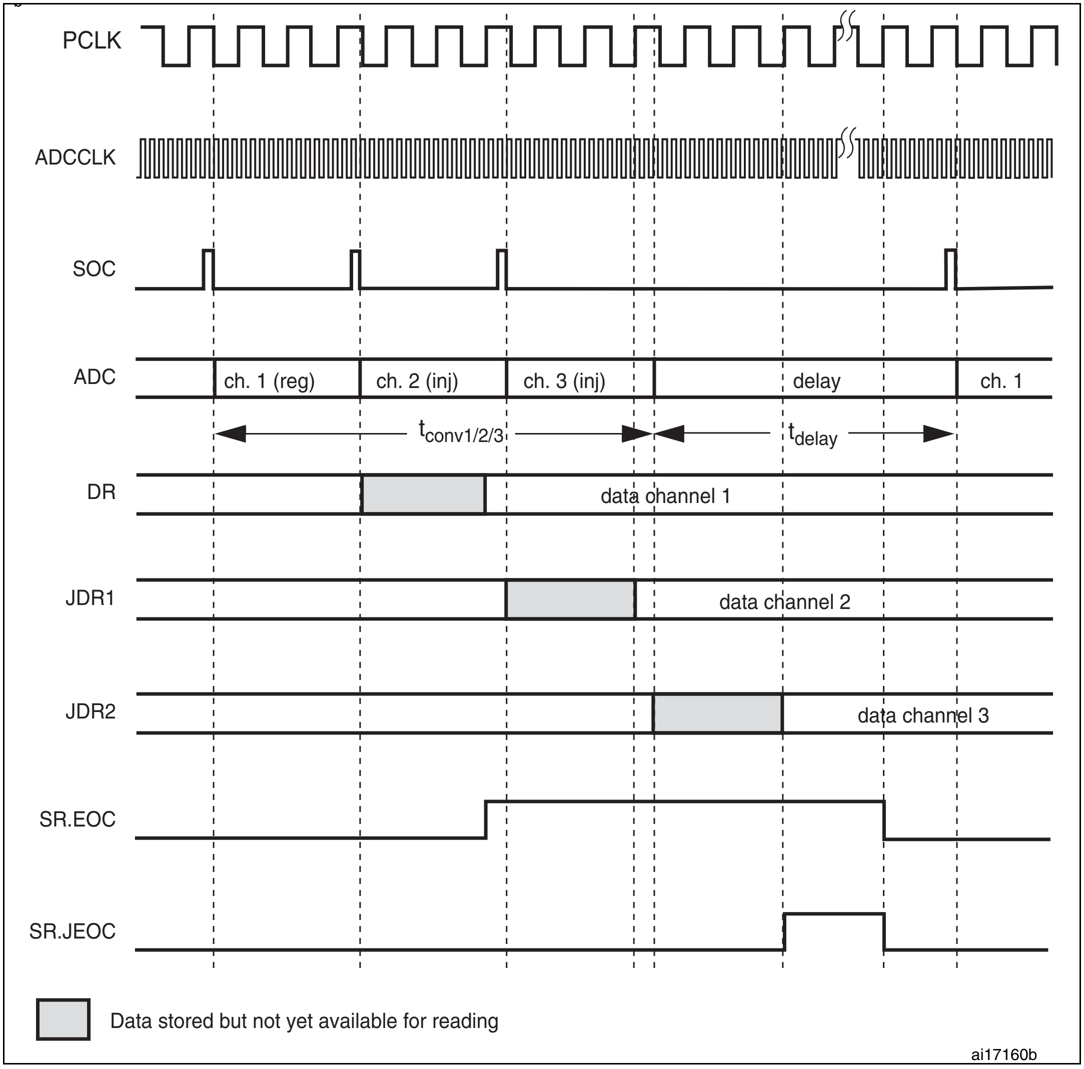


1. t_{conv1}: including sampling and conversion times (for instance 16 ADC clock cycles with the minimum sampling time)
2. t_{delay}: delay from the end of a conversion to the start of the next conversion (number of PCLK periods configured with the DELS[2:0] bits) + delay to synchronize the end of conversion (0 to 1 PCLK clock cycles) + delay to synchronize the end of delay (2 or 3 ADC clock cycles).

11.9.2 Inserting a delay after each sequence of auto-injected conversions

When enabled, a delay is inserted at the end of each sequence of injected conversions. Up to 5 conversion results can be stored into the ADC_DR and the ADC_JDRx registers. The length of the delay is configured by the DELS[2:0] bits. *Figure 46* shows an example of continuous conversions (the CONT bit is set) where a delay is inserted after each sequence of injected conversions. Here the JAUTO bit is set and the sequence ends after the last injected conversion (the sequence is made of 1 regular conversion + 2 injected conversions).

Figure 46. Continuous conversions with a delay between each conversion



1. $t_{conv1/2/3}$: including sampling and conversion times for channels 1, 2 and 3.
2. t_{delay}: delay from the end of the previous sequence to the start of the new sequence (number of PCLK periods configured with the DELS bits) + delay to synchronize the end of conversion (0 to 1 PCLK clock cycles) + delay to synchronize the end of delay (2 or 3 ADC clock cycles).

11.10 Power saving

ADC power-on and power-off can be managed by hardware to cut the consumption when the ADC is not converting. The ADC can be powered down:

- during the delay described above (when the PDD bit is set). Then the ADC is powered up again at the end of the delay
  and/or
- when the ADC is waiting for a trigger event (when the PDI bit is set). In this case the ADC is powered up at the next trigger event.

The ADC needs a certain time to start up before a conversion can actually be launched. This startup time must be taken into account before selecting the automatic power control modes or when configuring the delay. For this reason, it is also more efficient (from the power point of view and when possible) when scanning several channels to launch a sequence of several conversions and stop the consumption after the sequence, than when launching each conversion one by one with a delay after each conversion.

For a given sequence of conversions, the ADCCLK clock must be enabled before launching the first conversion, and be present until the EOC bit (or the JEOC bit in case of injected channels) is set.

Figure 47, *Figure 48* and *Figure 49* show examples of power management in different configurations. ADON=1 in all these examples.

Figure 47. Automatic power-down control: example 1

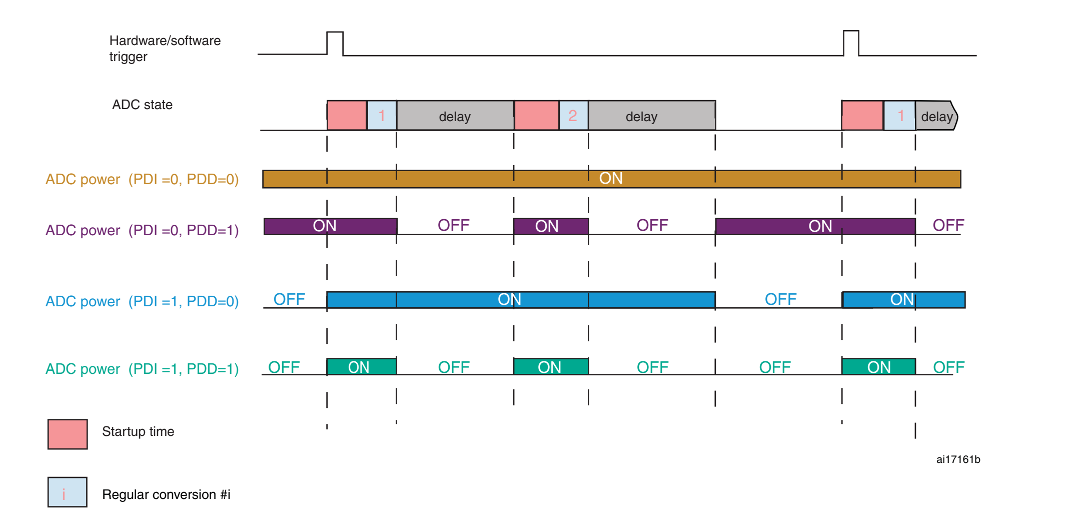

Hardware/software trigger
ADC state
1
delay
2
delay
1
delay
ADC power (PDI =0, PDD=0)
ON
ADC power (PDI =0, PDD=1)
ON
OFF
ON
OFF
ON
OFF
ADC power (PDI =1, PDD=0)
OFF
ON
OFF
ON
ADC power (PDI =1, PDD=1)
OFF
ON
OFF
ON
OFF
OFF
ON
OFF
Startup time
i
Regular conversion #i
ai17161b

Figure 48. Automatic power-down control: example 2

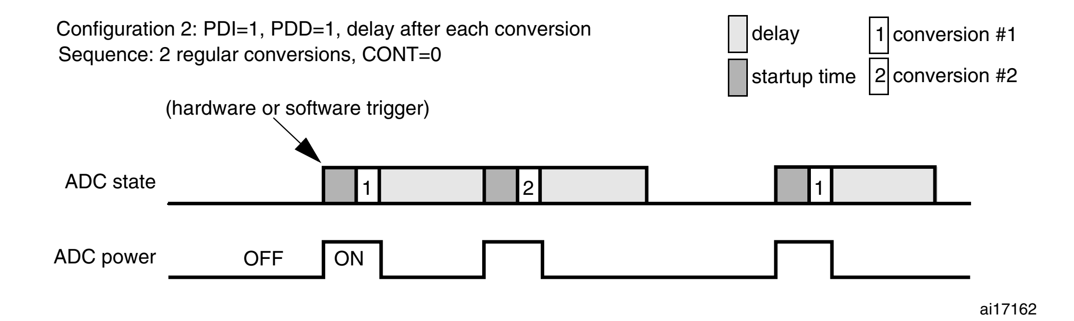
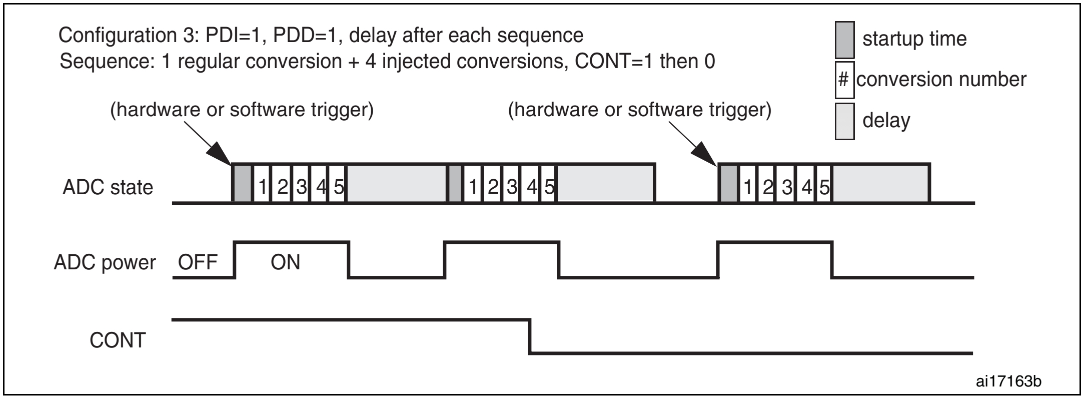

Configuration 2: PDI=1, PDD=1, delay after each conversion
Sequence: 2 regular conversions, CONT=0
delay
startup time
1 conversion #1
2 conversion #2
(hardware or software trigger)
ADC state
1
2
1
ADC power
OFF
ON
ai17162

Figure 49. Automatic power-down control: example 3


Configuration 3: PDI=1, PDD=1, delay after each sequence
Sequence: 1 regular conversion + 4 injected conversions, CONT=1 then 0
startup time
# conversion number
delay
(hardware or software trigger)
(hardware or software trigger)
ADC state
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
ADC power
OFF
ON
CONT
ai17163b

11.11 Data management and overrun detection

11.11.1 Using the DMA

Since converted regular channel values are stored into a unique data register, it is useful to use DMA for conversion of more than one regular channel. This avoids the loss of the data already stored in the ADC_DR register.

When the DMA mode is enabled (DMA bit set to 1 in the ADC_CR2 register), after each conversion of a regular channel, a DMA request is generated. This allows the transfer of the converted data from the ADC_DR register to the destination location selected by the software.

Despite this, if data are lost (overrun), the OVR bit in the ADC_SR register is set and an interrupt is generated (if the OVRIE enable bit is set). DMA transfers are then disabled and DMA requests are no longer accepted. In this case, if a DMA request is made, the regular conversion in progress is aborted and further regular triggers are ignored. It is then necessary to clear the OVR flag and the DMAEN bit in the used DMA stream, and to re-initialize both the DMA and the ADC to have the wanted converted channel data transferred to the right memory location. Only then can the conversion be resumed and the data transfer, enabled again. Injected channel conversions are not impacted by overrun errors.

When OVR = 1 in DMA mode, the DMA requests are blocked after the last valid data have been transferred, which means that all the data transferred to the RAM can be considered as valid.

At the end of the last DMA transfer (number of transfers configured in the DMA controller’s DMA_SxRTR register):

- No new DMA request is issued to the DMA controller if the DDS bit is cleared to 0 in the ADC_CR2 register (this avoids generating an overrun error). However the DMA bit is not cleared by hardware. It must be written to 0, then to 1 to start a new transfer.
- Requests can continue to be generated if the DDS bit is set to 1. This allows configuring the DMA in double-buffer circular mode.

11.11.2 Managing a sequence of conversions without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the EOCS bit must be set in the ADC_CR2 register for the EOC status bit to be set at the end of each conversion, and not only at the end of the sequence. When EOCS = 1, overrun detection is automatically enabled. Thus, each time a conversion is complete, EOC is set and the ADC_DR register can be read. The overrun management is the same as when the DMA is used.

11.11.3 Conversions without reading all the data

It may be useful to let the ADC convert one or more channels without reading the data each time (if there is an analog watchdog for instance). For that, the DMA must be disabled (DMA = 0) and the EOC bit must be set at the end of a sequence only (EOCS = 0). In this configuration no overrun error is reported if a conversion finishes when the result of the previous conversion has not been read.

11.11.4 Overrun detection

Overrun detection is always enabled. It takes place before the data are synchronized to the APB clock.

Note: Only regular channel conversions generate overrun errors.

At the end of a conversion, the result is stored into an intermediate buffer (in the ADC clock domain) until it is transferred to the data register (ADC_DR, in the APB clock domain). If new data arrive before the previous data are transferred, the new data are lost and an overrun error is detected. The OVR bit is set in the ADC_SR register and an interrupt is generated if the OVRIE bit is set.

This may occur in two cases:

- either the delay is not properly set with respect to the APB clock frequency (the delay is too short to synchronize the data), or
- the previous data could not be synchronized to the APB clock because the ADC_DR register is not empty (when DMA=1 or EOCS=1). Indeed, in these modes, the contents of the ADC_DR register cannot be overwritten and so the register always contains the last valid data. ADC_DR is emptied by reading it or by clearing the EOC bit in the ADC_SR register.

Note: An overrun may happen to be detected just after clearing the DMA (or EOCS) when the last data transferred by the DMA are read very late, which causes the next data to be lost.

After clearing the OVR bit, the software should not launch a new regular conversion until RCNR=0 in the ADC_SR register.

11.12 Temperature sensor

The temperature sensor can be used to measure the internal temperature of the device. It is internally connected to the ADC TS (ADC channel 16: temperature sensor) input channel that is used to convert the sensor output voltage into a digital value.

Note: When it is not used, this sensor can be put in power-down mode.

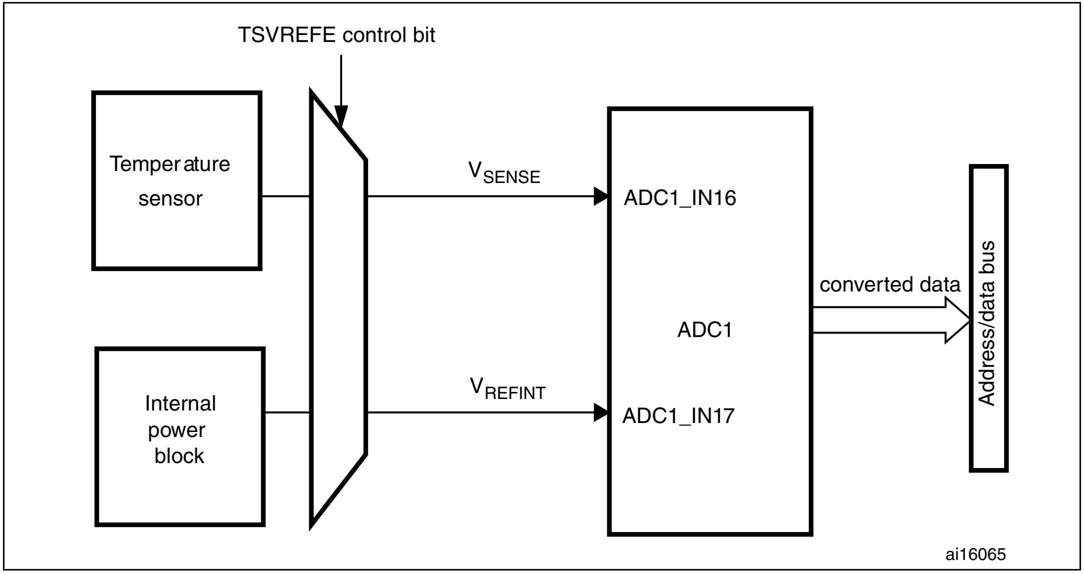
Note: The TSVREFE bit must be set to enable the conversion of both internal channels: ADC channel 16 (temperature sensor) and ADC channel 17 (VREFINT). If the temperature sensor conversion is required, this connection must be enabled.

The internal temperature sensor can also be used to detect temperature variations.

The temperature sensor is factory measured at high temperature and the result of the ADC conversion is stored in a specific data address : the TS_Factory_CONV_V90 data (result of the factory TS voltage conversion at 90°C; refer to the device datasheet for more details).

To reduce the temperature sensor error, the user can measure it at ambient temperature (25°C) to redefine more accurately the average slope (avg_slope) and the offset.

Figure 50. Temperature sensor and V_{REFINT} channel block diagram



11.12.1 How to read the temperature

To read the temperature from the sensor, use the following procedure:

1. Select the ADC TS (temperature sensor) input channel.
2. Select a sample time of 10 µs.
3. Set the TSVREFE bit in the ADC_CCR register to wake up the temperature sensor from power-down mode.
4. Start the ADC conversion.
5. Read the resulting VSENSE data in the ADC data register.
6. Calculate the temperature using the following formulae:

$$T[^\circ K] = \frac{V_{SENSE}}{AvgSlope}$$

$$T[^\circ C] = \frac{V_{SENSE}}{AvgSlope} - 273.15$$

Avg_Slope = average slope of the "Temperature vs. VSENSE" curves (given in mV/°C).

Refer to the Electrical characteristics section for the Avg_Slope value.

Note: When the sensor wakes up from power-down mode, a stabilization time is required before a correct voltage can be output.

After power-on, the ADC also needs a stabilization time. To minimize this delay, the ADON and TSON bits should be set at the same time.

11.13 Internal reference voltage (V_{REFINT}) conversion

The internal reference voltage is internally connected to the VREFINT channel. This analog input channel is used to convert the internal reference voltage into a digital value.

The TSVREFE bit in the ADC_CCR register must be set to enable the internal reference voltage (and also the Temperature sensor). This reference voltage must be enabled only if its conversion is required.

The internal reference voltage is factory measured and the result of the ADC conversion is stored in a specific data address : the VREFINT_Factory_CONV byte.

11.14 ADC interrupts

An interrupt can be produced on the end of conversion for regular and injected groups, when the analog watchdog status bit is set and when the overrun status bit is set. Separate interrupt enable bits are available for flexibility.

Five other flags are present in the ADC_SR register, but there is no interrupt associated with them:

- JCNR (injected channel not ready)
- RCNR (regular channel not ready)
- ADONS (ADON status)
- JSTRT (Start of conversion for channels of an injected group)
- STRT (Start of conversion for channels of a regular group)

Figure 51. ADC flags and interrupts

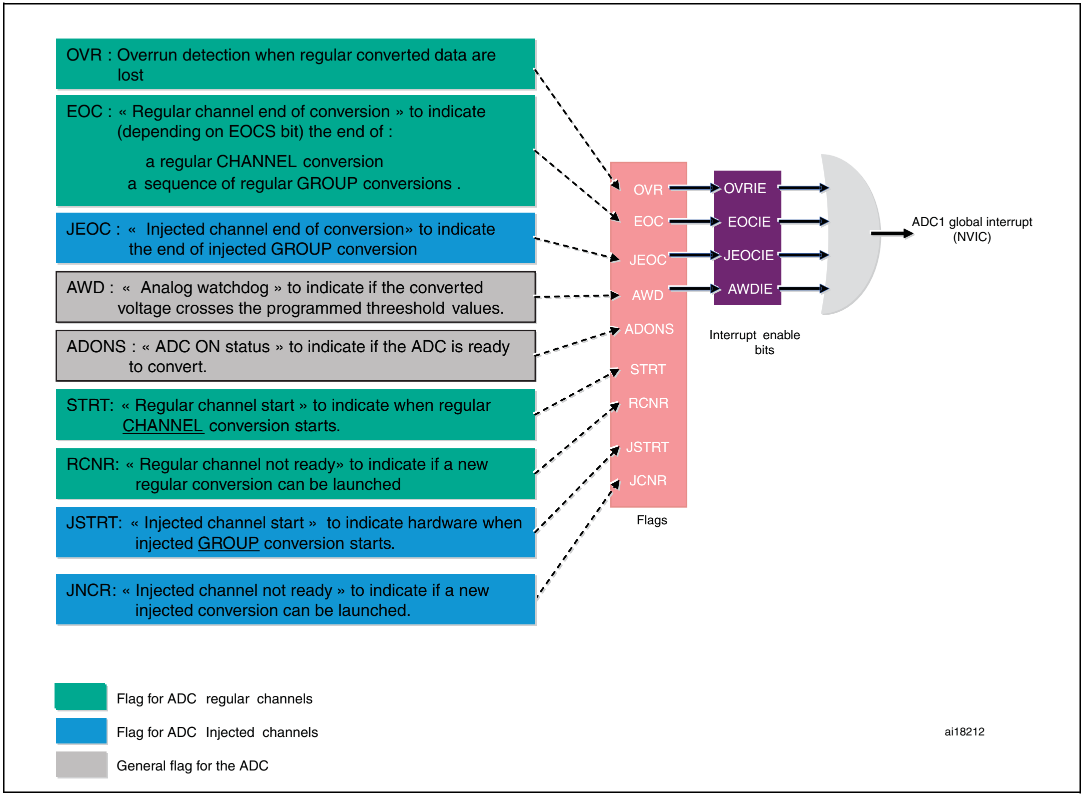

OVR : Overrun detection when regular converted data are lost
EOC : « Regular channel end of conversion » to indicate (depending on EOCS bit) the end of :
a regular CHANNEL conversion
a sequence of regular GROUP conversions .
JEOC : « Injected channel end of conversion» to indicate the end of injected GROUP conversion
AWD : « Analog watchdog » to indicate if the converted voltage crosses the programmed threeshold values.
ADONS : « ADC ON status » to indicate if the ADC is ready to convert.
STRT: « Regular channel start » to indicate when regular CHANNEL conversion starts.
RCNR: « Regular channel not ready» to indicate if a new regular conversion can be launched
JSTRT: « Injected channel start » to indicate hardware when injected GROUP conversion starts.
JNCR: « Injected channel not ready » to indicate if a new injected conversion can be launched.
OVR
EOC
JEOC
AWD
ADONS
STRT
RCNR
JSTRT
JCNR
Flags
OVRIE
EOCIE
JEOCIE
AWDIE
Interrupt enable bits
ADC1 global interrupt (NVIC)
🟩 Flag for ADC regular channels
🟦 Flag for ADC Injected channels
⬜ General flag for the ADC
ai18212

Table 49. ADC interrupts

Interrupt event	Event flag	Enable control bit
End of conversion of a regular group	EOC	EOCIE
End of conversion of an injected group	JEOC	JEOCIE
Analog watchdog status bit is set	AWD	AWDIE
Overrun	OVR	OVRIE

11.15 ADC registers

Refer to *Section 1.1 on page 37* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

11.15.1 ADC status register (ADC_SR)

Address offset: 0x00

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						JCNR	RCNR	Res.	ADONS	OVR	STRT	JSTRT	JEOC	EOC	AWD
						r	r		r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

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

Bit 9 **JCNR:** Injected channel not ready

This bit is set and cleared by hardware after the JSQR register is written. It indicates if a new injected conversion can be launched (by setting the JSWSTART bit).

0: Injected channel ready

1: Injected channel not ready, JSWSTART must not be set

Bit 8 **RCNR:** Regular channel not ready

This bit is set and cleared by hardware after one of the SQRx register is written or after the OVR bit is cleared. It indicates if a new regular conversion can be launched (by setting the SWSTART bit).

0: Regular channel ready

1: Regular channel not ready, SWSTART must not be set

Bit 7 Reserved, must be kept at reset value

Bit 6 **ADONS:** ADC ON status

This bit is set and cleared by hardware to indicate if the ADC is ready to convert.

0: The ADC is not ready

1: The ADC is ready to convert. External triggers can be enabled, the SWSTART and JSWSTART bits can be set.

Bit 5 **OVR:** Overrun

This bit is set by hardware when regular conversion data are lost. It is cleared by software. Overrun detection is enabled only when DMA = 1 or EOCS = 1.

0: No overrun occurredF
1: Overrun has occurred

Bit 4 **STRT:** Regular channel start flag

This bit is set by hardware when regular channel conversion starts. It is cleared by software.

0: No regular channel conversion started
1: Regular channel conversion has started

Bit 3 **JSTRT:** Injected channel start flag

This bit is set by hardware when injected group conversion starts. It is cleared by software.

0: No injected group conversion started
1: Injected group conversion has started

Bit 2 **JEOC:** Injected channel end of conversion

This bit is set by hardware at the end of the conversion of all injected channels in the group. It is cleared by software.

0: Conversion is not complete
1: Conversion complete

Bit 1 **EOC:** Regular channel end of conversion

This bit is set by hardware at the end of the conversion of a regular group of channels. It is cleared by software or by reading the ADC_DR register.

0: Conversion not complete (EOCS=0), or sequence of conversions not complete (EOCS=1)
1: Conversion complete (EOCS=0), or sequence of conversions complete (EOCS=1)

Bit 0 **AWD:** Analog watchdog flag

This bit is set by hardware when the converted voltage crosses the values programmed in the ADC_LTR and ADC_HTR registers. It is cleared by software.

0: No analog watchdog event occurred
1: Analog watchdog event occurred

11.15.2 ADC control register 1 (ADC_CR1)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					OVRIE	RES[1:0]		AWDEN	JAWDEN	Reserved				PDI	PDD
					rw	rw	rw	rw	rw					rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISCNUM[2:0]			JDISCEN	DISCEN	JAUTO	AWDSGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value

Bit 26 **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.

Bits 25:24 **RES[1:0]**: Resolution

These bits are written by software to select the resolution of the conversion.
00: 12-bit (T_{CONV} = 12 ADCCLK cycles)
01: 10-bit (T_{CONV} = 11 ADCCLK cycles)
10: 8-bit (T_{CONV} = 9 ADCCLK cycles)
11: 6-bit (T_{CONV} = 7 ADCCLK cycles)
This bit must be written only when ADON=0.

Bit 23 **AWDEN:** Analog watchdog enable on regular channels

This bit is set and cleared by software.
0: Analog watchdog disabled on regular channels
1: Analog watchdog enabled on regular channels

Bit 22 **JAWDEN:** Analog watchdog enable on injected channels

This bit is set and cleared by software.
0: Analog watchdog disabled on injected channels
1: Analog watchdog enabled on injected channels

Bits 21:18 Reserved, must be kept at reset value

Bit 17 **PDI:** Power down during the idle phase

This bit is written and cleared by software. When ADON=1, it determines whether the ADC is powered up or down when not converting (waiting for a hardware or software trigger event).
0: The ADC is powered up when waiting for a start event
1: The ADC is powered down when waiting for a start event

Note: This bit must be written only when ADON=0.

Bit 16 **PDD:** Power down during the delay phase

This bit is written and cleared by software. When ADON=1, it determines whether the ADC is powered up or down between 2 conversions (or sequences of conversions) when a delay is inserted (DELS bits).
0: The ADC is powered up during the delay
1: The ADC is powered down during the delay

Note: This bit must be written only when ADON=0.

Bits 15:13 **DISCNUM[2:0]:** Discontinuous mode channel count

These bits are written by software to define the number of channels to be converted in discontinuous mode, after receiving an external trigger.

000: 1 channel
001: 2 channels
...
111: 8 channels

Note: This bit must be written only when ADON=0.

Bit 12 **JDISCEN:** Discontinuous mode on injected channels

This bit is set and cleared by software to enable/disable discontinuous mode on the injected channels of a group.

0: Discontinuous mode on injected channels disabled
1: Discontinuous mode on injected channels enabled

Note: This bit must be written only when ADON=0.

Bit 11 **DISCEN:** Discontinuous mode on regular channels

This bit is set and cleared by software to enable/disable Discontinuous mode on regular channels.

0: Discontinuous mode on regular channels disabled
1: Discontinuous mode on regular channels enabled

Note: This bit must be written only when ADON=0.

Bit 10 **JAUTO:** Automatic injected group conversion

This bit is set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.

0: Automatic injected group conversion disabled
1: Automatic injected group conversion enabled

Note: This bit must be written only when ADON=0.

Bit 9 **AWDSGL:** Enable the watchdog on a single channel in scan mode

This bit is set and cleared by software to enable/disable the analog watchdog on the channel identified by the AWDCH[4:0] bits.

0: Analog watchdog enabled on all channels
1: Analog watchdog enabled on a single channel

Bit 8 **SCAN**: Scan mode

This bit is set and cleared by software to enable/disable the Scan mode. In the Scan mode, the inputs selected through the ADC_SQRx or ADC_JSQRx registers are converted.

0: Scan mode disabled
1: Scan mode enabled

Note: This bit must be written only when ADON=0.

Bit 7 **JEOCIE:** Interrupt enable for injected channels

This bit is set and cleared by software to enable/disable the end of conversion interrupt for injected channels.

0: JEOC interrupt disabled
1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

Bit 6 **AWDIE:** Analog watchdog interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog interrupt. In Scan mode if the watchdog thresholds are crossed, scan is aborted only if this bit is enabled.

0: Analog watchdog interrupt disabled
1: Analog watchdog interrupt enabled

Bit 5 **EOCIE:** Interrupt enable for EOC

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.

Bits 4:0 **AWDCH[4:0]:** Analog watchdog channel select bits

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

00000: ADC analog input Channel0
00001: ADC analog input Channel1
...

11000: ADC analog input Channel24
11001: ADC analog input Channel25
11010: ADC analog input Channel26
Other values reserved.

Note: ADC1 analog inputs Channel16, Channel 17 and Channel26 are internally connected to the temperature sensor, to $V_{REFINT\ and\ to\ VCOMP}$ respectively.

11.15.3 ADC control register 2 (ADC_CR2)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SWSTART	EXTEN		EXTSEL[3:0]				Res.	JSWSTART	JEXTEN		JEXTSEL[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
Reserved				ALIGN	EOCS	DDS	DMA	Res.	DELS			Res.	ADC_CFG	CONT	ADON
				rw	rw	rw	rw		rw	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value

Bit 30 **SWSTART:** Start conversion of regular channels

This bit is set by software to start conversion and cleared by hardware as soon as the conversion starts.

0: Reset state
1: Starts conversion of regular channels

Note: This bit must be set only when ADONS=1 and RCNR=0.

Bits 29:28 **EXTEN:** External trigger enable for regular channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.

00: Trigger detection disabled
01: Trigger detection on the rising edge
10: Trigger detection on the falling edge
11: Trigger detection on both the rising and falling edges

Note: The external trigger must be enabled only when ADONS=1.

Bits 27:24 **EXTSEL[3:0]:** External event select for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

0000: TIM9_CC2 event
0001: TIM9_TRGO event
0010: TIM2_CC3 event
0011: TIM2_CC2 event
0100: TIM3_TRGO event
0101: TIM4_CC4 event
0110: TIM2_TRGO event
0111: TIM3_CC1 event
1000: TIM3_CC3 event
1001: TIM4_TRGO event
1010: TIM6_TRGO event
1011: Reserved
1100: Reserved
1101: Reserved
1110: Reserved
1111: EXTI line11

Bit 23 Reserved, must be kept at reset value

Bit 22 **JSWSTART:** Start conversion of injected channels

This bit is set by software and cleared by hardware as soon as the conversion starts.

0: Reset state
1: Starts conversion of injected channels

Note: This bit must be set only when ADONS=1 and JCNR=0.

Bits 21:20 **JEXTEN:** External trigger enable for injected channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.

00: Trigger detection disabled
01: Trigger detection on the rising edge
10: Trigger detection on the falling edge
11: Trigger detection on both the rising and falling edges

Note: The external trigger must be enabled only when ADONS=1.

Bits 19:16 **JEXTSEL[3:0]:** External event select for injected group

These bits select the external event used to trigger the start of conversion of an injected group.

0000: TIM9_CC1 event
0001: TIM9_TRGO event
0010: TIM2_TRGO event
0011: TIM2_CC1 event
0100: TIM3_CC4 event
0101: TIM4_TRGO event
0110: TIM4_CC1 event
0111: TIM4_CC2 event
1000: TIM4_CC3 event
1001: TIM10_CC1 event
1010: TIM7_TRGO event
1011: Reserved
1100: Reserved
1101: Reserved
1110: Reserved
1111: EXTI line15

Bits 15:12 Reserved, must be kept at reset value

Bit 11 **ALIGN:** Data alignment

This bit is set and cleared by software. Refer to *Figure 41* and *Figure 42*.

0: Right alignment
1: Left alignment

Bit 10 **EOCS:** End of conversion selection

This bit is set and cleared by software.

0: The EOC bit is set at the end of each sequence of regular conversions
1: The EOC bit is set at the end of each regular conversion

Bit 9 **DDS:** DMA disable selection

This bit is set and cleared by software.

0: No new DMA request is issued after the last transfer (as configured in the DMA controller)
1: DMA requests are issued as long as data are converted and DMA=1

Bit 8 **DMA:** Direct memory access mode

This bit is set and cleared by software. Refer to the DMA controller chapter for more details.

0: DMA mode disabled
1: DMA mode enabled

Bit 7 Reserved, must be kept at reset value

Bit 6:4 **DELS:** Delay selection

These bits are set and cleared by software. They define the length of the delay which is applied after a conversion or a sequence of conversions.

000: No delay
001: Until the converted data have been read (DR read or EOC=0 for regular conversions, JEOC=0 for injected conversions)
010: 7 APB clock cycles after the end of conversion
011: 15 APB clock cycles after the end of conversion
100: 31 APB clock cycles after the end of conversion
101: 63 APB clock cycles after the end of conversion
110: 127 APB clock cycles after the end of conversion
111: 255 APB clock cycles after the end of conversion

Note: 1- This bit must be written only when ADON=0.

2- Due to clock domain crossing, a latency of 2 or 3 ADC clock cycles is added to the delay before a new conversion can start.

3- The delay required for a given frequency ratio between the APB clock and the ADC clock depends on the activity on the AHB and APB busses. If the ADC is the only peripheral that needs to transfer data, then a minimum delay should be configured: 15 APB clock cycles if $f_{APB} < f_{ADCCLK}/2$ or else 7 APB clock cycles if $f_{APB} < f_{ADCCLK}$, otherwise no delay is needed.

Bit 3 Reserved, must be kept at reset value

Bit 2 **ADC_CFG:** ADC configuration

This bit is set and cleared by software. It selects the bank of channels to be converted.

0: Bank A selected
1: Bank B selected

Note: This bit must be modified only when no conversion is on going.

This bit is available in high and medium+ density devices only

Bit 1 **CONT:** Continuous conversion

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

Bit 0 **ADON:** A/D Converter ON / OFF

This bit is set and cleared by software.

0: Disable ADC conversion and go to power down mode
1: Enable ADC: conversions can start as soon as a start event (hardware or software) is received. When not converting, the ADC goes to the power up or power down mode depending on the PDI and PDD bits.

Note: This bit must be set only when ADONS=0 and cleared only when ADONS=1.

11.15.4 ADC sample time register 1 (ADC_SMPR1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SMP29[2:0]			SMP28[2:0]			SMP27[2:0]			SMP26[2:0]			SMP25[2:1]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP25[0]	SMP24[2:0]			SMP23[2:0]			SMP22[2:0]			SMP21[2:0]			SMP20[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 29:0 **SMPx[2:0]:** Channel x sampling time selection

These bits are written by software to select the sampling time individually for each channel. During sampling cycles, the channel selection bits must remain unchanged.

000: 4 cycles
001: 9 cycles
010: 16 cycles
011: 24 cycles
100: 48 cycles
101: 96 cycles
110: 192 cycles
111: 384 cycles

Note: These bits must be written only when ADON=0.

11.15.5 ADC sample time register 2 (ADC_SMPR2)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SMP19[2:0]			SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP15[0]	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 29:0 **SMPx[2:0]:** Channel x sampling time selection

These bits are written by software to select the sampling time individually for each channel. During sample cycles, the channel selection bits must remain unchanged.

000: 4 cycles
001: 9 cycles
010: 16 cycles
011: 24 cycles
100: 48 cycles
101: 96 cycles
110: 192 cycles
111: 384 cycles

Note: These bits must be written only when ADON=0.

11.15.6 ADC sample time register 3 (ADC_SMPR3)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP5[0]	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 29:0 **SMPx[2:0]:** *Channel x Sample time selection*

These bits are written by software to select the sampling time individually for each channel. During the sampling cycles, the channel selection bits must remain unchanged.

000: 4 cycles
001: 9 cycles
010: 16 cycles
011: 24 cycles
100: 48 cycles
101: 96 cycles
110: 192 cycles
111: 384 cycles

Note: These bits must be written only when ADON=0.

11.15.7 ADC injected channel data offset register x (ADC_JOFRx)(x=1..4)

Address offset: 0x18-0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				JOFFSETx[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 **JOFFSETx[11:0]:** Data offset for injected channel x

These bits are written by software to define the offset to be subtracted from the raw converted data when converting injected channels. The conversion result can be read from in the ADC_JDRx registers.

11.15.8 ADC watchdog higher threshold register (ADC_HTR)

Address offset: 0x28

Reset value: 0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				HT[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 **HT[11:0]:** Analog watchdog higher threshold

These bits are written by software to define the higher threshold for the analog watchdog.

11.15.9 ADC watchdog lower threshold register (ADC_LTR)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				LT[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 **LT[11:0]:** Analog watchdog lower threshold

These bits are written by software to define the lower threshold for the analog watchdog.

11.15.10 ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							L[4:0]					SQ28[4:1]			
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SQ28[0]	SQ27[4:0]					SQ26[4:0]					SQ25[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

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

Bits 24:20 **L[4:0]:** Regular channel sequence length

These bits are written by software to define the total number of conversions in the regular channel conversion sequence.

00000: 1 conversion
00001: 2 conversions
...
11010: 27 conversions
11011: 28 conversions (applicable in high and medium+ density devices only)

Bits 19:15 **SQ28[4:0]:** 28th conversion in regular sequence

These bits are written by software with the channel number (0..31) assigned as the 28th in the conversion sequence. The channel is selected in bank A or bank B depending on the ADC_CFG bit in the ADC_CR2 register.

Note: These bits are available in high and medium+ density devices only

Bits 14:10 **SQ27[4:0]:** 27th conversion in regular sequence

Low and medium density devices: These bits are written by software with the channel number (0..26) assigned as the 27th in the conversion sequence.

High and medium+ density devices: 27th conversion in regular sequence

Bits 9:5 **SQ26[4:0]:** 26th conversion in regular sequence

Bits 4:0 **SQ25[4:0]:** 25th conversion in regular sequence

11.15.11 ADC regular sequence register 2 (ADC_SQR2)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ24[4:0]					SQ23[4:0]					SQ22[4:1]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SQ22[0]	SQ21[4:0]					SQ20[4:0]					SQ19[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 29:26 **SQ24[4:0]:** 24th conversion in regular sequence

These bits are written by software with the channel number (0.31) assigned as the 24th in the sequence to be converted.

Bits 24:20 **SQ23[4:0]:** 23rd conversion in regular sequence

Bits 19:15 **SQ22[4:0]:** 22nd conversion in regular sequence

Bits 14:10 **SQ21[4:0]:** 21st conversion in regular sequence

Bits 9:5 **SQ20[4:0]:** 20th conversion in regular sequence

Bits 4:0 **SQ19[4:0]:** 19th conversion in regular sequence

11.15.12 ADC regular sequence register 3 (ADC_SQR3)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ18[4:0]					SQ17[4:0]					SQ16[4:1]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ16[0]	SQ15[4:0]					SQ14[4:0]					SQ13[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 29:25 **SQ18[4:0]:** 18th conversion in regular sequence

These bits are written by software with the channel number (0..31) assigned as the 18th in the sequence to be converted.

Bits 24:20 **SQ17[4:0]:** 17th conversion in regular sequence

Bits 19:15 **SQ16[4:0]:** 16th conversion in regular sequence

Bits 14:10 **SQ15[4:0]:** 15th conversion in regular sequence

Bits 9:5 **SQ14[4:0]:** 14th conversion in regular sequence

Bits 4:0 **SQ13[4:0]:** 13th conversion in regular sequence

11.15.13 ADC regular sequence register 4 (ADC_SQR4)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ12[4:0]					SQ11[4:0]					SQ10[4:1]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ10[0]	SQ9[4:0]					SQ8[4:0]					SQ7[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 29:26 **SQ12[4:0]**: 12th conversion in regular sequence

These bits are written by software with the channel number (0..31) assigned as the 12th in the sequence to be converted.

Bits 24:20 **SQ11[4:0]**: 11th conversion in regular sequence

Bits 19:15 **SQ10[4:0]**: 10th conversion in regular sequence

Bits 14:10 **SQ9[4:0]:** 9th conversion in regular sequence

Bits 9:5 **SQ8[4:0]**: 8th conversion in regular sequence

Bits 4:0 **SQ7[4:0]**: 7th conversion in regular sequence

11.15.14 ADC regular sequence register 5 (ADC_SQR5)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ6[4:0]					SQ5[4:0]					SQ4[4:1]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ4_0	SQ3[4:0]					SQ2[4:0]					SQ1[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 29:25 **SQ6[4:0]**: 6th conversion in regular sequence

These bits are written by software with the channel number (0..31) assigned as the 6th in the sequence to be converted.

Bits 24:20 **SQ5[4:0]:** 5th conversion in regular sequence

Bits 19:15 **SQ4[4:0]**: 4th conversion in regular sequence

Bits 14:10 **SQ3[4:0]**: 3rd conversion in regular sequence

Bits 9:5 **SQ2[4:0]:** 2nd conversion in regular sequence

Bits 4:0 **SQ1[4:0]**: 1st conversion in regular sequence

11.15.15 ADC injected sequence register (ADC_JSQR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										JL[1:0]		JSQ4[4:1]			
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ4[0]	JSQ3[4:0]					JSQ2[4:0]					JSQ1[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value

Bits 21:20 **JL[1:0]:** Injected sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

00: 1 conversion
01: 2 conversions
10: 3 conversions
11: 4 conversions

Bits 19:15 **JSQ4[4:0]:** 4th conversion in injected sequence (when JL[1:0]=3, see note below)

These bits are written by software with the channel number (0..31) assigned as the 4th in the sequence to be converted. The channel is selected in bank A or bank B depending on the ADC_CFG bit in the ADC_CR2 register.

Bits 14:10 **JSQ3[4:0]:** 3rd conversion in injected sequence (when JL[1:0]=3, see note below)

Bits 9:5 **JSQ2[4:0]:** 2nd conversion in injected sequence (when JL[1:0]=3, see note below)

Bits 4:0 **JSQ1[4:0]:** 1st conversion in injected sequence (when JL[1:0]=3, see note below)

Note: When JL[1:0]=3 (4 injected conversions in the sequencer), the ADC converts the channels in the following order: JSQ1[4:0], JSQ2[4:0], JSQ3[4:0], and JSQ4[4:0].

When JL=2 (3 injected conversions in the sequencer), the ADC converts the channels in the following order: JSQ2[4:0], JSQ3[4:0], and JSQ4[4:0].

When JL=1 (2 injected conversions in the sequencer), the ADC converts the channels in starting from JSQ3[4:0], and then JSQ4[4:0].

When JL=0 (1 injected conversion in the sequencer), the ADC converts only JSQ4[4:0] channel.

11.15.16 ADC injected data register x (ADC_JDRx) (x= 1..4)

Address offset: 0x48 - 0x54

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value

Bits 15:0 **JDATA[15:0]:** Injected data

These bits are read-only. They contain the conversion result from injected channel x. The data are left -or right-aligned as shown in *Figure 41* and *Figure 42*.

11.15.17 ADC regular data register (ADC_DR)

Address offset: 0x58

Reset value: 0x0000 0000

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

Bits 15:0 **DATA[15:0]:** Regular data

These bits are read-only. They contain the conversion result from the regular channels. The data are left- or right-aligned as shown in *Figure 41* and *Figure 42*.

11.15.18 ADC sample time register 0 (ADC_SMPR0)

Address offset: 0x5C

Reset value: 0x0000 0000

Note: This register is available in high and medium+ density devices only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										SMP31[2:0]			SMP30[2:0]		
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value

Bits 5:0 **SMPx[2:0]:** *Channel x Sample time selection*

These bits are written by software to select the sampling time individually for each channel. During the sampling cycles, the channel selection bits must remain unchanged.

000: 4 cycles
001: 9 cycles
010: 16 cycles
011: 24 cycles
100: 48 cycles
101: 96 cycles
110: 192 cycles
111: 384 cycles

Note: These bits must be written only when ADON=0.

11.15.19 ADC common status register (ADC_CSR)

Address offset: 0x00 (this offset address is relative to the base address of ADC common registers, i.e. 0x300)

Reset value: 0x0000 0000

This register provides an image of the status bits of the different ADCs. Nevertheless it is read-only and does not allow to clear the different status bits. Instead each status bit must be cleared by writing it to 0 in the corresponding ADC_SR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									ADONS1	OVR1	STRT1	JSTRT1	JEOC 1	EOC1	AWD1
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value

Bit 6 **ADONS1:** *ADON Status of ADC1*

This bit is a copy of the ADONS bit in the ADC_SR register.

Bit 5 **OVR1:** Overrun flag of the ADC

This bit is a copy of the OVR bit in the ADC_SR register.

Bit 4 **STRT1:** Regular channel Start flag of the ADC

This bit is a copy of the STRT bit in the ADC_SR register.

Bit 3 **JSTRT1:** Injected channel Start flag of the ADC

This bit is a copy of the JSTRT bit in the ADC_SR register.

Bit 2 **JEOC1:** Injected channel end of conversion of the ADC

This bit is a copy of the JEOC bit in the ADC_SR register.

Bit 1 **EOC1:** End of conversion of the ADC

This bit is a copy of the EOC bit in the ADC_SR register.

Bit 0 **AWD1:** Analog watchdog flag of the ADC

This bit is a copy of the AWD bit in the ADC_SR register.

11.15.20 ADC common control register (ADC_CCR)

Address offset: 0x04 (this offset address is relative to the base address of ADC common registers, i.e. 0x300)

Reset value: 0x0000 0000

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

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

Bit 23 **TSVREFE:** Temperature sensor and V_{REFINT} enable

This bit is set and cleared by software to enable/disable the temperature sensor and the V_{REFINT} channel.
0: Temperature sensor and V_{REFINT} channel disabled
1: Temperature sensor and V_{REFINT} channel enabled

Bits 22:18 Reserved, must be kept at reset value

Bits 17:16 **ADCPRE:** ADC prescaler

Set and cleared by software to select the frequency of the clock to the ADC.
00: HSI divided by 1
01: HSI divided by 2
10: HSI divided by 4
11: Reserved

Bits 15:0 Reserved, must be kept at reset value

11.15.21 ADC register map

The following table summarizes the ADC registers.

Table 50. ADC global register map

Offset	Register
0x000 - 0x058	ADC
0x05C - 0x2FC	Reserved
0x300 - 0x304	Common registers

Table 51. ADC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	ADC_SR	Reserved																						JCNR	RCNR	Reserved	ADONS	OVR	STRT	JSTRT	JEOC	EOC	AWD
	Reset value																							0	0		0	0	0	0	0	0	0
0x04	ADC_CR1	Reserved					OVRIE	RES[1:0]		AWDEN	JAWDEN	Reserved				PDI	PDD	DISC NUM [2:0]			JDISCEN	DISCEN	JAUTO	AWD SGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[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
0x08	ADC_CR2	Reserved	SWSTART	EXTEN[1:0]		EXTSEL [3:0]				Reserved	JSWSTART	JEXTEN[1:0]		JEXTSEL [3:0]				Reserved				ALIGN	EOCS	DDS	DMA	Reserved	DELS[2:0]			Reserved	ADC_CFG	CONT	ADON
	Reset value		0	0	0	0	0	0	0		0	0	0	0	0	0	0					0	0		0		0	0	0		0	0	0
0x0C	ADC_SMPR1	Sample time bits SMPx_x																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	ADC_SMPR2	Sample time bits SMPx_x																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	ADC_SMPR3	Sample time bits SMPx_x																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	ADC_JOFR1	Reserved																				JOFFSET1[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x1C	ADC_JOFR2	Reserved																				JOFFSET2[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0

Table 51. ADC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20	ADC_JOFR3	Reserved																				JOFFSET3[11:0]										
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x24	ADC_JOFR4	Reserved																				JOFFSET4[11:0]										
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x28	ADC_HTR	Reserved																				HT[11:0]										
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x2C	ADC_LTR	Reserved																				LT[11:0]										
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x30	ADC_SQR1	Reserved						L[4:0]				Regular channel sequence SQx_x bits																		
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	ADC_SQR2	Reserved	Regular channel sequence SQx_x bits																													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	ADC_SQR3	Reserved	Regular channel sequence SQx_x bits																													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	ADC_SQR4	Reserved	Regular channel sequence SQx_x bits																													
	Reset value		0	0	0	0	0	0	0	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	ADC_SQR5	Reserved	Regular channel sequence SQx_x bits																													
	Reset value		0	0	0	0	0	0	0	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	ADC_JSQR	Reserved									JL[1:0]		Injected channel sequence JSQx_x bits																		
	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	ADC_JDR1	Reserved															JDATA[15:0]														
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	ADC_JDR2	Reserved															JDATA[15:0]														
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	ADC_JDR3	Reserved															JDATA[15:0]														
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 51. ADC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x54	**ADC_JDR4**	Reserved																JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x58	**ADC_DR**	Reserved																Regular DATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C	**ADC_SMPR0**	Sample time bits SMPx_x																															
	Reset value	0	0	0	0	0	0	0	0	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 52. ADC register map and reset values (common registers)

Off-set	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**ADC_CSR**	Reserved																									ADONS	OVR	STRT	JSTRT	JEOC	EOC	AWD
	Reset value																										0	0	0	0	0	0	0
																											ADC1						
0x04	**ADC_CCR**	Reserved								TSVREFE	Reserved					ADCPRE		Reserved															
	Reset value									0						0	0																

Refer to *Table 2 on page 42* for the *Register boundary addresses* table.

12 Digital-to-analog converter (DAC)

12.1 DAC introduction

The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data could be left- or right-aligned. The DAC has two output channels, each with its own converter. In dual DAC channel mode, conversions could be done independently or simultaneously when both channels are grouped together for synchronous update operations. An input reference pin, V_{REF+} (shared with ADC) is available for better resolution.

12.2 DAC main features

- Two DAC converters: one output channel each
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave generation
- Triangular-wave generation
- Dual DAC channel for independent or simultaneous conversions
- DMA capability for each channel
- DMA underrun error detection
- External triggers for conversion
- Input voltage reference, V_{REF+}

Figure 52 shows the block diagram of a DAC channel and *Table 53* gives the pin description.

Figure 52. DAC channel block diagram

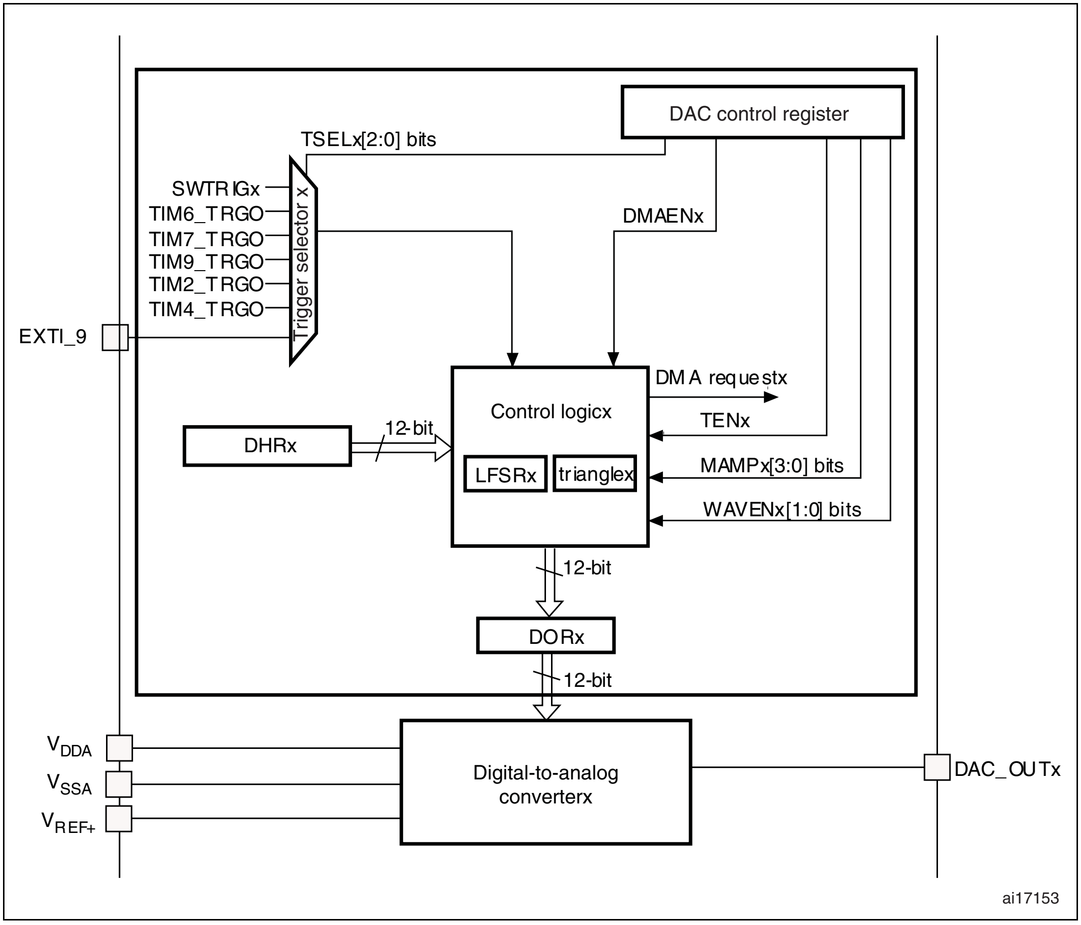


Table 53. DAC pins

Name	Signal type	Remarks
V_{REF+}	Input, analog reference positive	The higher/positive reference voltage for the DAC, 1.8 V $\leq V_{REF+} \leq V_{DDA}$
V_{DDA}	Input, analog supply	Analog power supply
V_{SSA}	Input, analog supply ground	Ground for analog power supply
DAC_OUTx	Analog output signal	DAC channelx analog output

Note: Once the DAC channelx is enabled, the corresponding GPIO pin (PA4 or PA5) is automatically connected to the analog converter output (DAC_OUTx). In order to avoid parasitic consumption, the PA4 or PA5 pin should first be configured to analog (AIN).

12.3 DAC functional description

12.3.1 DAC channel enable

Each DAC channel can be powered on by setting its corresponding ENx bit in the DAC_CR register. The DAC channel is then enabled after a startup time t_{WAKEUP}.

Note: The ENx bit enables the analog DAC Channelx macrocell only. The DAC Channelx digital interface is enabled even if the ENx bit is reset.

12.3.2 DAC output buffer enable

The DAC integrates two output buffers that can be used to reduce the output impedance, and to drive external loads directly without having to add an external operational amplifier. Each DAC channel output buffer can be enabled and disabled using the corresponding BOFFx bit in the DAC_CR register.

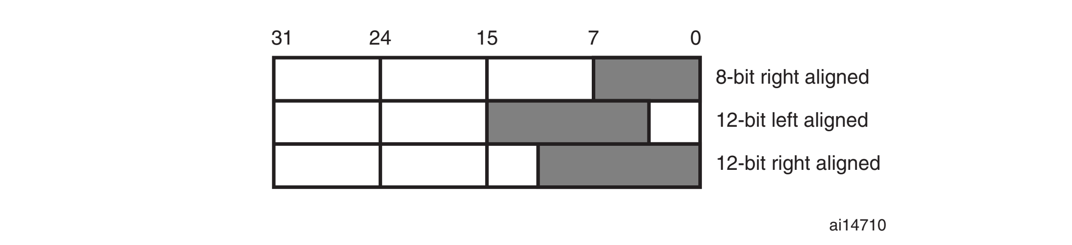
12.3.3 DAC data format

Depending on the selected configuration mode, the data have to be written into the specified register as described below:

- Single DAC channelx, there are three possibilities:
  - 8-bit right alignment: the software has to load data into the DAC_DHR8Rx [7:0] bits (stored into the DHRx[11:4] bits)
  - 12-bit left alignment: the software has to load data into the DAC_DHR12Lx [15:4] bits (stored into the DHRx[11:0] bits)
  - 12-bit right alignment: the software has to load data into the DAC_DHR12Rx [11:0] bits (stored into the DHRx[11:0] bits)

Depending on the loaded DAC_DHRyyyx register, the data written by the user is shifted and stored into the corresponding DHRx (data holding registerx, which are internal non-memory-mapped registers). The DHRx register is then loaded into the DORx register either automatically, by software trigger or by an external event trigger.

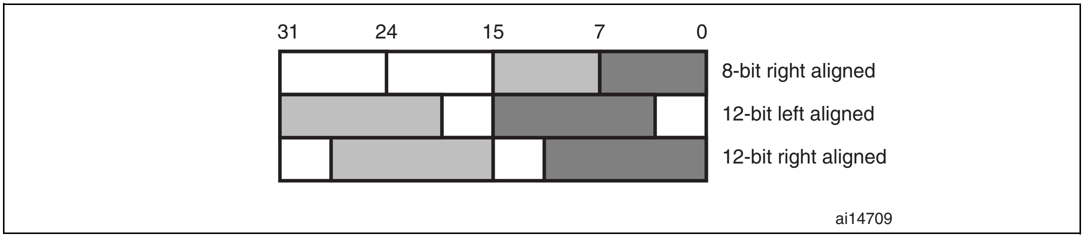
Figure 53. Data registers in single DAC channel mode



- Dual DAC channels, there are three possibilities:
  - 8-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR8RD [7:0] bits (stored into the DHR1[11:4] bits) and data for DAC channel2 to be loaded into the DAC_DHR8RD [15:8] bits (stored into the DHR2[11:4] bits)
  - 12-bit left alignment: data for DAC channel1 to be loaded into the DAC_DHR12LD [15:4] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12LD [31:20] bits (stored into the DHR2[11:0] bits)
  - 12-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR12RD [11:0] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12LD [27:16] bits (stored into the DHR2[11:0] bits)

Depending on the loaded DAC_DHRyyyD register, the data written by the user is shifted and stored into DHR1 and DHR2 (data holding registers, which are internal non-memory-mapped registers). The DHR1 and DHR2 registers are then loaded into the DOR1 and DOR2 registers, respectively, either automatically, by software trigger or by an external event trigger.

Figure 54. Data registers in dual DAC channel mode



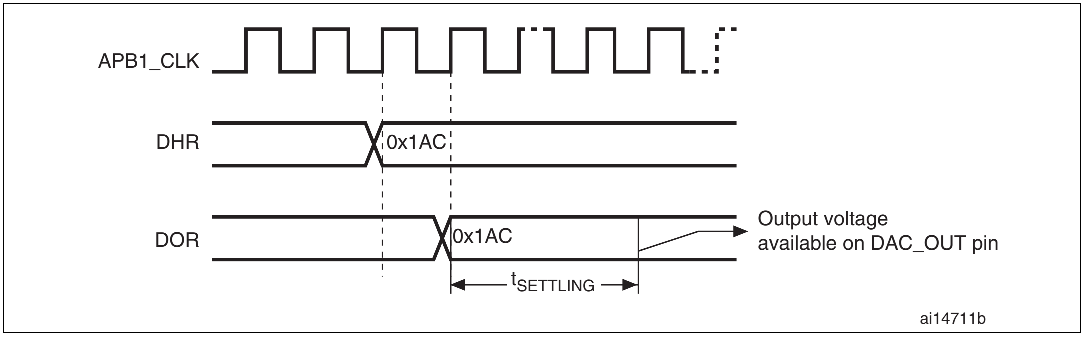
12.3.4 DAC conversion

The DAC_DORx cannot be written directly and any data transfer to the DAC channelx must be performed by loading the DAC_DHRx register (write to DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx, DAC_DHR8RD, DAC_DHR12LD or DAC_DHR12LD).

Data stored in the DAC_DHRx register are automatically transferred to the DAC_DORx register after one APB1 clock cycle, if no hardware trigger is selected (TENx bit in DAC_CR register is reset). However, when a hardware trigger is selected (TENx bit in DAC_CR register is set) and a trigger occurs, the transfer is performed three APB1 clock cycles later.

When DAC_DORx is loaded with the DAC_DHRx contents, the analog output voltage becomes available after a time $t_{SETTLING}$ that depends on the power supply voltage and the analog output load.

Figure 55. Timing diagram for conversion with trigger disabled TEN = 0



12.3.5 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between 0 and V_{REF+}.

The analog output voltages on each DAC channel pin are determined by the following equation:

$$\text{DACoutput} = V_{REF} \times \frac{\text{DOR}}{4095}$$

12.3.6 DAC trigger selection

If the TENx control bit is set, conversion can then be triggered by an external event (timer counter, external interrupt line). The TSELx[2:0] control bits determine which out of 8 possible events will trigger conversion as shown in *Table 54*.

Table 54. External triggers

Source	Type	TSEL[2:0]
Timer 6 TRGO event	Internal signal from on-chip timers	000
Reserved		001
Timer 7 TRGO event		010
Timer 9 TRGO event		011
Timer 2 TRGO event		100
Timer 4 TRGO event		101
EXTI line9	External pin	110
SWTRIG	Software control bit	111

Each time a DAC interface detects a rising edge on the selected timer TRGO output, or on the selected external interrupt line 9, the last data stored into the DAC_DHRx register are transferred into the DAC_DORx register. The DAC_DORx register is updated three APB1 cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC_DORx register has been loaded with the DAC_DHRx register contents.

Note: TSELx[2:0] bit cannot be changed when the ENx bit is set.

When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DORx register takes only one APB1 clock cycle.

12.3.7 DMA request

Each DAC channel has a DMA capability. Two DMA channels are used to service DAC channel DMA requests.

A DAC DMA request is generated when an external trigger (but not a software trigger) occurs while the DMAENx bit is set. The value of the DAC_DHRx register is then transferred into the DAC_DORx register.
In dual mode, if both DMAENx bits are set, two DMA requests are generated. If only one DMA request is needed, you should set only the corresponding DMAENx bit. In this way, the application can manage both DAC channels in dual mode by using one DMA request and a unique DMA channel.

DMA underrun

The DAC DMA request is not queued so that if a second external trigger arrives before the acknowledgement for the first external trigger is received (first request), then no new request is issued and the DMA channelx underrun flag DMAUDRx in the DAC_SR register is set, reporting the error condition. DMA data transfers are then disabled and no further DMA request is treated. The DAC channelx continues to convert old data.

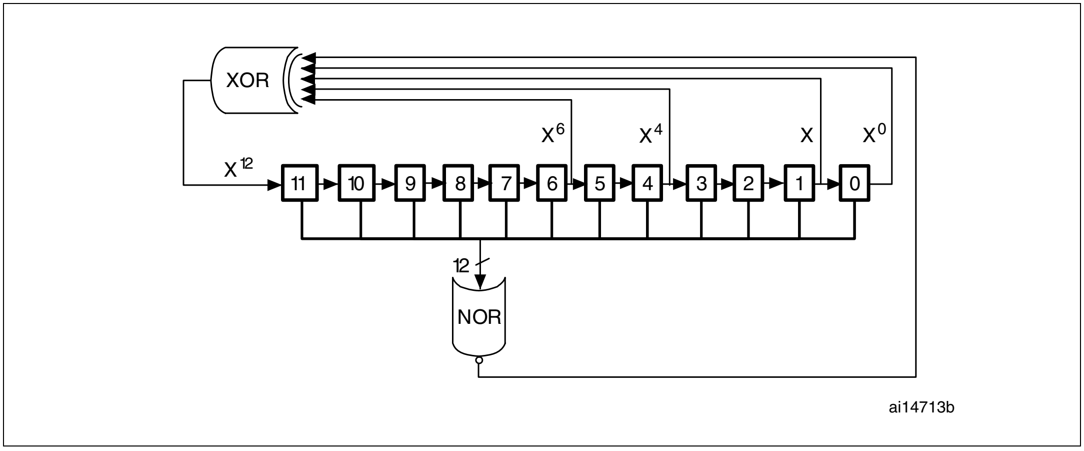
The software should clear the DMAUDRx flag by writing “1”, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channelx to restart the transfer correctly. The software should modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA underrun. Finally, the DAC conversion could be resumed by enabling both DMA data transfer and conversion trigger.

For each DAC channelx, an interrupt is also generated if its corresponding DMAUDRIEx bit in the DAC_CR register is enabled.

12.3.8 Noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVEx[1:0] to “01”. The preloaded value in LFSR is 0xAAA. This register is updated three APB1 clock cycles after each trigger event, following a specific calculation algorithm.

Figure 56. DAC LFSR register calculation algorithm

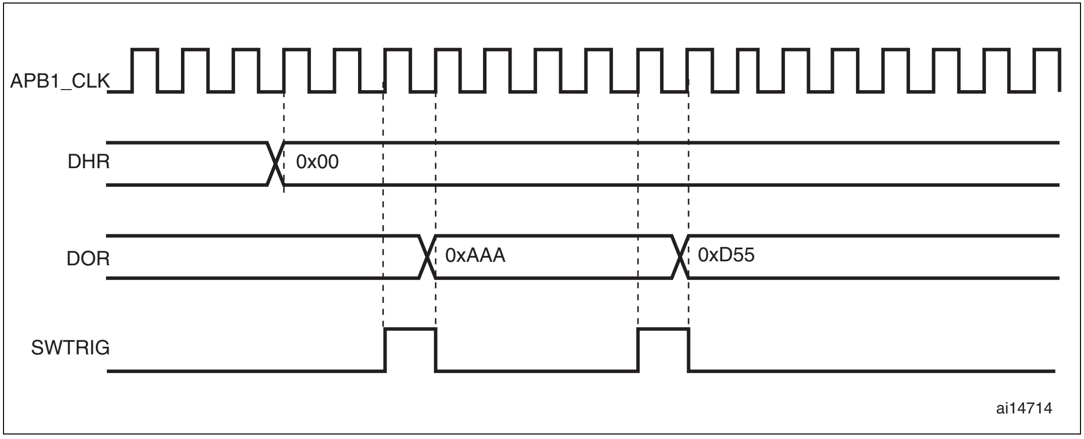


The LFSR value, that may be masked partially or totally by means of the MAMPx[3:0] bits in the DAC_CR register, is added up to the DAC_DHRx contents without overflow and this value is then stored into the DAC_DORx register.

If LFSR is 0x0000, a ‘1 is injected into it (antilock-up mechanism).

It is possible to reset LFSR wave generation by resetting the WAVEx[1:0] bits.

Figure 57. DAC conversion (SW trigger enabled) with LFSR wave generation



Note: The DAC trigger must be enabled for noise generation by setting the TENx bit in the DAC_CR register.

12.3.9 Triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVEx[1:0] to “10”. The amplitude is configured through the MAMPx[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three APB1 clock cycles after each trigger event. The value of this counter is then added to the DAC_DHRx register without overflow and the sum is stored into the DAC_DORx register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the MAMPx[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the WAVEx[1:0] bits.

Figure 58. DAC triangle wave generation

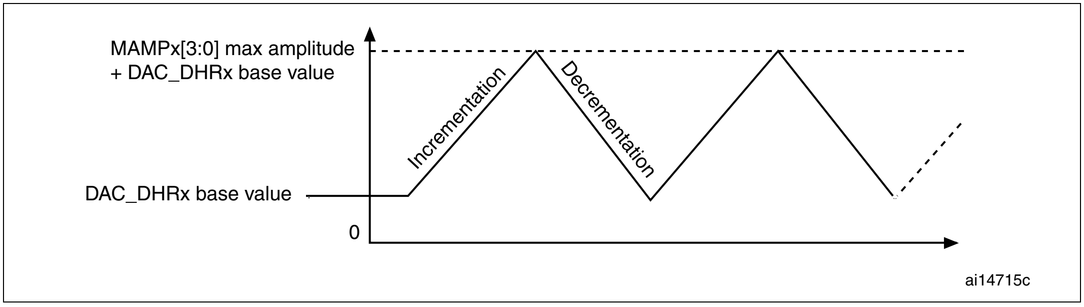
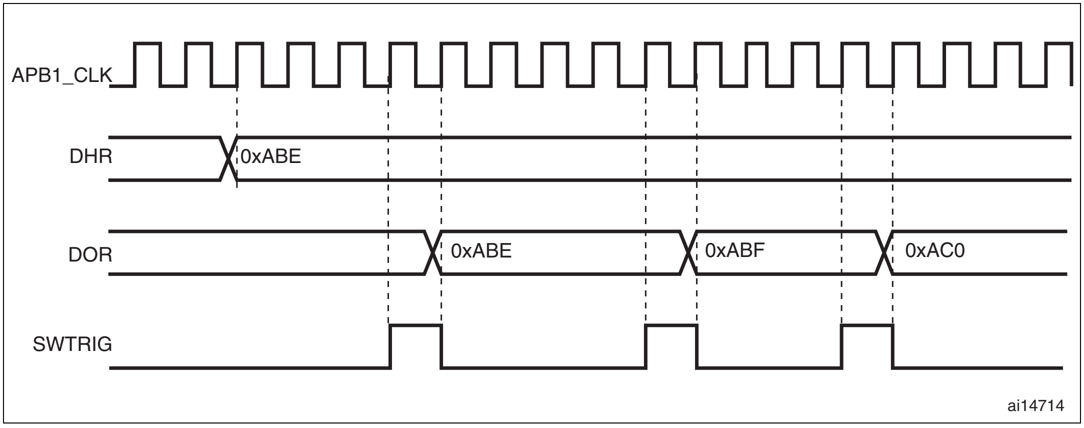


Figure 59. DAC conversion (SW trigger enabled) with triangle wave generation



Note: The DAC trigger must be enabled for noise generation by setting the TENx bit in the DAC_CR register.

The MAMPx[3:0] bits must be configured before enabling the DAC, otherwise they cannot be changed.

12.4 Dual DAC channel conversion

To efficiently use the bus bandwidth in applications that require the two DAC channels at the same time, three dual registers are implemented: DHR8RD, DHR12RD and DHR12LD. A unique register access is then required to drive both DAC channels at the same time.

Eleven possible conversion modes are possible using the two DAC channels and these dual registers. All the conversion modes can nevertheless be obtained using separate DHRx registers if needed.

All modes are described in the paragraphs below.

12.4.1 Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a DAC channel1 trigger arrives, the DHR1 register is transferred into DAC_DOR1 (three APB1 clock cycles later).

When a DAC channel2 trigger arrives, the DHR2 register is transferred into DAC_DOR2 (three APB1 clock cycles later).

12.4.2 Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “01” and the same LFSR mask value in the MAMPx[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DHR12RD, DHR12LD or DHR8RD)

When a DAC channel1 trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). Then the LFSR2 counter is updated.

12.4.3 Independent trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “01” and set different LFSR masks values in the MAMP1[3:0] and MAMP2[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a DAC channel1 trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). Then the LFSR2 counter is updated.

12.4.4 Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “1x” and the same maximum amplitude value in the MAMPx[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The DAC channel2 triangle counter is then updated.

12.4.5 Independent trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure different trigger sources by setting different values in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “1x” and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The DAC channel2 triangle counter is then updated.

12.4.6 Simultaneous software start

To configure the DAC in this conversion mode, the following sequence is required:

- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

In this configuration, one APB1 clock cycle later, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively.

12.4.7 Simultaneous trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a trigger arrives, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively (after three APB1 clock cycles).

12.4.8 Simultaneous trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “01” and the same LFSR mask value in the MAMPx[3:0] bits
- Load the dual DAC channel data to the desired DHR register (DHR12RD, DHR12LD or DHR8RD)

When a trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). The LFSR1 counter is then updated. At the same time, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The LFSR2 counter is then updated.

12.4.9 Simultaneous trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “01” and set different LFSR mask values using the MAMP1[3:0] and MAMP2[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). The LFSR1 counter is then updated.
At the same time, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The LFSR2 counter is then updated.

12.4.10 Simultaneous trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “1x” and the same maximum amplitude value using the MAMPx[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). The DAC channel1 triangle counter is then updated.
At the same time, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). The DAC channel2 triangle counter is then updated.

12.4.11 Simultaneous trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits TEN1 and TEN2
- Configure the same trigger source for both DAC channels by setting the same value in the TSEL1[2:0] and TSEL2[2:0] bits
- Configure the two DAC channel WAVEx[1:0] bits as “1x” and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits
- Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD)

When a trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB1 clock cycles later). Then the DAC channel1 triangle counter is updated.
At the same time, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three APB1 clock cycles later). Then the DAC channel2 triangle counter is updated.

12.5 DAC registers

Refer to *Section 1.1 on page 37* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

12.5.1 DAC control register (DAC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DMAU DRIE2	DMA EN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[2:0]			TEN2	BOFF2	EN2
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		DMAU DRIE1	DMA EN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[2:0]			TEN1	BOFF1	EN1
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 29 **DMAUDRIE2**: DAC channel2 DMA underrun interrupt enable

This bit is set and cleared by software.
0: DAC channel2 DMA underrun interrupt disabled
1: DAC channel2 DMA underrun interrupt enabled

Bit 28 **DMAEN2**: DAC channel2 DMA enable

This bit is set and cleared by software.
0: DAC channel2 DMA mode disabled
1: DAC channel2 DMA mode enabled

Bits 27:24 **MAMP2[3:0]**: DAC channel2 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.
0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bits 23:22 **WAVE2[1:0]**: DAC channel2 noise/triangle wave generation enable

These bits are set/reset by software.
00: wave generation disabled
01: Noise wave generation enabled
1x: Triangle wave generation enabled

Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled)

Bits 21:19 **TSEL2[2:0]**: DAC channel2 trigger selection

These bits select the external event used to trigger DAC channel2
000: Timer 6 TRGO event
001: Reserved
010: Timer 7 TRGO event
011: Timer 9 TRGO event
100: Timer 2 TRGO event
101: Timer 4 TRGO event
110: External line9
111: Software trigger

Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled).

Bit 18 **TEN2**: DAC channel2 trigger enable

This bit is set and cleared by software to enable/disable DAC channel2 trigger
0: DAC channel2 trigger disabled and data written into the DAC_DHRx register are transferred one APB1 clock cycle later to the DAC_DOR2 register
1: DAC channel2 trigger enabled and data from the DAC_DHRx register are transferred three APB1 clock cycles later to the DAC_DOR2 register

Note: When software trigger is selected, the transfer from the DAC_DHRx reg*ister to the DAC_DOR2 register takes only one APB1 clock cycle.*

Bit 17 **BOFF2**: DAC channel2 output buffer disable

This bit is set and cleared by software to enable/disable DAC channel2 output buffer.
0: DAC channel2 output buffer enabled
1: DAC channel2 output buffer disabled

Bit 16 **EN2**: DAC channel2 enable

This bit is set and cleared by software to enable/disable DAC channel2.
0: DAC channel2 disabled
1: DAC channel2 enabled

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **DMAUDRIE1**: DAC channel1 DMA Underrun Interrupt enable

This bit is set and cleared by software.
0: DAC channel1 DMA Underrun Interrupt disabled
1: DAC channel1 DMA Underrun Interrupt enabled

Bit 12 **DMAEN1**: DAC channel1 DMA enable

This bit is set and cleared by software.
0: DAC channel1 DMA mode disabled
1: DAC channel1 DMA mode enabled

Bits 11:8 **MAMP1[3:0]**: DAC channel1 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bits 7:6 **WAVE1[1:0]**: DAC channel1 noise/triangle wave generation enable

These bits are set and cleared by software.

00: wave generation disabled
01: Noise wave generation enabled
1x: Triangle wave generation enabled

Note: Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bits 5:3 **TSEL1[2:0]**: DAC channel1 trigger selection

These bits select the external event used to trigger DAC channel1.

000: Timer 6 TRGO event
001: Reserved
010: Timer 7 TRGO event
011: Timer 9 TRGO event
100: Timer 2 TRGO event
101: Timer 4 TRGO event
110: External line9
111: Software trigger

Note: Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bit 2 **TEN1**: DAC channel1 trigger enable

This bit is set and cleared by software to enable/disable DAC channel1 trigger.

0: DAC channel1 trigger disabled and data written into the DAC_DHRx register are transferred one APB1 clock cycle later to the DAC_DOR1 register

1: DAC channel1 trigger enabled and data from the DAC_DHRx register are transferred three APB1 clock cycles later to the DAC_DOR1 register

Note: When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DOR1 register takes only one APB1 clock cycle.

Bit 1 **BOFF1**: DAC channel1 output buffer disable

This bit is set and cleared by software to enable/disable DAC channel1 output buffer.

0: DAC channel1 output buffer enabled
1: DAC channel1 output buffer disabled

Bit 0 **EN1**: DAC channel1 enable

This bit is set and cleared by software to enable/disable DAC channel1.

0: DAC channel1 disabled
1: DAC channel1 enabled

12.5.2 DAC software trigger register (DAC_SWTRIGR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved														SWTRIG2	SWTRIG1
														w	w

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

Bit 1 **SWTRIG2**: DAC channel2 software trigger

This bit is set and cleared by software to enable/disable the software trigger.

0: Software trigger disabled

1: Software trigger enabled

Note: This bit is cleared by hardware (one APB1 clock cycle later) once the DAC_DHR2 register value has been loaded into the DAC_DOR2 register.

Bit 0 **SWTRIG1**: DAC channel1 software trigger

This bit is set and cleared by software to enable/disable the software trigger.

0: Software trigger disabled

1: Software trigger enabled

Note: This bit is cleared by hardware (one APB1 clock cycle later) once the DAC_DHR1 register value has been loaded into the DAC_DOR1 register.

12.5.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data

These bits are written by software which specifies 12-bit data for DAC channel1.

12.5.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DACC1DHR[11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data

These bits are written by software which specifies 12-bit data for DAC channel1.

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

12.5.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved								DACC1DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data

These bits are written by software which specifies 8-bit data for DAC channel1.

12.5.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2)

Address offset: 0x14

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data

These bits are written by software which specifies 12-bit data for DAC channel2.

12.5.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2)

Address offset: 0x18

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data

These bits are written by software which specify 12-bit data for DAC channel2.

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

12.5.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2)

Address offset: 0x1C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DACC2DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data

These bits are written by software which specifies 8-bit data for DAC channel2.

12.5.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 27:16 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data

These bits are written by software which specifies 12-bit data for DAC channel2.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data

These bits are written by software which specifies 12-bit data for DAC channel1.

12.5.10 DUAL DAC 12-bit left aligned data holding register (DAC_DHR12LD)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHR[11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data

These bits are written by software which specifies 12-bit data for DAC channel2.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data

These bits are written by software which specifies 12-bit data for DAC channel1.

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

12.5.11 DUAL DAC 8-bit right aligned data holding register (DAC_DHR8RD)

Address offset: 0x28

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7:0]								DACC1DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data

These bits are written by software which specifies 8-bit data for DAC channel2.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data

These bits are written by software which specifies 8-bit data for DAC channel1.

12.5.12 DAC channel1 data output register (DAC_DOR1)

Address offset: 0x2C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC1DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11:0 **DACC1DOR[11:0]**: DAC channel1 data output

These bits are read-only, they contain data output for DAC channel1.

12.5.13 DAC channel2 data output register (DAC_DOR2)

Address offset: 0x30
Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC2DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DOR[11:0]**: DAC channel2 data output

These bits are read-only, they contain data output for DAC channel2.

12.5.14 DAC status register (DAC_SR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DMAUDR2	Reserved												
		rc_w1													
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		DMAUDR1	Reserved												
		rc_w1													

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

Bit 29 **DMAUDR2**: DAC channel2 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).
0: No DMA underrun error condition occurred for DAC channel2
1: DMA underrun error condition occurred for DAC channel2 (the currently selected trigger is driving DAC channel2 conversion at a frequency higher than the DMA service capability rate)

Bits 28:14 Reserved, must be kept at reset value.

Bit 13 **DMAUDR1**: DAC channel1 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).
0: No DMA underrun error condition occurred for DAC channel1
1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)

Bits 12:0 Reserved, must be kept at reset value.

12.5.15 DAC register map

Table 55 summarizes the DAC registers.

Table 55. DAC register map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**DAC_CR**	Reserved		DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE 2[2:0]		TSEL2[2:0]			TEN2	BOFF2	EN2	Reserved		DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE 1[2:0]		TSEL1[2 :0]			TEN1	BOFF1	EN1
0x04	**DAC_SWT RIGR**	Reserved																														SWTRIG2	SWTRIG1
0x08	**DAC_DHR 12R1**	Reserved																				DACC1DHR[11:0]											
0x0C	**DAC_DHR 12L1**	Reserved																DACC1DHR[11:0]												Reserved			

Table 55. DAC register map (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x10	DAC_DHR8R1	Reserved																								DACC1DHR[7:0]							
0x14	DAC_DHR12R2	Reserved																				DACC2DHR[11:0]											
0x18	DAC_DHR12L2	Reserved																DACC2DHR[11:0]												Reserved			
0x1C	DAC_DHR8R2	Reserved																								DACC2DHR[7:0]							
0x20	DAC_DHR12RD	Reserved				DACC2DHR[11:0]												Reserved				DACC1DHR[11:0]											
0x24	DAC_DHR12LD	DACC2DHR[11:0]												Reserved				DACC1DHR[11:0]												Reserved			
0x28	DAC_DHR8RD	Reserved																DACC2DHR[7:0]								DACC1DHR[7:0]							
0x2C	DAC_DOR1	Reserved																				DACC1DOR[11:0]											
0x30	DAC_DOR2	Reserved																				DACC2DOR[11:0]											
0x34	DAC_SR	Reserved		DMAUDR2	Reserved															DMAUDR1	Reserved												

Refer to *Table 2 on page 42* for the register boundary addresses.

13 Comparators (COMP)

This section applies to the whole STM32L1xxxx family, unless otherwise specified.

13.1 Introduction

The STM32L1xxxx contains two zero-crossing comparators COMP1 and COMP2, that share the same current bias.

Note: For all I/Os used as comparator inputs, the GPIO registers must be configured in analog mode.

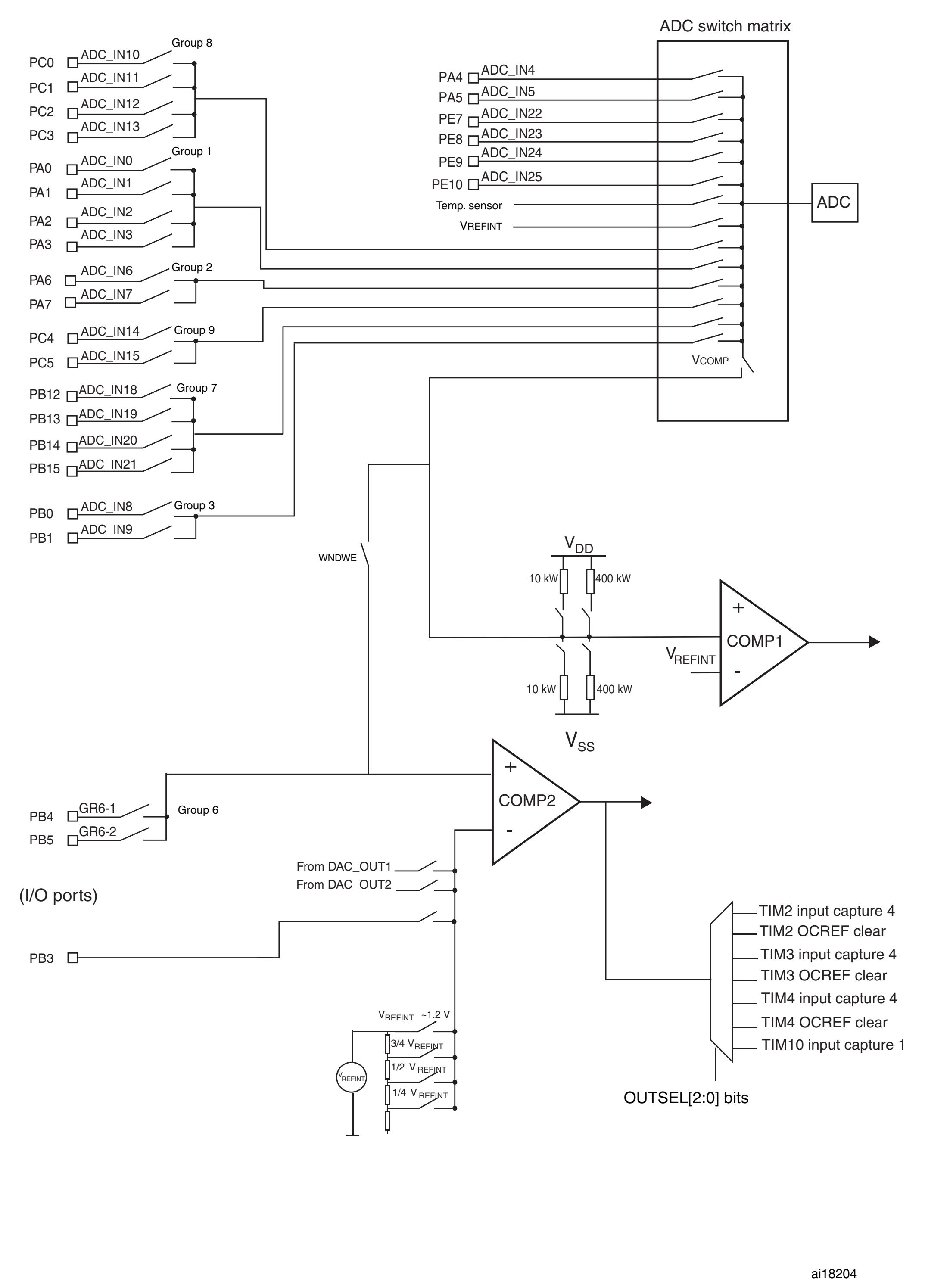
When using the routing interface (see Section 7: System configuration controller (SYSCFG) and routing interface (RI)), the comparator inputs can be connected to external I/Os.

13.2 Main features

- A comparator (COMP1) with fixed threshold (internal reference voltage). The non-inverting input can be selected among 24 external I/Os.
- A rail-to-rail comparator (COMP2) with selectable threshold. The non-inverting input can be selected among 2 I/Os. The threshold can be selected among:
  - the internal reference voltage (V_{REFINT})
  - an internal reference voltage submultiple (1/4, 1/2, 3/4)
  - the DAC1 output
  - the DAC2 output
  - an external I/O (PB3)
- The 2 comparators can be combined to form window comparators.
- Zero-crossing can generate a rising or falling edge on the comparator outputs depending on the trigger configuration.
- Each comparator has an interrupt generation capability with wakeup from the Sleep and Stop.
- The COMP2 output can be redirected to TIM2/TIM3/TIM4's input capture 4 (IC4) or OCREF_CLR inputs, or to the TIM10s input capture 1 (IC1).
- COMP2 speed is configurable for optimum speed/consumption ratio.

The block diagram of the COMP is shown in *Figure 60*.

Figure 60. Comparator block diagram



Note: The internal reference voltage and temperature sensor cannot be used as COMP1 non-inverting input.

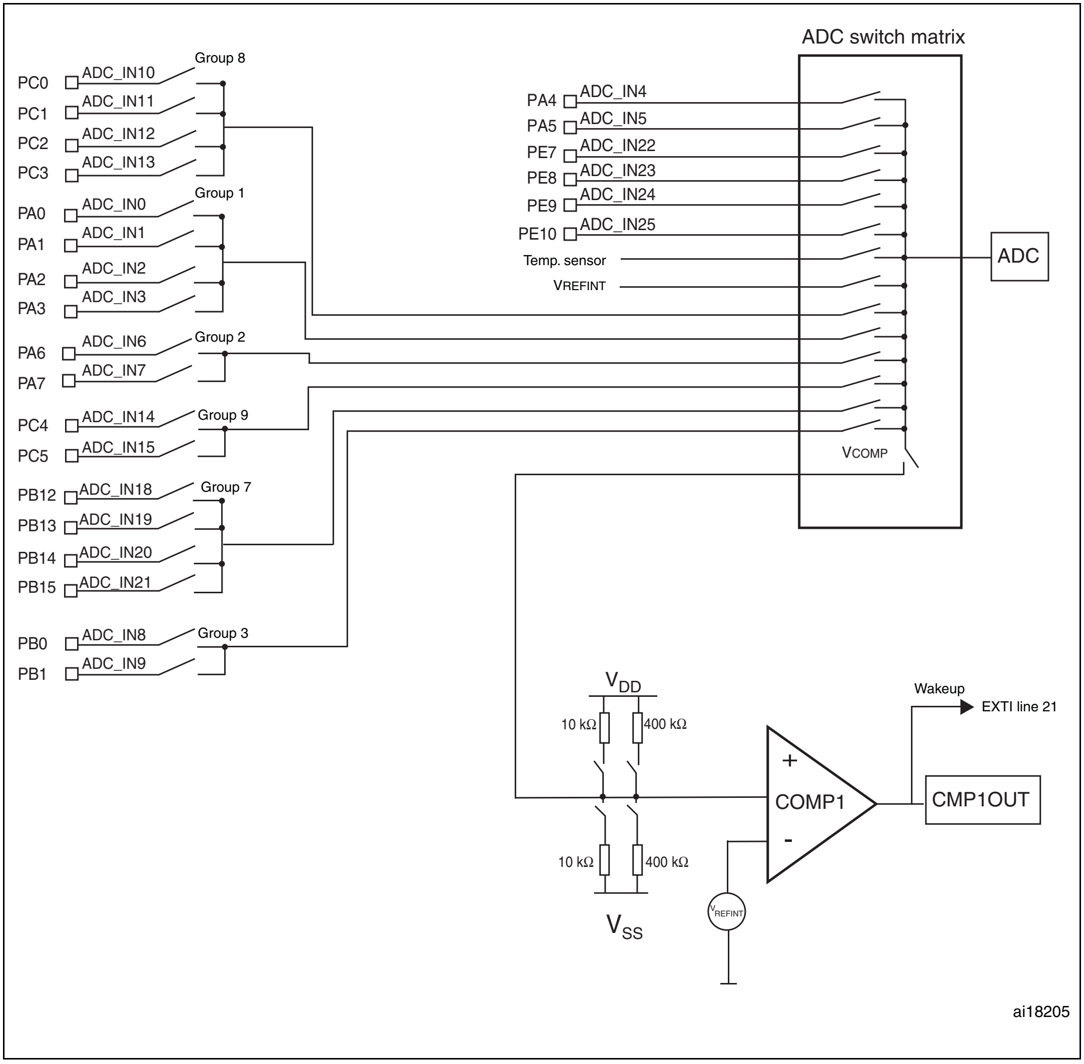
13.3 COMP clock

The COMP clock provided by the clock controller is synchronous with the PCLK1 (APB1 clock).

13.4 Comparator 1 (COMP1)

Figure 61 and *Figure 62* show the comparator 1 interconnections.

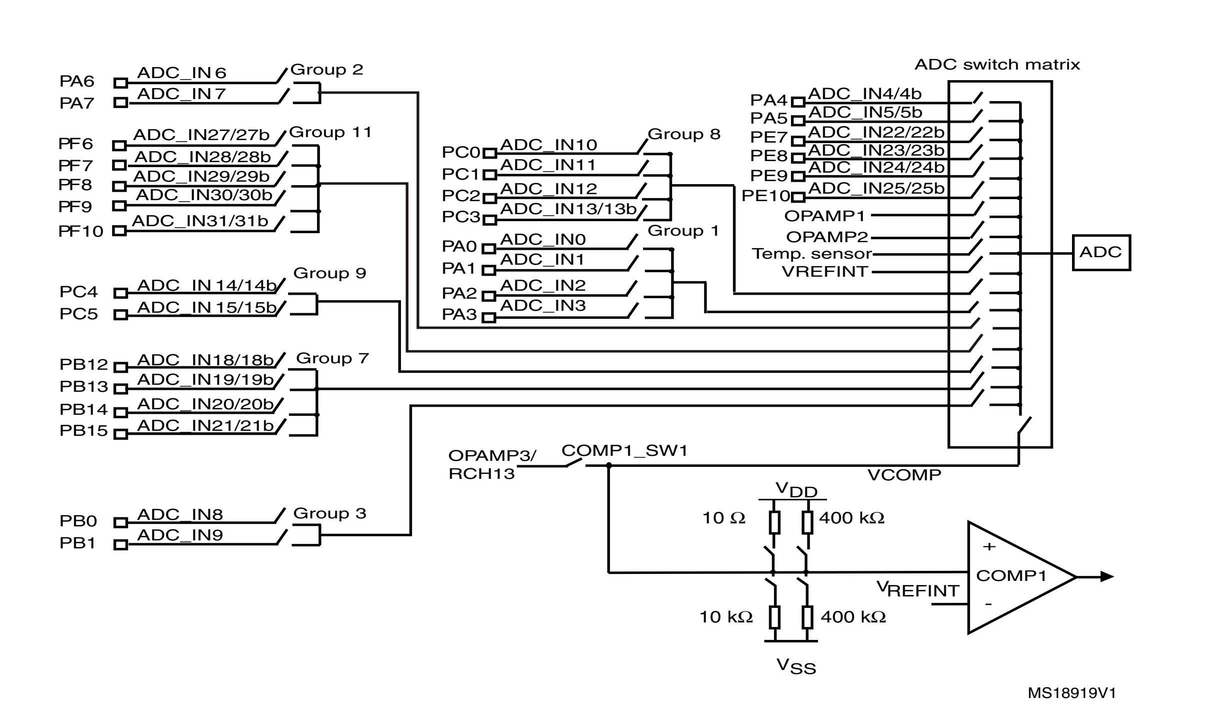
Figure 61. COMP1 interconnections (low and medium density devices)



Note: The internal reference voltage and temperature sensor cannot be used as COMP1 non-inverting input.

COMP1 comparator and ADC cannot be used at the same time since they share the ADC switch matrix.

Figure 62. COMP1 interconnections (high and medium+ density devices)



Note: The internal reference voltage and temperature sensor cannot be used as COMP1 non-inverting input.

COMP1 comparator and ADC cannot be used at the same time since they share the ADC switch matrix.

To use the COMP1 comparator, the application has to perform the following steps:

1. Enable the comparator 1 by setting the CMP1EN bit in the COMP_CSR register
2. Wait until the comparator is ready (when the startup time has elapsed). Refer to the electrical characteristics of the STM32L1xxxx datasheet.
3. Set the SCM bit in the RI_ASCR1 register so as to close the ADC switches if the corresponding I/O switch is also closed
4. Close the ADC switches to create the path from the selected I/O to the non-inverting input. The input can be any of the up to 29 available I/Os and can be split into groups or not (see *Figure 27: I/O groups and selection on page 155*):
   a) Close the VCOMP ADC analog switch by setting the VCOMP bit in the RI_ASCR1 register.
   b) Close the I/O analog switch number n corresponding to the I/O group that must be connected to the COMP1 non-inverting input, by setting the CHn bit in RI_ASCR1.
5. If required enable the COMP1 interrupt by configuring and enabling EXTI line21 in interrupt mode and selecting the desired trigger event (rising edge, falling edge or both).

13.5 Comparator 2 (COMP2)

Figure 63 and *Figure 64* show the comparator 2 interconnections.

Figure 63. COMP2 interconnections (low and medium density devices)

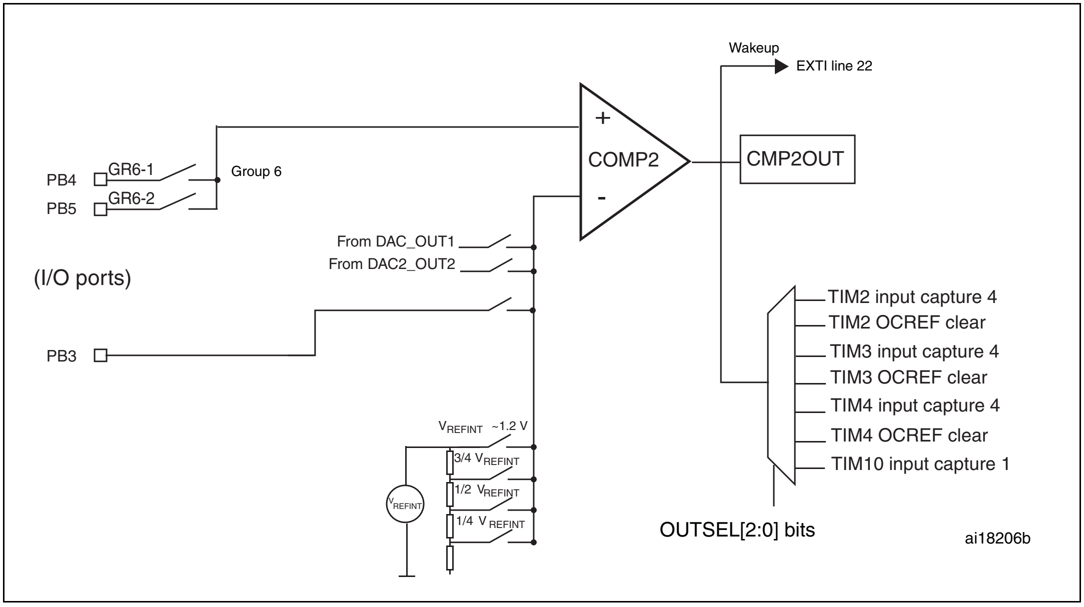
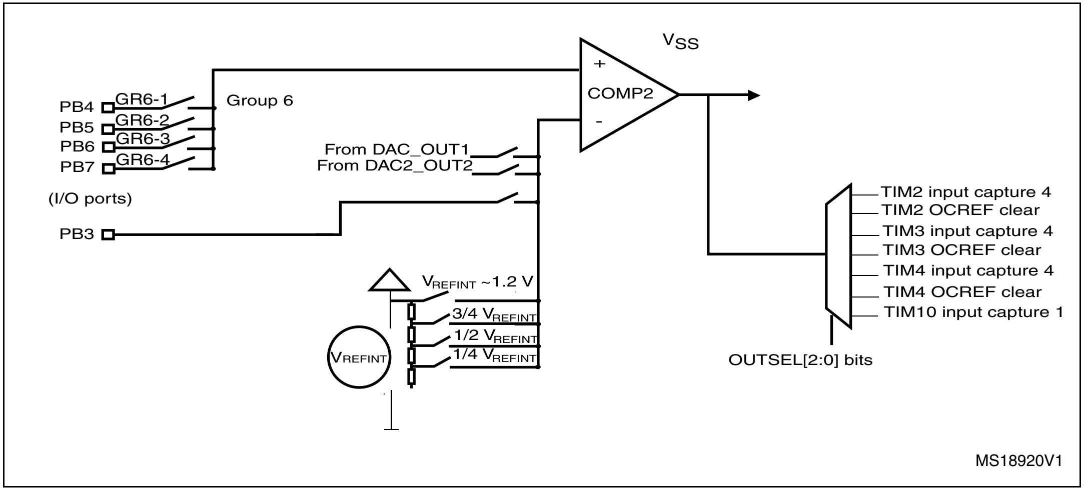


Figure 64. COMP2 interconnections (high and medium+ density devices)



To use the COMP2 comparator, the application has to perform the following steps:

1. Select COMP2's inverting input with the INSEL[2:0] bits in COMP_CSR.
   – In the case of an external I/O selection (PB3 I/O), the I/O should be configured in analog input mode.
2. Close the I/O's analog switch to connect to COMP2 non-inverting input. The input can be any I/O in group 6 (see *Table 27: I/O groups and selection on page 155*). GR6-1 or

GR6-2 switches are closed as soon as the corresponding I/O is configured in analog mode.

3. Wait until the comparator is ready (when the startup time has elapsed). Refer to the electrical characteristics of the STM32L1xxxx datasheet.
4. If required, perform the following procedures:
   - Select the speed with the SPEED bit in COMP_CSR.
   - Redirect the COMP2 output to TIM2, TIM3, TIM4 or TIM10 by configuring the OUTSEL[2:0] bits in COMP_CSR (refer to *Figure 65*).
   - Enable the COMP2 interrupt by configuring and enabling EXTI line22 in interrupt mode and selecting the desired sensitivity level.

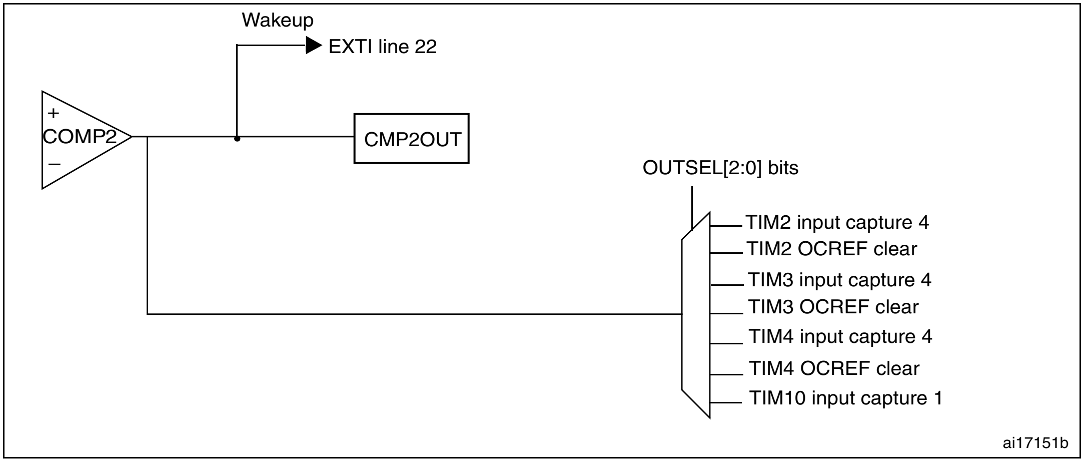
Note: GR6-1 and GR6-2 I/O switches can be closed by either configuring the corresponding I/O (PB4 or PB5) in analog mode (Schmitt trigger disabled) or configuring the I/O in input floating mode and setting GR6-1 or GR6-2 in RI_ASCR2 (Schmitt trigger enabled).
If PB4 or PB5 is used as comparator input, it is recommended to use analog configuration to avoid any overconsumption around V_{DD}/2.

Note: The COMP2 comparator is enabled as soon as the inverting input is selected.

The channel can be changed when the comparator is enabled.

The following figure shows the output redirection possibilities of the COMP2 output.

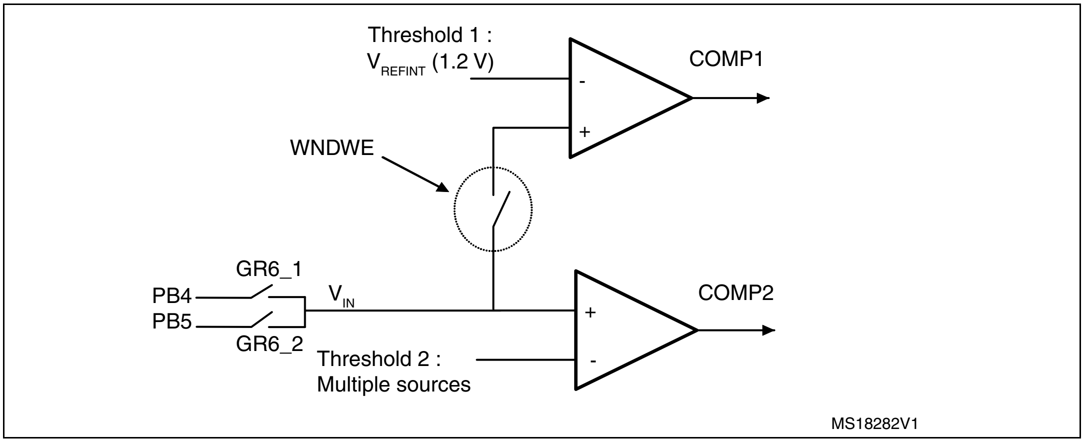
Figure 65. Redirecting the COMP2 output



Note: For more details about “clearing TIMx OCREF”, refer to Section 16.3.11: Clearing the OCxREF signal on an external event on page 363.

13.6 Comparators in Window mode

Figure 66. Comparators in Window mode



To use the COMP1 and COMP2 comparators in window mode, the application has to perform the following steps:

1. Select COMP2’s inverting input as explained in *Section 13.5: Comparator 2 (COMP2)*.
2. Enable the Window mode by setting WNDWE in the COMP_CSR register.
3. Select the non-inverting input:
   - for COMP1: follow the steps 2 and 3 from *Section 13.4: Comparator 1 (COMP1)*
   - for COMP2: follow steps 3 and 4 from *Section 13.5: Comparator 2 (COMP2)*
4. Enable COMP1 by setting the CMP1EN in the COMP_CSR register.

Note: In Window mode, only the Group 6 (PB4 and PB5) can be used as a non-inverting input.

13.7 Low power modes

Table 56. Comparator behavior in the low power modes

Mode	Description
Sleep	No effect on the comparators. Comparator interrupts cause the device to exit the Sleep mode.
Stop	No effect on the comparators. Comparator interrupts cause the device to exit the Stop mode.

Note: Comparators cannot be used to exit the device from Sleep or Stop mode when the internal reference voltage is switched off using the ULP bit in the PWR_CR register.

13.8 Interrupts

The comparator interrupts are connected to EXTI controller (lines 21 and 22).

To enable the COMP interrupt, the following sequence is required:

1. Configure and enable the EXTI line 21 (COMP1) or EXTI line 22 (COMP2) in interrupt mode and select the desired trigger event (rising edge, falling edge, or both),
2. Configure and enable the COMP_IRQ channel in the NVIC.

13.9 COMP registers

The peripheral registers have to be accessed by words (32-bit).

13.9.1 COMP comparator control and status register (COMP_CSR)

The COMP_CSR register is the control/status register of the comparators. It contains all the bits related to both comparators.

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSUSP	CAIF	CAIE	RCH13	FCH8	FCH3	Reserved		OUTSEL[2:0]			INSEL[2:0]			WNDWE	VREFOUTEN
rw	r	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CMP2OUT	SPEED	Reserved				CMP1OUT	Res.	COMP1_SW1	CMP1EN	400KPD	10KPD	400KPU	10KPU
		r	rw					r		rw	rw	rw	rw	rw	rw

Bit 31 **TSUSP:** Suspend Timer Mode

0: TIM9 ITR enabled to suspend OC TIM9 generation

1: TIM9 ITR not used to suspend OC TIM9 generation

Note: This bit is available in high and medium+ density devices only

Bit 30 **CAIF**: Channel acquisition interrupt flag

0: Channel acquisition ongoing or not started

1: Channel acquisition completed

Note: This bit is available in high and medium+ density devices only

Bit 29 **CAIE**: Channel Acquisition Interrupt Enable / Clear

This bit is set and cleared to enable the Channel Acquisition interrupt. When the Caif bit is set, it must be cleared by writing 0 to the CAIE bit.

0: Channel acquisition interrupt disabled

1: Channel acquisition interrupt enabled

This bit is available in high and medium+ density devices only

Bit 28 **RCH13:** Select GPIO port PC3 as re-routed ADC input channel CH13.

This bit is set and cleared by software in order configure PC3 to be used as re-routed channel CH13 (selected by the ADC interface) if OPAMP3 is in power down mode (OPA1PD bit = 0 in OPAMP_CSR register (high density device only). See *Figure 69: OPAMP3 signal routing (high-density devices only) on page 303*.

0: PC3 can be used as slow ADC channel

1: PC3 can be used as re-routed ADC channel

Note: This bit is available in high and medium+ density devices only

Bit 27 **FCH8:** Select GPIO port PB0 as fast ADC input channel CH8.

This bit is set and cleared by software in order configure PB0 to be used as fast channel CH13 (selected by the ADC interface) if OPAMP2 is in power down mode (OPA2PD bit = 0 in OPAMP_CSR register. See *Figure 68: OPAMP2 signal routing on page 303*.

0: PB0 can be used as slow ADC channel

1: PB0 can be used as fast ADC channel

Note: This bit is available in high and medium+ density devices only

Bit 26 **FCH3:** Select GPIO port PA3 as fast ADC input channel CH3.

This bit is set and cleared by software in order configure PA3 to be used as fast channel CH13 (selected by the ADC interface) if OPAMP1 is in power down mode (OPA1PD bit = 0 in OPAMP_CSR register. See *Figure 67: OPAMP1 signal routing on page 302*.

0: PA3 can be used as slow ADC channel

1: PA3 can be used as fast ADC channel

Note: This bit is available in high and medium+ density devices only

Bits 25:24 Reserved, must be kept cleared.

Bits 23:21 **OUTSEL:** Comparator 2 output selection

These bits are written by software to connect the output of COMP2 to a selected timer input.

000 = TIM2 Input Capture 4
001 = TIM2 OCREF_CLR
010 = TIM3 Input Capture 4
011 = TIM3 OCREF_CLR
100 = TIM4 Input Capture 4
101 = TIM4 OCREF_CLR
110 = TIM10 Input Capture 1
111 = no redirection

Bits 20:18 **INSEL:** Inverted input selection

000 = no selection
001 = External I/O: PB3 (COMP2_INM)
010 = V_{REFINT}
011 = 3/4 V_{REFINT}
100 = 1/2 V_{REFINT}
101 = 1/4 V_{REFINT}
110 = DAC_OUT1
111 = DAC_OUT2

Note: The COMP2 comparator is enabled when the INSEL bit values are different from "000".

Bit 17 **WNDWE:** Window mode enable

0: Disabled

1: Enabled

Bit 16 **VREFOUTEN:** V_{REFINT} output enable

This bit is used to output V_{REFINT} on Group 3 (refer to *Figure 25: Internal reference voltage output*).
0: Disabled
1: Enabled

Bits 15:14 Reserved, must be kept at reset value

Bit 13 **CMP2OUT:** Comparator 2 output

This bit indicates the low or high level of the comparator 2 output.

0: Comparator 2 output is low when the non-inverting input is at a lower voltage than the inverting input
1: Comparator 2 output is high when the non-inverting input is at a higher voltage than the inverting input

Bit 12 **SPEED:** Comparator 2 speed mode

0: slow speed
1: fast speed

Bits 11:8 Reserved, must be kept at reset value

Bit 7 **CMP1OUT:** Comparator 1 output

This bit indicates the high or low level of the comparator 1 output.

0: Comparator 1 output is low when the non-inverting input is at a lower voltage than the inverting input
1: Comparator 1 output is high when the non-inverting input is at a higher voltage than the inverting input

Bit 6 Reserved, must be kept at reset value

Bit 5 **SW1**: COMP1_SW1 analog switch enable

This bit is set and cleared by software to control the COMP1_SW1 analog switch in order to redirect OPAMP3 output or PC3 to the ADC switch matrix and/or the negative input of COMP1.

0: COMP1_SW1 analog switch open

1: COMP1_SW1 analog switch closed

Note: This bit is available in high-density devices only

Bit 4 **CMP1EN:** Comparator 1 enable

0: Comparator 1 disabled
1: Comparator 1 enabled

Bit 3 **400KPD:** 400 kΩ pull-down resistor

This bit enables the 400 kΩ pull-down resistor.

0: 400 kΩ pull-down resistor disabled
1: 400 kΩ pull-down resistor enabled

Bit 2 **10KPD:** 10 kΩ pull-down resistor

This bit enables the 10 kΩ pull-down resistor.

0: 10 kΩ pull-down resistor disabled
1: 10 kΩ pull-down resistor enabled

Bit 1 **400KPU:** 400 kΩ pull-up resistor

This bit enables the 400 kΩ pull-up resistor.

0: 400 kΩ pull-up resistor disabled
1: 400 kΩ pull-up resistor enabled

Bit 0 **10KPU:** 10 kΩ pull-up resistor

This bit enables the 10 kΩ pull-up resistor.

0: 10 kΩ pull-up resistor disabled
1: 10 kΩ pull-up resistor enabled

Note: To avoid extra power consumption, only one resistor should be enabled at a time.

13.9.2 COMP register map

Table 57: COMP register map and reset values summarizes the COMP registers.

Table 57. COMP register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**COMP_CSR**	TSUSP	CAIF	CAIE	RCH13	FCH8	FCH3	Res.		OUTSEL [2:0]			INSEL [2:0]			WNDWE	VREFOUTEN	Reserved		CMP2OUT	SPEED	Reserved				CMP1OUT	Reserved	COMP1_SW1	CMP1EN	400KPD	10KPD	400KPU	10KPU
	Reset value	0	0	0	0	0	0			0	0	0	0	0	0	0	0			0	0					0		0	0	0	0	0	0

Refer to *Table 2 on page 42* for the *Register boundary addresses* table.

14 Operational amplifiers (OPAMP)

This section applies to high and medium+ density devices only.

14.1 OPAMP introduction

The MCU has three operational amplifiers with external or internal follower routing capability (or even amplifier and filter capability with external components). When one operational amplifier is selected, one external ADC channel is used to enable output measurement.

14.2 OPAMP main features

- Rail-to-rail input and output voltage range
- Low input bias current
- Low input offset voltage
- Low power mode

14.3 OPAMP functional description

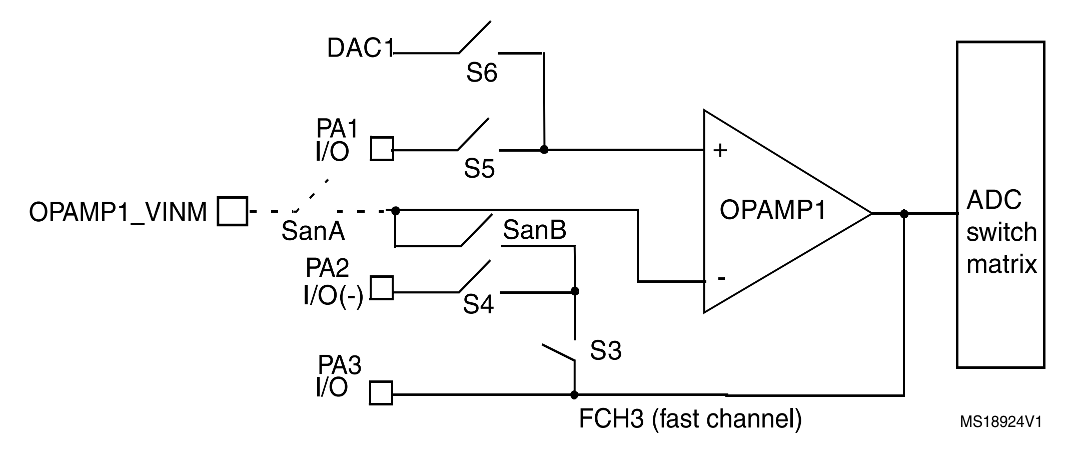
Three operational amplifiers (OPAMP1, OPAMP2 and OPAMP3) are available on high-density devices and two operational amplifiers (OPAMP1 and OPAMP2) are available on medium+ devices. The connection with dedicated I/O are listed below:

- OPAMP1_VINP --> PA1
- OPAMP1_VINM- --> PA2[1]
- OPAMP1_VOUT --> PA3 (ADC input CH3)
- OPAMP2_VINP --> PA6
- OPAMP2_VINM --> PA7 [1]
- OPAMP2_VOUT --> PB0 (ADC input CH8)
- OPAMP3_VINP --> PC1
- OPAMP3_VINM --> PC2 [1]
- OPAMP3_VOUT --> PC3 (ADC input CH13)

[1] Or dedicated OPAMPx_VINM pin available on some packages.

14.3.1 Signal routing

Figure 67. OPAMP1 signal routing



The routing for the three operational amplifiers can be selected by OPAMP_CSR register.

Analog switches S3 to S6 and SanA can be opened and closed by programming the corresponding OPAMP_CSR register bits independently of whether the amplifiers are enabled or not by the OPA1_PD, OPA2_PD and/or OPA3_PD bits.

Analog switch SanB automatically follows the selection of the S3 or S4 switches. It is not controlled individually.

For OPAMP1, S6 is used to connect DAC1 to its positive input.

For OPAMP2, there is an additional S7 switch in parallel with S6 in order to select the positive input source as either I/O or DAC1 or DAC2.

For OPAMP3, S6 is used to connect DAC2 to its positive input.

All operational amplifiers can be powered down by setting the OPAx_PD bit. The corresponding inputs and outputs are then in high impedance.

Figure 68. OPAMP2 signal routing

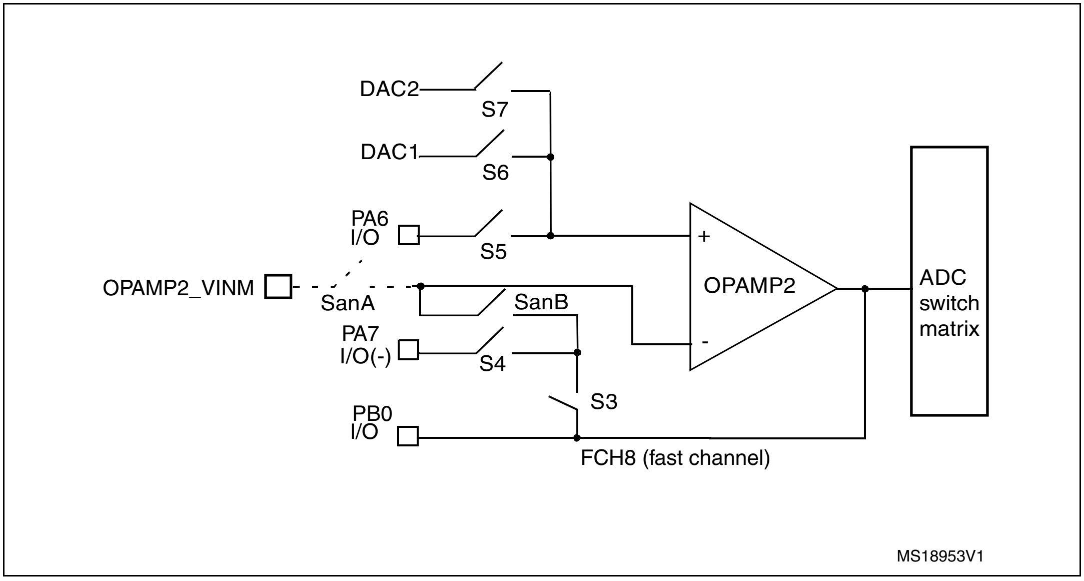
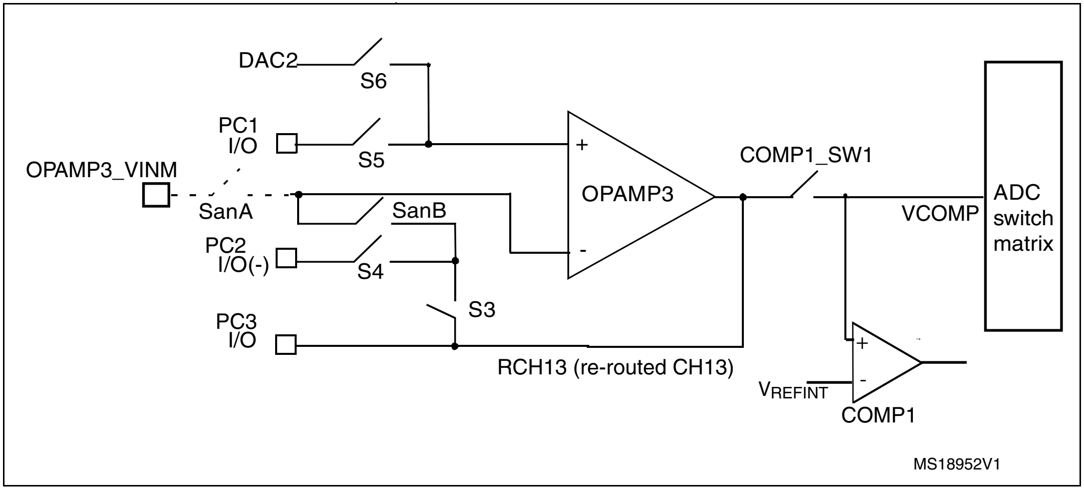


Figure 69. OPAMP3 signal routing (high-density devices only)



14.3.2 Using the OPAMP outputs as ADC inputs

In order to use OPAMP outputs as ADC inputs, the operational amplifiers must be enabled and the ADC must use the OPAMP output channel number. (OPA1: CH3 ; OPA2: CH8 ; OPA3: CH13).

In addition for OPA3 or FCH13, the user must close COMP1_SW1 analog switch to do an acquisition (refer to *Section 13.9.1: COMP comparator control and status register (COMP_CSR) on page 297*).

14.3.3 Calibration

At startup, the trimming values are initialized with the preset ‘factory’ trimming value.

Furthermore each operational amplifier offset can be trimmed by the user. All switches related to the inputs of each operational amplifier must be open during the trimming operation (SanA, S3, S4, S5,S6).

There are two registers for trimming the offsets of the 3 operational amplifiers for normal mode and low power mode. Two words of 30-bits, one for standard mode and the other for low power mode are available in OPAMP_OTR and OPAMP_LPOTR registers. This is the 'user' value.

The user is able to switch from 'factory' values to 'user' trimmed values using the OT_USER bit in the OPAMP_OTR register. This bit is reset at startup to send 'factory' value to the OPAMPs. It is common to the 3 OPAMPs.

The offset of each operational amplifier can be trimmed by programming the *OPAMP offset trimming register for normal mode (OPAMP_OTR)*. The trimming values are stored in non-volatile memory.

The offset trimming register can be written, typically after a calibration operation initialized by the OPAx_CAL bits.

- Setting the OPAxCAL_L bit initializes offset calibration for the P differential pair (low voltage reference used).
- Setting the OPAxCAL_H bit initializes offset calibration for the N differential pair (high voltage reference used).

The 30 useful bits of OPA_OTR or OPA_LP_OTR are composed of three 10-bit words one for each operational amplifier. Each 10-bit is composed of 2 calibration values, the 5 lower bits are for trimming the offset of the PMOS differential pair. The 5 upper bits are for the NMOS ones.

After offset calibration is initialized by setting the control bit as shown in *Table 58*, write the new trimming values in the *OPAMP offset trimming register for normal mode (OPAMP_OTR)* register value until the OPAxCALOUT flag toggles to indicate that the calibration has successfully completed.

Table 58. Operating modes and calibration

Mode	Control bits				Output	
	OPAxPD	OPAxLPM	OPAxCAL_H	OPAxCAL_L	V_{OUT}	CALout flag
Normal operating mode	0	0	0	0	analog	0
			1	1		
Low power mode	0	1	0	0	analog	0
			1	1		
Power down	1	X	X	X	Z	0
Offset cal high	0	X	1	0	analog	X
Offset cal low	0	X	0	1	analog	X

Calibration procedure

Follow these steps to perform a full calibration of either one of the operational amplifiers:

1. Program the OPAMP_CSR register to open all the switches connected to the operational amplifier.
2. Set the OT_USER bit in the OPAMP_OTR register to 1.
3. Choose a calibration mode (refer to *Table 58: Operating modes and calibration*). You can begin with:

- Normal mode, offset cal high

  To do this, set OPAxPD=0, OPAxLPM=0, OPAxCAL_H=1, OPAxCAL_L=0 in the OPAM_CSR register.

4. The code in OPAMP_OTR[OPAxOPT_OFFSET_TRIM_High] is incremented from 00000b to the first value code that causes the OPAxCALOUT output level to change from 0 to 1.

Note: *Between the write to the OPAMP_OTR register and the read of the OPAxCALOUT value, take care to wait for the $t_{OFFTRIM}$max delay specified in the datasheet electrical characteristics section, to get the correct OPAxCALOUT value.*

The commutation means that the offset is correctly compensated and the corresponding trim code must kept in the OPAMP_OTR register.

The value 11111b is forbidden for OPAMP_OTR[OPAxOPT_OFFSET_TRIM_High].

Repeat steps 3 to 4 for:

- Normal_mode and offset cal low
- Low power mode and offset cal high
- Low power mode and offset cal low

If a mode is not used its calibration serves no purpose.

Note: *During the whole calibration phase the external connection of the operational amplifier output must not pull up or down currents higher than 500 µA.*

14.4 OPAMP registers

14.4.1 OPAMP control/status register (OPAMP_CSR)

Address offset: 0x00

Reset value: 0x0001 0101

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPA3CALOUT	OPA2CALOUT	OPA1CALOUT	OPA_RANGE	S7SEL2	ANAWSEL3	ANAWSEL2	ANAWSEL1	OPA3LPM	OPA3CAL_H	OPA3CAL_L	S6SEL3	S5SEL3	S4SEL3	S3SEL3	OPA3PD
r	r	r		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPA2LPM	OPA2CAL_H	OPA2CAL_L	S6SEL2	S5SEL2	S4SEL2	S3SEL2	OPA2PD	OPA1LPM	OPA1CAL_H	OPA1CAL_L	S6SEL1	S5SEL1	S4SEL1	S3SEL1	OPA1PD
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **OPA3CALOUT:** OPAMP3 calibration output

During calibration mode, the offset is trimmed when this signal toggles.

Bit 30 **OPA2CALOUT:** OPAMP2 calibration output

During calibration mode, the offset is trimmed when this signal toggles.

Bit 29 **OPA1CALOUT:** OPAMP1 calibration output

During calibration mode, the offset is trimmed when this signal toggles.

Bit 28 **OPA_RANGE:** Power range selection

This bit can be set and cleared by software when the operational amplifiers are in powered down. It select the operational amplifier power supply range for stability.

0: Low range ($V_{DDA} < 2.4$ V)

1: High range ($V_{DDA} > 2.4$ V)

Bit 27 **S7SEL2:** Switch 7 for OPAMP2 enable

0: S7 opened

1: S7 closed

Bit 26 **ANAWSEL3:** Switch SanA enable for OPAMP3

0: SanA switch opened

1: SanA switch closed

Bit 25 **ANAWSEL2:** Switch SanA enable for OPAMP2

0: SanA switch opened

1: SanA switch closed

Bit 24 **ANAWSEL1:** Switch SanA enable for OPAMP1

0: SanA switch opened

1: SanA switch closed

Bit 23 **OPA3LPM:** OPAMP3 low power mode

0: OPAMP3 low power mode off

1: OPAMP3 low power mode on

Bit 22 **OPA3CAL_H:** OPAMP3 offset calibration for N differential pair

0: OPAMP3 offset calibration for N diff OFF

1: OPAMP3 offset calibration for N diff ON if OPA3CAL_L = 0

Bit 21 **OPA3CAL_L:** OPAMP3 offset Calibration for P differential pair

0: OPAMP3 offset calibration for P diff OFF

1: OPAMP3 offset calibration for P diff ON if OPA3CAL_H = 0

Bit 20 **S6SEL3:** Switch 6 for OPAMP3 enable

0: S6 switch opened

1: S6 switch closed

Bit 19 **S5SEL3:** Switch 5 for OPAMP3 enable

0: S5 switch opened

1: S5 switch closed

Bit 18 **S4SEL3:** Switch 4 for OPAMP3 enable

0: S4 switch opened

1: S4 switch closed

Bit 17 **S3SEL3:** Switch 3 for OPAMP3 Enable

0: S3 switch opened

1: S3 switch closed

Bit 16 **OPA3PD:** OPAMP3 power down
0: OPAMP3 enabled
1: OPAMP3 disabled

Bit 15 **OPA2LPM:** OPAMP2 low power mode
0: OPAMP2 low power mode off
1: OPAMP2 low power mode on

Bit 14 **OPA2CAL_H:** OPAMP2 offset calibration for N differential pair
0: OPAMP2 offset calibration for N diff OFF
1: OPAMP2 offset calibration for N diff ON if OPA2CAL_L = 0

Bit 13 **OPA2CAL_L:** OPAMP2 offset Calibration for P differential pair
0: OPAMP2 offset calibration for P diff OFF
1: OPAMP2 offset calibration for P diff ON if OPA2CAL_H = 0

Bit 12 **S6SEL2:** Switch 6 for OPAMP2 enable
0: S6 switch opened
1: S6 switch closed

Bit 11 **S5SEL2:** Switch 5 for OPAMP2 enable
0: S5 switch opened
1: S5 switch closed

Bit 10 **S4SEL2:** Switch 4 for OPAMP2 enable
0: S4 switch opened
1: S4 switch closed

Bit 9 **S3SEL2:** Switch 3 for OPAMP2 enable
0: S3 switch opened
1: S3 switch closed

Bit 8 **OPA2PD:** OPAMP2 power down
0: OPAMP2 enabled
1: OPAMP2 disabled

Bit 7 **OPA1LPM:** OPAMP1 low power mode
0: OPAMP1 low power mode off
1: OPAMP1 in low power mode on

Bit 6 **OPA1CAL_H:** OPAMP1 offset calibration for N differential pair
0: OPAMP1 offset calibration for N diff OFF
1: OPAMP1 offset calibration for N diff ON if OPA1CAL_L = 0

Bit 5 **OPA1CAL_L:** OPAMP1 offset calibration for P differential pair
0: OPAMP1 offset calibration for P diff OFF
1: OPAMP1 offset calibration for P diff ON if OPA1CAL_H = 0

Bit 4 **S6SEL1:** Switch 6 for OPAMP1 enable
0: S6 switch opened
1: S6 switch closed

Bit 3 **S5SEL1:** Switch 5 for OPAMP1 enable
0: S5 switch opened
1: S5 switch closed

Bit 2 **S4SEL1:** Switch 4 for OPAMP1 enable

0: S4 switch opened

1: S4 switch closed

Bit 1 **S3SEL1:** Switch 3 for OPAMP1 enable

0: S3 switch opened

1: S3 switch closed

Bit 0 **OPA1PD:** OPAMP1 power down

0: OPAMP1 enabled

1: OPAMP1 disabled

14.4.2 OPAMP offset trimming register for normal mode (OPAMP_OTR)

Address offset: 0x04

Bit 31 reset value: 0

Bits 29:0 reset value: Factory trimmed value is restored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OT_ USER	Res.	OA3_OPT_OFFSET_TRIM_HIGH					OA3_OPT_OFFSET_TRIM_LOW					OA2_OPT_OFFSET_TRIM_ HIGH[4:1]			
w		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OA2_OPT_OFFSET_TRIM_HIGH0	OA2_OPT_OFFSET_TRIM_LOW					OA1_OPT_OFFSET_TRIM_HIGH					OA1_OPT_OFFSET_TRIM_LOW				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **OT_USER** Select user or factory trimming value

This bit is set and cleared by software, it is always read as 0. It is used to select if the OPAMPx offset is trimmed by the preset factory-programmed trimming values or the user programmed trimming value.

0: Trim the OPAMP offset using default factory values

1: Trim the OPAMP offset using user programmed values

Bit 30 Reserved, must be kept at reset value

Bit 29:25 **OA3_OPT_OFFSET_TRIM_HIGH[4:0]:** OPAMP3, normal mode 5-bit offset trim value for NMOS pairs

Bit 24:20 **OA3_OPT_OFFSET_TRIM_LOW[4:0]:** OPAMP3, normal mode 5-bit offset trim value for PMOS pairs

Bit 19:15 **OA2_OPT_OFFSET_TRIM_HIGH[4:0]:** OPAMP2, normal mode 5-bit offset trim value for NMOS pairs

Bit 14:10 **OA2_OPT_OFFSET_TRIM_LOW[4:0]:** OPAMP2, normal mode 5-bit offset trim value for PMOS pairs

Bit 9:5 **OA1_OPT_OFFSET_TRIM_HIGH[4:0]:** OPAMP1, normal mode 5-bit offset trim value for NMOS pairs

Bit 4:0 **OA1_OPT_OFFSET_TRIM_LOW[4:0]:** OPAMP1, normal mode 5-bit offset trim value for PMOS pairs

14.4.3 OPAMP offset trimming register for low power mode (OPAMP_LPOTR)

Address offset: 0x08

Bits 29:0 reset value: Factory trimmed value is restored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		OA3_OPT_OFFSET_TRIM_LP_HIGH					OA3_OPT_OFFSET_TRIM_LP_LOW					OA2_OPT_OFFSET_TRIM_LP_HIGH[4:1]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA2_OPT_OFFSET_TRIM_LP_HIGH0	OA2_OPT_OFFSET_TRIM_LP_LOW					OA1_OPT_OFFSET_TRIM_LP_HIGH					OA1_OPT_OFFSET_TRIM_LP_LOW				
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 29:25 **OA3_OPT_OFFSET_TRIM_LP_HIGH[4:0]:** OPAMP3, low power mode 5-bit offset trim value for NMOS pairs

Bit 24:20 **OA3_OPT_OFFSET_TRIM_LP_LOW[4:0]:** OPAMP3, low power mode 5-bit offset trim value for PMOS pairs

Bit 19:15 **OA2_OPT_OFFSET_TRIM_LP_HIGH[4:0]:** OPAMP2, low power mode 5-bit offset trim value for NMOS pairs

Bit 14:10 **OA2_OPT_OFFSET_TRIM_LP_LOW[4:0]:** OPAMP2, low power mode 5-bit offset trim value for PMOS pairs

Bit 9:5 **OA1_OPT_OFFSET_TRIM_LP_HIGH[4:0]:** OPAMP1, low power mode 5-bit offset trim value for NMOS pairs

Bit 4:0 **OA1_OPT_OFFSET_TRIM_LP_LOW[4:0]:** OPAMP1, low power mode 5-bit offset trim value for PMOS pairs

14.4.4 OPAMP register map

Table 59. OPAMP register map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**OPAMP_CSR**	OPA3CALOUT	OPA2CALOUT	OPA1CALOUT	OPA_RANGE	S7SEL2	ANAWSEL3	ANAWSEL2	ANAWSEL1	OPA3LPM	OPA3CAL_H	OPA3CAL_L	S6SEL3	S5SEL3	S4SEL3	S3SEL3	OPA3PD	OPA2LPM	OPA2CAL_H	OPA2CAL_L	S6SEL2	S5SEL2	S4SEL2	S3SEL2	OPA2PD	OPA1LPM	OPA1CAL_H	OPA1CAL_L	S6SEL1	S5SEL1	S4SEL1	S3SEL1	OPA1PD
0x04	**OPAMP_OTR**	OT_USER	Reserved	OA3_OPT_OFFSET_TRIM_HIGH					OA3_OPT_OFFSET_TRIM_LOW					OA2_OPT_OFFSET_TRIM_HIGH					OA2_OPT_OFFSET_TRIM_LOW					OA1_OPT_OFFSET_TRIM_HIGH					OA1_OPT_OFFSET_TRIM_LOW				
0x08	**OPAMP_LPOTR**	Reserved		OA3_OPT_OFFSET_TRIM_LP_HIGH					OA3_OPT_OFFSET_TRIM_LP_LOW					OA2_OPT_OFFSET_TRIM_LP_HIGH					OA2_OPT_OFFSET_TRIM_LP_LOW					OA1_OPT_OFFSET_TRIM_LP_HIGH					OA1_OPT_OFFSET_TRIM_LP_LOW				

Refer to *Table 2 on page 42* for the *Register boundary addresses* table.

15 Liquid crystal display controller (LCD)

15.1 Introduction

The LCD controller is a digital controller/driver for monochrome passive liquid crystal display (LCD) with up to 8 common terminals and up to 44 segment terminals to drive 176 (44x4) or 320 (40x8) LCD picture elements (pixels). The exact number of terminals depends on the device pinout as described in the datasheet.

The LCD is made up of several segments (pixels or complete symbols) which can be turned visible or invisible. Each segment consists of a layer of liquid crystal molecules aligned between two electrodes. When a voltage greater than a threshold voltage is applied across the liquid crystal, the segment becomes visible. The segment voltage must be alternated to avoid an electrophoresis effect in the liquid crystal (which degrades the display). The waveform across a segment must then be generated so as to avoid having a direct current (DC).

15.2 LCD main features

- Highly flexible frame rate control.
- Supports Static, 1/2, 1/3, 1/4 and 1/8 duty.
- Supports Static, 1/2, 1/3 and 1/4 bias.
- Double buffered memory allows data in LCD_RAM registers to be updated at any time by the application firmware without affecting the integrity of the data displayed.
  - LCD data RAM of up to 16 x 32-bit registers which contain pixel information (active/inactive)
- Software selectable LCD output voltage (contrast) from V_{LCDmin} to V_{LCDmax}.
- No need for external analog components:
  - A step-up converter is embedded to generate an internal V_{LCD} voltage higher than V_{DD}
  - Software selection between external and internal V_{LCD} voltage source. In case of an external source, the internal boost circuit is disabled to reduce power consumption
  - A resistive network is embedded to generate intermediate V_{LCD} voltages ($V_{LCDrail1}$, $V_{LCDrail2}$, $V_{LCDrail3}$)
  - The structure of the resistive network is configurable by software to adapt the power consumption to match the capacitive charge required by the LCD panel.
- The contrast can be adjusted using two different methods:
  - When using the internal step-up converter, the software can adjust V_{LCD} between V_{LCDmin} and V_{LCDmax}.
  - Programmable dead time (up to 8 phase periods) between frames.
- Full support of Low power modes: the LCD controller can be displayed in Sleep, Low power run, Low power sleep and STOP modes or can be fully disabled to reduce power consumption
- Built in phase inversion for reduced power consumption and EMI. (electromagnetic interference)
- Start of frame interrupt to synchronize the software when updating the LCD data RAM.
- Blink capability:
  - Up to 1, 2, 3, 4, 8 or all pixels which can be programmed to blink at a configurable frequency.
  - Software adjustable blink frequency to achieve around 0.5 Hz, 1 Hz, 2 Hz or 4 Hz.
- Used LCD segment and common pins should be configured as GPIO alternate functions and unused segment and common pins can be used for general purpose I/O or for another peripheral alternate function.
- V_{LCD} rails ($V_{LCDrail1}$, $V_{LCDrail2}$, $V_{LCDrail3}$) decoupling capability

Note: When the LCD relies on the internal step-up converter, the V_{LCD} pin should be connected to V_{SS} with a capacitor. Its typical value is 1 µF (see C_{EXT} value in the product datasheets for further information).

The V_{LCD} pin should be connected to V_{DDA}:
- For devices without LCD
- If the LCD peripheral is not used for devices with LCD.

15.3 Glossary

Bias: Number of voltage levels used when driving an LCD. It is defined as 1/(number of voltage levels used to drive an LCD display - 1).

Boost circuit: Contrast controller circuit

Common: Electrical connection terminal connected to several segments (44 segments).

Duty ratio: Number defined as 1/(number of common terminals on a given LCD display).

Frame: One period of the waveform written to a segment.

Frame rate: Number of frames per second, that is the number of times the LCD segments are energized per second.

LCD: (liquid crystal display) a passive display panel with terminals leading directly to a segment.

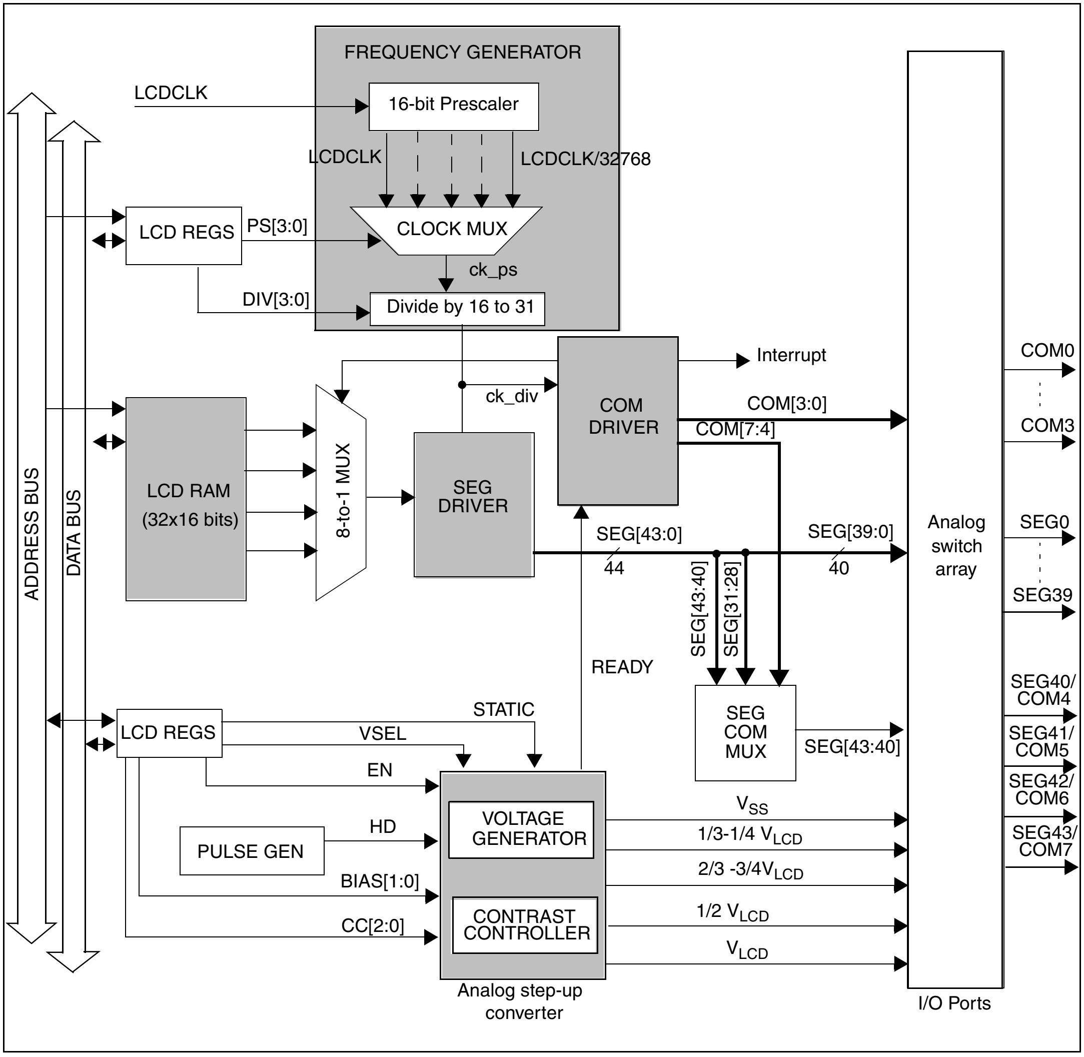
Segment: The smallest viewing element (a single bar or dot that is used to help create a character on an LCD display).

15.4 LCD functional description

15.4.1 General description

The LCD controller has five main blocks (see *Figure 70*):

Figure 70. LCD controller block diagram



Note: LCDCLK is the same as RTCCLK. Please refer to the RTC/LCD clock description in the RCC section of this manual.

15.4.2 Frequency generator

The frequency generator allows you to achieve various LCD frame rates starting from an LCD input clock frequency (LCDCLK) which can vary from 32 kHz up to 1 MHz.

3 different clock sources can be used to provide the LCD clock (LCDCLK/RTCCLK):

- 32 kHz Low speed external RC (LSE)
- 37 kHz Low speed internal RC (LSI)
- 1-24 MHz High speed external crystal oscillator (HSE) divided by 2, 4, 8 or 16 to obtain a 1 MHz clock

Please refer to the RTC/LCD Clock configuration in the RCC section of this manual.

This clock source must be stable in order to obtain accurate LCD timing and hence minimize DC voltage offset across LCD segments. The input clock LCDCLK can be divided by any value from 1 to 2^{15}x 31 (see *Section 15.5.2: LCD frame control register (LCD_FCR) on page 333*). The frequency generator consists of a prescaler (16-bit ripple counter) and a 16 to 31 clock divider. The PS[3:0] bits, in the LCD_FCR register, select LCDCLK divided by $2^{PS[3:0]}$. If a finer resolution rate is required, the DIV[3:0] bits, in the LCD_FCR register, can be used to divide the clock further by 16 to 31. In this way you can roughly scale the frequency, and then fine-tune it by linearly scaling the clock with the counter. *The output of the frequency generator block is f_{ck_div} which constitutes the time base for the entire LCD controller. The ck_div frequency is equivalent to the LCD phase frequency,* rather than the frame frequency (they are equal only in case of static duty). The frame frequency (f_{frame}) is obtained from f_{ck_div} by dividing it by the number of active common terminals (or by multiplying it for the duty). Thus the relation between the input clock frequency (f_{LCDCLK}) of the frequency generator and its output clock frequency f_{ck_div} is:

$$f_{ckdiv} = \frac{f_{LCDCLK}}{2^{PS} \times \langle 16 + DIV \rangle}$$

$$f_{frame} = f_{ckdiv} \times duty$$

This makes the frequency generator very flexible. An example of frame rate calculation is shown in *Table 60*.

Table 60. Example of frame rate calculation

LCDCLK	PS[3:0]	DIV[3:0]	Ratio	Duty	f_{frame}
32.768 kHz	3	1	136	1/8	30.12 Hz
32.768 kHz	4	1	272	1/4	30.12 Hz
32.768 kHz	4	6	352	1/3	31.03 Hz
32.768 kHz	5	1	544	1/2	30.12 Hz
32.768 kHz	6	1	1088	static	30.12 Hz
32.768 kHz	1	4	40	1/8	102.40 Hz
32.768 kHz	2	4	80	1/4	102.40 Hz
32.768 kHz	2	11	108	1/3	101.14 Hz
32.768 kHz	3	4	160	1/2	102.40 Hz
32.768 kHz	4	4	320	static	102.40 Hz

Table 60. Example of frame rate calculation (continued)

LCDCLK	PS[3:0]	DIV[3:0]	Ratio	Duty	f_{frame}
1.00 MHz	6	3	1216	1/8	102.80 Hz
1.00 MHz	7	3	2432	1/4	102.80 Hz
1.00 MHz	7	10	3328	1/3	100.16 Hz
1.00 MHz	8	3	4864	1/2	102.80 Hz
1.00 MHz	9	3	9728	static	102.80 Hz

The frame frequency must be selected to be within a range of around ~30 Hz to ~100 Hz and is a compromise between power consumption and the acceptable refresh rate. In addition, a dedicated blink prescaler selects the blink frequency. This frequency is defined as:

$f_{BLINK} = f_{ck\_div}/2^{(BLINKF + 3)}$,

with BLINKF[2:0] = 0, 1, 2, ..,7

The blink frequency achieved is in the range of 0.5 Hz, 1 Hz, 2 Hz or 4 Hz.

15.4.3 Common driver

Common signals are generated by the common driver block (see *Figure 70*).

COM signal bias

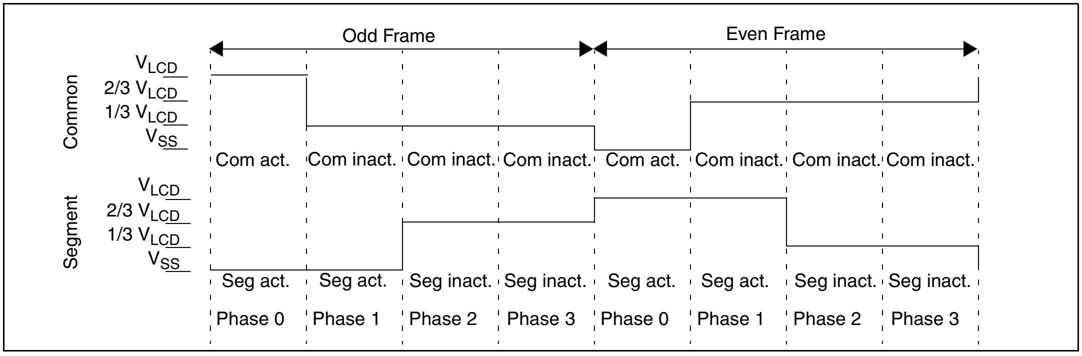
Each COM signal has identical waveforms, but different phases. It has its max amplitude V_{LCD} or V_{SS} only in the corresponding phase of a frame cycle, while during the other phases, the signal amplitude is:

- 1/4 V_{LCD} or 3/4 V_{LCD} in case of 1/4 bias
- 1/3 V_{LCD} or 2/3 V_{LCD} in case of 1/3 bias
- and 1/2 V_{LCD} in case of 1/2 bias.

Selection between 1/2, 1/3 and 1/4 bias mode can be done through the BIAS bits in the LCD_CR register.

A pixel is activated when both of its corresponding common and segment lines have max amplitudes during the same phase. Common signals are phase inverted in order to reduce EMI. As shown in *Figure 71*, with phase inversion, there is a mean voltage of 1/2 V_{LCD} at the end of every odd cycle.

Figure 71. 1/3 bias, 1/4 duty



In case of 1/2 bias (BIAS = 01) the V_{LCD} pin generates an intermediate voltage ($V_{LCDrail1}$ = $V_{LCDrail3}$) equal to 1/2 V_{LCD} for odd and even frames (see *Figure 78*).

COM signal duty

Depending on the DUTY[2:0] bits in the LCD_CR register, the COM signals are generated with static duty (see *Figure 73*), 1/2 duty (see *Figure 74*), 1/3 duty (see *Figure 75*), 1/4 duty (see *Figure 76*) or 1/8 duty (see *Figure 77*).

COM[*n*] *n*[0 to 7] is active during phase *n* in the odd frame, so the COM pin is driven to V_{LCD},

During phase *n* of the even frame the COM pin is driven to V_{SS}.

In the case of 1/3 or 1/4) bias:

- COM[*n*] is inactive during phases other than n so the COM pin is driven to 1/3 (1/4) V_{LCD} during odd frames and to 2/3 (3/4) V_{LCD} during even frames

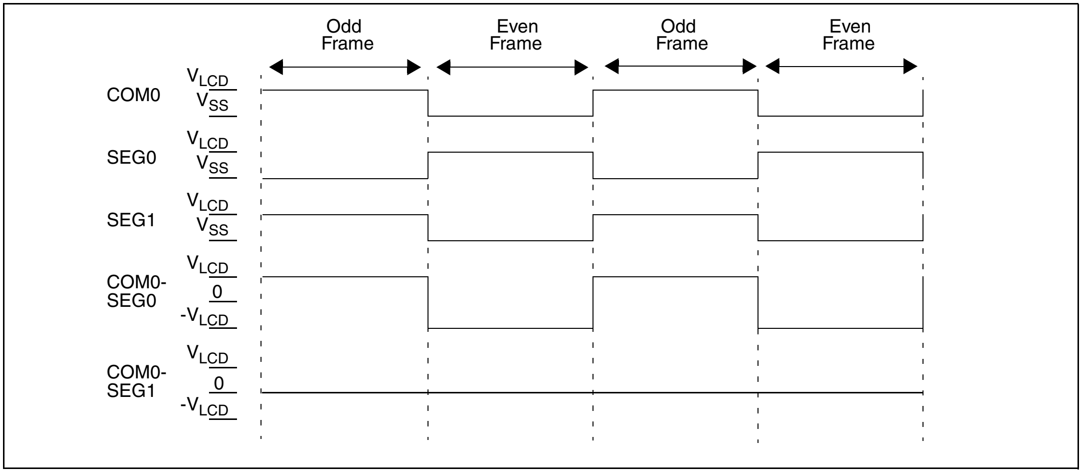
In the case of 1/2 bias:

- If COM[*n*] is inactive during phases other than n, the COM pin is always driven (odd and even frame) to 1/2 V_{LCD}.

When static duty is selected, the segment lines are not multiplexed, which means that each segment output corresponds to one pixel. In this way only up to 44 pixels can be driven. COM[0] is always active while COM[7:1] are not used and are driven to V_{SS}.

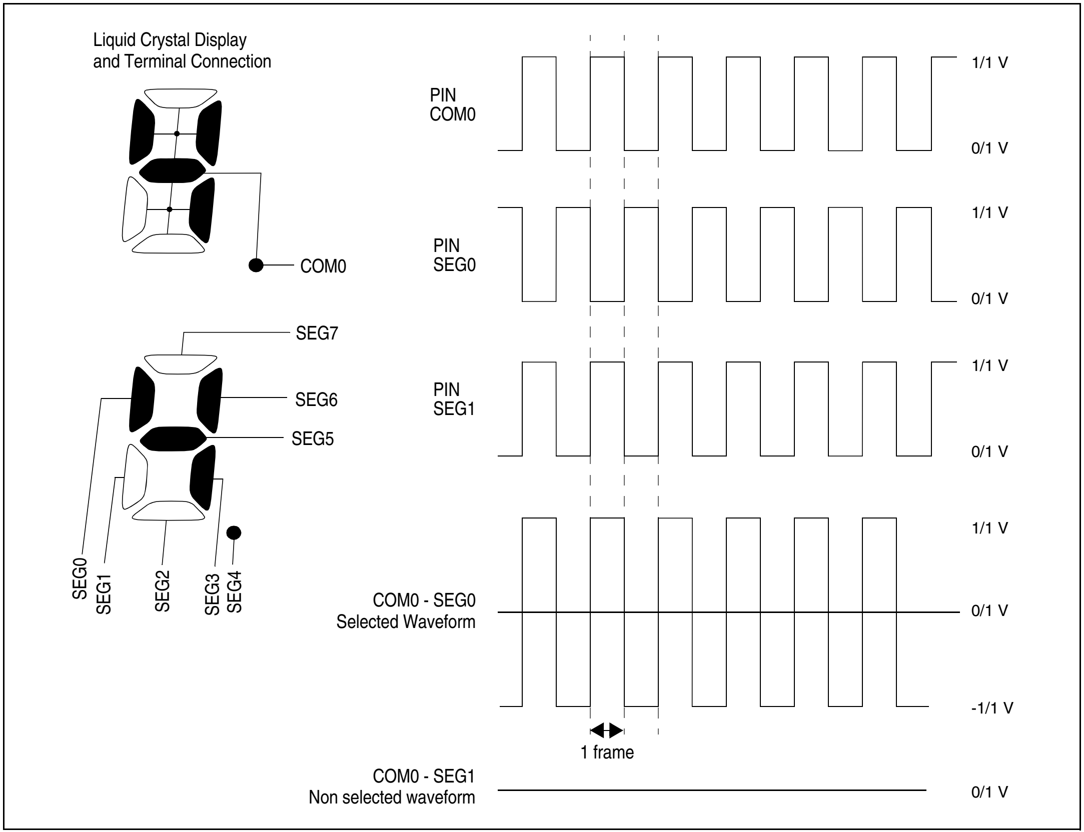
When the LCDEN bit in the LCD_CR register is reset, all common lines are pulled down to V_{SS} and the ENS flag in the LCD_SR register becomes 0. Static duty means that COM[0] is always active and only two voltage levels are used for the segment and common lines: V_{LCD} and V_{SS}. A pixel is active if the corresponding SEG line has a voltage opposite to that of the COM, and inactive when the voltages are equal. In this way the LCD has maximum contrast (see *Figure 72*, *Figure 73*). In the *Figure 72* pixel 0 is active while pixel 1 is inactive.

Figure 72. Static duty



In each frame there is only one phase, this is why f_{frame} is equal to f_{LCD}. If 1/4 duty is selected there are four phases in a frame in which COM[0] is active during phase 0, COM[1] is active during phase 1, COM[2] is active during phase 2, and COM[3] is active during phase 3.

Figure 73. Static duty



In this mode, the segment terminals are multiplexed and each of them control four pixels. A pixel is activated only when both of its corresponding SEG and COM lines are active in the same phase. In case of 1/4 duty, to deactivate pixel 0 connected to COM[0] the SEG[0] needs to be inactive during the phase 0 when COM[0] is active. To activate pixel44 connected to COM[1] the SEG[0] needs to be active during phase 1 when COM[1] is active (see *Figure 76*). To activate pixels from 0 to 43 connected to COM[0], SEG[0:43] need to be active during phase 0 when COM[0] is active. These considerations can be extended to the other pixels.

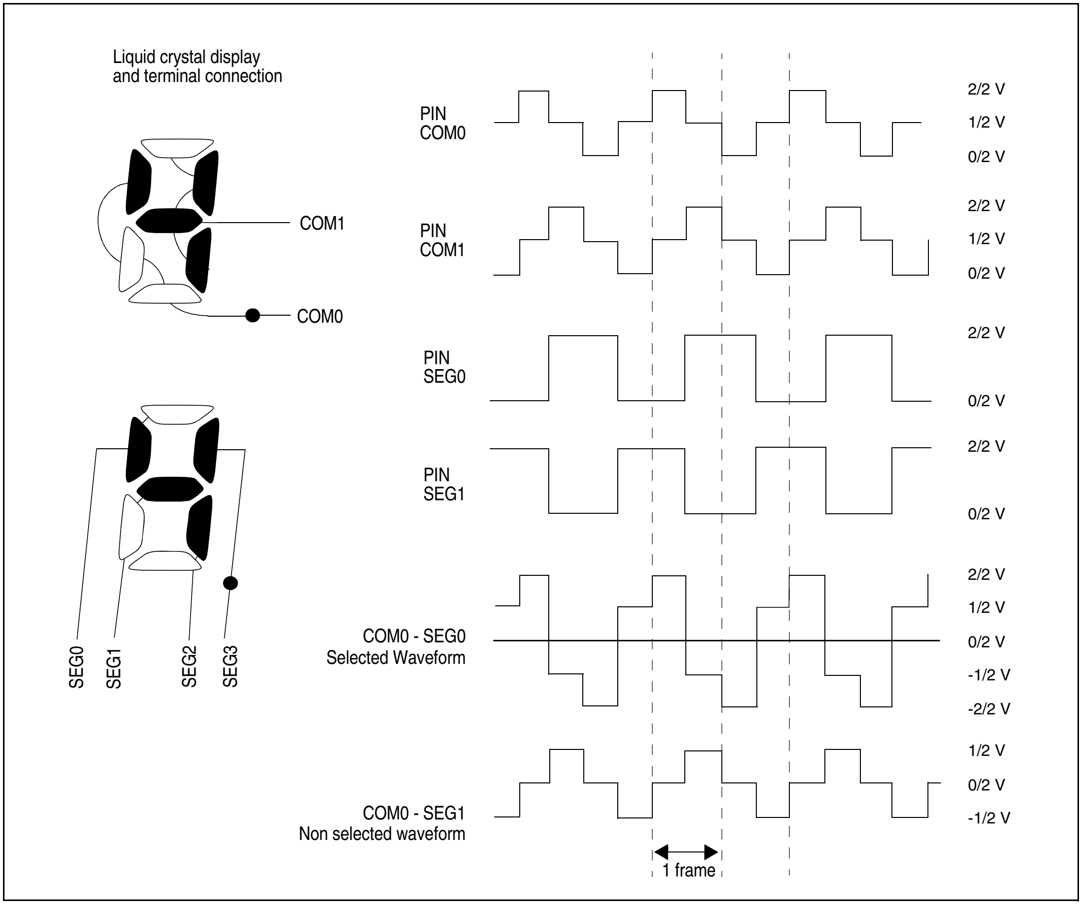
8 to 1 Mux

When COM[0] is active the common driver block, also drives the 8 to 1 mux shown in *Figure 70* in order to select the content of first two RAM register locations. When *COM[7]* is active, the output of the 8 to 1 mux is the content of the last two RAM locations.

Start of frame (SOF)

The common driver block is also able to generate an SOF (start of frame flag) (see *Section 15.5.3: LCD status register (LCD_SR)*). The LCD start of frame interrupt is executed if the SOFIE (start of frame interrupt enable) bit is set (see *Section 15.5.2: LCD frame control register (LCD_FCR)*). SOF is cleared by writing the SOFC bit to 1 in the LCD_CLR register when executing the corresponding interrupt handling vector.

Figure 74. 1/2 duty, 1/2 bias



15.4.4 Segment driver

The segment driver block controls the SEG lines according to the pixel data coming from the 8 to 1 mux driven in each phase by the common driver block.

In the case of 1/4 or 1/8 duty

When COM[0] is active, the pixel information (active/inactive) related to the pixel connected to COM[0] (content of the first two LCD_RAM locations) goes through the 8 to 1 mux.

The SEG[*n*] pin *n* [0 to 43] is driven to V_{SS} (indicating pixel *n* is active when COM[0] is active) in phase 0 of the odd frame.

The SEG[*n*] pin is driven to V_{LCD} in phase 0 of the even frame. If pixel *n* is inactive then the SEG[*n*] pin is driven to 2/3 (2/4) V_{LCD} in the odd frame or 1/3 (2/4) V_{LCD} in the even frame (current inversion in V_{LCD} pad) (see *Figure 71*).

In case of 1/2 bias, if the pixel is inactive the SEG[*n*] pin is driven to V_{LCD} in the odd and to V_{SS} in the even frame.

When the LCD controller is disabled (LCDEN bit cleared in the LCD_CR register) then the SEG lines are pulled down to V_{SS}.

Figure 75. 1/3 duty, 1/3 bias

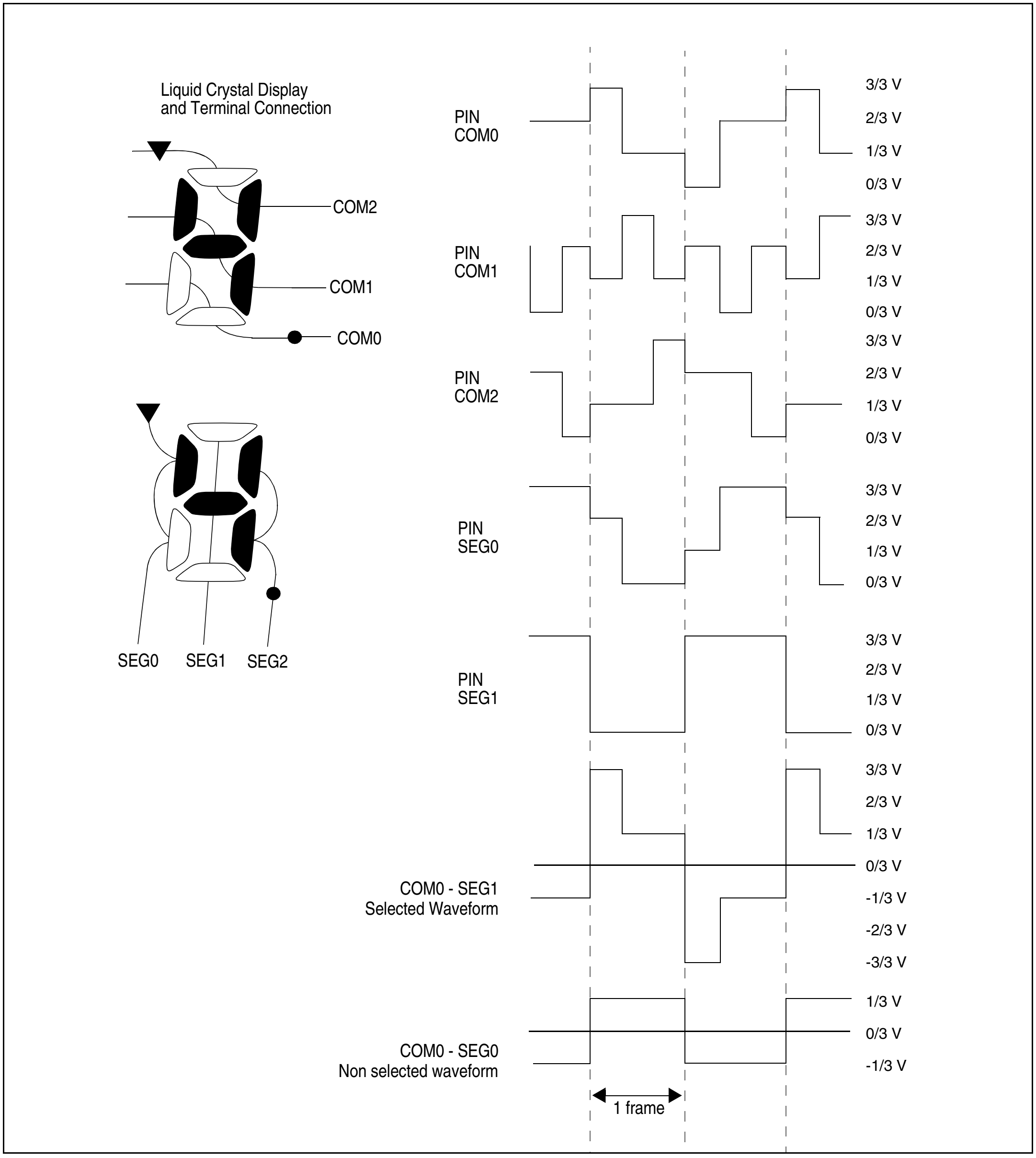


Figure 76. 1/4 duty, 1/3 bias

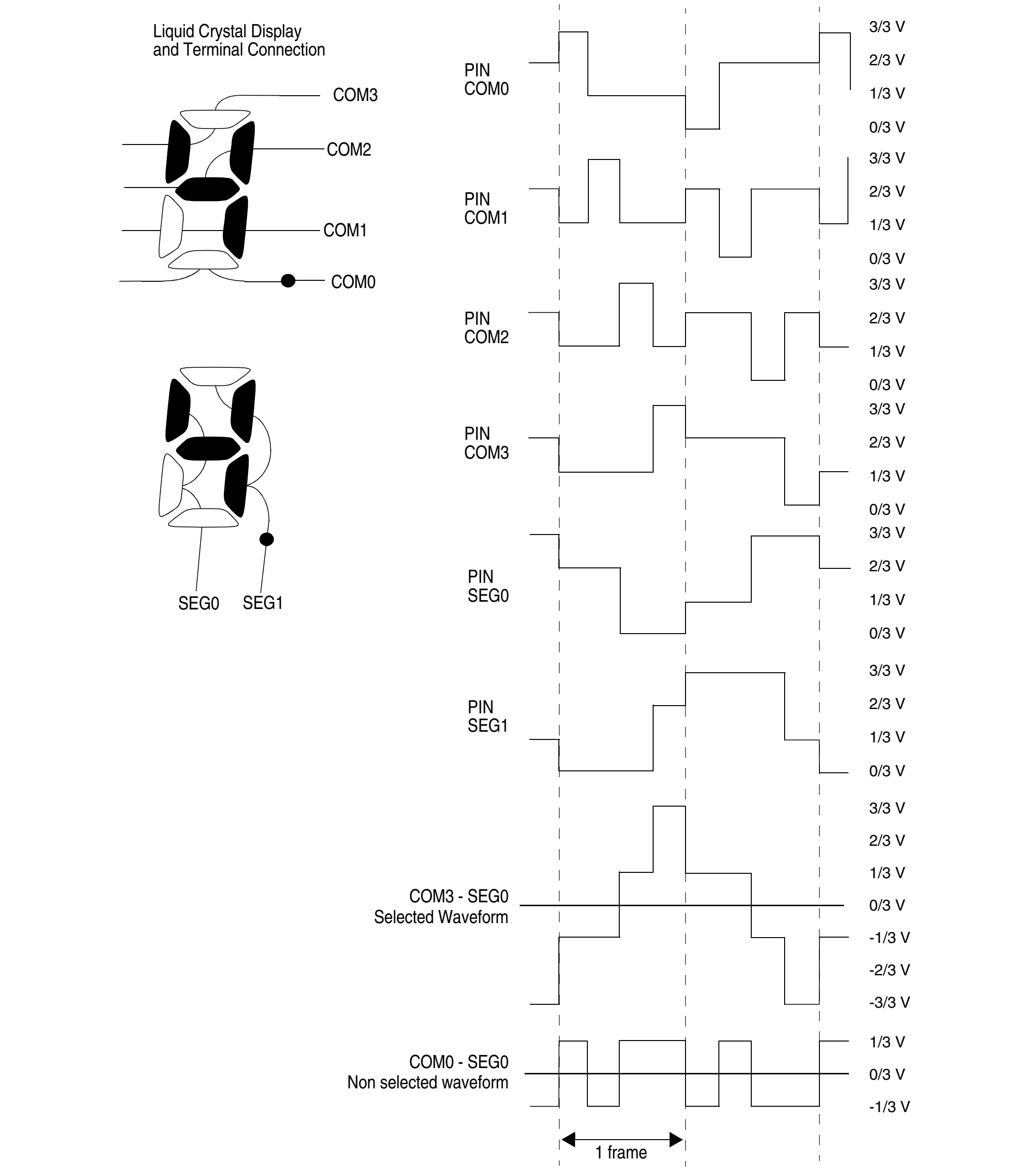
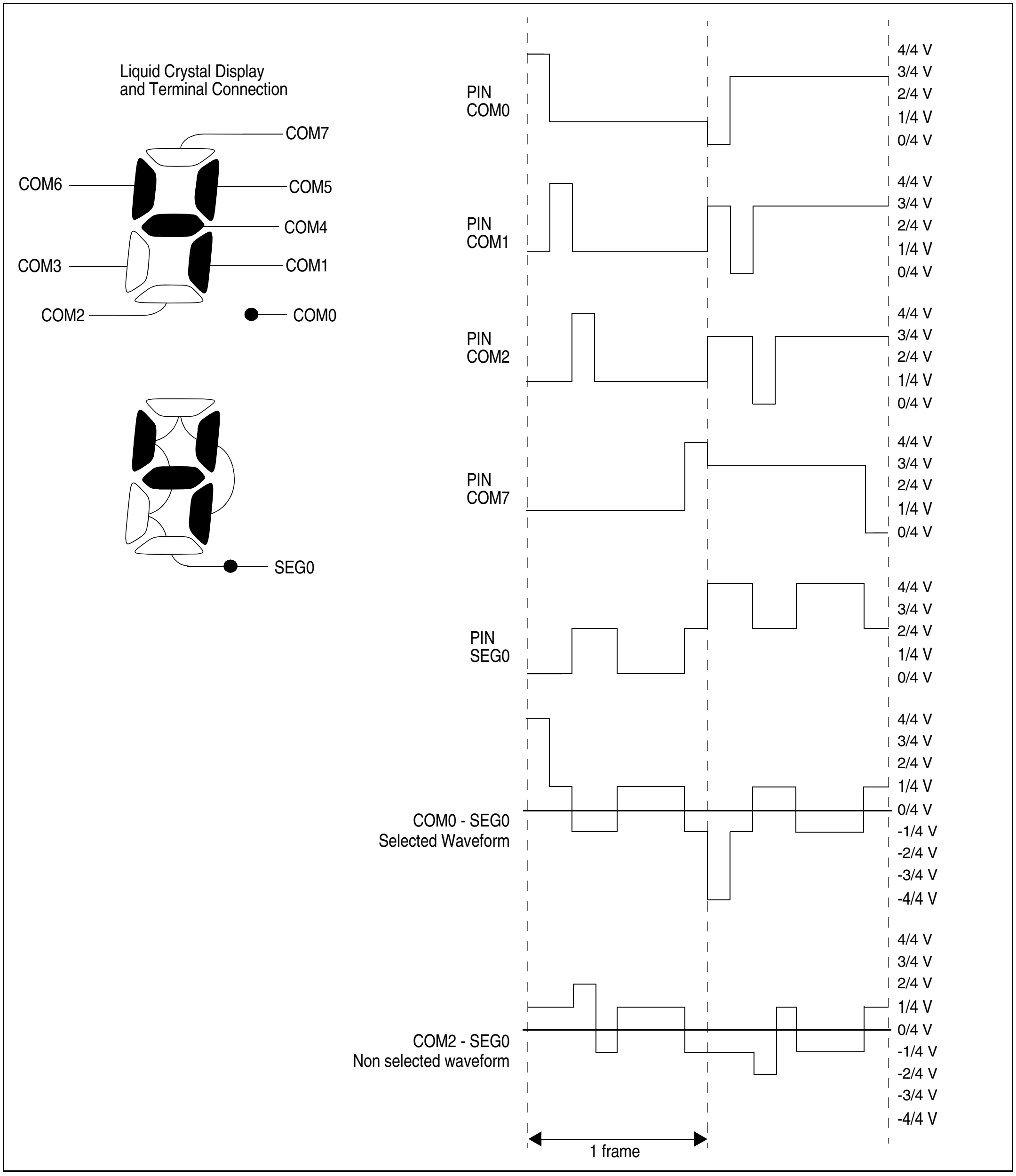


Figure 77. 1/8 duty, 1/4 bias



Blink

The segment driver also implements a programmable blink feature to allow some pixels to continuously switch on at a specific frequency. The blink mode can be configured by the BLINK[1:0] bits in the LCD_FCR register, making possible to blink up to 1, 2, 4, 8 or all pixels (see *Section 15.5.2: LCD frame control register (LCD_FCR)*). The blink frequency can be selected from eight different values using the BLINKF[2:0] bits in the LCD_FCR register.

Table 61 gives examples of different blink frequencies (as a function of ck_div frequency).

Table 61. Blink frequency

BLINKF[2:0] bits			ck_div frequency (with LCDCLK frequency of 32.768 kHz)			
			32 Hz	64 Hz	128 Hz	256 Hz
0	0	0	4.0 Hz	N/A	N/A	N/A
0	0	1	2.0 Hz	4.0 Hz	N/A	N/A
0	1	0	1.0 Hz	2.0 Hz	4.0 Hz	N/A
0	1	1	0.5 Hz	1.0 Hz	2.0 Hz	4.0 Hz
1	0	0	0.25 Hz	0.5 Hz	1.0 Hz	2.0 Hz
1	0	1	N/A	0.25 Hz	0.5 Hz	1.0 Hz
1	1	0	N/A	N/A	0.25 Hz	0.5 Hz
1	1	1	N/A	N/A	N/A	0.25 Hz

15.4.5 Voltage generator

The LCD voltage levels are generated by the V_{LCD} pin or by the internal voltage step-up converter (depending on the VSEL bit in the LCD_CR register), through an internal resistor divider network as shown in *Figure 78*.

The LCD voltage generator generates up to three intermediate voltage levels (1/3 V_{LCD}, 2/3 V_{LCD} or 1/4 V_{LCD}, 2/4 V_{LCD}, 3/4 V_{LCD}) between V_{SS} and V_{LCD} in case of 1/3 (1/4) bias and only one voltage level (1/2 V_{LCD}) between V_{SS} and V_{LCD} in case of 1/2 bias.

In the case of 1/3 or 1/4 bias:

- **During odd frames**, $V_{LCDrail1}$ voltage (V_{bCOM}) is 1/3 (1/4) V_{LCD}, while $V_{LCDrail3}$ voltage (V_{aSEG}) is 2/3 (3/4) V_{LCD},
- **During even frames**, $V_{LCDrail1}$ voltage is 2/3 (3/4) V_{LCD} and $V_{LCDrail3}$ voltage is 1/3 (1/4) V_{LCD}.

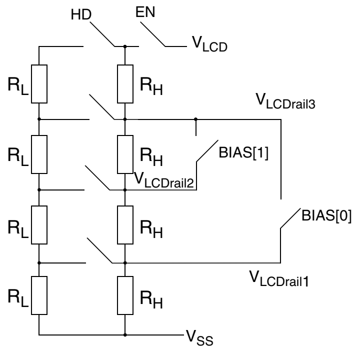
In the case of 1/2 bias:

- $V_{LCDrail3}$ voltage is equal to $V_{LCDrail1}$ voltage and its value is 1/2 V_{LCD}.

For the divider network, two resistive networks one with low value resistors (R_L) and one with high value resistors (R_H) are respectively used to increase the current during transitions and to reduce power consumption in static state.

The PON[2:0] (Pulse ON duration) bits in the LCD_FCR register configure the time during which R_L is enabled (see *Figure 70*) when the levels of the common and segment lines change. A short drive time will lead to lower power consumption, but displays with high internal resistance may need a longer drive time to achieve satisfactory contrast.

Figure 78. LCD voltage control



The R_L divider can be always switched on using the HD bit in the LCD_FCR configuration register (see *Section 15.5.2*). The V_{LCD} value can be chosen among a wide set of values from V_{LCDmin} to V_{LCDmax} by means of CC[2:0] (Contrast Control) bits inside LCD_FCR (see *Section 15.5.2*) register. New values of V_{LCD} takes effect every beginning of a new frame.

After the LCDEN bit is activated the voltage generator sets the RDY bit in the LCD_SR register to indicate that the voltage levels are stable and the LCD controller can start to work.

External decoupling

Devices with V_{LCD} rails decoupling capability (see devices' datasheet) offer the possibility to add decoupling capacitors on V_{LCD} intermediate voltage rails ($V_{LCDrail1}$, $V_{LCDrail2}$, $V_{LCDrail3}$ - see *Figure 78*) for stabilization purpose. Spikes may be observed when voltage applied to the pixel is alternated. In this case, these decoupling capacitors will help to get a steady voltage resulting in a higher contrast. This capability is particulary useful for consumption reason as it allow to select low PON[2:0] values in the LCD_FCR register.

To connect the Vlcd rails as described in *Table 62* to the dedicated GPIOs, configure the LCD_CAPA[4:0] bits of the SYSCFG_PMC register.

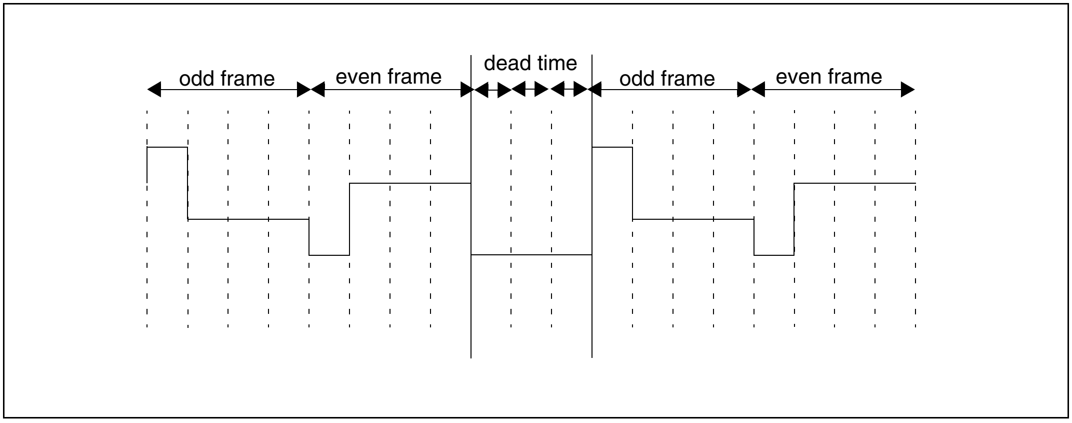
Table 62. $V_{LCDrail}$ connections to GPIO pins

	Bias			Pin (selection by LCD_CAPA[4:0] bits)	
	1/2	1/3	1/4		
$V_{LCDrail2}$	1/2 Vlcd	2/3 Vlcd	1/2 Vlcd	PB2	
$V_{LCDrail1}$	1/2 Vlcd	1/3 Vlcd	1/4 Vlcd	PB12	PE11
$V_{LCDrail3}$	1/2 Vlcd	2/3 Vlcd	3/4 Vlcd	PB0	PE12

15.4.6 Deadtime

In addition to using the CC[2:0] bits, the contrast can be controlled by programming a dead time between each frame. During the dead time the COM and SEG values are put to V_{SS}. The DEAD[2:0] bits in the LCD_FCR register can be used to program a time of up to eight phase periods. This dead time reduces the contrast without modifying the frame rate.

Figure 79. Deadtime



15.4.7 Double buffer memory

Using its double buffer memory the LCD controller ensures the coherency of the displayed information without having to use interrupts to control LCD_RAM modification.

The application software can access the first buffer level (LCD_RAM) through the APB interface. Once it has modified the LCD_RAM, it sets the UDR flag in the LCD_SR register. This UDR flag (update display request) requests the updated information to be moved into the second buffer level (LCD_DISPLAY).

This operation is done synchronously with the frame (at the beginning of the next frame), until the update is completed, the LCD_RAM is write protected and the UDR flag stays high. Once the update is completed another flag (UDD - Update Display Done) is set and generates an interrupt if the UDDIE bit in the LCD_FCR register is set.

The time it takes to update LCD_DISPLAY is, in the worst case, one odd and one even frame.

The update will not occur (UDR = 1 and UDD = 0) until the display is enabled (LCDEN = 1)

15.4.8 COM and SEG multiplexing

Output pins versus duty modes

The output pins consists of:

- SEG[43:0]
- COM[3:0]

Depending on the duty configuration, the COM and SEG output pins may have different functions:

- In static, 1/2, 1/3 and 1/4 duty modes there are up to 44 SEG pins and respectively 1, 2, 3 and 4 COM pins
- In 1/8 duty mode (DUTY[2:0] = 100), the COM[7:4] outputs are available on the SEG[43:40] pins, reducing to the number of available segments *40* .

Remapping capability

Additionally, it is possible to remap 4 segments by setting the MUX_SEG bit in the LCD_CR register. This is particularly useful when using smaller device types with fewer external pins.

When MUX_SEG is set, output pins SEG[43:40] have function SEG[31:28].

Summary of COM and SEG functions versus duty and remap

All the possible ways of multiplexing the COM and SEG functions are described in *Table 63*. *Figure 80* gives examples showing the signal connections to the external pins.

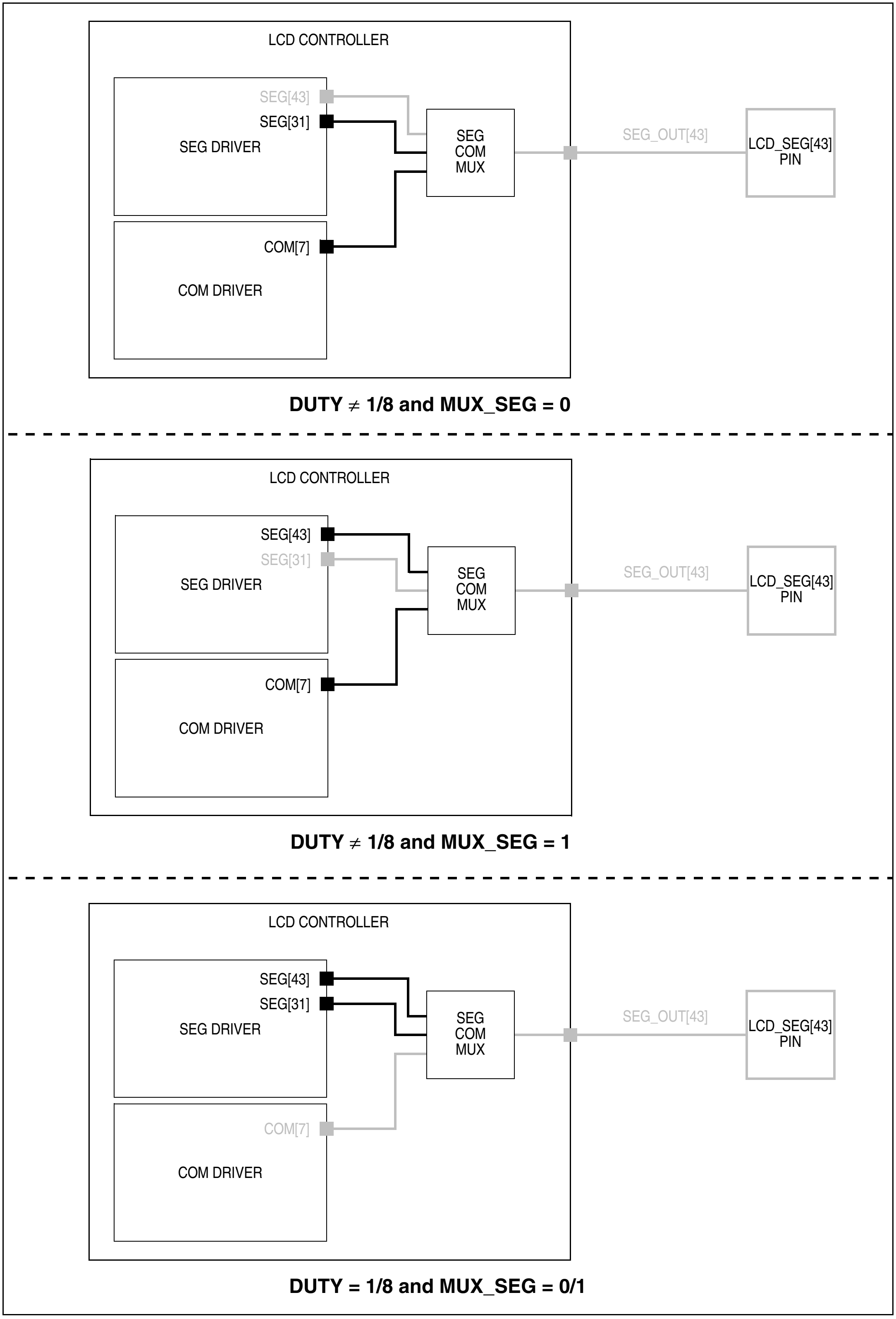
Table 63. Remapping capability

Configuration bits		Capability	Output pin	Function
DUTY	MUX_SEG			
1/8	0	40x8	SEG[43:40]	COM[7:4]
			COM[3:0]	COM[3:0]
			SEG[39:0]	SEG[39:0]
	1	28x8	SEG[43:40]	COM[7:4]
			COM[3:0]	COM[3:0]
			SEG[39:28]	not used
			SEG[27:0]	SEG[27:0]
1/4	0	44x4	COM[3:0]	COM[3:0]
			SEG[43:0]	SEG[43:0]
	1	32x4	COM[3:0]	COM[3:0]
			SEG[43:40]	SEG[31:28]
			SEG[39:28]	not used
			SEG[27:0]	SEG[27:0]
1/3	0	44x3	COM[3]	not used
			COM[2:0]	COM[2:0]
			SEG[43:0]	SEG[43:0]
	1	32x3	COM[3]	not used
			COM[2:0]	COM[2:0]
			SEG[43:40]	SEG[31:28]
			SEG[39:28]	not used
			SEG[27:0]	SEG[27:0]
1/2	0	44x2	COM[3:2]	not used
			COM[1:0]	COM[1:0]
			SEG[43:0]	SEG[43:0]
	1	32x2	COM[3:2]	not used
			COM[1:0]	COM[1:0]
			SEG[43:40]	SEG[31:28]
			SEG[39:28]	not used
			SEG[27:0]	SEG[27:0]

Table 63. Remapping capability (continued)

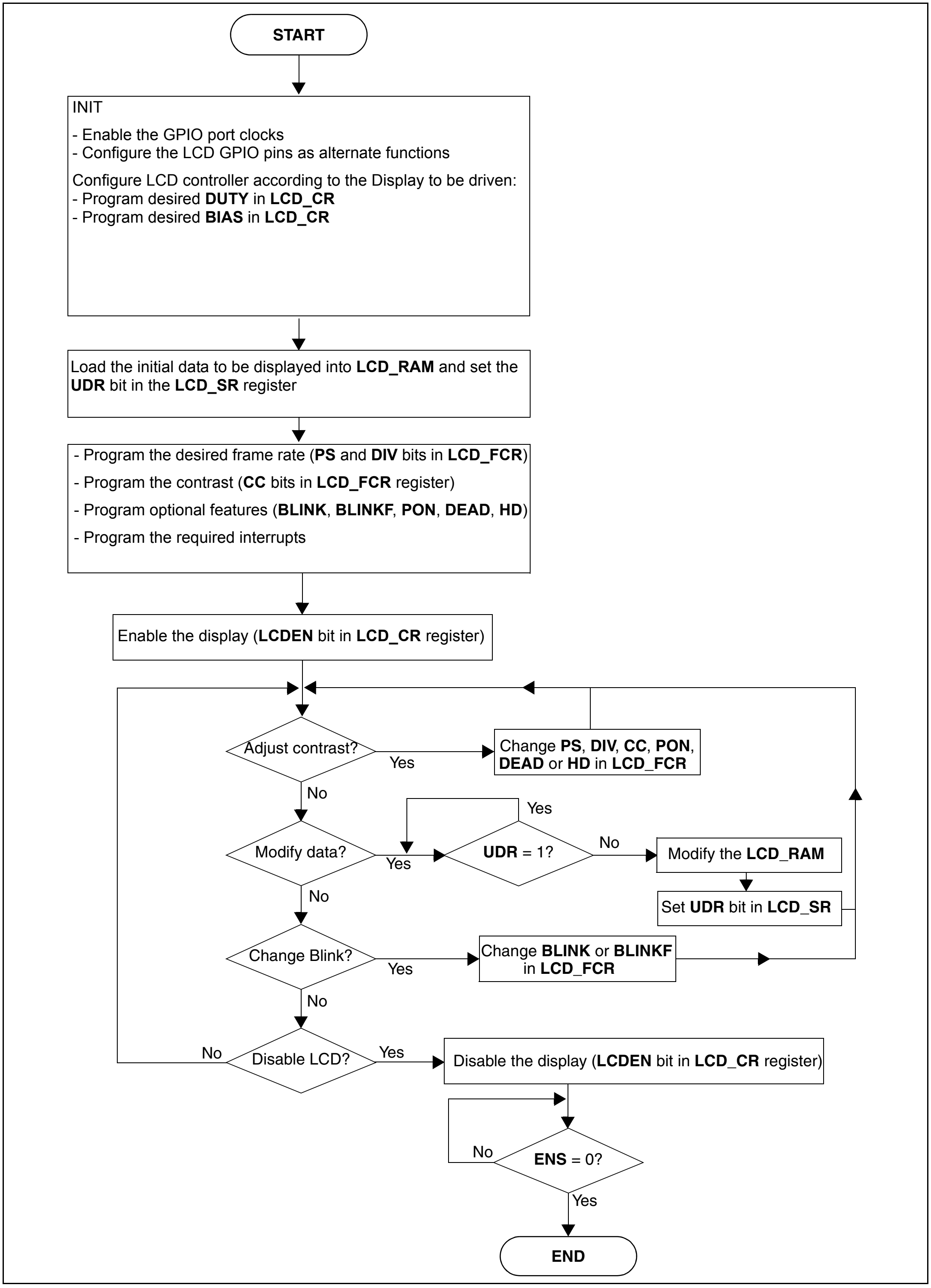
Configuration bits		Capability	Output pin	Function
DUTY	MUX_SEG			
STATIC	0	44x1	COM[3:1]	not used
			COM[0]	COM[0]
			SEG[43:0]	SEG[43:0]
	1	32x1	COM[3:1]	not used
			COM[0]	COM[0]
			SEG[43:40]	SEG[31:28]
			SEG[39:28]	not used
			SEG[27:0]	SEG[27:0]

Figure 80. SEG/COM mux feature example



15.4.9 Flowchart

Figure 81. Flowchart example


START
INIT
- Enable the GPIO port clocks
- Configure the LCD GPIO pins as alternate functions
Configure LCD controller according to the Display to be driven:
- Program desired **DUTY** in **LCD_CR**
- Program desired **BIAS** in **LCD_CR**
Load the initial data to be displayed into **LCD_RAM** and set the **UDR** bit in the **LCD_SR** register
- Program the desired frame rate (**PS** and **DIV** bits in **LCD_FCR**)
- Program the contrast (**CC** bits in **LCD_FCR** register)
- Program optional features (**BLINK**, **BLINKF**, **PON**, **DEAD**, **HD**)
- Program the required interrupts
Enable the display (**LCDEN** bit in **LCD_CR** register)
Adjust contrast?
Yes
Change **PS**, **DIV**, **CC**, **PON**, **DEAD** or **HD** in **LCD_FCR**
No
Modify data?
Yes
UDR = 1?
Yes
No
Modify the **LCD_RAM**
Set **UDR** bit in **LCD_SR**
No
Change Blink?
Yes
Change **BLINK** or **BLINKF** in **LCD_FCR**
No
Disable LCD?
No
Yes
Disable the display (**LCDEN** bit in **LCD_CR** register)
ENS = 0?
No
Yes
END

15.5 LCD registers

The peripheral registers have to be accessed by words (32-bit).

15.5.1 LCD control register (LCD_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								MUX_SEG	BIAS[1:0]		DUTY[2:0]			VSEL	LCDEN
								rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw

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

Bit 7 **MUX_SEG**: Mux segment enable

This bit is used to enable SEG pin remapping. Four SEG pins can be multiplexed with SEG[31:28]. See *Section 15.4.8*.

0: SEG pin multiplexing disabled
1: SEG[31:28] are multiplexed with SEG[43:40]

Bits 6:5 **BIAS[1:0]**: Bias selector

These bits determine the bias used. Value 11 is forbidden.

00: Bias 1/4
01: Bias 1/2
10: Bias 1/3
11: Reserved

Bits 4:2 **DUTY[2:0]**: Duty selection

These bits determine the duty cycle. Values 101, 110 and 111 are forbidden.

000: Static duty
001: 1/2 duty
010: 1/3 duty
011: 1/4 duty
100: 1/8 duty
101: Reserved
110: Reserved
111: Reserved

Bit 1 **VSEL**: Voltage source selection

The VSEL bit determines the voltage source for the LCD.

0: Internal source (voltage step-up converter)
1: External source (V_{LCD} pin)

Bit 0 **LCDEN**: LCD controller enable

This bit is set by software to enable the LCD Controller/Driver. It is cleared by software to turn off the LCD at the beginning of the next frame. When the LCD is disabled all COM and SEG pins are driven to V_{SS}.

0: LCD Controller disabled

1: LCD Controller enabled

Note: The VSEL, MUX_SEG, BIAS and DUTY bits are write protected when the LCD is enabled (ENS bit in LCD_SR to 1).

15.5.2 LCD frame control register (LCD_FCR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						PS[3:0]				DIV[3:0]				BLINK[1:0]	
						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
BLINKF[2:0]			CC[2:0]			DEAD[2:0]			PON[2:0]			UDDIE	Res.	SOFIE	HD
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bits 31:26 Reserved, must be kept at reset value

Bits 25:22 **PS[3:0]**: PS 16-bit prescaler

These bits are written by software to define the division factor of the PS 16-bit prescaler. ck_ps = LCDCLK/(2). See *Section 15.4.2*.

0000: ck_ps = LCDCLK

0001: ck_ps = LCDCLK/2

0011: ck_ps = LCDCLK/4

...

1111: ck_ps = LCDCLK/32768

Bits 21:18 **DIV[3:0]**: DIV clock divider

These bits are written by software to define the division factor of the DIV divider. See *Section 15.4.2*.

0000: ck_div = ck_ps/16

0001: ck_div = ck_ps/17

0011: ck_div = ck_ps/18

...

1111: ck_div = ck_ps/31

Bits 17:16 **BLINK[1:0]**: Blink mode selection

00: Blink disabled

01: Blink enabled on SEG[0], COM[0] (1 pixel)

10: Blink enabled on SEG[0], all COMs (up to 8 pixels depending on the programmed duty)

11: Blink enabled on all SEGs and all COMs (all pixels)

Bits 15:13 **BLINKF[2:0]**: Blink frequency selection

000: f_{LCD}/8
001: f_{LCD}/16
010: f_{LCD}/32
011: f_{LCD}/64
100: f_{LCD}/128
101: f_{LCD}/256
110: f_{LCD}/512
111: f_{LCD}/1024

Bits 12:10 **CC[2:0]**: Contrast control

These bits specify one of the V_{LCD} maximum voltages (independent of V_{DD}). It ranges from 2.60 V to 3.51V.

000: V_{LCD0}
001: V_{LCD1}
010: V_{LCD2}
011: V_{LCD3}
100: V_{LCD4}
101: V_{LCD5}
110 V_{LCD6}
111: V_{LCD7}

Note: Refer to the product datasheet for the V_{LCDx} values.

Bits 9:7 **DEAD[2:0]**: Dead time duration

These bits are written by software to configure the length of the dead time between frames. During the dead time the COM and SEG voltage levels are held at 0 V to reduce the contrast without modifying the frame rate.

000: No dead time
001: 1 phase period dead time
010: 2 phase period dead time
......
111: 7 phase period dead time

Bits 6:4 **PON[2:0]**: Pulse ON duration

These bits are written by software to define the pulse duration in terms of ck_ps pulses. A short pulse will lead to lower power consumption, but displays with high internal resistance may need a longer pulse to achieve satisfactory contrast. Note that the pulse will never be longer than one half prescaled LCD clock period.

000: 0
001: 1/ck_ps
010: 2/ck_ps
011: 3/ck_ps
100: 4/ck_ps
101: 5/ck_ps
110: 6/ck_ps
111: 7/ck_ps

PON duration example with LCDCLK = 32.768 kHz and PS=0x03:

000: 0 µs
001: 244 µs
010: 488 µs
011: 782 µs
100: 976 µs
101: 1.22 ms
110: 1.46 ms
111: 1.71 ms

Bit 3 **UDDIE**: Update display done interrupt enable

This bit is set and cleared by software.

0: LCD Update Display Done interrupt disabled
1: LCD Update Display Done interrupt enabled

Bit 2 Reserved, must be kept at reset value

Bit 1 **SOFIE**: Start of frame interrupt enable

This bit is set and cleared by software.

0: LCD Start of Frame interrupt disabled

1: LCD Start of Frame interrupt enabled

Bit 0 **HD**: High drive enable

This bit is written by software to enable a low resistance divider. Displays with high internal resistance may need a longer drive time to achieve satisfactory contrast. This bit is useful in this case if some additional power consumption can be tolerated.

0: Permanent high drive disabled

1: Permanent high drive enabled. When HD=1, then the PON bits have to be programmed to 001.

Note: The data in this register can be updated any time, however the new values are applied only at the beginning of the next frame (except for CC, UDDIE, SOFIE that affect the device behavior immediately).

Reading this register obtains the last value written in the register and not the configuration used to display the current frame.

15.5.3 LCD status register (LCD_SR)

Address offset: 0x08

Reset value: 0x0000 0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										FCRSF	RDY	UDD	UDR	SOF	ENS
										r	r	r	rs	r	r

Bits 31:6 Reserved, must be kept at reset value

Bit 5 **FCRSF**: LCD Frame Control Register Synchronization flag

This bit is set by hardware each time the LCD_FCR register is updated in the LCDCLK domain. It is cleared by hardware when writing to the LCD_FCR register.

0: LCD Frame Control Register not yet synchronized

1: LCD Frame Control Register synchronized

Bit 4 **RDY**: Ready flag

This bit is set and cleared by hardware. It indicates the status of the step-up converter.

0: Not ready

1: Step-up converter is enabled and ready to provide the correct voltage.

Bit 3 **UDD**: Update Display Done

This bit is set by hardware. It is cleared by writing 1 to the UDDC bit in the LCD_CLR register. The bit set has priority over the clear.

0: No event

1: Update Display Request done. A UDD interrupt is generated if the UDDIE bit in the LCD_FCR register is set.

Note: If the device is in STOP mode (PCLK not provided) UDD will not generate an interrupt even if UDDIE = 1.

If the display is not enabled the UDD interrupt will never occur.

Bit 2 **UDR**: Update display request

Each time software modifies the LCD_RAM it must set the UDR bit to transfer the updated data to the second level buffer. The UDR bit stays set until the end of the update and during this time the LCD_RAM is write protected.

0: No effect

1: Update Display request

Note: When the display is disabled, the update is performed for all LCD_DISPLAY locations. When the display is enabled, the update is performed only for locations for which commons are active (depending on DUTY). For example if DUTY = 1/2, only the LCD_DISPLAY of COM0 and COM1 will be updated.

Note: Writing 0 on this bit or writing 1 when it is already 1 has no effect. This bit can be cleared by hardware only. It can be cleared only when LCDEN = 1

Bit 1 **SOF**: Start of frame flag

This bit is set by hardware at the beginning of a new frame, at the same time as the display data is updated. It is cleared by writing a 1 to the SOFC bit in the LCD_CLR register. The bit clear has priority over the set.

0: No event

1: Start of Frame event occurred. An LCD Start of Frame Interrupt is generated if the SOFIE bit is set.

Bit 0 **ENS**: LCD enabled status

This bit is set and cleared by hardware. It indicates the LCD controller status.

0: LCD Controller disabled.

1: LCD Controller enabled

Note: The ENS bit is set immediately when the LCDEN bit in the LCD_CR goes from 0 to 1. On deactivation it reflects the real status of LCD so it becomes 0 at the end of the last displayed frame.

15.5.4 LCD clear register (LCD_CLR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												UDDC	Res.	SOFC	Res.
												w		w	

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

Bit 3 **UDDC:** Update display done clear

This bit is written by software to clear the UDD flag in the LCD_SR register.

0: No effect

1: Clear UDD flag

Bit 2 Reserved, must be kept at reset value

Bit 1 **SOFC:** Start of frame flag clear

This bit is written by software to clear the SOF flag in the LCD_SR register.

0: No effect

1: Clear SOF flag

Bit 0 Reserved, must be kept at reset value

15.5.5 LCD display memory (LCD_RAM)

Address offset: 0x14-0x50

Reset value: 0x0000 0000

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

Bits 31:0 **SEGMENT_DATA[31:0]**

Each bit corresponds to one pixel of the LCD display.

0: Pixel inactive

1: Pixel active

15.5.6 LCD register map

The following table summarizes the LCD registers.

Table 64. LCD register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**LCD_CR**	Reserved																								MUX_SEG	BIAS[1:0]		DUTY[2:0]			VSEL	LCDEN
	Reset value																									0	0	0	0	0	0	0	0

Table 64. LCD register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x04	**LCD_FCR**	Reserved						PS[3:0]				DIV[3:0]				BLINK[1:0]		BLINKF[2:0]			CC[2:0]			DEAD [2:0]			PON[2:0]			UDDIE	Reserved	SOFIE	HD
	Reset value							0	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	**LCD_SR**	Reserved																										FCRSF	RDY	UDD	UDDR	SOF	ENS
	Reset value																											1	0	0	0	0	0
0x0C	**LCD_CLR**	Reserved																												UDDC	Reserved	SOFC	Reserved
	Reset value																													0		0	
0x14	**LCD_RAM (COM0)**	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16	S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18		Reserved																				S43	S42	S41	S40	S39	S38	S37	S36	S35	S34	S33	S32
																						0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**LCD_RAM (COM1)**	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16	S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
		0	0	0	0	0	0	0	0	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		Reserved																				S43	S42	S41	S40	S39	S38	S37	S36	S35	S34	S33	S32
																						0	0	0	0	0	0	0	0	0	0	0	0
0x24	**LCD_RAM (COM2)**	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16	S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28		Reserved																				S43	S42	S41	S40	S39	S38	S37	S36	S35	S34	S33	S32
																						0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**LCD_RAM (COM3)**	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16	S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30		Reserved																				S43	S42	S41	S40	S39	S38	S37	S36	S35	S34	S33	S32
																						0	0	0	0	0	0	0	0	0	0	0	0

Table 64. LCD 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
0x34	LCD_RAM (COM4)	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16	S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38		Reserved																								S39	S38	S37	S36	S35	S34	S33	S32
																										0	0	0	0	0	0	0	0
0x3C	LCD_RAM (COM5)	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16	S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
		0	0	0	0	0	0	0	0	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		Reserved																								S39	S38	S37	S36	S35	S34	S33	S32
																										0	0	0	0	0	0	0	0
0x44	LCD_RAM (COM6)	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16	S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48		Reserved																								S39	S38	S37	S36	S35	S34	S33	S32
																										0	0	0	0	0	0	0	0
0x4C	LCD_RAM (COM7)	S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16	S15	S14	S13	S12	S11	S10	S09	S08	S07	S06	S05	S04	S03	S02	S01	S00
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50		Reserved																								S39	S38	S37	S36	S35	S34	S33	S32
																										0	0	0	0	0	0	0	0

Refer to *Table 2 on page 42* for the *Register boundary addresses* table.

16 General-purpose timers (TIM2 to TIM5)

16.1 TIM2 to TIM5 introduction

The general-purpose timers consist of a 16-bit or 32-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

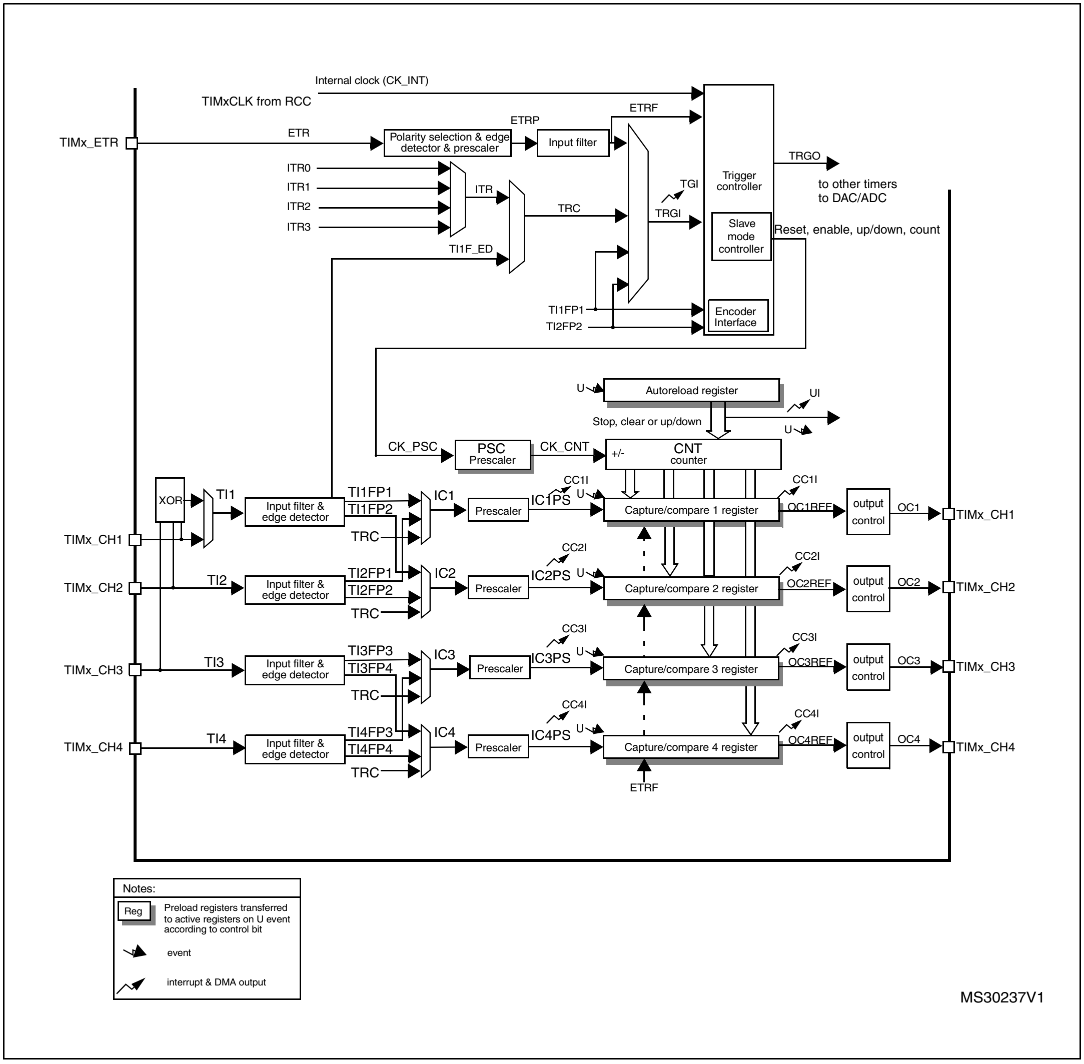
The timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 16.3.15*.

16.2 TIM2 to TIM5 main features

General-purpose TIMx timer features include:

- 16-bit (TIM3 and TIM4) or 32-bit (TIM5)up, down, up/down auto-reload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65536.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge- and Center-aligned modes)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 82. General-purpose timer block diagram



16.3 TIM2 to TIM5 functional description

16.3.1 Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related auto-reload register. The counter can count up but also down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC):
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 83 and *Figure 84* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 83. Counter timing diagram with prescaler division change from 1 to 2

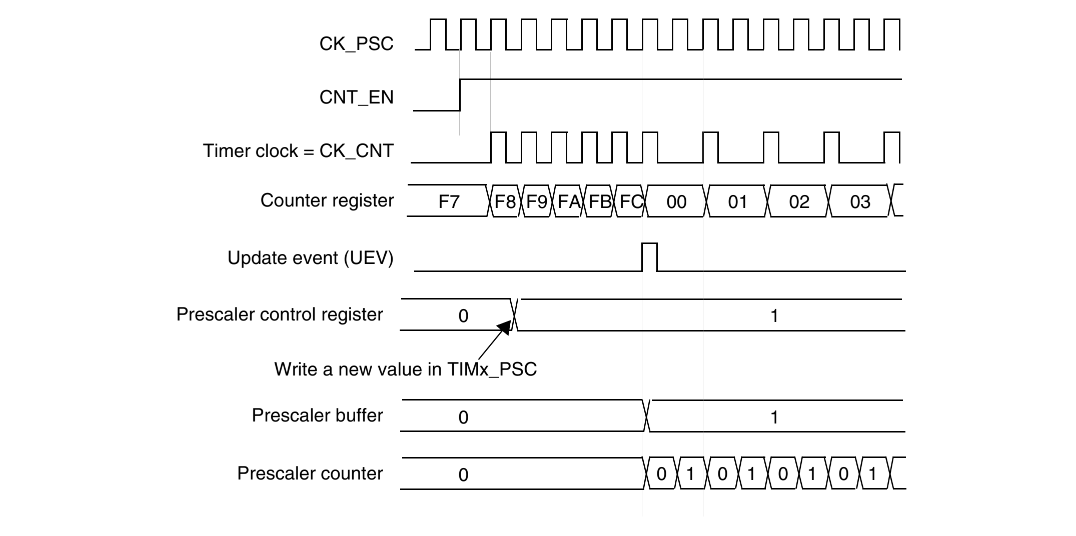
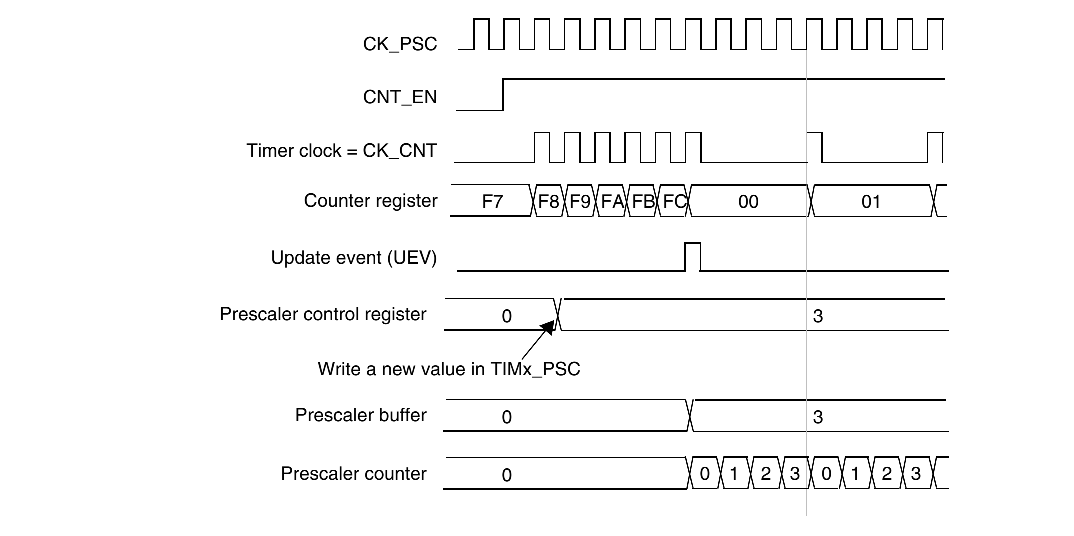


Figure 84. Counter timing diagram with prescaler division change from 1 to 4



16.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An Update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 85. Counter timing diagram, internal clock divided by 1

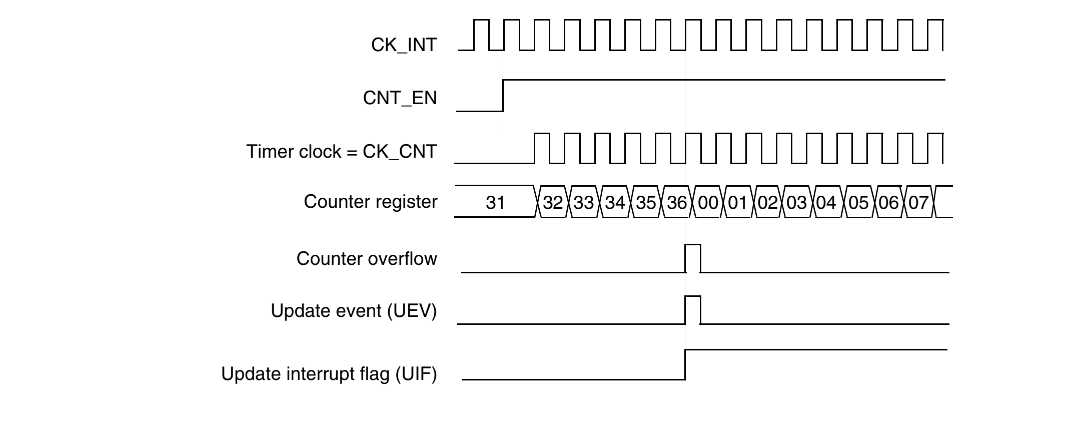


Figure 86. Counter timing diagram, internal clock divided by 2

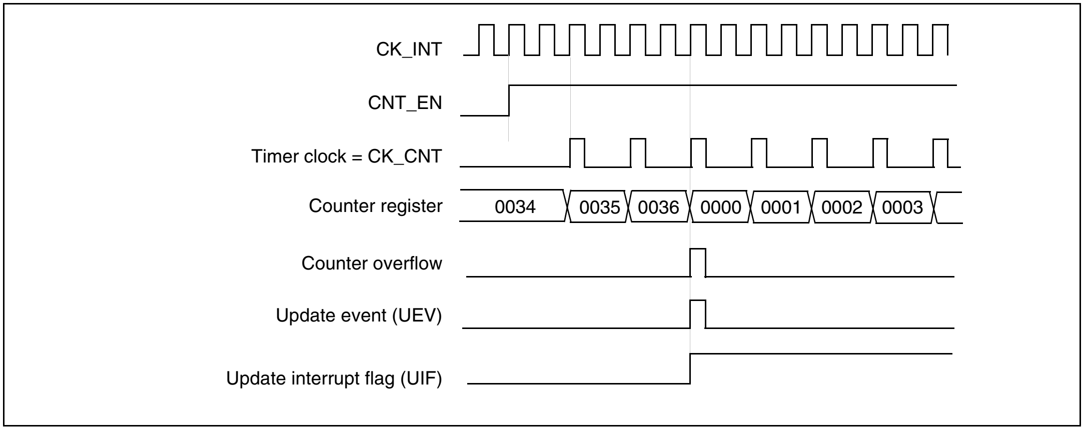


Figure 87. Counter timing diagram, internal clock divided by 4

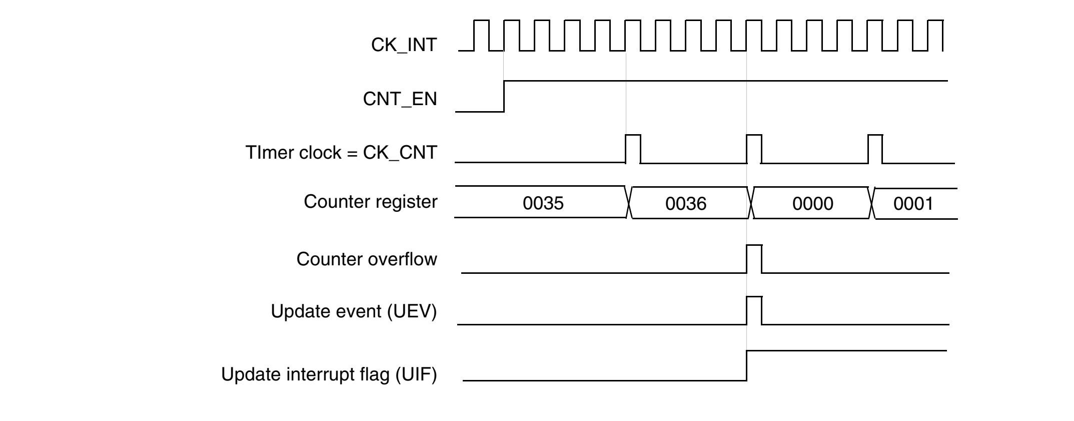


Figure 88. Counter timing diagram, internal clock divided by N

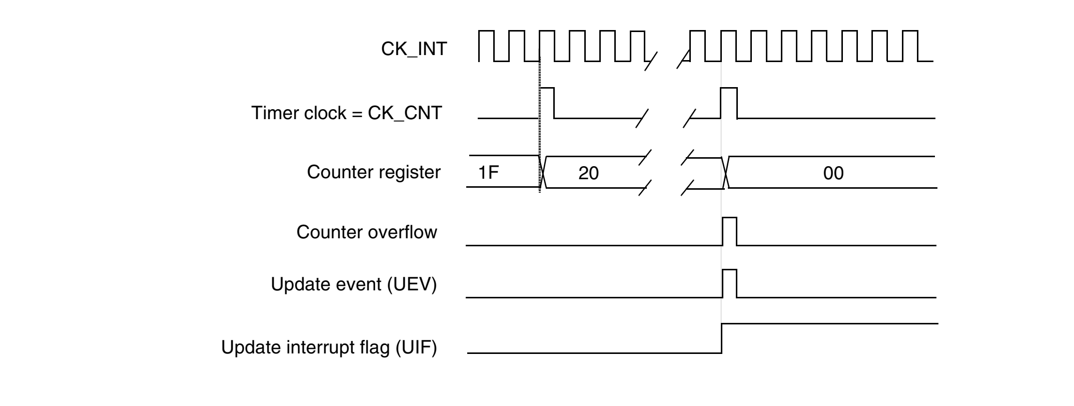


Figure 89. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)

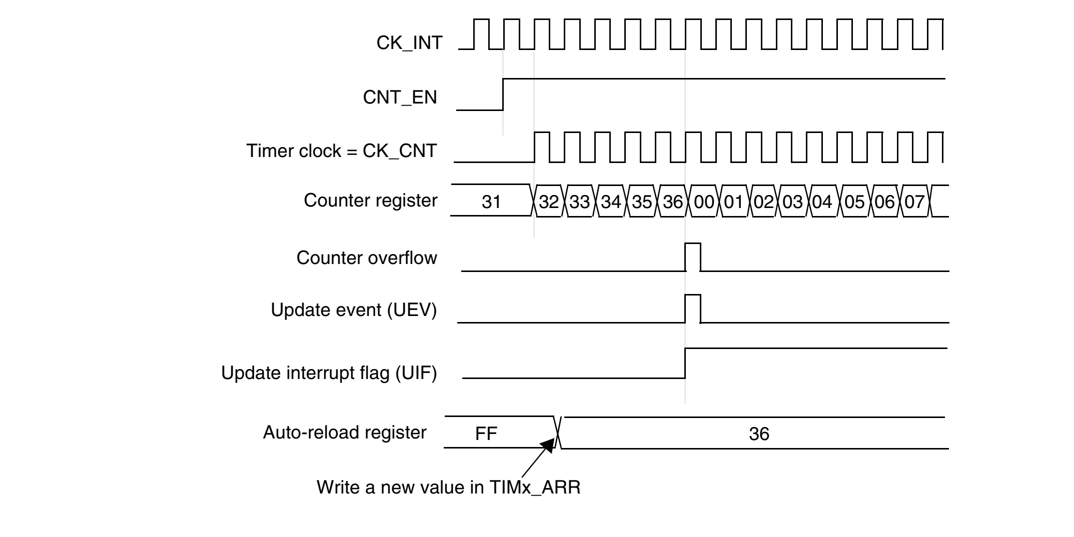
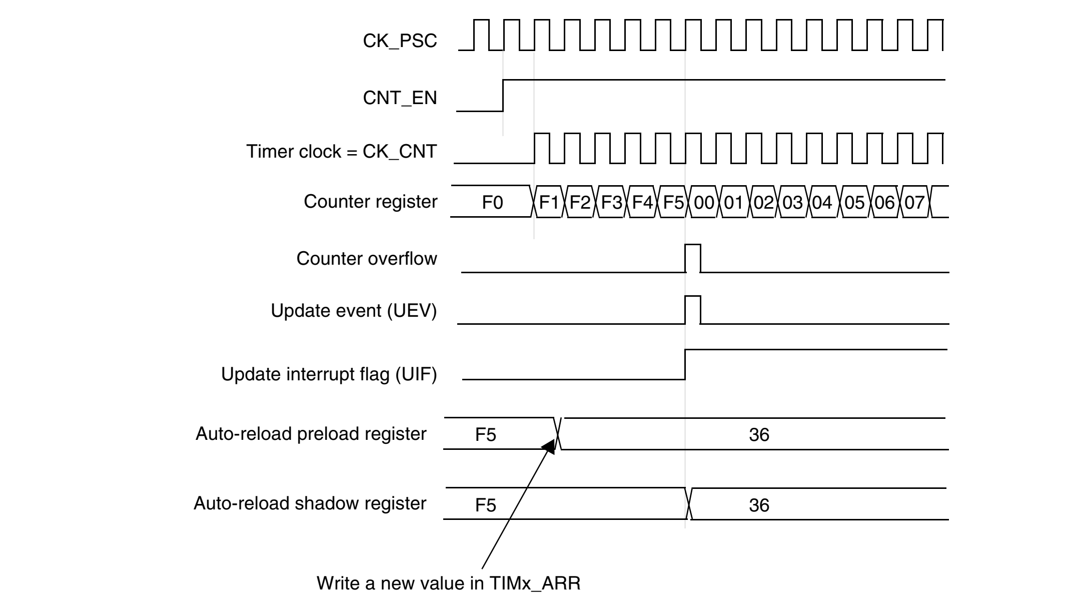


Figure 90. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

An Update event can be generate at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller)

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 91. Counter timing diagram, internal clock divided by 1

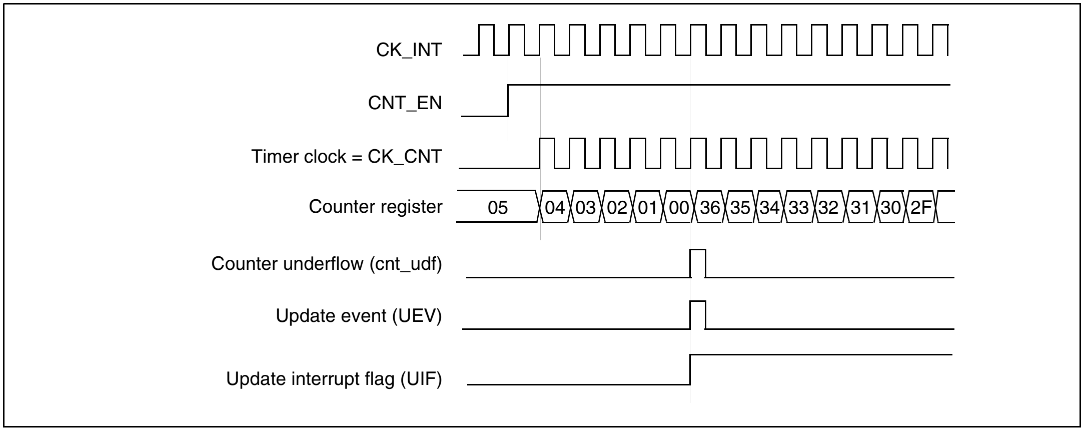


Figure 92. Counter timing diagram, internal clock divided by 2

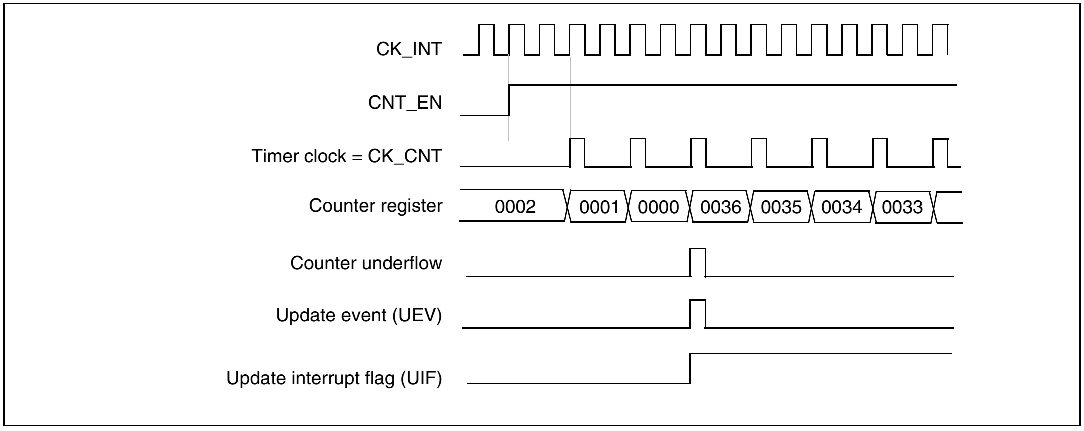


Figure 93. Counter timing diagram, internal clock divided by 4

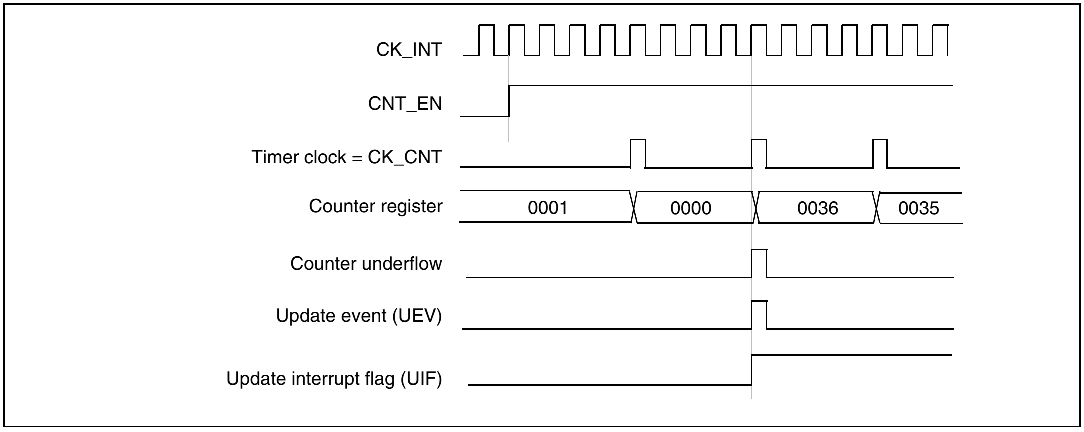


Figure 94. Counter timing diagram, internal clock divided by N

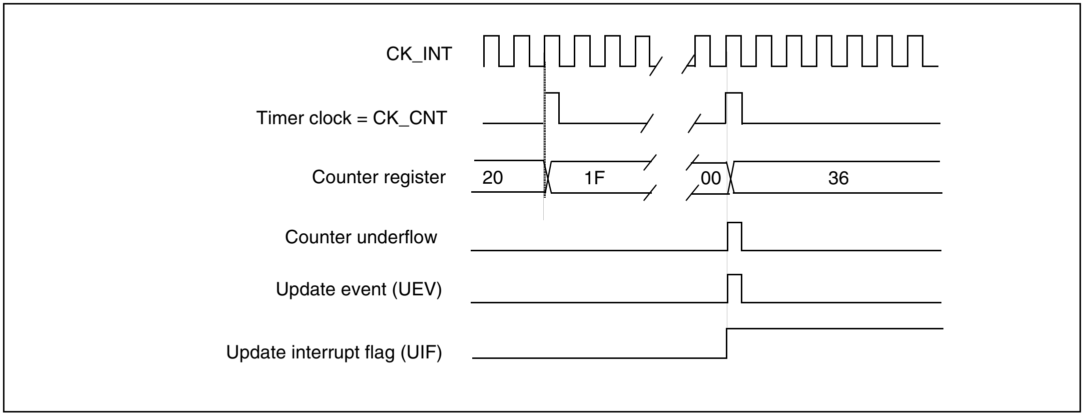
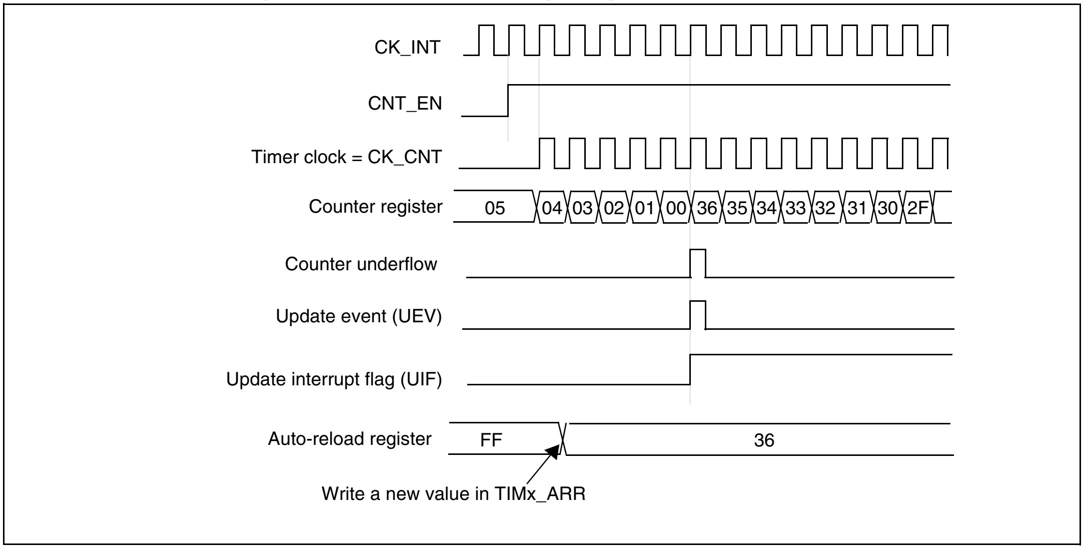


Figure 95. Counter timing diagram, Update event



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the direction bit (DIR from TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

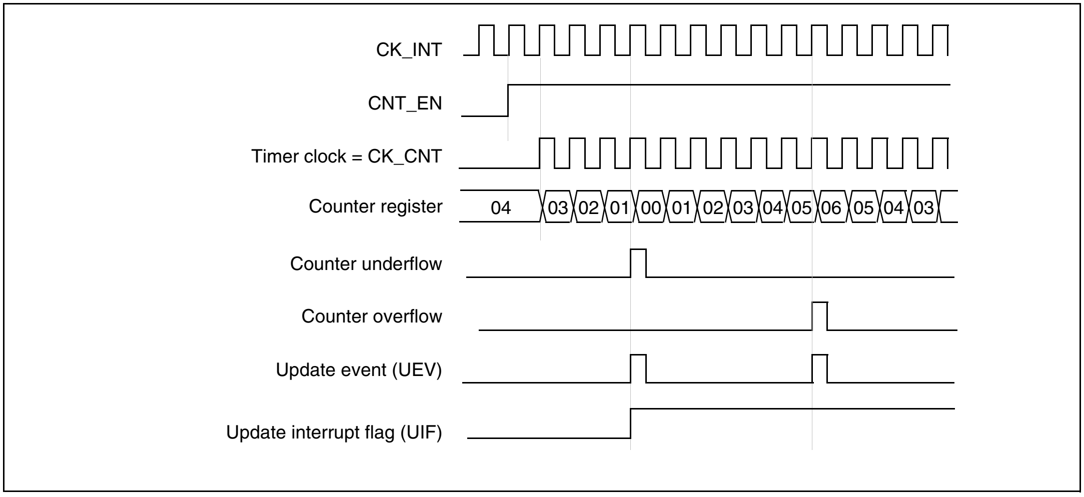
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 96. 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 16.4.1: TIMx control register 1 (TIMx_CR1) on page 375*).

Figure 97. Counter timing diagram, internal clock divided by 2

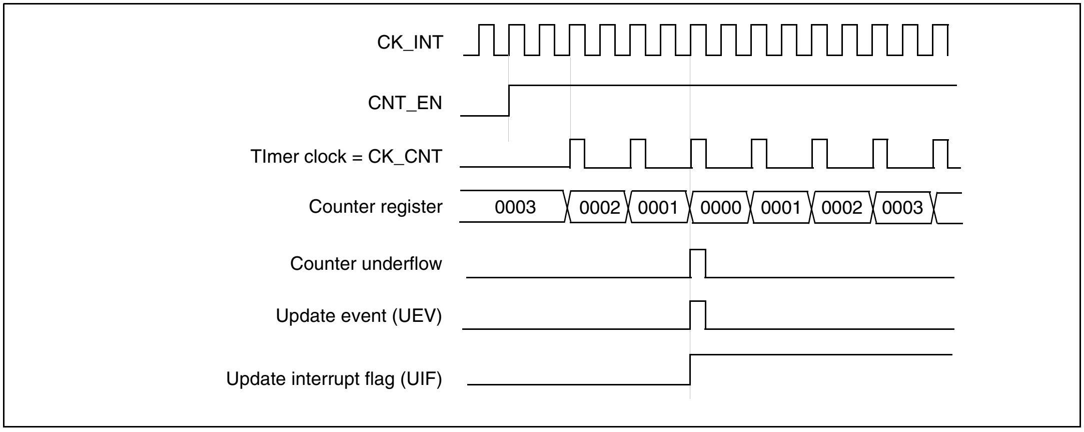
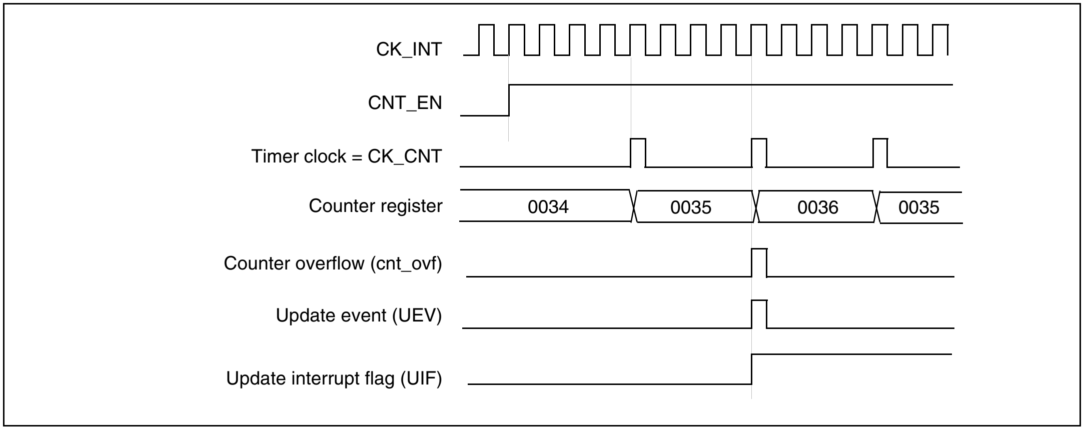


Figure 98. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36



1. Center-aligned mode 2 or 3 is used with an UIF on overflow.

Figure 99. Counter timing diagram, internal clock divided by N

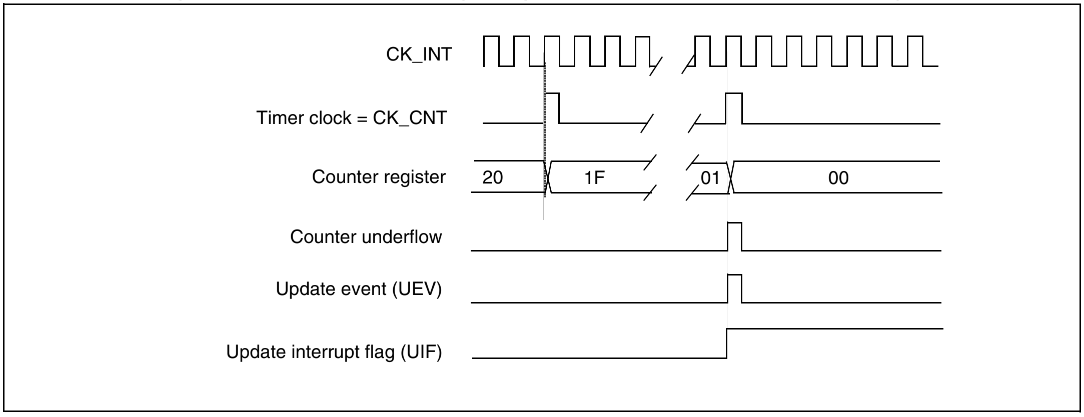


Figure 100. Counter timing diagram, Update event with ARPE=1 (counter underflow)

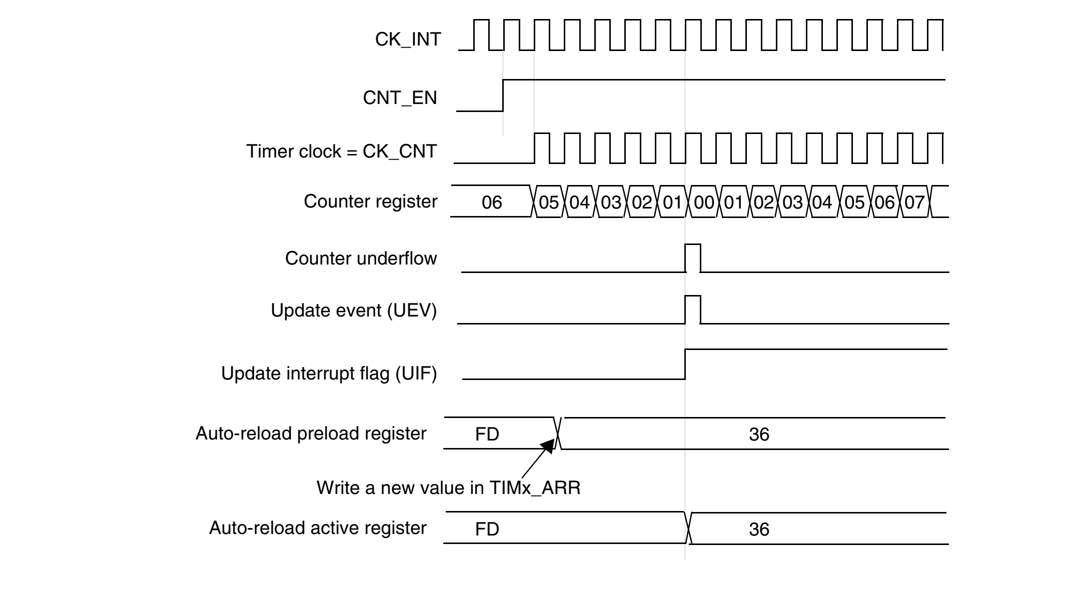
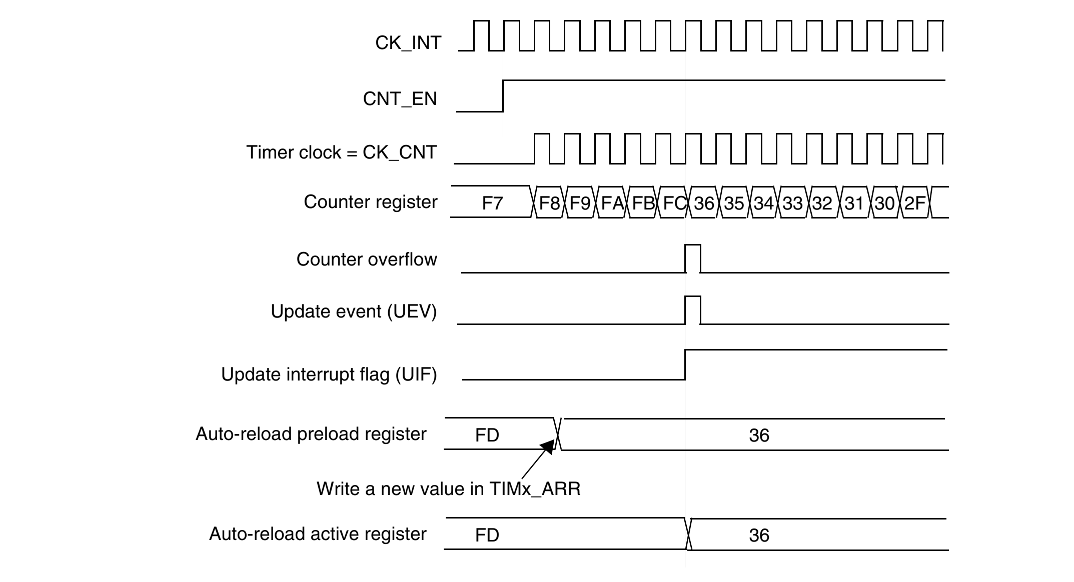


Figure 101. Counter timing diagram, Update event with ARPE=1 (counter overflow)


CK_INT
CNT_EN
Timer clock = CK_CNT
Counter register
F7
F8 F9 FA FB FC 36 35 34 33 32 31 30 2F
Counter overflow
Update event (UEV)
Update interrupt flag (UIF)
Auto-reload preload register
FD
36
Write a new value in TIMx_ARR
Auto-reload active register
FD
36

16.3.3 Clock selection

The counter clock can be provided by the following clock sources:

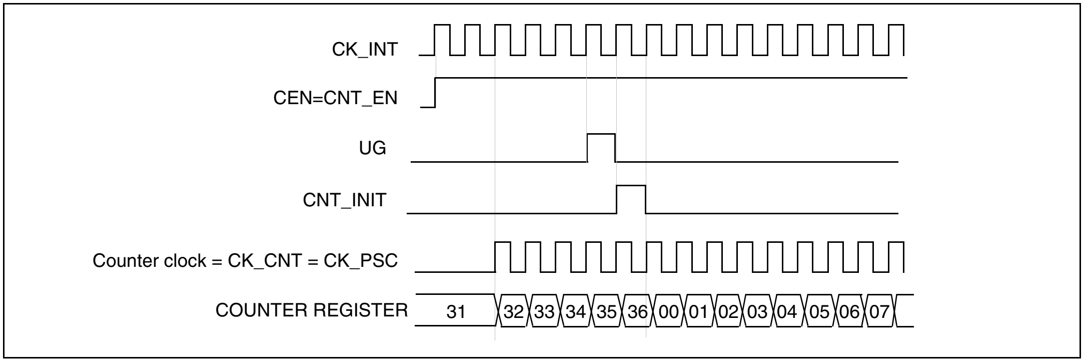
- Internal clock (CK_INT)
- External clock mode1: external input pin (TIx)
- External clock mode2: external trigger input (ETR).
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, you can configure Timer3 to act as a prescaler for Timer 2. Refer to *: Using one timer as prescaler for another on page 370* for more details.

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000 in the TIMx_SMCR register), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 102 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

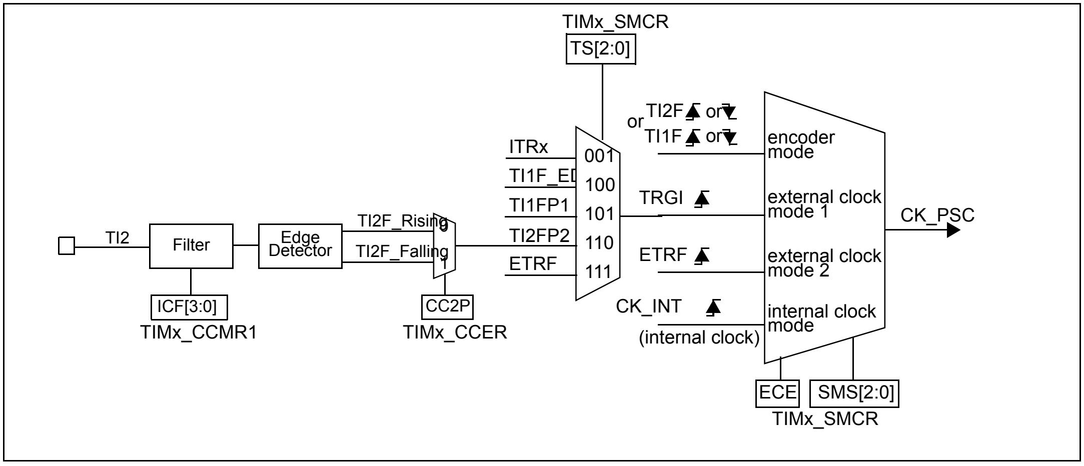
Figure 102. Control circuit in normal mode, internal clock divided by 1



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 103. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S= ‘01 in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).

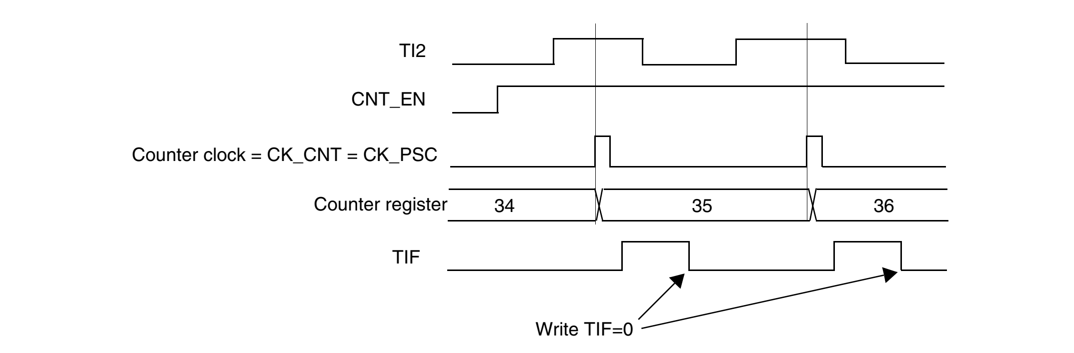
Note: The capture prescaler is not used for triggering, so you don't need to configure it.

3. Select rising edge polarity by writing CC2P=0 and CC2NP=0 in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
5. Select TI2 as the input source by writing TS=110 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 104. Control circuit in external clock mode 1



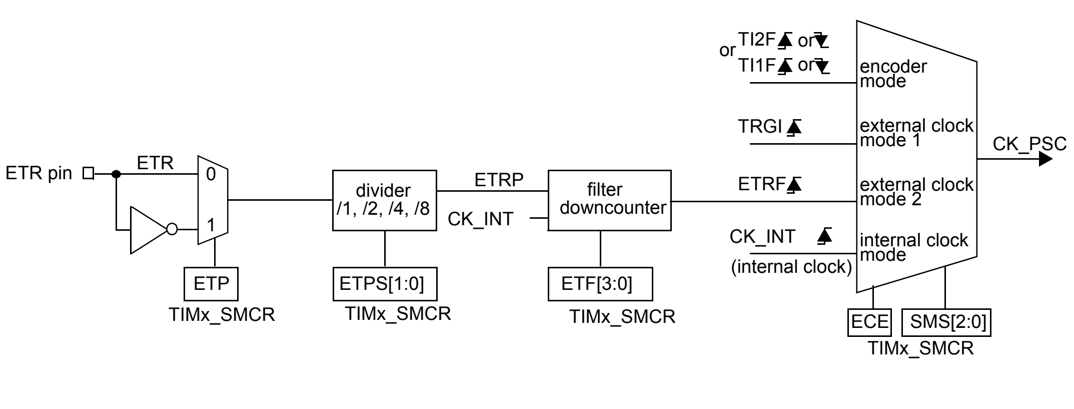
External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

Figure 105 gives an overview of the external trigger input block.

Figure 105. External trigger input block



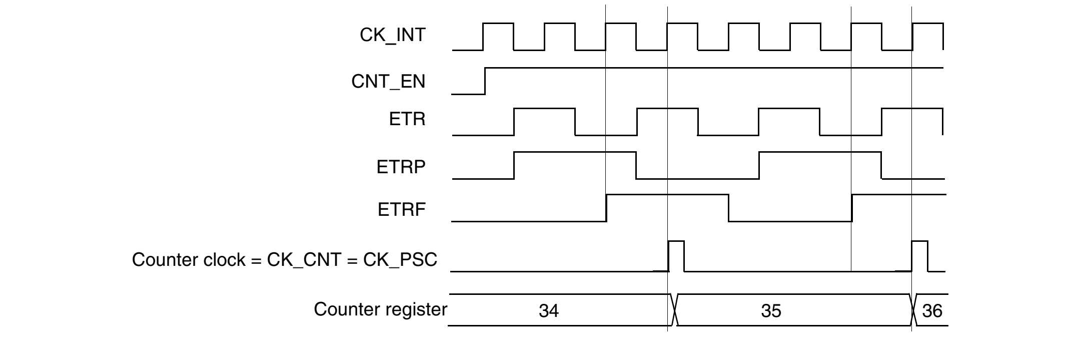
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
2. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register
3. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register
4. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
5. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal.

Figure 106. Control circuit in external clock mode 2



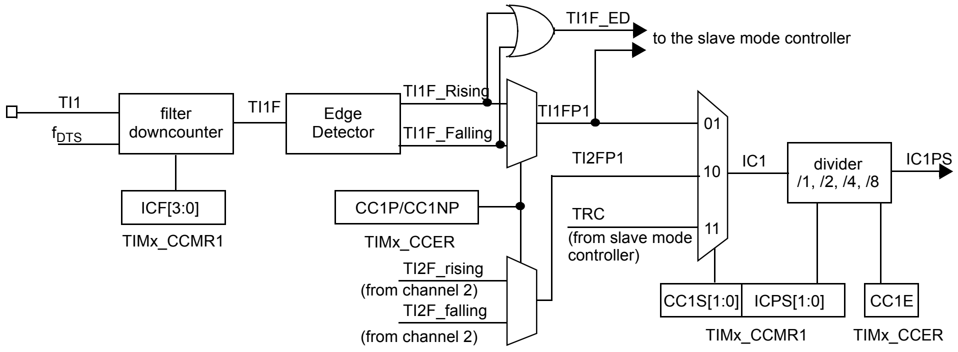
16.3.4 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figure gives an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 107. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 108. Capture/compare channel 1 main circuit

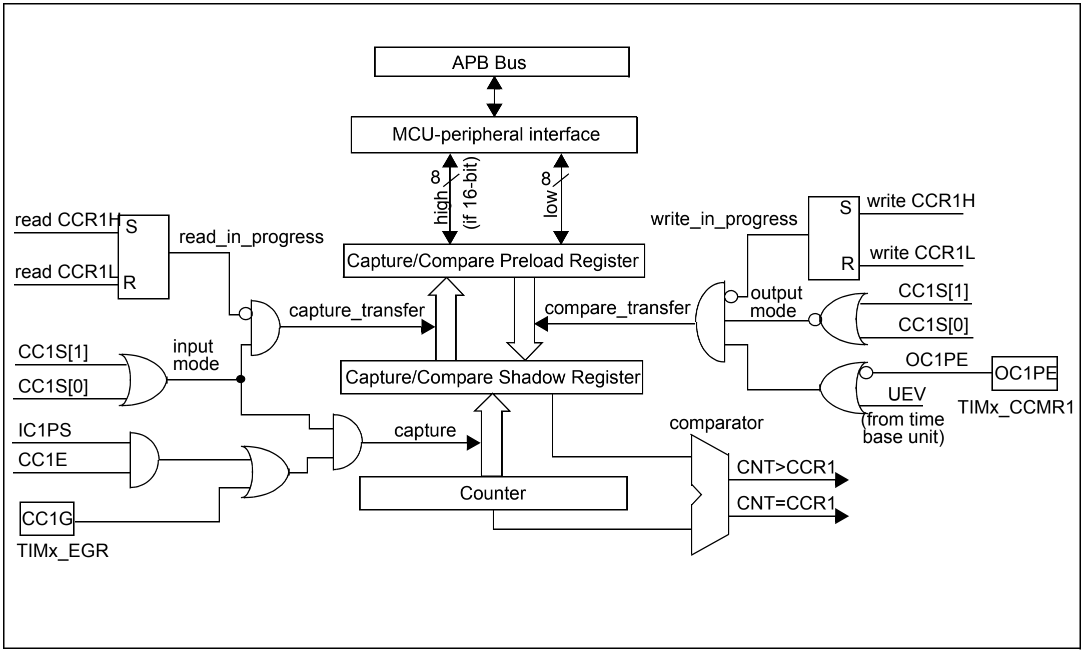
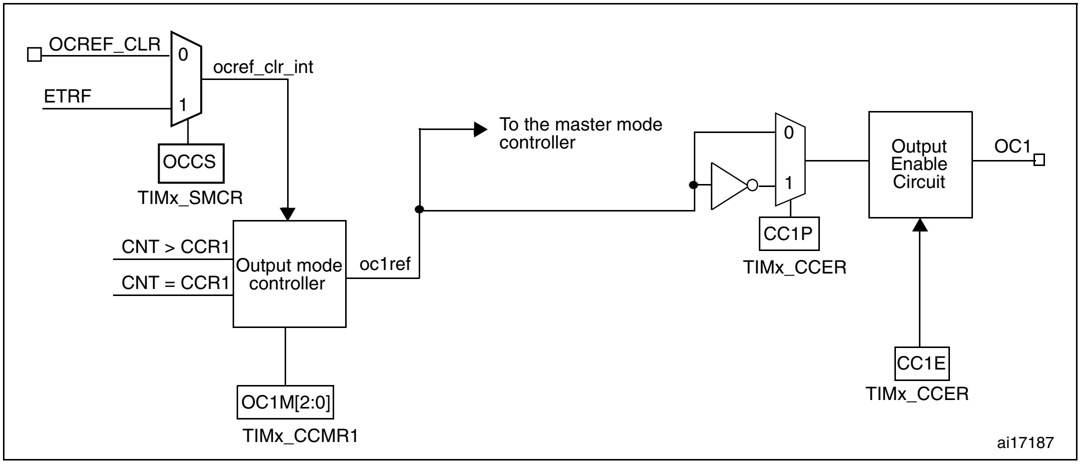


Figure 109. Output stage of capture/compare channel (channel 1)



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.

16.3.5 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when you write it to 0.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

- Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
- Program the input filter duration you need with respect to the signal you connect to the timer (by programming the ICxF bits in the TIMx_CCMRx register if the input is one of the TIx inputs). Let's imagine that, when toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
- Select the edge of the active transition on the TI1 channel by writing the CC1P and CC1NP bits to 0 in the TIMx_CCER register (rising edge in this case).
- Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

16.3.6 PWM input mode

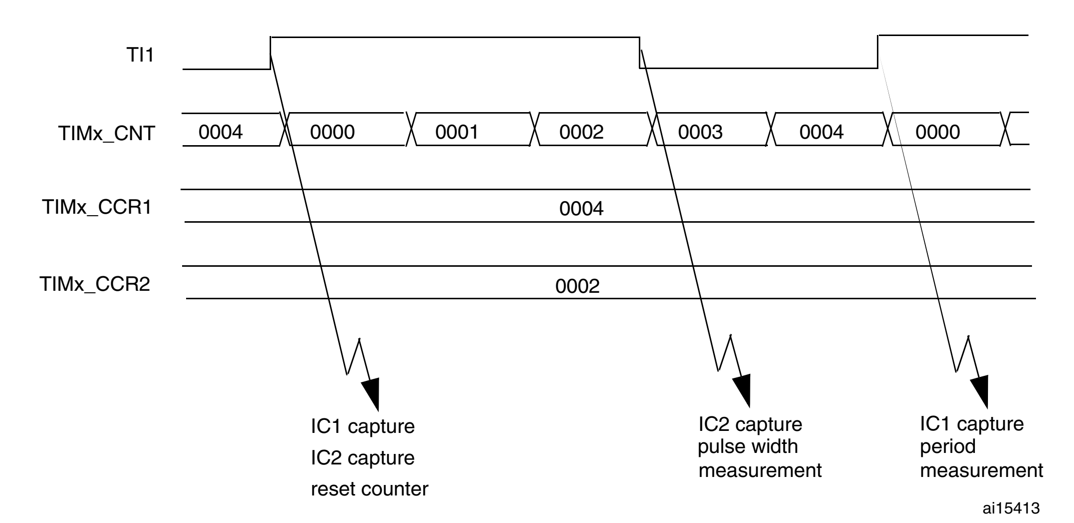
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, you can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P to '0' and the CC1NP bit to '0' (active on rising edge).
- Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P bit to '1' and the CC2NP bit to '0' (active on falling edge).
- Select the valid trigger input: write the TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to '1 in the TIMx_CCER register.

Figure 110. PWM input mode timing



16.3.7 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (ocxref/OCx) to its active level, you just need to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus ocxref is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

e.g.: CCxP=0 (OCx active high) => OCx is forced to high level.

ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the Output Compare Mode section.

16.3.8 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

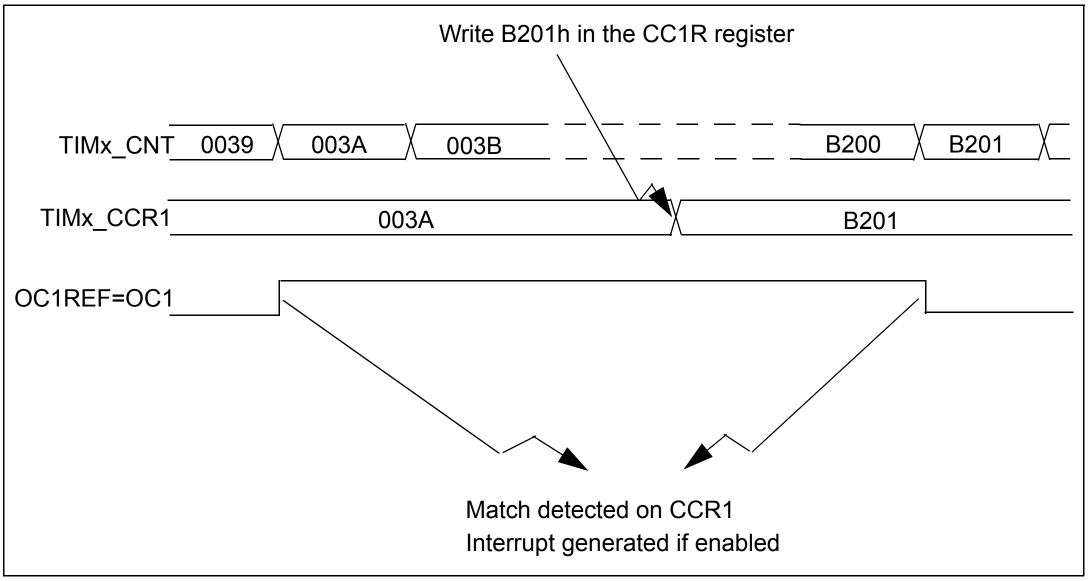
In output compare mode, the update event UEV has no effect on ocxref and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example, you must write OCxM=011, OCxPE=0, CCxP=0 and CCxE=1 to toggle OCx output pin when CNT matches CCRx, CCRx preload is not used, OCx is enabled and active high.
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE=0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 111*.

Figure 111. Output compare mode, toggle on OC1



16.3.9 PWM mode

Pulse width modulation mode allows you to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing 110 (PWM mode 1) or ‘111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. You must enable the corresponding preload register by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, you have to initialize all the registers by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx≤TIMx_CNT or TIMx_CNT≤TIMx_CCRx (depending on the direction of the counter). However, to comply with the ETRF (OCREF can be cleared by an external event through the ETR signal until the next PWM period), the OCREF signal is asserted only:

- When the result of the comparison changes, or
- When the output compare mode (OCxM bits in TIMx_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM=‘000) to one of the PWM modes (OCxM=‘110 or ‘111).

This forces the PWM by software while the timer is running.

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

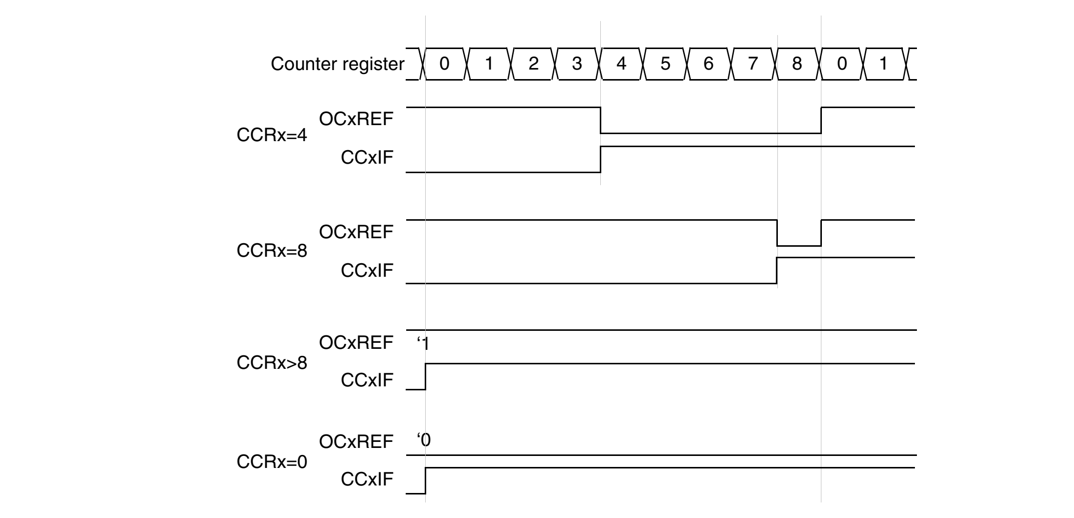
PWM edge-aligned mode

Upcounting configuration

Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Section : Upcounting mode on page 343*.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT <TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at '1. If the compare value is 0 then OCxREF is held at '0. *Figure 112* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 112. Edge-aligned PWM waveforms (ARR=8)



Downcounting configuration

Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to *Section : Downcounting mode on page 346*.

In PWM mode 1, the reference signal ocxref is low as long as TIMx_CNT>TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then ocxref is held at '1. 0% PWM is not possible in this mode.

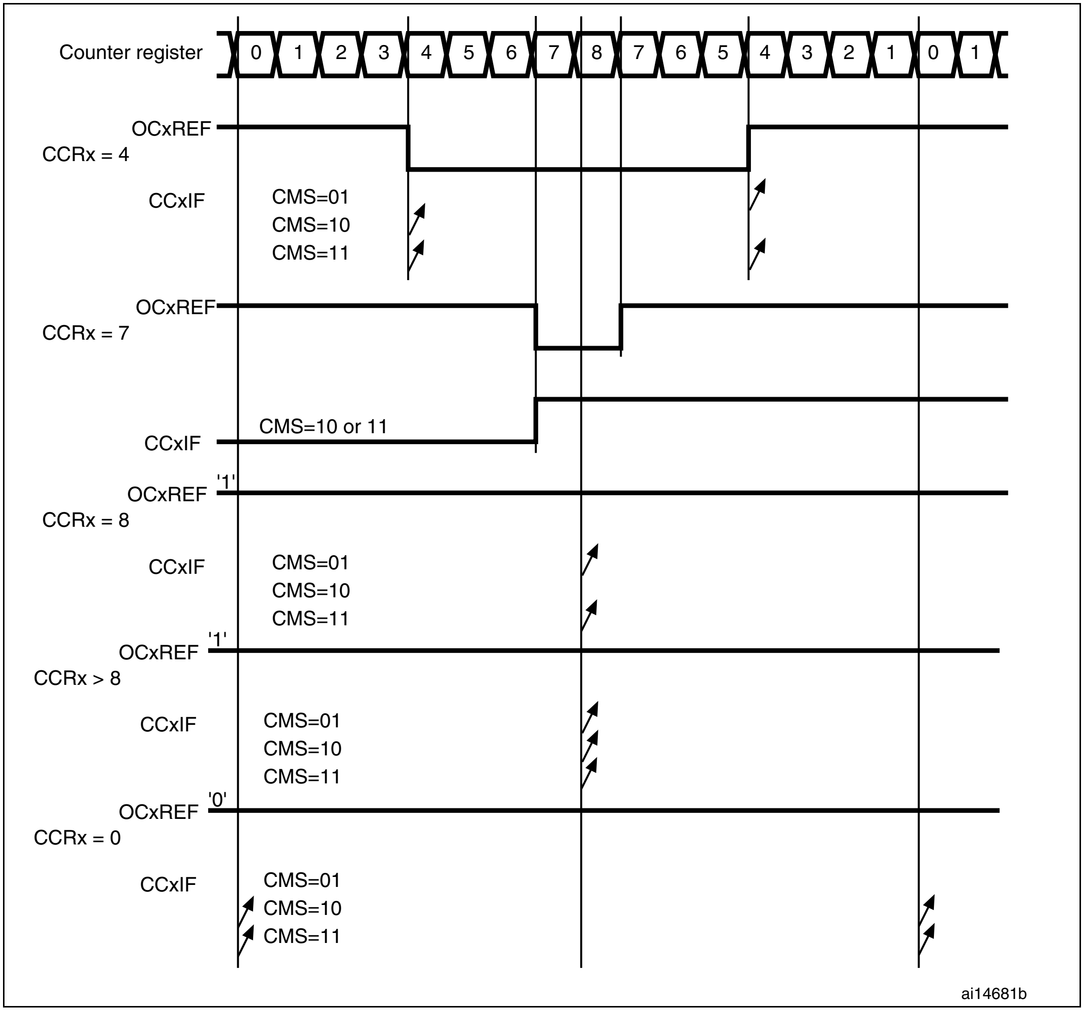
PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from '00 (all the remaining configurations having the same effect on the ocxref/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to *Section : Center-aligned mode (up/down counting) on page 348*.

Figure 113 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 113. 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 you write a value in the counter that is greater than the auto-reload value (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if you write 0 or write the TIMx_ARR value 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.

16.3.10 One-pulse mode

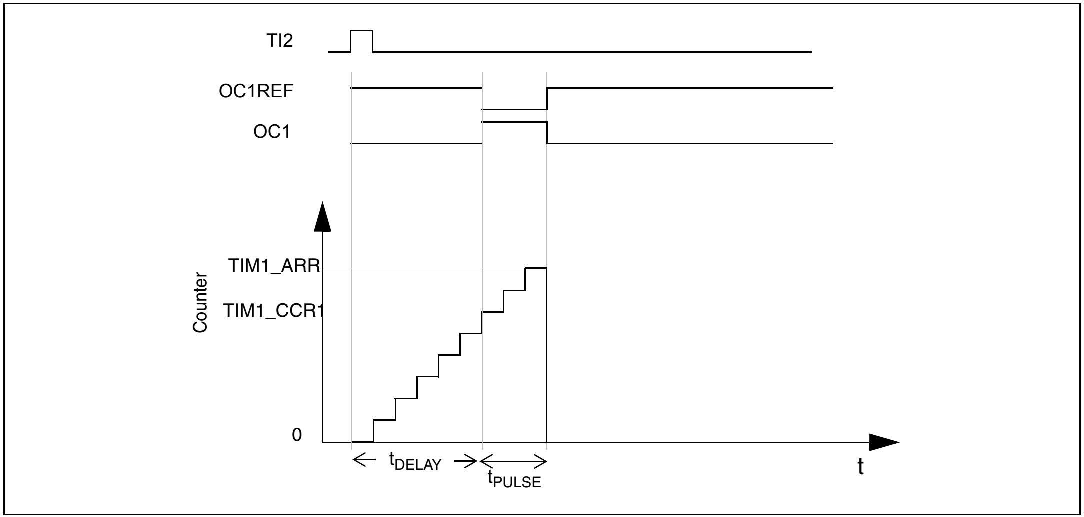
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode 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 114. Example of one-pulse mode



For example you may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

- Map TI2FP2 on TI2 by writing IC2S=01 in the TIMx_CCMR1 register.
- TI2FP2 must detect a rising edge, write CC2P=0 and CC2NP='0' in the TIMx_CCER register.
- Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=110 in the TIMx_SMCR register.
- TI2FP2 is used to start the counter by writing SMS to '110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let’s say you want to build a waveform with a transition from ‘0 to ‘1 when a compare match occurs and a transition from ‘1 to ‘0 when the counter reaches the auto-reload value. To do this you enable PWM mode 2 by writing OC1M=111 in the TIMx_CCMR1 register. You can optionally enable the preload registers by writing OC1PE=1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case you have to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to ‘0 in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low.

You only want 1 pulse (Single mode), so you write '1 in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If you want to output a waveform with the minimum delay, you can set the OCxFE bit in the TIMx_CCMRx register. Then OCxRef (and OCx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

16.3.11 Clearing the OCxREF signal on an external event

The OCxREF signal for a given channel can be driven Low by applying a High level to the ETRF input (OCxCE enable bit of the corresponding TIMx_CCMRx register set to '1'). The OCxREF signal remains Low until the next update event, UEV, occurs.

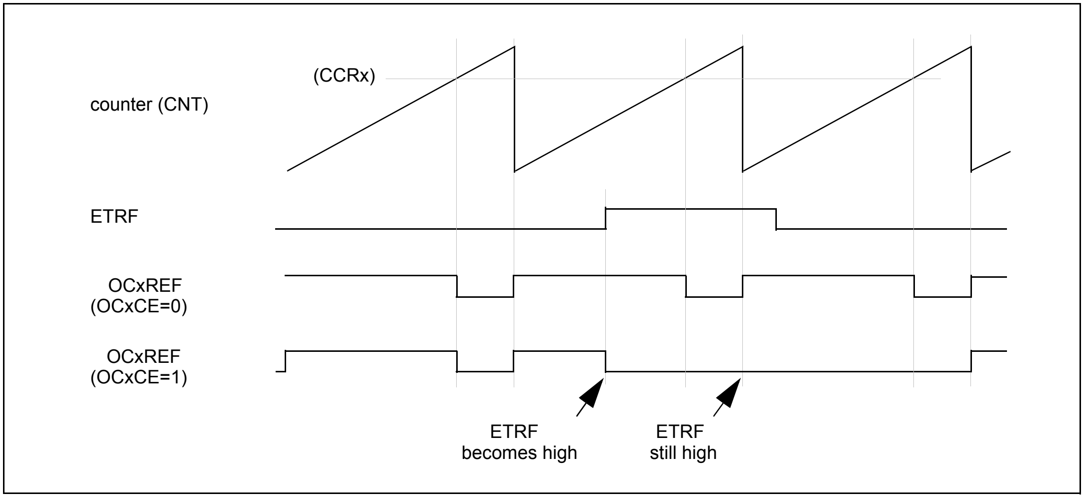
This function can only be used in output compare and PWM modes, and does not work in forced mode.

For example, the ETR signal can be connected to the output of a comparator to be used for current handling. In this case, ETR must be configured as follows:

1. The external trigger prescaler should be kept off: bits ETPS[1:0] in the TIMx_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIM1_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application’s needs.

Figure 115 shows the behavior of the OCxREF signal when the ETRF input becomes high, for both values of the OCxCE enable bit. In this example, the timer TIMx is programmed in PWM mode.

Figure 115. Clearing TIMx OCxREF



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), OCxREF is enabled again at the next counter overflow.

16.3.12 Encoder interface mode

To select Encoder Interface mode write SMS=‘001 in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS=010 if it is counting on TI1 edges only and SMS=011 if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. CC1NP and CC2NP must be kept cleared. When needed, you can program the input filter as well.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. Refer to *Table 65*. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to ‘1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So you must configure TIMx_ARR before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal.

In this mode, the counter is modified automatically following the speed and the direction of the incremental encoder and its content, therefore, always represents the encoder’s position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 don’t switch at the same time.

Table 65. Counting direction versus encoder signals

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

An external incremental encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

Figure 116 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S= '01' (TIMx_CCMR1 register, TI1FP1 mapped on TI1)
- CC2S= '01' (TIMx_CCMR2 register, TI2FP2 mapped on TI2)
- CC1P= '0', CC1NP = '0', IC1F ='0000' (TIMx_CCER register, TI1FP1 noninverted, TI1FP1=TI1)
- CC2P= '0', CC2NP = '0', IC2F ='0000' (TIMx_CCER register, TI2FP2 noninverted, TI2FP2=TI2)
- SMS= '011' (TIMx_SMCR register, both inputs are active on both rising and falling edges)
- CEN = 1 (TIMx_CR1 register, Counter is enabled)

Figure 116. Example of counter operation in encoder interface mode

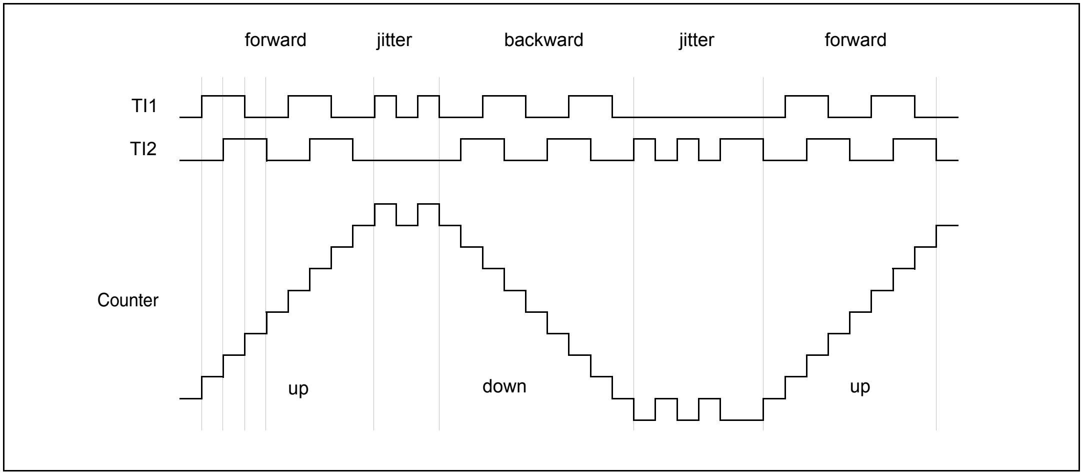
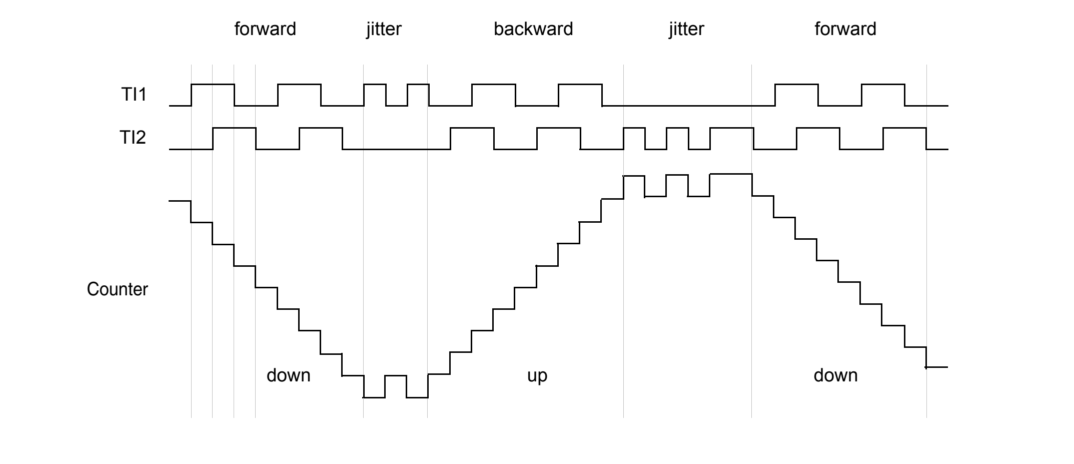


Figure 117 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P=1).

Figure 117. Example of encoder interface mode with TI1FP1 polarity inverted



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. You can obtain dynamic information (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. You can do this by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request generated by a Real-Time clock.

16.3.13 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 a XOR gate, combining the three input pins TIMx_CH1 to TIMx_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

16.3.14 Timers and external trigger synchronization

The TIMx Timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write

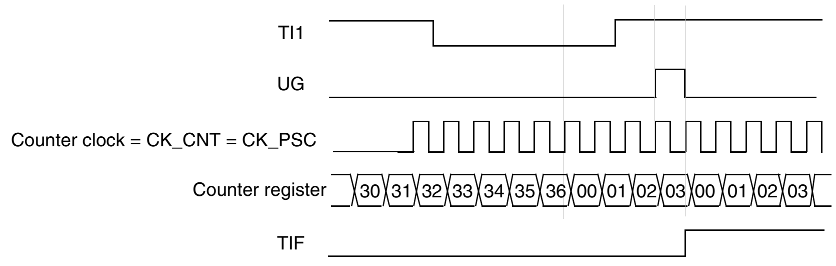
CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edges only).

- Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.
- Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 118. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

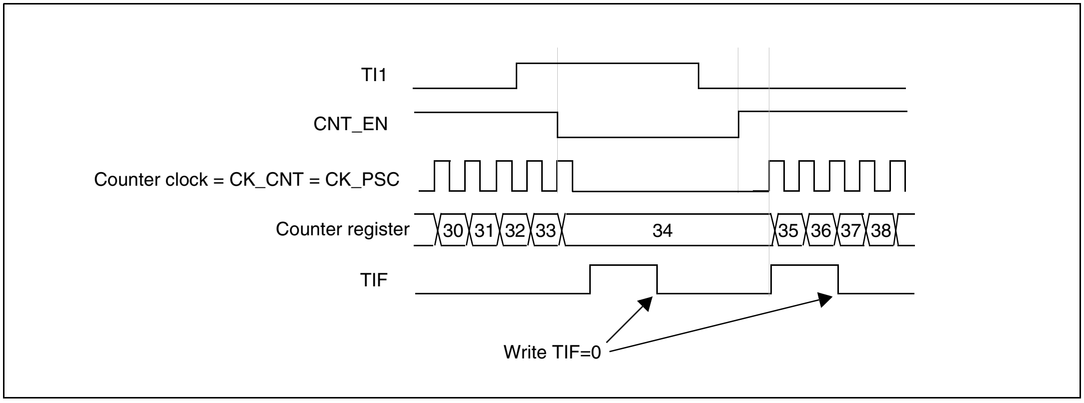
In the following example, the upcounter counts only when TI1 input is low:

- Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don’t need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so you don’t need to configure it. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.
- Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn’t start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 119. 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.

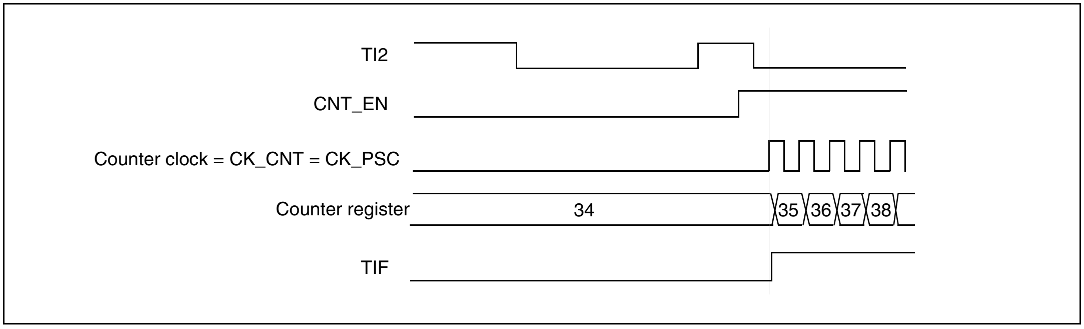
In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don’t need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so you don’t need to configure it. 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).
- Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI2 as the input source by writing TS=110 in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Figure 120. Control circuit in trigger mode



Slave mode: External Clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode or trigger mode. It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

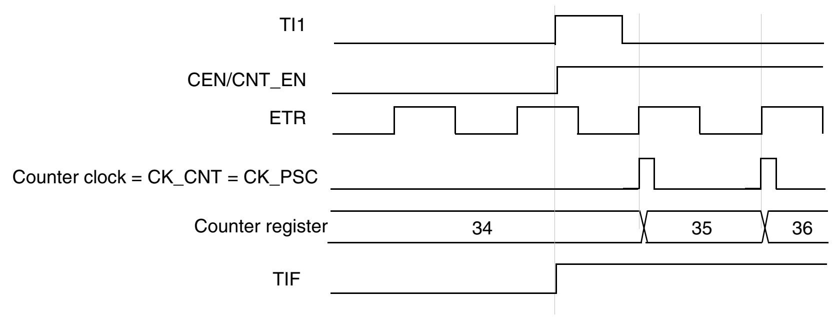
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS = 00: prescaler disabled
   - ETP = 0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F = 0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S = 01 in TIMx_CCMR1 register to select only the input capture source
   - CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

Figure 121. Control circuit in external clock mode 2 + trigger mode



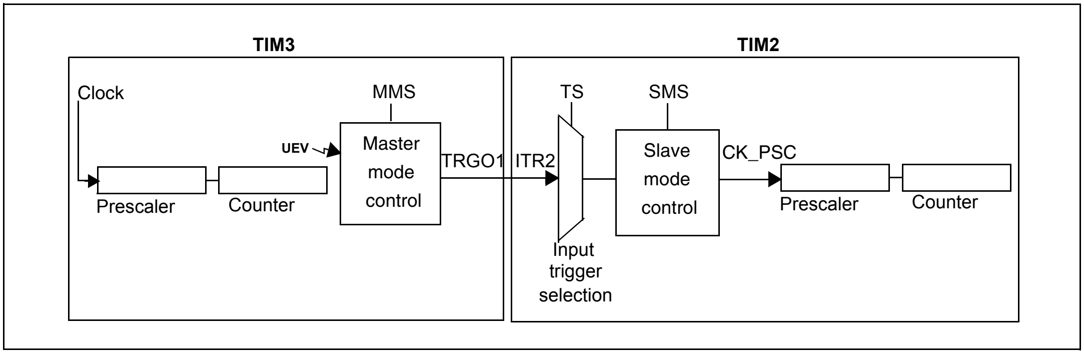
16.3.15 Timer synchronization

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 122: Master/Slave timer example presents an overview of the trigger selection and the master mode selection blocks.

Using one timer as prescaler for another

Figure 122. Master/Slave timer example



For example, you can configure TIM3 to act as a prescaler for TIM2. Refer to *Figure 122*. To do this:

- Configure TIM3 in master mode so that it outputs a periodic trigger signal on each update event UEV. If you write MMS=010 in the TIM3_CR2 register, a rising edge is output on TRGO1 each time an update event is generated.
- To connect the TRGO1 output of TIM3 to TIM2, TIM2 must be configured in slave mode using ITR2 as internal trigger. You select this through the TS bits in the TIM2_SMCR register (writing TS=010).
- Then you put the slave mode controller in external clock mode 1 (write SMS=111 in the TIM2_SMCR register). This causes TIM2 to be clocked by the rising edge of the periodic TIM3 trigger signal (which correspond to the TIM3 counter overflow).
- Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: If OCx is selected on TIM3 as the trigger output (MMS=1xx), its rising edge is used to clock the counter of TIM2.

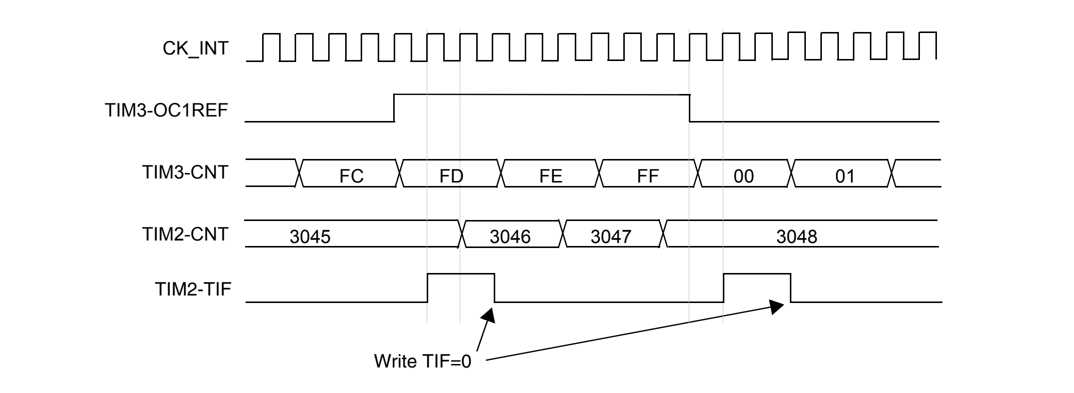
Using one timer to enable another timer

In this example, we control the enable of TIM2 with the output compare 1 of Timer 3. Refer to *Figure 122* for connections. TIM2 counts on the divided internal clock only when OC1REF of TIM3 is high. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK\_CNT} = f_{CK\_INT}/3$).

- Configure TIM3 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM3_CR2 register).
- Configure the TIM3 OC1REF waveform (TIM3_CCMR1 register).
- Configure TIM2 to get the input trigger from TIM3 (TS=000 in the TIM2_SMCR register).
- Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).
- Enable TIM2 by writing '1 in the CEN bit (TIM2_CR1 register).
- Start TIM3 by writing '1 in the CEN bit (TIM3_CR1 register).

Note: The counter 2 clock is not synchronized with counter 1, this mode only affects the TIM2 counter enable signal.

Figure 123. Gating TIM2 with OC1REF of TIM3

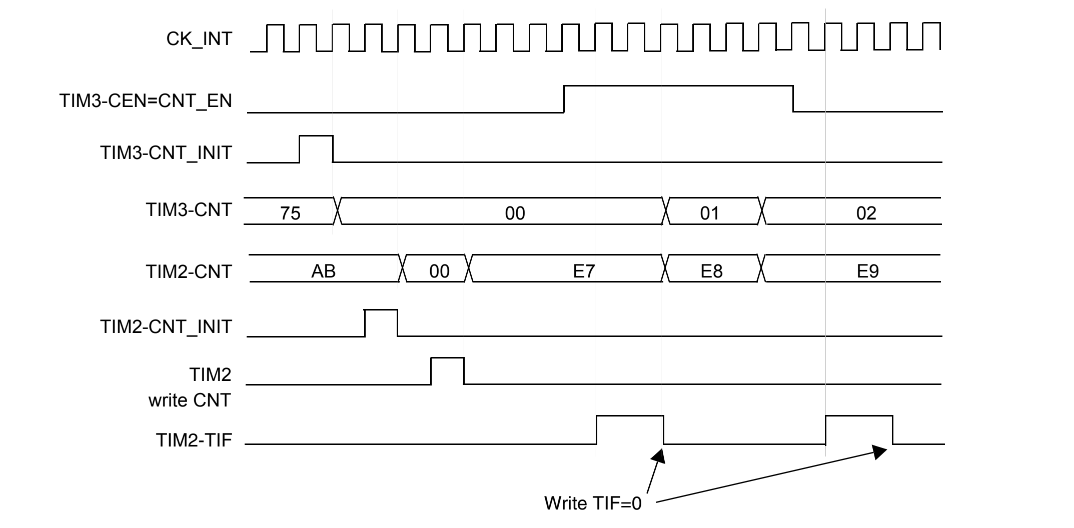


In the example in *Figure 123*, the TIM2 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM3. You can then write any value you want in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example, we synchronize TIM3 and TIM2. TIM3 is the master and starts from 0. TIM2 is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM2 stops when TIM3 is disabled by writing ‘0 to the CEN bit in the TIM3_CR1 register:

- Configure TIM3 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM3_CR2 register).
- Configure the TIM3 OC1REF waveform (TIM3_CCMR1 register).
- Configure TIM2 to get the input trigger from TIM3 (TS=000 in the TIM2_SMCR register).
- Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).
- Reset TIM3 by writing ‘1 in UG bit (TIM3_EGR register).
- Reset TIM2 by writing ‘1 in UG bit (TIM2_EGR register).
- Initialize TIM2 to 0xE7 by writing ‘0xE7’ in the TIM2 counter (TIM2_CNTL).
- Enable TIM2 by writing ‘1 in the CEN bit (TIM2_CR1 register).
- Start TIM3 by writing ‘1 in the CEN bit (TIM3_CR1 register).
- Stop TIM3 by writing ‘0 in the CEN bit (TIM3_CR1 register).

Figure 124. Gating TIM2 with Enable of TIM3

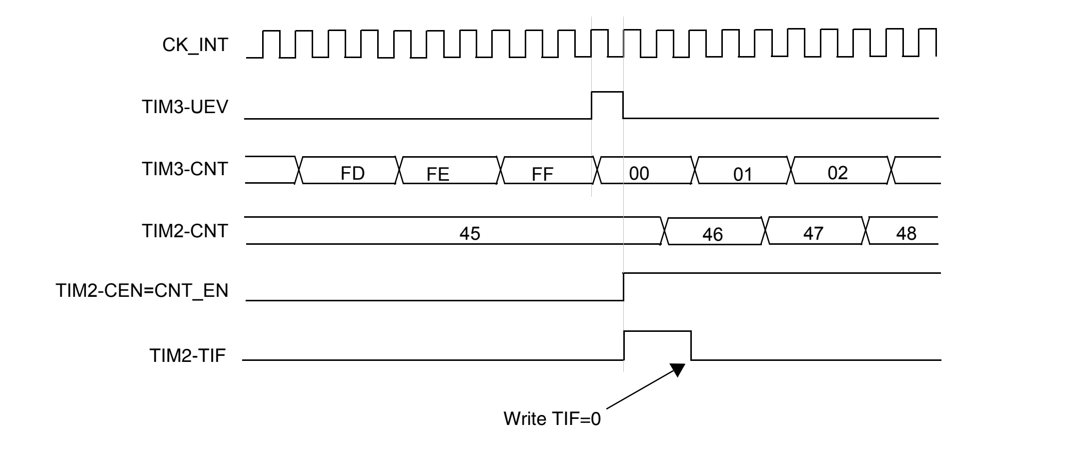


Using one timer to start another timer

In this example, we set the enable of Timer 2 with the update event of Timer 3. Refer to *Figure 122* for connections. Timer 2 starts counting from its current value (which can be non-zero) on the divided internal clock as soon as the update event is generated by Timer 1. When Timer 2 receives the trigger signal its CEN bit is automatically set and the counter counts until we write ‘0 to the CEN bit in the TIM2_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK\_CNT} = f_{CK\_INT}/3$).

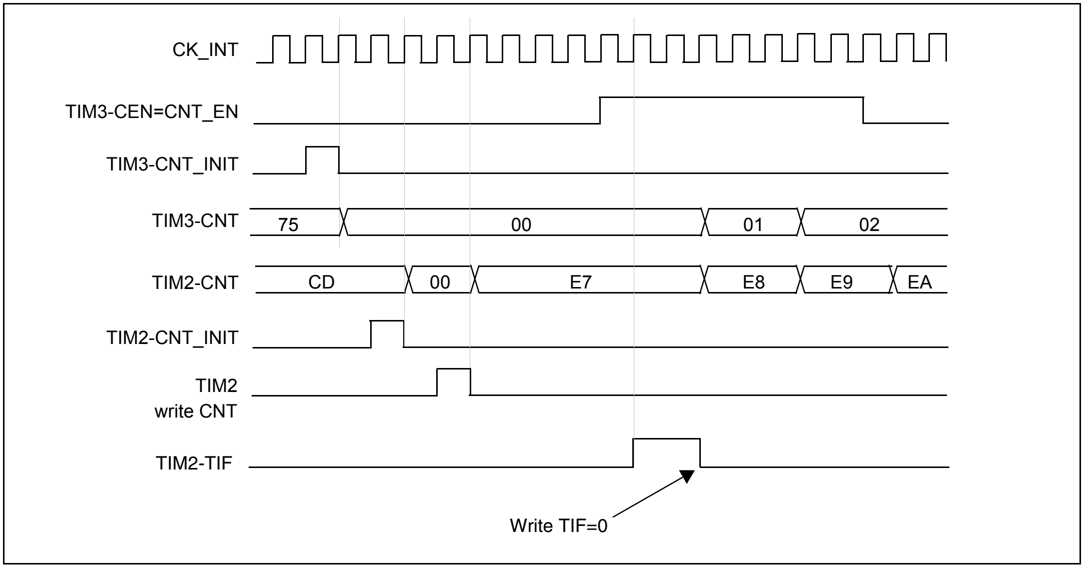
- Configure TIM3 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM3_CR2 register).
- Configure the TIM3 period (TIM3_ARR registers).
- Configure TIM2 to get the input trigger from TIM3 (TS=000 in the TIM2_SMCR register).
- Configure TIM2 in trigger mode (SMS=110 in TIM2_SMCR register).
- Start TIM3 by writing ‘1 in the CEN bit (TIM3_CR1 register).

Figure 125. Triggering TIM2 with update of TIM3



As in the previous example, you can initialize both counters before starting counting. *Figure 126* shows the behavior with the same configuration as in *Figure 125* but in trigger mode instead of gated mode (SMS=110 in the TIM2_SMCR register).

Figure 126. Triggering TIM2 with Enable of TIM3



Using one timer as prescaler for another timer

For example, you can configure TIM3 to act as a prescaler for TIM2. Refer to *Figure 122* for connections. To do this:

- Configure TIM3 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM3_CR2 register). then it outputs a periodic signal on each counter overflow.
- Configure the TIM3 period (TIM3_ARR registers).
- Configure TIM2 to get the input trigger from TIM3 (TS=000 in the TIM2_SMCR register).
- Configure TIM2 in external clock mode 1 (SMS=111 in TIM2_SMCR register).
- Start TIM2 by writing ‘1 in the CEN bit (TIM2_CR1 register).
- Start TIM3 by writing ‘1 in the CEN bit (TIM3_CR1 register).

Starting 2 timers synchronously in response to an external trigger

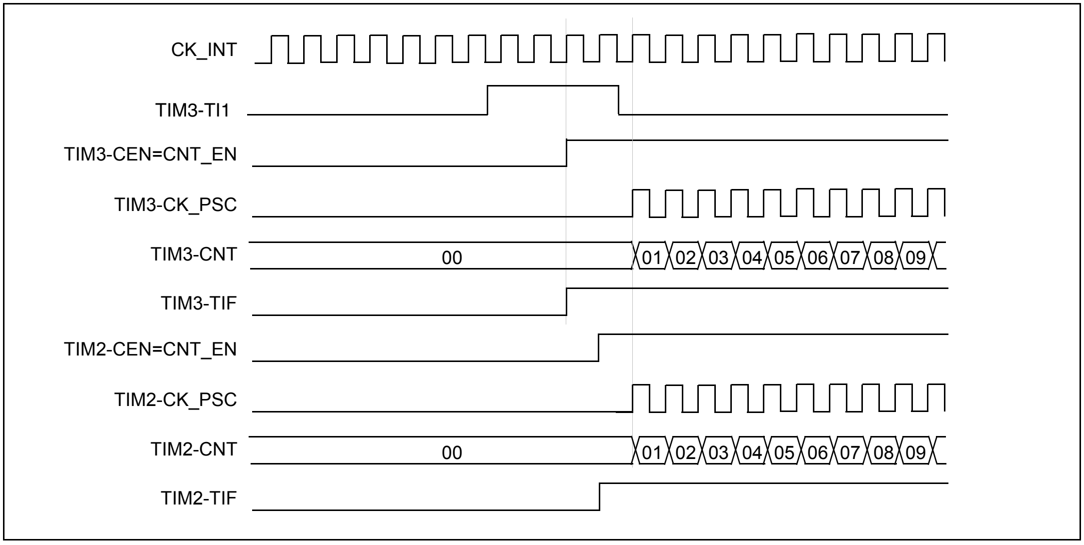
In this example, we set the enable of TIM3 when its TI1 input rises, and the enable of TIM2 with the enable of TIM3. Refer to *Figure 122* for connections. To ensure the counters are aligned, TIM3 must be configured in Master/Slave mode (slave with respect to TI1, master with respect to TIM2):

- Configure TIM3 master mode to send its Enable as trigger output (MMS=001 in the TIM3_CR2 register).
- Configure TIM3 slave mode to get the input trigger from TI1 (TS=100 in the TIM3_SMCR register).
- Configure TIM3 in trigger mode (SMS=110 in the TIM3_SMCR register).
- Configure the TIM3 in Master/Slave mode by writing MSM=1 (TIM3_SMCR register).
- Configure TIM2 to get the input trigger from TIM3 (TS=000 in the TIM2_SMCR register).
- Configure TIM2 in trigger mode (SMS=110 in the TIM2_SMCR register).

When a rising edge occurs on TI1 (TIM3), both counters starts counting synchronously on the internal clock and both TIF flags are set.

Note: *In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but you can easily insert an offset between them by writing any of the counter registers (TIMx_CNT). You can see that the master/slave mode insert a delay between CNT_EN and CK_PSC on TIM3.*

Figure 127. Triggering TIM3 and TIM2 with TIM3 TI1 input



16.3.16 Debug mode

When the microcontroller enters debug mode (Cortex™-M3 core - halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBGMCU module. For more details, refer to *Section 29.16.2: Debug support for timers, watchdog and I2C*.

16.4 TIMx registers

Refer to *Section 1.1 on page 37* for a list of abbreviations used in register descriptions.

The 32-bit peripheral registers have to be written by words (32 bits). All other peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

16.4.1 TIMx control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 9:8 **CKD**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (ETR, TIx),

00: $t_{DTS} = t_{CK\_INT}$
01: $t_{DTS} = 2 \times t_{CK\_INT}$
10: $t_{DTS} = 4 \times t_{CK\_INT}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS**: 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.

16.4.2 TIMx control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TI1S	MMS[2:0]			CCDS	Reserved		
								rw	rw	rw	rw	rw			

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

Bit 7 **TI1S**: TI1 selection

0: The TIMx_CH1 pin is connected to TI1 input
1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

001: **Enable** - the Counter enable signal, CNT_EN, is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode.

When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO)

100: **Compare** - OC1REF signal is used as trigger output (TRGO)
101: **Compare** - OC2REF signal is used as trigger output (TRGO)
110: **Compare** - OC3REF signal is used as trigger output (TRGO)
111: **Compare** - OC4REF signal is used as trigger output (TRGO)

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

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

16.4.3 TIMx slave mode control register (TIMx_SMCR)

Address offset: 0x08

Reset value: 0x0000

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

Bit 15 **ETP**: External trigger polarity

This bit selects whether ETR or $\overline{\text{ETR}}$ is used for trigger operations

0: ETR is non-inverted, active at high level or rising edge

1: ETR is inverted, active at low level or falling edge

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

1: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=111).

2: It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 111).

3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.

Bits 13:12 **ETPS**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of CK_INT frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

00: Prescaler OFF

01: ETRP frequency divided by 2

10: ETRP frequency divided by 4

11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000:	No filter, sampling is done at f_{DTS}	1000:	$f_{SAMPLING}=f_{DTS}/8$, N=6
0001:	$f_{SAMPLING}=f_{CK\_INT}$, N=2	1001:	$f_{SAMPLING}=f_{DTS}/8$, N=8
0010:	$f_{SAMPLING}=f_{CK\_INT}$, N=4	1010:	$f_{SAMPLING}=f_{DTS}/16$, N=5
0011:	$f_{SAMPLING}=f_{CK\_INT}$, N=8	1011:	$f_{SAMPLING}=f_{DTS}/16$, N=6
0100:	$f_{SAMPLING}=f_{DTS}/2$, N=6	1100:	$f_{SAMPLING}=f_{DTS}/16$, N=8
0101:	$f_{SAMPLING}=f_{DTS}/2$, N=8	1101:	$f_{SAMPLING}=f_{DTS}/32$, N=5
0110:	$f_{SAMPLING}=f_{DTS}/4$, N=6	1110:	$f_{SAMPLING}=f_{DTS}/32$, N=6
0111:	$f_{SAMPLING}=f_{DTS}/4$, N=8	1111:	$f_{SAMPLING}=f_{DTS}/32$, N=8

Bit 7 **MSM:** Master/Slave mode

0: No action

1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 6:4 **TS:** Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.
000: Internal Trigger 0 (ITR0).
001: Internal Trigger 1 (ITR1).
010: Internal Trigger 2 (ITR2).
011: Internal Trigger 3 (ITR3).
100: TI1 Edge Detector (TI1F_ED)
101: Filtered Timer Input 1 (TI1FP1)
110: Filtered Timer Input 2 (TI2FP2)
111: External Trigger input (ETRF)

See for more details on ITRx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS:** OCREF clear selection

This bit is used to select the OCREF clear source

0: OCREF_CLR_INT is connected to the OCREF_CLR input
1: OCREF_CLR_INT is connected to ETRF

Bits 2:0 **SMS:** Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.
000: Slave mode disabled - if CEN = ‘1 then the prescaler is clocked directly by the internal clock.
001: Encoder mode 1 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.
010: Encoder mode 2 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.
011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.
100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.

101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.
111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=100). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Table 66. TIMx internal trigger connection

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM2	TIM9	TIM10	TIM3	TIM4
TIM3	TIM9	TIM2	TIM11	TIM4
TIM4	TIM10	TIM2	TIM3	TIM9

16.4.4 TIMx DMA/Interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable
0: CC4 DMA request disabled
1: CC4 DMA request enabled

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable
0: CC3 DMA request disabled
1: CC3 DMA request enabled

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable
0: CC4 interrupt disabled
1: CC4 interrupt enabled

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable
0: CC3 interrupt disabled
1: CC3 interrupt enabled

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable

0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable

0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled
1: Update interrupt enabled

16.4.5 TIMx status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CC4OF	CC3OF	CC2OF	CC1OF	Reserved		TIF	Res	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag

refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag

refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag

refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0.
0: No overcapture has been detected
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred
1: Trigger interrupt pending

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag

refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag

refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

If channel CC1 is configured as output:

This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the TIMx_CR1 register description). It is cleared by software.

0: No match

1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in upcounting and up/down-counting modes) or underflow (in downcounting mode)

If channel CC1 is configured as input:

This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.

0: No input capture occurred

1: The counter value has been captured in TIMx_CCR1 register (An edge has been detected on IC1 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

At overflow or underflow (for TIM2 to TIM4) and if UDIS=0 in the TIMx_CR1 register.

When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS=0 and UDIS=0 in the TIMx_CR1 register.

16.4.6 TIMx event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

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

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4G**: Capture/compare 4 generation

refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation

refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation

refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

16.4.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

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]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bit 15 **OC2CE:** Output compare 2 clear enable

Bits 14:12 **OC2M[2:0]**: Output compare 2 mode

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 **OC1CE:** Output compare 1 clear enable

OC1CE: Output Compare 1 Clear Enable
0: OC1Ref is not affected by the ETRF input
1: OC1Ref is cleared as soon as a High level is detected on ETRF input

Bits 6:4 **OC1M**: Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.
000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).
001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.
100: Force inactive level - OC1REF is forced low.
101: Force active level - OC1REF is forced high.
110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF=‘0) as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF=1).
111: 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.

Note: In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

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: **1:** 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).*

***2:** The PWM mode can be used without validating the preload register only in one-pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.*

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
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**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output.
01: CC1 channel is configured as input, IC1 is mapped on TI1.
10: CC1 channel is configured as input, IC1 is mapped on TI2.
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

Input capture mode

Bits 15:12 **IC2F**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S**: Capture/compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output.
01: CC2 channel is configured as input, IC2 is mapped on TI2.
10: CC2 channel is configured as input, IC2 is mapped on TI1.
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}$=f_{CK_INT}, N=2
0010: $f_{SAMPLING}$=f_{CK_INT}, N=4
0011: $f_{SAMPLING}$=f_{CK_INT}, N=8
0100: $f_{SAMPLING}$=f_{DTS}/2, N=6
0101: $f_{SAMPLING}$=f_{DTS}/2, N=8
0110: $f_{SAMPLING}$=f_{DTS}/4, N=6
0111: $f_{SAMPLING}$=f_{DTS}/4, N=8
1000: $f_{SAMPLING}$=f_{DTS}/8, N=6
1001: $f_{SAMPLING}$=f_{DTS}/8, N=8
1010: $f_{SAMPLING}$=f_{DTS}/16, N=5
1011: $f_{SAMPLING}$=f_{DTS}/16, N=6
1100: $f_{SAMPLING}$=f_{DTS}/16, N=8
1101: $f_{SAMPLING}$=f_{DTS}/32, N=5
1110: $f_{SAMPLING}$=f_{DTS}/32, N=6
1111: $f_{SAMPLING}$=f_{DTS}/32, N=8

Note: In current silicon revision, f_{DTS} is replaced in the formula by CK_INT when ICxF[3:0]= 1, 2 or 3.

Bits 3:2 **IC1PSC**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).
The prescaler is reset as soon as CC1E=0 (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

16.4.8 TIMx capture/compare mode register 2 (TIMx_CCMR2)

Address offset: 0x1C

Reset value: 0x0000

Refer to the above CCMR1 register description.

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]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 14:12 **OC4M**: Output compare 4 mode

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bit 7 **OC3CE:** Output compare 3 clear enable

Bits 6:4 **OC3M**: Output compare 3 mode

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

Input capture mode

Bits 15:12 **IC4F**: Input capture 4 filter

Bits 11:10 **IC4PSC**: Input capture 4 prescaler

Bits 9:8 **CC4S**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bits 7:4 **IC3F**: Input capture 3 filter

Bits 3:2 **IC3PSC**: Input capture 3 prescaler

Bits 1:0 **CC3S**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

16.4.9 TIMx capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

Bit 15 **CC4NP**: *Capture/Compare 4 output Polarity.*
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: *Capture/Compare 4 output Polarity.*
refer to CC1P description

Bit 12 **CC4E**: *Capture/Compare 4 output enable.*
refer to CC1E description

Bit 11 **CC3NP**: *Capture/Compare 3 output Polarity.*
refer to CC1NP description

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC3P**: *Capture/Compare 3 output Polarity.*
refer to CC1P description

Bit 8 **CC3E**: *Capture/Compare 3 output enable.*
refer to CC1E description

Bit 7 **CC2NP**: *Capture/Compare 2 output Polarity.*
refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: *Capture/Compare 2 output Polarity.*
refer to CC1P description

Bit 4 **CC2E**: *Capture/Compare 2 output enable.*
refer to CC1E description

Bit 3 **CC1NP**: *Capture/Compare 1 output Polarity.*
CC1 channel configured as output:
CC1NP must be kept cleared in this case.
CC1 channel configured as input:
This bit is used in conjunction with CC1P to define TI1FP1/TI2FP1 polarity. refer to CC1P description.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: *Capture/Compare 1 output Polarity.*
CC1 channel configured as output:
0: OC1 active high
1: OC1 active low
CC1 channel configured as input:
CC1NP/CC1P bits select TI1FP1 and TI2FP1 polarity for trigger or capture operations.
00: noninverted/rising edge
Circuit is sensitive to TIxFP1 rising edge (capture, trigger in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger in gated mode, encoder mode).
01: inverted/falling edge
Circuit is sensitive to TIxFP1 falling edge (capture, trigger in reset, external clock or trigger mode), TIxFP1 is inverted (trigger in gated mode, encoder mode).
10: reserved, do not use this configuration.
11: noninverted/both edges
Circuit is sensitive to both TIxFP1 rising and falling edges (capture, trigger in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger in gated mode). This configuration must not be used for encoder mode.

Bit 0 **CC1E**: *Capture/Compare 1 output enable.*
CC1 channel configured as output:
0: Off - OC1 is not active
1: On - OC1 signal is output on the corresponding output pin
CC1 channel configured as input:
This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
0: Capture disabled
1: Capture enabled

Table 67. Output control bit for standard OCx channels

CCxE bit	OCx output state
0	Output Disabled (OCx=0, OCx_EN=0)
1	OCx=OCxREF + Polarity, OCx_EN=1

Note: The state of the external IO pins connected to the standard OCx channels depends on the OCx channel state and the GPIO and AFIO registers.

16.4.10 TIMx counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

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

Bits 15:0 **CNT[15:0]**: Counter value

16.4.11 TIMx prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event.

16.4.12 TIMx auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0x0000

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

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 16.3.1: Time-base unit on page 341* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

16.4.13 TIMx capture/compare register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

16.4.14 TIMx capture/compare register 2 (TIMx_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC2 output.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (IC2).

16.4.15 TIMx capture/compare register 3 (TIMx_CCR3)

Address offset: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR3[15:0]**: Capture/Compare value

If channel CC3 is configured as output:

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC3 output.

If channel CC3is configured as input:

CCR3 is the counter value transferred by the last input capture 3 event (IC3).

16.4.16 TIMx capture/compare register 4 (TIMx_CCR4)

Address offset: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR4[15:0]**: Capture/Compare value

1. if CC4 channel is configured as output (CC4S bits):
   CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).
   It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.
   The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC4 output.
2. if CC4 channel is configured as input (CC4S bits in TIMx_CCMR4 register):
   CCR4 is the counter value transferred by the last input capture 4 event (IC4).

16.4.17 TIMx DMA control register (TIMx_DCR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DBL[4:0]					Reserved			DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the number of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).
00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

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

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.
Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

Example: Let us consider the following transfer: DBL = 7 transfers & DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

16.4.18 TIMx DMA address for full transfer (TIMx_DMAR)

Address offset: 0x4C

Reset value: 0x0000

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

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

Example of how to use the DMA burst feature

In this example the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

Note: This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

16.4.19 TIM2 option register (TIM2_OR)

Address offset: 0x50

Reset value: 0x0000

This register is available on high and medium+ density devices.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															ITR1_RMP
															rw

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **ITR1_RMP:** Timer 2 Internal trigger 1 remap

Set and cleared by software.

0: TIM2 ITR1 input is connected to TIM10 OC

1: TIM2 ITR1 input is connected to TIM5 TGO

16.4.20 TIM3 option register (TIM3_OR)

Address offset: 0x50

Reset value: 0x0000

This register is available on high and medium+ density devices.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															ITR2_RMP
															rw

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **ITR2_RMP:** Timer 2 Internal trigger 2 remap

Set and cleared by software.

0: TIM3 ITR2 input is connected to TIM11 OC

1: TIM3 ITR2 input is connected to TIM5 TGO

16.4.21 TIMx register map

TIMx registers are mapped as described in the table below:

Table 68. TIMx register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Reserved																						CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x04	**TIMx_CR2**	Reserved																								TI1S	MMS[2:0]			CCDS	Reserved		
	Reset value																									0	0	0	0	0			
0x08	**TIMx_SMCR**	Reserved																ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Reserved	SMS[2:0]		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0C	**TIMx_DIER**	Reserved																	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Reserved	TIE	Reserved	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0	0	0	0	0	0	0		0		0	0	0	0	0
0x10	**TIMx_SR**	Reserved																			CC4OF	CC3OF	CC2OF	CC1OF	Reserved		TIF	Reserved	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																				0	0	0	0			0		0	0	0	0	0
0x14	**TIMx_EGR**	Reserved																									TG	Reserved	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																										0		0	0	0	0	0
0x18	**TIMx_CCMR1** Output Compare mode	Reserved																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
	TIMx_CCMR1 Input Capture mode	Reserved																IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**TIMx_CCMR2** Output Compare mode	Reserved																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
	TIMx_CCMR2 Input Capture mode	Reserved																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

Table 68. TIMx register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20	TIMx_CCER	Reserved																CC4NP	Reserved	CC4P	CC4E	CC3NP	Reserved	CC3P	CC3E	CC2NP	Reserved	CC2P	CC2E	CC1NP	Reserved	CC1P	CC1E
	Reset value																	0		0	0	0		0	0	0		0	0	0		0	0
0x24	TIMx_CNT	CNT[32:16] (TIM5 only, reserved on the other timers)																CNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	ARR[32:16] (TIM5 only, reserved on the other timers)																ARR[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
0x30	Reserved																																
0x34	TIMx_CCR1	CCR1[32:16] (TIM5 only, reserved on the other timers)																CCR1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	TIMx_CCR2	CCR4[32:16] (TIM5 only, reserved on the other timers)																CCR2[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	TIMx_CCR3	CCR4[32:16] (TIM5 only, reserved on the other timers)																CCR3[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	TIMx_CCR4	CCR4[32:16] (TIM5 only, reserved on the other timers)																CCR4[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	Reserved																																
0x48	TIMx_DCR	Reserved																			DBL[4:0]					Reserved			DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	TIMx_DMAR	Reserved																DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	TIM2_OR	Reserved																															ITR1_RMP
	Reset value																																0

Table 68. TIMx register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x50	TIM3_OR	Reserved																															ITR2_RMP
	Reset value																																0

Refer to *Table 2 on page 42* for the register boundary addresses.

17 General-purpose timers (TIM9/10/11)

17.1 TIM9/10/11 introduction

The TIM9/10/11 general-purpose timers consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM).

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 TIM9/10/11 timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 17.4.12*.

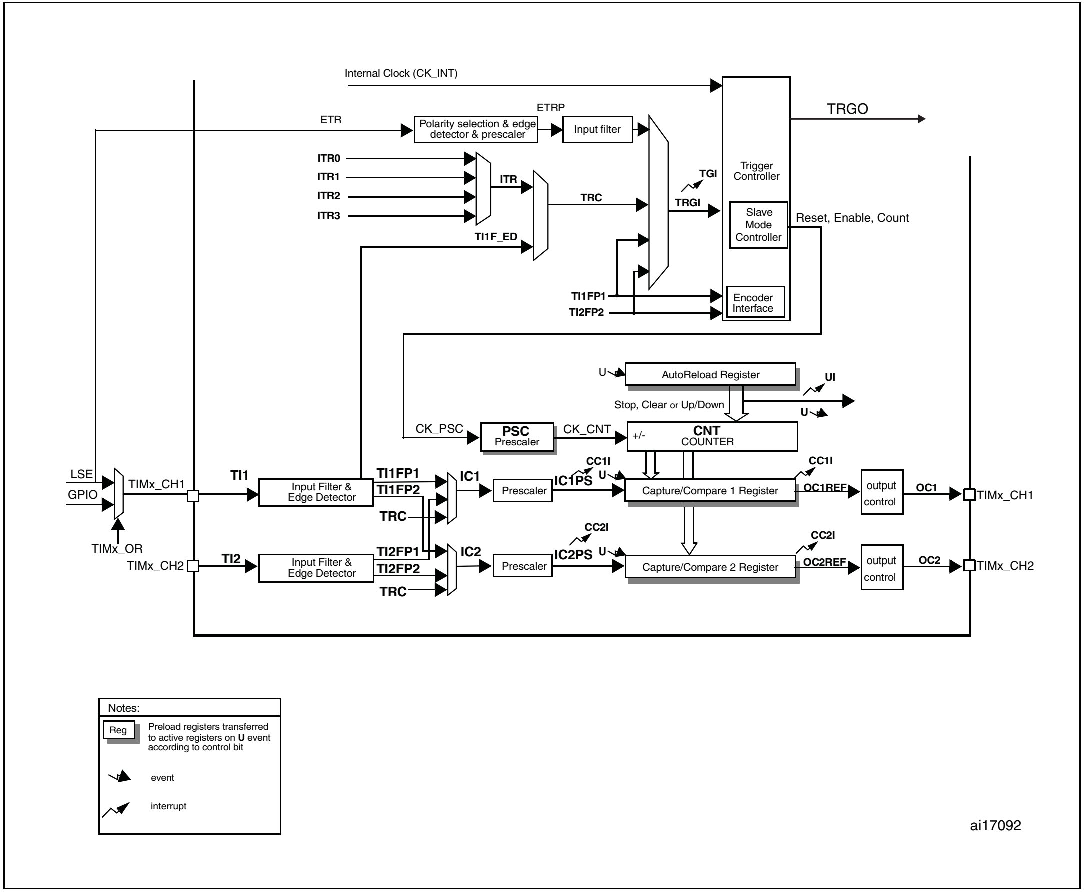
17.2 TIM9/10/11 main features

17.2.1 TIM9 main features

The features of the general-purpose timer include:

- 16-bit auto-reload upcounter (in low and medium density devices)
- 16-bit up, down, up/down auto-reload counter (in high and medium+ density devices)
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- Up to 2 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Interrupt generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal trigger)
  - Trigger event (counter start, stop, initialization or count by internal trigger)
  - Input capture
  - Output compare
- Trigger input for external clock or cycle-by-cycle current management

Figure 128. General-purpose timer block diagram (TIM9)



17.3 TIM10/TIM11 main features

The features of general-purpose timers TIM10/TIM11 include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed "on the fly")
- independent channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software)
  - Input capture
  - Output compare

Figure 129. General-purpose timer block diagram (TIM10)

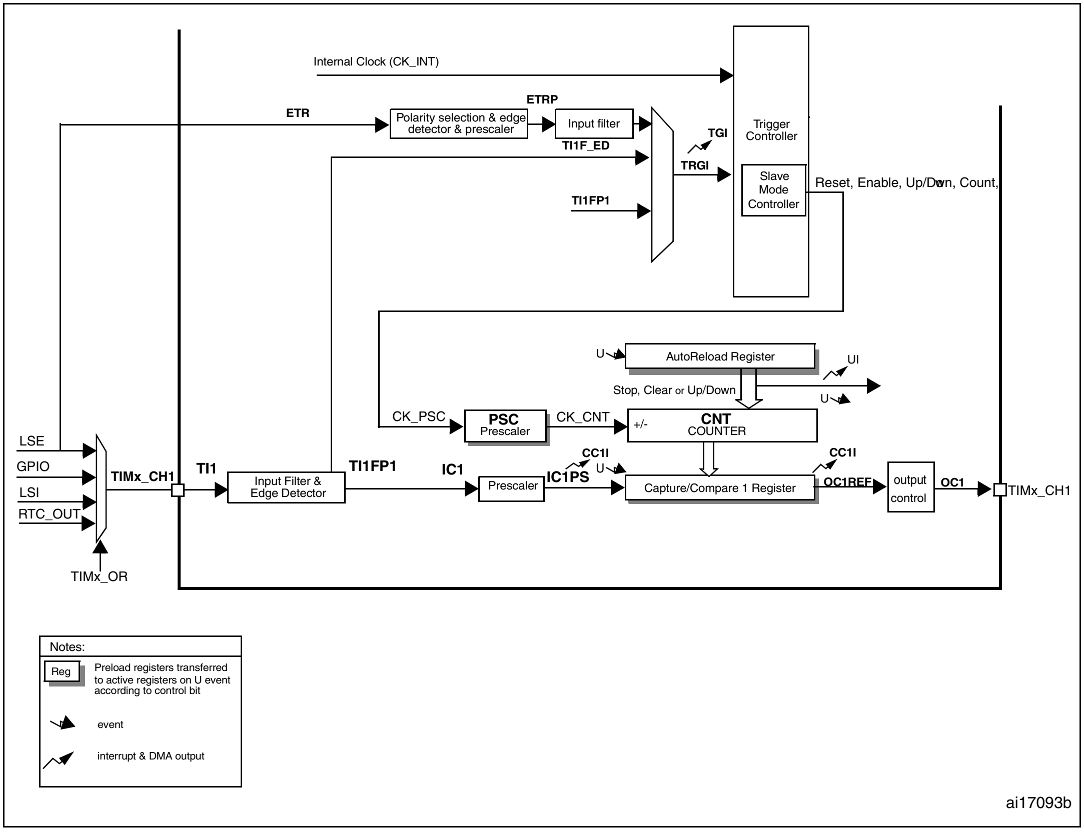
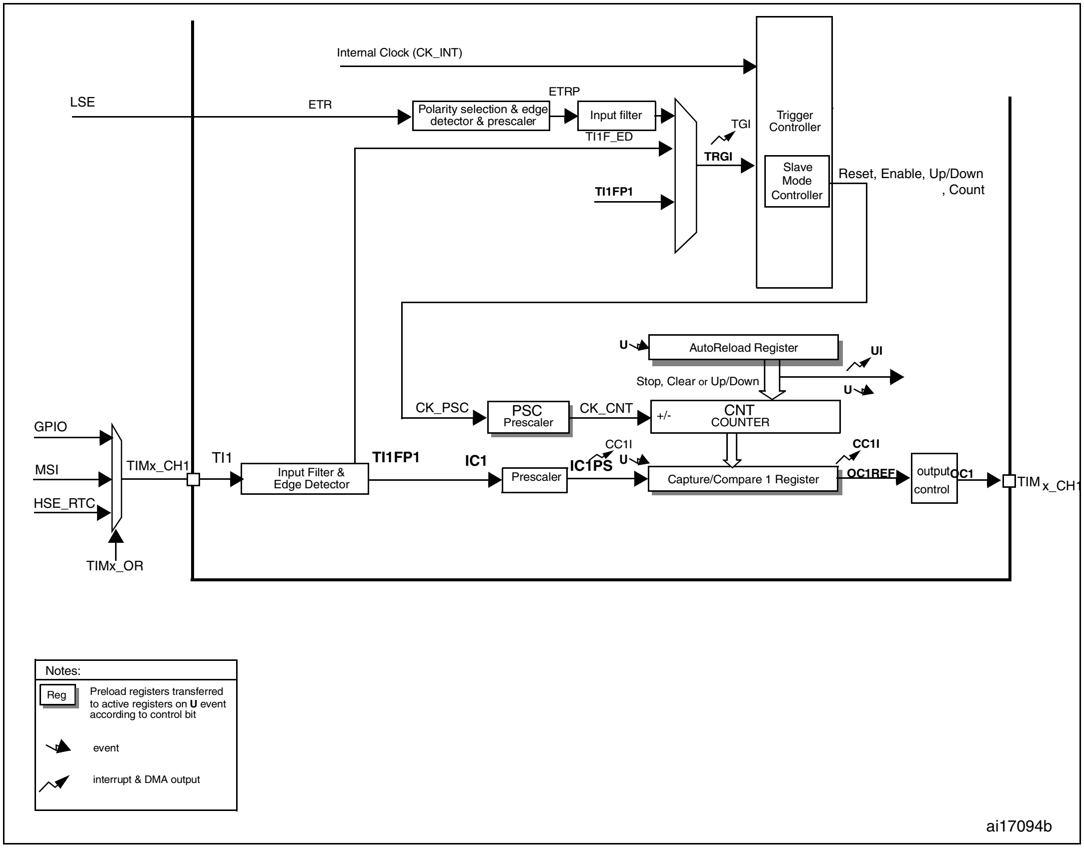


Figure 130. General-purpose timer block diagram (TIM11)



17.4 TIM9/10/11 functional description

17.4.1 Time-base unit

The main block of the timer is a 16-bit counter with its related auto-reload register.

- In high and medium+ density devices: the TIM9 counter can count up, down or both up and down. The TIM10/11 counters operate in an upcounting mode only.
- In low and medium density devices, the TIM9/10/11 counters operate in upcounting mode only.

The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow 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 details for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 131 and *Figure 132* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 131. Counter timing diagram with prescaler division change from 1 to 2

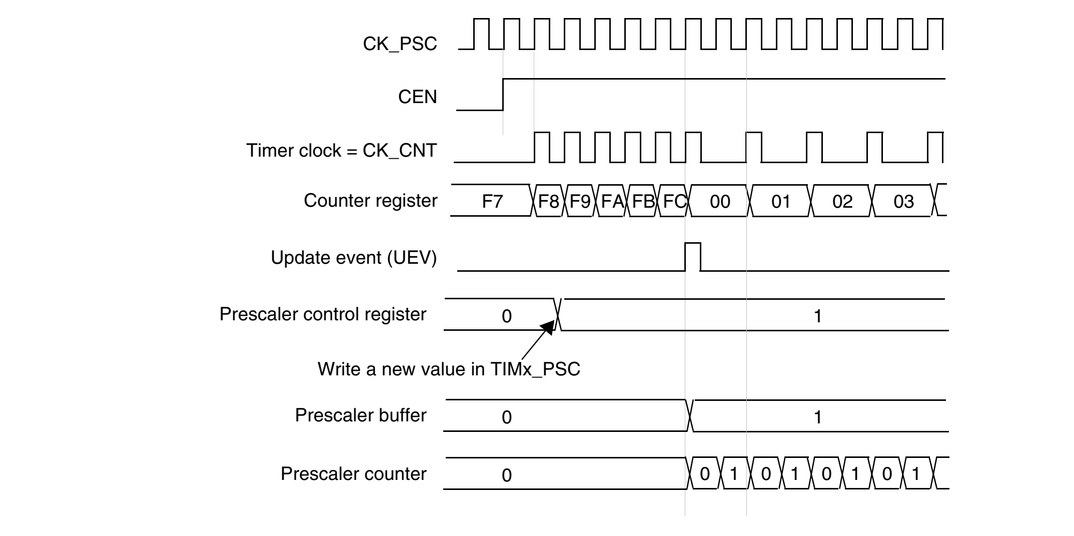
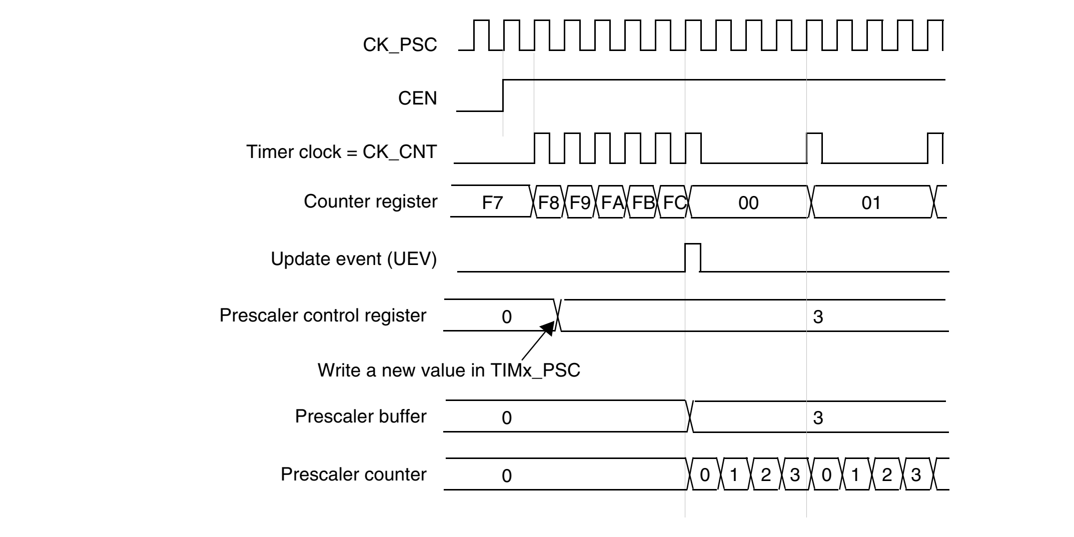


Figure 132. Counter timing diagram with prescaler division change from 1 to 4



17.4.2 Counter modes

In high and medium+ density devices, TIM9 can operate in downcounting and center-aligned modes. Refer to *Section 16.3.2 on page 343* for the description of this modes.

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller on TIM9) 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 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 auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 133. Counter timing diagram, internal clock divided by 1

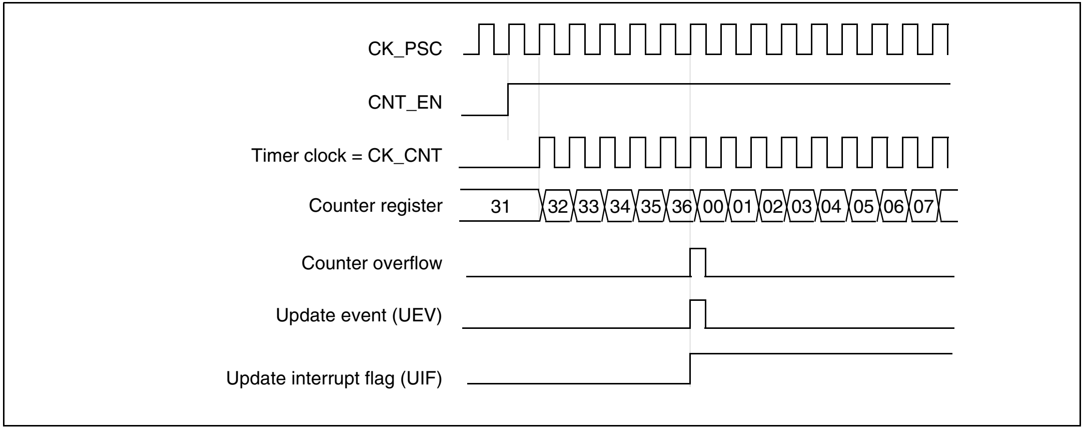


Figure 134. Counter timing diagram, internal clock divided by 2

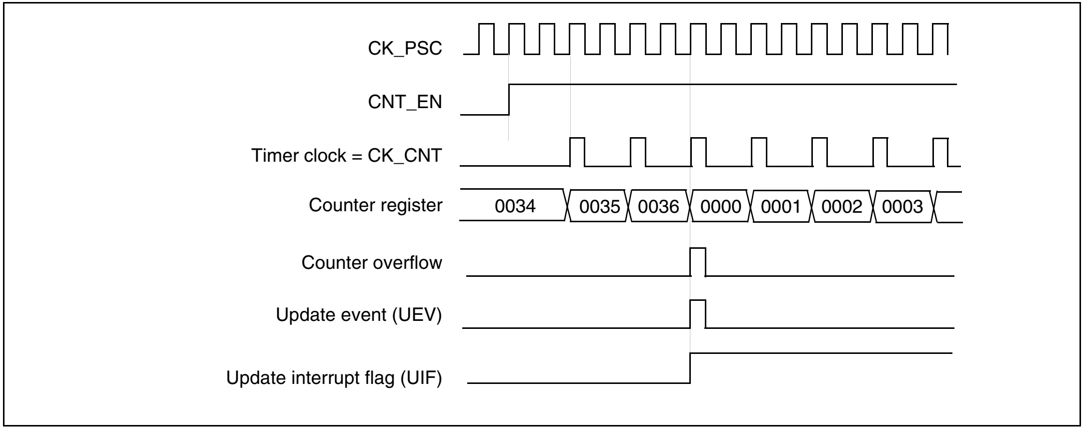


Figure 135. Counter timing diagram, internal clock divided by 4

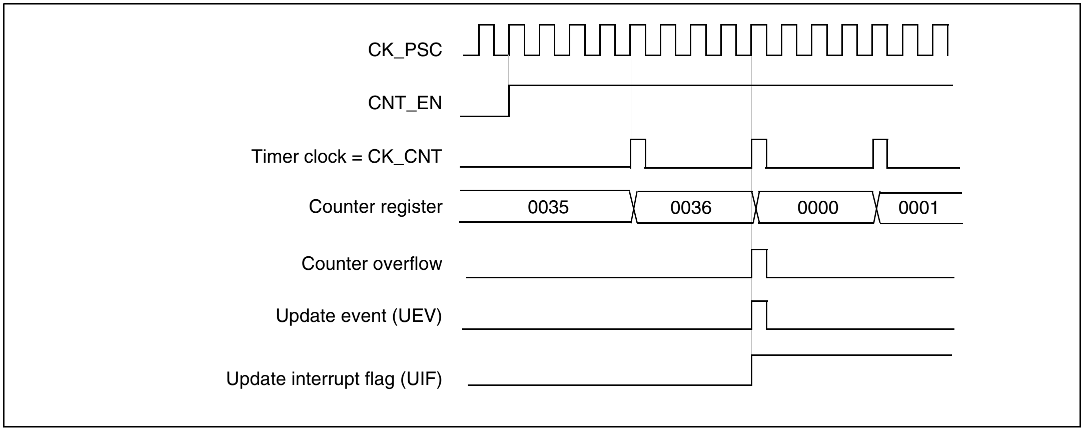


Figure 136. Counter timing diagram, internal clock divided by N

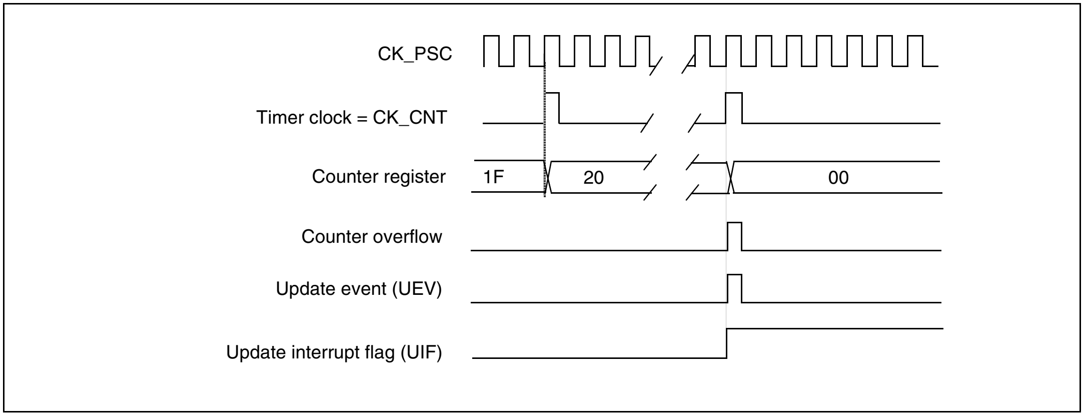


Figure 137. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

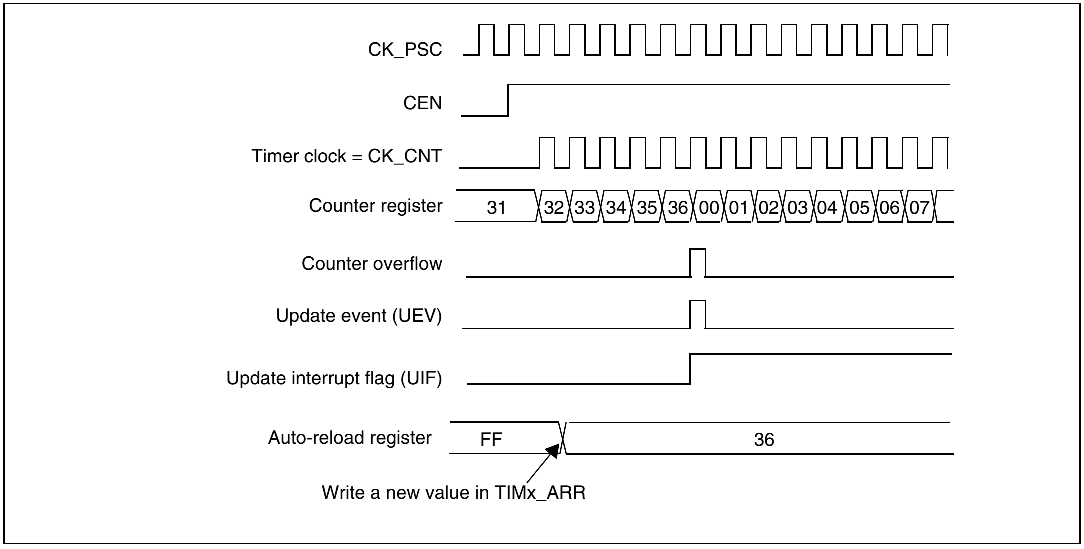
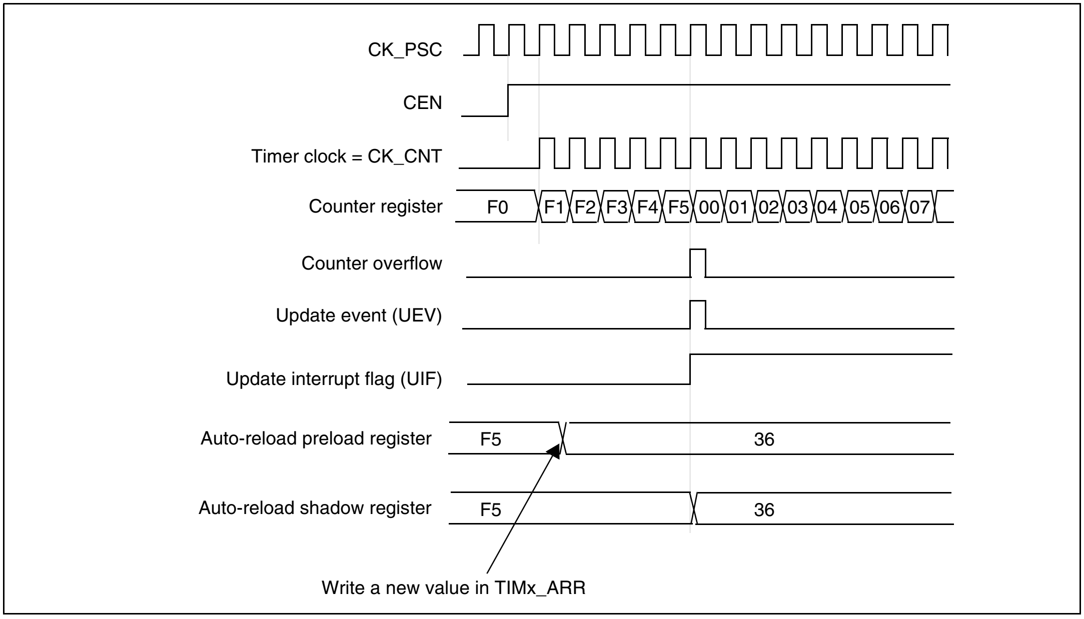


Figure 138. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



17.4.3 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode1 (for **TIM9**): external input pin (TIx)
- External clock mode2: external trigger input (ETR connected internally to LSE)
- Internal trigger inputs (ITRx) (for **TIM9**): connecting the trigger output from another timer. Refer to *Section : Using one timer as prescaler for another* for more details.

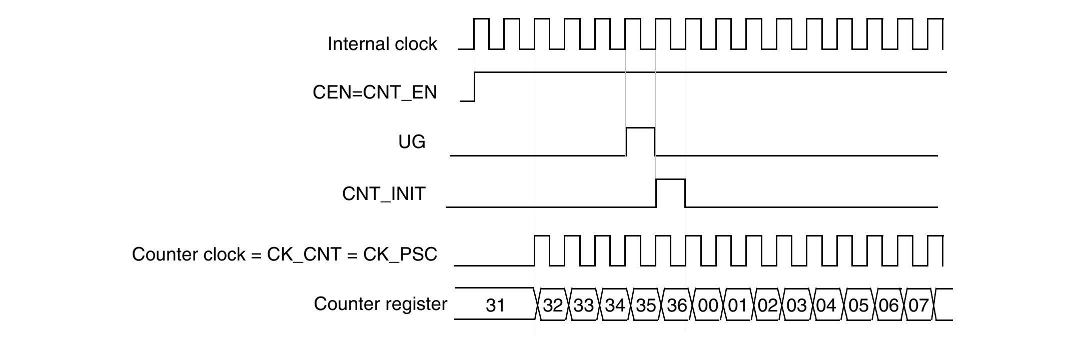
Internal clock source (CK_INT)

The internal clock source is the default clock source for TIM10/TIM11.

For TIM9 and TIM12, the internal clock source is selected when the slave mode controller is disabled (SMS=’000’). The CEN bit in the TIMx_CR1 register and the UG bit in the TIMx_EGR register are then used as control bits and can be changed only by software (except for UG which remains cleared). As soon as the CEN bit is programmed to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 139 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

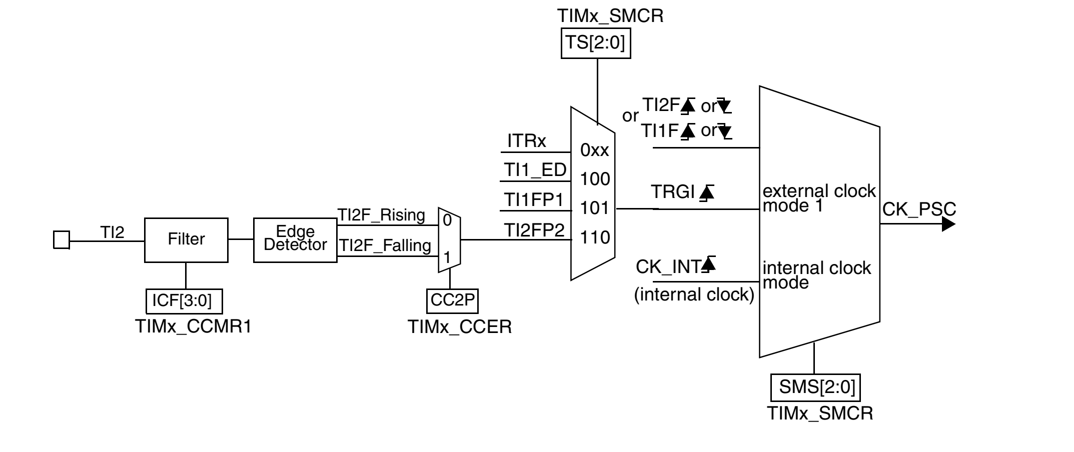
Figure 139. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1(TIM9)

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 140. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

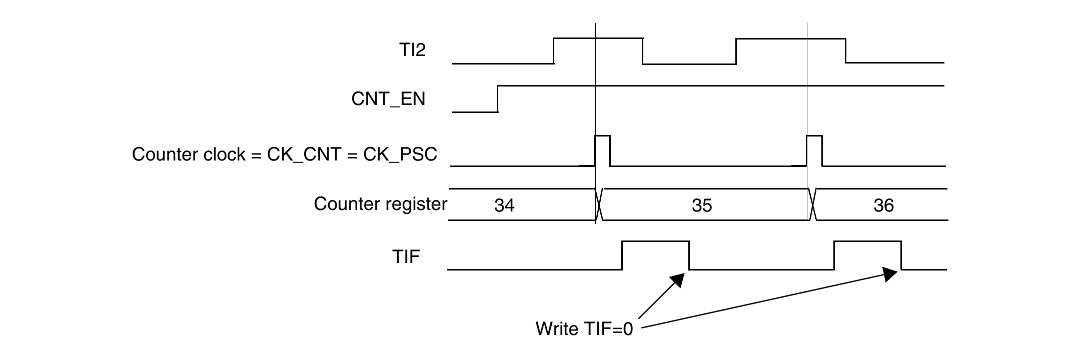
1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = '01' in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F='0000').
3. Select the rising edge polarity by writing CC2P='0' and CC2NP='0' in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS='111' in the TIMx_SMCR register.
5. Select TI2 as the trigger input source by writing TS='110' in the TIMx_SMCR register.
6. Enable the counter by writing CEN='1' in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, so you don't need to configure it.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 141. Control circuit in external clock mode 1



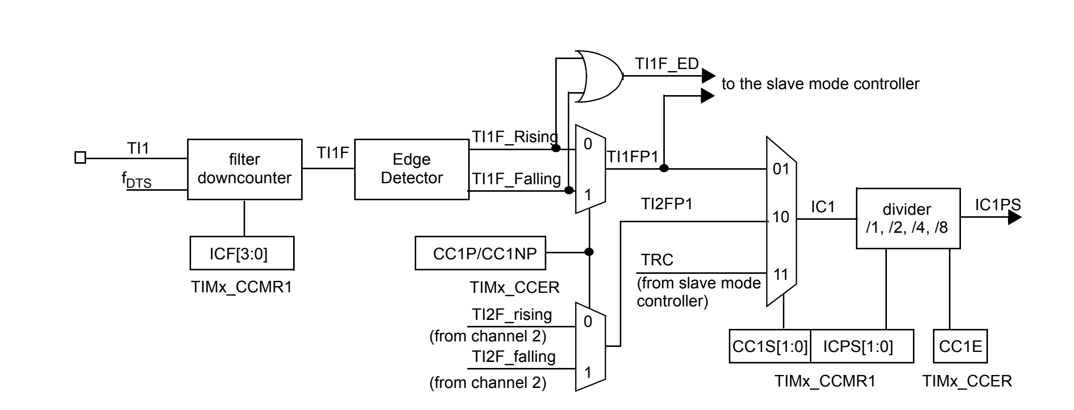
17.4.4 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 142 to *Figure 144* give an overview of one capture/compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 142. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 143. Capture/compare channel 1 main circuit

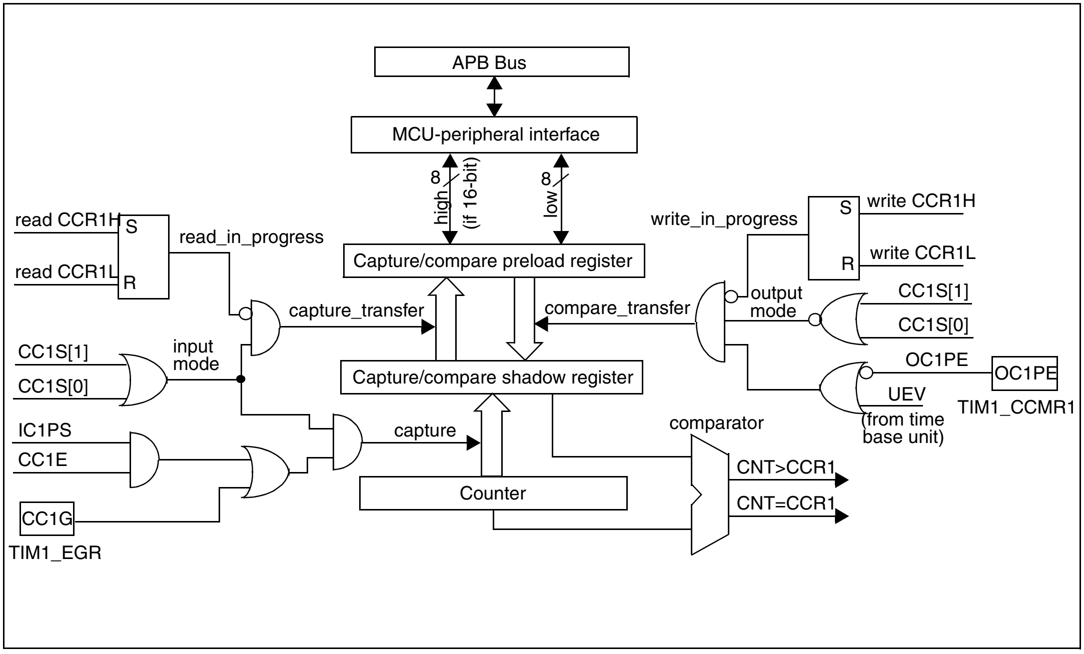
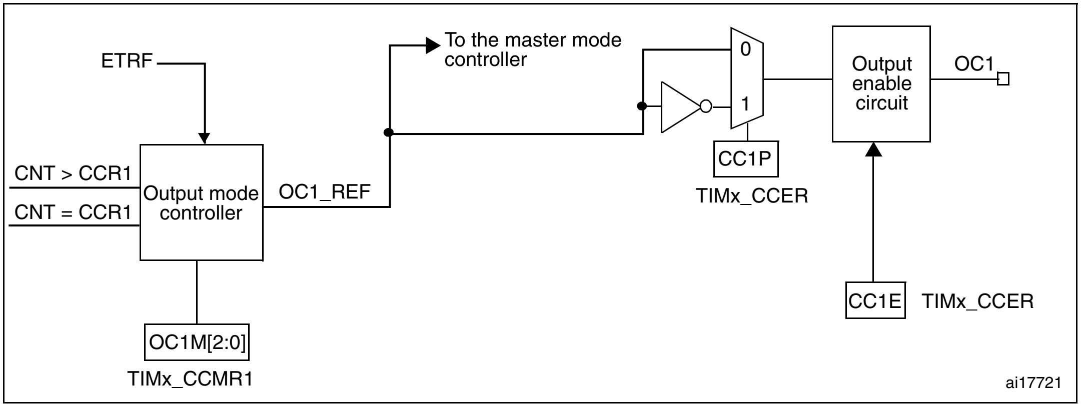


Figure 144. Output stage of capture/compare channel (channel 1)



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.

17.4.5 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to ‘0’ or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when you write it to ‘0’.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to ‘01’ in the TIMx_CCMR1 register. As soon as CC1S becomes different from ‘00’, the channel is configured in input mode and the TIMx_CCR1 register becomes read-only.
2. Program the input filter duration you need with respect to the signal you connect to the timer (by programming the ICxF bits in the TIMx_CCMRx register if the input is one of the TIx inputs). Let’s imagine that, when toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to ‘0011’ in the TIMx_CCMR1 register.
3. Select the edge of the active transition on the TI1 channel by programming CC1P and CC1NP bits to ‘00’ in the TIMx_CCER register (rising edge in this case).
4. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to ‘00’ in the TIMx_CCMR1 register).
5. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
6. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register.

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.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

17.4.6 PWM input mode (only for TIM9)

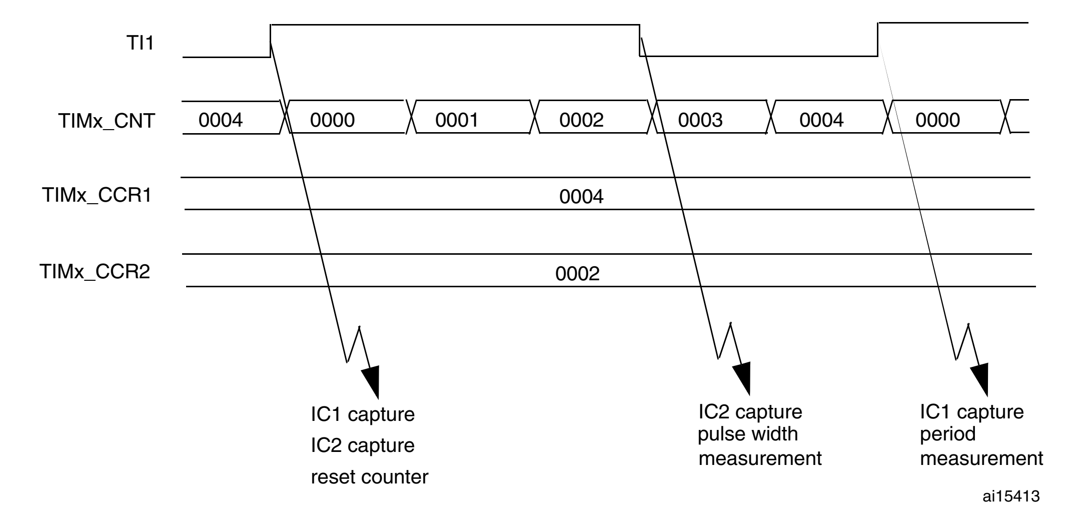
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, you can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the active input for TIMx_CCR1: write the CC1S bits to ‘01’ in the TIMx_CCMR1 register (TI1 selected).
2. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): program the CC1P and CC1NP bits to ‘00’ (active on rising edge).
3. Select the active input for TIMx_CCR2: write the CC2S bits to ‘10’ in the TIMx_CCMR1 register (TI1 selected).
4. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): program the CC2P and CC2NP bits to ‘11’ (active on falling edge).
5. Select the valid trigger input: write the TS bits to ‘101’ in the TIMx_SMCR register (TI1FP1 selected).
6. Configure the slave mode controller in reset mode: write the SMS bits to ‘100’ in the TIMx_SMCR register.
7. Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.

Figure 145. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

17.4.7 Forced output mode

In output mode (CCxS bits = ‘00’ in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, you just need to write ‘101’ in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=’0’ (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to ‘100’ in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt requests can be sent accordingly. This is described in the output compare mode section below.

17.4.8 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

1. 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.
2. Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
3. Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

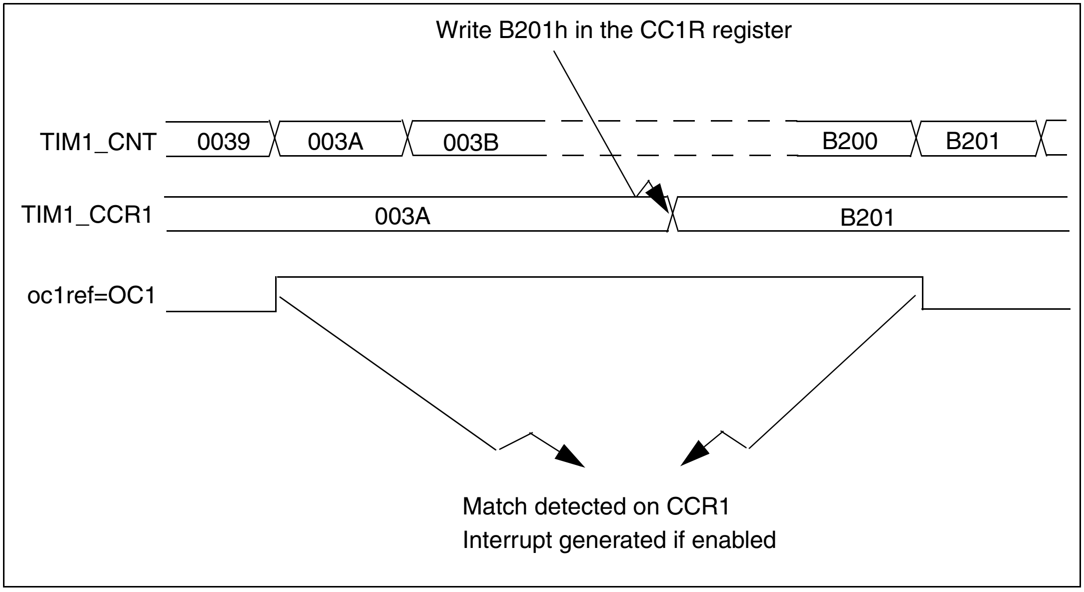
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = ‘011’ to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = ‘0’ to disable preload register
   - Write CCxP = ‘0’ to select active high polarity
   - Write CCxE = ‘1’ to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 146*.

Figure 146. Output compare mode, toggle on OC1.



17.4.9 PWM mode

Pulse Width Modulation mode allows you to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing ‘110’ (PWM mode 1) or ‘111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. You must enable the corresponding preload register by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, you have to initialize all the registers by setting the UG bit in the TIMx_EGR register.

The OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. The 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_CNT ≤TIMx_CCRx.

However, to comply with the ETRF (OCREF can be cleared by an external event through the ETR signal until the next PWM period), the OCREF signal is asserted only:

- When the result of the comparison changes, or
- When the output compare mode (OCxM bits in the TIMx_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM=000) to one of the PWM modes (OCxM=110 or 111).

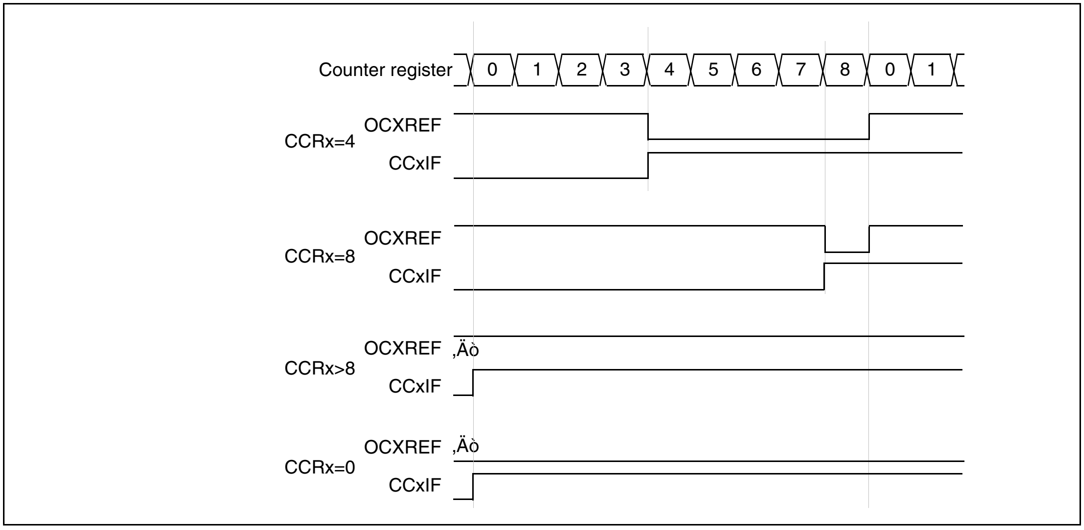
This forces the PWM by software while the timer is running.

The timer is able to generate PWM in edge-aligned mode only since the counter is upcounting.

PWM edge-aligned mode

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at '1'. If the compare value is 0 then OCxRef is held at '0'. *Figure 147* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 147. Edge-aligned PWM waveforms (ARR=8)



17.4.10 One-pulse mode

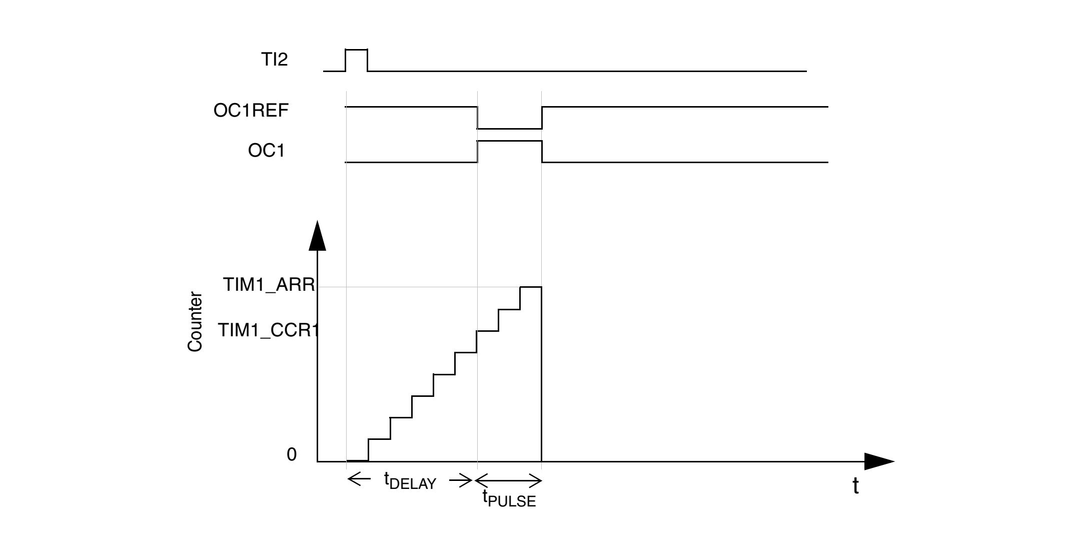
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. You select One-pulse mode 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 as follows:

CNT < CCRx≤ ARR (in particular, 0 < CCRx)

Figure 148. Example of one pulse mode.



For example you may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Use TI2FP2 as trigger 1:

1. Map TI2FP2 to TI2 by writing CC2S=’01’ in the TIMx_CCMR1 register.
2. TI2FP2 must detect a rising edge, write CC2P=’0’ and CC2NP = ‘0’ in the TIMx_CCER register.
3. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=’110’ in the TIMx_SMCR register.
4. TI2FP2 is used to start the counter by writing SMS to ‘110’ in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let’s say you want to build a waveform with a transition from ‘0’ to ‘1’ when a compare match occurs and a transition from ‘1’ to ‘0’ when the counter reaches the auto-reload value. To do this you enable PWM mode 2 by writing OC1M=’111’ in the TIMx_CCMR1 register. You can optionally enable the preload registers by writing OC1PE=’1’ in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case you have to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to ‘0’ in this example.

You only want 1 pulse (Single mode), so you write '1 in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If you want to output a waveform with the minimum delay, you can set the OCxFE bit in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

17.4.11 TIM9 external trigger synchronization

The TIM9 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

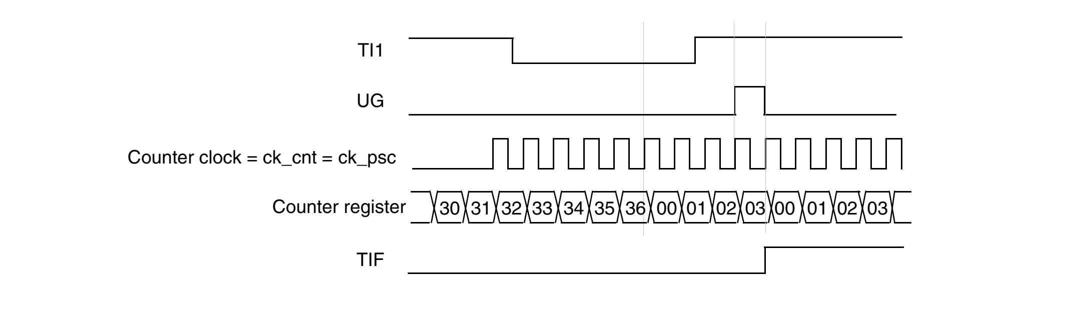
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F='0000'). The capture prescaler is not used for triggering, so you don't need to configure it. The CC1S bits select the input capture source only, CC1S = '01' in the TIMx_CCMR1 register. Program CC1P and CC1NP to '00' in TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS='100' in TIMx_SMCR register. Select TI1 as the input source by writing TS='101' in TIMx_SMCR register.
3. Start the counter by writing CEN='1' in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request can be sent if enabled (depending on the TIE bit in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 149. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

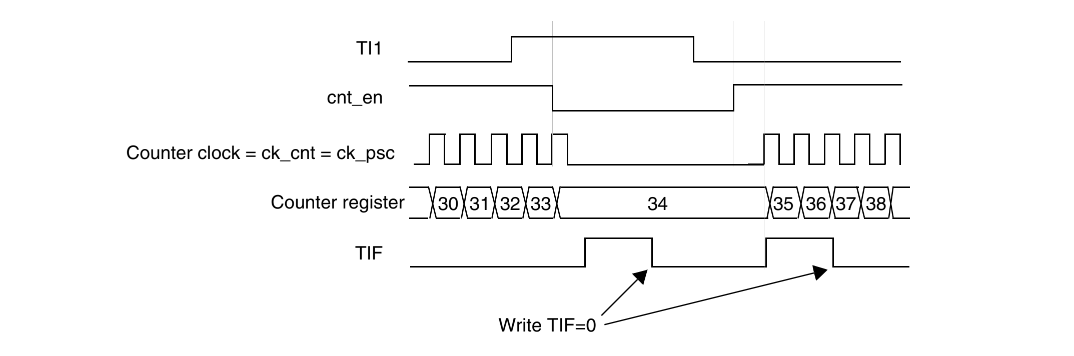
In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we don’t need any filter, so we keep IC1F=’0000’). The capture prescaler is not used for triggering, so you don’t need to configure it. The CC1S bits select the input capture source only, CC1S=’01’ in TIMx_CCMR1 register. Program CC1P=’1’ and CC1NP= ‘0’ in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS=’101’ in TIMx_SMCR register. Select TI1 as the input source by writing TS=’101’ in TIMx_SMCR register.
3. Enable the counter by writing CEN=’1’ in the TIMx_CR1 register (in gated mode, the counter doesn’t start if CEN=’0’, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 150. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

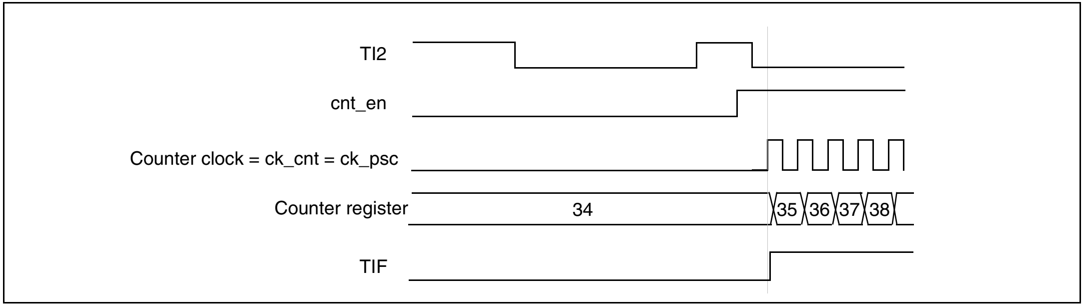
In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we don’t need any filter, so we keep IC2F=’0000’). The capture prescaler is not used for triggering, so you don’t need to configure it. The CC2S bits are configured to select the input capture source only, CC2S=’01’ in TIMx_CCMR1 register. Program CC2P=’1’ and CC2NP=’0’ in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS=’110’ in TIMx_SMCR register. Select TI2 as the input source by writing TS=’110’ in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Figure 151. Control circuit in trigger mode



17.4.12 Timer synchronization (TIM9)

The TIM timers are linked together internally for timer synchronization or chaining. Refer to *Section 16.3.15: Timer synchronization on page 369* for details.

17.4.13 Debug mode

When the microcontroller enters debug mode (Cortex™-M3 core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBG module. For more details, refer to *Section 29.16.2: Debug support for timers, watchdog and I2C*.

17.4.14 Encoder interface mode (only for TIM9)

This section is only applicable for high and medium+ density devices.

Refer to *Section 16.3.12: Encoder interface mode*.

17.5 TIM9 registers

Refer to *Section 1.1 on page 37* for a list of abbreviations used in register descriptions.

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

17.5.1 TIM9 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw			rw	rw	rw	rw	rw

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

Bits 9:8 **CKD**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (TIx),

00: $t_{DTS} = t_{CK\_INT}$
01: $t_{DTS} = 2 \times t_{CK\_INT}$
10: $t_{DTS} = 4 \times t_{CK\_INT}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered.
1: TIMx_ARR register is buffered.

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Note: These bits are available in high and medium+ density devices only.

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.

Note: In low and medium density devices this bit is reserved and must be kept at reset value.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped on the update event
1: Counter stops counting on the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generates an update interrupt if enabled:

- Counter overflow
- Setting the UG bit

1: Only counter overflow generates an update interrupt if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable update event (UEV) generation.

0: UEV enabled. An UEV is generated by one of the following events:

- Counter overflow
- Setting the UG bit

Buffered registers are then loaded with their preload values.

1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable

0: Counter disabled
1: Counter enabled

CEN is cleared automatically in one-pulse mode, when an update event occurs.

17.5.2 TIM9 control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

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

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

Bits 6:4 **MMS**: Master mode selection

These bits are used to select the information to be sent in Master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit in the TIMx_EGR register is used as the trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

001: **Enable** - the Counter enable signal, CNT_EN, is used as the trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between 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 TRGO, except if the master/slave mode is selected (see the MSM bit description in the TIMx_SMCR register).

010: **Update** - The update event is selected as the trigger output (TRGO). For instance a master timer can be used as a prescaler for a slave timer.

011: **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 a compare match occurs. (TRGO).

100: **Compare** - OC1REF signal is used as the trigger output (TRGO).
101: **Compare** - OC2REF signal is used as the trigger output (TRGO).
110: Reserved
111: Reserved

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

17.5.3 TIM9 slave mode control register (TIMx_SMCR)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bit 15 **ETP**: External trigger polarity

This bit selects whether ETR or $\overline{ETR}$ is used for trigger operations
0: ETR is non-inverted, active at high level or rising edge
1: ETR is inverted, active at low level or falling edge

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.
0: External clock mode 2 disabled
1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.
1: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=111).
2: It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 111).
3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.

Bits 13:12 **ETPS**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of CK_INT frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.
00: Prescaler OFF
01: ETRP frequency divided by 2
10: ETRP frequency divided by 4
11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8^{2}
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6^{2}
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bit 7 **MSM:** Master/Slave mode

0: No action
1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful in order to synchronize several timers on a single external event.

Bits 6:4 **TS:** Trigger selection

This bitfield selects the trigger input to be used to synchronize the counter.
000: Internal Trigger 0 (ITR0)
001: Internal Trigger 1 (ITR1)
010: Internal Trigger 2 (ITR2)
011: Internal Trigger 3 (ITR3)
100: TI1 Edge Detector (TI1F_ED)
101: Filtered Timer Input 1 (TI1FP1)
110: Filtered Timer Input 2 (TI2FP2)
111: Reserved.

See *Table 69: TIMx internal trigger connection on page 425* for more details on the meaning of ITRx for each timer.

Note: These bits must be changed only when they are not used (e.g. when SMS='000') to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SMS:** Slave mode selection

When external signals are selected, the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input control register and Control register descriptions.
000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock
001: Encoder mode 1 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level (mode available in high and medium+ density devices only).
010: Encoder mode 2 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level (mode available in high and medium+ density devices only).
011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input (mode available in high and medium+ density devices only).
100: Reset mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers

101: Gated mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Counter starts and stops are both controlled
110: Trigger mode - The counter starts on a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled
111: External clock mode 1 - Rising edges of the selected trigger (TRGI) clock the counter

Note: The Gated mode must not be used if TI1F_ED is selected as the trigger input (TS='100'). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the Gated mode checks the level of the trigger signal.

Table 69. TIMx internal trigger connection

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM9	TIM2	TIM3	TIM10	TIM11

17.5.4 TIM9 Interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TIE	Res			CC2IE	CC1IE	UIE
									rw				rw	rw	rw

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

Bit 6 **TIE**: Trigger interrupt enable

0: Trigger interrupt disabled.
1: Trigger interrupt enabled.

Bit 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable

0: CC2 interrupt disabled.
1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable

0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled.
1: Update interrupt enabled.

17.5.5 TIM9 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CC2OF	CC1OF	Reserved		TIF	Reserved			CC2IF	CC1IF	UIF
					rc_w0	rc_w0			rc_w0				rc_w0	rc_w0	rc_w0

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

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

If channel CC1 is configured as output:

This flag is set by hardware when the counter matches the compare value. It is cleared by software.

0: No match.

1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow.

If channel CC1 is configured as input:

This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.

0: No input capture occurred.

1: The counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1 which matches the selected polarity).

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 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.

17.5.6 TIM9 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TG	Reserved			CC2G	CC1G	UG
									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 the TIMx_SR register. Related interrupt can occur if enabled

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/compare 2 generation

refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software 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:
the CC1IF flag is set, the corresponding interrupt is sent if enabled.
If channel CC1 is configured as input:
The current counter value is captured in the TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt 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-initializes the counter and generates an update of the registers. The prescaler counter is also cleared and the prescaler ratio is not affected. The counter is cleared.

17.5.7 TIM9 capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits in this register have different functions in input and output modes. For a given bit, OCxx describes its function when the channel is configured in output mode, ICxx describes its function when the channel is configured in input mode. So you must take care that the same bit can have different meanings for the input stage and the output stage.

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]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bit 15 **OC2CE:** Output compare 2 clear enable

Bits 14:12 **OC2M[2:0]**: 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, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register

Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 **OC1CE:** Output compare 1 clear enable

OC1CE: Output Compare 1 Clear Enable
0: OC1Ref is not affected by the ETRF input
1: OC1Ref is cleared as soon as a High level is detected on ETRF input

Bits 6:4 **OC1M**: Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas the active levels of OC1 and OC1N depend on the CC1P and CC1NP bits, respectively.
000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).
001: Set channel 1 to active level on match. The OC1REF signal is forced high when the TIMx_CNT counter matches the capture/compare register 1 (TIMx_CCR1).
010: Set channel 1 to inactive level on match. The OC1REF signal is forced low when the TIMx_CNT counter matches the capture/compare register 1 (TIMx_CCR1).
011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1
100: Force inactive level - OC1REF is forced low
101: Force active level - OC1REF is forced high
110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else it is inactive. In downcounting, channel 1 is inactive (OC1REF=‘0) as long as TIMx_CNT>TIMx_CCR1, else it is active (OC1REF=’1’)
111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else it is active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else it is inactive.

Note: In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

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 into account immediately
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded into the active register at each update event

Note: The PWM mode can be used without validating the preload register only in one-pulse mode (OPM bit set in the TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
0: CC1 behaves normally depending on the counter and CCR1 values even when the trigger is ON. The minimum delay to activate the 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 the CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S**: 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, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

Input capture mode

Bits 15:12 **IC2F**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S**: 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, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F**: Input capture 1 filter

This bitfield defines the frequency used to sample the TI1 input and the length of the digital filter applied to TI1. 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, sampling is done at f_{DTS}
0001: $f_{SAMPLING}$=f_{CK_INT}, N=2
0010: $f_{SAMPLING}$=f_{CK_INT}, N=4
0011: $f_{SAMPLING}$=f_{CK_INT}, N=8
0100: $f_{SAMPLING}$=f_{DTS}/2, N=6
0101: $f_{SAMPLING}$=f_{DTS}/2, N=8
0110: $f_{SAMPLING}$=f_{DTS}/4, N=6
0111: $f_{SAMPLING}$=f_{DTS}/4, N=8
1000: $f_{SAMPLING}$=f_{DTS}/8, N=6
1001: $f_{SAMPLING}$=f_{DTS}/8, N=8
1010: $f_{SAMPLING}$=f_{DTS}/16, N=5
1011: $f_{SAMPLING}$=f_{DTS}/16, N=6
1100: $f_{SAMPLING}$=f_{DTS}/16, N=8
1101: $f_{SAMPLING}$=f_{DTS}/32, N=5
1110: $f_{SAMPLING}$=f_{DTS}/32, N=6
1111: $f_{SAMPLING}$=f_{DTS}/32, N=8

Note: In the current silicon revision, f_{DTS} is replaced in the formula by CK_INT when ICxF[3:0]= 1, 2 or 3.

Bits 3:2 **IC1PSC**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on the CC1 input (IC1).
The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S**: 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, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

17.5.8 TIM9 capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
								rw		rw	rw	rw		rw	rw

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

Bit 7 **CC2NP**: Capture/Compare 2 output Polarity

refer to CC1NP description

Bits 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output Polarity

refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable

refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity

CC1 channel configured as output: CC1NP must be kept cleared

CC1 channel configured as input: CC1NP is used in conjunction with CC1P to define TI1FP1/TI2FP1 polarity (refer to CC1P description).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.

CC1 channel configured as output:
0: OC1 active high.
1: OC1 active low.
CC1 channel configured as input:
CC1NP/CC1P bits select TI1FP1 and TI2FP1 polarity for trigger or capture operations.

00: noninverted/rising edge

Circuit is sensitive to TIxFP1 rising edge (capture, trigger in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger in gated mode, encoder mode).

01: inverted/falling edge

Circuit is sensitive to TIxFP1 falling edge (capture, trigger in reset, external clock or trigger mode), TIxFP1 is inverted (trigger in gated mode, encoder mode).

10: reserved, do not use this configuration.

Note: 11: noninverted/both edges
Circuit is sensitive to both TIxFP1 rising and falling edges (capture, trigger in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger in gated mode). This configuration must not be used for encoder mode.

Bit 0 **CC1E**: Capture/Compare 1 output enable.

CC1 channel configured as output:
0: Off - OC1 is not active.
1: On - OC1 signal is output on the corresponding output pin.
CC1 channel configured as input:
This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
0: Capture disabled.
1: Capture enabled.

Table 70. Output control bit for standard OCx channels

CCxE bit	OCx output state
0	Output disabled (OCx='0', OCx_EN='0')
1	OCx=OCxREF + Polarity, OCx_EN='1'

Note: The states of the external I/O pins connected to the standard OCx channels depend on the state of the OCx channel and on the GPIO registers.

17.5.9 TIM9 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bits 15:0 **CNT[15:0]**: Counter value

17.5.10 TIM9 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).

PSC contains the value to be loaded into the active prescaler register at each update event.

17.5.11 TIM9 auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0x0000 0000

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

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded into the actual auto-reload register.

Refer to the *Section 17.4.1: Time-base unit on page 403* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

17.5.12 TIM9 capture/compare register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:
CCR1 is the value to be loaded into the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (OC1PE bit). Else the preload value is copied into the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the TIMx_CNT counter and signaled on the OC1 output.
If channel CC1is configured as input:
CCR1 is the counter value transferred by the last input capture 1 event (IC1).

17.5.13 TIM9 capture/compare register 2 (TIMx_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output:
CCR2 is the value to be loaded into the actual capture/compare 2 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (OC2PE bit). Else the preload value is copied into the active capture/compare 2 register when an update event occurs.
The active capture/compare register contains the value to be compared to the TIMx_CNT counter and signalled on the OC2 output.
If channel CC2 is configured as input:
CCR2 is the counter value transferred by the last input capture 2 event (IC2).

17.5.14 TIM9 option register 1 (TIM9_OR)

Address offset: 0x50

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													ITR1_RMP	TI1_RMP[1:0]	
													rw	rw	

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **ITR1_RMP** Timer 9 ITR1 remap

Set and cleared by software.

0: TIM9 ITR1 input is connected to TIM3_TGO signal

1: TIM9 ITR1 input is connected to touch sensing I/O See *Figure 28: Timer mode acquisition logic on page 187*

This bit is available in high and medium+ density devices only.

Bits 1:0 **TI1_RMP[1:0]** Timer 9 input 1 remap

Set and cleared by software.

00: TIM9 Channel1 is connected to GPIO: Refer to Alternate Function mapping

01: LSE internal clock is connected to the TIM9_CH1 input for measurement purposes

10: TIM9 Channel1 is connected to GPIO

11: TIM9 Channel1 is connected to GPIO

17.5.15 TIM9 register map

TIM9 registers are mapped as 16-bit addressable registers as described below:

Table 71. TIM9 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Reserved																						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x04	TIMx_CR2	Reserved																									MMS[2:0]			Reserved			
	Reset value																										0	0	0				
0x08	TIMx_SMCR	Reserved																ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Reserved	SMS[2:0]		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0C	TIMx_DIER	Reserved																									TIE	Reserved			CC2IE	CC1IE	UIE
	Reset value																										0				0	0	0

Table 71. TIM9 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x10	TIMx_SR	Reserved																					CC2OF	CC1OF	Reserved		TIF	Reserved			CC2IF	CC1IF	UIF
	Reset value																						0	0			0				0	0	0
0x14	TIMx_EGR	Reserved																									TG	Reserved			CC2G	CC1G	UG
	Reset value																										0				0	0	0
0x18	TIMx_CCMR1 Output Compare mode	Reserved																	OC2M [2:0]			OC2PE	OC2FE	CC2S [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
	TIMx_CCMR1 Input Capture mode	Reserved																IC2F[3:0]				IC2 PSC [1:0]		CC2S [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
0x1C	Reserved																																
0x20	TIMx_CCER	Reserved																								CC2NP	Reserved	CC2P	CC2E	CC1NP	Reserved	CC1P	CC1E
	Reset value																									0		0	0	0		0	0
0x24	TIMx_CNT	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Reserved																ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	Reserved																																
0x34	TIMx_CCR1	Reserved																CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	TIMx_CCR2	Reserved																CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C to 0x4C	Reserved																																

Table 71. TIM9 register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x50	TIM9_OR	Reserved																													ITR1_RMP	TI1_RMP	
	Reset value																														0	0	0

Refer to *Table 2 on page 42* for the register boundary addresses.

17.6 TIM10/11 registers

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

17.6.1 TIM10/11 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	Reserved				URS	UDIS	CEN
						rw	rw	rw					rw	rw	rw

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

Bits 9:8 **CKD**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (ETR, TIx),

00: $t_{DTS} = t_{CK\_INT}$
01: $t_{DTS} = 2 \times t_{CK\_INT}$
10: $t_{DTS} = 4 \times t_{CK\_INT}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

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

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the update interrupt (UEV) sources.

0: Any of the following events generate an UEV if enabled:

- Counter overflow
- Setting the UG bit

1: Only counter overflow generates an UEV if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable update interrupt (UEV) event generation.

0: UEV enabled. An UEV is generated by one of the following events:

- Counter overflow
- Setting the UG bit.

Buffered registers are then loaded with their preload values.

1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable

0: Counter disabled
1: Counter enabled

17.6.2 TIM10/11 slave mode control register 1 (TIMx_SMCR)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				Reserved							
rw	rw	rw	rw	rw	rw	rw	rw								

Bit 15 **ETP**: External trigger polarity

This bit selects whether ETR or $\overline{\text{ETR}}$ is used for trigger operations

0: ETR is non-inverted, active at high level or rising edge

1: ETR is inverted, active at low level or falling edge

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

1: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=111).

2: It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 111).

3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.

Bits 13:12 **ETPS**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of CK_INT frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

00: Prescaler OFF

01: ETRP frequency divided by 2

10: ETRP frequency divided by 4

11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS} 1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2 1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=41010: $f_{SAMPLING}=f_{DTS}/16$, N=5

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8^{2}1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

0100: $f_{SAMPLING}=f_{DTS}/2$, N=61100: $f_{SAMPLING}=f_{DTS}/16$, N=8

0101: $f_{SAMPLING}=f_{DTS}/2$, N=81101: $f_{SAMPLING}=f_{DTS}/32$, N=5

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6^{2}1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=81111: $f_{SAMPLING}=f_{DTS}/32$, N=8

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

17.6.3 TIM10/11 Interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CC1IE	UIE
														rw	rw

Bits 15:2 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

17.6.4 TIM10/11 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CC1OF	Reserved							CC1IF	UIF
						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

Bits 8:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

If channel CC1 is configured as output:

This flag is set by hardware when the counter matches the compare value. It is cleared by software.

0: No match.

1: The content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the contents of TIMx_CCR1 are greater than the contents of TIMx_ARR, the CC1IF bit goes high on the counter overflow.

If channel CC1 is configured as input:

This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx_CCR1 register.

0: No input capture occurred.

1: The counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1 which matches the selected polarity).

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 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.

17.6.5 TIM10/11 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CC1G	UG
														w	w

Bits 15:2 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 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 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.

17.6.6 TIM10/11 capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So you must take care that the same bit can have a different meaning for the input stage and for the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
								IC1F[3:0]				IC1PSC[1:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

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

Bit 7 **OC1CE:** Output compare 1 clear enable

0: OC1REF is not affected by the ETRF input

1: OC1REF is cleared as soon as a high level is detected on the ETRF input

Bits 6:4 **OC1M**: Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 is derived. OC1REF is active high whereas OC1 active level depends on CC1P bit.

000: Frozen. The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.

001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

011: Toggle - OC1REF toggles when TIMx_CNT = TIMx_CCR1.

100: Force inactive level - OC1REF is forced low.

101: Force active level - OC1REF is forced high.

110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT < TIMx_CCR1 else inactive.

111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT < TIMx_CCR1 else active.

Note: In PWM mode 1 or 2, the OCREF level changes when the result of the comparison changes or when the output compare mode switches from frozen to PWM mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in TIMx_CR1 register). Else the behavior is not guaranteed.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.

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. OC is then 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**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output.
01: CC1 channel is configured as input, IC1 is mapped on TI1.
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

Input capture mode

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

Bits 7:4 **IC1F**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Note: In current silicon revision, f_{DTS} is replaced in the formula by CK_INT when ICxF[3:0]= 1, 2 or 3.

Bits 3:2 **IC1PSC**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).
The prescaler is reset as soon as CC1E=’0’ (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10:
11:

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

17.6.7 TIM10/11 capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CC1NP	Res.	CC1P	CC1E
												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: CC1NP must be kept cleared.

CC1 channel configured as input: CC1NP bit is used in conjunction with CC1P to define TI1FP1 polarity (refer to CC1P description).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.

CC1 channel configured as output:
0: OC1 active high
1: OC1 active low
CC1 channel configured as input:
The CC1P bit selects TI1FP1 and TI2FP1 polarity for trigger or capture operations.
00: noninverted/rising edge
Circuit is sensitive to TI1FP1 rising edge (capture mode), TI1FP1 is not inverted.
01: inverted/falling edge
Circuit is sensitive to TI1FP1 falling edge (capture mode), TI1FP1 is inverted.
10: reserved, do not use this configuration.
11: noninverted/both edges
Circuit is sensitive to both TI1FP1 rising and falling edges (capture mode), TI1FP1 is not inverted.

Bit 0 **CC1E**: Capture/Compare 1 output enable.

CC1 channel configured as output:
0: Off - OC1 is not active
1: On - OC1 signal is output on the corresponding output pin
CC1 channel configured as input:
This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (TIMx_CCR1) or not.
0: Capture disabled
1: Capture enabled

Table 72. Output control bit for standard OCx channels

CCxE bit	OCx output state
0	Output Disabled (OCx=’0’, OCx_EN=’0’)
1	OCx=OCxREF + Polarity, OCx_EN=’1’

Note: The state of the external I/O pins connected to the standard OCx channels depends on the OCx channel state and the GPIO registers.

17.6.8 TIM10/11 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

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

Bits 15:0 **CNT[15:0]**: Counter value

17.6.9 TIM10/11 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event.

17.6.10 TIM10/11 auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0x0000

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

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to *Section 17.4.1: Time-base unit on page 403* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

17.6.11 TIM10/11 capture/compare register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1is configured as input:
CCR1 is the counter value transferred by the last input capture 1 event (IC1).

17.6.12 TIM10 option register 1 (TIM10_OR)

Address offset: 0x50

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												T1_RMP_RI	ETR_RMP	TI1_RMP[1:0]	
												rw	rw	rw	

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **TI1_RMP_RI:** Timer10 Input 1 remap for Routing Interface (RI)
Set and cleared by software.
0: TIM10 Channel1 connection depends on TI1_RMP[1:0] bit values (see below)
1: TIM10 Channel1 is connected to RI (whatever the value on TI1_RMP bits).
This bit is available in high and medium+ density devices only.

Bit 2 **ETR_RMP:** Timer10 ETR remap
Set and cleared by software.
0: TIM10 ETR input is connected to LSE clock
1: TIM10 ETR input is connected to TIM9_TGO
This bit is available in high and medium+ density devices only.

Bits 1:0 **TI1_RMP[1:0]**: TIM10 Input 1 remapping capability
Set and cleared by software.
00: TIM10 Channel1 is connected to GPIO: Refer to Alternate Function mapping
01: LSI internal clock is connected to the TIM10_CH1 input for measurement purposes
10: LSE internal clock is connected to the TIM10_CH1 input for measurement purposes
11: RTC output event is connected to the TIM10_CH1 input for measurement purposes

17.6.13 TIM11 option register 1 (TIM11_OR)

Address offset: 0x50

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												TI1_RMP_RI	ETR_RMP	TI1_RMP[1:0]	
												rw	rw	rw	
Reserved														TI1_RMP[1:0]	
														rw	

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **TI1_RMP_RI:** Timer11 Input 1 remap for Routing Interface (RI)
Set and cleared by software.
0: TIM11 Channel1 connection depends on TI1_RMP[1:0] bit values (see below)
1: TIM11 Channel1 is connected to RI (whatever the value of the TI1_RMP bits).
This bit is available in high and medium+ density devices only.

Bit 2 **ETR_RMP:** Timer11 ETR remap
Set and cleared by software.
0: TIM11 ETR input is connected to LSE clock
1: TIM11 ETR input is connected to TIM9_TGO
This bit is available in high and medium+ density devices only.

Bits 1:0 **TI1_RMP[1:0]**: TIM11 Input 1 remapping capability
Set and cleared by software.
00: TIM11 Channel1 is connected to the GPIO (refer to the Alternate function mapping table in the datasheets).
01: MSI internal clock is connected to the TIM11_CH1 input for measurement purposes
10: HSE_RTC clock (HSE divided by programmable prescaler) is connected to the TIM11_CH1 input for measurement purposes
11: TIM11 Channel1 is connected to GPIO

17.6.14 TIM10/11 register map

TIMx registers are mapped as 16-bit addressable registers as described in the tables below:

Table 73. TIM10/11 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Reserved																						CKD [1:0]		ARPE	Reserved				URS	UDIS	CEN
	Reset value																							0	0	0					0	0	0
0x08	TIMx_SMCR	Reserved																ETP	ECE	ETPS [1:0]		ETF [3:0]				Reserved							
	Reset value																	0	0	0	0	0	0	0	0								
0x0C	TIMx_DIER	Reserved																														CC1IE	UIE
	Reset value																															0	0
0x10	TIMx_SR	Reserved																						CC1OF	Reserved							CC1IF	UIF
	Reset value																							0								0	0
0x14	TIMx_EGR	Reserved																														CC1G	UG
	Reset value																															0	0
0x18	TIMx_CCMR1 Output compare mode	Reserved																								OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
	TIMx_CCMR1 Input capture mode	Reserved																								IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0x1C	Reserved																																
0x20	TIMx_CCER	Reserved																												CC1NP	Reserved	CC1P	CC1E
	Reset value																													0		0	0
0x24	TIMx_CNT	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Reserved																ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	Reserved																																

Table 73. TIM10/11 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
0x34	TIMx_CCR1	Reserved																CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 to 0x4C	Reserved																																
0x50	TIMx_OR	Reserved																												TI1_RMP_RI	ETR_RMP	TI1_RMP	
	Reset value																													0	0	0	0

Refer to *Table 2 on page 42* for the register boundary addresses.

18 Basic timers (TIM6&TIM7)

18.1 TIM6&TIM7 introduction

The basic timers TIM6 and TIM7 consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used as generic timers for time-base generation but they are also specifically used to drive the digital-to-analog converter (DAC). In fact, the timers are internally connected to the DAC and are able to drive it through their trigger outputs.

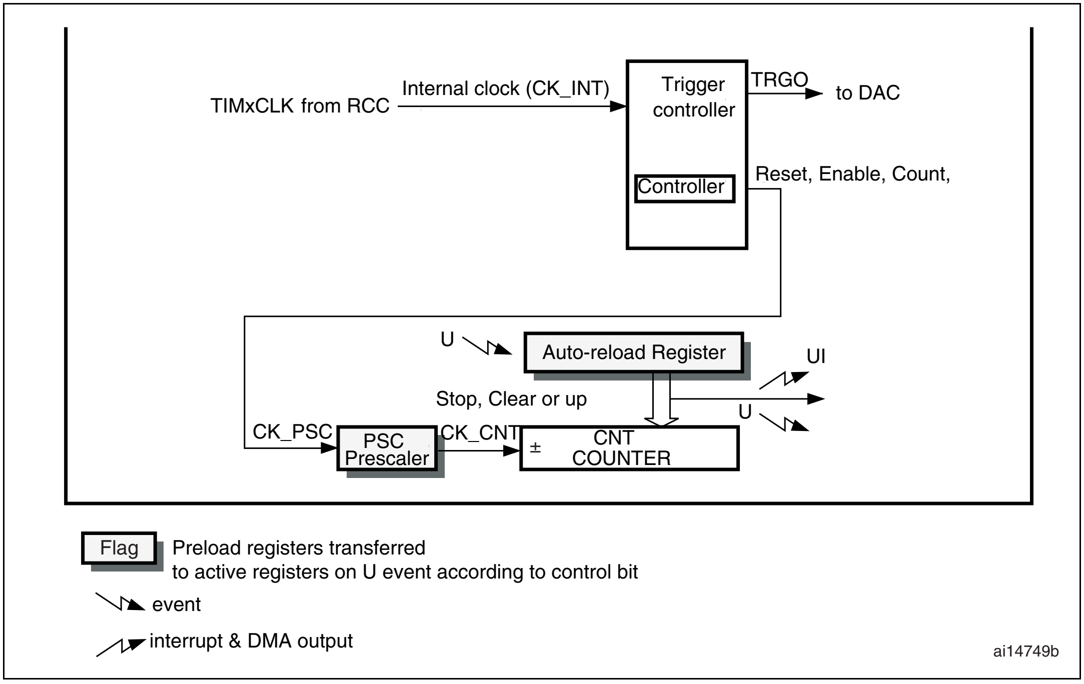
The timers are completely independent, and do not share any resources.

18.2 TIM6&TIM7 main features

Basic timer (TIM6&TIM7) features include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also "on the fly") the counter clock frequency by any factor between 1 and 65536
- Synchronization circuit to trigger the DAC
- Interrupt/DMA generation on the update event: counter overflow

Figure 152. Basic timer block diagram



18.3 TIM6&TIM7 functional description

18.3.1 Time-base unit

The main block of the programmable timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC)
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. The preload register is accessed each time an attempt is made to write or read the auto-reload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the auto-reload preload enable bit (ARPE) in the TIMx_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in the TIMx_CR1 register is set.

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as the TIMx_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 153 and *Figure 154* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 153. Counter timing diagram with prescaler division change from 1 to 2

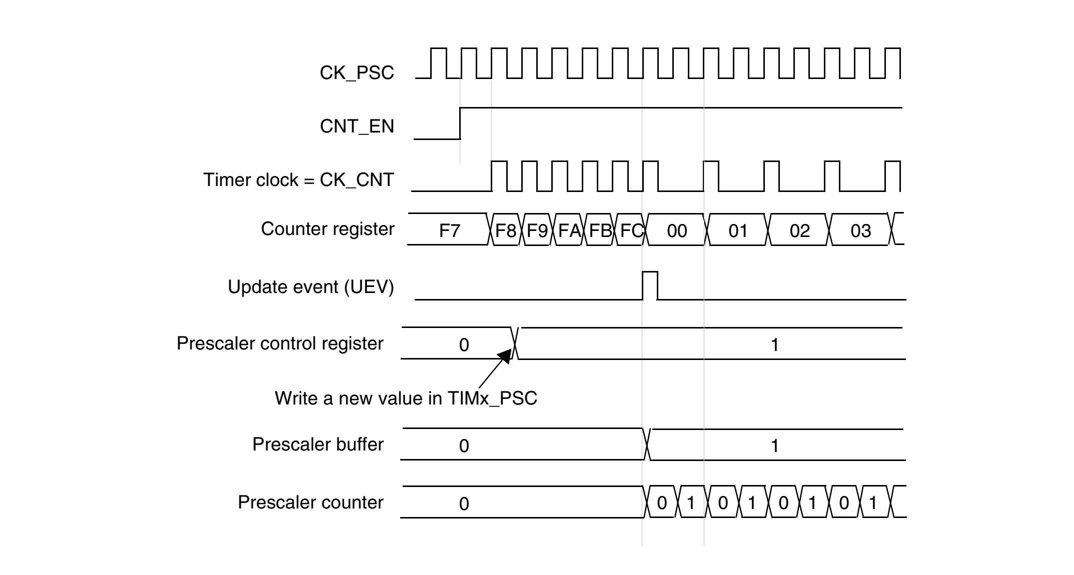
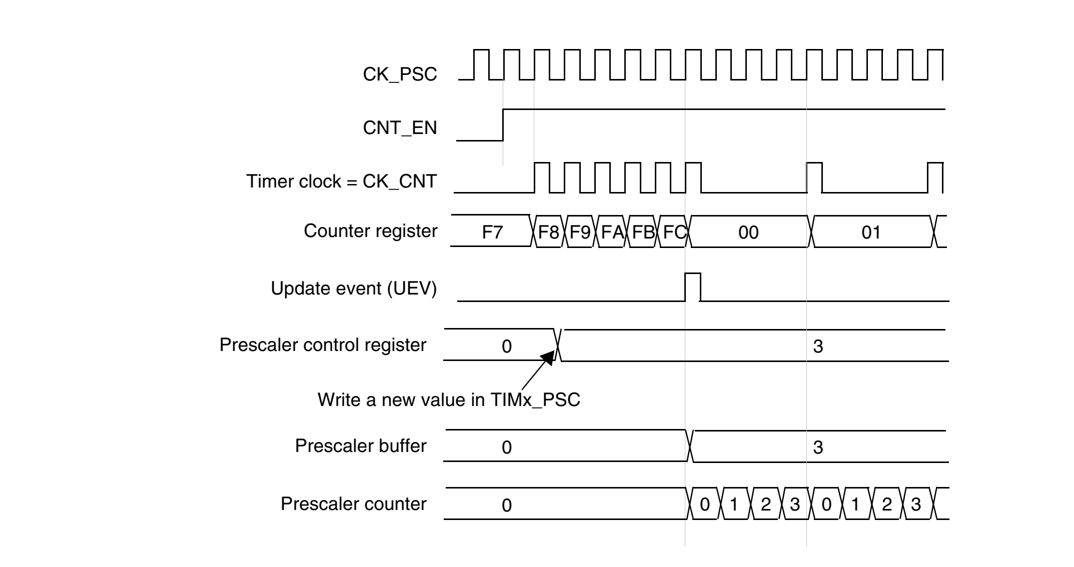


Figure 154. Counter timing diagram with prescaler division change from 1 to 4



18.3.2 Counting mode

The counter counts from 0 to the auto-reload value (contents of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generate at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been written to 0, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIMx_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents of the TIMx_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 155. Counter timing diagram, internal clock divided by 1

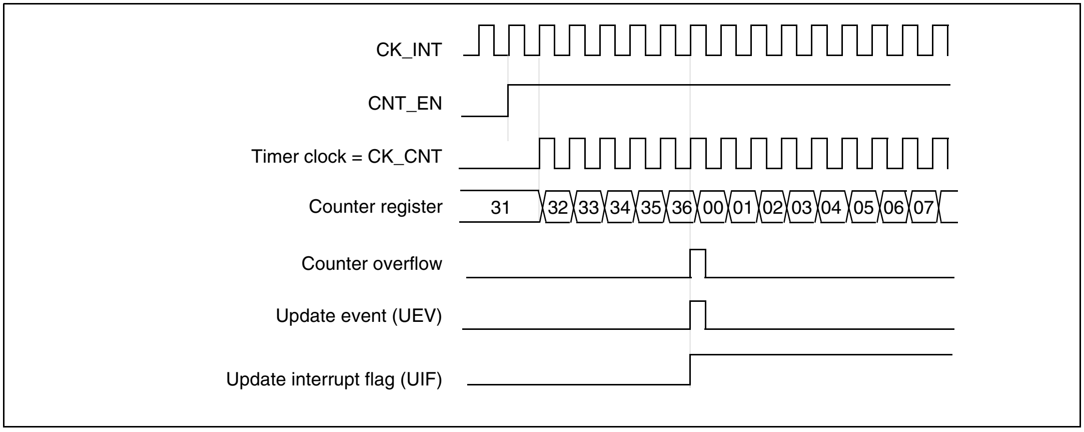


Figure 156. Counter timing diagram, internal clock divided by 2

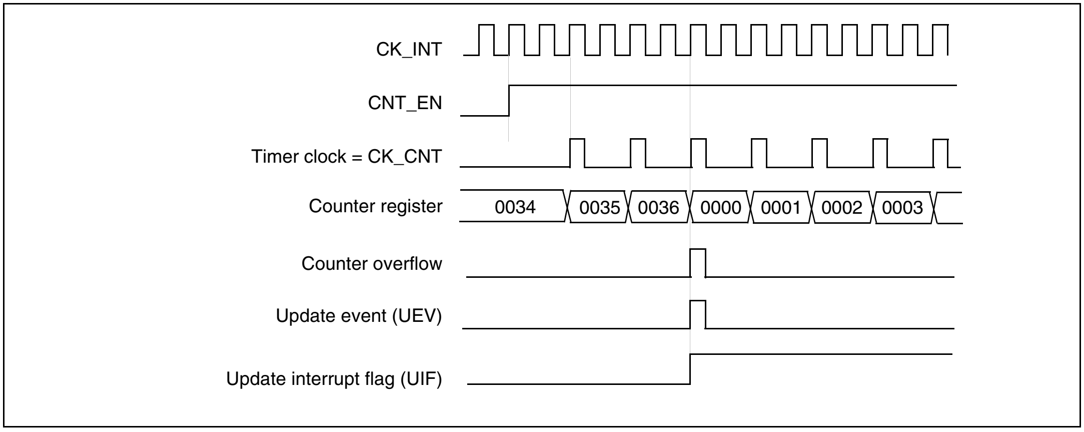


Figure 157. Counter timing diagram, internal clock divided by 4

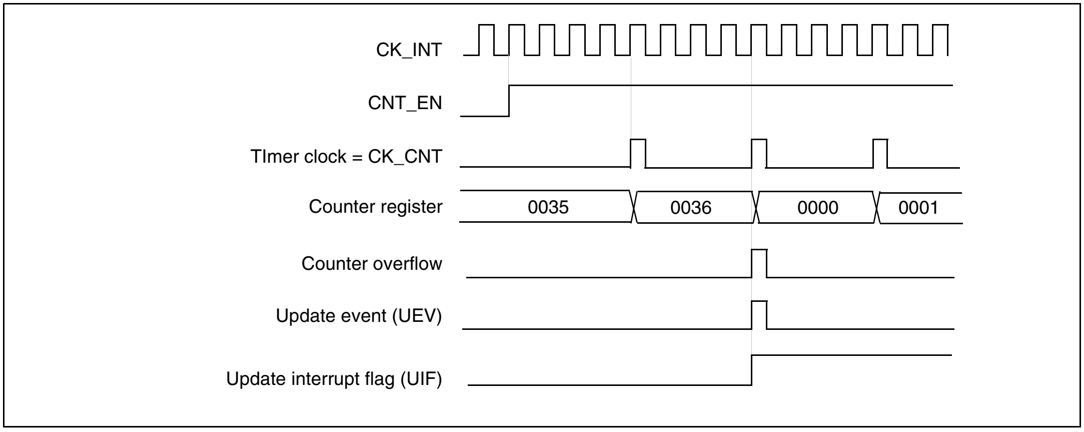


Figure 158. Counter timing diagram, internal clock divided by N

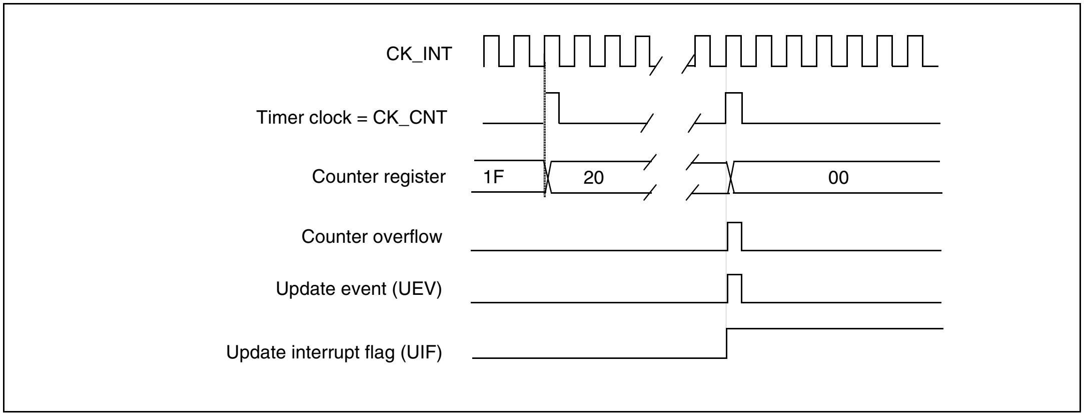


Figure 159. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

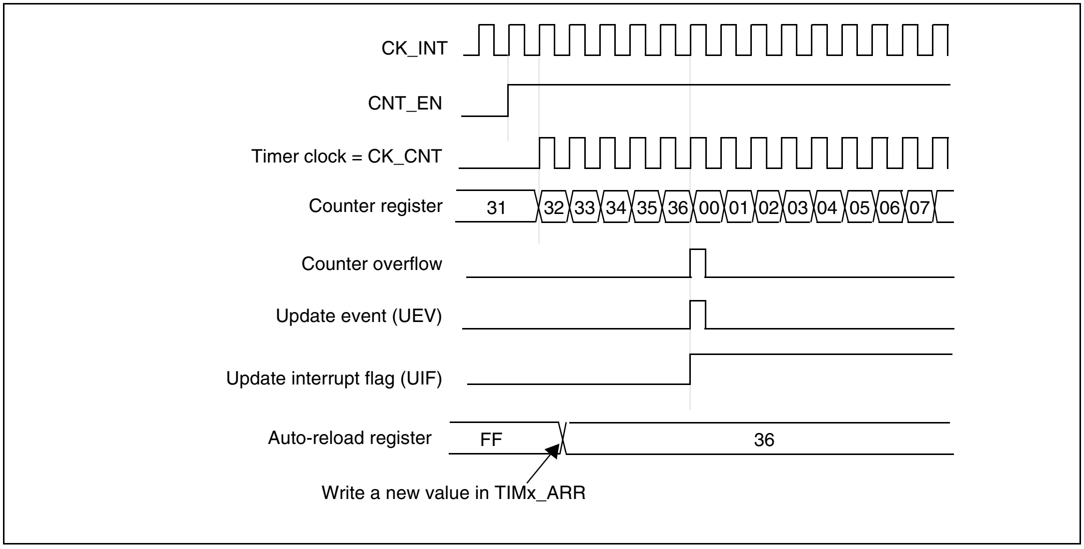
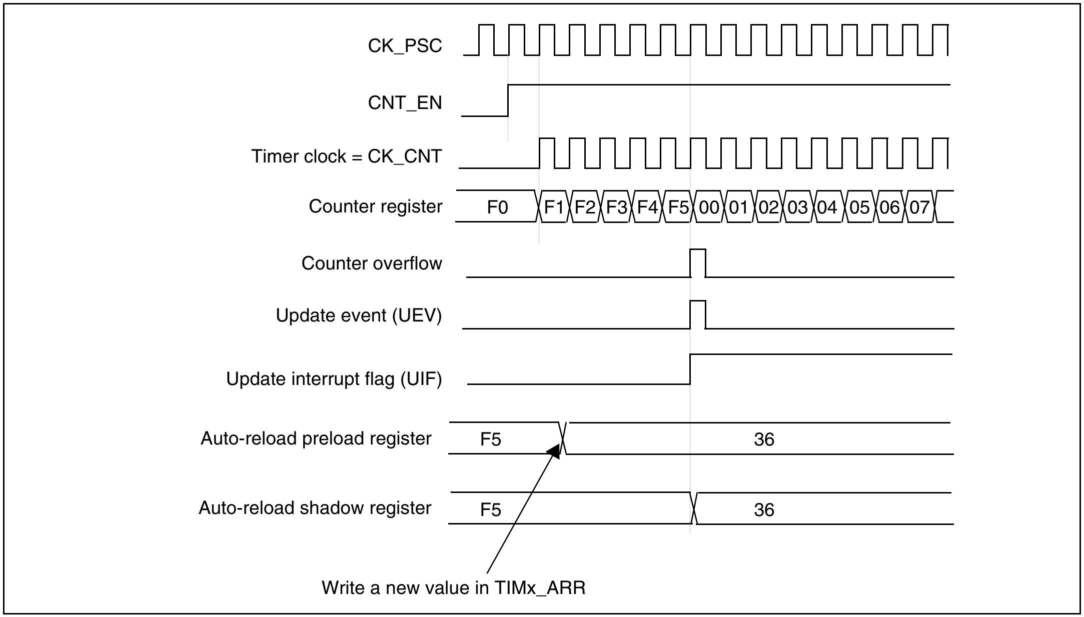


Figure 160. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



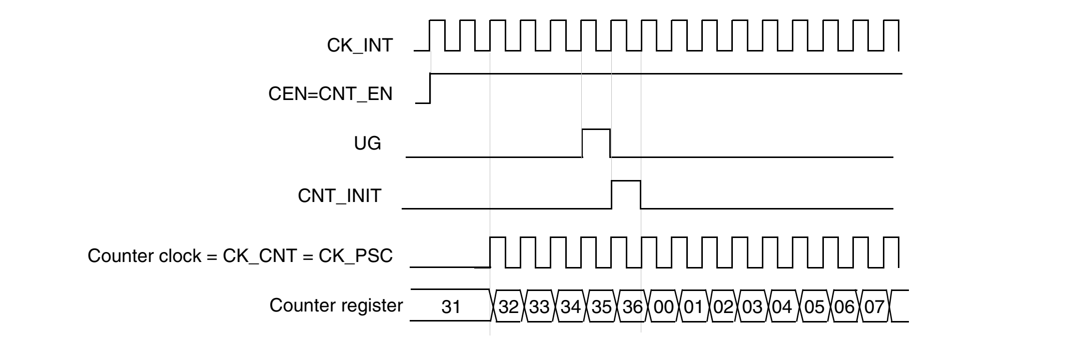
18.3.3 Clock source

The counter clock is provided by the Internal clock (CK_INT) source.

The CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 161 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 161. Control circuit in normal mode, internal clock divided by 1



18.3.4 Debug mode

When the microcontroller enters the debug mode (Cortex™-M3 core - halted), the TIMx counter either continues to work normally or stops, depending on the DBG_TIMx_STOP configuration bit in the DBG module. For more details, refer to *Section 29.16.2: Debug support for timers, watchdog and I2C*.

18.4 TIM6&TIM7 registers

Refer to *Section 1.1 on page 37* for a list of abbreviations used in register descriptions.

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

18.4.1 TIM6&TIM7 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ARPE	Reserved			OPM	URS	UDIS	CEN
								rw				rw	rw	rw	rw

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

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered.
1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generates an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled
1: Counter enabled

Note: Gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

18.4.2 TIM6&TIM7 control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000

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

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

Bits 6:4 **MMS**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit from the TIMx_EGR register is used as a trigger output (TRGO). If reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

001: **Enable** - the Counter enable signal, CNT_EN, is used as a trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode.

When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in the TIMx_SMCR register).

010: **Update** - The update event is selected as a trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

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

18.4.3 TIM6&TIM7 DMA/Interrupt enable register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							UDE	Reserved							UIE
							rw								rw

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

Bit 8 **UDE**: Update DMA request enable

0: Update DMA request disabled.
1: Update DMA request enabled.

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

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled.
1: Update interrupt enabled.

18.4.4 TIM6&TIM7 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															UIF
															rc_w0

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow or underflow and if UDIS = 0 in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in the TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

18.4.5 TIM6&TIM7 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															UG
															w

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action.
1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

18.4.6 TIM6&TIM7 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000

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

Bits 15:0 **CNT[15:0]**: Counter value

18.4.7 TIM6&TIM7 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).

PSC contains the value to be loaded into the active prescaler register at each update event.

18.4.8 TIM6&TIM7 auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0x0000

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

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded into the actual auto-reload register.

Refer to *Section 18.3.1: Time-base unit on page 453* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

18.4.9 TIM6&TIM7 register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

Table 74. TIM6&TIM7 register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Reserved																							ARPE	Reserved			OPM	URS	UDIS	CEN
	Reset value																									0				0	0	0	0
0x04	TIMx_CR2	Reserved																								MMS[2:0]			Reserved			
	Reset value																										0	0	0				
0x08	Reserved																																
0x0C	TIMx_DIER	Reserved																						UDE	Reserved							UIE
	Reset value																								0								0
0x10	TIMx_SR	Reserved																															UIF
	Reset value																																0
0x14	TIMx_EGR	Reserved																															UG
	Reset value																																0
0x18	Reserved																																
0x1C	Reserved																																
0x20	Reserved																																
0x24	TIMx_CNT	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Reserved																ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Table 2 on page 42* for the register boundary addresses.

19 Real-time clock (RTC)

19.1 Introduction

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar, two programmable alarm interrupts, and a periodic programmable wakeup flag with interrupt capability. The RTC also includes an automatic wakeup unit to manage low power modes.

Two 32-bit registers contain the seconds, minutes, hours (12- or 24-hour format), day (day of week), date (day of month), month, and year, expressed in binary coded decimal format (BCD). The sub-seconds value is also available in binary format (in high and medium+ density devices only).

Compensations for 28-, 29- (leap year), 30-, and 31-day months are performed automatically. Daylight saving time compensation can also be performed.

Additional 32-bit registers contain the programmable alarm subseconds (in high and medium+ density devices only), seconds, minutes, hours, day, and date.

A digital calibration feature is available to compensate for any deviation in crystal oscillator accuracy.

After power-on reset, all RTC registers are protected against possible parasitic write accesses.

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).

19.2 RTC main features

The RTC unit main features are the following (see *Figure 162: RTC block diagram (low and medium density devices)*):

- Calendar with subseconds(high density and medium+ devices only), seconds, minutes, hours (12 or 24 format), day (day of week), date (day of month), month, and year.
- Daylight saving compensation programmable by software.
- Two programmable alarms with interrupt function. The alarms can be triggered by any combination of the calendar fields.
- Automatic wakeup unit generating a periodic flag that triggers an automatic wakeup interrupt.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Accurate synchronization with an external clock using the subsecond shift feature (on high and medium+ density devices only).
- Maskable interrupts/events:
  - Alarm A
  - Alarm B
  - Wakeup interrupt
  - Timestamp
  - Tamper detection
- Digital calibration circuit (periodic counter correction)
  - 5 ppm accuracy (on low and medium density devices)
  - 0.95 ppm accuracy (on high and medium+ density devices), obtained in a calibration window of several seconds
- Timestamp function for event saving (1 event)
- Tamper detection:
  - 1 tamper event on edge detection (in low and medium density devices)
  - 3 tamper events with configurable filter and internal pull-up (high and medium+ density devices only).
  - 5 backup registers (20 bytes) in value line, 20 backup registers (80 bytes) in low and medium density devices (except value line) and 32 registers (128 bytes) in high and medium+ density devices.
- RTC alternate function outputs (RTC_AFO):
  - AFO_CALIB: 512 Hz or 1Hz (high and medium+ density devices only) clock output (with an LSE frequency of 32.768 kHz). It is routed to the device RTC_AF1 pin.
  - AFO_ALARM: Alarm A or Alarm B or wakeup (only one can be selected). It is routed to the device RTC_AF1 pin.
- RTC alternate function inputs (RTC_AFI):
  - AFI_TAMPER1: tamper event detection. It is routed to the device RTC_AF1 pin.
  - AFI_TAMPER2: tamper2 event detection. It is routed to the device RTC_TAMPER2 pin.
  - AFI_TAMPER3 : tamper3 event detection. It is routed to the device RTC_TAMPER3 pin.
  - AFI_TIMESTAMP: timestamp event detection. It is routed to the device RTC_AF1

pin.

Note: Refer to Section 6.4.1: GPIO port mode register (GPIOx_MODER) (x = A..H) for more details on how to select RTC alternate functions (RTC_AF1).

Figure 162. RTC block diagram (low and medium density devices)

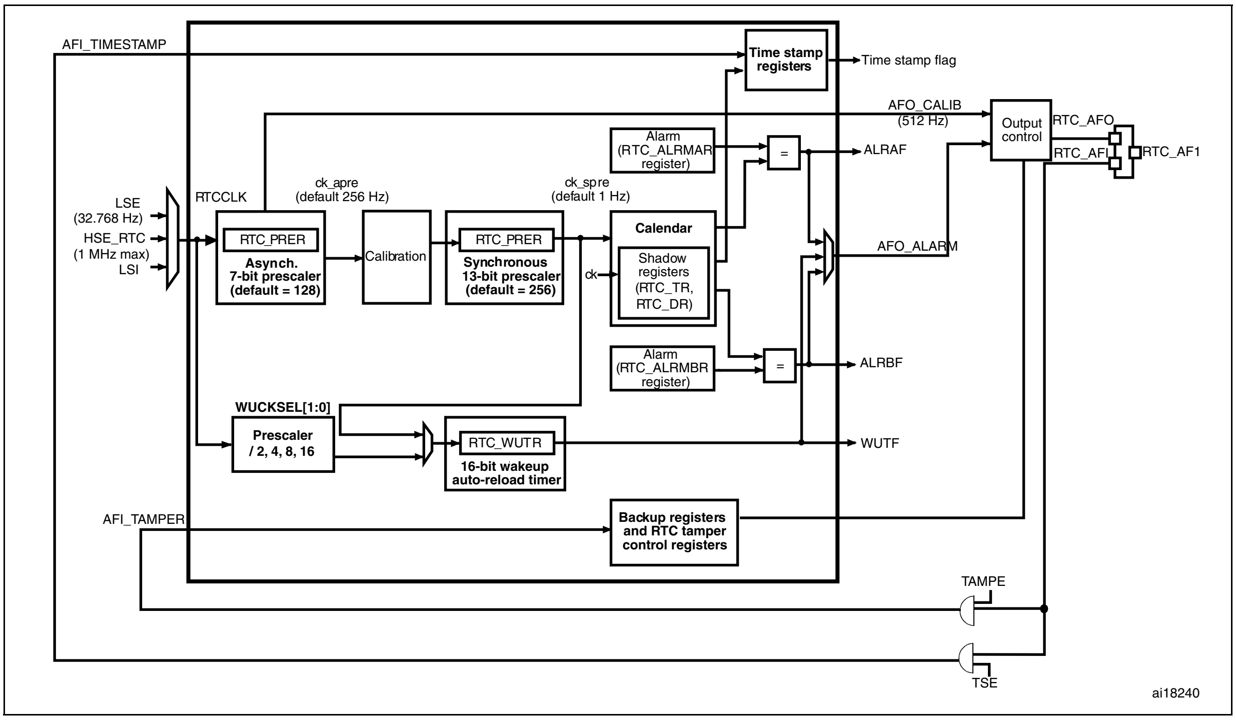
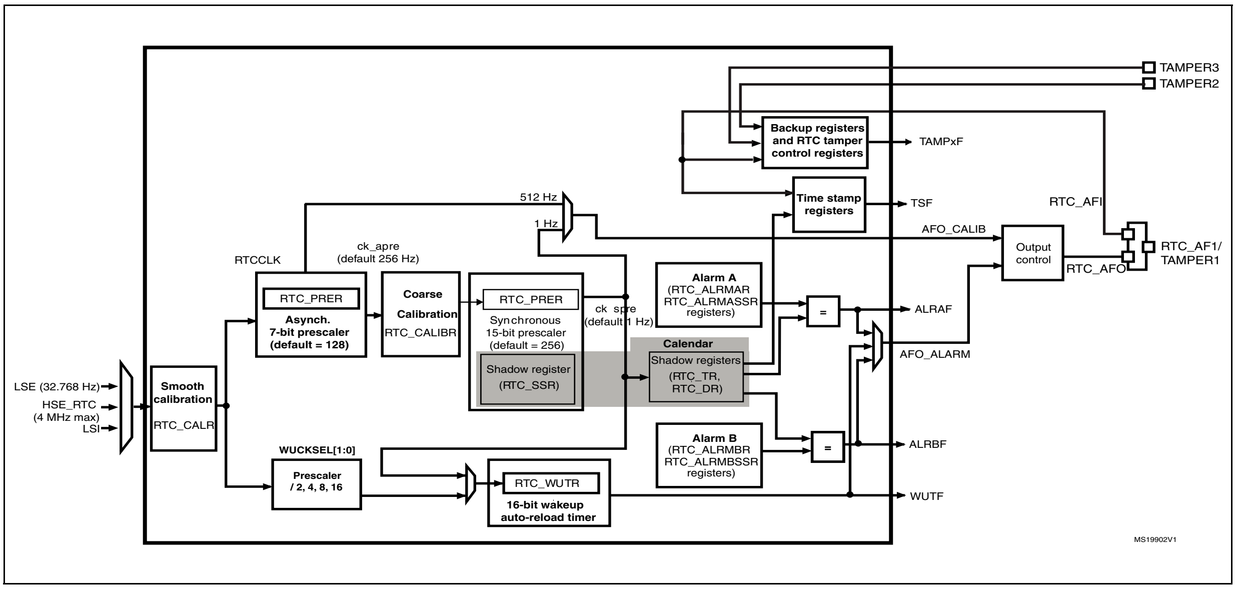


Figure 163. RTC block diagram (high and medium+ density devices)


TAMPER3
TAMPER2
Backup registers and RTC tamper control registers
TAMPxF
Time stamp registers
TSF
RTC_AFI
512 Hz
1 Hz
AFO_CALIB
Output control
RTC_AF1/ TAMPER1
RTC_AFO
RTCCLK
ck_apre (default 256 Hz)
RTC_PRER
Asynch. 7-bit prescaler (default = 128)
Coarse Calibration RTC_CALIBR
RTC_PRER
Synchronous 15-bit prescaler (default = 256)
ck_spre (default 1 Hz)
Alarm A (RTC_ALRMAR RTC_ALRMASSR registers)
=
ALRAF
AFO_ALARM
Shadow register (RTC_SSR)
Calendar
Shadow registers (RTC_TR, RTC_DR)
LSE (32.768 Hz)
HSE_RTC (4 MHz max)
LSI
Smooth calibration
RTC_CALR
Alarm B (RTC_ALRMBR RTC_ALRMBSSR registers)
=
ALRBF
WUCKSEL[1:0]
Prescaler / 2, 4, 8, 16
RTC_WUTR
16-bit wakeup auto-reload timer
WUTF
MS19902V1

19.3 RTC functional description

19.3.1 Clock and prescalers

The RTC clock source (RTCCLK) is selected through the clock controller among the LSE clock, the LSI oscillator clock, and the HSE clock. For more information on the RTC clock source configuration, refer to *Section 5: Reset and clock control (RCC)*.

A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into 2 programmable prescalers (see *Figure 162: RTC block diagram (low and medium density devices)*):

- A 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.
- The prescaler features 13 bits for low and medium density devices, and 15 bits for high and medium+ density devices.

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 2 in low and medium density devices and 1 in high and medium+ density devices.

The maximum division factor is 2^{20} in low and medium density devices and 2^{22} in high and medium+ density devices.

This corresponds to a maximum input frequency of around 1 MHz for low and medium density devices and 4 MHz for high and medium+ density devices.

f_{ck_apre} is given by the following formula:

$$f_{CK\_APRE} = \frac{f_{RTCCLK}}{PREDIV\_A + 1}$$

The ck_apre clock is used to clock the binary RTC_SSR subseconds downcounter. When it reaches 0, RTC_SSR is reloaded with the content of PREDIV_S. RTC_SSR is available in in high and medium+ density devices only.

f_{ck_spre} is given by the following formula:

$$f_{CK\_SPRE} = \frac{f_{RTCCLK}}{(PREDIV\_S + 1) \times (PREDIV\_A + 1)}$$

The ck_spre clock can be used either to update the calendar or as timebase for the 16-bit wakeup auto-reload timer. To obtain short timeout periods, the 16-bit wakeup auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see *Section 19.3.4: Periodic auto-wakeup* for details).

19.3.2 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK1 (APB1 clock). In high and medium+ density devices, they can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC_SSR for the subseconds (available on high and medium+ density devices only)
- RTC_TR for the time
- RTC_DR for the date

Every two RTCCLK periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ISR register is set (see *Section 19.6.4*). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 2 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. In high and medium+ density devices, 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.

19.3.3 Programmable alarms

The RTC unit provides two programmable alarms, Alarm A and Alarm B.

The programmable alarm functions are enabled through the ALRAIE and ALRBIE bits in the RTC_CR register. The ALRAF and ALRBF flags are set to 1 if the calendar subseconds(high and medium+ density devices only), seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASSR/RTC_ALRMAR and RTC_ALRMBSSR/RTC_ALRMBR, respectively. Each calendar field can be independently selected through the MSKx bits of the RTC_ALRMAR and RTC_ALRMBR registers, and through the MASKSSx bits of the RTC_ALRMASSR and RTC_ALRMBSSR registers. The alarm interrupts are enabled through the ALRAIE and ALRBIE bits in the RTC_CR register.

Alarm A and Alarm B (if enabled by bits OSEL[0:1] in RTC_CR register) can be routed to the AFO_ALARM output. AFO_ALARM polarity can be configured through bit POL in the RTC_CR register.

Caution: If the seconds field is selected (MSK0 bit reset in RTC_ALRMAR or RTC_ALRMBR), the synchronous prescaler division factor set in the RTC_PRER register must be at least 3 to ensure correct behavior.

19.3.4 Periodic auto-wakeup

The periodic wakeup flag is generated by a 16-bit programmable auto-reload down-counter. The wakeup timer range can be extended to 17 bits.

The wakeup function is enabled through the WUTE bit in the RTC_CR register.

The wakeup timer clock input can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.

  When RTCCLK is LSE(32.768kHz), this allows to configure the wakeup interrupt period from 122 µs to 32 s, with a resolution down to 61µs.
- ck_spre (usually 1 Hz internal clock)

  When ck_spre frequency is 1Hz, this allows to achieve a wakeup time from 1 s to around 36 hours with one-second resolution. This large programmable time range is divided in 2 parts:
  - from 1s to 18 hours when WUCKSEL [2:1] = 10
  - and from around 18h to 36h when WUCKSEL[2:1] = 11. In this last case 2^{16} is added to the 16-bit counter current value.When the initialization sequence is complete (see *Programming the wakeup timer on page 470*), the timer starts counting down.When the wakeup function is enabled, the down-counting remains active in low power modes. In addition, when it reaches 0, the WUTF flag is set in the RTC_ISR register, and the wakeup counter is automatically reloaded with its reload value (RTC_WUTR register value).

The WUTF flag must then be cleared by software.

When the periodic wakeup interrupt is enabled by setting the WUTIE bit in the RTC_CR2 register, it can exit the device from low power modes.

The periodic wakeup flag can be routed to the AFO_ALARM output provided it has been enabled through bits OSEL[0:1] of RTC_CR register. AFO_ALARM polarity can be configured through the POL bit in the RTC_CR register.

System reset, as well as low power modes (Sleep, Stop and Standby) have no influence on the wakeup timer.

19.3.5 RTC initialization and configuration

RTC register access

The RTC registers are 32-bit registers. The APB interface introduces 2 wait-states in RTC register accesses except on read accesses to calendar shadow registers when BYPSHAD=0.

RTC register write protection

After system reset, the RTC registers are protected against parasitic write access with the DBP bit of the PWR power control register (PWR_CR). The DBP bit must be set to enable RTC registers write access.

After power-on reset, all the RTC registers are write-protected. Writing to the RTC registers is enabled by writing a key into the Write Protection register, RTC_WPR.

The following steps are required to unlock the write protection on all the RTC registers except for RTC_ISR[13:8], RTC_TAFCR, and RTC_BKPxR.

1. Write ‘0xCA’ into the RTC_WPR register.
2. Write ‘0x53’ into the RTC_WPR register.

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC_ISR 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_ISR register. The initialization phase mode is entered when INITF is set to 1. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. To generate a 1 Hz clock for the calendar counter, program first the synchronous prescaler factor in RTC_PRER register, and then program the asynchronous prescaler factor. Even if only one of the two fields needs to be changed, 2 separate write accesses must be performed to the RTC_PRER register.
4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded and the counting restarts after 4 RTCCLK clock cycles.

When the initialization sequence is complete, the calendar starts counting.

Note: *After a system reset, the application can read the INITS flag in the RTC_ISR 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 power-on reset default value (0x00).*

To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ISR 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 (Alarm A or Alarm B):

1. Clear ALRAE or ALRBIE in RTC_CR to disable Alarm A or Alarm B.
2. Poll ALRAWF or ALRBWF in RTC_ISR until it is set to make sure the access to alarm registers is allowed. In low and medium density devices, tThis takes around 2 RTCCLK clock cycles (due to clock synchronization). In high and medium+ density devices, ALRAWF and ALRBWF are always set, so this step can be skipped.
3. Program the Alarm A or Alarm B registers (RTC_ALRMASSR/RTC_ALRMAR or RTC_ALRMBSSR/RTC_ALRMBR).
4. Set ALRAE or ALRBIE in the RTC_CR register to enable Alarm A or Alarm B again.

Note: *Each change of the RTC_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.*

Programming the wakeup timer

The following sequence is required to configure or change the wakeup timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wakeup timer.
2. Poll WUTWF until it is set in RTC_ISR to make sure the access to wakeup auto-reload counter and to WUCKSEL[2:0] bits is allowed. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. Program the wakeup auto-reload value WUT[15:0], and the wakeup clock selection (WUCKSEL[2:0] bits in RTC_CR).Set WUTE in RTC_CR to enable the timer again. The wakeup timer restarts down-counting.

19.3.6 Reading the calendar

In low and medium density devices, or when BYPSHAD control bit is cleared in the RTC_CR register

To read the RTC calendar registers (RTC_SSR, RTC_TR and RTC_DR) properly, the APB1 clock frequency (f_{PCLK1}) must be equal to or greater than seven times the f_{RTCCLK} RTC clock frequency. This ensures a secure behavior of the synchronization mechanism.

If the APB1 clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third read access must be done. In any case the APB1 clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in RTC_ISR register each time the calendar registers are copied into the RTC_SSR, RTC_TR and RTC_DR shadow registers. The copy is performed every two RTCCLK cycles. To ensure consistency between the 3 values, reading either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read. In case the software makes read accesses to the calendar in a time interval smaller than 2 RTCCLK periods: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC_SSR, RTC_TR and RTC_DR registers.

After waking up from low power mode (Stop or Standby), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC_SSR, RTC_TR and RTC_DR registers.

The RSF bit must be cleared after wakeup and not before entering low power mode.

Note: 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 469): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to Section 19.3.8: RTC synchronization (high and medium+ density devices only)): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

In high and medium+ density devices, 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.

19.3.7 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and some bits of the RTC status register (RTC_ISR) 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 power-on 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 registers (RTC_CALIBR or RTC_CALR), the RTC shift register (RTC_SHIFTR), the RTC timestamp registers (RTC_TSSSR, RTC_TSTR and RTC_TSDR), the RTC tamper and alternate function configuration register (RTC_TAFCR), the RTC backup registers (RTC_BKPxR), the wakeup timer register (RTC_WUTR), the Alarm A and Alarm B registers (RTC_ALRMASSR/RTC_ALRMAR and RTC_ALRMBSSR/RTC_ALRMBR).

In addition, the RTC keeps on running under system reset if the reset source is different from the power-on reset one. When a power-on reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

19.3.8 RTC synchronization (high and medium+ density devices only)

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the sub-second field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC_SHIFTR.

RTC_SSR contains the value of the synchronous prescaler’s counter. This allows one to calculate the exact time being maintained by the RTC down to a resolution of 1 / (PREDIV_S + 1) seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV_S[14:0]. The maximum resolution allowed (30.52 µs with a 32768 Hz clock) is obtained with PREDIV_S set to 0x7FFF.

However, increasing PREDIV_S means that PREDIV_A must be decreased in order to maintain the synchronous prescaler’s output at 1 Hz. In this way, the frequency of the asynchronous prescaler’s output increases, which may increase the RTC dynamic consumption.

The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock by up to a second with a resolution of 1 / (PREDIV_S + 1) seconds. The shift operation consists of adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this will delay the clock. If at the same time the ADD1S bit is set, this results in adding one second and at the same time subtracting a fraction of second, so this will advance the clock.

Caution: Before initiating a shift operation, the user must check that SS[15] = 0 in order to ensure that no overflow will occur.

As soon as a shift operation is initiated by a write to the RTC_SHIFTR register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.

Caution: This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC_SHIFTR when REFCKON=1.

19.3.9 RTC reference clock detection

The RTC calendar update can be synchronized to a reference clock RTC_REFIN, usually the mains (50 or 60 Hz). The RTC_REFIN reference clock should have a higher precision 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 reference 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_apre 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 reference clock detection is enabled, PREDIV_A and PREDIV_S must be set to their default values:

- PREDIV_A = 0x007F
- PREVID_S = 0x00FF

Note: The reference clock detection is not available in Standby mode.

Caution: The reference clock detection feature cannot be used in conjunction with the coarse digital calibration: RTC_CALIBR must be kept at 0x0000 0000 when REFCKON=1.

19.3.10 RTC coarse digital calibration

Two digital calibration methods are available: coarse and smooth calibration. To perform coarse calibration refer to *Section 19.6.7: RTC calibration register (RTC_CALIBR)*.

Smooth digital calibration is available on high and medium+ density devices only. The two calibration methods are not intended to be used together, the application must select one of the two methods. Coarse calibration is provided for compatibly reasons. To perform smooth calibration refer to *Section 19.3.11: RTC smooth digital calibration (high and medium+ density devices only)* and the *Section 19.6.16: RTC calibration register (RTC_CALR)*

The coarse digital calibration can be used to compensate crystal inaccuracy by adding (positive calibration) or masking (negative calibration) clock cycles at the output of the asynchronous prescaler (ck_apre).

Positive and negative calibration are selected by setting the DCS bit in RTC_CALIBR register to ‘0’ and ‘1’, respectively.

When positive calibration is enabled (DCS = ‘0’), 2 ck_apre cycles are added every minute (around 15360 ck_apre cycles) for 2xDC minutes. This causes the calendar to be updated sooner, thereby adjusting the effective RTC frequency to be a bit higher.

When negative calibration is enabled (DCS = ‘1’), 1 ck_apre cycle is removed every minute (around 15360 ck_apre cycles) for 2xDC minutes. This causes the calendar to be updated later, thereby adjusting the effective RTC frequency to be a bit lower.

DC is configured through bits DC[4:0] of RTC_CALIBR register. This number ranges from 0 to 31 corresponding to a time interval (2xDC) ranging from 0 to 62.

The coarse digital calibration can be configured only in initialization mode, and starts when the INIT bit is cleared. The full calibration cycle lasts 64 minutes. The first 2xDC minutes of the 64 -minute cycle are modified as just described.

Negative calibration can be performed with a resolution of about 2 ppm while positive calibration can be performed with a resolution of about 4 ppm. The maximum calibration ranges from -63 ppm to 126 ppm.

The calibration can be performed either on the LSE or on the HSE clock.

Caution: Digital calibration may not work correctly if PREDIV_A < 6.

Case of RTCCLK=32.768 kHz and PREDIV_A+1=128

The following description assumes that ck_apre frequency is 256 Hz obtained with an LSE clock nominal frequency of 32.768 kHz, and PREDIV_A set to 127 (default value).

The ck_spre clock frequency is only modified during the first 2xDC minutes of the 64-minute cycle. For example, when DC equals 1, only the first 2 minutes are modified. This means that the first 2xDC minutes of each 64-minute cycle have, once per minute, one second either shortened by 256 or lengthened by 128 RTCCLK cycles, given that each ck_apre cycle represents 128 RTCCLK cycles (with PREDIV_A+1=128).

Therefore each calibration step has the effect of adding 512 or subtracting 256 oscillator cycles for every 125829120 RTCCLK cycles (64min x 60 s/min x 32768 cycles/s). This is equivalent to +4.069 ppm or-2.035 ppm per calibration step. As a result, the calibration resolution is +10.5 or -5.27 seconds per month, and the total calibration ranges from +5.45 to –2.72 minutes per month.

In order to measure the clock deviation, a 512 Hz clock is output for calibration.Refer to *Section 19.3.14: Calibration clock output*.

19.3.11 RTC smooth digital calibration (high and medium+ density devices only)

In high and medium+ density devices, the RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual RTCCLK pulses). These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

The smooth digital calibration is performed during a cycle of about 2^{20} RTCCLK pulses, or 32 seconds when the input frequency is 32768 Hz.

The smooth calibration register (RTC_CALR) specifies the number of RTCCLK clock cycles to be masked during the 32-second cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the 32-second cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked
- Setting SMC[2] to 1 causes four additional cycles to be masked
- and so on up to SMC[8] set to 1 which causes 256 clocks to be masked.

While CALM allows the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP to ‘1’ effectively inserts an extra RTCCLK pulse every 2^{11} RTCCLK cycles, which means that 512 clocks are added during every 32-second cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 RTCCLK cycles can be added during the 32-second cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (FCAL) given the input frequency (FRTCCLK) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [\ 1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)\ ]$$

Calibration when PREDIV_A<3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV_A bits in RTC_PRER register) is less than 3. If CALP was already set to 1 and PREDIV_A bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

To perform a calibration with PREDIV_A less than 3, the synchronous prescaler value (PREDIV_S) should be reduced so that each second is accelerated by 8 RTCCLK clock cycles, which is equivalent to adding 256 clock cycles every 32 seconds. 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 32-second 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

RTC precision is performed by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided on high and medium+ density devices 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 8s). 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_ISR/INITF=0, by using the follow process:

1. Poll the RTC_ISR/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.

19.3.12 Timestamp function

Timestamp is enabled by setting the TSE bit of RTC_CR register to 1.

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a timestamp event is detected on the pin to which the TIMESTAMP alternate function is mapped. When a timestamp event occurs, the timestamp flag bit (TSF) in RTC_ISR 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 2 ck_apre cycles after the timestamp event occurs due to synchronization process.

There is no delay in the setting of TSOVF. This means that if two timestamp events are close together, TSOVF can be seen as '1' while TSF is still '0'. As a consequence, it is recommended to poll TSOVF only after TSF has been set.

Caution: If a timestamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a timestamp event occurring at the same moment, the application must not write ‘0’ into TSF bit unless it has already read it to ‘1’.

Optionally, a tamper event can cause a timestamp to be recorded. See the description of the TAMPTS control bit in *Section 19.6.17: RTC tamper and alternate function configuration register (RTC_TAFCR)*. If the timestamp event is on the same pin as a tamper event configured in filtered mode (TAMPFLT set to a non-zero value), the timestamp on tamper detection event mode must be selected by setting TAMPTS='1' in RTC_TAFCR register.

TIMESTAMP alternate function

The TIMESTAMP alternate function is mapped to RTC_AF1.

19.3.13 Tamper detection

One tamper detection input on edge detection is available on low and medium density devices.

Three tamper detection inputs are available on high and medium+ density devices. They can be configured either for edge detection, or for level detection with filtering.

RTC backup registers

The backup registers (RTC_BKPxR) are five 32-bit registers in value line devices for storing 20 bytes of user application data, twenty 32-bit registers in low and medium density devices (except value line) for storing 80 bytes of user application data and thirty-two 32-bit registers in high and medium+ density devices for storing 128 bytes of user application data. They are implemented in the V_{DD} domain . They are not reset by system reset, or when the device wakes up from Standby mode. They are reset by a power-on reset.

The backup registers are reset when a tamper detection event occurs (see *Section 19.6.20: RTC backup registers (RTC_BKPxR)* and *Tamper detection initialization on page 476*.

Tamper detection initialization

Each tamper detection input is associated with a flag TAMP1F/TAMP2F/ TAMP3F in the RTC_ISR2 register. Each input can be enabled by setting the corresponding TAMP1E/TAMP2E/TAMP3E bits to 1 in the RTC_TAFCR register.

A tamper detection event resets all backup registers (RTC_BKPxR).

By setting the TAMPIE bit in the RTC_TAFCR register, an interrupt is generated when a tamper detection event occurs.

Timestamp on tamper event

With TAMPTS set to ‘1 (high devices only), any tamper event causes a timestamp to occur. In this case, either the TSF bit or the TSOVF bit are set in RTC_ISR, in the same manner as if a normal timestamp event occurs. The affected tamper flag register (TAMP1F, TAMP2F, or TAMP3F) is set at the same time that TSF or TSOVF is set.

Edge detection on tamper inputs

If the TAMPFLT bits are “00”, the TAMPER pins generate tamper detection events (AFI_TAMPER[3:1]) when either a rising edge is observed or an falling edge is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the TAMPER inputs are deactivated when edge detection is selected.

Caution: To avoid losing tamper detection events, the signal used for edge detection is logically ANDed with TAMPxE in order to detect a tamper detection event in case it occurs before the TAMPERx pin is enabled.

- When TAMPxTRG = 0: if the TAMPERx alternate function is already high before tamper detection is enabled (TAMPxE bit set to 1), a tamper event is detected as soon as TAMPERx is enabled, even if there was no rising edge on TAMPERx after TAMPxE was set.
- When TAMPxTRG = 1: if the TAMPERx alternate function is already low before tamper detection is enabled, a tamper event is detected as soon as TAMPERx is enabled (even if there was no falling edge on TAMPERx after TAMPxE was set.

After a tamper event has been detected and cleared, the TAMPERx alternate function should be disabled and then re-enabled (TAMPxE set to 1) before re-programming the backup registers (RTC_BKPxR). This prevents the application from writing to the backup registers while the TAMPERx value still indicates a tamper detection. This is equivalent to a level detection on the TAMPERx alternate function.

Note: Tamper detection is still active when V_{DD} power is switched off. To avoid unwanted resetting of the backup registers, the pin to which the TAMPER alternate function is mapped should be externally tied to the correct level.

Level detection with filtering on tamper inputs (high and medium+ density devices only)

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 (TAMP1TRG/TAMP2TRG/TAMP3TRG).

The TAMPER inputs are pre-charged 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 tamper inputs.

The trade-off between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection.

Note: Refer to the datasheets for the electrical characteristics of the pull-up resistors.

TAMPER alternate function detection

The TAMPER1 alternate function is mapped to the RTC_AF1 pin. The TAMPER 2 and TAMPER 3 alternate functions are mapped to RTC_TAMP2 and RTC_TAMPER3 pins respectively.

19.3.14 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the RTC_CALIB device output. If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the RTC_CALIB frequency is $f_{RTCCLK/64}$. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz.

The RTC_CALIB output is not impacted by the calibration value programmed in RTC_CALIBR register. The RTC_CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

If COSEL is set (on high and medium+ density devices) and “PREDIV_S+1” is a non-zero multiple of 256 (i.e: PREDIV_S[7:0] = 0xFF), the RTC_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.

Calibration alternate function output

When the COE bit in the RTC_CR register is set to 1, the calibration alternate function (AFO_CALIB) is enabled on RTC_AF1.

19.3.15 Alarm output

Three functions can be selected on Alarm output: ALRAF, ALRBF and WUTF. These functions reflect the contents of the corresponding flags in the RTC_ISR register.

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm alternate function output (AFO_ALARM) in RTC_AF1, and to select the function which is output on AFO_ALARM.

The polarity of the output is determined by the POL control bit in RTC_CR so that the opposite of the selected flag bit is output when POL is set to 1.

Alarm alternate function output

AFO_ALARM can be configured in output open drain or output push-pull using the control bit ALARMOUTTYPE in the RTC_TAFCR register.

Note: the AFO_CALIB should be disabled (COE bit must be kept cleared).

When AFO_CALIB or AFO_ALARM is selected, RTC_AF1 *is automatically configured in output alternate function.*

19.4 RTC and low power modes

Table 75. 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 alarm, RTC tamper event, RTC time stamp event, and RTC Wakeup cause the device to exit the Stop mode.
Standby	The RTC remains active when the RTC clock source is LSE or LSI. RTC alarm, RTC tamper event, RTC time stamp event, and RTC Wakeup cause the device to exit the Standby mode.

19.5 RTC interrupts

All RTC interrupts are connected to the EXTI controller.

To enable the RTC Alarm interrupt, the following sequence is required:

1. Configure and enable the EXTI Line 17 in interrupt mode and select the rising edge sensitivity.
2. Configure and enable the RTC_Alarm IRQ channel in the NVIC.
3. Configure the RTC to generate RTC alarms (Alarm A or Alarm B).

To enable the RTC Wakeup interrupt, the following sequence is required:

1. Configure and enable the EXTI Line 20 in interrupt mode and select the rising edge sensitivity.
2. Configure and enable the RTC_WKUP IRQ channel in the NVIC.
3. Configure the RTC to generate the RTC wakeup timer event.

To enable the RTC Tamper interrupt, the following sequence is required:

1. Configure and enable the EXTI Line 19 in interrupt mode and select the rising edge sensitivity.
2. Configure and Enable the TAMP_STAMP IRQ channel in the NVIC.
3. Configure the RTC to detect the RTC tamper event.

To enable the RTC TimeStamp interrupt, the following sequence is required:

1. Configure and enable the EXTI Line 19 in interrupt mode and select the rising edge sensitivity.
2. Configure and Enable the TAMP_STAMP IRQ channel in the NVIC.
3. Configure the RTC to detect the RTC timestamp event.

Table 76. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Exit the Sleep mode	Exit the Stop mode	Exit the Standby mode
Alarm A	ALRAF	ALRAIE	yes	yes[1]	yes[1]
Alarm B	ALRBF	ALRBIE	yes	yes[1]	yes[1]
Wakeup	WUTF	WUTIE	yes	yes[1]	yes[1]
TimeStamp	TSF	TSIE	yes	yes[1]	yes[1]
Tamper1 detection	TAMP1F	TAMPIE	yes	yes[1]	yes[1]
Tamper2 detection[2]	TAMP2F	TAMPIE	yes	yes[1]	yes[1]
Tamper3 detection[3]	TAMP3F	TAMPIE	yes	yes[1]	yes[1]

1. Wakeup from STOP and Standby modes is possible only when the RTC clock source is LSE or LSI.
2. If RTC_TAMPER2 pin is present (only on high and medium+ density devices). Refer to device datasheet pinout.
3. Only on high and medium+ density devices if RTC_TAMPER3 pin is present. Refer to device datasheet pinout.

19.6 RTC registers

Refer to *Section 1.1* of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32 bits).

19.6.1 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 469* and *Reading the calendar on page 470*.

Address offset: 0x00

Power-on reset value: 0x0000 0000

System reset: 0x0000 0000 when BYPSHAD = 0. Not affected when BYPSHAD = 1

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

Bit 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

Bit 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

Note: This register is write protected. The write access procedure is described in RTC register write protection on page 468.

19.6.2 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 469* and *Reading the calendar on page 470*.

Address offset: 0x04

Power-on reset value: 0x0000 2101

System reset: 0x0000 2101 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								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]				Reserved		DT[1:0]		DU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31-24 Reserved

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**: 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: This register is write protected. The write access procedure is described in RTC register write protection on page 468.

19.6.3 RTC control register (RTC_CR)

Address offset: 0x08

Power-on value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
								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	ALRBIE	ALRAE	DCE	FMT	BYPSHAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 23 **COE**: Calibration output enable

This bit enables the AFO_CALIB RTC 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 AFO_ALARM RTC output
00: Output disabled
01: Alarm A output enabled
10: Alarm B output enabled
11: Wakeup output enabled

Bit 20 **POL**: Output polarity

This bit is used to configure the polarity of AFO_ALARM RTC output
0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0])
1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]).

Bit 19 **COSEL**: Calibration output selection

This bit is available in high and medium+ density devices only.

When COE=1, this bit selects which signal is output on RTC_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 19.3.14: Calibration clock output*

Bit 18 **BKP**: Backup

This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: *S*ubtract 1 hour (winter time change)

When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.
Setting this bit has no effect when current hour is 0.
0: No effect
1: Subtracts 1 hour to the current time. This can be used for winter time change.

Bit 16 **ADD1H**: Add 1 hour (summer time change)

When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.

0: No effect
1: Adds 1 hour to the current time. This can be used for summer time change

Bit 15 **TSIE**: Timestamp interrupt enable

0: Timestamp Interrupt disable
1: Timestamp Interrupt enable

Bit 14 **WUTIE**: Wakeup timer interrupt enable

0: Wakeup timer interrupt disabled
1: Wakeup timer interrupt enabled

Bit 13 **ALRBIE**: *Alarm B interrupt enable*

0: Alarm B Interrupt disable
1: Alarm B Interrupt enable

Bit 12 **ALRAIE**: Alarm A interrupt enable

0: Alarm A interrupt disabled
1: Alarm A interrupt enabled

Bit 11 **TSE**: Time stamp enable

0: Time stamp disable
1: Time stamp enable

Bit 10 **WUTE**: Wakeup timer enable

0: Wakeup timer disabled
1: Wakeup timer enabled

Bit 9 **ALRBIE**: *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 **DCE:** Coarse digital calibration enable

0: Digital calibration disabled
1: Digital calibration enabled
PREDIV_A must be 6 or greater

Bit 6 **FMT**: Hour format

0: 24 hour/day format
1: AM/PM hour format

Bit 5 **BYPSHAD**: Bypass the shadow registers

This bit is available on high and medium+ density devices only.

0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.

1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters.

Note: If the frequency of the APB1 clock is less than seven times the frequency of RTCCLK, BYPSHAD must be set to ‘1’.

Bit 4 **REFCKON**: Reference clock detection enable (50 or 60 Hz)

0: Reference clock detection disabled
1: Reference clock detection enabled

Note: PREDIV_S must be 0x00FF.

Bit 3 **TSEDGE**: Timestamp event active edge

0: TIMESTAMP rising edge generates a timestamp event
1: TIMESTAMP 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]**: Wakeup clock selection

000: RTC/16 clock is selected
001: RTC/8 clock is selected
010: RTC/4 clock is selected
011: RTC/2 clock is selected
10x: ck_spre (usually 1 Hz) clock is selected
11x: ck_spre (usually 1 Hz) clock is selected and 2^{16} is added to the WUT counter value (see note below)

Note: WUT = Wakeup unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].

Bits 7, 6 and 4 of this register can be written in initialization mode only (RTC_ISR/INITF = 1).

Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ISR WUTWF bit = 1.

It is recommended not to change the hour during the calendar hour increment as it could mask the incrementation of the calendar hour.

ADD1H and SUB1H changes are effective in the next second.

This register is write protected. The write access procedure is described in RTC register write protection on page 468.

19.6.4 RTC initialization and status register (RTC_ISR)

Address offset: 0x0C

Power-on reset value: 0x0000 0007

System reset value: Not affected except INIT, INITF and RSF which are cleared to 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															RECALPF
															r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAMP3F	TAMP2F	TAMP1F	TSOVF	TSF	WUTF	ALRBF	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUTWF	ALRBWF	ALRAWF
rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rw	r	rc_w0	r	rc_w0	r	r	r

Bits 31:17 Reserved

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 *Section : Re-calibration on-the-fly*.

Bit 15 **TAMP3F**: TAMPER3 detection flag

This flag is set by hardware when a tamper detection event is detected on tamper input 3.

It is cleared by software writing 0.

Bit 14 **TAMP2F**: TAMPER2 detection flag

This flag is set by hardware when a tamper detection event is detected on tamper input 2.

It is cleared by software writing 0.

Bit 13 **TAMP1F**: Tamper detection flag

This flag is set by hardware when a tamper detection event is detected.

It is cleared by software writing 0.

Bit 12 **TSOVF**: Timestamp overflow flag

This flag is set by hardware when a timestamp event occurs while TSF is already set.

This flag is cleared by software by writing 0. 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 11 **TSF**: Timestamp flag

This flag is set by hardware when a timestamp event occurs.

This flag is cleared by software by writing 0.

Bit 10 **WUTF**: Wakeup timer flag

This flag is set by hardware when the wakeup auto-reload counter reaches 0.

This flag is cleared by software by writing 0.

This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 9 **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).

This flag is cleared by software by writing 0.

Bit 8 **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).

This flag is cleared by software by writing 0.

Bit 7 **INIT**: Initialization mode

0: Free running mode

1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER). Counters are stopped and start counting from the new value when INIT is reset.

Bit 6 **INITF**: Initialization flag

When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.

0: Calendar registers update is not allowed

1: Calendar registers update is allowed.

Bit 5 **RSF**: Registers synchronization flag

This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSRx, RTC_TRx and RTC_DRx). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF=1), or when in bypass shadow register mode (BYPSHAD=1). This bit can also be cleared by software.

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 (power-on reset state).

0: Calendar has not been initialized

1: Calendar has been initialized

Bit 3 **SHPF**: Shift operation pending

0: No shift operation is pending

1: A shift operation is pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR. It is cleared by hardware when the corresponding shift operation has been executed. Writing to SHPF has no effect.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **WUTWF**: Wakeup timer write flag

This bit is set by hardware when the wakeup timer values can be changed, after the WUTE bit has been set to 0 in RTC_CR.

0: Wakeup timer configuration update not allowed

1: Wakeup timer configuration update allowed

Bit 1 **ALRBWF**: Alarm B write flag

This bit is set by hardware when Alarm B values can be changed, after the ALRBIE bit has been set to 0 in RTC_CR.

It is cleared by hardware in initialization mode.

0: Alarm B update not allowed

1: Alarm B update allowed.

Bit 0 **ALRAWF**: Alarm A write flag

This bit is set by hardware when Alarm A values can be changed, after the ALRAE bit has been set to 0 in RTC_CR.

It is cleared by hardware in initialization mode.

0: Alarm A update not allowed

1: Alarm A update allowed

Note: The ALRAF, ALRBF, WUTF and TSF bits are cleared 2 APB clock cycles after programming them to 0.

This register is write protected (except for RTC_ISR[13:8] bits). The write access procedure is described in RTC register write protection on page 468.

19.6.5 RTC prescaler register (RTC_PRER)

Address offset: 0x10

Power-on reset value: 0x007F 00FF

System reset: not affected

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

Bit 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)

Note: PREDIV_A [6:0]= 000000 is a prohibited value on low and medium density devices.

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)

Note: PREDIV[14:13] are reserved in low and medium density devices.

Note: 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 469

This register is write protected. The write access procedure is described in RTC register write protection on page 468.

19.6.6 RTC wakeup timer register (RTC_WUTR)

Address offset: 0x14

Power-on reset value: 0x0000 FFFF

System reset: not affected

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

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

Bits 31:16 Reserved

Bits 15:0 **WUT[15:0]**: Wakeup auto-reload value bits

When the wakeup timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck_wut cycles. The ck_wut period is selected through WUCKSEL[2:0] bits of the RTC_CR register

When WUCKSEL[2] = 1, the wakeup timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

Note: The first assertion of WUTF occurs (WUT+1) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] =011 (RTCCLK/2) is forbidden.

Note: This register can be written only when WUTWF is set to 1 in RTC_ISR.

This register is write protected. The write access procedure is described in RTC register write protection on page 468.

19.6.7 RTC calibration register (RTC_CALIBR)

Address offset: 0x18

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
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DCS	Reserved		DC[4:0]				
								rw			rw	rw	rw	rw	rw

Bits 31:8 Reserved

Bit 7 **DCS**: Digital calibration sign

0: Positive calibration: calendar update frequency is increased
1: Negative calibration: calendar update frequency is decreased

Bits 6:5 Reserved, must be kept at reset value.

Bits 4:0 **DC[4:0]**: Digital calibration

DCS = 0 (positive calibration)
00000: + 0 ppm
00001: + 4 ppm (rounded value)
00010: + 8 ppm (rounded value)
..
11111: + 126 ppm (rounded value)
DCS = 1 (negative calibration)
00000: –0 ppm
00001: –2 ppm (rounded value)
00010: –4 ppm (rounded value)
..
11111: –63 ppm (rounded value)
Refer to Case of RTCCLK=32.768 kHz and PREDIV_A+1=128 on page 473 for the exact step value.

Note: This register can be written in initialization mode only (RTC_ISR/INITF = ‘1’).

This register is write protected. The write access procedure is described in RTC register write protection on page 468.

19.6.8 RTC alarm A register (RTC_ALRMAR)

Address offset: 0x1C

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
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.

Note: This register can be written only when ALRAWF is set to 1 in RTC_ISR, or in initialization mode.

This register is write protected. The write access procedure is described in RTC register write protection on page 468.

19.6.9 RTC alarm B register (RTC_ALRMBR)

Address offset: 0x20

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
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

Note: This register can be written only when ALRBWF is set to 1 in RTC_ISR, or in initialization mode.

This register is write protected. The write access procedure is described in RTC register write protection on page 468.

19.6.10 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
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								KEY							
								w	w	w	w	w	w	w	w

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

Bits 7:0 **KEY**: 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.

19.6.11 RTC sub second register (RTC_SSR)

The RTC_SSR register is available only on high and medium+ density devices.

Address offset: 0x28

Power-on reset value: 0x0000 0000

System reset: 0x0000 0000 when BYPSHAD = 0. Not affected when BYPSHAD = 1.

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

Bits 15:0 **SS**: Sub second value

SS[15:0] is the value in the synchronous prescaler's counter. The fraction of a second is given by the formula below:

Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)

Note: SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.

19.6.12 RTC shift control register (RTC_SHIFTR)

The RTC_SHIFTR register is available only in high and medium+ density devices.

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Reserved														
w	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.	SUBFS[14:0]														
r	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_ISR).

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

Bits 14:0 **SUBFS**: 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_ISR).

The value which is written to SUBFS is added to the synchronous prescaler’s counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:

Delay (seconds) = SUBFS / (PREDIV_S + 1)

A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:

Advance (seconds) = (1 - (SUBFS / (PREDIV_S + 1))) .

Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF=1 to be sure that the shadow registers have been updated with the shifted time.

Refer to *Section 19.3.8: RTC synchronization (high and medium+ density devices only)*.

Note: This register is write protected. The write access procedure is described in RTC register write protection on page 468

19.6.13 RTC time stamp time register (RTC_TSTR)

Address offset: 0x30

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
Reserved									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.

Note: The content of this register is valid only when TSF is set to 1 in RTC_ISR. It is cleared when TSF bit is reset.

19.6.14 RTC time stamp date register (RTC_TSDR)

Address offset: 0x34

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
Reserved															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WDU[1:0]			MT	MU[3:0]				Reserved		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[1:0]**: 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

Bit 3:0 **DU[3:0]**: Date units in BCD format

Note: The content of this register is valid only when TSF is set to 1 in RTC_ISR. It is cleared when TSF bit is reset.

19.6.15 RTC timestamp sub second register (RTC_TSSSR)

The RTC_TSSSR register is available only on high and medium+ density devices.

Address offset: 0x38

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
Reserved															
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 31:16 Reserved

Bits 15:0 **SS**: Sub second value

SS[15:0] is the value of the synchronous prescaler’s counter when the timestamp event occurred.

Note: The content of this register is valid only when RTC_ISR/TSF is set. It is cleared when the RTC_ISR/TSF bit is reset.

19.6.16 RTC calibration register (RTC_CALR)

The RTC_CALR register is available only on high and medium+ density devices.

Address offset: 0x3C

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
Reserved															
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
CALP	CALW8	CALW16	Reserved				CALM[8:0]								
rw	rw	rw	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:16 Reserved

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 19.3.11: RTC smooth digital calibration (high and medium+ density devices only)*.

Bit 14 **CALW8:** Use an 8-second calibration cycle period

When CALW8 is set to ‘1’, the 8-second calibration cycle period is selected.

CALM[1:0] are stuck at “00” when CALW8=’1’.

Refer to *Section 19.3.11: RTC smooth digital calibration (high and medium+ density devices only)*.

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 19.3.11: RTC smooth digital calibration (high and medium+ density devices only)*.

Bits 12:9 Reserved

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 19.3.11: RTC smooth digital calibration (high and medium+ density devices only) on page 474*.

Note: This register is write protected. The write access procedure is described in RTC register write protection on page 468

19.6.17 RTC tamper and alternate function configuration register (RTC_TAFCR)

Address offset: 0x40

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
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAMP-PUDIS	TAMP-PRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]			TAMPTS	TAMP3TRG	TAMP3E	TAMP2TRG	TAMP2E	TAMPIE	TAMP1TRG	TAMP1E
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved. Always read as 0.

Bit 18 **ALARMOUTTYPE**: AFO_ALARM output type

0: ALARM_AF0 is an open-drain output

1: ALARM_AF0 is a push-pull output

Bit 17:3 Reserved. Always read as 0.

Bit 15 **TAMPPUDIS**: TAMPER pull-up disable

This bit determines if each of the tamper pins are pre-charged before each sample.

0: Precharge tamper pins before sampling (enable internal pull-up)

1: Disable precharge of tamper pins

Note: This bit is available in high and medium+ density devices only

Bits 14:13 **TAMPPRCH[1:0]**: Tamper 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 tamper inputs.

0x0: 1 RTCCLK cycle

0x1: 2 RTCCLK cycles

0x2: 4 RTCCLK cycles

0x3: 8 RTCCLK cycles

Note: This bits is available in high and medium+ density devices only

Bits 12:11 **TAMPFLT[1:0]**: Tamper filter count

These bits determines the number of consecutive samples at the specified level (TAMP*TRG) necessary to activate a Tamper event. TAMPFLT is valid for each of the tamper inputs.

0x0: Tamper is activated on edge of tamper input transitions to the active level (no internal pull-up on tamper input).

0x1: Tamper is activated after 2 consecutive samples at the active level.

0x2: Tamper is activated after 4 consecutive samples at the active level.

0x3: Tamper is activated after 8 consecutive samples at the active level.

Note: This bit is available in high and medium+ density devices only

Bits 10:8 **TAMPFREQ[2:0]**: Tamper sampling frequency

Determines the frequency at which each of the tamper 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: These bits are available in high and medium+ density devices only

Bit 7 **TAMPTS**: Activate timestamp on tamper detection event

0: Tamper detection event does not cause a timestamp to be saved
1: Save timestamp on tamper detection event
TAMPTS is valid even if TSE=0 in the RTC_CR register.

Note: This bit is available in high and medium+ density devices only

Bit 6 **TAMP3TRG**: Active level for tamper 2

if TAMPFLT != 00 (high and medium+ density device only):

0: TAMPER3 staying low triggers a tamper detection event.
1: TAMPER3 staying high triggers a tamper detection event.

if TAMPFLT = 00:

0: TAMPER3 rising edge triggers a tamper detection event.
1: TAMPER3 falling edge triggers a tamper detection event.

Bit 5 **TAMP3E**: Tamper 3 detection enable

0: Tamper 3 detection disabled
1: Tamper 3 detection enabled

Bit 4 **TAMP2TRG**: Active level for tamper 2

if TAMPFLT != 00 (high and medium+ density device only):
0: TAMPER2 staying low triggers a tamper detection event.
1: TAMPER2 staying high triggers a tamper detection event.
if TAMPFLT = 00:
0: TAMPER2 rising edge triggers a tamper detection event.
1: TAMPER2 falling edge triggers a tamper detection event.

Note: This bit is available in high and medium+ density devices only

Bit 3 **TAMP2E**: Tamper 2 detection enable

0: Tamper 2 detection disabled
1: Tamper 2 detection enabled

Note: This bit is available in high and medium+ density devices only

Bit 2 **TAMPIE**: Tamper interrupt enable

0: Tamper interrupt disabled
1: Tamper interrupt enabled

Bit 1 **TAMP1TRG**: Active level for tamper 1

if TAMPFLT != 00 (high and medium+ density device only)

0: TAMPER1 staying low triggers a tamper detection event.

1: TAMPER1 staying high triggers a tamper detection event.

if TAMPFLT = 00:

0: TAMPER1 rising edge triggers a tamper detection event.

1: TAMPER1 falling edge triggers a tamper detection event.

Caution: When TAMPFLT = 0, TAMP1E must be reset when TAMP1TRG is changed to avoid spuriously setting TAMP1F.

Bit 0 **TAMP1E**: Tamper 1 detection enable

0: Tamper 1 detection disabled

1: Tamper 1 detection enabled

19.6.18 RTC alarm A sub second register (RTC_ALRMASSR)

The RTC_ALRMASSR register is available in high and medium+ density devices only.

Address offset: 0x44

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
Reserved				MASKSS[3:0]				Reserved							
r	r	r	r	rw	rw	rw	rw	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	SS[14:0]														
r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bits 31:28 Reserved

Bits 27:24 **MASKSS[3:0]**: Mask the most-significant bits starting at this bit

0: No comparison on sub seconds for Alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

1: SS[14:1] are don't care in Alarm A comparison. Only SS[0] is compared.

2: SS[14:2] are don't care in Alarm A comparison. Only SS[1:0] are compared.

3: SS[14:3] are don't care in Alarm A comparison. Only SS[2:0] are compared.

...

12: SS[14:12] are don't care in Alarm A comparison. SS[11:0] are compared.

13: SS[14:13] are don't care in Alarm A comparison. SS[12:0] are compared.

14: SS[14] is don't care in Alarm A comparison. SS[13:0] are compared.

15: All 15 SS bits are compared and must match to activate alarm.

The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved

Bits 14:0 **SS[14:0]**: Sub seconds value

This value is compared with the contents of the synchronous prescaler's counter to determine if Alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

Note: This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register is write protected. The write access procedure is described in RTC register write protection on page 468

19.6.19 RTC alarm B sub second register (RTC_ALRMBSSR)

The RTC_ALRMBSSR register is available only in high and medium+ density devices.

Address offset: 0x48

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
Reserved				MASKSS[3:0]				Reserved							
r	r	r	r	rw	rw	rw	rw	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	SS[14:0]														
r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bits 31:28 Reserved

Bits 27:24 **MASKSS[3:0]**: Mask the most-significant bits starting at this bit

0x0: No comparison on sub seconds for Alarm B. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).
0x1: SS[14:1] are don't care in Alarm B comparison. Only SS[0] is compared.
0x2: SS[14:2] are don't care in Alarm B comparison. Only SS[1:0] are compared.
0x3: SS[14:3] are don't care in Alarm B comparison. Only SS[2:0] are compared.
...
0xC: SS[14:12] are don't care in Alarm B comparison. SS[11:0] are compared.
0xD: SS[14:13] are don't care in Alarm B comparison. SS[12:0] are compared.
0xE: SS[14] is don't care in Alarm B comparison. SS[13:0] are compared.
0xF: All 15 SS bits are compared and must match to activate alarm.
The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved

Bits 14:0 **SS[14:0]**: Sub seconds value

This value is compared with the contents of the synchronous prescaler's counter to determine if Alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.

Note: This register can be written only when ALRBIE is reset in RTC_CR register, or in initialization mode.

This register is write protected.The write access procedure is described in Section : RTC register write protection

19.6.20 RTC backup registers (RTC_BKPxR)

Address offset: 0x50 to 0x60 (value line devices)

Address offset: 0x50 to 0x9C (low and medium density devices)

Address offset: 0x50 to 0xCC (high and medium+ density devices)

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
BKP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BKP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bits 31:0 BKP[31:0]

The application can write or read data to and from these registers. They are not powered-on when V_{DD} is switched off. They are not reset by System reset and their contents remain valid when the device operates in low power mode. This register is reset on a tamper detection event, as long as TAMPxF=1, or when the Flash readout protection is disabled.

19.6.21 RTC register map

Table 77. RTC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**RTC_TR**	Reserved									PM	HT [1:0]		HU[3:0]				Reserved	MNT[2:0]			MNU[3:0]				Reserved	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**	Reserved								YT[3:0]				YU[3:0]				WDU[2:0]			MT	MU[3:0]				Reserved		DT [1:0]		DU[3:0]			
	Reset value																	0	0	1	0	0	0	0	1			0	0	0	0	0	1
0x08	**RTC_CR**	Reserved								COE	OSEL [1:0]		POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBIE	ALRAE	DCE	FMT	BYPSHAD	REFCKON	TSEDGE	WCKSEL [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
0x0C	**RTC_ISR**	Reserved																TAMP3F	TAMP2F	TAMP1F	TSOVF	TSF	WUTF	ALRBF	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUTWF	ALRBWF	ALRAWF
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

Table 77. RTC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x10	RTC_PRER	Reserved									PREDIV_A[6:0]							Reserved	PREDIV_S[14:0]														
	Reset value										1	1	1	1	1	1	1		0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x14	RTC_WUTR	Reserved																WUT[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x18	RTC_CALIBR	Reserved																								DCS	Reserved		DC[4:0]				
	Reset value																									0			0	0	0	0	0
0x1C	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
0x20	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]				MSK2	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
0x24	RTC_WPR	Reserved																								KEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	RTC_SSR	Reserved																SS[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	RTC_SHIFTR	ADD1S	Reserved																SUBFS[14:0]														
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	RTC_TSTR	Reserved									PM	HT[1:0]		HU[3:0]				Reserved	MNT[2:0]			MNU[3:0]				Reserved	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	Reserved																WDU[2:0]			MT	MU[3:0]				Reserved		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	Reserved																SS[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	RTC_ CALR	Reserved																CALP	CALW8	CALW16	Reserved				CALM[8:0]								
	Reset value																	0	0	0					0	0	0	0	0	0	0	0	0

Table 77. RTC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x40	RTC_TAFCR	Reserved													ALARMOUTTYPE	Reserved		TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]			TAMPTS	TAMP3TRG	TAMP3E	TAMP2TRG	TAMP2E	TAMPIE	TAMP1ETRG	TAMP1E
	Reset value														0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	RTC_ ALRMASSR	Reserved				MASKSS[3:0]				Reserved									SS[14:0]														
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	RTC_ ALRMBSSR	Reserved				MASKSS[3:0]				Reserved									SS[14:0]														
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50 to 0xCC	RTC_BKP0R	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
	to RTC_BKP31R	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 *Table 2 on page 42* for the register boundary addresses.

20 Independent watchdog (IWDG)

20.1 IWDG introduction

The STM32L1xxxx have two embedded watchdog peripherals which offer a combination of high safety level, timing accuracy and flexibility of use. Both watchdog peripherals (Independent and Window) serve to detect and resolve malfunctions due to software failure, and to trigger system reset or an interrupt (window watchdog only) when the counter reaches a given timeout value.

The independent watchdog (IWDG) is clocked by its own dedicated low-speed clock (LSI) and thus stays active even if the main clock fails. The window watchdog (WWDG) clock is prescaled from the APB1 clock and has a configurable time-window that can be programmed to detect abnormally late or early application behavior.

The IWDG is best suited to applications which require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints. The WWDG is best suited to applications which require the watchdog to react within an accurate timing window. For further information on the window watchdog, refer to *Section 21 on page 512*.

20.2 IWDG main features

- Free-running downcounter
- clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Reset (if watchdog activated) when the downcounter value of 0x000 is reached

20.3 IWDG functional description

Figure 164 shows the functional blocks of the independent watchdog module.

When the independent watchdog is started by writing the value 0xCCCC in the Key register (IWDG_KR), the counter starts counting down from the reset value of 0xFFF. When it reaches the end of count value (0x000) a reset signal is generated (IWDG reset).

Whenever the key value 0xAAAA is written in the IWDG_KR register, the IWDG_RLR value is reloaded in the counter and the watchdog reset is prevented.

20.3.1 Hardware watchdog

If the “Hardware watchdog” feature is enabled through the device option bits, the watchdog is automatically enabled at power-on, and will generate a reset unless the Key register is written by the software before the counter reaches end of count.

20.3.2 Register access protection

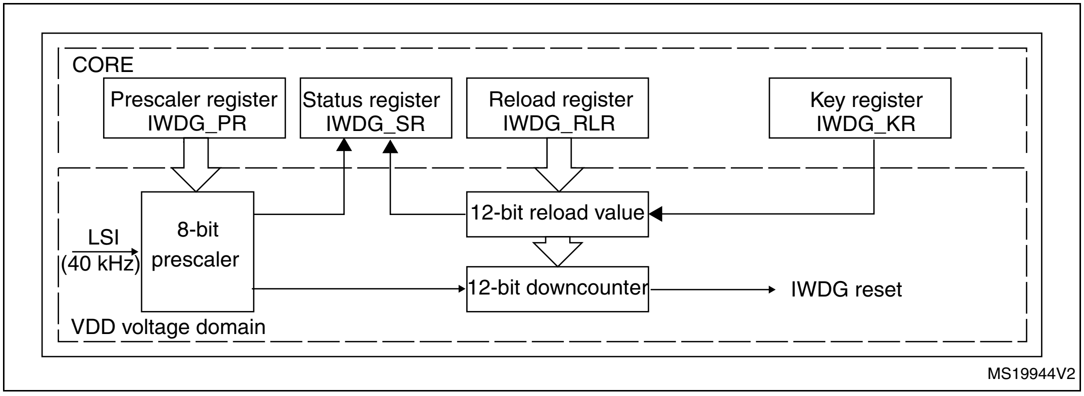
Write access to the IWDG_PR and IWDG_RLR registers is protected. To modify them, you must first write the code 0x5555 in the IWDG_KR register. A write access to this register with a different value will break the sequence and register access will be protected again. This implies that it is the case of the reload operation (writing 0xAAAA).

A status register is available to indicate that an update of the prescaler or the down-counter reload value is on going.

20.3.3 Debug mode

When the microcontroller enters debug mode (Cortex™-M3 core halted), the IWDG counter either continues to work normally or stops, depending on DBG_IWDG_STOP configuration bit in DBG module. For more details, refer to *Section 29.16.2: Debug support for timers, watchdog and I2C*.

Figure 164. Independent watchdog block diagram



Note: The watchdog function is implemented in the V_{DD} voltage domain that is still functional in Stop and Standby modes.

Table 78. Min/max IWDG timeout period at 37 kHz (LSI) [(1)]

Prescaler divider	PR[2:0] bits	Min timeout (ms) RL[11:0]= 0x000	Max timeout (ms) RL[11:0]= 0xFFF
/4	0	0.108	442.81
/8	1	0.216	885.621
/16	2	0.432	1771.243
/32	3	0.864	3542.486
/64	4	1.729	7084.972
/128	5	3.459	14169.945
/256	6	6.918	28339.891

1. These timings are given for a 37 kHz clock but the microcontroller’s internal RC frequency can vary from 30 to 60 kHz. Moreover, given an exact RC oscillator frequency, the exact timings still depend on the phasing of the APB interface clock versus the LSI clock so that there is always a full RC period of uncertainty.

20.4 IWDG registers

Refer to for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16 bits) or words (32 bits).

20.4.1 Key register (IWDG_KR)

Address offset: 0x00

Reset value: 0x0000 0000 (reset by Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																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 0000h)

These bits must be written by software at regular intervals with the key value AAAAh, otherwise the watchdog generates a reset when the counter reaches 0.

Writing the key value 5555h to enable access to the IWDG_PR and IWDG_RLR registers (see *Section 20.3.2*)

Writing the key value CCCCh starts the watchdog (except if the hardware watchdog option is selected)

20.4.2 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	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													PR[2:0]		
																													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **PR[2:0]:** Prescaler divider

These bits are write access protected seeSection 20.3.2. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of IWDG_SR must be reset in order to be able to change the prescaler divider.

000: divider /4
001: divider /8
010: divider /16
011: divider /32
100: divider /64
101: divider /128
110: divider /256
111: divider /256

Note: Reading this register returns the prescaler value from the VDD 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_SR register is reset.

20.4.3 Reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF (reset by Standby mode)

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

Bits11:0 **RL[11:0]:** Watchdog counter reload value

These bits are write access protected see Section 20.3.2. They are written by software to define the value to be loaded in the watchdog counter each time the value AAAAh is written in the IWDG_KR register. The watchdog counter counts down from this value. The timeout period is a function of this value and the clock prescaler. Refer to *Table 78.*

The RVU bit in the IWDG_SR register must be reset in order to be able to change the reload value.

Note: Reading this register returns the reload value from the VDD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing on this register. For this reason the value read from this register is valid only when the RVU bit in the IWDG_SR register is reset.

20.4.4 Status register (IWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000 (not reset by Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														RVU	PVU
																														r	r

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

Bit 1 **RVU:** Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain (takes up to 5 RC 40 kHz cycles).

Reload value can be updated only when RVU bit is reset.

Bit 0 **PVU:** Watchdog prescaler value update

This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V_{DD} voltage domain (takes up to 5 RC 40 kHz cycles).

Prescaler value can be updated only when PVU bit is reset.

Note: If several reload values or prescaler values are used by application, it is mandatory to wait until RVU bit is reset before changing the reload value and to wait until PVU bit is reset before changing the prescaler value. However, after updating the prescaler and/or the reload value it is not necessary to wait until RVU or PVU is reset before continuing code execution (even in case of low-power mode entry, the write operation is taken into account and will complete)

20.4.5 IWDG register map

The following table gives the IWDG register map and reset values.

Table 79. IWDG register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	IWDG_KR	Reserved																KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	IWDG_PR	Reserved																													PR[2:0]		
	Reset value																														0	0	0
0x08	IWDG_RLR	Reserved																				RL[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x0C	IWDG_SR	Reserved																														RVU	PVU
	Reset value																															0	0

Refer to *Table 2 on page 42* for the register boundary addresses.

21 Window watchdog (WWDG)

21.1 WWDG introduction

The window watchdog is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the contents of the downcounter before the T6 bit becomes cleared. An MCU reset is also generated if the 7-bit downcounter value (in the control register) is refreshed before the downcounter has reached the window register value. This implies that the counter must be refreshed in a limited window.

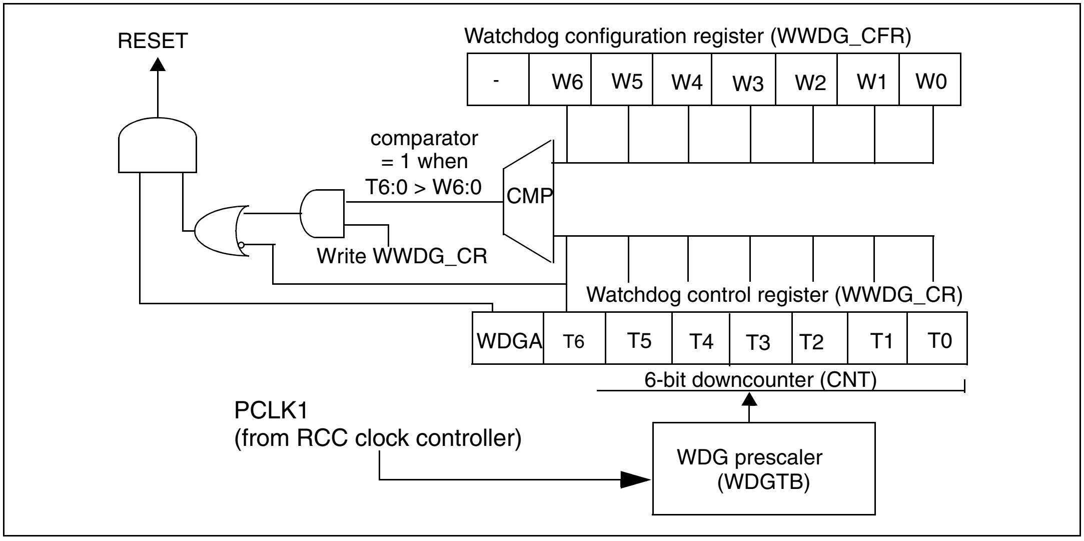
21.2 WWDG main features

- Programmable free-running downcounter
- Conditional reset
  - Reset (if watchdog activated) when the downcounter value becomes less than 0x40
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window (see *Figure 166*)
- Early wakeup interrupt (EWI): triggered (if enabled and the watchdog activated) when the downcounter is equal to 0x40.

21.3 WWDG functional description

If the watchdog is activated (the WDGA bit is set in the WWDG_CR register) and when the 7-bit downcounter (T[6:0] bits) rolls over 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.

Figure 165. Watchdog block diagram



The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent an MCU reset. This operation must occur only when the counter value is lower than the window register value. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0:

Enabling the watchdog

The watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again except by a reset.

Controlling the downcounter

This downcounter is free-running: It counts 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 which represents 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 166*).The Configuration register (WWDG_CFR) contains the high limit of the window: To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 0x3F. *Figure 166* 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).

Advanced watchdog interrupt feature

The Early Wakeup Interrupt (EWI) can be used if specific safety operations or data logging must be performed before the actual reset is generated. The EWI interrupt is enabled by setting the EWI bit in the WWDG_CFR register. When the downcounter reaches the value 0x40, an EWI interrupt is generated and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case, the corresponding interrupt service routine (ISR) should reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The EWI interrupt is cleared by writing '0' to the EWIF bit in the WWDG_SR register.

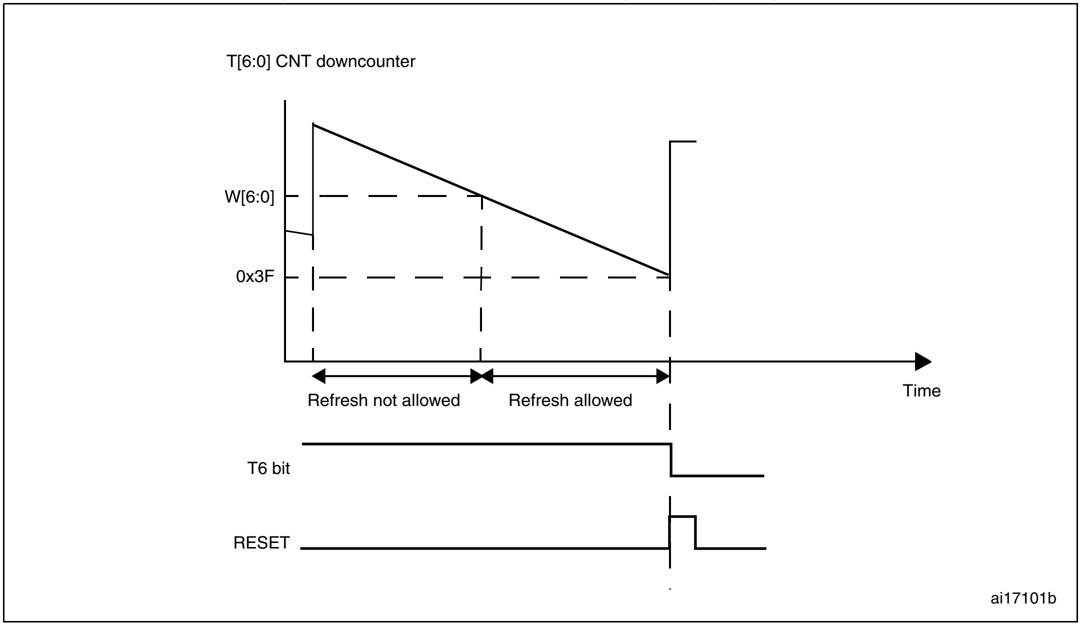
Note: When the EWI interrupt cannot be served, e.g. due to a system lock in a higher priority task, the WWDG reset will eventually be generated.

21.4 How to program the watchdog timeout

You can use the formula in *Figure 166* 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 166. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$t_{WWDG} = t_{PCLK1} \times 4096 \times 2^{WDGTB} \times (t[5:0] + 1) \qquad (ms)$$

where:

t_{WWDG}: WWDG timeout

t_{PCLK1}: APB1 clock period measured in ms

Refer to the table below for the minimum and maximum values of the $T_{WWDG.}$

Table 80. Min-max timeout value @32 MHz (f_{PCLK1})

Prescaler	WDGTB	Min timeout value	Max timeout value
1	0	128 µs	8.19 ms
2	1	256 µs	16.38 ms
4	2	512 µs	32.67 ms
8	3	1024 µs	65.54 ms

21.5 Debug mode

When the microcontroller enters debug mode (Cortex™-M3 core halted), the WWDG counter either continues to work normally or stops, depending on DBG_WWDG_STOP configuration bit in DBG module. For more details, refer to *Section 29.16.2: Debug support for timers, watchdog and I2C*.

21.6 WWDG registers

Refer to *Section 1.1 on page 37* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16 bits) or words (32 bits).

21.6.1 Control register (WWDG_CR)

Address offset: 0x00

Reset value: 0x0000 007F

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								WDGA	T[6:0]						
								rs	rw						

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

Bit 7 **WDGA:** Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Bits 6:0 **T[6:0]:** 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter. It is decremented every (4096 x 2^{WDGTB}) PCLK1 cycles. A reset is produced when it rolls over from 0x40 to 0x3F (T6 becomes cleared).

21.6.2 Configuration register (WWDG_CFR)

Address offset: 0x04

Reset value: 0x0000 007F

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

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

Bit 9 **EWI:** Early wakeup interrupt

When set, an interrupt occurs whenever the counter reaches the value 0x40. This interrupt is only cleared by hardware after a reset.

Bits 8:7 **WDGTB[1:0]:** Timer base

The time base of the prescaler can be modified as follows:
00: CK Counter Clock (PCLK1 div 4096) div 1
01: CK Counter Clock (PCLK1 div 4096) div 2
10: CK Counter Clock (PCLK1 div 4096) div 4
11: CK Counter Clock (PCLK1 div 4096) div 8

Bits 6:0 **W[6:0]:** 7-bit window value

These bits contain the window value to be compared to the downcounter.

21.6.3 Status register (WWDG_SR)

Address offset: 0x08
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															EWIF
															rc_w0

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

Bit 0 **EWIF:** Early wakeup interrupt flag

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing ‘0. A write of ‘1 has no effect. This bit is also set if the interrupt is not enabled.

21.6.4 WWDG register map

The following table gives the WWDG register map and reset values.

Table 81. WWDG register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	WWDG_CR	Reserved																								WDGA	T[6:0]						
	Reset value																									0	1	1	1	1	1	1	1
0x04	WWDG_CFR	Reserved																						EWI	WDGTB1	WDGTB0	W[6:0]						
	Reset value																							0	0	0	1	1	1	1	1	1	1
0x08	WWDG_SR	Reserved																															EWIF
	Reset value																																0

Refer to *Table 2 on page 42* for the register boundary addresses.

22 Advanced encryption standard hardware accelerator (AES)

This section applies to STM32L162xx devices only.

22.1 Introduction

The AES hardware accelerator can be used to both encipher and decipher data using AES algorithm. It is a fully compliant implementation of the following standard:

- The advanced encryption standard (AES) as defined by Federal Information Processing Standards Publication (FIPS PUB 197, 2001 November 26)

The accelerator encrypts and decrypts 128-bit blocks using 128-bit key length. It can also perform key derivation. The encryption or decryption key is stored in an internal register in order to minimize write operations by the CPU or DMA when processing several data blocks using the same key.

By default, Electronic codebook mode (ECB) is selected. Cipher block chaining (CBC) or Counter (CTR) mode) chaining algorithms are also supported by the hardware.

The AES supports DMA transfer for incoming and for outcoming data (2 DMA channels required).

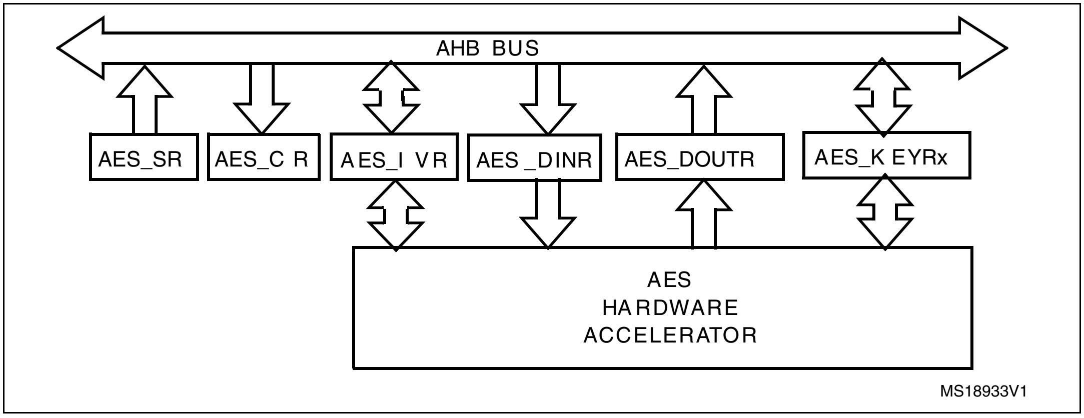
22.2 AES main features

- Encryption/Decryption using AES Rijndael Block Cipher algorithm
- NIST FIPS 197 compliant implementation of AES encryption/decryption algorithm
- Internal 128-bit register for storing the encryption or derivation key (4x 32-bit registers)
- Electronic codebook (ECB), Cipher block chaining (CBC), and Counter mode (CTR) supported
- Key scheduler
- Key derivation for decryption
- 128-bit data block processing
- 128-bit key length
- 213 clock cycles to encrypt or decrypt one 128-bit block (including the input and output phases)
- 1x32-bit INPUT buffer and 1x32-bit OUTPUT buffer.
- Register access supporting 32-bit data width only.
- One 128-bit Register for the initialization vector when AES is configured in CBC mode or for the 32-bit counter initialization when CTR mode is selected.
- Automatic data flow control with support of direct memory access (DMA) using 2 channels, one for incoming data, and one for outcoming data.

22.3 AES functional description

Figure 167 shows the block diagram of the AES accelerator.

Figure 167. Block diagram



The AES accelerator processes data blocks of 128-bits (4 words) using a key with a length of 128 bits, and an initialization vector when CBC or CTR chaining mode is selected.

It provides 4 operating modes:

- Mode 1: Encryption using the encryption key stored in the AES_KEYRx registers.
- Mode 2: Key Derivation stored internally in the AES_KEYRx registers at the end of the key derivation processed from the encryption key stored in this register before enabling the AES. This mode is independent from the AES chaining mode selection.
- Mode 3: Decryption using a given (pre-computed) decryption key stored in the AES_KEYRx registers.
- Mode 4: Key Derivation + Decryption using an encryption key stored in the AES_KEYRx registers (not used when the AES is configured in Counter mode for perform a chaining algorithm).

The operating mode is selected by programming bits MODE[1:0] into the AES_CR register. The mode must be changed only when the AES is disabled (bit EN=0 in the AES_CR register). The KEY registers (AES_KEYRx) must be stored before enabling the AES.

To select which one of the ECB, CBC or CTR mode is going to be used for the cryptographic solution, it is mandatory to write the bit CHMOD[1:0] of the AES_CR register and the AES_IVR register (only used for the CBC and CTR chaining modes) when the AES is disabled (bit EN =0 in the AES_CR register).

Once enabled (bit EN=1), the AES is in the input phase, waiting for the software to write the input data words into the AES_DINR (4 words) for the modes 1, 3 or 4. The data corresponds either to the plaintext message or the cipher message. A wait cycle is automatically inserted between two consecutive writes to the AES_DINR register in order to send, interleaved with the data, the key to the AES processor.

For mode 2, the key derivation processing is started immediately after the EN bit in the AES_CR register is set. It requires that the AES_KEYRx registers are loaded with the encrypted KEY before enabling the AES. At the end of the Key derivation processing (CCF flag is set), the derivative key is available in the AES_KEYRx registers and the AES is disabled by hardware. In this mode, the AES_KEYRx registers must not be read when AES is enabled and until the CCF flag is set to 1 by hardware.

The status flag CCF (Computation Complete Flag) in the AES_SR register is set once the computation phase is complete. An interrupt can be generated if bit CCFIE=1 in the AES_CR register. The software can then read back the data from the AES_DOUTR register (for modes 1, 3, 4) or from the AES_KEYRx registers (if mode 2 is selected).

The flag CCF has no meaning when DMAOUTEN = 1 in the AES_CR register, because the reading the AES_DOUTR register is managed by DMA automatically without any software action at the end of the computation phase.

The operation ends with the output phase, during which the software reads successively the 4 output data words from the AES_DOUTR register in mode 1, 3 or 4. In mode 2 (key derivation mode), the data is automatically stored in the AES_KEYRx registers and the AES is disabled by hardware. Then, software can select mode 3 (decryption mode) before it enables the AES to start the decryption using this derivative key.

During the input and output phases, the software must read or write the data bytes successively (except in mode 2) but the AES is tolerant of any delays occurring between each read or write operation (example: if servicing another interrupt at this time).

The RDERR and WRERR flags in the AES_SR register are set when an unexpected read or write operation is detected. An interrupt can be generated if the ERRIE bit is set in the AES_CR register. AES is not disabled after an error detection and continues processing as normal.

It is also possible to use the general purpose DMA to write the input words and to read the output words (refer to *Figure 182* and *Figure 183*).

The AES can be re-initialized at any moment by resetting the EN bit in the AES_CR register. Then the AES can be re-started from the beginning by setting EN=1, waiting for the first input data byte to be written (except in mode 2 where Key derivation processing starts as soon as the EN bit is set, starting from the value stored in the AES_KEYRx registers).

22.4 Encryption and derivation keys

The AES_KEYRx registers are used to store the encryption or decryption keys. These four registers are organized in little-endian configuration: Register AES_KEYR0 has to be loaded with the 32-bit LSB of the key. Consequently, AES_KEYR3 has to be loaded with the 32-bit MSB of the 128-bit key.

The key for encryption or decryption must be stored in these registers when the AES is disabled (EN = 0 into the AES_CR register). Their endianess are fixed.

In mode 2 (key derivation), the AES_KEYRx needs to be loaded with the encryption key. Then, the AES has to be enabled. At the end of the computation phase, the derivation key is stored automatically in the AES_KEYRx registers, overwriting the previous encryption key. The AES is disabled by hardware when the derivation key is available. If the software needs to switch the AES to mode 3 (decryption mode), there is no need to write the AES_KEYRx registers if their content corresponds to the derivation key (previously computed by mode 2).

In mode 4 (key derivation + decryption), the AES_KEYRx registers contain only the encryption key. The derivation key is calculated internally without any write to these registers.

22.5 AES chaining algorithms

Three algorithms are supported by the AES hardware and can be selected through the CHMOD[1:0] bits in the AES_CR register when the AES is disabled (bit EN = 0):

- Electronic CodeBook (ECB)
- Cipher Block Chaining (CBC)
- Counter Mode (CTR)

22.5.1 Electronic CodeBook (ECB)

This is the default mode. This mode doesn't use the AES_IVR register. There are no chaining operations. The message is divided into blocks and each block is encrypted separately.

Figure 168 and *Figure 169* describe the principle of the Electronic Codebook algorithm for encryption and decryption respectively.

Figure 168. ECB encryption mode

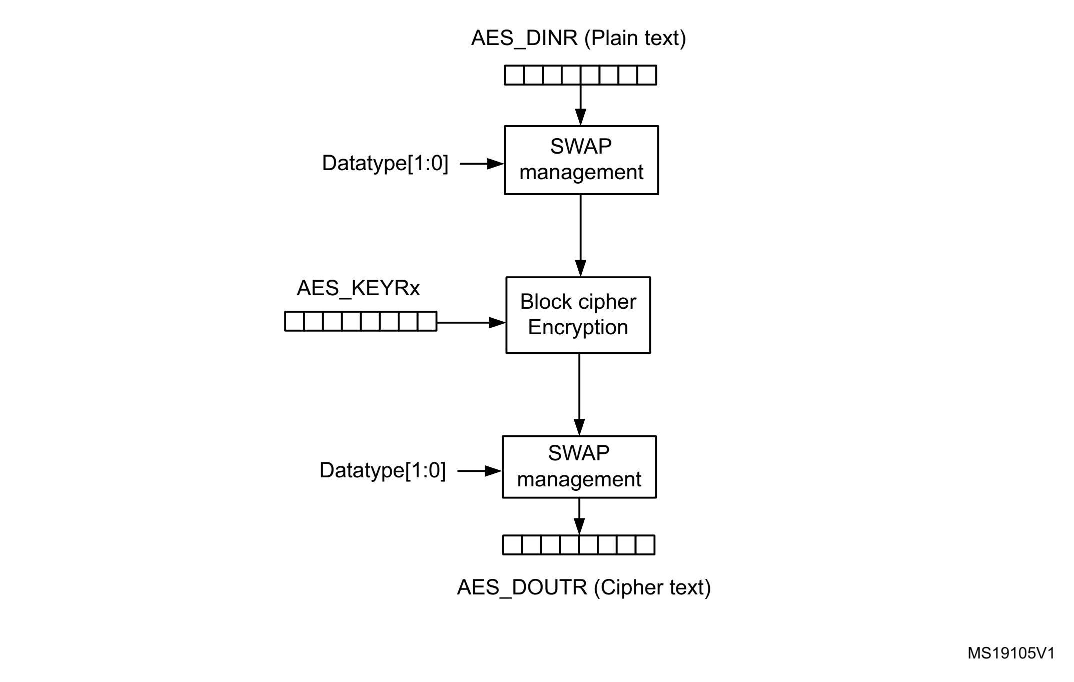
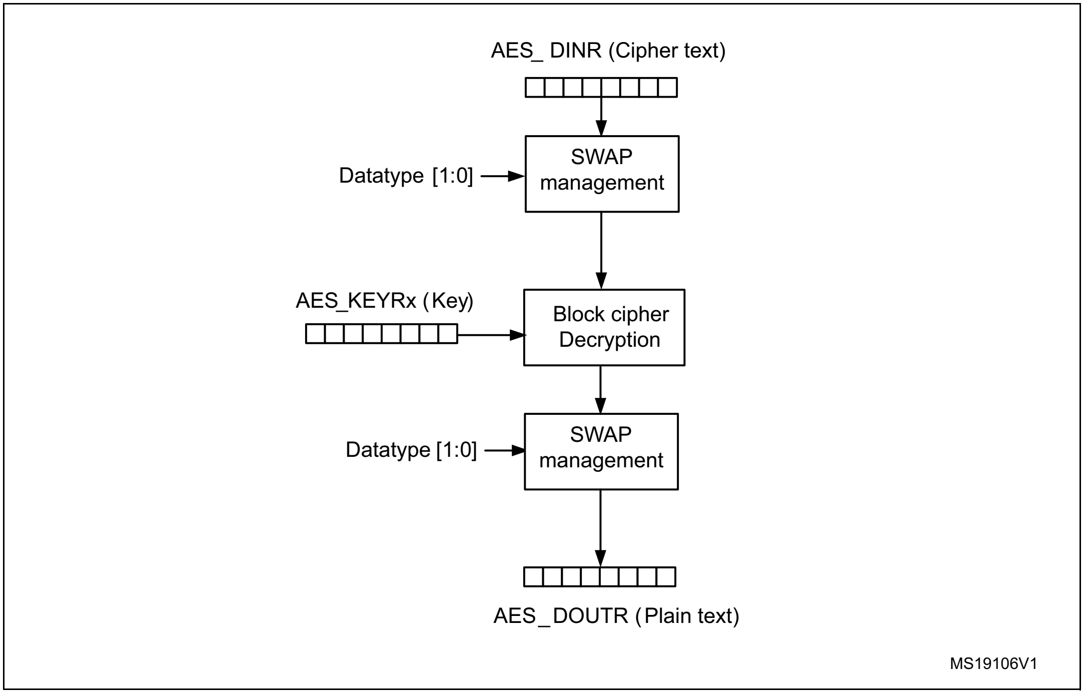


Figure 169. ECB decryption mode



22.5.2 Cipher block chaining (CBC)

In cipher-block chaining (CBC) mode, each block of plain text is XORed with the previous cipher text block before being encrypted. To make each message unique, an initialization vector (AES_IVRx) is used during the first block processing.

The initialization vector is XORed after the swapping management block in during encryption mode and before it in decryption mode (refer to *Figure 170* and *Figure 171*).

Figure 170. CBC mode encryption

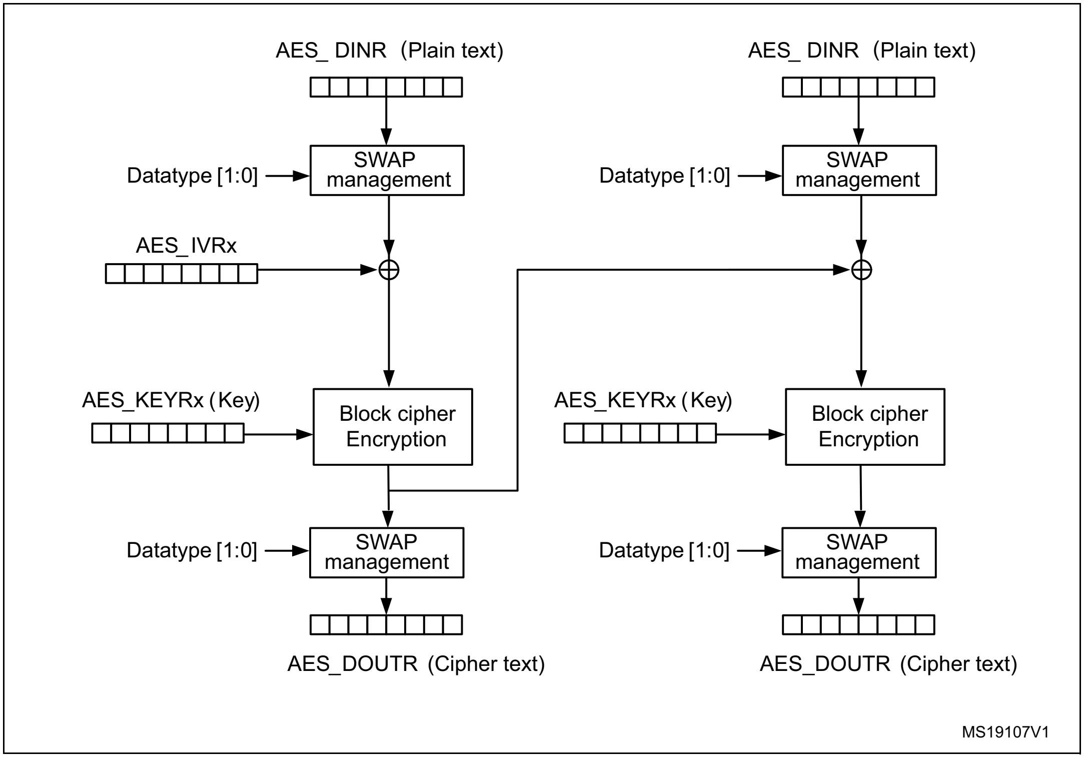
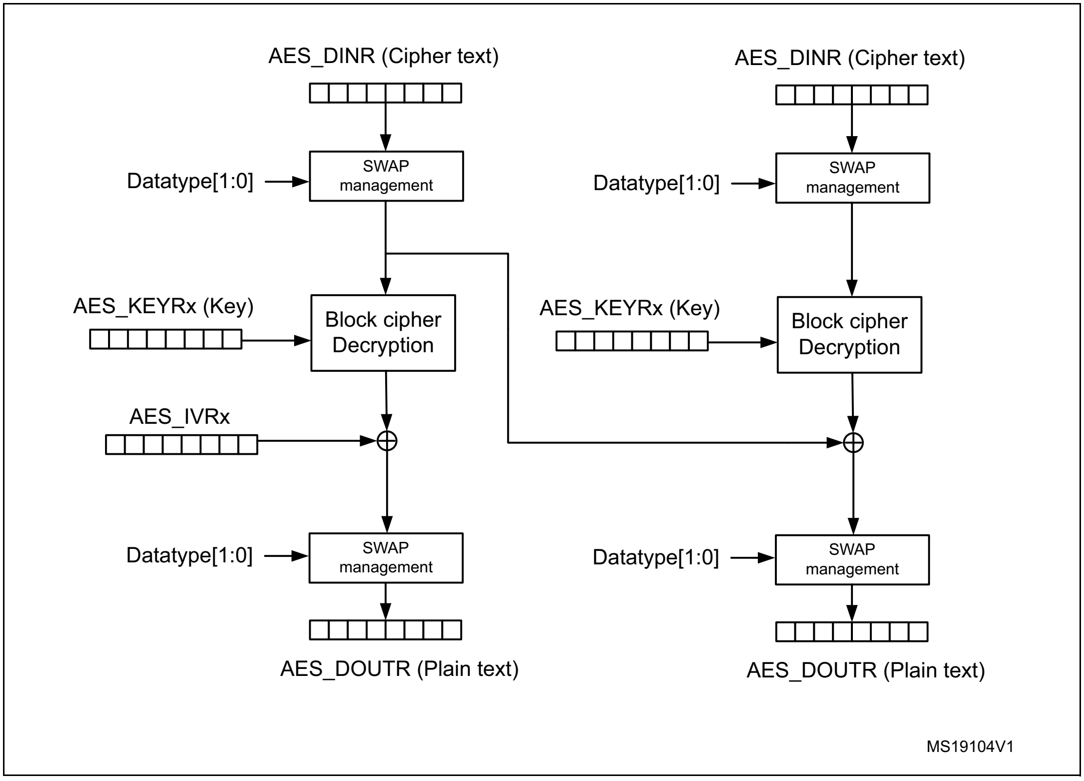


Figure 171. CBC mode decryption



Note: When the AES is enabled, reading the AES_IVR returns the value 0x00000000.

Suspended mode for a given message

It is possible to suspend a message if another message with a higher priority needs to be processed. At the end of sending of this highest priority message, the suspended message may be resumed in both encryption or decryption mode. This feature is available only when the data transfer is done by CPU accesses to the AES_DOUTR and AES_DINR registers. It is advised to not use it when the DMA controller is managing the data transfer.

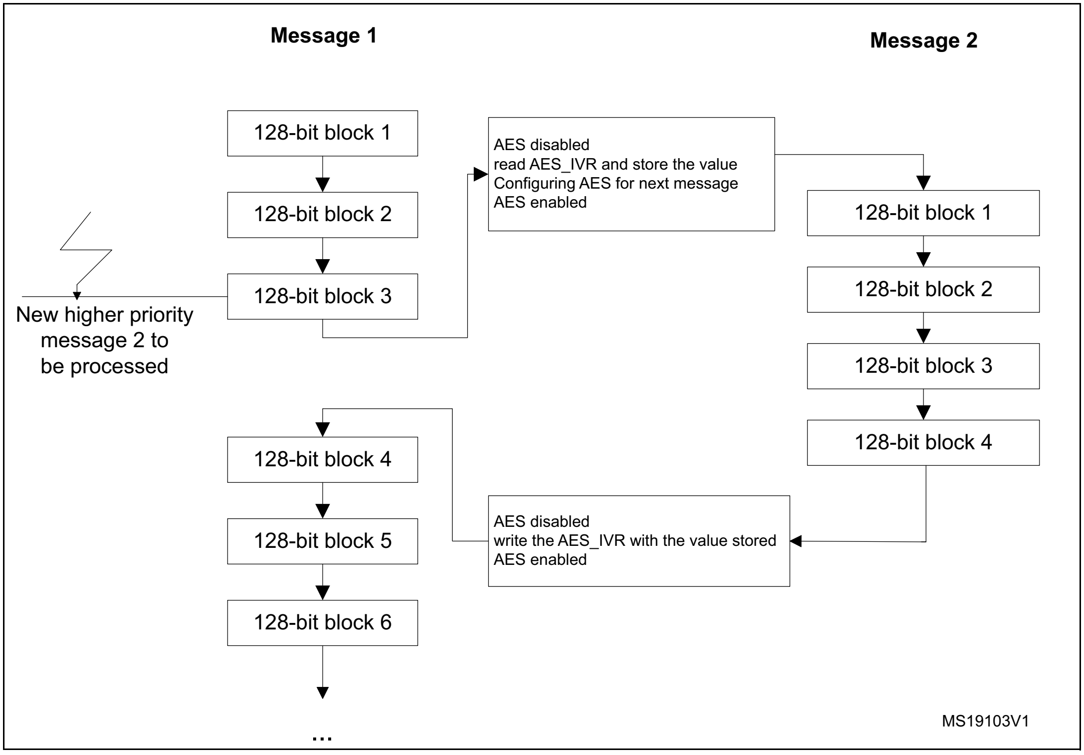
For correct operation, the message must be suspended at the end of processing a block (after the fourth read of the AES_DOUTR register and before the next AES_DINR write access corresponding to the input of the next block to be processed).

The AES should be disabled writing bit EN = 0 in the AES_CR register. The software has to read the AES_IVRx which contains the latest value to be used for the chaining XOR operation before message interruption. This value has to be stored for reuse by writing the AES_IVRx registers as soon as the interrupted message has to be resumed (when AES is disabled). It should be noted that this does not break the chaining operation and the message processing can be resumed as soon as the AES is enabled again to send the next 128-bit data block.

This behavior is valid whatever the AES configuration (encryption or decryption mode).

Figure 172 gives an example of a message 1 which is suspended in order to send a higher priority message 2, shorter than message 1. At the end of the 128-bit block processing, AES is disabled. The AES_IVR register is read back to store the value to be retrieved later on when the message is resumed, in order not to break the chaining operation. Then, the AES is configured to send message 2 and it is enabled to start processing. At the end of message 2 processing, AES has to be disabled again and the AES_IVRx registers have to be loaded with the value previously stored when the message 1 was interrupted. Then software has to restart from the input value corresponding to block 4 as soon as AES is enabled to resume message 1.

Figure 172. Example of suspend mode management



22.5.3 Counter Mode (CTR)

In counter mode, a 32-bit counter is used in addition to a nonce value for the XOR operation with the cipher text or plain text (refer to *Figure 173* and *Figure 174*).

Figure 173. CTR mode encryption

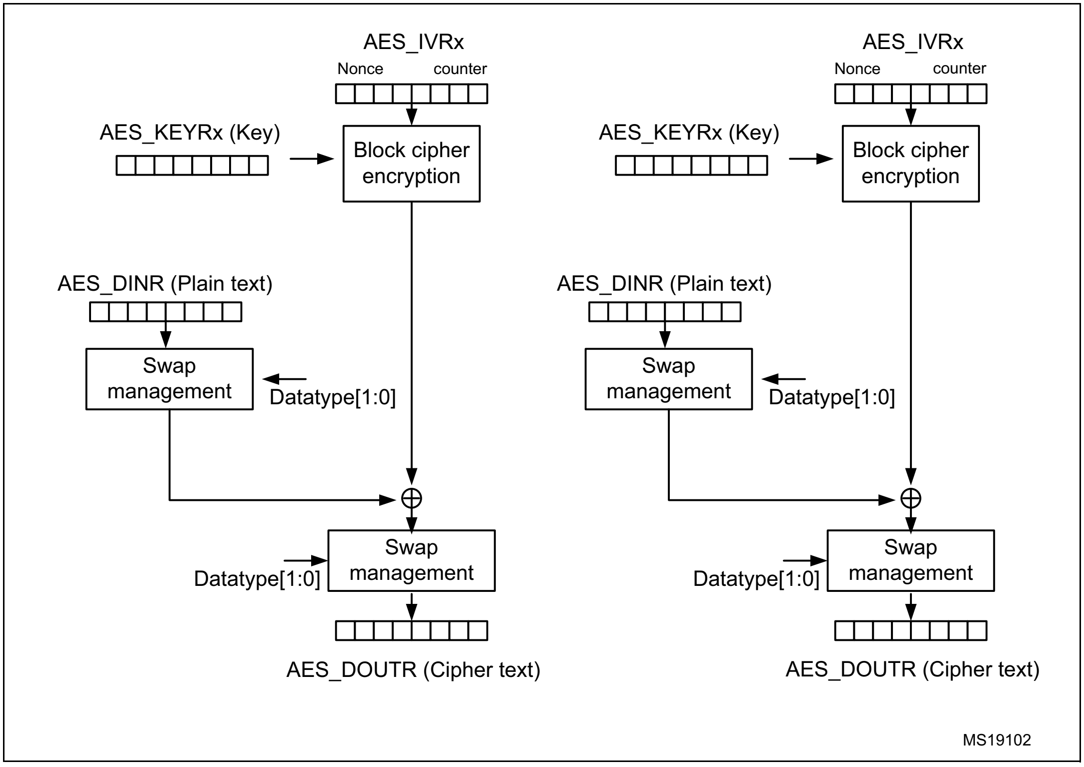
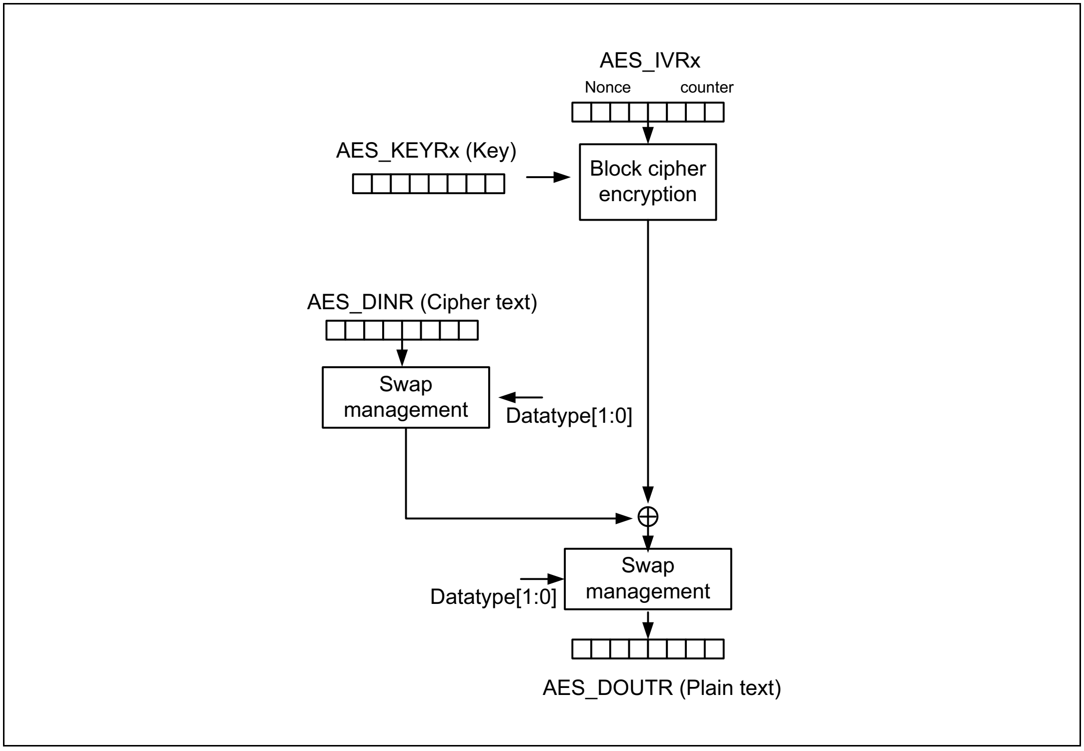
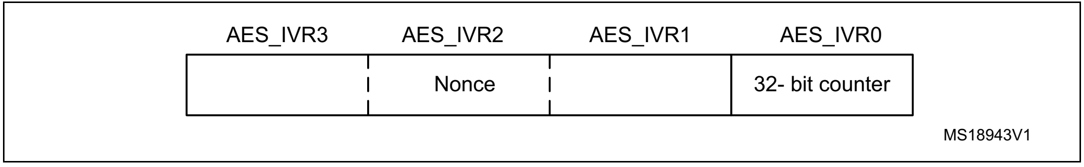


Figure 174. CTR mode decryption



The nonce value and 32-bit counter are accessible through the AES_IVRx register and organized like below in *Figure 175*:

Figure 175. 32-bit counter + nonce organization



In Counter Mode, the counter is incremented from the initialized value for each block to be processed in order to guarantee a unique sequence which is not repeated for a long time. It is a 32-bit counter, meaning that the nonce message is kept to the initialized value stored when the AES was disabled. Only the 32-bit LSB of the 128-bit initialization vector register represents the counter. In contrast to CBC mode (which uses the AES_IVRx registers only once when processing the first data block), in Counter mode, the AES_IVRx registers are used for processing each data block.

In counter mode, key derivation+decryption mode is not applicable.

Note: The AES_IVRx register has be written only when the AES is disabled (bit EN = 0) to guarantee good AES behavior.

Reading it while AES is enabled returns the value 0x00000000.

Reading it while the AES is disabled returns the latest counter value (useful for managing suspend mode).

In CTR mode, key derivation + decryption serves no purpose. Consequently it is forbidden to set MODE[1:0] = 11 in the AES_CR register and any attempt to set this configuration is forced to MODE[1:0] = 10 (which corresponds to CTR mode decryption). This uses the encryption block of the AES processor to decipher the message as shown in *Figure 174*).

Suspend mode in CTR mode

Like for the CBC mode, it is possible to interrupt a message, sending a higher priority message and resume the message which was interrupted. Refer to the *Figure 172* and *Chapter 22.5.2* for more details about the suspend mode capability.

22.6 Data type

Data are entered in the AES processor 32 bits at a time (words), by writing them in the AES_DINR register. AES handles 128-bit data blocks. The AES_DINR or AES_DOUTR registers must be read or written four times to handle one 128-bit data block with the MSB first.

The system memory organization is little-endian: whatever the data type (bit, byte, 16-bit half-word, 32-bit word) used, the less-significant data occupies the lowest address location.

Thus, there must be a bit, byte, or half-word swapping operation to be performed on data to be written in the AES_DINR from system memory before entering the AES processor, and the same swapping must be performed for AES data to be read from the AES_DOUTR register to the system memory, depending on to the kind of data to be encrypted or decrypted.

The DATATYPE bits in the AES_CR register offer different swap modes to be applied to the AES_DINR register before sending it to the AES processor and to be applied on the AES_DOUTR register on the data coming out from the processor (refer to *Figure 176*).

Note: The swapping operation concerns only the AES_DOUTR and AES_DINR registers. The AES_KEYRx and AES_IVRx registers are not sensitive to the swap mode selected. They have a fixed little-endian configuration (refer to Section 22.4 and Section 22.12).

Figure 176. 128-bit block construction according to the data type

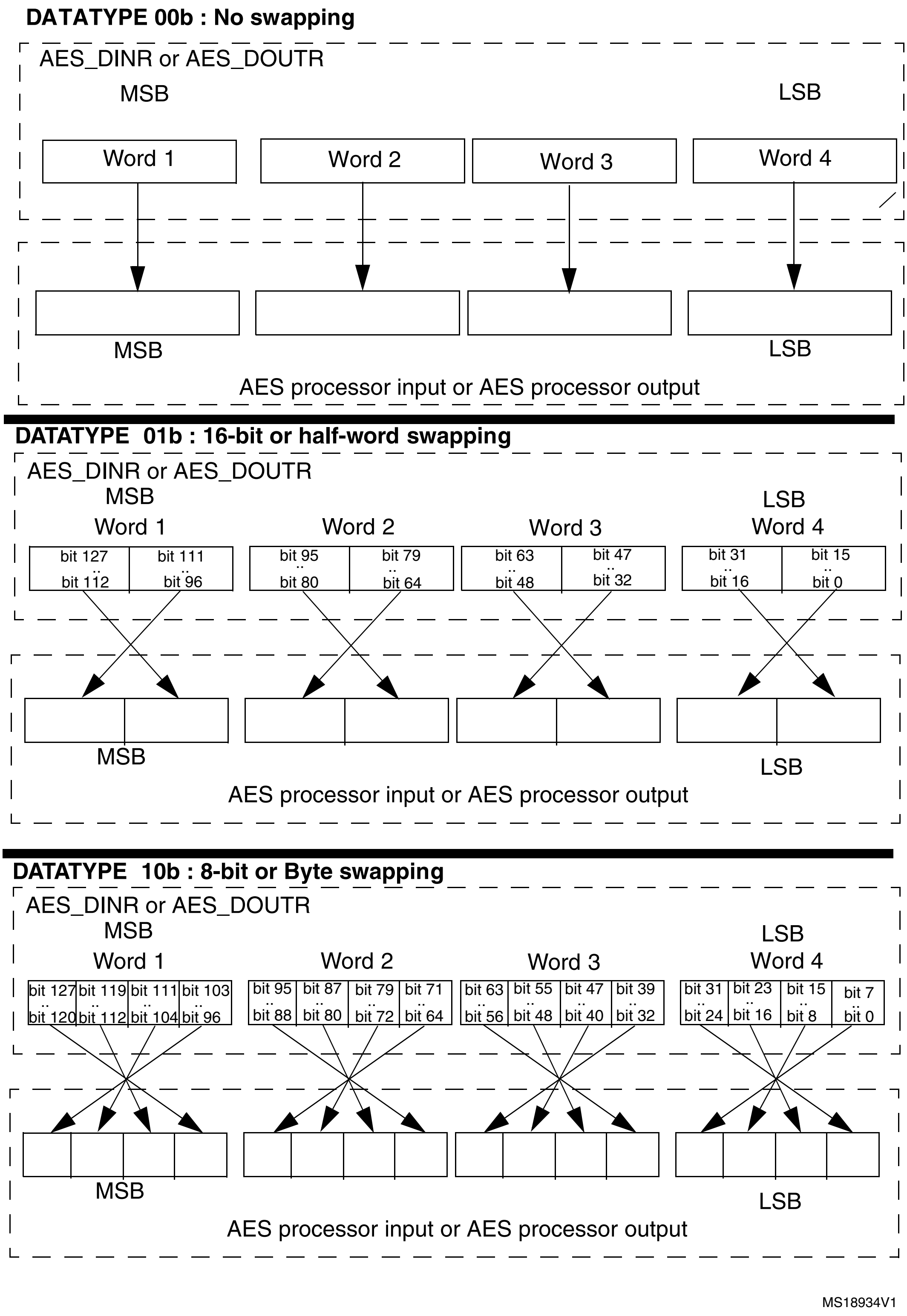
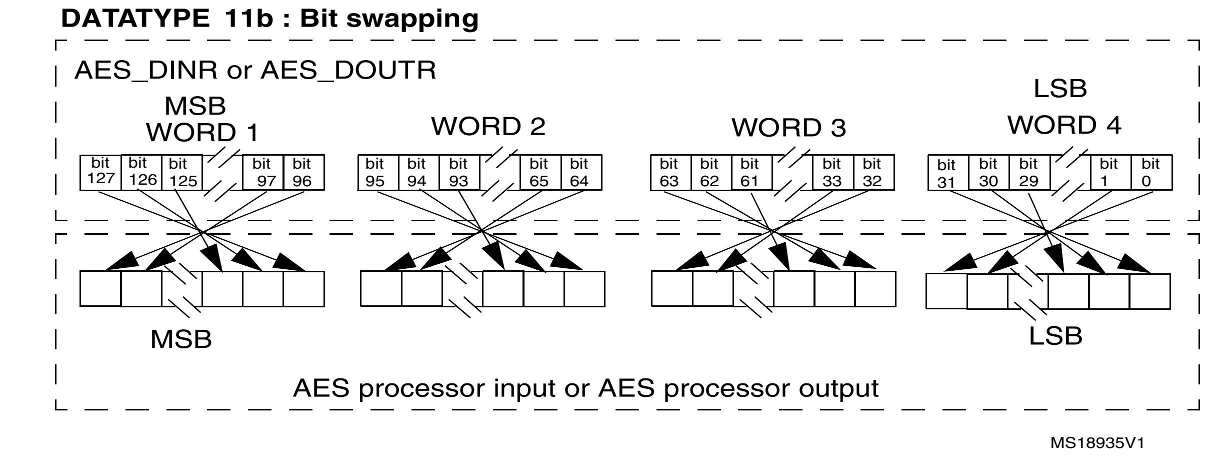


Figure 177. 128-bit block construction according to the data type (continued)

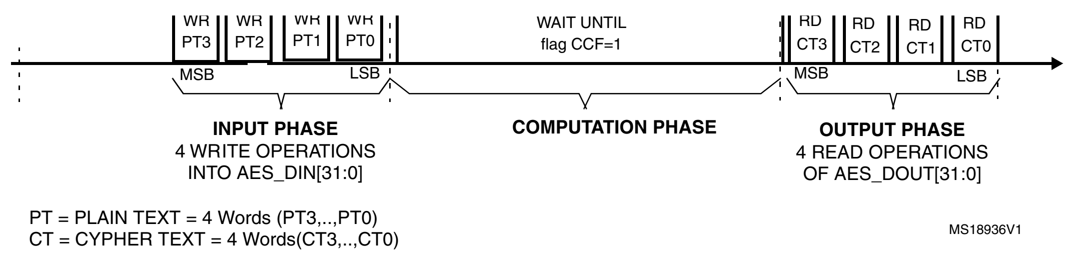


22.7 Operating modes

22.7.1 Mode 1: encryption

1. Disable the AES by resetting bit the EN bit in the AES_CR register.
2. Configure the Mode 1 by programming MODE[1:0]=00 in the AES_CR register and select which type of chaining mode needs to be performed by programming the CHMOD[1:0] bits.
3. Write the AES_KEYRx registers (128-bit encryption key) and the AES_IVRx registers if CTR or CBC mode is selected. For EBC mode, the AES_IVRx register is not used.
4. Enable the AES by setting the EN bit in the AES_CR register.
5. Write the AES_DINR register 4 times to input the plain text (MSB first) as shown in *Figure 178: Mode 1: encryption on page 531*.
6. Wait until the CCF flag is set in the AES_SR register.
7. Reads the AES_DOUTR register 4 times to get the cipher text (MSB first) as shown in *Figure 178: Mode 1: encryption on page 531*.
8. Repeat steps 5,6,7 to process all the blocks with the same encryption key.

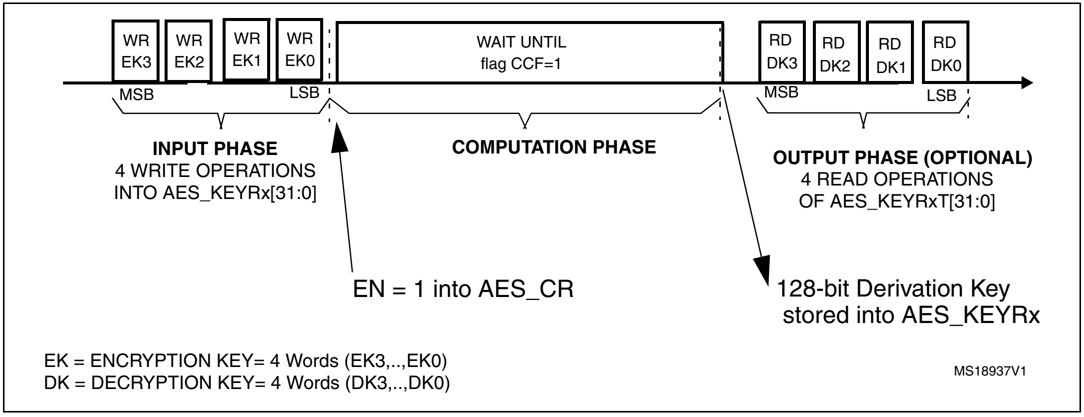
Figure 178. Mode 1: encryption


WR PT3
WR PT2
WR PT1
WR PT0
MSB
LSB
WAIT UNTIL flag CCF=1
RD CT3
RD CT2
RD CT1
RD CT0
MSB
LSB
INPUT PHASE
4 WRITE OPERATIONS INTO AES_DIN[31:0]
COMPUTATION PHASE
OUTPUT PHASE
4 READ OPERATIONS OF AES_DOUT[31:0]
PT = PLAIN TEXT = 4 Words (PT3,..,PT0)
CT = CYPHER TEXT = 4 Words(CT3,..,CT0)
MS18936V1

22.7.2 Mode 2: key derivation

1. Disable the AES by resetting the EN bit in the AES_CR register.
2. Configure Mode 2 by programming MODE[1:0]=01 in the AES_CR register. Note that the CHMOD[1:0] bits are not significant in this case because this key derivation mode is independent from the chaining algorithm selected.
3. Write the AES_KEYRx registers with the encryption key to obtain the derivative key. A write to the AES_IVRx has no effect.
4. Enable the AES by setting the EN bit in the AES_CR register.
5. Wait until the CCF flag is set in the AES_SR register.
6. The derivation key is put automatically into the AES_KEYRx registers. Read the AES_KEYRx register to obtain the decryption key if needed. The AES is disabled by hardware. To restart a derivation key calculation, repeat steps 3, 4, 5 and 6.

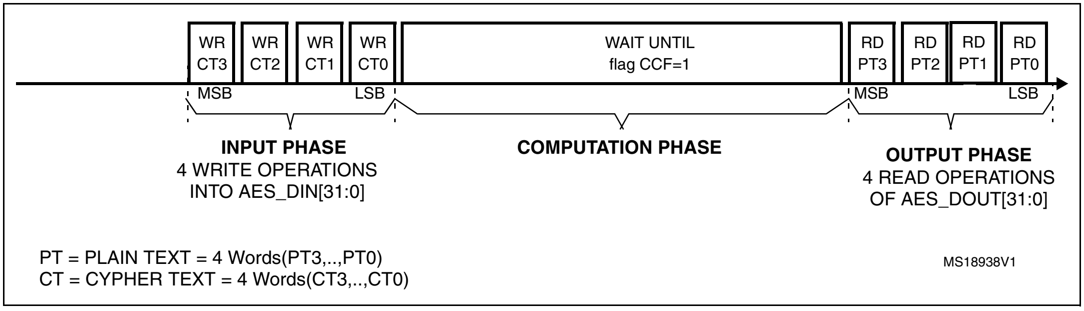
Figure 179. Mode 2: key derivation


WR EK3
WR EK2
WR EK1
WR EK0
WAIT UNTIL flag CCF=1
RD DK3
RD DK2
RD DK1
RD DK0
MSB
LSB
MSB
LSB
INPUT PHASE
4 WRITE OPERATIONS INTO AES_KEYRx[31:0]
COMPUTATION PHASE
OUTPUT PHASE (OPTIONAL)
4 READ OPERATIONS OF AES_KEYRxT[31:0]
EN = 1 into AES_CR
128-bit Derivation Key stored into AES_KEYRx
EK = ENCRYPTION KEY= 4 Words (EK3,..,EK0)
DK = DECRYPTION KEY= 4 Words (DK3,..,DK0)
MS18937V1

22.7.3 Mode 3: decryption

1. Disable the AES by resetting the EN bit in the AES_CR register.
2. Configure Mode 3 by programming MODE[1:0] =10 in the AES_CR register and select which type of chaining mode needs to be performed by programming the CHMOD[1:0] bits.
3. Write the AES_KEYRx registers with the decryption key (this step can be bypassed if the derivation key is already stored in the AES_KEYRx registers using mode 2: key derivation). Write the AES_IVRx registers if CTR or CBC mode is selected. For EBC mode, the AES_IVRx registers are not used.
4. Enable the AES by setting the EN bit in the AES_CR register.
5. Write the AES_DINR register 4 times to input the cipher text (MSB first) as shown in *Figure 180: Mode 3: decryption on page 533*.
6. Wait until the CCF flag is set in the AES_SR register.
7. Read the AES_DOUTR register 4 times to get the plain text (MSB first) as shown in *Figure 180: Mode 3: decryption on page 533*.
8. Repeat steps 5, 6, 7 to process all the blocks using the same derivation key stored in the AES_KEYRx registers.

Figure 180. Mode 3: decryption

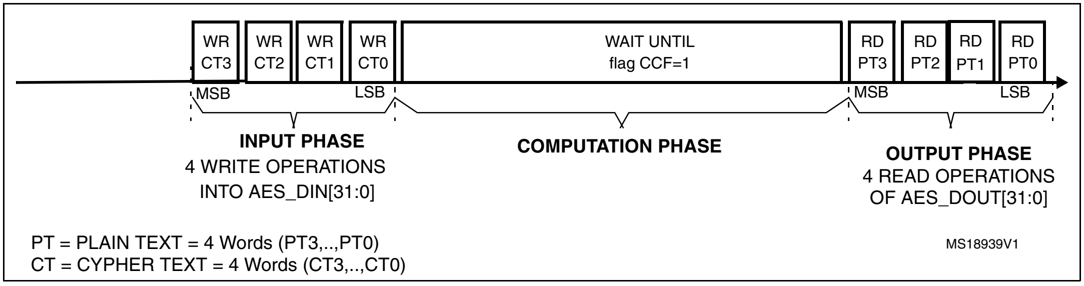

WR CT3
WR CT2
WR CT1
WR CT0
WAIT UNTIL flag CCF=1
RD PT3
RD PT2
RD PT1
RD PT0
MSB
LSB
MSB
LSB
INPUT PHASE
4 WRITE OPERATIONS INTO AES_DIN[31:0]
COMPUTATION PHASE
OUTPUT PHASE
4 READ OPERATIONS OF AES_DOUT[31:0]
PT = PLAIN TEXT = 4 Words(PT3,..,PT0)
CT = CYPHER TEXT = 4 Words(CT3,..,CT0)
MS18938V1

22.7.4 Mode 4: key derivation and decryption

1. Disable the AES by resetting the EN bit in the AES_CR register.
2. Configure Mode 4 by programming MODE[1:0]=11 in the AES_CR register. This mode is forbidden when AES is configured in CTR mode. It will be forced to CTR decryption mode if the software writes MODE[1:0] = 11 and CHMOD[1:0] = 10.
3. Write the AES_KEYRx register with the encryption key. Write the AES_IVRx register if the CBC mode is selected.
4. Enable the AES by setting the EN bit in the AES_CR register.
5. Write the AES_DINR register 4 times to input the cipher text (MSB first) as shown in *Figure 181: Mode 4: key derivation and decryption on page 533*.
6. Wait until the CCF flag is set in the AES_SR register.
7. Read the AES_DOUTR register 4 times to get the plain text (MSB first) as shown in *Figure 181: Mode 4: key derivation and decryption on page 533*.
8. Repeat steps 5, 6, 7 to process all the blocks with the same encryption key

Note: The AES_KEYRx registers contain the encryption key during all phases of the processing, No derivation key is stored in these registers. The derivation key starting from the encryption key is stored internally in the AES without storing a copy in the AES_KEYRx registers.

Figure 181. Mode 4: key derivation and decryption


WR CT3
WR CT2
WR CT1
WR CT0
WAIT UNTIL flag CCF=1
RD PT3
RD PT2
RD PT1
RD PT0
MSB
LSB
MSB
LSB
INPUT PHASE
4 WRITE OPERATIONS INTO AES_DIN[31:0]
COMPUTATION PHASE
OUTPUT PHASE
4 READ OPERATIONS OF AES_DOUT[31:0]
PT = PLAIN TEXT = 4 Words (PT3,..,PT0)
CT = CYPHER TEXT = 4 Words (CT3,..,CT0)
MS18939V1

22.8 AES DMA interface

The AES accelerator provides an interface to connect to the DMA controller.

The DMA must be configured to transfer words.

The AES can be associated with two distinct DMA request channels:

- A DMA request channel for the inputs: When the DMAINEN bit is set in the AES_CR register, the AES initiates a DMA request (AES_IN) during the INPUT phase each time it requires a word to be written to the AES_DINR register. The DMA channel must be configured in memory-to-peripheral mode with 32-bit data size.

- A DMA request channel for the outputs: When the DMAOUTEN bit is enabled, the AES initiates a DMA request (AES_OUT) during the OUTPUT phase each time it requires a word to be read from the AES_DOUTR register. The DMA channel must be configured in peripheral-to-memory mode with a data size equal to 32-bit.

Four DMA requests are asserted for each phase, these are described in *Figure 182* and *Figure 183*.

DMA requests are generated until the AES is disabled. So, after the data output phase at the end of processing a 128-bit data block, the AES switches automatically to a new data input phase for the next data block if any.

Note: For mode 2 (key derivation), access to the AES_KEYRx registers can be done by software using the CPU. No DMA channel is provided for this purpose. Consequently, the DMAINEN bit and DMAOUTEN bits in the AES_CR register have no effect during this mode.

The CCF flag is not relevant when DMAOUTEN = 1 and software does not need to read it in this case. This bit may stay high and has to be cleared by software if the application needs to disable the AES to cancel the DMA management and use CPU access for the data input or data output phase.

Figure 182. DMA requests and data transfers during Input phase (AES_IN)

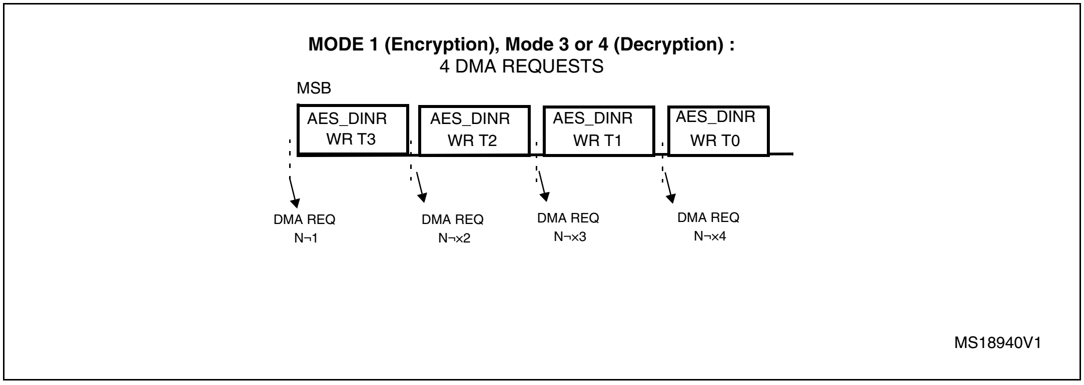
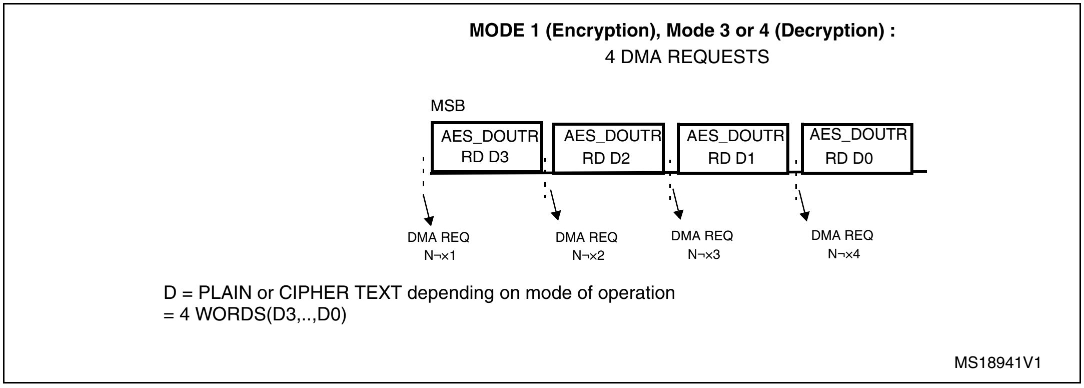


Figure 183. DMA requests during Output phase (AES_OUT)



22.9 Error flags

The RDERR flag in the AES_SR register is set when an unexpected read operation is detected during the computation phase or during the input phase.

The WRERR flag in the AES_SR register is set when an unexpected write operation is detected during the output phase or during the computation phase.

The flags may be cleared setting the respective bit in the AES_CR register (CCFC bit to clear the CCF flag, ERRC bit to clear the WERR and RDERR flags).

An interrupt can be generated when one of the error flags is set if the ERRIE bit in the AES_CR register has been previously set.

If an error is detected, AES is not disabled by hardware and continues processing as normal.

22.10 Processing time

The table summarizes the time required to process a 128-bit block for each mode of operation.

Table 82. Processing time (in clock cycle)

Mode of operation	Input phase	Computation phase	Output phase	Total
Mode 1: Encryption	8	202	4	214
Mode 2: Key derivation	-	80	-	80
Mode 3: Decryption	8	202	4	214
Mode 4: Key derivation + decryption	8	276	4	288

22.11 AES interrupts

Table 83. AES interrupt requests

Interrupt event	Event flag	Enable control bit	Exit from Wait
AES computation completed flag	CCF	CCFIE	yes
AES read error flag	RDERR	ERRIE	yes
AES write error flag	WRERR	ERRIE	yes

22.12 AES registers

22.12.1 AES control register (AES_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
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
Reserved			DMAO UTEN	DMAI NEN	ERRIE	CCFIE	ERRC	CCFC	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:13 Reserved, read as 0

Bit 12 **DMAOUTEN**: Enable DMA management of data output phase

0: DMA (during data output phase) disabled

1: DMA (during data output phase) enabled

If the DMAOUTEN bit is set, DMA requests are generated for the output data phase in mode 1, 3 or 4. This bit has no effect in mode 2 (Key derivation).

Bit 11 **DMAINEN**: Enable DMA management of data input phase

0: DMA (during data input phase) disabled

1: DMA (during data input phase) enabled

If the DMAINEN bit is set, DMA requests are generated for the data input phase in mode 1, 3 or 4. This bit has no action in mode 2 (Key Derivation).

Bit 10 **ERRIE**: Error interrupt enable

An interrupt is generated if at least one of the both flags RDERR or WRERR is set.

0: Error interrupt disabled

1: Error interrupt enabled

Bit 9 **CCFIE**: CCF flag interrupt enable

An interrupt is generated if the CCF flag is set.

0: CCF interrupt disabled

1: CCF interrupt enabled

Bit 8 **ERRC**: Error clear

Writing 1 to this bit clears the RDERR and WRERR flags.

This bit is always read low.

Bit 7 **CCFC**: Computation Complete Flag Clear

Writing 1 to this bit clears the CCF flag.

This bit is always read low.

Bits 6:5 **CHMOD[1:0]**: AES chaining mode

00: Electronic codebook (EBC)

01: Cipher-Block Chaining (CBC)

10: Counter Mode (CTR)

11: Reserved.

The AES chaining mode must only be changed while the AES is disabled. Writing these bits while the AES is enabled is forbidden to avoid unpredictable AES behavior.

Bits 4:3 **MODE[1:0]**: AES operating mode

00: Mode 1: Encryption

01: Mode 2: Key derivation

10: Mode 3: Decryption

11: Mode 4: Key derivation + decryption

The operation mode must only be changed if the AES is disabled. Writing these bits while the AES is enabled is forbidden to avoid unpredictable AES behavior.

Mode 4 is forbidden if CTR mode is selected. It will be forced to Mode 3 if the software, nevertheless, attempts to set mode 4 for this CTR mode configuration.

Bits 2:1 **DATATYPE[1:0]**: Data type selection (for data in and data out to/from the cryptographic block)

00: 32-bit data. No swapping.

01: 16-bit data or half-word. In the word, each half-word is swapped. For example, if one of the four 32-bit data written in the AES_DINR register is 0x764356AB, the value given to the cryptographic block is 0x56AB7643

10: 8-bit data or bytes. In the word, all the bytes are swapped. For example, if one of the four 32-bit data written in the AES_DINR register is 0x764356AB, the value given to the cryptographic block is 0xAB564376.

11: Bit data. In the word all the bits are swapped. For example, if one of the four 32-bit data written in the AES_DINR register is 0x764356AB, the value given to the cryptographic block is 0xD56AC26E

The Datatype selection must be changed if the AES is disabled. Writing these bits while the AES is enabled is forbidden to avoid unpredictable AES behavior.

Bits 0 **EN**: AES enable

0: AES disable

1: AES enable

The AES can be re-initialized at any moment by resetting this bit: the AES is then ready to start processing a new block when EN is set.

This bit is cleared by hardware when the AES computation is finished in mode 2 (Key derivation)

22.12.2 AES status register (AES_SR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
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
Reserved													WRERR	RDERR	CCF
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:3 Reserved, read as 0

Bit 2 **WRERR**: Write error flag

This bit is set by hardware when an unexpected write operation to the AES_DINR register is detected (during computation or data output phase). An interrupt is generated if the ERRIE bit has been previously set in the AES_CR register. This flag has no impact on the AES which continues running if even if WERR is set.

It is cleared by software by setting the ERRC bit in the AES_CR register.

0: No write error detected

1: Write error detected

Bit 1 **RDERR**: Read error flag

This bit is set by hardware when an unexpected read operation from the AES_DOUTR register is detected (during computation or data input phase). An interrupt is generated if the ERRIE bit has been previously set in the AES_CR register.This flag has no impact on the AES which continues running if even if RDERR is set.

It is cleared by software by setting the ERRC bit i in the AES_CR register.

0: No read error detected

1: Read error detected

Bit 0 **CCF**: Computation complete flag

This bit is set by hardware when the computation is complete. An interrupt is generated if the CCFIE bit has been previously set in the AES_CR register.

It is cleared by software by setting the CCFC bit in the AES_CR register.

0: Computation complete

1: Computation is not complete

Note: This bit is significant only when DMAOUTEN = 0. It may stay high when DMA_EN = 1.

22.12.3 AES data input register (AES_DINR)

Address offset: 0x08

Reset value: 0x0000 0000

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

Bits 31:0 **DINR[31:0]**: Data Input Register.

This register must be written 4 times during the input phase:

– In Mode 1 (Encryption), 4 words must be written which represent the plain text from MSB to LSB.

– In Mode 2 (Key Derivation), This register is not used because this mode concerns only derivative key calculation starting from the AES_KEYRx register.

– In Mode 3 (Decryption) and 4 (Key Derivation+Decryption), 4 words must be written which represent the cipher text MSB to LSB.

Note: This register must be accessed with 32-bit data width.

22.12.4 AES data output register (AES_DOUTR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bits 31:0 **DOUTR[31:0]**: Data output register

This register is read only.

Once the CCF flag (Computation Complete Flag) is set, reading this data register 4 times gives access to the 128-bit output results:

- In Mode 1 (Encryption), the 4 words read represent the cipher text from MSB to LSB.
- In Mode 2 (Key Derivation), there is no need to read this register because the derivative key is located in the AES_KEYRx registers.
- In Mode 3 (Decryption) and Mode 4 (Key Derivation+Decryption), the 4 words read represent the plain text from MSB to LSB.

Note: This register must be accessed with 32-bit data width.

22.12.5 AES key register 0(AES_KEYR0) (LSB: key [31:0])

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR0[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
KEYR0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **KEYR0[31:0]**: Data Output Register (LSB key [31:0])

This register must be written before the EN bit in the AES_CR register is set:

In Mode 1 (Encryption), mode 2 (Key Derivation) and mode 4 (Key Derivation + Decryption), the value to be written represents the encryption key from LSB, meaning Key [31:0].

In Mode 3 (Decryption), the value to be written represents the decryption key from LSB, meaning Key [31:0]. When the register is written with the encryption key in this decryption mode, reading it before the AES is enabled will return the encryption value. Reading it after CCF flag is set will return the derivation key.

Reading this register while AES is enabled return an unpredictable value.

Note: This register does not contain the derivation key in mode 4 (derivation key + decryption). It always contains the encryption key value.

22.12.6 AES key register 1 (AES_KEYR1) (Key[63:32])

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR1[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
KEYR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **KEYR1[31:0]**: AES key register (key [63:32])

Refer to the description of AES_KEYR0.

22.12.7 AES key register 2 (AES_KEYR2) (Key [95:64])

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR2[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
KEYR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **KEYR2[31:0]**: AES key register (key [95:64])
Refer to the description of AES_KEYR0.

22.12.8 AES key register 3 (AES_KEYR3) (MSB: key[127:96])

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR3[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
KEYR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **KEYR3[31:0]**: AES key register (MSB key [127:96])
Refer to the description of AES_KEYR0.

22.12.9 AES initialization vector register 0 (AES_IVR0) (LSB: IVR[31:0])

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVR0[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
IVR0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVR0[31:0]**: initialization vector register (LSB IVR [31:0])

This register must be written before the EN bit in the AES_CR register is set:

The register value has no meaning if:

- The EBC mode (Electronic codebook) is selected.
- The CTR or CBC mode is selected in addition with the Key derivation.

In CTR mode (Counter mode), this register contains the 32-bit counter value.

Reading this register while AES is enabled will return the value 0x00000000.

22.12.10 AES initialization vector register 1 (AES_IVR1) (IVR[63:32])

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVR1[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
IVR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVR1[31:0]**: Initialization Vector Register (IVR [63:32])

This register must be written before the EN bit in the AES_CR register is set:

The register value has no meaning if:

- The EBC mode (Electronic codebook) is selected.
- The CTR or CBC mode is selected in addition with the Key derivation or key derivation+decryption mode.

In CTR mode (Counter mode), this register contains the nonce value.

Reading this register while AES is enabled will return the value 0x00000000.

22.12.11 AES initialization vector register 2 (AES_IVR2) (IVR[95:64])

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVR2[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
IVR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVR2[31:0]**: Initialization Vector Register (IVR [95:64])

This register must be written before the EN bit in the AES_CR register is set:

The register value has no meaning if:

- The EBC mode (Electronic codebook) is selected.

- The CTR or CBC mode is selected in addition with the Key derivation or key derivation+decryption mode.

In CTR mode (Counter mode), this register contains the nonce value.

Reading this register while AES is enabled will return the value 0x00000000.

22.12.12 AES initialization vector register 3 (AES_IVR3) (MSB: IVR[127:96])

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVR3[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
IVR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVR3[31:0]**: Initialization Vector Register (MSB IVR [127:96])

This register must be written before the EN bit in the AES_CR register is set:

The register value has no meaning if:

- The EBC mode (Electronic codebook) is selected.

- The CTR or CBC mode is selected in addition with the Key derivation or key derivation+decryption mode.

In CTR mode (Counter mode), this register contains the nonce value.

Reading this register while AES is enabled will return the value 0x00000000.

22.12.13 AES register map

Table 84. AES register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	AES_CR	Reserved																			DMAOUTEN	DMAINEN	ERRIE	CCFIE	ERRC	CCFC	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0004	AES_SR	Reserved																													WRERR	RDERR	CCF
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0008	AES_DINR	AES_DINR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x000C	AES_DOUTR	AES_DOUTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0010	AES_KEYR0	AES_KEYR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0014	AES_KEYR1	AES_KEYR1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018	AES_KEYR2	AES_KEYR2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	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	AES_KEYR3	AES_KEYR3[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0020	AES_IVR0	AES_IVR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	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	AES_IVR1	AES_IVR1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0028	AES_IVR2	AES_IVR2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	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	AES_IVR3	AES_IVR3[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	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 *Table 2 on page 42* for the register boundary addresses.

23 Universal serial bus full-speed device interface (USB)

23.1 USB introduction

The USB peripheral implements an interface between a full-speed USB 2.0 bus and the APB1 bus.

USB suspend/resume are supported which allows to stop the device clocks for low-power consumption.

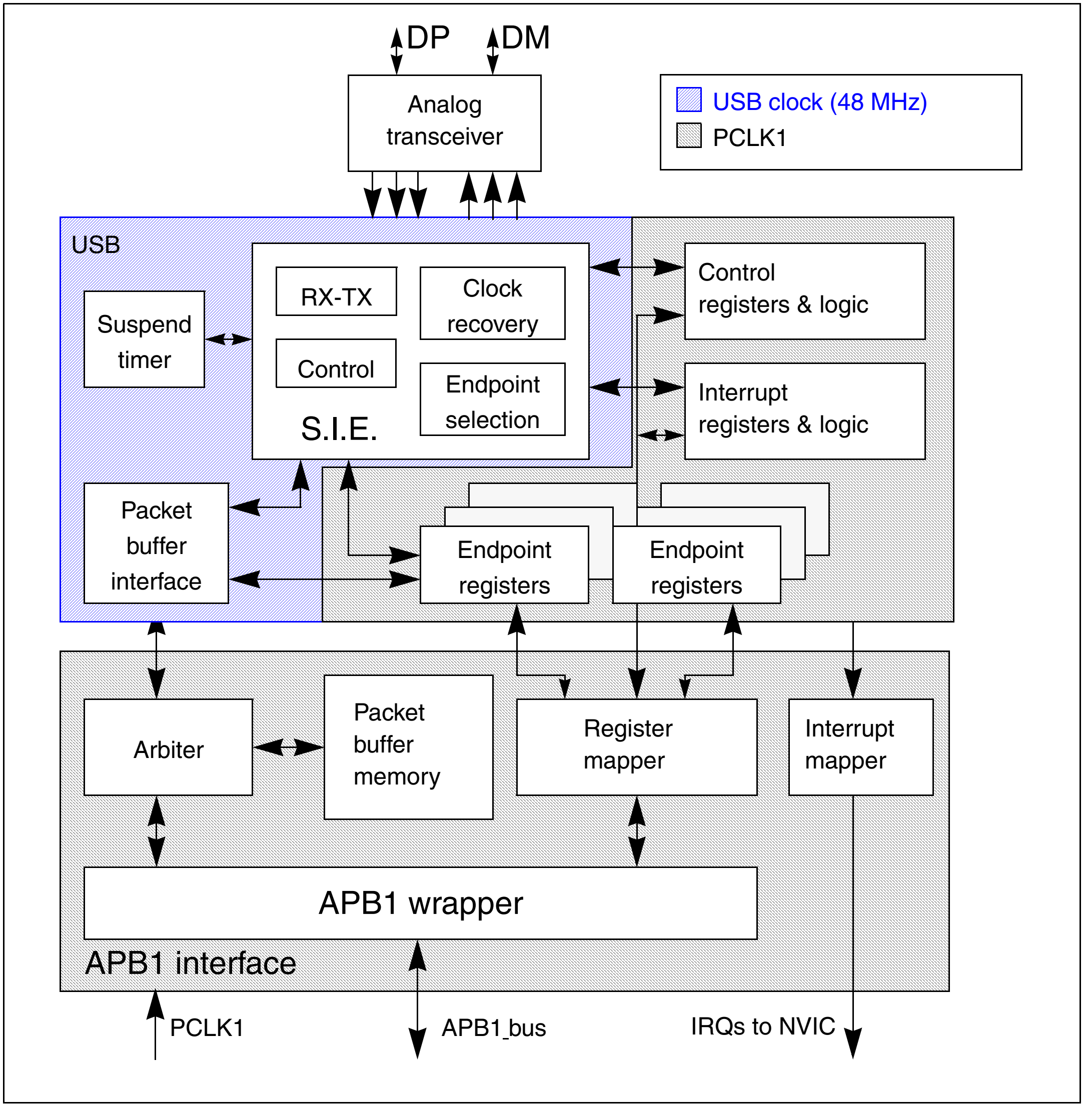
23.2 USB main features

- USB specification version 2.0 full-speed compliant
- Configurable number of endpoints from 1 to 8
- Cyclic redundancy check (CRC) generation/checking, Non-return-to-zero Inverted (NRZI) encoding/decoding and bit-stuffing
- Isochronous transfers support
- Double-buffered bulk/isochronous endpoint support
- USB Suspend/Resume operations
- Frame locked clock pulse generation
- USB internal connect/disconnect feature (controlled by system configuration register) with an internal pull-up resistor on the USB data+ (DP) line.

23.3 USB functional description

Figure 184 shows the block diagram of the USB peripheral.

Figure 184. USB peripheral block diagram



The USB peripheral provides an USB compliant connection between the host PC and the function implemented by the microcontroller. Data transfer between the host PC and the system memory occurs through a dedicated packet buffer memory accessed directly by the USB peripheral. The size of this dedicated buffer memory must be according to the number of endpoints used and the maximum packet size. This dedicated memory is sized to 512 bytes and up to 16 mono-directional or 8 bidirectional endpoints can be used.The USB peripheral interfaces with the USB host, detecting token packets, handling data transmission/reception, and processing handshake packets as required by the USB standard. Transaction formatting is performed by the hardware, including CRC generation and checking.

Each endpoint is associated with a buffer description block indicating where the endpoint related memory area is located, how large it is or how many bytes must be transmitted. When a token for a valid function/endpoint pair is recognized by the USB peripheral, the related data transfer (if required and if the endpoint is configured) takes place. The data buffered by the USB peripheral is loaded in an internal 16 bit register and memory access to the dedicated buffer is performed. When all the data has been transferred, if needed, the proper handshake packet over the USB is generated or expected according to the direction of the transfer.

At the end of the transaction, an endpoint-specific interrupt is generated, reading status registers and/or using different interrupt response routines. The microcontroller can determine:

- Which endpoint has to be served
- Which type of transaction took place, if errors occurred (bit stuffing, format, CRC, protocol, missing ACK, over/underrun, etc.)

Special support is offered to Isochronous transfers and high throughput bulk transfers, implementing a double buffer usage, which allows to always have an available buffer for the USB peripheral while the microcontroller uses the other one.

The unit can be placed in low-power mode (SUSPEND mode), by writing in the control register, whenever required. At this time, all static power dissipation is avoided, and the USB clock can be slowed down or stopped. The detection of activity at the USB inputs, while in low-power mode, wakes the device up asynchronously. A special interrupt source can be connected directly to a wakeup line to allow the system to immediately restart the normal clock generation and/or support direct clock start/stop.

23.3.1 Description of USB blocks

The USB peripheral implements all the features related to USB interfacing, which include the following blocks:

- Serial Interface Engine (SIE): The functions of this block include: synchronization pattern recognition, bit-stuffing, CRC generation and checking, PID verification/generation, and handshake evaluation. It must interface with the USB transceivers and uses the virtual buffers provided by the packet buffer interface for local data storage,. This unit also generates signals according to USB peripheral events, such as Start of Frame (SOF), USB_Reset, Data errors etc. and to Endpoint related events like end of transmission or correct reception of a packet; these signals are then used to generate interrupts.
- Timer: This block generates a start-of-frame locked clock pulse and detects a global suspend (from the host) when no traffic has been received for 3 ms.
- Packet Buffer Interface: This block manages the local memory implementing a set of buffers in a flexible way, both for transmission and reception. It can choose the proper buffer according to requests coming from the SIE and locate them in the memory addresses pointed by the Endpoint registers. It increments the address after each exchanged word until the end of packet, keeping track of the number of exchanged bytes and preventing the buffer to overrun the maximum capacity.
- Endpoint-Related Registers: Each endpoint has an associated register containing the endpoint type and its current status. For mono-directional/single-buffer endpoints, a single register can be used to implement two distinct endpoints. The number of registers is 8, allowing up to 16 mono-directional/single-buffer or up to 7 double-buffer

endpoints* in any combination. For example the USB peripheral can be programmed to have 4 double buffer endpoints and 8 single-buffer/mono-directional endpoints.

- Control Registers: These are the registers containing information about the status of the whole USB peripheral and used to force some USB events, such as resume and power-down.
- Interrupt Registers: These contain the Interrupt masks and a record of the events. They can be used to inquire an interrupt reason, the interrupt status or to clear the status of a pending interrupt.

*Note: * Endpoint 0 is always used for control transfer in single-buffer mode.*

The USB peripheral is connected to the APB1 bus through an APB1 interface, containing the following blocks:

- Packet Memory: This is the local memory that physically contains the Packet Buffers. It can be used by the Packet Buffer interface, which creates the data structure and can be accessed directly by the application software. The size of the Packet Memory is 512 bytes, structured as 256 words by 16 bits.
- Arbiter: This block accepts memory requests coming from the APB1 bus and from the USB interface. It resolves the conflicts by giving priority to APB1 accesses, while always reserving half of the memory bandwidth to complete all USB transfers. This time-duplex scheme implements a virtual dual-port SRAM that allows memory access, while an USB transaction is happening. Multiword APB1 transfers of any length are also allowed by this scheme.
- Register Mapper: This block collects the various byte-wide and bit-wide registers of the USB peripheral in a structured 16-bit wide word set addressed by the APB1.
- APB1 Wrapper: This provides an interface to the APB1 for the memory and register. It also maps the whole USB peripheral in the APB1 address space.
- Interrupt Mapper: This block is used to select how the possible USB events can generate interrupts and map them to three different lines of the NVIC:
  - USB low-priority interrupt (Channel 20): Triggered by all USB events (Correct transfer, USB reset, etc.). The firmware has to check the interrupt source before serving the interrupt.
  - USB high-priority interrupt (Channel 19): Triggered only by a correct transfer event for isochronous and double-buffer bulk transfer to reach the highest possible transfer rate.
  - USB wakeup interrupt (Channel 42): Triggered by the wakeup event from the USB Suspend mode.

23.4 Programming considerations

In the following sections, the expected interactions between the USB peripheral and the application program are described, in order to ease application software development.

23.4.1 Generic USB device programming

This part describes the main tasks required of the application software in order to obtain USB compliant behavior. The actions related to the most general USB events are taken into account and paragraphs are dedicated to the special cases of double-buffered endpoints and Isochronous transfers. Apart from system reset, action is always initiated by the USB peripheral, driven by one of the USB events described below.

23.4.2 System and power-on reset

Upon system and power-on reset, the first operation the application software should perform is to provide all required clock signals to the USB peripheral and subsequently de-assert its reset signal so to be able to access its registers. The whole initialization sequence is hereafter described.

An internal pull-up resistor is connected to Data+ (DP) line and controlled by software using the USB_PU bit in the SYSCFG_PMC register of the SYSCFG module (refer to *Section 7: System configuration controller (SYSCFG) and routing interface (RI)*. When the USB_PU bit is reset, no pull-up is connected to the DP line and the device cannot be detected on the USB bus (if no external pull-up is connected). When the USB_PU bit is set, the internal pull-up is connected and the device can be detected on the USB bus.

As a first step application software needs to activate register macrocell clock and de-assert macrocell specific reset signal using related control bits provided by device clock management logic.

After that, the analog part of the device related to the USB transceiver must be switched on using the PDWN bit in CNTR register, which requires a special handling. This bit is intended to switch on the internal voltage references that supply the port transceiver. This circuit has a defined startup time ($t_{STARTUP}$ specified in the datasheet) during which the behavior of the USB transceiver is not defined. It is thus necessary to wait this time, after setting the PDWN bit in the CNTR register, before removing the reset condition on the USB part (by clearing the FRES bit in the CNTR register). Clearing the ISTR register then removes any spurious pending interrupt before any other macrocell operation is enabled.

At system reset, the microcontroller must initialize all required registers and the packet buffer description table, to make the USB peripheral able to properly generate interrupts and data transfers. All registers not specific to any endpoint must be initialized according to the needs of application software (choice of enabled interrupts, chosen address of packet buffers, etc.). Then the process continues as for the USB reset case (see further paragraph).

USB reset (RESET interrupt)

When this event occurs, the USB peripheral is put in the same conditions it is left by the system reset after the initialization described in the previous paragraph: communication is disabled in all endpoint registers (the USB peripheral will not respond to any packet). As a response to the USB reset event, the USB function must be enabled, having as USB address 0, implementing only the default control endpoint (endpoint address is 0 too). This is accomplished by setting the Enable Function (EF) bit of the USB_DADDR register and initializing the EP0R register and its related packet buffers accordingly. During USB enumeration process, the host assigns a unique address to this device, which must be written in the ADD[6:0] bits of the USB_DADDR register, and configures any other necessary endpoint.

When a RESET interrupt is received, the application software is responsible to enable again the default endpoint of USB function 0 within 10mS from the end of reset sequence which triggered the interrupt.

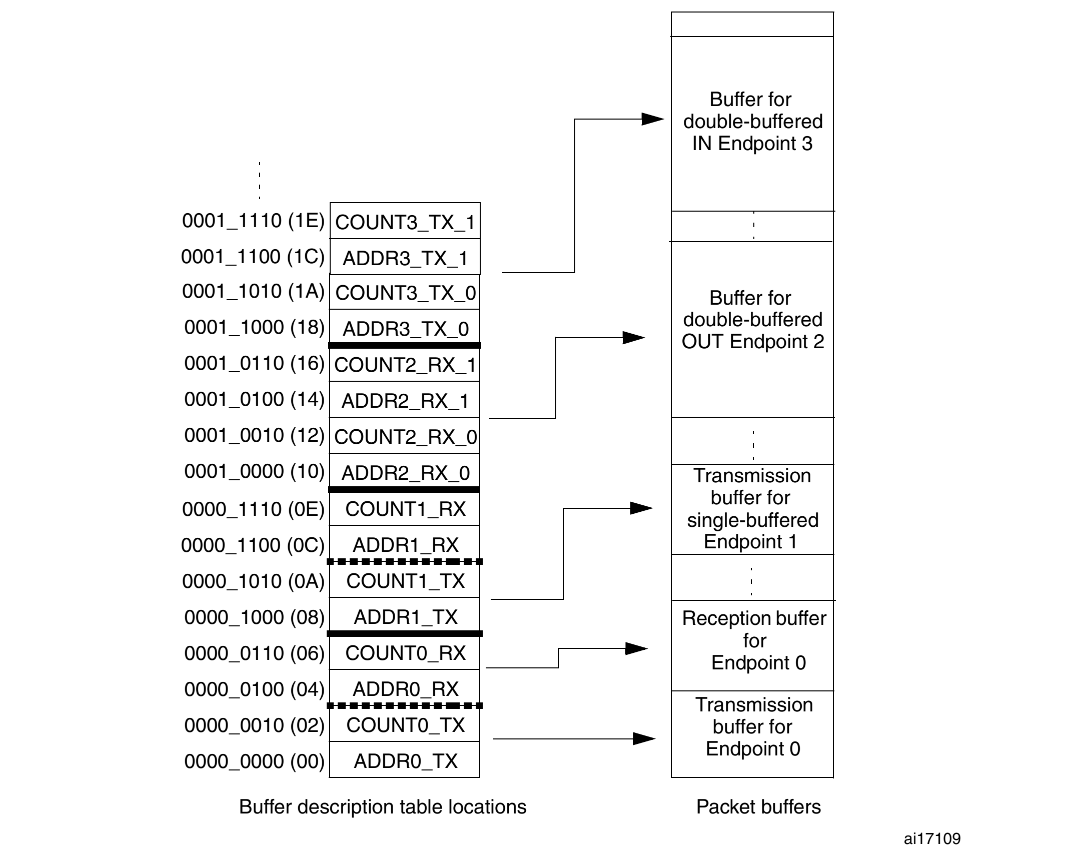
Structure and usage of packet buffers

Each bidirectional endpoint may receive or transmit data from/to the host. The received data is stored in a dedicated memory buffer reserved for that endpoint, while another memory buffer contains the data to be transmitted by the endpoint. Access to this memory is performed by the packet buffer interface block, which delivers a memory access request and waits for its acknowledgement. Since the packet buffer memory has to be accessed by the microcontroller also, an arbitration logic takes care of the access conflicts, using half APB1 cycle for microcontroller access and the remaining half for the USB peripheral access. In this way, both the agents can operate as if the packet memory is a dual-port SRAM, without being aware of any conflict even when the microcontroller is performing back-to-back accesses. The USB peripheral logic uses a dedicated clock. The frequency of this dedicated clock is fixed by the requirements of the USB standard at 48 MHz, and this can be different from the clock used for the interface to the APB1 bus. Different clock configurations are possible where the APB1 clock frequency can be higher or lower than the USB peripheral one.

Note: Due to USB data rate and packet memory interface requirements, the APB1 clock must have a minimum frequency of 10 MHz to avoid data overrun/underrun problems.

Each endpoint is associated with two packet buffers (usually one for transmission and the other one for reception). Buffers can be placed anywhere inside the packet memory because their location and size is specified in a buffer description table, which is also located in the packet memory at the address indicated by the USB_BTABLE register. Each table entry is associated to an endpoint register and it is composed of four 16-bit words so that table start address must always be aligned to an 8-byte boundary (the lowest three bits of USB_BTABLE register are always “000”). Buffer descriptor table entries are described in the *Section 23.5.3: Buffer descriptor table*. If an endpoint is unidirectional and it is neither an Isochronous nor a double-buffered bulk, only one packet buffer is required (the one related to the supported transfer direction). Other table locations related to unsupported transfer directions or unused endpoints, are available to the user. Isochronous and double-buffered bulk endpoints have special handling of packet buffers (Refer to *Section 23.4.4: Isochronous transfers* and *Section 23.4.3: Double-buffered endpoints* respectively). The relationship between buffer description table entries and packet buffer areas is depicted in *Figure 185*.

Figure 185. Packet buffer areas with examples of buffer description table locations



Each packet buffer is used either during reception or transmission starting from the bottom. The USB peripheral will never change the contents of memory locations adjacent to the allocated memory buffers; if a packet bigger than the allocated buffer length is received (buffer overrun condition) the data will be copied to the memory only up to the last available location.

Endpoint initialization

The first step to initialize an endpoint is to write appropriate values to the ADDRn_TX/ADDRn_RX registers so that the USB peripheral finds the data to be transmitted already available and the data to be received can be buffered. The EP_TYPE bits in the USB_EPnR register must be set according to the endpoint type, eventually using the EP_KIND bit to enable any special required feature. On the transmit side, the endpoint must be enabled using the STAT_TX bits in the USB_EPnR register and COUNTn_TX must be initialized. For reception, STAT_RX bits must be set to enable reception and COUNTn_RX must be written with the allocated buffer size using the BL_SIZE and NUM_BLOCK fields. Unidirectional endpoints, except Isochronous and double-buffered bulk endpoints, need to initialize only bits and registers related to the supported direction. Once the transmission and/or reception are enabled, register USB_EPnR and locations ADDRn_TX/ADDRn_RX, COUNTn_TX/COUNTn_RX (respectively), should not be modified by the application software, as the hardware can change their value on the fly. When the data transfer operation is completed, notified by a CTR interrupt event, they can be accessed again to re-enable a new operation.

IN packets (data transmission)

When receiving an IN token packet, if the received address matches a configured and valid endpoint one, the USB peripheral accesses the contents of ADDRn_TX and COUNTn_TX locations inside buffer descriptor table entry related to the addressed endpoint. The content of these locations is stored in its internal 16 bit registers ADDR and COUNT (not accessible by software). The packet memory is accessed again to read the first word to be transmitted (Refer to *Structure and usage of packet buffers on page 549*) and starts sending a DATA0 or DATA1 PID according to USB_EPnR bit DTOG_TX. When the PID is completed, the first byte from the word, read from buffer memory, is loaded into the output shift register to be transmitted on the USB bus. After the last data byte is transmitted, the computed CRC is sent. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the data packet, according to STAT_TX bits in the USB_EPnR register.

The ADDR internal register is used as a pointer to the current buffer memory location while COUNT is used to count the number of remaining bytes to be transmitted. Each word read from the packet buffer memory is transmitted over the USB bus starting from the least significant byte. Transmission buffer memory is read starting from the address pointed by ADDRn_TX for COUNTn_TX/2 words. If a transmitted packet is composed of an odd number of bytes, only the lower half of the last word accessed will be used.

On receiving the ACK receipt by the host, the USB_EPnR register is updated in the following way: DTOG_TX bit is toggled, the endpoint is made invalid by setting STAT_TX=10 (NAK) and bit CTR_TX is set. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the EP_ID and DIR bits in the USB_ISTR register. Servicing of the CTR_TX event starts clearing the interrupt bit; the application software then prepares another buffer full of data to be sent, updates the COUNTn_TX table location with the number of byte to be transmitted during the next transfer, and finally sets STAT_TX to ‘11 (VALID) to re-enable transmissions. While the STAT_TX bits are equal to ‘10 (NAK), any IN request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host will retry the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second IN transaction addressed to the same endpoint immediately following the one which triggered the CTR interrupt.

OUT and SETUP packets (data reception)

These two tokens are handled by the USB peripheral more or less in the same way; the differences in the handling of SETUP packets are detailed in the following paragraph about control transfers. When receiving an OUT/SETUP PID, if the address matches a valid endpoint, the USB peripheral accesses the contents of the ADDRn_RX and COUNTn_RX locations inside the buffer descriptor table entry related to the addressed endpoint. The content of the ADDRn_RX is stored directly in its internal register ADDR. While COUNT is now reset and the values of BL_SIZE and NUM_BLOCK bit fields, which are read within COUNTn_RX content are used to initialize BUF_COUNT, an internal 16 bit counter, which is used to check the buffer overrun condition (all these internal registers are not accessible by software). Data bytes subsequently received by the USB peripheral are packed in words (the first byte received is stored as least significant byte) and then transferred to the packet buffer starting from the address contained in the internal ADDR register while BUF_COUNT is decremented and COUNT is incremented at each byte transfer. When the end of DATA packet is detected, the correctness of the received CRC is tested and only if no errors occurred during the reception, an ACK handshake packet is sent back to the transmitting host.

In case of wrong CRC or other kinds of errors (bit-stuff violations, frame errors, etc.), data bytes are still copied in the packet memory buffer, at least until the error detection point, but ACK packet is not sent and the ERR bit in USB_ISTR register is set. However, there is usually no software action required in this case: the USB peripheral recovers from reception errors and remains ready for the next transaction to come. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the ACK, according to bits STAT_RX in the USB_EPnR register and no data is written in the reception memory buffers.

Reception memory buffer locations are written starting from the address contained in the ADDRn_RX for a number of bytes corresponding to the received data packet length, CRC included (i.e. data payload length + 2), or up to the last allocated memory location, as defined by BL_SIZE and NUM_BLOCK, whichever comes first. In this way, the USB peripheral never writes beyond the end of the allocated reception memory buffer area. If the length of the data packet payload (actual number of bytes used by the application) is greater than the allocated buffer, the USB peripheral detects a buffer overrun condition. in this case, a STALL handshake is sent instead of the usual ACK to notify the problem to the host, no interrupt is generated and the transaction is considered failed.

When the transaction is completed correctly, by sending the ACK handshake packet, the internal COUNT register is copied back in the COUNTn_RX location inside the buffer description table entry, leaving unaffected BL_SIZE and NUM_BLOCK fields, which normally do not require to be re-written, and the USB_EPnR register is updated in the following way: DTOG_RX bit is toggled, the endpoint is made invalid by setting STAT_RX = ‘10 (NAK) and bit CTR_RX is set. If the transaction has failed due to errors or buffer overrun condition, none of the previously listed actions take place. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the EP_ID and DIR bits in the USB_ISTR register. The CTR_RX event is serviced by first determining the transaction type (SETUP bit in the USB_EPnR register); the application software must clear the interrupt flag bit and get the number of received bytes reading the COUNTn_RX location inside the buffer description table entry related to the endpoint being processed. After the received data is processed, the application software should set the STAT_RX bits to ‘11 (Valid) in the USB_EPnR, enabling further transactions. While the STAT_RX bits are equal to ‘10 (NAK), any OUT request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host will retry the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second OUT transaction addressed to the same endpoint following immediately the one which triggered the CTR interrupt.

Control transfers

Control transfers are made of a SETUP transaction, followed by zero or more data stages, all of the same direction, followed by a status stage (a zero-byte transfer in the opposite direction). SETUP transactions are handled by control endpoints only and are very similar to OUT ones (data reception) except that the values of DTOG_TX and DTOG_RX bits of the addressed endpoint registers are set to 1 and 0 respectively, to initialize the control transfer, and both STAT_TX and STAT_RX are set to ‘10 (NAK) to let software decide if subsequent transactions must be IN or OUT depending on the SETUP contents. A control endpoint must check SETUP bit in the USB_EPnR register at each CTR_RX event to distinguish normal OUT transactions from SETUP ones. A USB device can determine the number and direction of data stages by interpreting the data transferred in the SETUP stage, and is required to STALL the transaction in the case of errors. To do so, at all data stages before the last, the unused direction should be set to STALL, so that, if the host reverses the transfer direction too soon, it gets a STALL as a status stage.

While enabling the last data stage, the opposite direction should be set to NAK, so that, if the host reverses the transfer direction (to perform the status stage) immediately, it is kept waiting for the completion of the control operation. If the control operation completes successfully, the software will change NAK to VALID, otherwise to STALL. At the same time, if the status stage will be an OUT, the STATUS_OUT (EP_KIND in the USB_EPnR register) bit should be set, so that an error is generated if a status transaction is performed with not-zero data. When the status transaction is serviced, the application clears the STATUS_OUT bit and sets STAT_RX to VALID (to accept a new command) and STAT_TX to NAK (to delay a possible status stage immediately following the next setup).

Since the USB specification states that a SETUP packet cannot be answered with a handshake different from ACK, eventually aborting a previously issued command to start the new one, the USB logic doesn’t allow a control endpoint to answer with a NAK or STALL packet to a SETUP token received from the host.

When the STAT_RX bits are set to ‘01 (STALL) or ‘10 (NAK) and a SETUP token is received, the USB accepts the data, performing the required data transfers and sends back an ACK handshake. If that endpoint has a previously issued CTR_RX request not yet acknowledged by the application (i.e. CTR_RX bit is still set from a previously completed reception), the USB discards the SETUP transaction and does not answer with any handshake packet regardless of its state, simulating a reception error and forcing the host to send the SETUP token again. This is done to avoid losing the notification of a SETUP transaction addressed to the same endpoint immediately following the transaction, which triggered the CTR_RX interrupt.

23.4.3 Double-buffered endpoints

All different endpoint types defined by the USB standard represent different traffic models, and describe the typical requirements of different kind of data transfer operations. When large portions of data are to be transferred between the host PC and the USB function, the bulk endpoint type is the most suited model. This is because the host schedules bulk transactions so as to fill all the available bandwidth in the frame, maximizing the actual transfer rate as long as the USB function is ready to handle a bulk transaction addressed to it. If the USB function is still busy with the previous transaction when the next one arrives, it will answer with a NAK handshake and the host PC will issue the same transaction again until the USB function is ready to handle it, reducing the actual transfer rate due to the bandwidth occupied by re-transmissions. For this reason, a dedicated feature called ‘double-buffering’ can be used with bulk endpoints.

When ‘double-buffering’ is activated, data toggle sequencing is used to select, which buffer is to be used by the USB peripheral to perform the required data transfers, using both ‘transmission’ and ‘reception’ packet memory areas to manage buffer swapping on each successful transaction in order to always have a complete buffer to be used by the application, while the USB peripheral fills the other one. For example, during an OUT transaction directed to a ‘reception’ double-buffered bulk endpoint, while one buffer is being filled with new data coming from the USB host, the other one is available for the microcontroller software usage (the same would happen with a ‘transmission’ double-buffered bulk endpoint and an IN transaction).

Since the swapped buffer management requires the usage of all 4 buffer description table locations hosting the address pointer and the length of the allocated memory buffers, the USB_EPnR registers used to implement double-buffered bulk endpoints are forced to be used as unidirectional ones. Therefore, only one STAT bit pair must be set at a value different from ‘00 (Disabled): STAT_RX if the double-buffered bulk endpoint is enabled for reception, STAT_TX if the double-buffered bulk endpoint is enabled for transmission. In case it is required to have double-buffered bulk endpoints enabled both for reception and transmission, two USB_EPnR registers must be used.

To exploit the double-buffering feature and reach the highest possible transfer rate, the endpoint flow control structure, described in previous chapters, has to be modified, in order to switch the endpoint status to NAK only when a buffer conflict occurs between the USB peripheral and application software, instead of doing it at the end of each successful transaction. The memory buffer which is currently being used by the USB peripheral is defined by the DTOG bit related to the endpoint direction: DTOG_RX (bit 14 of USB_EPnR register) for ‘reception’ double-buffered bulk endpoints or DTOG_TX (bit 6 of USB_EPnR register) for ‘transmission’ double-buffered bulk endpoints. To implement the new flow control scheme, the USB peripheral should know which packet buffer is currently in use by the application software, so to be aware of any conflict. Since in the USB_EPnR register, there are two DTOG bits but only one is used by USB peripheral for data and buffer sequencing (due to the unidirectional constraint required by double-buffering feature) the other one can be used by the application software to show which buffer it is currently using. This new buffer flag is called SW_BUF. In the following table the correspondence between USB_EPnR register bits and DTOG/SW_BUF definition is explained, for the cases of ‘transmission’ and ‘reception’ double-buffered bulk endpoints.

Table 85. Double-buffering buffer flag definition

Buffer flag	‘Transmission’ endpoint	‘Reception’ endpoint
DTOG	DTOG_TX (USB_EPnRbit 6)	DTOG_RX (USB_EPnRbit 14)
SW_BUF	USB_EPnR bit 14	USB_EPnR bit 6

The memory buffer which is currently being used by the USB peripheral is defined by DTOG buffer flag, while the buffer currently in use by application software is identified by SW_BUF buffer flag. The relationship between the buffer flag value and the used packet buffer is the same in both cases, and it is listed in the following table.

Table 86. Bulk double-buffering memory buffers usage

Endpoint Type	DTOG	SW_BUF	Packet buffer used by USB Peripheral	Packet buffer used by Application Software
IN	0	1	ADDRn_TX_0 / COUNTn_TX_0 Buffer description table locations.	ADDRn_TX_1 / COUNTn_TX_1 Buffer description table locations.
	1	0	ADDRn_TX_1 / COUNTn_TX_1 Buffer description table locations	ADDRn_TX_0 / COUNTn_TX_0 Buffer description table locations.
	0	0	None [1]	ADDRn_TX_0 / COUNTn_TX_0 Buffer description table locations.
	1	1	None [1]	ADDRn_TX_0 / COUNTn_TX_0 Buffer description table locations.
OUT	0	1	ADDRn_RX_0 / COUNTn_RX_0 Buffer description table locations.	ADDRn_RX_1 / COUNTn_RX_1 Buffer description table locations.
	1	0	ADDRn_RX_1 / COUNTn_RX_1 Buffer description table locations.	ADDRn_RX_0 / COUNTn_RX_0 Buffer description table locations.
	0	0	None [1]	ADDRn_RX_0 / COUNTn_RX_0 Buffer description table locations.
	1	1	None [1]	ADDRn_RX_1 / COUNTn_RX_1 Buffer description table locations.

1. Endpoint in NAK Status.

Double-buffering feature for a bulk endpoint is activated by:

- Writing EP_TYPE bit field at ‘00 in its USB_EPnR register, to define the endpoint as a bulk, and
- Setting EP_KIND bit at ‘1 (DBL_BUF), in the same register.

The application software is responsible for DTOG and SW_BUF bits initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. The end of the first transaction occurring after having set DBL_BUF, triggers the special flow control of double-buffered bulk endpoints, which is used for all other transactions addressed to this endpoint until DBL_BUF remain set. At the end of each transaction the CTR_RX or CTR_TX bit of the addressed endpoint USB_EPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB_EPnR register is hardware toggled making the USB peripheral buffer swapping completely software independent. Unlike common transactions, and the first one after DBL_BUF setting, STAT bit pair is not affected by the transaction termination and its value remains ‘11 (Valid). However, as the token packet of a new transaction is received, the actual endpoint status will be masked as ‘10 (NAK) when a buffer conflict between the USB peripheral and the application software is detected (this condition is identified by DTOG and SW_BUF having the same value, see *Table 86 on page 556*). The application software responds to the CTR event notification by clearing the interrupt flag and starting any required handling of the completed transaction. When the application packet buffer usage is over, the software toggles the SW_BUF bit, writing ‘1 to it, to notify the USB peripheral about the availability of that buffer. In this way, the number of NAKed transactions is limited only by the application elaboration time of a transaction data: if the elaboration time is shorter than the time required to complete a transaction on the USB bus, no re-transmissions due to flow control will take place and the actual transfer rate will be limited only by the host PC.

The application software can always override the special flow control implemented for double-buffered bulk endpoints, writing an explicit status different from ‘11 (Valid) into the STAT bit pair of the related USB_EPnR register. In this case, the USB peripheral will always use the programmed endpoint status, regardless of the buffer usage condition.

23.4.4 Isochronous transfers

The USB standard supports full speed peripherals requiring a fixed and accurate data production/consume frequency, defining this kind of traffic as ‘Isochronous’. Typical examples of this data are: audio samples, compressed video streams, and in general any sort of sampled data having strict requirements for the accuracy of delivered frequency. When an endpoint is defined to be ‘isochronous’ during the enumeration phase, the host allocates in the frame the required bandwidth and delivers exactly one IN or OUT packet each frame, depending on endpoint direction. To limit the bandwidth requirements, no re-transmission of failed transactions is possible for Isochronous traffic; this leads to the fact that an isochronous transaction does not have a handshake phase and no ACK packet is expected or sent after the data packet. For the same reason, Isochronous transfers do not support data toggle sequencing and always use DATA0 PID to start any data packet.

The Isochronous behavior for an endpoint is selected by setting the EP_TYPE bits at ‘10 in its USB_EPnR register; since there is no handshake phase the only legal values for the STAT_RX/STAT_TX bit pairs are ‘00 (Disabled) and ‘11 (Valid), any other value will produce results not compliant to USB standard. Isochronous endpoints implement double-buffering to ease application software development, using both ‘transmission’ and ‘reception’ packet memory areas to manage buffer swapping on each successful transaction in order to have always a complete buffer to be used by the application, while the USB peripheral fills the other.

The memory buffer which is currently used by the USB peripheral is defined by the DTOG bit related to the endpoint direction (DTOG_RX for ‘reception’ isochronous endpoints, DTOG_TX for ‘transmission’ isochronous endpoints, both in the related USB_EPnR register) according to *Table 87*.

Table 87. Isochronous memory buffers usage

Endpoint Type	DTOG bit value	Packet buffer used by the USB peripheral	Packet buffer used by the application software
IN	0	ADDRn_TX_0 / COUNTn_TX_0 buffer description table locations.	ADDRn_TX_1 / COUNTn_TX_1 buffer description table locations.
	1	ADDRn_TX_1 / COUNTn_TX_1 buffer description table locations.	ADDRn_TX_0 / COUNTn_TX_0 buffer description table locations.
OUT	0	ADDRn_RX_0 / COUNTn_RX_0 buffer description table locations.	ADDRn_RX_1 / COUNTn_RX_1 buffer description table locations.
	1	ADDRn_RX_1 / COUNTn_RX_1 buffer description table locations.	ADDRn_RX_0 / COUNTn_RX_0 buffer description table locations.

As it happens with double-buffered bulk endpoints, the USB_EPnR registers used to implement Isochronous endpoints are forced to be used as unidirectional ones. In case it is required to have Isochronous endpoints enabled both for reception and transmission, two USB_EPnR registers must be used.

The application software is responsible for the DTOG bit initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. At the end of each transaction, the CTR_RX or CTR_TX bit of the addressed endpoint USB_EPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB_EPnR register is hardware toggled making buffer swapping completely software independent. STAT bit pair is not affected by transaction completion; since no flow control is possible for Isochronous transfers due to the lack of handshake phase, the endpoint remains always ‘11 (Valid). CRC errors or buffer-overrun conditions occurring during Isochronous OUT transfers are anyway considered as correct transactions and they always trigger an CTR_RX event. However, CRC errors will anyway set the ERR bit in the USB_ISTR register to notify the software of the possible data corruption.

23.4.5 Suspend/Resume events

The USB standard defines a special peripheral state, called SUSPEND, in which the average current drawn from the USB bus must not be greater than 2.5 mA. This requirement is of fundamental importance for bus-powered devices, while self-powered devices are not required to comply to this strict power consumption constraint. In suspend mode, the host PC sends the notification to not send any traffic on the USB bus for more than 3mS: since a SOF packet must be sent every mS during normal operations, the USB peripheral detects the lack of 3 consecutive SOF packets as a suspend request from the host PC and set the SUSP bit to ‘1 in USB_ISTR register, causing an interrupt if enabled. Once the device is suspended, its normal operation can be restored by a so called RESUME sequence, which can be started from the host PC or directly from the peripheral itself, but it is always terminated by the host PC. The suspended USB peripheral must be anyway able to detect a RESET sequence, reacting to this event as a normal USB reset event.

The actual procedure used to suspend the USB peripheral is device dependent since according to the device composition, different actions may be required to reduce the total consumption.

A brief description of a typical suspend procedure is provided below, focused on the USB-related aspects of the application software routine responding to the SUSP notification of the USB peripheral:

1. Set the FSUSP bit in the USB_CNTR register to 1. This action activates the suspend mode within the USB peripheral. As soon as the suspend mode is activated, the check on SOF reception is disabled to avoid any further SUSP interrupts being issued while the USB is suspended.
2. Remove or reduce any static power consumption in blocks different from the USB peripheral.
3. Set LP_MODE bit in USB_CNTR register to 1 to remove static power consumption in the analog USB transceivers but keeping them able to detect resume activity.
4. Optionally turn off external oscillator and device PLL to stop any activity inside the device.

When an USB event occurs while the device is in SUSPEND mode, the RESUME procedure must be invoked to restore nominal clocks and regain normal USB behavior. Particular care must be taken to insure that this process does not take more than 10mS when the wakening event is an USB reset sequence (See “Universal Serial Bus Specification” for more details). The start of a resume or reset sequence, while the USB peripheral is suspended, clears the LP_MODE bit in USB_CNTR register asynchronously. Even if this event can trigger an WKUP interrupt if enabled, the use of an interrupt response routine must be carefully evaluated because of the long latency due to system clock restart; to have the shorter latency before re-activating the nominal clock it is suggested to put the resume procedure just after the end of the suspend one, so its code is immediately executed as soon as the system clock restarts. To prevent ESD discharges or any other kind of noise from waking-up the system (the exit from suspend mode is an asynchronous event), a suitable analog filter on data line status is activated during suspend; the filter width is about 70ns.

The following is a list of actions a resume procedure should address:

1. Optionally turn on external oscillator and/or device PLL.
2. Clear FSUSP bit of USB_CNTR register.
3. If the resume triggering event has to be identified, bits RXDP and RXDM in the USB_FNR register can be used according to *Table 88*, which also lists the intended software action in all the cases. If required, the end of resume or reset sequence can be detected monitoring the status of the above mentioned bits by checking when they reach the “10” configuration, which represent the Idle bus state; moreover at the end of a reset sequence the RESET bit in USB_ISTR register is set to 1, issuing an interrupt if enabled, which should be handled as usual.

Table 88. Resume event detection

[RXDP,RXDM] status	Wakeup event	Required resume software action
“00”	Root reset	None
“10”	None (noise on bus)	Go back in Suspend mode
“01”	Root resume	None
“11”	Not allowed (noise on bus)	Go back in Suspend mode

A device may require to exit from suspend mode as an answer to particular events not directly related to the USB protocol (e.g. a mouse movement wakes up the whole system). In this case, the resume sequence can be started by setting the RESUME bit in the USB_CNTR register to ‘1 and resetting it to 0 after an interval between 1 mS and 15 mS (this interval can be timed using ESOF interrupts, occurring with a 1mS period when the system clock is running at nominal frequency). Once the RESUME bit is clear, the resume sequence will be completed by the host PC and its end can be monitored again using the RXDP and RXDM bits in the USB_FNR register.

Note: The RESUME bit must be anyway used only after the USB peripheral has been put in suspend mode, setting the FSUSP bit in USB_CNTR register to 1.

23.5 USB registers

The USB peripheral registers can be divided into the following groups:

- Common Registers: Interrupt and Control registers
- Endpoint Registers: Endpoint configuration and status
- Buffer Descriptor Table: Location of packet memory used to locate data buffers

All register addresses are expressed as offsets with respect to the USB peripheral registers base address 0x4000 5C00, except the buffer descriptor table locations, which starts at the address specified by the USB_BTABLE register. Due to the common limitation of APB1 bridges on word addressability, all register addresses are aligned to 32-bit word boundaries although they are 16-bit wide. The same address alignment is used to access packet buffer memory locations, which are located starting from 0x4000 6000.

Refer to *Section 1.1 on page 37* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

23.5.1 Common registers

These registers affect the general behavior of the USB peripheral defining operating mode, interrupt handling, device address and giving access to the current frame number updated by the host PC.

USB control register (USB_CNTR)

Address offset: 0x40

Reset value: 0x0003

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RESETM	SOFM	ESOFM	Reserved			RESUME	FSUSP	LP_MODE	PDWN	FRES
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bit 15 **CTRM:** Correct transfer interrupt mask

0: Correct Transfer (CTR) Interrupt disabled.
1: CTR Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 14 **PMAOVRM:** Packet memory area over / underrun interrupt mask

0: PMAOVR Interrupt disabled.
1: PMAOVR Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 13 **ERRM:** Error interrupt mask

0: ERR Interrupt disabled.
1: ERR Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 12 **WKUPM:** Wakeup interrupt mask

0: WKUP Interrupt disabled.
1: WKUP Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 11 **SUSPM:** Suspend mode interrupt mask

0: Suspend Mode Request (SUSP) Interrupt disabled.
1: SUSP Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 10 **RESETM:** USB reset interrupt mask

0: RESET Interrupt disabled.
1: RESET Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 9 **SOFM:** Start of frame interrupt mask

0: SOF Interrupt disabled.
1: SOF Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 8 **ESOFM:** Expected start of frame interrupt mask

0: Expected Start of Frame (ESOF) Interrupt disabled.
1: ESOF Interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bits 7:5 Reserved.

Bit 4 **RESUME:** Resume request

The microcontroller can set this bit to send a Resume signal to the host. It must be activated, according to USB specifications, for no less than 1 mS and no more than 15 mS after which the Host PC is ready to drive the resume sequence up to its end.

Bit 3 **FSUSP:** Force suspend

Software must set this bit when the SUSP interrupt is received, which is issued when no traffic is received by the USB peripheral for 3 mS.
0: No effect.
1: Enter suspend mode. Clocks and static power dissipation in the analog transceiver are left unaffected. If suspend power consumption is a requirement (bus-powered device), the application software should set the LP_MODE bit after FSUSP as explained below.

Bit 2 **LP_MODE:** Low-power mode

This mode is used when the suspend-mode power constraints require that all static power dissipation is avoided, except the one required to supply the external pull-up resistor. This condition should be entered when the application is ready to stop all system clocks, or reduce their frequency in order to meet the power consumption requirements of the USB suspend condition. The USB activity during the suspend mode (WKUP event) asynchronously resets this bit (it can also be reset by software).
0: No Low-power mode.
1: Enter Low-power mode.

Bit 1 **PDWN:** Power down

This bit is used to completely switch off all USB-related analog parts if it is required to completely disable the USB peripheral for any reason. When this bit is set, the USB peripheral is disconnected from the transceivers and it cannot be used.
0: Exit Power Down.
1: Enter Power down mode.

Bit 0 **FRES:** Force USB Reset

0: Clear USB reset.
1: Force a reset of the USB peripheral, exactly like a RESET signalling on the USB. The USB peripheral is held in RESET state until software clears this bit. A “USB-RESET” interrupt is generated, if enabled.

USB interrupt status register (USB_ISTR)

Address offset: 0x44

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR	PMA OVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	Reserved			DIR	EP_ID[3:0]			
r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0				r	r	r	r	r

This register contains the status of all the interrupt sources allowing application software to determine, which events caused an interrupt request.

The upper part of this register contains single bits, each of them representing a specific event. These bits are set by the hardware when the related event occurs; if the corresponding bit in the USB_CNTR register is set, a generic interrupt request is generated. The interrupt routine, examining each bit, will perform all necessary actions, and finally it will clear the serviced bits. If any of them is not cleared, the interrupt is considered to be still pending, and the interrupt line will be kept high again. If several bits are set simultaneously, only a single interrupt will be generated.

Endpoint transaction completion can be handled in a different way to reduce interrupt response latency. The CTR bit is set by the hardware as soon as an endpoint successfully completes a transaction, generating a generic interrupt request if the corresponding bit in USB_CNTR is set. An endpoint dedicated interrupt condition is activated independently from the CTRM bit in the USB_CNTR register. Both interrupt conditions remain active until software clears the pending bit in the corresponding USB_EPnR register (the CTR bit is actually a read only bit). For endpoint-related interrupts, the software can use the Direction of Transaction (DIR) and EP_ID read-only bits to identify, which endpoint made the last interrupt request and called the corresponding interrupt service routine.

The user can choose the relative priority of simultaneously pending USB_ISTR events by specifying the order in which software checks USB_ISTR bits in an interrupt service routine. Only the bits related to events, which are serviced, are cleared. At the end of the service routine, another interrupt will be requested, to service the remaining conditions.

To avoid spurious clearing of some bits, it is recommended to clear them with a load instruction where all bits which must not be altered are written with 1, and all bits to be cleared are written with ‘0 (these bits can only be cleared by software). Read-modify-write cycles should be avoided because between the read and the write operations another bit could be set by the hardware and the next write will clear it before the microprocessor has the time to serve the event.

The following describes each bit in detail:

Bit 15 **CTR:** Correct transfer

This bit is set by the hardware to indicate that an endpoint has successfully completed a transaction; using DIR and EP_ID bits software can determine which endpoint requested the interrupt. This bit is read-only.

Bit 14 **PMAOVR:** Packet memory area over / underrun

This bit is set if the microcontroller has not been able to respond in time to an USB memory request. The USB peripheral handles this event in the following way: During reception an ACK handshake packet is not sent, during transmission a bit-stuff error is forced on the transmitted stream; in both cases the host will retry the transaction. The PMAOVR interrupt should never occur during normal operations. Since the failed transaction is retried by the host, the application software has the chance to speed-up device operations during this interrupt handling, to be ready for the next transaction retry; however this does not happen during Isochronous transfers (no isochronous transaction is anyway retried) leading to a loss of data in this case. This bit is read/write but only ‘0 can be written and writing ‘1 has no effect.

Bit 13 **ERR:** Error

This flag is set whenever one of the errors listed below has occurred:
NANS: No ANSwer. The timeout for a host response has expired.
CRC: Cyclic Redundancy Check error. One of the received CRCs, either in the token or in the data, was wrong.
BST: Bit Stuffing error. A bit stuffing error was detected anywhere in the PID, data, and/or CRC.
FVIO: Framing format Violation. A non-standard frame was received (EOP not in the right place, wrong token sequence, etc.).
The USB software can usually ignore errors, since the USB peripheral and the PC host manage retransmission in case of errors in a fully transparent way. This interrupt can be useful during the software development phase, or to monitor the quality of transmission over the USB bus, to flag possible problems to the user (e.g. loose connector, too noisy environment, broken conductor in the USB cable and so on). This bit is read/write but only ‘0 can be written and writing ‘1 has no effect.

Bit 12 **WKUP:** Wakeup

This bit is set to 1 by the hardware when, during suspend mode, activity is detected that wakes up the USB peripheral. This event asynchronously clears the LP_MODE bit in the CTLR register and activates the USB_WAKEUP line, which can be used to notify the rest of the device (e.g. wakeup unit) about the start of the resume process. This bit is read/write but only ‘0 can be written and writing ‘1 has no effect.

Bit 11 **SUSP:** Suspend mode request

This bit is set by the hardware when no traffic has been received for 3mS, indicating a suspend mode request from the USB bus. The suspend condition check is enabled immediately after any USB reset and it is disabled by the hardware when the suspend mode is active (FSUSP=1) until the end of resume sequence. This bit is read/write but only ‘0 can be written and writing ‘1 has no effect.

Bit 10 **RESET:** USB reset request

Set when the USB peripheral detects an active USB RESET signal at its inputs. The USB peripheral, in response to a RESET, just resets its internal protocol state machine, generating an interrupt if RESETM enable bit in the USB_CNTR register is set. Reception and transmission are disabled until the RESET bit is cleared. All configuration registers do not reset: the microcontroller must explicitly clear these registers (this is to ensure that the RESET interrupt can be safely delivered, and any transaction immediately followed by a RESET can be completed). The function address and endpoint registers are reset by an USB reset event.

This bit is read/write but only ‘0 can be written and writing ‘1 has no effect.

Bit 9 **SOF:** Start of frame

This bit signals the beginning of a new USB frame and it is set when a SOF packet arrives through the USB bus. The interrupt service routine may monitor the SOF events to have a 1 mS synchronization event to the USB host and to safely read the USB_FNR register which is updated at the SOF packet reception (this could be useful for isochronous applications).

This bit is read/write but only ‘0 can be written and writing ‘1 has no effect.

Bit 8 **ESOF:** Expected start of frame

This bit is set by the hardware when an SOF packet is expected but not received. The host sends an SOF packet each mS, but if the hub does not receive it properly, the Suspend Timer issues this interrupt. If three consecutive ESOF interrupts are generated (i.e. three SOF packets are lost) without any traffic occurring in between, a SUSP interrupt is generated. This bit is set even when the missing SOF packets occur while the Suspend Timer is not yet locked. This bit is read/write but only ‘0 can be written and writing ‘1 has no effect.

Bits 7:5 Reserved.

Bit 4 **DIR:** Direction of transaction

This bit is written by the hardware according to the direction of the successful transaction, which generated the interrupt request.

If DIR bit=0, CTR_TX bit is set in the USB_EPnR register related to the interrupting endpoint. The interrupting transaction is of IN type (data transmitted by the USB peripheral to the host PC).

If DIR bit=1, CTR_RX bit or both CTR_TX/CTR_RX are set in the USB_EPnR register related to the interrupting endpoint. The interrupting transaction is of OUT type (data received by the USB peripheral from the host PC) or two pending transactions are waiting to be processed.

This information can be used by the application software to access the USB_EPnR bits related to the triggering transaction since it represents the direction having the interrupt pending. This bit is read-only.

Bits 3:0 **EP_ID[3:0]:** Endpoint Identifier

These bits are written by the hardware according to the endpoint number, which generated the interrupt request. If several endpoint transactions are pending, the hardware writes the endpoint identifier related to the endpoint having the highest priority defined in the following way: Two endpoint sets are defined, in order of priority: Isochronous and double-buffered bulk endpoints are considered first and then the other endpoints are examined. If more than one endpoint from the same set is requesting an interrupt, the EP_ID bits in USB_ISTR register are assigned according to the lowest requesting endpoint register, EP0R having the highest priority followed by EP1R and so on. The application software can assign a register to each endpoint according to this priority scheme, so as to order the concurring endpoint requests in a suitable way. These bits are read only.

USB frame number register (USB_FNR)

Address offset: 0x48

Reset value: 0x0XXX where X is undefined

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDP	RXDM	LCK	LSOF[1:0]		FN[10:0]										
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 15 **RXDP:** Receive data + line status

This bit can be used to observe the status of received data plus upstream port data line. It can be used during end-of-suspend routines to help determining the wakeup event.

Bit 14 **RXDM:** Receive data - line status

This bit can be used to observe the status of received data minus upstream port data line. It can be used during end-of-suspend routines to help determining the wakeup event.

Bit 13 **LCK:** Locked

This bit is set by the hardware when at least two consecutive SOF packets have been received after the end of an USB reset condition or after the end of an USB resume sequence. Once locked, the frame timer remains in this state until an USB reset or USB suspend event occurs.

Bits 12:11 **LSOF[1:0]:** Lost SOF

These bits are written by the hardware when an ESOF interrupt is generated, counting the number of consecutive SOF packets lost. At the reception of an SOF packet, these bits are cleared.

Bits 10:0 **FN[10:0]:** Frame number

This bit field contains the 11-bits frame number contained in the last received SOF packet. The frame number is incremented for every frame sent by the host and it is useful for Isochronous transfers. This bit field is updated on the generation of an SOF interrupt.

USB device address (USB_DADDR)

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								EF	ADD6	ADD5	ADD4	ADD3	ADD2	ADD1	ADD0
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved

Bit 7 **EF:** Enable function

This bit is set by the software to enable the USB device. The address of this device is contained in the following ADD[6:0] bits. If this bit is at ‘0 no transactions are handled, irrespective of the settings of USB_EPnR registers.

Bits 6:0 **ADD[6:0]:** Device address

These bits contain the USB function address assigned by the host PC during the enumeration process. Both this field and the Endpoint Address (EA) field in the associated USB_EPnR register must match with the information contained in a USB token in order to handle a transaction to the required endpoint.

Buffer table address (USB_BTABLE)

Address offset: 0x50

Reset value: 0x0000

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

Bits 15:3 **BTABLE[15:3]:** Buffer table

These bits contain the start address of the buffer allocation table inside the dedicated packet memory. This table describes each endpoint buffer location and size and it must be aligned to an 8 byte boundary (the 3 least significant bits are always ‘0). At the beginning of every transaction addressed to this device, the USP peripheral reads the element of this table related to the addressed endpoint, to get its buffer start location and the buffer size (Refer to *Structure and usage of packet buffers on page 549*).

Bits 2:0 Reserved, forced by hardware to 0.

23.5.2 Endpoint-specific registers

The number of these registers varies according to the number of endpoints that the USB peripheral is designed to handle. The USB peripheral supports up to 8 bidirectional endpoints. Each USB device must support a control endpoint whose address (EA bits) must be set to 0. The USB peripheral behaves in an undefined way if multiple endpoints are enabled having the same endpoint number value. For each endpoint, an USB_EPnR register is available to store the endpoint specific information.

USB endpoint n register (USB_EPnR), n=[0..7]

Address offset: 0x00 to 0x1C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR_ RX	DTOG_ RX	STAT_RX[1:0]		SETUP	EP TYPE[1:0]		EP_ KIND	CTR_ TX	DTOG_ TX	STAT_TX[1:0]		EA[3:0]			
rc_w0	t	t	t	r	rw	rw	rw	rc_w0	t	t	t	rw	rw	rw	rw

They are also reset when an USB reset is received from the USB bus or forced through bit FRES in the CTLR register, except the CTR_RX and CTR_TX bits, which are kept unchanged to avoid missing a correct packet notification immediately followed by an USB reset event. Each endpoint has its USB_EPnR register where *n* is the endpoint identifier.

Read-modify-write cycles on these registers should be avoided because between the read and the write operations some bits could be set by the hardware and the next write would modify them before the CPU has the time to detect the change. For this purpose, all bits affected by this problem have an ‘invariant’ value that must be used whenever their modification is not required. It is recommended to modify these registers with a load instruction where all the bits, which can be modified only by the hardware, are written with their ‘invariant’ value.

Bit 15 **CTR_RX:** Correct Transfer for reception

This bit is set by the hardware when an OUT/SETUP transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in USB_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated. The type of occurred transaction, OUT or SETUP, can be determined from the SETUP bit described below.

A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.

This bit is read/write but only ‘0 can be written, writing 1 has no effect.

Bit 14 **DTOG_RX:** Data Toggle, for reception transfers

If the endpoint is not Isochronous, this bit contains the expected value of the data toggle bit (0=DATA0, 1=DATA1) for the next data packet to be received. Hardware toggles this bit, when the ACK handshake is sent to the USB host, following a data packet reception having a matching data PID value; if the endpoint is defined as a control one, hardware clears this bit at the reception of a SETUP PID addressed to this endpoint.

If the endpoint is using the double-buffering feature this bit is used to support packet buffer swapping too (Refer to *Section 23.4.3: Double-buffered endpoints*).

If the endpoint is Isochronous, this bit is used only to support packet buffer swapping since no data toggling is used for this sort of endpoints and only DATA0 packet are transmitted (Refer to *Section 23.4.4: Isochronous transfers*). Hardware toggles this bit just after the end of data packet reception, since no handshake is used for isochronous transfers.

This bit can also be toggled by the software to initialize its value (mandatory when the endpoint is not a control one) or to force specific data toggle/packet buffer usage. When the application software writes ‘0, the value of DTOG_RX remains unchanged, while writing ‘1 makes the bit value toggle. This bit is read/write but it can be only toggled by writing 1.

Bits 13:12 **STAT_RX [1:0]:** Status bits, for reception transfers

These bits contain information about the endpoint status, which are listed in *Table 89: Reception status encoding on page 570*.These bits can be toggled by software to initialize their value. When the application software writes ‘0, the value remains unchanged, while writing ‘1 makes the bit value toggle. Hardware sets the STAT_RX bits to NAK when a correct transfer has occurred (CTR_RX=1) corresponding to a OUT or SETUP (control only) transaction addressed to this endpoint, so the software has the time to elaborate the received data before it acknowledge a new transaction

Double-buffered bulk endpoints implement a special transaction flow control, which control the status based upon buffer availability condition (Refer to *Section 23.4.3: Double-buffered endpoints*).

If the endpoint is defined as Isochronous, its status can be only “VALID” or “DISABLED”, so that the hardware cannot change the status of the endpoint after a successful transaction. If the software sets the STAT_RX bits to ‘STALL’ or ‘NAK’ for an Isochronous endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing ‘1.

Bit 11 **SETUP:** Setup transaction completed

This bit is read-only and it is set by the hardware when the last completed transaction is a SETUP. This bit changes its value only for control endpoints. It must be examined, in the case of a successful receive transaction (CTR_RX event), to determine the type of transaction occurred. To protect the interrupt service routine from the changes in SETUP bits due to next incoming tokens, this bit is kept frozen while CTR_RX bit is at 1; its state changes when CTR_RX is at 0. This bit is read-only.

Bits 10:9 **EP_TYPE[1:0]:** Endpoint type

These bits configure the behavior of this endpoint as described in *Table 90: Endpoint type encoding on page 570*. Endpoint 0 must always be a control endpoint and each USB function must have at least one control endpoint which has address 0, but there may be other control endpoints if required. Only control endpoints handle SETUP transactions, which are ignored by endpoints of other kinds. SETUP transactions cannot be answered with NAK or STALL. If a control endpoint is defined as NAK, the USB peripheral will not answer, simulating a receive error, in the receive direction when a SETUP transaction is received. If the control endpoint is defined as STALL in the receive direction, then the SETUP packet will be accepted anyway, transferring data and issuing the CTR interrupt. The reception of OUT transactions is handled in the normal way, even if the endpoint is a control one.

Bulk and interrupt endpoints have very similar behavior and they differ only in the special feature available using the EP_KIND configuration bit.

The usage of Isochronous endpoints is explained in *Section 23.4.4: Isochronous transfers*

Bit 8 **EP_KIND:** Endpoint kind

The meaning of this bit depends on the endpoint type configured by the EP_TYPE bits. *Table 91* summarizes the different meanings.

DBL_BUF: This bit is set by the software to enable the double-buffering feature for this bulk endpoint. The usage of double-buffered bulk endpoints is explained in *Section 23.4.3: Double-buffered endpoints*.

STATUS_OUT: This bit is set by the software to indicate that a status out transaction is expected: in this case all OUT transactions containing more than zero data bytes are answered ‘STALL’ instead of ‘ACK’. This bit may be used to improve the robustness of the application to protocol errors during control transfers and its usage is intended for control endpoints only. When STATUS_OUT is reset, OUT transactions can have any number of bytes, as required.

Bit 7 **CTR_TX:** Correct Transfer for transmission

This bit is set by the hardware when an IN transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in the USB_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated.

A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.

This bit is read/write but only ‘0 can be written.

Bit 6 **DTOG_TX:** Data Toggle, for transmission transfers

If the endpoint is non-isochronous, this bit contains the required value of the data toggle bit (0=DATA0, 1=DATA1) for the next data packet to be transmitted. Hardware toggles this bit when the ACK handshake is received from the USB host, following a data packet transmission. If the endpoint is defined as a control one, hardware sets this bit to 1 at the reception of a SETUP PID addressed to this endpoint.

If the endpoint is using the double buffer feature, this bit is used to support packet buffer swapping too (Refer to *Section 23.4.3: Double-buffered endpoints*)

If the endpoint is Isochronous, this bit is used to support packet buffer swapping since no data toggling is used for this sort of endpoints and only DATA0 packet are transmitted (Refer to *Section 23.4.4: Isochronous transfers*). Hardware toggles this bit just after the end of data packet transmission, since no handshake is used for Isochronous transfers.

This bit can also be toggled by the software to initialize its value (mandatory when the endpoint is not a control one) or to force a specific data toggle/packet buffer usage. When the application software writes ‘0, the value of DTOG_TX remains unchanged, while writing ‘1 makes the bit value toggle. This bit is read/write but it can only be toggled by writing 1.

Bits 5:4 **STAT_TX [1:0]:** Status bits, for transmission transfers

These bits contain the information about the endpoint status, listed in *Table 92*. These bits can be toggled by the software to initialize their value. When the application software writes ‘0, the value remains unchanged, while writing ‘1 makes the bit value toggle. Hardware sets the STAT_TX bits to NAK, when a correct transfer has occurred (CTR_TX=1) corresponding to a IN or SETUP (control only) transaction addressed to this endpoint. It then waits for the software to prepare the next set of data to be transmitted.

Double-buffered bulk endpoints implement a special transaction flow control, which controls the status based on buffer availability condition (Refer to *Section 23.4.3: Double-buffered endpoints*).

If the endpoint is defined as Isochronous, its status can only be “VALID” or “DISABLED”. Therefore, the hardware cannot change the status of the endpoint after a successful transaction. If the software sets the STAT_TX bits to ‘STALL’ or ‘NAK’ for an Isochronous endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing ‘1.

Bits 3:0 **EA[3:0]:** Endpoint address

Software must write in this field the 4-bit address used to identify the transactions directed to this endpoint. A value must be written before enabling the corresponding endpoint.

Table 89. Reception status encoding

STAT_RX[1:0]	Meaning
00	**DISABLED:** all reception requests addressed to this endpoint are ignored.
01	**STALL**: the endpoint is stalled and all reception requests result in a STALL handshake.
10	**NAK**: the endpoint is naked and all reception requests result in a NAK handshake.
11	**VALID**: this endpoint is enabled for reception.

Table 90. Endpoint type encoding

EP_TYPE[1:0]	Meaning
00	BULK
01	CONTROL

Table 90. Endpoint type encoding (continued)

EP_TYPE[1:0]	Meaning
10	ISO
11	INTERRUPT

Table 91. Endpoint kind meaning

EP_TYPE[1:0]		EP_KIND Meaning
00	BULK	DBL_BUF
01	CONTROL	STATUS_OUT
10	ISO	Not used
11	INTERRUPT	Not used

Table 92. Transmission status encoding

STAT_TX[1:0]	Meaning
00	**DISABLED:** all transmission requests addressed to this endpoint are ignored.
01	**STALL**: the endpoint is stalled and all transmission requests result in a STALL handshake.
10	**NAK**: the endpoint is naked and all transmission requests result in a NAK handshake.
11	**VALID**: this endpoint is enabled for transmission.

23.5.3 Buffer descriptor table

Although the buffer descriptor table is located inside the packet buffer memory, its entries can be considered as additional registers used to configure the location and size of the packet buffers used to exchange data between the USB macro cell and the STM32L1xxxx. Due to the common APB bridge limitation on word addressability, all packet memory locations are accessed by the APB using 32-bit aligned addresses, instead of the actual memory location addresses utilized by the USB peripheral for the USB_BTABLE register and buffer description table locations.

In the following pages two location addresses are reported: the one to be used by application software while accessing the packet memory, and the local one relative to USB Peripheral access. To obtain the correct STM32L1xxxx memory address value to be used in the application software while accessing the packet memory, the actual memory location address must be multiplied by two. The first packet memory location is located at 0x4000 6000. The buffer descriptor table entry associated with the USB_EPnR registers is described below.

A thorough explanation of packet buffers and the buffer descriptor table usage can be found in *Structure and usage of packet buffers on page 549*.

Transmission buffer address n (USB_ADDRn_TX)

Address offset: [USB_BTABLE] + n*16

USB local address: [USB_BTABLE] + n*8

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

Bits 15:1 **ADDRn_TX[15:1]:** Transmission buffer address

These bits point to the starting address of the packet buffer containing data to be transmitted by the endpoint associated with the USB_EPnR register at the next IN token addressed to it.

Bit 0 Must always be written as ‘0 since packet memory is word-wide and all packet buffers must be word-aligned.

Transmission byte count n (USB_COUNTn_TX)

Address offset: [USB_BTABLE] + n*16 + 4

USB local Address: [USB_BTABLE] + n*8 + 2

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

Bits 15:10 These bits are not used since packet size is limited by USB specifications to 1023 bytes. Their value is not considered by the USB peripheral.

Bits 9:0 **COUNTn_TX[9:0]:** Transmission byte count

These bits contain the number of bytes to be transmitted by the endpoint associated with the USB_EPnR register at the next IN token addressed to it.

Note: Double-buffered and Isochronous IN Endpoints have two USB_COUNTn_TX registers: named USB_COUNTn_TX_1 and USB_COUNTn_TX_0 with the following content.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						COUNTn_TX_1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						COUNTn_TX_0[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Reception buffer address n (USB_ADDRn_RX)

Address offset: [USB_BTABLE] + n*16 + 8

USB local Address: [USB_BTABLE] + n*8 + 4

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

Bits 15:1 **ADDRn_RX[15:1]:** Reception buffer address

These bits point to the starting address of the packet buffer, which will contain the data received by the endpoint associated with the USB_EPnR register at the next OUT/SETUP token addressed to it.

Bit 0 This bit must always be written as ‘0 since packet memory is word-wide and all packet buffers must be word-aligned.

Reception byte count n (USB_COUNTn_RX)

Address offset: [USB_BTABLE] + n*16 + 12

USB local Address: [USB_BTABLE] + n*8 + 6

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLSIZE	NUM_BLOCK[4:0]					COUNTn_RX[9:0]									
rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r	r	r

This table location is used to store two different values, both required during packet reception. The most significant bits contains the definition of allocated buffer size, to allow buffer overflow detection, while the least significant part of this location is written back by the USB peripheral at the end of reception to give the actual number of received bytes. Due to the restrictions on the number of available bits, buffer size is represented using the number of allocated memory blocks, where block size can be selected to choose the trade-off between fine-granularity/small-buffer and coarse-granularity/large-buffer. The size of allocated buffer is a part of the endpoint descriptor and it is normally defined during the enumeration process according to its maxPacketSize parameter value (See “Universal Serial Bus Specification”).

Bit 15 **BL_SIZE:** BLock size

This bit selects the size of memory block used to define the allocated buffer area.

– If BL_SIZE=0, the memory block is 2 byte large, which is the minimum block allowed in a word-wide memory. With this block size the allocated buffer size ranges from 2 to 62 bytes.

– If BL_SIZE=1, the memory block is 32 `byte` large, which allows to reach the maximum packet length defined by USB specifications. With this block size the allocated buffer size ranges from 32 to 1024 bytes, which is the longest packet size allowed by USB standard specifications.

Bits 14:10 **NUM_BLOCK[4:0]:** Number of blocks

These bits define the number of memory blocks allocated to this packet buffer. The actual amount of allocated memory depends on the BL_SIZE value as illustrated in *Table 93*.

Bits 9:0 **COUNTn_RX[9:0]:** Reception byte count

These bits contain the number of bytes received by the endpoint associated with the USB_EPnR register during the last OUT/SETUP transaction addressed to it.

Note: Double-buffered and Isochronous IN Endpoints have two USB_COUNTn_TX registers: named USB_COUNTn_TX_1 and USB_COUNTn_TX_0 with the following content.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLSIZE _1	NUM_BLOCK_1[4:0]					COUNTn_RX_1[9:0]									
rw	rw	rw	rw	rw	rw	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
BLSIZE _0	NUM_BLOCK_0[4:0]					COUNTn_RX_0[9:0]									
rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r	r	r

Table 93. Definition of allocated buffer memory

Value of NUM_BLOCK[4:0]	Memory allocated when BL_SIZE=0	Memory allocated when BL_SIZE=1
0 (‘00000)	Not allowed	32 bytes
1 (‘00001)	2 bytes	64 bytes
2 (‘00010)	4 bytes	96 bytes
3 (‘00011)	6 bytes	128 bytes
...	...	...
15 (‘01111)	30 bytes	512 bytes
16 (‘10000)	32 bytes	N/A
17 (‘10001)	34 bytes	N/A
18 (‘10010)	36 bytes	N/A
...	...	...
30 (‘11110)	60 bytes	N/A
31 (‘11111)	62 bytes	N/A

23.5.4 USB register map

The table below provides the USB register map and reset values.

Table 94. USB register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USB_EP0R	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	USB_EP1R	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	USB_EP2R	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	USB_EP3R	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 94. USB register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x10	USB_EP4R	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	USB_EP5R	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	USB_EP6R	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	USB_EP7R	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20-0x3F	Reserved																																
0x40	USB_CNTR	Reserved																CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RESETM	SOFM	ESOFM	Reserved			RESUME	FSUSP	LPMODE	PDWN	FRES
	Reset value																	0	0	0	0	0	0	0	0				0	0	0	1	1
0x44	USB_ISTR	Reserved																CTR	PMAOVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	Reserved			DIR	EP_ID[3:0]			
	Reset value																	0	0	0	0	0	0	0	0				0	0	0	0	0
0x48	USB_FNR	Reserved																RXDP	RXDM	LCK	LSOF [1:0]		FN[10:0]										
	Reset value																	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x
0x4C	USB_DADDR	Reserved																								EF	ADD[6:0]						
	Reset value																									0	0	0	0	0	0	0	0
0x50	USB_BTABLE	Reserved																BTABLE[15:3]													Reserved		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0			

Refer to *Table 2 on page 42* for the register boundary addresses.

24 Flexible static memory controller (FSMC)

This section applies to high density devices only.

24.1 FSMC main features

The FSMC block is able to interface with synchronous and asynchronous memories. Its main purpose is to:

- Translate the AHB transactions into the appropriate external device protocol
- Meet the access timing requirements of the external devices

All external memories share the addresses, data and control signals with the controller. Each external device is accessed by means of a unique chip select. The FSMC performs only one access at a time to an external device.

The FSMC has the following main features:

- Interfaces with static memory-mapped devices including:
  - Static random access memory (SRAM)
  - Read-only memory (ROM)
  - NOR Flash memory/OneNAND Flash memory
  - PSRAM (4 memory banks)
- Supports burst mode access to synchronous devices (NOR Flash and PSRAM)
- 8- or 16-bit wide databus
- Independent chip select control for each memory bank
- Independent configuration for each memory bank
- Programmable timings to support a wide range of devices, in particular:
  - Programmable wait states (up to 15)
  - Programmable bus turnaround cycles (up to 15)
  - Programmable output enable and write enable delays (up to 15)
  - Independent read and write timings and protocol, so as to support the widest variety of memories and timings
- Write enable and byte lane select outputs for use with PSRAM and SRAM devices
- Translation of 32-bit wide AHB transactions into consecutive 16-bit or 8-bit accesses to external 16-bit or 8-bit devices
- A Write FIFO, 2-word long , each word is 32 bits wide, only stores data and not the address. Therefore, this FIFO only buffers AHB write burst transactions. This makes it possible to write to slow memories and free the AHB quickly for other operations. Only one burst at a time is buffered: if a new AHB burst or single transaction occurs while an operation is in progress, the FIFO is drained. The FSMC will insert wait states until the current memory access is complete).
- External asynchronous wait control

The FSMC registers that define the external device type and associated characteristics are usually set at boot time and do not change until the next reset or power-up. However, it is possible to change the settings at any time.

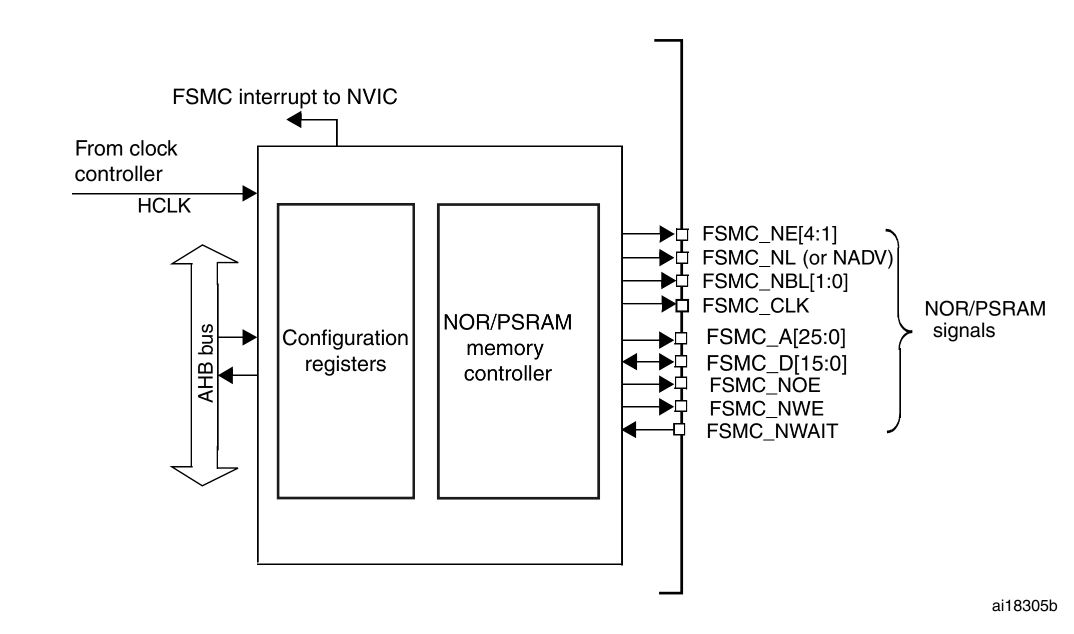
24.2 Block diagram

The FSMC consists of four main blocks:

- The AHB interface (including the FSMC configuration registers)
- The NOR Flash/PSRAM controller
- The external device interface

The block diagram is shown in *Figure 186*.

Figure 186. FSMC block diagram



24.3 AHB interface

The AHB slave interface enables internal CPUs and other bus master peripherals to access the external static memories.

AHB transactions are translated into the external device protocol. In particular, if the selected external memory is 16 or 8 bits wide, 32-bit wide transactions on the AHB are split into consecutive 16- or 8-bit accesses. The Chip Select toggles for each access.

The FSMC generates an AHB error in the following conditions:

- When reading or writing to an FSMC bank which is not enabled
- When reading or writing to the NOR Flash bank while the FACCEN bit is reset in the FSMC_BCRx register.

The effect of this AHB error depends on the AHB master which has attempted the R/W access:

- If it is the Cortex™-M3 CPU, a hard fault interrupt is generated
- If is a DMA, a DMA transfer error is generated and the corresponding DMA channel is automatically disabled.

The AHB clock (HCLK) is the reference clock for the FSMC.

24.3.1 Supported memories and transactions

General transaction rules

The requested AHB transaction data size can be 8-, 16- or 32-bit wide whereas the accessed external device has a fixed data width. This may lead to inconsistent transfers.

Therefore, some simple transaction rules must be followed:

- AHB transaction size and memory data size are equal
  There is no issue in this case.
- AHB transaction size is greater than the memory size
  In this case, the FSMC splits the AHB transaction into smaller consecutive memory accesses in order to meet the external data width.
- AHB transaction size is smaller than the memory size
  Asynchronous transfers may or not be consistent depending on the type of external device.
  - Asynchronous accesses to devices that have the byte select feature (SRAM, ROM, PSRAM).

    a) FSMC allows write transactions accessing the right data through its byte lanes NBL[1:0]

    b) Read transactions are allowed. All memory bytes are read and the useless ones are discarded. The NBL[1:0] are kept low during read transactions.
  - Asynchronous accesses to devices that do not have the byte select feature (NOR).
    This situation occurs when a byte access is requested to a 16-bit wide Flash memory. Clearly, the device cannot be accessed in byte mode (only 16-bit words can be read from/written to the Flash memory) therefore:

  a) Write transactions are not allowed

  b) Read transactions are allowed. All memory bytes are read and the useless ones are discarded. The NBL[1:0] are set to 0 during read transactions.

Configuration registers

The FSMC can be configured using a register set. See *Section 24.5.6*, for a detailed description of the NOR Flash/PSRAM control registers.

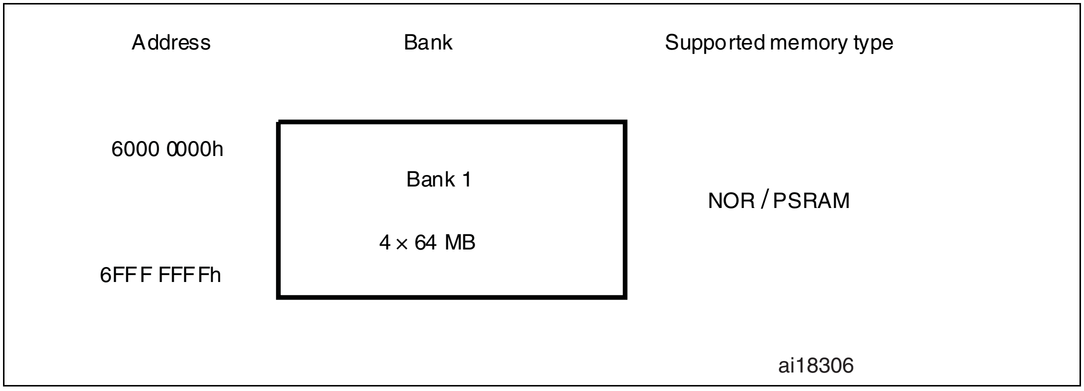
24.4 External device address mapping

From the FSMC point of view, the external memory is composed of a single fixed size bank of 256 Mbytes (Refer to *Figure 187*):

- Bank 1 used to address up to 4 NOR Flash or PSRAM memory devices. This bank is split into 4 NOR/PSRAM regions with 4 dedicated Chip Select.

For each bank the type of memory to be used is user-defined in the Configuration register.

Figure 187. FSMC memory banks



24.4.1 NOR/PSRAM address mapping

HADDR[27:26] bits are used to select one of the four memory banks as shown in *Table 95*.

Table 95. NOR/PSRAM bank selection

HADDR[27:26][1]	Selected bank
00	Bank 1 NOR/PSRAM 1
01	Bank 1 NOR/PSRAM 2
10	Bank 1 NOR/PSRAM 3
11	Bank 1 NOR/PSRAM 4

[1] HADDR are internal AHB address lines that are translated to external memory.

HADDR[25:0] contain the external memory address. Since HADDR is a byte address whereas the memory is addressed in words, the address actually issued to the memory varies according to the memory data width, as shown in the following table.

Table 96. External memory address

Memory width[1]	Data address issued to the memory	Maximum memory capacity (bits)
8-bit	HADDR[25:0]	64 Mbytes x 8 = 512 Mbit
16-bit	HADDR[25:1] >> 1	64 Mbytes/2 x 16 = 512 Mbit

[1] In case of a 16-bit external memory width, the FSMC will internally use HADDR[25:1] to generate the address for external memory FSMC_A[24:0].
Whatever the external memory width (16-bit or 8-bit), FSMC_A[0] should be connected to external memory address A[0].

Wrap support for NOR Flash/PSRAM

Wrap burst mode for synchronous memories is not supported. The memories must be configured in linear burst mode of undefined length.

24.5 NOR Flash/PSRAM controller

The FSMC generates the appropriate signal timings to drive the following types of memories:

- Asynchronous SRAM and ROM
  - 8-bit
  - 16-bit
  - 32-bit
- PSRAM (Cellular RAM)
  - Asynchronous mode
  - Burst mode
- NOR Flash
  - Asynchronous mode or burst mode
  - Multiplexed or nonmultiplexed

The FSMC outputs a unique chip select signal NE[4:1] per bank. All the other signals (addresses, data and control) are shared.

For synchronous accesses, the FSMC issues the clock (CLK) to the selected external device only during the read/write transactions. This clock is a submultiple of the HCLK clock. The size of each bank is fixed and equal to 64 Mbytes.

Each bank is configured by means of dedicated registers (see *Section 24.5.6*).

The programmable memory parameters include access timings (see *Table 97*) and support for wait management (for PSRAM and NOR Flash accessed in burst mode).

Table 97. Programmable NOR/PSRAM access parameters

Parameter	Function	Access mode	Unit	Min.	Max.
Address setup	Duration of the address setup phase	Asynchronous	AHB clock cycle (HCLK)	0	15
Address hold	Duration of the address hold phase	Asynchronous, muxed I/Os	AHB clock cycle (HCLK)	1	15
Data setup	Duration of the data setup phase	Asynchronous	AHB clock cycle (HCLK)	1	256
Bust turn	Duration of the bus turnaround phase	Asynchronous and synchronous read	AHB clock cycle (HCLK)	0	15
Clock divide ratio	Number of AHB clock cycles (HCLK) to build one memory clock cycle (CLK)	Synchronous	AHB clock cycle (HCLK)	2	16
Data latency	Number of clock cycles to issue to the memory before the first data of the burst	Synchronous	Memory clock cycle (CLK)	2	17

24.5.1 External memory interface signals

Table 98, *Table 99* and *Table 100* list the signals that are typically used to interface NOR Flash, SRAM and PSRAM.

Note: Prefix "N". specifies the associated signal as active low.

NOR Flash, nonmultiplexed I/Os

Table 98. Nonmultipled I/O NOR Flash

FSMC signal name	I/O	Function
CLK	O	Clock (for synchronous burst)
A[25:0]	O	Address bus
D[15:0]	I/O	Bidirectional data bus
NE[x]	O	Chip select, x = 1..4
NOE	O	Output enable
NWE	O	Write enable
NL(=NADV)	O	Latch enable (this signal is called address valid, NADV, by some NOR Flash devices)
NWAIT	I	NOR Flash wait input signal to the FSMC

NOR Flash memories are addressed in 16-bit words. The maximum capacity is 512 Mbit (26 address lines).

NOR Flash, multiplexed I/Os

Table 99. Multiplexed I/O NOR Flash

FSMC signal name	I/O	Function
CLK	O	Clock (for synchronous burst)
A[25:16]	O	Address bus
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus
NE[x]	O	Chip select, x = 1..4
NOE	O	Output enable
NWE	O	Write enable
NL(=NADV)	O	Latch enable (this signal is called address valid, NADV, by some NOR Flash devices)
NWAIT	I	NOR Flash wait input signal to the FSMC

NOR-Flash memories are addressed in 16-bit words. The maximum capacity is 512 Mbit (26 address lines).

PSRAM/SRAM, nonmultiplexed I/Os

Table 100. Nonmultiplexed I/Os PSRAM/SRAM

FSMC signal name	I/O	Function
CLK	O	Clock (only for PSRAM synchronous burst)
A[25:0]	O	Address bus

Table 100. Nonmultiplexed I/Os PSRAM/SRAM (continued)

FSMC signal name	I/O	Function
D[15:0]	I/O	Data bidirectional bus
NE[x]	O	Chip select, x = 1..4 (called NCE by PSRAM (Cellular RAM i.e. CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid only for PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FSMC
NBL[1]	O	Upper byte enable (memory signal name: NUB)
NBL[0]	O	Lowed byte enable (memory signal name: NLB)

PSRAM memories are addressed in 16-bit words. The maximum capacity is 512 Mbit (26 address lines).

PSRAM, multiplexed I/Os

Table 101. Multiplexed I/O PSRAM

FSMC signal name	I/O	Function
CLK	O	Clock (for synchronous burst)
A[25:16]	O	Address bus
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus
NE[x]	O	Chip select, x = 1..4 (called NCE by PSRAM (Cellular RAM i.e. CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FSMC
NBL[1]	O	Upper byte enable (memory signal name: NUB)
NBL[0]	O	Lowed byte enable (memory signal name: NLB)

PSRAM memories are addressed in 16-bit words. The maximum capacity is 512 Mbit (26 address lines).

24.5.2 Supported memories and transactions

Table 102 below displays an example of the supported devices, access modes and transactions when the memory data bus is 16-bit for NOR, PSRAM and SRAM. Transactions not allowed (or not supported) by the FSMC in this example appear in gray.

Table 102. NOR Flash/PSRAM controller: example of supported memories and transactions

Device	Mode	R/W	AHB data size	Memory data size	Allowed/ not allowed	Comments
NOR Flash (muxed I/Os and nonmuxed I/Os)	Asynchronous	R	8	16	Y	
	Asynchronous	W	8	16	N	
	Asynchronous	R	16	16	Y	
	Asynchronous	W	16	16	Y	
	Asynchronous	R	32	16	Y	Split into 2 FSMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FSMC accesses
	Asynchronous page	R	-	16	N	Mode is not supported
	Synchronous	R	8	16	N	
	Synchronous	R	16	16	Y	
	Synchronous	R	32	16	Y	
PSRAM (multiplexed I/Os and nonmultiplexed I/Os)	Asynchronous	R	8	16	Y	
	Asynchronous	W	8	16	Y	Use of byte lanes NBL[1:0]
	Asynchronous	R	16	16	Y	
	Asynchronous	W	16	16	Y	
	Asynchronous	R	32	16	Y	Split into 2 FSMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FSMC accesses
	Asynchronous page	R	-	16	N	Mode is not supported
	Synchronous	R	8	16	N	
	Synchronous	R	16	16	Y	
	Synchronous	R	32	16	Y	
	Synchronous	W	8	16	Y	Use of byte lanes NBL[1:0]
	Synchronous	W	16/32	16	Y	
SRAM and ROM	Asynchronous	R	8 / 16	16	Y	
	Asynchronous	W	8 / 16	16	Y	Use of byte lanes NBL[1:0]
	Asynchronous	R	32	16	Y	Split into 2 FSMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FSMC accesses. Use of byte lanes NBL[1:0]

24.5.3 General timing rules

Signals synchronization

- All controller output signals change on the rising edge of the internal clock (HCLK)
- In synchronous mode (read or write), all output signals change on the rising edge of HCLK. Whatever the CLKDIV value, all outputs change as follows:
  - NOEL/NWEL/ NEL/NADVL/ NADVH /NBLL/ Address valid outputs change on the falling edge of FSMC_CLK clock.
  - NOEH/ NWEH / NEH/ NOEH/NBLH/ Address invalid outputs change on the rising edge of FSMC_CLK clock.

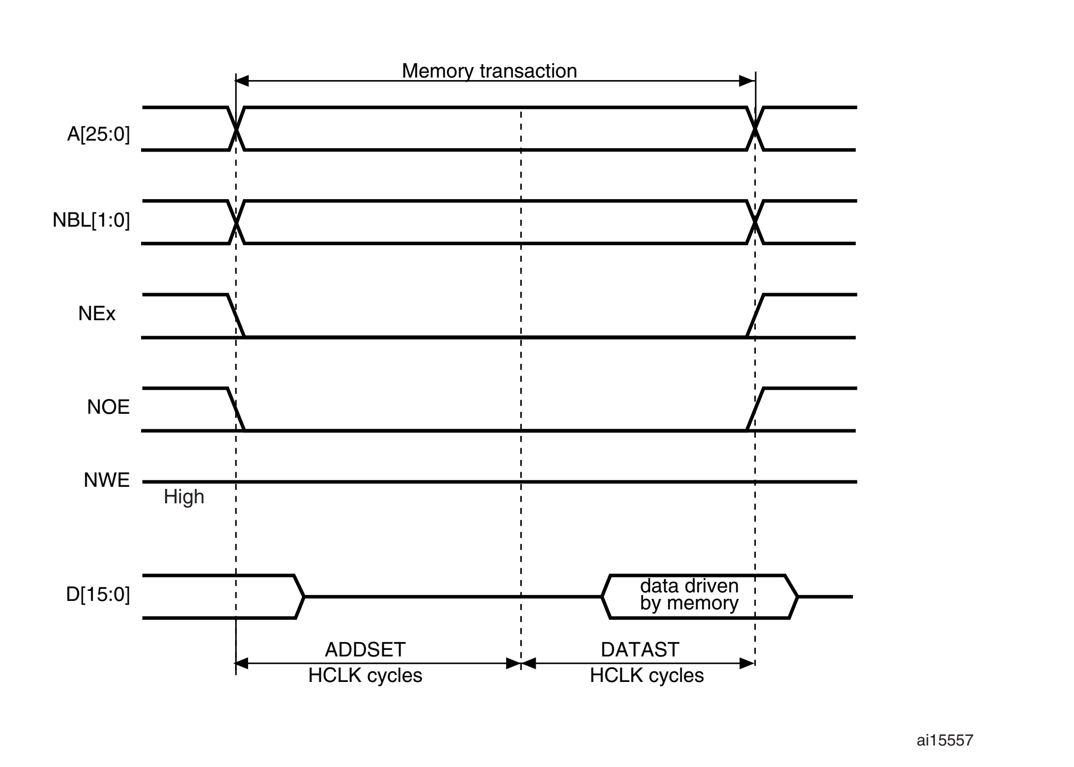
24.5.4 NOR Flash/PSRAM controller asynchronous transactions

Asynchronous static memories (NOR Flash memory, PSRAM, SRAM)

- Signals are synchronized by the internal clock HCLK. This clock is not issued to the memory
- The FSMC always samples the data before de-asserting the chip select signal NE. This guarantees that the memory data-hold timing constraint is met (chip enable high to data transition, usually 0 ns min.)
- If the extended mode is enabled (EXTMOD bit is set in the FSMC_BCRx register), up to four extended modes (A, B, C and D) are available. It is possible to mix A, B, C and D modes for read and write operations. For example, read operation can be performed in mode A and write in mode B.
- If the extended mode is disabled (EXTMOD bit is reset in the FSMC_BCRx register), the FSMC can operate in Mode1 or Mode2 as follows:
  - Mode 1 is the default mode when SRAM/CRAM memory type is selected (MTYP = 0x0 or 0x01 in the FSMC_BCRx register)
  - Mode 2 is the default mode when NOR memory type is selected (MTYP = 0x10 in the FSMC_BCRx register).

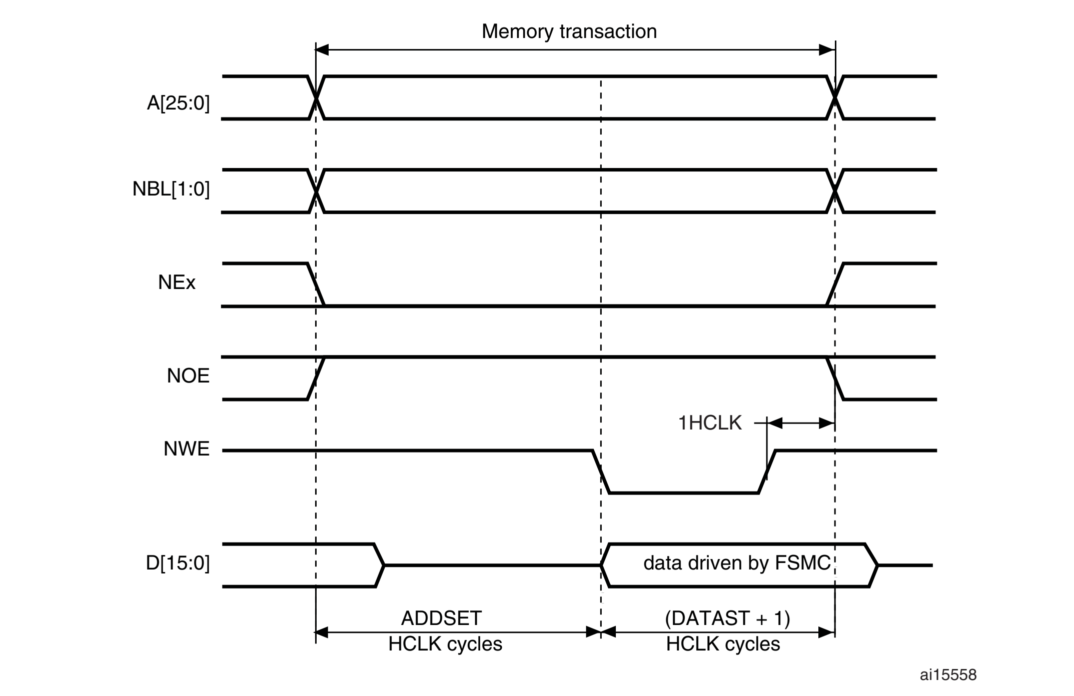
Mode 1 - SRAM/PSRAM (CRAM)

Figure 188. Mode1 read accesses



1. NBL[1:0] are driven low during read access.

Figure 189. Mode1 write accesses



The one HCLK cycle at the end of the write transaction helps guarantee the address and data hold time after the NWE rising edge. Due to the presence of this one HCLK cycle, the DATAST value must be greater than zero (DATAST > 0).

Table 103. FSMC_BCRx bit fields

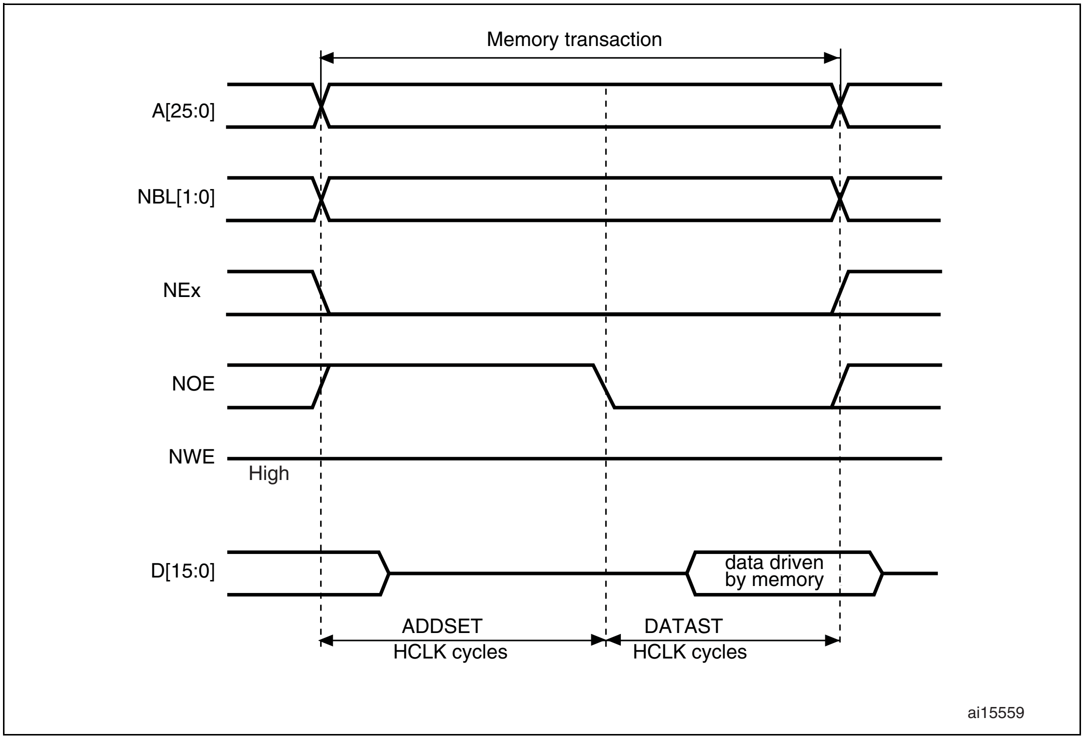
Bit number	Bit name	Value to set
31-20	Reserved	0x000
19	CBURSTRW	0x0 (no effect on asynchronous mode)
18:16	Reserved	0x0
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x0
13	WAITEN	0x0 (no effect on asynchronous mode)
12	WREN	0x1
11	WAITCFG	As needed
10	WRAPMOD	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	Don’t care
5-4	MWID	As needed
3-2	MTYP	As needed, exclude 0x2 (NOR Flash)
1	MUXE	0x0
0	MBKEN	0x1

Table 104. FSMC_BTRx bit fields

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	Don’t care
23-20	CLKDIV	Don’t care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST+1 HCLK cycles for write accesses, DATAST HCLK cycles for read accesses).
7-4	ADDHLD	Don’t care
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles). Minimum value for ADDSET is 0.

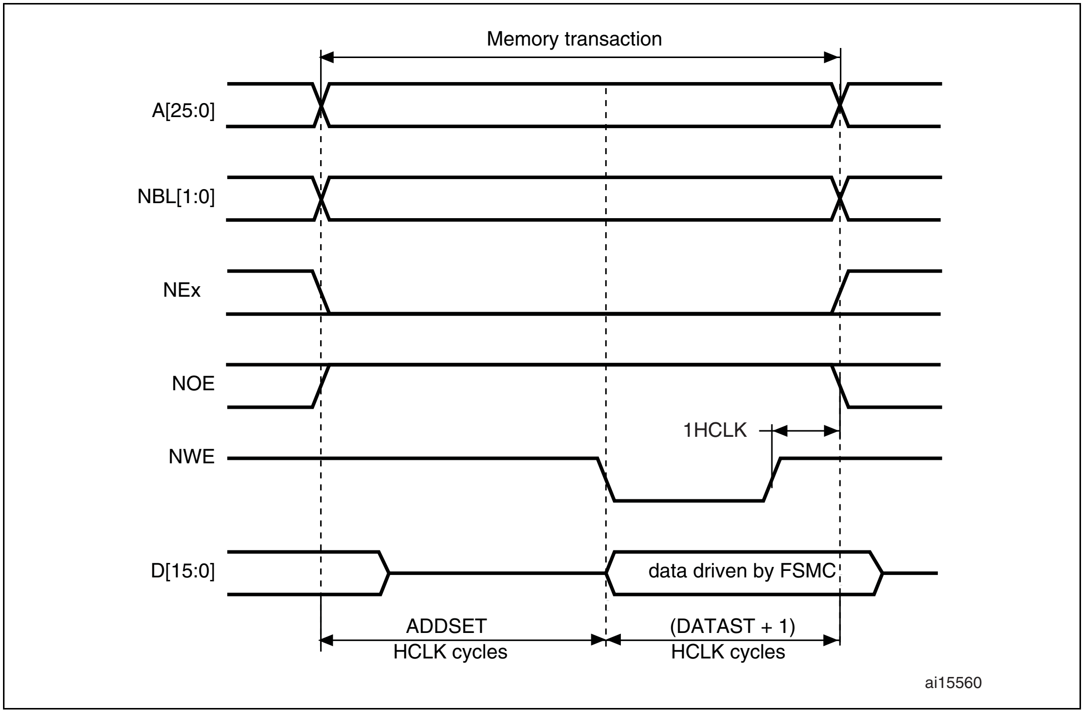
Mode A - SRAM/PSRAM (CRAM) OE toggling

Figure 190. ModeA read accesses



1. NBL[1:0] are driven low during read access.

Figure 191. ModeA write accesses



The differences compared with mode1 are the toggling of NOE and the independent read and write timings.

Table 105. FSMC_BCRx bit fields

Bit number	Bit name	Value to set
31-20	Reserved	0x000
19	CBURSTRW	0x0 (no effect on asynchronous mode)
18:16	Reserved	0x0
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect on asynchronous mode)
12	WREN	0x1
11	WAITCFG	As needed
10	WRAPMOD	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	Don’t care
5-4	MWID	As needed
3-2	MTYP	As needed, exclude 0x2 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Table 106. FSMC_BTRx bit fields

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	Don’t care
23-20	CLKDIV	Don’t care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST HCLK cycles) for read accesses.
7-4	ADDHLD	Don’t care
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for read accesses. Minimum value for ADDSET is 0.

Table 107. FSMC_BWTRx bit fields

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	Don't care
23-20	CLKDIV	Don't care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST+1 HCLK cycles for write accesses,
7-4	ADDHLD	Don't care
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for write accesses. Minimum value for ADDSET is 0.

Mode 2/B - NOR Flash

Figure 192. Mode2 and mode B read accesses

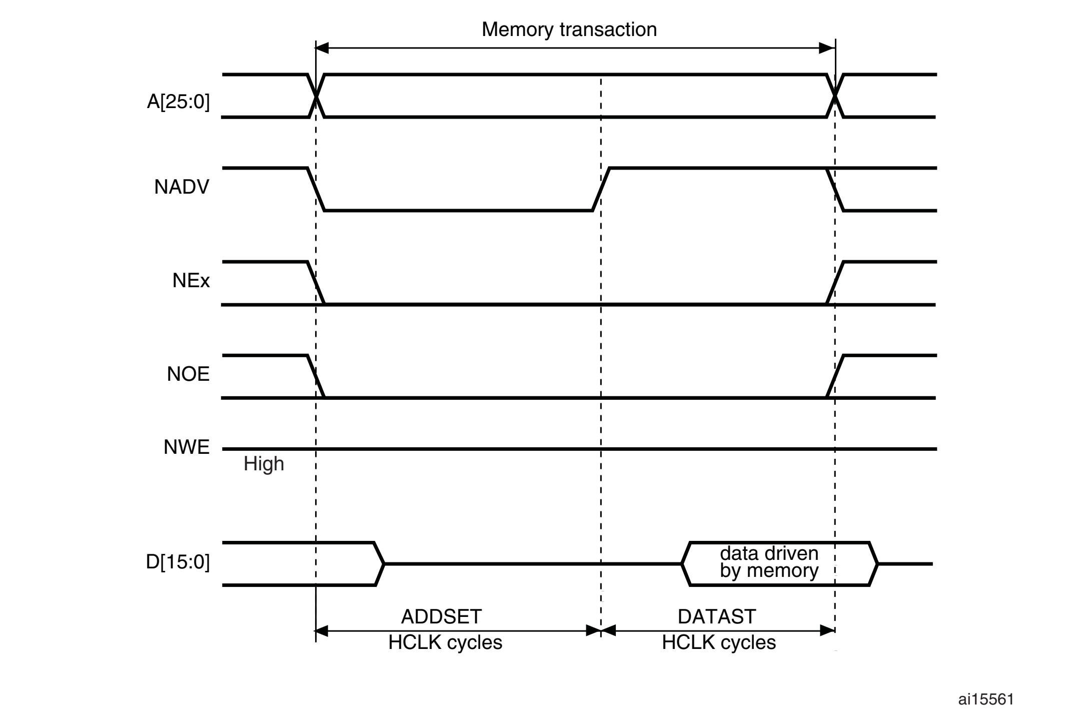


Figure 193. Mode2 write accesses

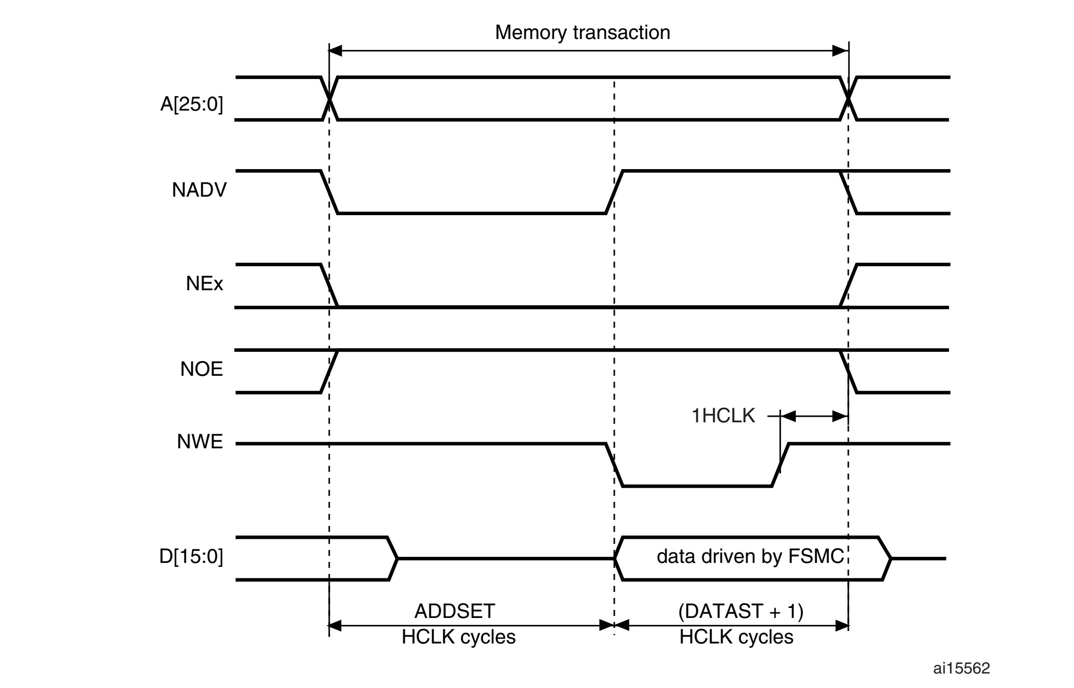
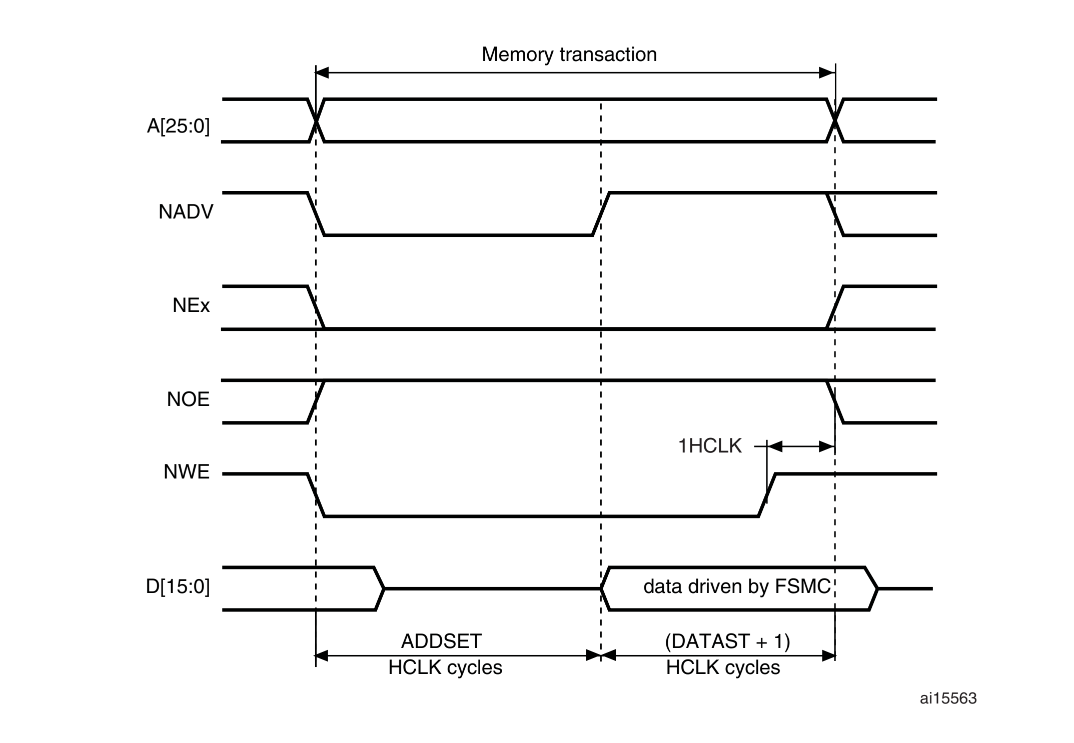


Figure 194. ModeB write accesses



The differences with mode1 are the toggling of NWE and the independent read and write timings when extended mode is set (Mode B).

Table 108. FSMC_BCRx bit fields

Bit number	Bit name	Value to set
31-20	Reserved	0x000
19	CBURSTRW	0x0 (no effect on asynchronous mode)
18:16	Reserved	0x0
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1 for mode B, 0x0 for mode 2
13	WAITEN	0x0 (no effect on asynchronous mode)
12	WREN	0x1
11	WAITCFG	As needed
10	WRAPMOD	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1
5-4	MWID	As needed
3-2	MTYP	0x2 (NOR Flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 109. FSMC_BTRx bit fields

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29-28	ACCMOD	0x1
27-24	DATLAT	Don’t care
23-20	CLKDIV	Don’t care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST HCLK cycles) for read accesses.
7-4	ADDHLD	Don’t care
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for read accesses. Minimum value for ADDSET is 0.

Table 110. FSMC_BWTRx bit fields

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29-28	ACCMOD	0x1
27-24	DATLAT	Don’t care
23-20	CLKDIV	Don’t care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST+1 HCLK cycles for write accesses,
7-4	ADDHLD	Don’t care
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for write accesses. Minimum value for ADDSET is 0.

Note: The FSMC_BWTRx register is valid only if extended mode is set (mode B), otherwise all its content is don’t care.

Mode C - NOR Flash - OE toggling

Figure 195. Mode C read accesses

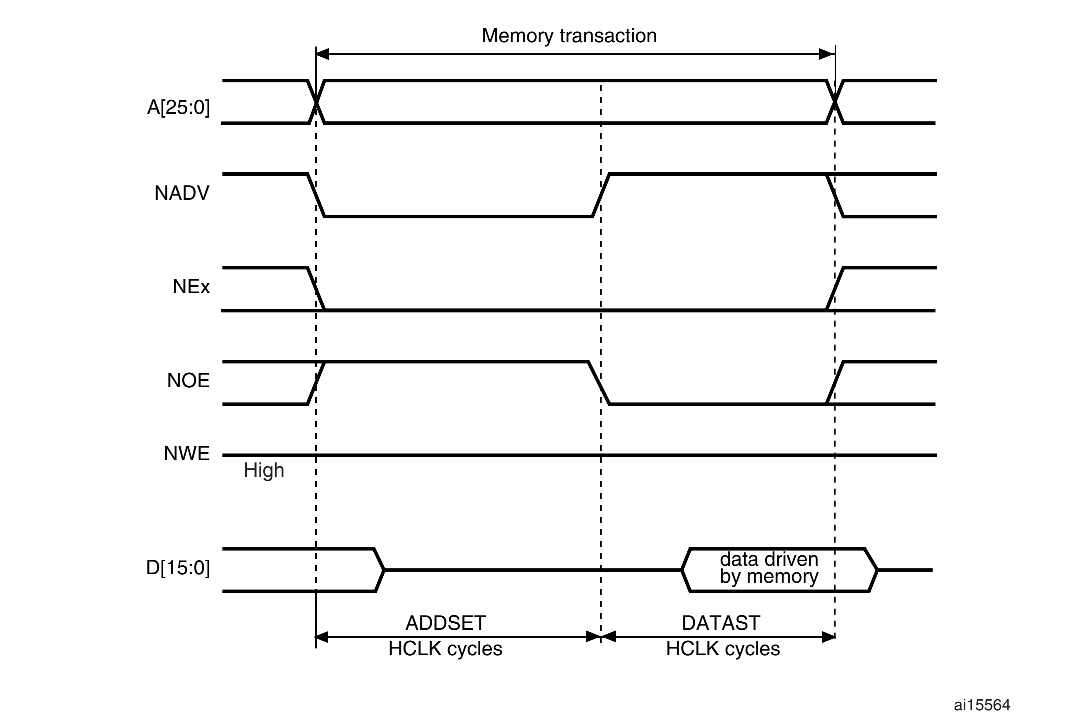
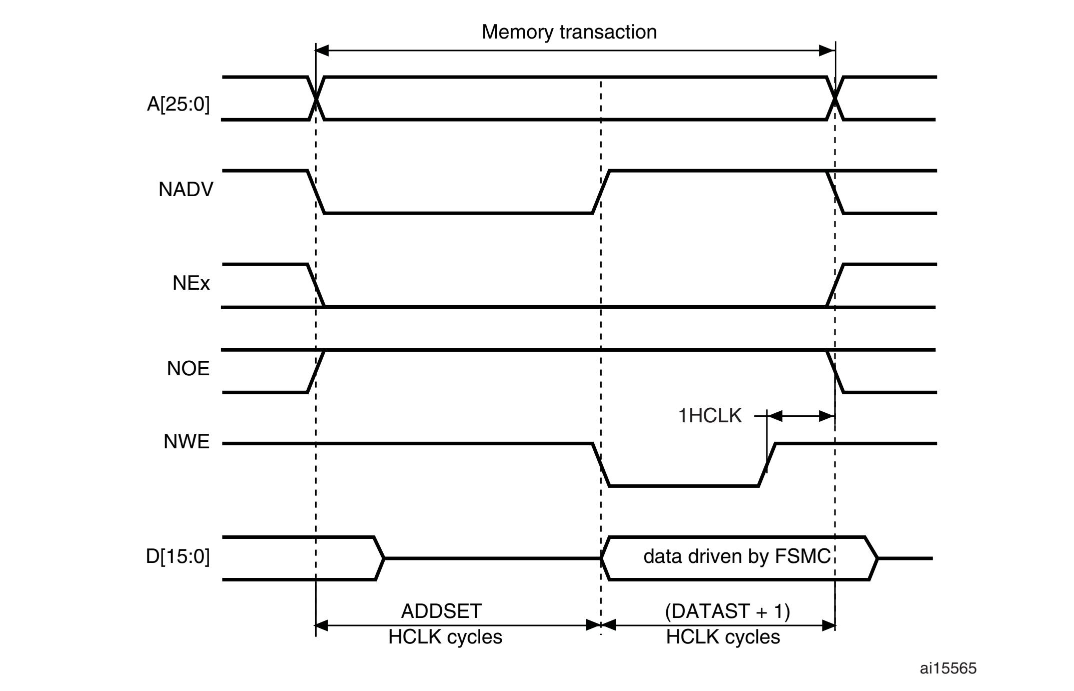


Figure 196. Mode C write accesses



The differences compared with mode1 are the toggling of NOE and the independent read and write timings.

Table 111. FSMC_BCRx bit fields

Bit No.	Bit name	Value to set
31-20	Reserved	0x000
19	CBURSTRW	0x0 (no effect on asynchronous mode)
18:16	Reserved	0x0
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect on asynchronous mode)
12	WREN	0x1
11	WAITCFG	As needed
10	WRAPMOD	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1
5-4	MWID	As needed
3-2	MTYP	0x2 (NOR Flash memory)

Table 111. FSMC_BCRx bit fields (continued)

Bit No.	Bit name	Value to set
1	MUXEN	0x0
0	MBKEN	0x1

Table 112. FSMC_BTRx bit fields

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29-28	ACCMOD	0x2
27-24	DATLAT	0x0
23-20	CLKDIV	0x0
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST HCLK cycles) for read accesses.
7-4	ADDHLD	Don’t care
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for read accesses. Minimum value for ADDSET is 0.

Table 113. FSMC_BWTRx bit fields

Bit number	Bit name	Value to set
31:30	Reserved	0x0
29-28	ACCMOD	0x2
27-24	DATLAT	Don’t care
23-20	CLKDIV	Don’t care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST+1 HCLK cycles for write accesses,
7-4	ADDHLD	Don’t care
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for write accesses. Minimum value for ADDSET is 0.

Mode D - asynchronous access with extended address

Figure 197. Mode D read accesses

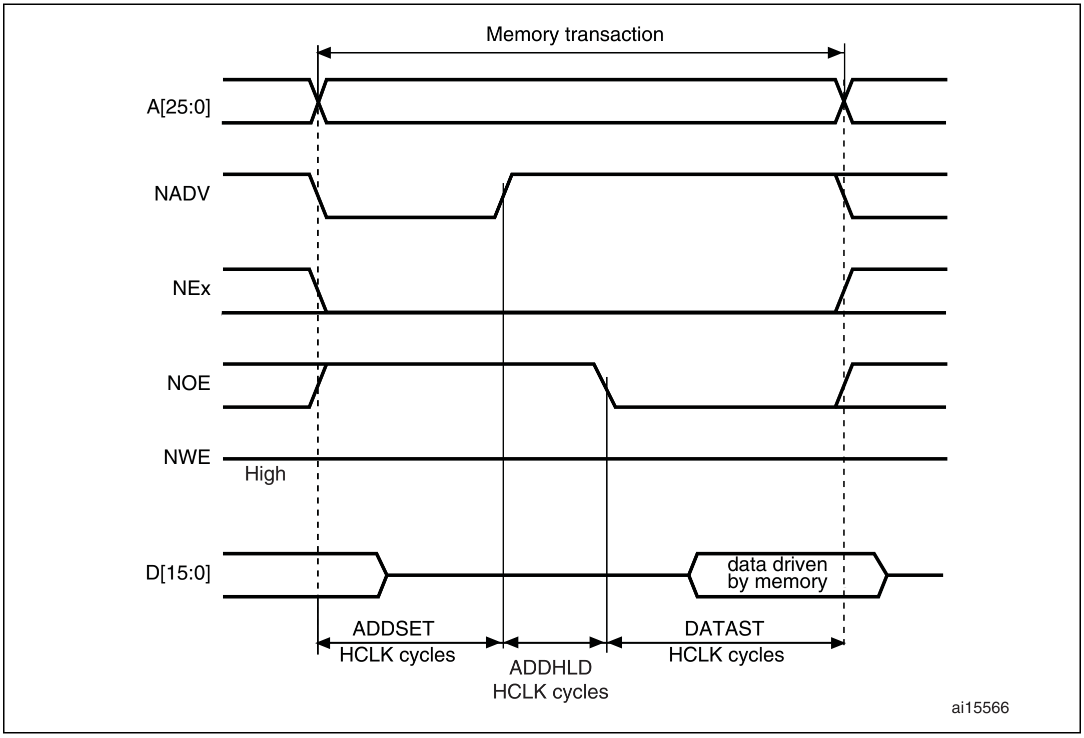
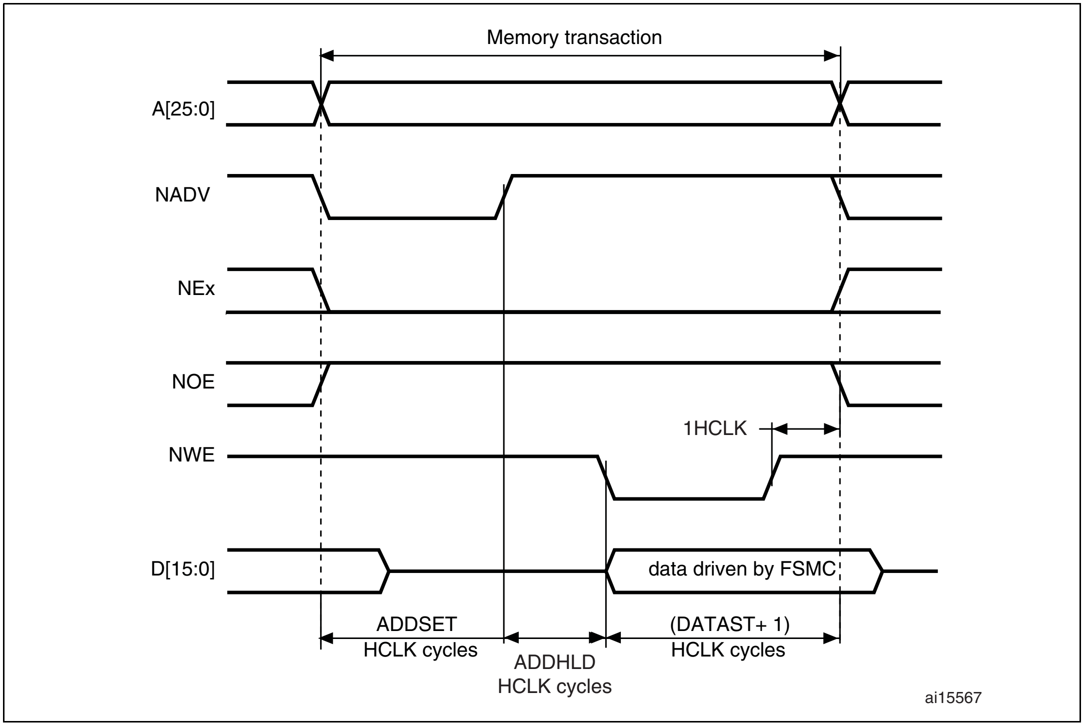


Figure 198. Mode D write accesses



The differences with mode1 are the toggling of NOE that goes on toggling after NADV changes and the independent read and write timings.

Table 114. FSMC_BCRx bit fields

Bit No.	Bit name	Value to set
31-20	Reserved	0x000
19	CBURSTRW	0x0 (no effect on asynchronous mode)
18:16	Reserved	0x0
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect on asynchronous mode)
12	WREN	0x1
11	WAITCFG	As needed
10	WRAPMOD	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	Set according to memory support
5-4	MWID	As needed
3-2	MTYP	As needed
1	MUXEN	0x0
0	MBKEN	0x1

Table 115. FSMC_BTRx bit fields

Bit No.	Bit name	Value to set
31:30	Reserved	0x0
29-28	ACCMOD	0x3
27-24	DATLAT	Don’t care
23-20	CLKDIV	Don’t care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST HCLK cycles) for read accesses.
7-4	ADDHLD	Duration of the middle phase of the read access (ADDHLD HCLK cycles)
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for read accesses. Minimum value for ADDSET is 1.

Table 116. FSMC_BWTRx bit fields

Bit No.	Bit name	Value to set
31:30	Reserved	0x0
29-28	ACCMOD	0x3
27-24	DATLAT	0x0
23-20	CLKDIV	0x0
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase
7-4	ADDHLD	Duration of the middle phase of the write access (ADDHLD HCLK cycles)
3-0	ADDSET	Duration of the first access phase . Minimum value for ADDSET is 1.

Muxed mode - multiplexed asynchronous access to NOR Flash memory

Figure 199. Multiplexed read accesses

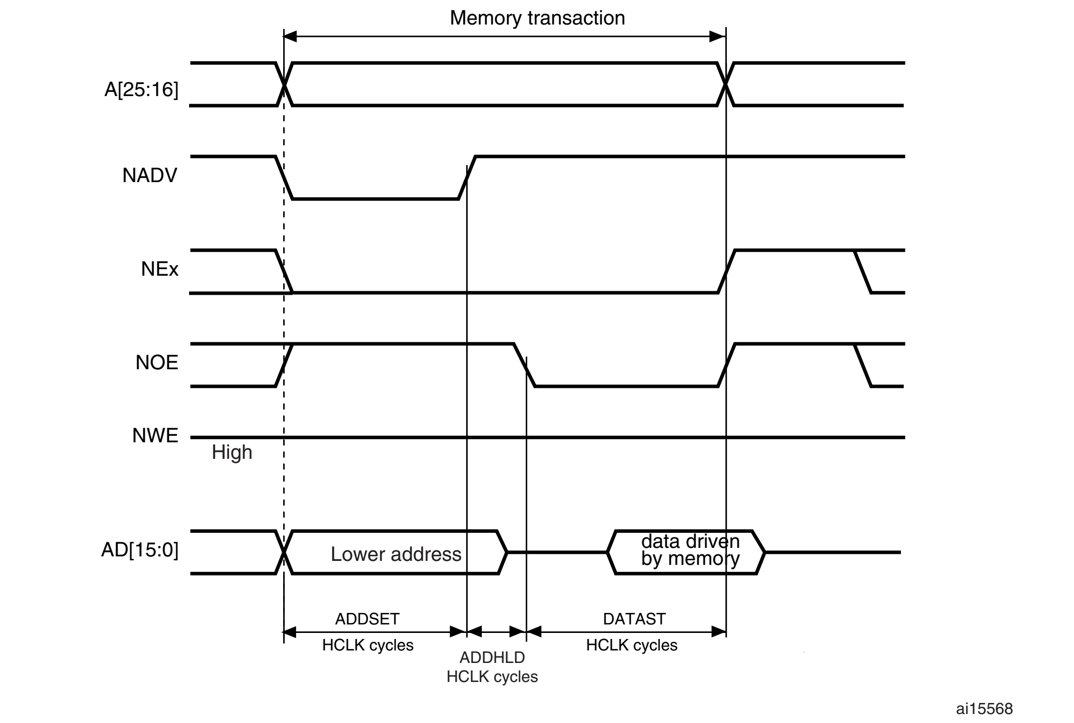
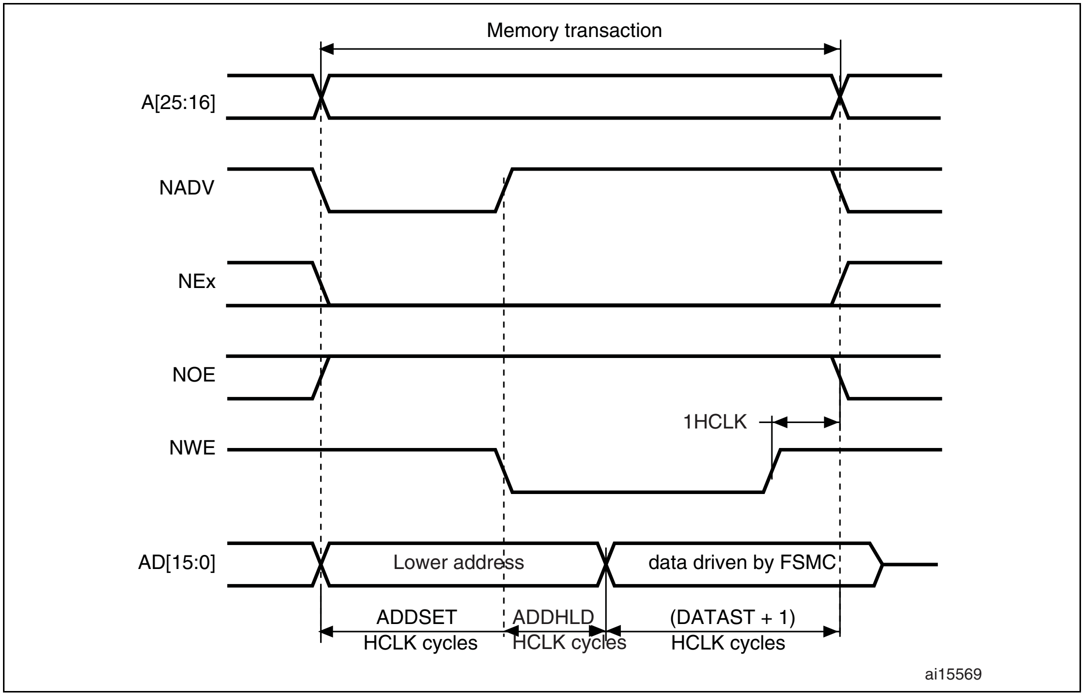


Figure 200. Multiplexed write accesses



The difference with mode D is the drive of the lower address byte(s) on the databus.

Table 117. FSMC_BCRx bit fields

Bit No.	Bit name	Value to set
31-21	Reserved	0x000
19	CBURSTRW	0x0 (no effect on asynchronous mode)
18:16	Reserved	0x0
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x0
13	WAITEN	0x0 (no effect on asynchronous mode)
12	WREN	0x1
11	WAITCFG	As needed
10	WRAPMOD	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1
5-4	MWID	As needed
3-2	MTYP	0x2 (NOR Flash memory)

Table 117. FSMC_BCRx bit fields (continued)

Bit No.	Bit name	Value to set
1	MUXEN	0x1
0	MBKEN	0x1

Table 118. FSMC_BTRx bit fields

Bit No.	Bit name	Value to set
31:30	Reserved	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	Don’t care
23-20	CLKDIV	Don’t care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST HCLK cycles for read accesses and DATAST+1 HCLK cycles for write accesses).
7-4	ADDHLD	Duration of the middle phase of the access (ADDHLD HCLK cycles).
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles). Minimum value for ADDSET is 1.

WAIT management in asynchronous accesses

If the asynchronous memory asserts a WAIT signal to indicate that it is not yet ready to accept or to provide data, the ASYNCWAIT bit has to be set in FSMC_BCRx register.

If the WAIT signal is active (high or low depending on the WAITPOL bit), the second access phase (Data setup phase) programmed by the DATAST bits, is extended until WAIT becomes inactive. Unlike the data setup phase, the first access phases (Address setup and Address hold phases), programmed by the ADDSET and ADDHLD bits, are not WAIT sensitive and so they are not prolonged.

The data setup phase (DATAST in the FSMC_BTRx register) must be programmed so that WAIT can be detected 4 HCLK cycles before the end of memory transaction. The following cases must be considered:

1. Memory asserts the WAIT signal aligned to NOE/NWE which toggles:

$$\text{DATAST} \geq (4 \times \text{HCLK}) + \text{max\_wait\_assertion\_time}$$

2. Memory asserts the WAIT signal aligned to NEx (or NOE/NWE not toggling):
   if

$$\text{max\_wait\_assertion\_time} > \text{address\_phase} + \text{hold\_phase}$$

   then

$$\text{DATAST} \geq (4 \times \text{HCLK}) + (\text{max\_wait\_assertion\_time} - \text{address\_phase} - \text{hold\_phase})$$

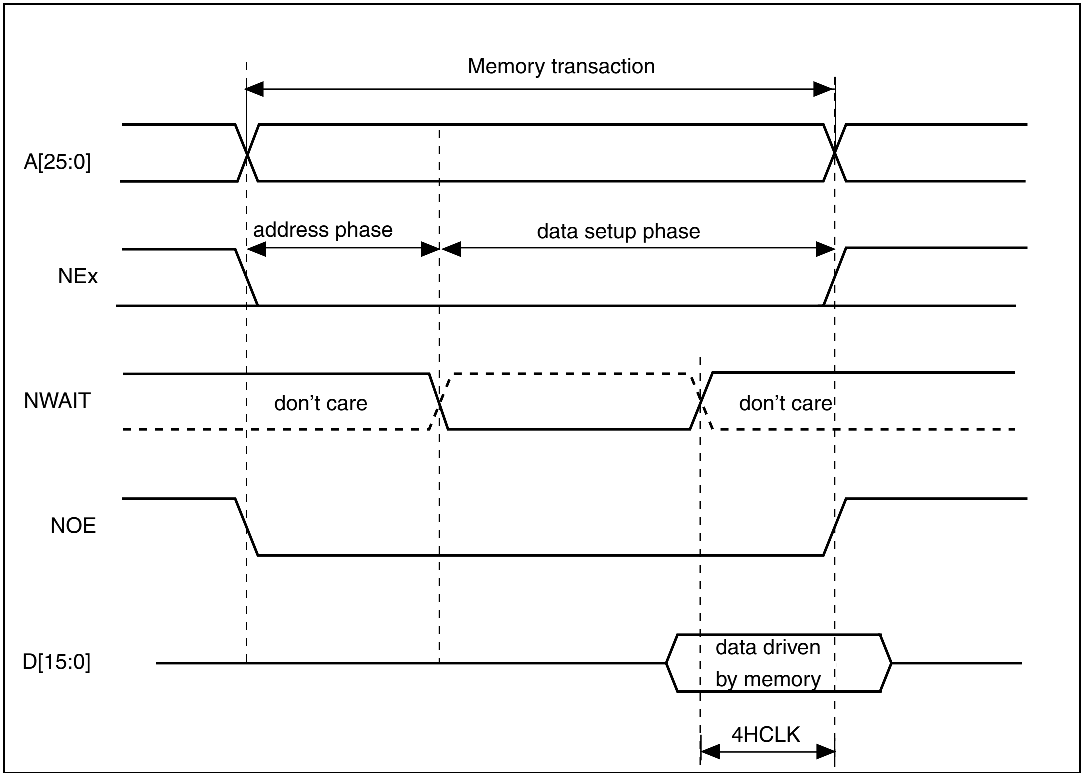
   otherwise

$$\text{DATAST} \geq 4 \times \text{HCLK}$$

   where max_wait_assertion_time is the maximum time taken by the memory to assert the WAIT signal once NEx/NOE/NWE is low.

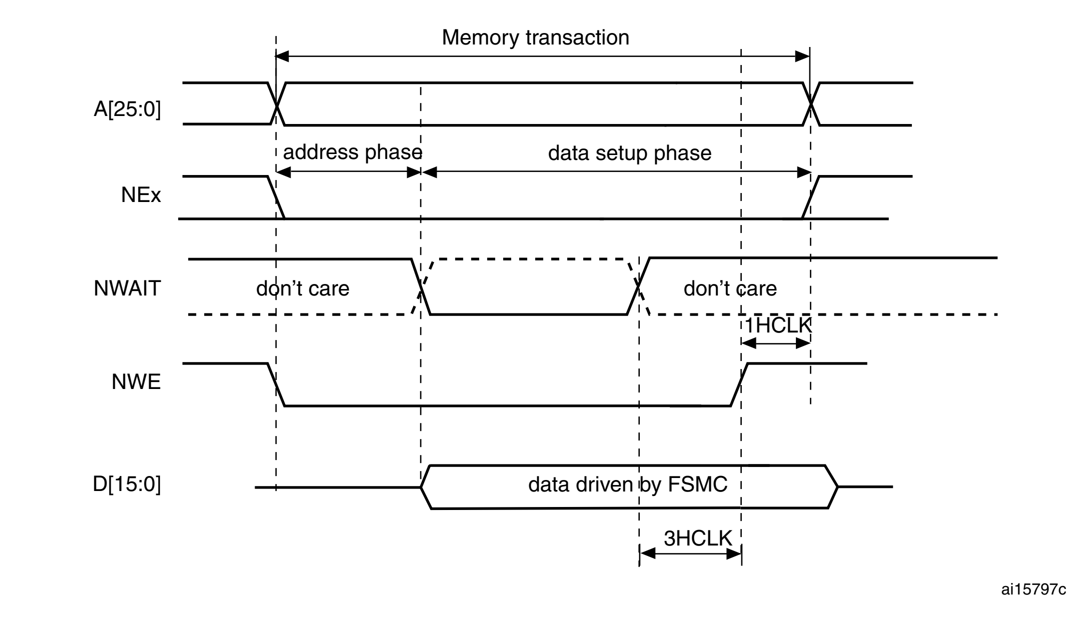
Figure 201 and *Figure 202* show the number of HCLK clock cycles that are added to the memory access after WAIT is released by the asynchronous memory (independently of the above cases).

Figure 201. Asynchronous wait during a read access



1. NWAIT polarity depends on WAITPOL bit setting in FSMC_BCRx register.

Figure 202. Asynchronous wait during a write access



1. NWAIT polarity depends on WAITPOL bit setting in FSMC_BCRx register.

24.5.5 Synchronous burst transactions

The memory clock, CLK, is a submultiple of HCLK according to the value of parameter CLKDIV.

NOR Flash memories specify a minimum time from NADV assertion to CLK high. To meet this constraint, the FSMC does not issue the clock to the memory during the first internal clock cycle of the synchronous access (before NADV assertion). This guarantees that the rising edge of the memory clock occurs *in the middle* of the NADV low pulse.

Data latency versus NOR Flash latency

The data latency is the number of cycles to wait before sampling the data. The DATLAT value must be consistent with the latency value specified in the NOR Flash configuration register. The FSMC does not include the clock cycle when NADV is low in the data latency count.

Caution: Some NOR Flash memories include the NADV Low cycle in the data latency count, so the exact relation between the NOR Flash latency and the FMSC DATLAT parameter can be either of:

- NOR Flash latency = DATLAT + 2
- NOR Flash latency = DATLAT + 3

Some recent memories assert NWAIT during the latency phase. In such cases DATLAT can be set to its minimum value. As a result, the FSMC samples the data and waits long enough to evaluate if the data are valid. Thus the FSMC detects when the memory exits latency and real data are taken.

Other memories do not assert NWAIT during latency. In this case the latency must be set correctly for both the FSMC and the memory, otherwise invalid data are mistaken for good data, or valid data are lost in the initial phase of the memory access.

Single-burst transfer

When the selected bank is configured in synchronous burst mode, if an AHB single-burst transaction is requested, the FSMC performs a burst transaction of length 1 (if the AHB transfer is 16-bit), or length 2 (if the AHB transfer is 32-bit) and de-assert the chip select signal when the last data is strobed.

Clearly, such a transfer is not the most efficient in terms of cycles (compared to an asynchronous read). Nevertheless, a random asynchronous access would first require to re-program the memory access mode, which would altogether last longer.

Wait management

For synchronous burst NOR Flash, NWAIT is evaluated after the programmed latency period, (DATALAT+2) CLK clock cycles.

If NWAIT is sensed active (low level when WAITPOL = 0, high level when WAITPOL = 1), wait states are inserted until NWAIT is sensed inactive (high level when WAITPOL = 0, low level when WAITPOL = 1).

When NWAIT is inactive, the data is considered valid either immediately (bit WAITCFG = 1) or on the next clock edge (bit WAITCFG = 0).

During wait-state insertion via the NWAIT signal, the controller continues to send clock pulses to the memory, keeping the chip select and output enable signals valid, and does not consider the data valid.

There are two timing configurations for the NOR Flash NWAIT signal in burst mode:

- Flash memory asserts the NWAIT signal one data cycle before the wait state (default after reset)
- Flash memory asserts the NWAIT signal during the wait state

These two NOR Flash wait state configurations are supported by the FSMC, individually for each chip select, thanks to the WAITCFG bit in the FSMC_BCRx registers (x = 0..3).

Figure 203. Wait configurations

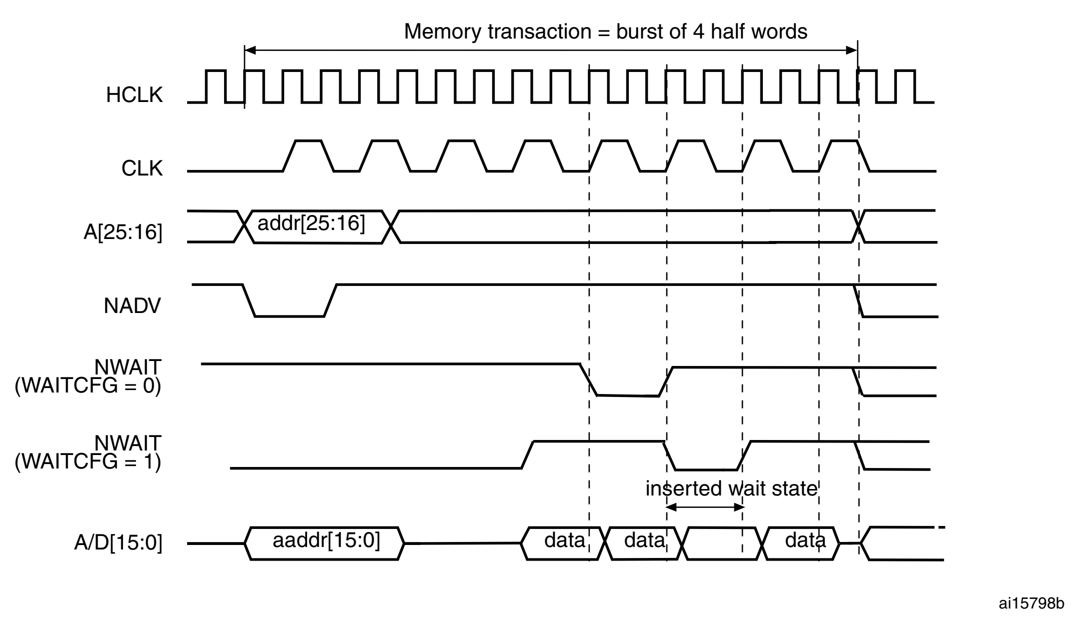
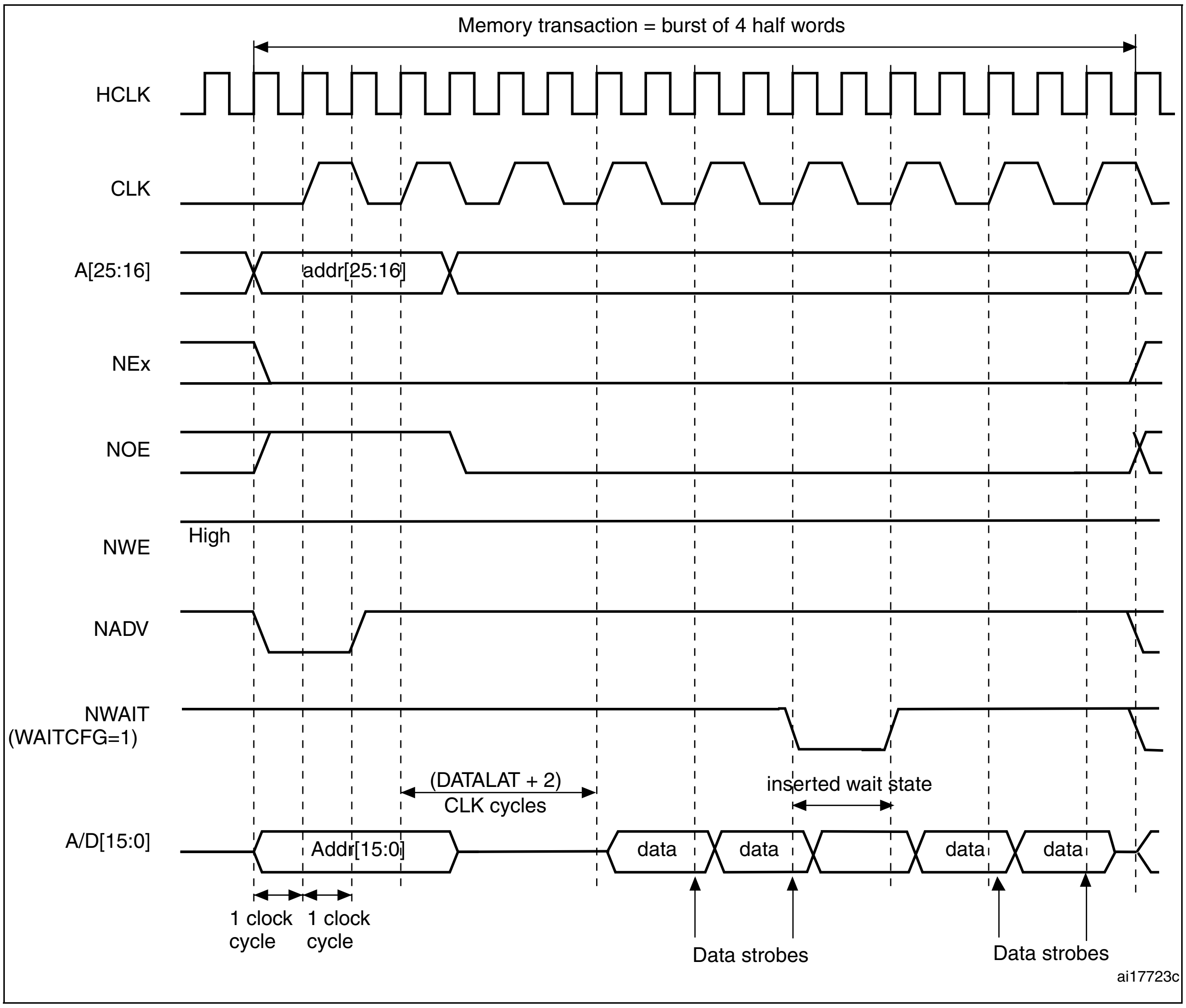


Figure 204. Synchronous multiplexed read mode - NOR, PSRAM (CRAM)



1. Byte lane outputs BL are not shown; for NOR access, they are held high, and, for PSRAM (CRAM) access, they are held low.
2. NWAIT polarity is set to 0.

Table 119. FSMC_BCRx bit fields

Bit No.	Bit name	Value to set
31-20	Reserved	0x000
19	CBURSTRW	No effect on synchronous read
18-16	Reserved	0x0
15	ASCYCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	to be set to 1 if the memory supports this feature, to be kept at 0 otherwise.
12	WREN	no effect on synchronous read
11	WAITCFG	to be set according to memory

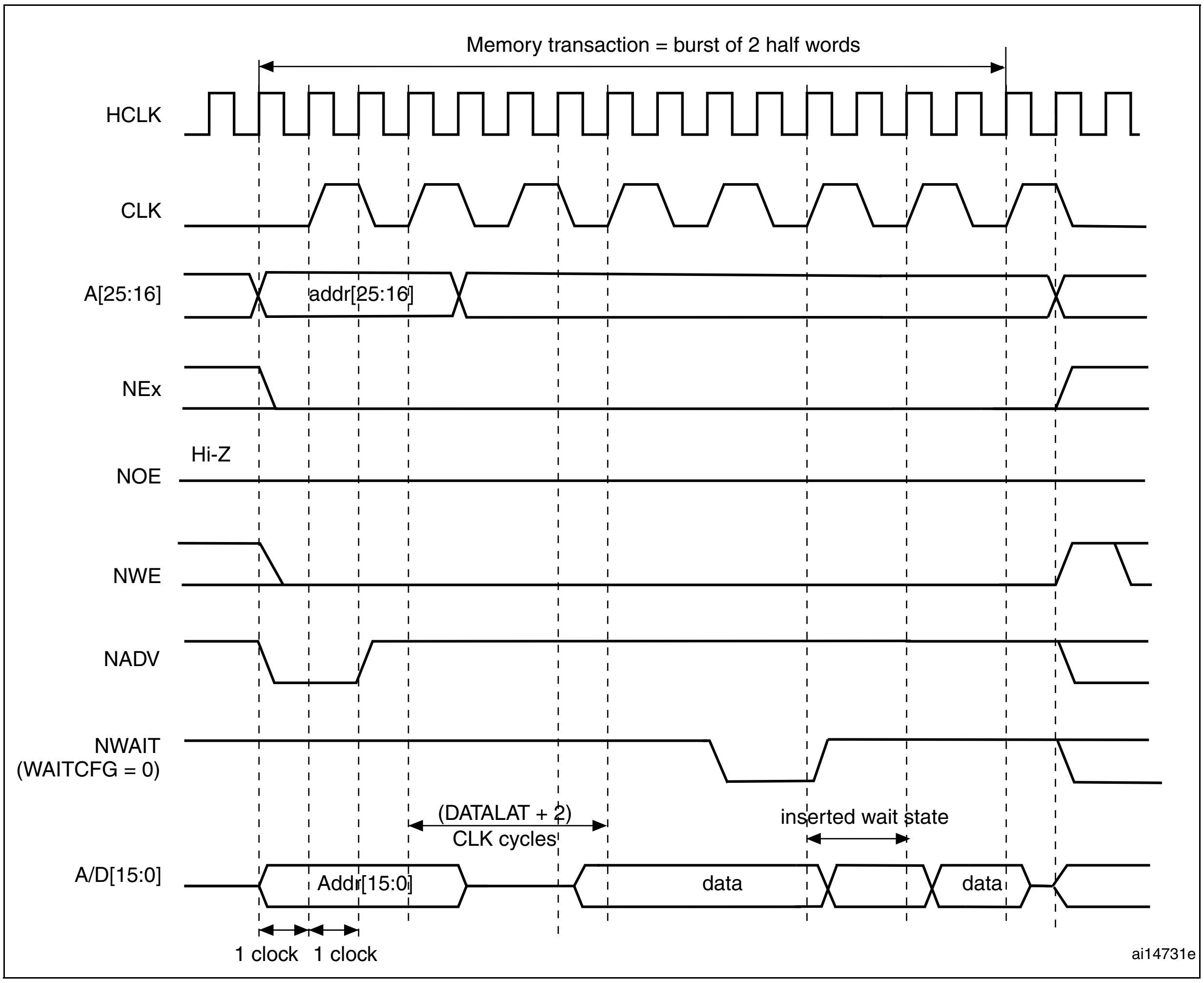
Table 119. FSMC_BCRx bit fields (continued)

Bit No.	Bit name	Value to set
10	WRAPMOD	0x0
9	WAITPOL	to be set according to memory
8	BURSTEN	0x1
7	Reserved	0x1
6	FACCEN	Set according to memory support (NOR Flash memory)
5-4	MWID	As needed
3-2	MTYP	0x1 or 0x2
1	MUXEN	As needed
0	MBKEN	0x1

Table 120. FSMC_BTRx bit fields

Bit No.	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x0
27-24	DATLAT	Data latency
23-20	CLKDIV	0x0 to get CLK = HCLK (not supported) 0x1 to get CLK = 2 × HCLK ..
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Don’t care
7-4	ADDHLD	Don’t care
3-0	ADDSET	Don’t care

Figure 205. Synchronous multiplexed write mode - PSRAM (CRAM)



1. Memory must issue NWAIT signal one cycle in advance, accordingly WAITCFG must be programmed to 0.
2. NWAIT polarity is set to 0.
3. Byte Lane (NBL) outputs are not shown, they are held low while NEx is active.

Table 121. FSMC_BCRx bit fields

Bit No.	Bit name	Value to set
31-20	Reserved	0x000
19	CBURSTRW	0x1
18-16	Reserved	0x0
15	ASCYCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	to be set to 1 if the memory supports this feature, to be kept at 0 otherwise.
12	WREN	no effect on synchronous read
11	WAITCFG	0x0

Table 121. FSMC_BCRx bit fields (continued)

Bit No.	Bit name	Value to set
10	WRAPMOD	0x0
9	WAITPOL	to be set according to memory
8	BURSTEN	no effect on synchronous write
7	Reserved	0x1
6	FACCEN	Set according to memory support
5-4	MWID	As needed
3-2	MTYP	0x1
1	MUXEN	As needed
0	MBKEN	0x1

Table 122. FSMC_BTRx bit fields

Bit No.	Bit name	Value to set
31:30	Reserved	0x0
29:28	ACCMOD	0x0
27-24	DATLAT	Data latency
23-20	CLKDIV	0x0 to get CLK = HCLK (not supported) 0x1 to get CLK = 2 × HCLK
19-16	BUSTURN	Don’t care
15-8	DATAST	Don’t care
7-4	ADDHLD	Don’t care
3-0	ADDSET	Don’t care

24.5.6 NOR/PSRAM control registers

The NOR/PSRAM control registers have to be accessed by words (32 bits).

SRAM/NOR-Flash chip-select control registers 1..4 (FSMC_BCR1..4)

Address offset: 0xA000 0000 + 8 * (x – 1), x = 1...4

Reset value: 0x0000 30DX

This register contains the control information of each memory bank, used for SRAMs, ROMs and asynchronous or burst NOR Flash memories.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CBURSTRW	Reserved			ASCYCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
												rw				rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

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

Bit 19 **CBURSTRW:** Write burst enable.

For Cellular RAM (PSRAM), the bit enables the synchronous burst protocol during write operations. The enable bit for the synchronous burst protocol during read access is the BURSTEN bit in the FSMC_BCRx register.

0: Write operations are always performed in asynchronous mode

1: Write operations are performed in synchronous mode.

Bits 18: 16 Reserved, must be kept at reset value.

Bit 15 **ASYNCWAIT**: Wait signal during asynchronous transfers

This bit enables/disables the FSMC to use the wait signal even during an asynchronous protocol.

0: NWAIT signal is not taken in to account when running an asynchronous protocol (default after reset)

1: NWAIT signal is taken in to account when running an asynchronous protocol

Bit 14 **EXTMOD:** Extended mode enable.

This bit enables the FSMC to program the write timings inside the FSMC_BWTR register, thus resulting in different timings for read and write operations.

0: values inside FSMC_BWTR register are not taken into account (default after reset)

1: values inside FSMC_BWTR register are taken into account

Bit 13 **WAITEN:** Wait enable bit.

This bit enables/disables wait-state insertion via the NWAIT signal when accessing the Flash memory in synchronous mode.

0: NWAIT signal is disabled (its level not taken into account, no wait state inserted after the programmed Flash latency period)

1: NWAIT signal is enabled (its level is taken into account after the programmed Flash latency period to insert wait states if asserted) (default after reset)

Bit 12 **WREN:** Write enable bit.

This bit indicates whether write operations are enabled/disabled in the bank by the FSMC:

0: Write operations are disabled in the bank by the FSMC, an AHB error is reported,

1: Write operations are enabled for the bank by the FSMC (default after reset).

Bit 11 **WAITCFG:** Wait timing configuration.

The NWAIT signal indicates whether the data from the memory are valid or if a wait state must be inserted when accessing the Flash memory in synchronous mode. This configuration bit determines if NWAIT is asserted by the memory one clock cycle before the wait state or during the wait state:

0: NWAIT signal is active one data cycle before wait state (default after reset),

1: NWAIT signal is active during wait state (not for Cellular RAM).

Bit 10 **WRAPMOD:** Wrapped burst mode support.

Defines whether the controller will or not split an AHB burst wrap access into two linear accesses. Valid only when accessing memories in burst mode

0: Direct wrapped burst is not enabled (default after reset),

1: Direct wrapped burst is enabled.

Note: This bit has no effect as the CPU and DMA cannot generate wrapping burst transfers.

Bit 9 **WAITPOL:** Wait signal polarity bit.

Defines the polarity of the wait signal from memory. Valid only when accessing the memory in burst mode:

0: NWAIT active low (default after reset),

1: NWAIT active high.

Bit 8 **BURSTEN:** Burst enable bit.

This bit enables/disables the synchronous burst access during read operations. It is valid only with synchronous burst memories:

0: Burst access mode disabled (default after reset)

1: Burst access mode enable

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FACCEN:** Flash access enable

Enables NOR Flash memory access operations.

0: Corresponding NOR Flash memory access is disabled

1: Corresponding NOR Flash memory access is enabled (default after reset)

Bits 5:4 **MWID:** Memory databus width.

Defines the external memory device width, valid for all type of memories.

00: 8 bits,

01: 16 bits (default after reset),

10: reserved, do not use,

11: reserved, do not use.

Bits 3:2 **MTYP:** Memory type.

Defines the type of external memory attached to the corresponding memory bank:
00: SRAM, ROM (default after reset for Bank 2...4)
01: PSRAM (Cellular RAM: CRAM)
10: NOR Flash/OneNAND Flash (default after reset for Bank 1)
11: reserved

Bit 1 **MUXEN:** Address/data multiplexing enable bit.

When this bit is set, the address and data values are multiplexed on the databus, valid only with NOR and PSRAM memories:
0: Address/Data nonmultiplexed
1: Address/Data multiplexed on databus (default after reset)

Bit 0 **MBKEN:** Memory bank enable bit.

Enables the memory bank. After reset Bank1 is enabled, all others are disabled. Accessing a disabled bank causes an ERROR on AHB bus.
0: Corresponding memory bank is disabled
1: Corresponding memory bank is enabled

SRAM/NOR-Flash chip-select timing registers 1..4 (FSMC_BTR1..4)

Address offset: 0xA000 0000 + 0x04 + 8 * (x – 1), x = 1..4

Reset value: 0x0FFF FFFF

This register contains the control information of each memory bank, used for SRAMs, ROMs and NOR Flash memories. If the EXTMOD bit is set in the FSMC_BCRx register, then this register is partitioned for write and read access, that is, 2 registers are available: one to configure read accesses (this register) and one to configure write accesses (FSMC_BWTRx registers).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		ACCMOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 29:28 **ACCMOD:** Access mode

Specifies the asynchronous access modes as shown in the timing diagrams. These bits are taken into account only when the EXTMOD bit in the FSMC_BCRx register is 1.

00: access mode A

01: access mode B

10: access mode C

11: access mode D

Bits 27:24 **DATLAT**: Data latency for synchronous burst NOR Flash memory

For NOR Flash with synchronous burst mode enabled, defines the number of memory clock cycles (+2) to issue to the memory before getting the first data:

0000: Data latency of 2 CLK clock cycles for first burst access

1111: Data latency of 17 CLK clock cycles for first burst access (default value after reset)

Note: This timing parameter is not expressed in HCLK periods, but in Flash clock (CLK) periods. In asynchronous NOR Flash, SRAM or ROM accesses, this value is don't care. In the case of CRAM, this field must be set to '0'.

Bits 23:20 **CLKDIV:** Clock divide ratio (for CLK signal)

Defines the period of CLK clock output signal, expressed in number of HCLK cycles:

0000: Reserved

0001: CLK period = 2 × HCLK periods

0010: CLK period = 3 × HCLK periods

1111: CLK period = 16 × HCLK periods (default value after reset)

In asynchronous NOR Flash, SRAM or ROM accesses, this value is don't care.

Bits 19:16 **BUSTURN:** Bus turnaround phase duration

These bits are written by software to add a delay at the end of a write/read transaction. This delay allows to match the minimum time between consecutive transactions (t_{EHEL} from NEx high to NEx low) and the maximum time needed by the memory to free the data bus after a read access (tEHQZ):

(BUSTRUN + 1)HCLK period ≥ $t_{EHELmin}$ and (BUSTRUN + 2)HCLK period ≥ $t_{EHQZmax}$ if EXTMOD = ‘0’

(BUSTRUN + 2)HCLK period ≥ max ($t_{EHELmin}$, $t_{EHQZmax}$) if EXTMOD = ‘1’.

0000: BUSTURN phase duration = 0 HCLK clock cycle added

...

1111: BUSTURN phase duration = 15 × HCLK clock cycles (default value after reset)

Bits 15:8 **DATAST:** Data-phase duration

These bits are written by software to define the duration of the data phase (refer to *Figure 188* to *Figure 200*), used in SRAMs, ROMs and asynchronous NOR Flash accesses:

0000 0000: Reserved
0000 0001: DATAST phase duration = 1 × HCLK clock cycles
0000 0010: DATAST phase duration = 2 × HCLK clock cycles
...
1111 1111: DATAST phase duration = 255 × HCLK clock cycles (default value after reset)

For each memory type and access mode data-phase duration, please refer to the respective figure (*Figure 188* to *Figure 200*).

Example: Mode1, write access, DATAST=1: Data-phase duration= DATAST+1 = 2 HCLK clock cycles.

Note: *In synchronous accesses, this value is don't care.*

Bits 7:4 **ADDHLD:** Address-hold phase duration

These bits are written by software to define the duration of the *address hold* phase (refer to *Figure 197* to *Figure 200*), used in mode D and multiplexed accesses:

0000: Reserved
0001: ADDHLD phase duration =1 × HCLK clock cycle
0010: ADDHLD phase duration = 2 × HCLK clock cycle
...
1111: ADDHLD phase duration = 15 × HCLK clock cycles (default value after reset)

For each access mode address-hold phase duration, please refer to the respective figure (*Figure 197* to *Figure 200*).

Note: *In synchronous accesses, this value is not used, the address hold phase is always 1 memory clock period duration.*

Bits 3:0 **ADDSET:** Address setup phase duration

These bits are written by software to define the duration of the *address setup* phase (refer to *Figure 188* to *Figure 200*), used in SRAMs, ROMs and asynchronous NOR Flash accesses:

0000: ADDSET phase duration = 0 × HCLK clock cycle
...
1111: ADDSET phase duration = 1615 × HCLK clock cycles (default value after reset)

For each access mode address setup phase duration, please refer to the respective figure (refer to *Figure 188* to *Figure 200*).

Note: *In synchronous accesses, this value is don’t care.*

Note: *PSRAMs (CRAMs) have a variable latency due to internal refresh. Therefore these memories issue the NWAIT signal during the whole latency phase to prolong the latency as needed.*
With PSRAMs (CRAMs) the filed DATLAT must be set to 0, so that the FSMC exits its latency phase soon and starts sampling NWAIT from memory, then starts to read or write

when the memory is ready.
This method can be used also with the latest generation of synchronous Flash memories that issue the NWAIT signal, unlike older Flash memories (check the datasheet of the specific Flash memory being used).

SRAM/NOR-Flash write timing registers 1..4 (FSMC_BWTR1..4)

Address offset: 0xA000 0000 + 0x104 + 8 * (x – 1), x = 1...4

Reset value: 0x0FFF FFFF

This register contains the control information of each memory bank, used for SRAMs, ROMs and NOR Flash memories. When the EXTMOD bit is set in the FSMC_BCRx register, then this register is active for write access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		ACCMOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 29:28 **ACCMOD:** Access mode.

Specifies the asynchronous access modes as shown in the next timing diagrams.These bits are taken into account only when the EXTMOD bit in the FSMC_BCRx register is 1.
00: access mode A
01: access mode B
10: access mode C
11: access mode D

Bits 27:24 **DATLAT:** Data latency (for synchronous burst NOR Flash).

For NOR Flash with Synchronous burst mode enabled, defines the number of memory clock cycles (+2) to issue to the memory before getting the first data:
0000: (0x0) Data latency of 2 CLK clock cycles for first burst access
...
1111: (0xF) Data latency of 17 CLK clock cycles for first burst access (default value after reset)

*Note: This timing parameter is not expressed in HCLK periods, but in Flash clock (**CLK**) periods. In asynchronous NOR Flash, SRAM or ROM accesses, this value is don't care. In case of CRAM, this field must be set to 0*

Bits 23:20 **CLKDIV:** Clock divide ratio (for CLK signal).

Defines the period of CLK clock output signal, expressed in number of HCLK cycles:
0000: Reserved
0001 CLK period = 2 × HCLK periods
0010 CLK period = 3 × HCLK periods
1111: CLK period = 16 × HCLK periods (default value after reset)
In asynchronous NOR Flash, SRAM or ROM accesses, this value is don't care.

Bits 19:16 **BUSTURN**: Bus turnaround phase duration

These bits are written by software to add a delay at the end of a write transaction to match the minimum time between consecutive transactions (t_{EHEL} from ENx high to ENx low):
(BUSTRUN + 1) HCLK period ≥ $t_{EHELmin}$.
0000: BUSTURN phase duration = 0 HCLK clock cycle added
...
1111: BUSTURN phase duration = 15 HCLK clock cycles added (default value after reset)

Bits 15:8 **DATAST:** Data-phase duration.

These bits are written by software to define the duration of the data phase (refer to *Figure 188* to *Figure 200*), used in SRAMs, ROMs and asynchronous NOR Flash accesses:

0000 0000: Reserved

0000 0001: DATAST phase duration = 1 × HCLK clock cycles

0000 0010: DATAST phase duration = 2 × HCLK clock cycles

...

1111 1111: DATAST phase duration = 255 × HCLK clock cycles (default value after reset)

Note: In synchronous accesses, this value is don't care.

Bits 7:4 **ADDHLD:** Address-hold phase duration.

These bits are written by software to define the duration of the *address hold* phase (refer to *Figure 197* to *Figure 200*), used in SRAMs, ROMs and asynchronous multiplexed NOR Flash accesses:

0000: Reserved

0001: ADDHLD phase duration = 1 × HCLK clock cycle

0010: ADDHLD phase duration = 2 × HCLK clock cycle

...

1111: ADDHLD phase duration = 15 × HCLK clock cycles (default value after reset)

Note: In synchronous NOR Flash accesses, this value is not used, the address hold phase is always 1 Flash clock period duration.

Bits 3:0 **ADDSET:** Address setup phase duration.

These bits are written by software to define the duration of the *address setup* phase in HCLK cycles (refer to *Figure 197* to *Figure 200*), used in SRAMs, ROMs and asynchronous NOR Flash accessed:

0000: ADDSET phase duration = 0 × HCLK clock cycle

...

1111: ADDSET phase duration = 15 × HCLK clock cycles (default value after reset)

Note: In synchronous NOR Flash accesses, this value is don’t care.

24.5.7 FSMC register map

The following table summarizes the FSMC registers.

Table 123. FSMC register map

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
0000	FSMC_BCR1	Reserved												CBURSTRW	Reserved			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
0008	FSMC_BCR2	Reserved												CBURSTRW	Reserved			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
0010	FSMC_BCR3	Reserved												CBURSTRW	Reserved			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
0018	FSMC_BCR4	Reserved												CBURSTRW	Reserved			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
0004	FSMC_BTR1	Res.		ACC MOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
000C	FSMC_BTR2	Res.		ACC MOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
0014	FSMC_BTR3	Res.		ACC MOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
001C	FSMC_BTR4	Res.		ACC MOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
0104	FSMC_BWTR1	Res.		ACC MOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
010C	FSMC_BWTR2	Res.		ACC MOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
0114	FSMC_BWTR3	Res.		ACC MOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
011C	FSMC_BWTR4	Res.		ACC MOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			

Refer to *Table 2 on page 42* for the register boundary addresses.

25 Inter-integrated circuit (I^{2}C) interface

25.1 I^{2}C introduction

I^{2}C (inter-integrated circuit) bus Interface serves as an interface between the microcontroller and the serial I^{2}C bus. It provides multimaster capability, and controls all I^{2}C bus-specific sequencing, protocol, arbitration and timing. It supports standard and fast speed modes. It is also SMBus 2.0 compatible.

It may be used for a variety of purposes, including CRC generation and verification, SMBus (system management bus) and PMBus (power management bus).

Depending on specific device implementation DMA capability can be available for reduced CPU overload.

25.2 I^{2}C main features

- Parallel-bus/I^{2}C protocol converter
- Multimaster capability: the same interface can act as Master or Slave
- I^{2}C Master features:
  - Clock generation
  - Start and Stop generation
- I^{2}C Slave features:
  - Programmable I^{2}C Address detection
  - Dual Addressing Capability to acknowledge 2 slave addresses
  - Stop bit detection
- Generation and detection of 7-bit/10-bit addressing and General Call
- Supports different communication speeds:
  - Standard Speed (up to 100 kHz)
  - Fast Speed (up to 400 kHz)
- Status flags:
  - Transmitter/Receiver mode flag
  - End-of-Byte transmission flag
  - I^{2}C busy flag
- Error flags:
  - Arbitration lost condition for master mode
  - Acknowledgement failure after address/ data transmission
  - Detection of misplaced start or stop condition
  - Overrun/Underrun if clock stretching is disabled
- 2 Interrupt vectors:
  - 1 Interrupt for successful address/ data communication
  - 1 Interrupt for error condition
- Optional clock stretching
- 1-byte buffer with DMA capability
- Configurable PEC (packet error checking) generation or verification:
  - PEC value can be transmitted as last byte in Tx mode
  - PEC error checking for last received byte
- SMBus 2.0 Compatibility:
  - 25 ms clock low timeout delay
  - 10 ms master cumulative clock low extend time
  - 25 ms slave cumulative clock low extend time
  - Hardware PEC generation/verification with ACK control
  - Address Resolution Protocol (ARP) supported
- PMBus Compatibility

Note: Some of the above features may not be available in certain products. The user should refer to the product data sheet, to identify the specific features supported by the I^{2}C interface implementation.

25.3 I^2C functional description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the I^2C bus by a data pin (SDA) and by a clock pin (SCL). It can be connected with a standard (up to 100 kHz) or fast (up to 400 kHz) I^2C bus.

25.3.1 Mode selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

By default, it operates in slave mode. The interface automatically switches from slave to master, after it generates a START condition and from master to slave, if an arbitration loss or a Stop generation occurs, allowing multimaster capability.

Communication flow

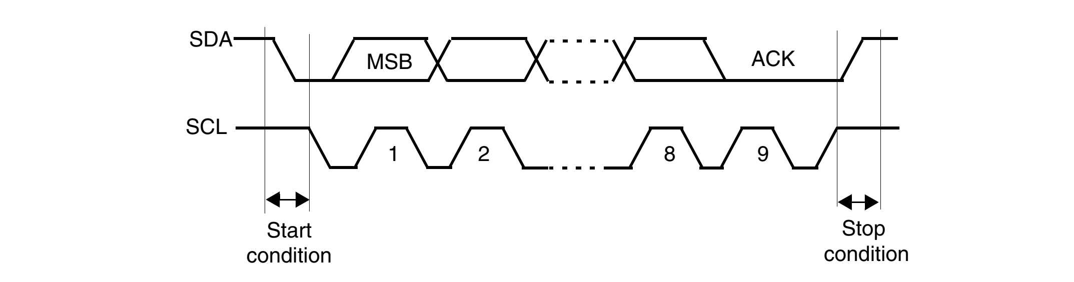
In Master mode, the I^2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a start condition and ends with a stop condition. Both start and stop conditions are generated in master mode by software.

In Slave mode, the interface is capable of recognizing its own addresses (7 or 10-bit), and the General Call address. The General Call address detection may be enabled or disabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the start condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to *Figure 206*.

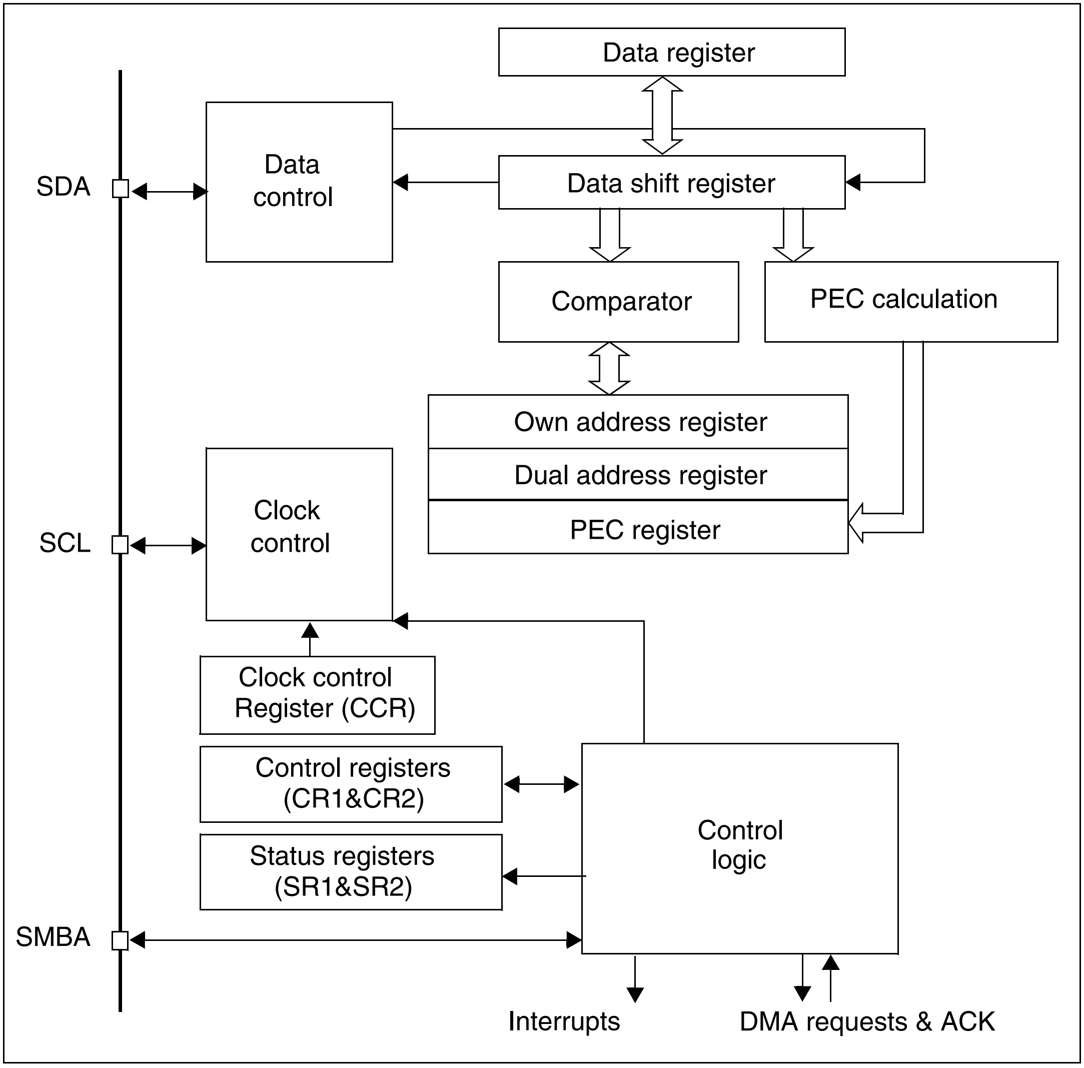
Figure 206. I^2C bus protocol



Acknowledge may be enabled or disabled by software. The I^2C interface addresses (dual addressing 7-bit/ 10-bit and/or general call address) can be selected by software.

The block diagram of the I^2C interface is shown in *Figure 207*.

Figure 207. I^2C block diagram



1. SMBA is an optional signal in SMBus mode. This signal is not applicable if SMBus is disabled.

25.3.2 I^2C slave mode

By default the I^2C interface operates in Slave mode. To switch from default Slave mode to Master mode a Start condition generation is needed.

The peripheral input clock must be programmed in the I2C_CR2 register in order to generate correct timings. The peripheral input clock frequency must be at least:

- 2 MHz in Standard mode
- 4 MHz in Fast mode

Note: Voltage scaling range 3 is not allowed in Fast mode, refer to Section 4.1.5: Dynamic voltage scaling management on page 61 for more details.

As soon as a start condition is detected, the address is received from the SDA line and sent to the shift register. Then it is compared with the address of the interface (OAR1) and with OAR2 (if ENDUAL=1) or the General Call address (if ENGC = 1).

Note: In 10-bit addressing mode, the comparison includes the header sequence (11110xx0), where xx denotes the two most significant bits of the address.

Header or address not matched: the interface ignores it and waits for another Start condition.

Header matched (10-bit mode only): the interface generates an acknowledge pulse if the ACK bit is set and waits for the 8-bit slave address.

Address matched: the interface generates in sequence:

- An acknowledge pulse if the ACK bit is set
- The ADDR bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.
- If ENDUAL=1, the software has to read the DUALF bit to check which slave address has been acknowledged.

In 10-bit mode, after receiving the address sequence the slave is always in Receiver mode. It will enter Transmitter mode on receiving a repeated Start condition followed by the header sequence with matching address bits and the least significant bit set (11110xx1).

The TRA bit indicates whether the slave is in Receiver or Transmitter mode.

Slave transmitter

Following the address reception and after clearing ADDR, the slave sends bytes from the DR register to the SDA line via the internal shift register.

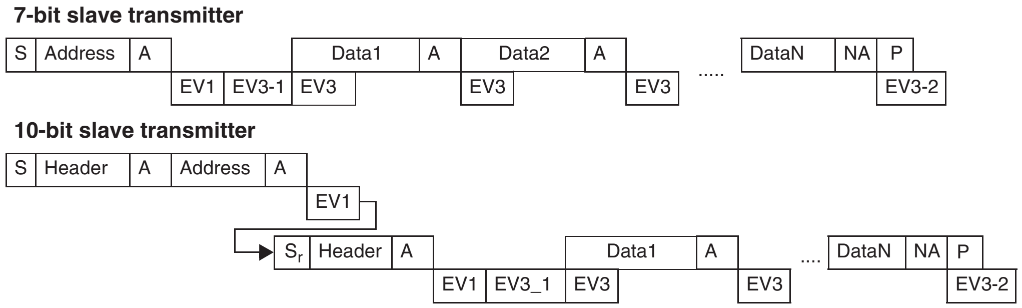
The slave stretches SCL low until ADDR is cleared and DR filled with the data to be sent (see *Figure 208* Transfer sequencing EV1 EV3).

When the acknowledge pulse is received:

- The TxE bit is set by hardware with an interrupt if the ITEVFEN and the ITBUFEN bits are set.

If TxE is set and some data were not written in the I2C_DR register before the end of the next data transmission, the BTF bit is set and the interface waits until BTF is cleared by a read to I2C_SR1 followed by a write to the I2C_DR register, stretching SCL low.

Figure 208. Transfer sequence diagram for slave transmitter



Legend: S= Start, S_r = Repeated Start, P= Stop, A= Acknowledge, NA= Non-acknowledge, EVx= Event (with interrupt if ITEVFEN=1)

EV1: ADDR=1, cleared by reading SR1 followed by reading SR2
EV3-1: TxE=1, shift register empty, data register empty, write Data1 in DR.
EV3: TxE=1, shift register not empty, data register empty, cleared by writing DR
EV3-2: AF=1; AF is cleared by writing ‘0’ in AF bit of SR1 register.

ai18209

1. The EV1 and EV3_1 events stretch SCL low until the end of the corresponding software sequence.
2. The EV3 event stretches SCL low if the software sequence is not completed before the end of the next byte transmission

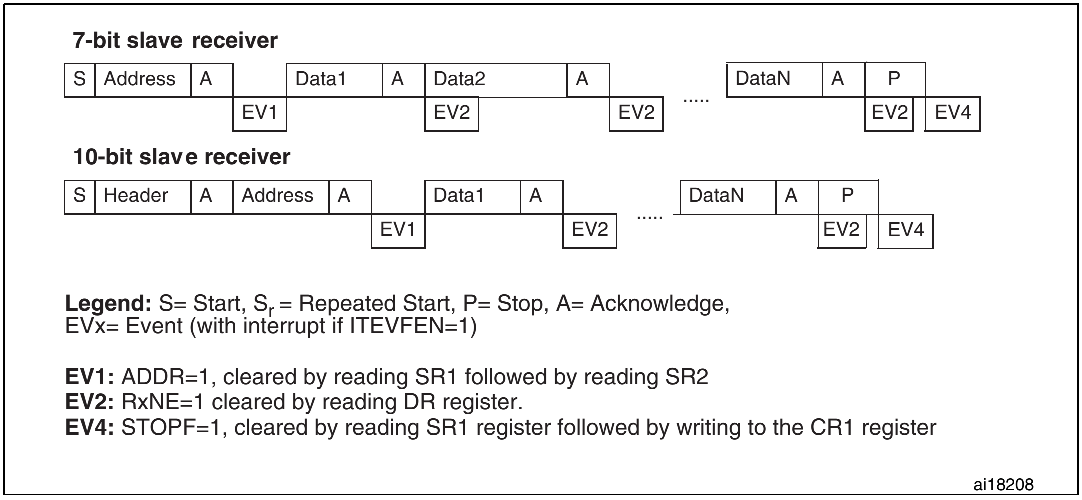
Slave receiver

Following the address reception and after clearing ADDR, the slave receives bytes from the SDA line into the DR register via the internal shift register. After each byte the interface generates in sequence:

- An acknowledge pulse if the ACK bit is set
- The RxNE bit is set by hardware and an interrupt is generated if the ITEVFEN and ITBUFEN bit is set.

If RxNE is set and the data in the DR register is not read before the end of the next data reception, the BTF bit is set and the interface waits until BTF is cleared by a read from the I2C_DR register, stretching SCL low (see *Figure 209* Transfer sequencing).

Figure 209. Transfer sequence diagram for slave receiver



Legend: S= Start, S_r = Repeated Start, P= Stop, A= Acknowledge,
EVx= Event (with interrupt if ITEVFEN=1)

EV1: ADDR=1, cleared by reading SR1 followed by reading SR2
EV2: RxNE=1 cleared by reading DR register.
EV4: STOPF=1, cleared by reading SR1 register followed by writing to the CR1 register

1. The EV1 event stretches SCL low until the end of the corresponding software sequence.
2. The EV2 event stretches SCL low if the software sequence is not completed before the end of the next byte reception.
3. After checking the SR1 register content, the user should perform the complete clearing sequence for each flag found set.
   Thus, for ADDR and STOPF flags, the following sequence is required inside the I2C interrupt routine:
   READ SR1
   if (ADDR == 1) {READ SR1; READ SR2}
   if (STOPF == 1) {READ SR1; WRITE CR1}
   The purpose is to make sure that both ADDR and STOPF flags are cleared if both are found set.

Closing slave communication

After the last data byte is transferred a Stop Condition is generated by the master. The interface detects this condition and sets:

- The STOPF bit and generates an interrupt if the ITEVFEN bit is set.

The STOPF bit is cleared by a read of the SR1 register followed by a write to the CR1 register (see *Figure 209: Transfer sequence diagram for slave receiver* EV4).

25.3.3 I^2C master mode

In Master mode, the I^2C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a Start condition and ends with a Stop condition. Master mode is selected as soon as the Start condition is generated on the bus with a START bit.

The following is the required sequence in master mode.

- Program the peripheral input clock in I2C_CR2 Register in order to generate correct timings
- Configure the clock control registers
- Configure the rise time register
- Program the I2C_CR1 register to enable the peripheral
- Set the START bit in the I2C_CR1 register to generate a Start condition

The peripheral input clock frequency must be at least:

- 2 MHz in Standard mode
- 4 MHz in Fast mode

Note: Voltage scaling range 3 is not allowed in Fast mode, refer to Section 4.1.5: Dynamic voltage scaling management on page 61 for more details.

Start condition

Setting the START bit causes the interface to generate a Start condition and to switch to Master mode (M/SL bit set) when the BUSY bit is cleared.

Note: In master mode, setting the START bit causes the interface to generate a ReStart condition at the end of the current byte transfer.

Once the Start condition is sent:

- The SB bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.

Then the master waits for a read of the SR1 register followed by a write in the DR register with the Slave address (see *Figure 210* and *Figure 211* Transfer sequencing EV5).

Slave address transmission

Then the slave address is sent to the SDA line via the internal shift register.

- In 10-bit addressing mode, sending the header sequence causes the following event:
  - The ADD10 bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.

  Then the master waits for a read of the SR1 register followed by a write in the DR register with the second address byte (see *Figure 210* and *Figure 211* Transfer sequencing).
  - The ADDR bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.

  Then the master waits for a read of the SR1 register followed by a read of the SR2 register (see *Figure 210* and *Figure 211* Transfer sequencing).
- In 7-bit addressing mode, one address byte is sent.

  As soon as the address byte is sent,
  - The ADDR bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.

  Then the master waits for a read of the SR1 register followed by a read of the SR2 register (see *Figure 210* and *Figure 211* Transfer sequencing).

The master can decide to enter Transmitter or Receiver mode depending on the LSB of the slave address sent.

- In 7-bit addressing mode,
  - To enter Transmitter mode, a master sends the slave address with LSB reset.
  - To enter Receiver mode, a master sends the slave address with LSB set.
- In 10-bit addressing mode,
  - To enter Transmitter mode, a master sends the header (11110xx0) and then the slave address, (where xx denotes the two most significant bits of the address).
  - To enter Receiver mode, a master sends the header (11110xx0) and then the slave address. Then it should send a repeated Start condition followed by the header (11110xx1), (where xx denotes the two most significant bits of the address).

  The TRA bit indicates whether the master is in Receiver or Transmitter mode.

Master transmitter

Following the address transmission and after clearing ADDR, the master sends bytes from the DR register to the SDA line via the internal shift register.

The master waits until the first data byte is written into I2C_DR (see *Figure 210* Transfer sequencing EV8_1).

When the acknowledge pulse is received, the TxE bit is set by hardware and an interrupt is generated if the ITEVFEN and ITBUFEN bits are set.

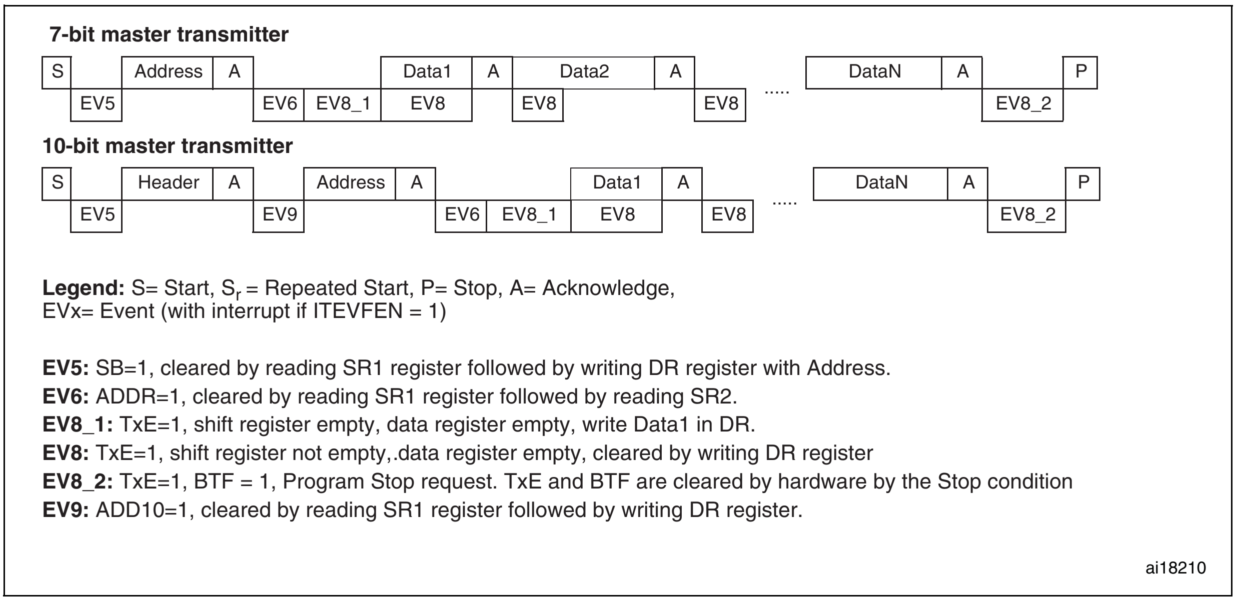
If TxE is set and a data byte was not written in the DR register before the end of the last data transmission, BTF is set and the interface waits until BTF is cleared by a write to I2C_DR, stretching SCL low.

Closing the communication

After the last byte is written to the DR register, the STOP bit is set by software to generate a Stop condition (see *Figure 210* Transfer sequencing EV8_2). The interface automatically goes back to slave mode (M/SL bit cleared).

Note: Stop condition should be programmed during EV8_2 event, when either TxE or BTF is set.

Figure 210. Transfer sequence diagram for master transmitter



Legend: S= Start, S_r = Repeated Start, P= Stop, A= Acknowledge,
EVx= Event (with interrupt if ITEVFEN = 1)

EV5: SB=1, cleared by reading SR1 register followed by writing DR register with Address.
EV6: ADDR=1, cleared by reading SR1 register followed by reading SR2.
EV8_1: TxE=1, shift register empty, data register empty, write Data1 in DR.
EV8: TxE=1, shift register not empty,.data register empty, cleared by writing DR register
EV8_2: TxE=1, BTF = 1, Program Stop request. TxE and BTF are cleared by hardware by the Stop condition
EV9: ADD10=1, cleared by reading SR1 register followed by writing DR register.

1. The EV5, EV6, EV9, EV8_1 and EV8_2 events stretch SCL low until the end of the corresponding software sequence.
2. The EV8 event stretches SCL low if the software sequence is not complete before the end of the next byte transmission.

Master receiver

Following the address transmission and after clearing ADDR, the I^2C interface enters Master Receiver mode. In this mode the interface receives bytes from the SDA line into the DR register via the internal shift register. After each byte the interface generates in sequence:

1. An acknowledge pulse if the ACK bit is set
2. The RxNE bit is set and an interrupt is generated if the ITEVFEN and ITBUFEN bits are set (see *Figure 211* Transfer sequencing EV7).

If the RxNE bit is set and the data in the DR register is not read before the end of the last data reception, the BTF bit is set by hardware and the interface waits until BTF is cleared by a read in the DR register, stretching SCL low.

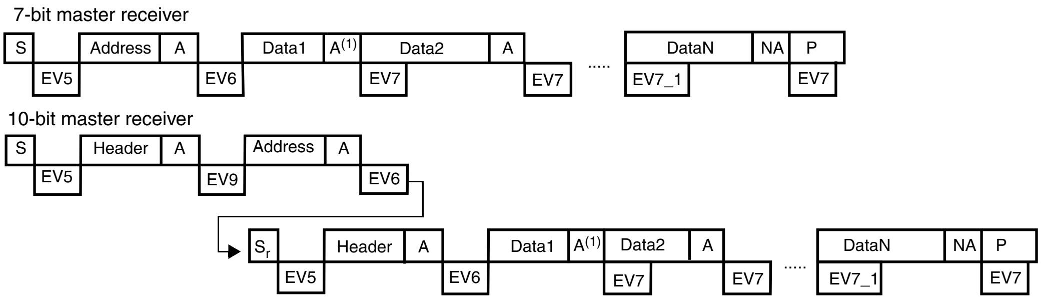
Closing the communication

The master sends a NACK for the last byte received from the slave. After receiving this NACK, the slave releases the control of the SCL and SDA lines. Then the master can send a Stop/Restart condition.

1. To generate the nonacknowledge pulse after the last received data byte, the ACK bit must be cleared just after reading the second last data byte (after second last RxNE event).
2. In order to generate the Stop/Restart condition, software must set the STOP/START bit after reading the second last data byte (after the second last RxNE event).
3. In case a single byte has to be received, the Acknowledge disable is made during EV6 (before ADDR flag is cleared) and the STOP condition generation is made after EV6.

After the Stop condition generation, the interface goes automatically back to slave mode (M/SL bit cleared).

Figure 211. Transfer sequence diagram for master receiver



Legend: S= Start, S_r = repeated Start, P = Stop, A= Ackowledge, NA = Non-acknowledge,
EVx= Event (with interrupt if ITEVFEN=1)

EV5: SB=1, cleared by reading SR1 register followed by writing DR register.

EV6: ADDR=1, cleared by reading SR1 register followed by reading SR2. In 10-bit master receiver mode, this sequence should be followed by writing CR2 with SART = 1.

In case of the reception of 1 byte, the Acknowledge disable must be performed suring EV6 event, i.e. before clearing ADDR flag.

EV7: RxNE = 1 cleared by reading DR register.

EV7_1: RxNE = 1 cleared by reading DR register, programming ACK = 0 and STOP request.

EV9: ADD10 = 1, cleared by reading SR1 register followed by writng DR register.

ai17540c

1. If a single byte is received, it is NA.
2. The EV5, EV6 and EV9 events stretch SCL low until the end of the corresponding software sequence.
3. The EV7 event stretches SCL low if the software sequence is not completed before the end of the next byte reception.
4. The EV7_1 software sequence must be completed before the ACK pulse of the current byte transfer.

The procedures described below are recommended if the EV7-1 software sequence is not completed before the ACK pulse of the current byte transfer.

These procedures must be followed to make sure:

- The ACK bit is set low on time before the end of the last data reception
- The STOP bit is set high after the last data reception without reception of supplementary data.

For 2-byte reception:

- Wait until ADDR = 1 (SCL stretched low until the ADDR flag is cleared)
- Set ACK low, set POS high
- Clear ADDR flag
- Wait until BTF = 1 (Data 1 in DR, Data2 in shift register, SCL stretched low until a data 1 is read)
- Set STOP high
- Read data 1 and 2

For N >2 -byte reception, from N-2 data reception

- Wait until BTF = 1 (data N-2 in DR, data N-1 in shift register, SCL stretched low until data N-2 is read)
- Set ACK low
- Read data N-2
- Wait until BTF = 1 (data N-1 in DR, data N in shift register, SCL stretched low until a data N-1 is read)
- Set STOP high
- Read data N-1 and N

25.3.4 Error conditions

The following are the error conditions which may cause communication to fail.

Bus error (BERR)

This error occurs when the I^2C interface detects an external Stop or Start condition during an address or a data transfer. In this case:

- the BERR bit is set and an interrupt is generated if the ITERREN bit is set
- in Slave mode: data are discarded and the lines are released by hardware:
  - in case of a misplaced Start, the slave considers it is a restart and waits for an address, or a Stop condition
  - in case of a misplaced Stop, the slave behaves like for a Stop condition and the lines are released by hardware
- In Master mode: the lines are not released and the state of the current transmission is not affected. It is up to the software to abort or not the current transmission

Acknowledge failure (AF)

This error occurs when the interface detects a nonacknowledge bit. In this case:

- the AF bit is set and an interrupt is generated if the ITERREN bit is set
- a transmitter which receives a NACK must reset the communication:
  - If Slave: lines are released by hardware
  - If Master: a Stop or repeated Start condition must be generated by software

Arbitration lost (ARLO)

This error occurs when the I^2C interface detects an arbitration lost condition. In this case,

- the ARLO bit is set by hardware (and an interrupt is generated if the ITERREN bit is set)
- the I^2C Interface goes automatically back to slave mode (the M/SL bit is cleared). When the I^2C loses the arbitration, it is not able to acknowledge its slave address in the same transfer, but it can acknowledge it after a repeated Start from the winning master.
- lines are released by hardware

Overrun/underrun error (OVR)

An overrun error can occur in slave mode when clock stretching is disabled and the I^2C interface is receiving data. The interface has received a byte (RxNE=1) and the data in DR has not been read, before the next byte is received by the interface. In this case,

- The last received byte is lost.
- In case of Overrun error, software should clear the RxNE bit and the transmitter should re-transmit the last received byte.

Underrun error can occur in slave mode when clock stretching is disabled and the I^2C interface is transmitting data. The interface has not updated the DR with the next byte (TxE=1), before the clock comes for the next byte. In this case,

- The same byte in the DR register will be sent again
- The user should make sure that data received on the receiver side during an underrun error are discarded and that the next bytes are written within the clock low time specified in the I^2C bus standard.

For the first byte to be transmitted, the DR must be written after ADDR is cleared and before the first SCL rising edge. If not possible, the receiver must discard the first data.

25.3.5 SDA/SCL line control

- If clock stretching is enabled:
  - Transmitter mode: If TxE=1 and BTF=1: the interface holds the clock line low before transmission to wait for the microcontroller to write the byte in the Data Register (both buffer and shift register are empty).
  - Receiver mode: If RxNE=1 and BTF=1: the interface holds the clock line low after reception to wait for the microcontroller to read the byte in the Data Register (both buffer and shift register are full).
- If clock stretching is disabled in Slave mode:
  - Overrun Error in case of RxNE=1 and no read of DR has been done before the next byte is received. The last received byte is lost.
  - Underrun Error in case TxE=1 and no write into DR has been done before the next byte must be transmitted. The same byte will be sent again.
  - Write Collision not managed.

25.3.6 SMBus

Introduction

The System Management Bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on I^2C principles of operation. SMBus provides a control bus for system and power management related tasks. A system may use SMBus to pass messages to and from devices instead of toggling individual control lines.

The System Management Bus Specification refers to three types of devices. A *slave* is a device that is receiving or responding to a command. A *master* is a device that issues commands, generates the clocks, and terminates the transfer. A *host* is a specialized master that provides the main interface to the system's CPU. A host must be a master-slave and must support the SMBus host notify protocol. Only one host is allowed in a system.

Similarities between SMBus and I^2C

- 2 wire bus protocol (1 Clk, 1 Data) + SMBus Alert line optional
- Master-slave communication, Master provides clock
- Multi master capability
- SMBus data format similar to I^2C 7-bit addressing format (*Figure 206*).

Differences between SMBus and I^2C

The following table describes the differences between SMBus and I^2C.

Table 124. SMBus vs. I^2C

SMBus	I^2C
Max. speed 100 kHz	Max. speed 400 kHz
Min. clock speed 10 kHz	No minimum clock speed
35 ms clock low timeout	No timeout
Logic levels are fixed	Logic levels are V_{DD} dependent
Different address types (reserved, dynamic etc.)	7-bit, 10-bit and general call slave address types
Different bus protocols (quick command, process call etc.)	No bus protocols

SMBus application usage

With System Management Bus, a device can provide manufacturer information, tell the system what its model/part number is, save its state for a suspend event, report different types of errors, accept control parameters, and return its status. SMBus provides a control bus for system and power management related tasks.

Device identification

Any device that exists on the System Management Bus as a slave has a unique address called the Slave Address. For the list of reserved slave addresses, refer to the SMBus specification version. 2.0 (*http://smbus.org/specs/*).

Bus protocols

The SMBus specification supports up to 9 bus protocols. For more details of these protocols and SMBus address types, refer to SMBus specification version. 2.0 (*http://smbus.org/specs/*). These protocols should be implemented by the user software.

Address resolution protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. The Address Resolution Protocol (ARP) has the following attributes:

- Address assignment uses the standard SMBus physical layer arbitration mechanism
- Assigned addresses remain constant while device power is applied; address retention through device power loss is also allowed
- No additional SMBus packet overhead is incurred after address assignment. (i.e. subsequent accesses to assigned slave addresses have the same overhead as accesses to fixed address devices.)
- Any SMBus master can enumerate the bus

Unique device identifier (UDID)

In order to provide a mechanism to isolate each device for the purpose of address assignment, each device must implement a unique device identifier (UDID).

For the details on 128 bit UDID and more information on ARP, refer to SMBus specification version 2.0 (*http://smbus.org/specs/*).

SMBus alert mode

SMBus Alert is an optional signal with an interrupt line for devices that want to trade their ability to master for a pin. SMBA is a wired-AND signal just as the SCL and SDA signals are. SMBA is used in conjunction with the SMBus General Call Address. Messages invoked with the SMBus are 2 bytes long.

A slave-only device can signal the host through SMBA that it wants to talk by setting ALERT bit in I2C_CR1 register. The host processes the interrupt and simultaneously accesses all SMBA devices through the *Alert Response Address* (known as ARA having a value 0001 100X). Only the device(s) which pulled SMBA low will acknowledge the Alert Response Address. This status is identified using SMBALERT Status flag in I2C_SR1 register. The host performs a modified Receive Byte operation. The 7 bit device address provided by the slave transmit device is placed in the 7 most significant bits of the byte. The eighth bit can be a zero or one.

If more than one device pulls SMBA low, the highest priority (lowest address) device will win communication rights via standard arbitration during the slave address transfer. After acknowledging the slave address the device must disengage its SMBA pull-down. If the host still sees SMBA low when the message transfer is complete, it knows to read the ARA again.
A host which does not implement the SMBA signal may periodically access the ARA.

For more details on SMBus Alert mode, refer to SMBus specification version 2.0 (*http://smbus.org/specs/*).

Timeout error

There are differences in the timing specifications between I^2C and SMBus.
SMBus defines a clock low timeout, TIMEOUT of 35 ms. Also SMBus specifies TLOW: SEXT as the cumulative clock low extend time for a slave device. SMBus specifies TLOW: MEXT as the cumulative clock low extend time for a master device. For more details on these timeouts, refer to SMBus specification version 2.0 (*http://smbus.org/specs/*).

The status flag Timeout or Tlow Error in I2C_SR1 shows the status of this feature.

How to use the interface in SMBus mode

To switch from I^{2}C mode to SMBus mode, the following sequence should be performed.

- Set the SMBus bit in the I2C_CR1 register
- Configure the SMBTYPE and ENARP bits in the I2C_CR1 register as required for the application

If you want to configure the device as a master, follow the Start condition generation procedure in *Section 25.3.3: I2C master mode*. Otherwise, follow the sequence in *Section 25.3.2: I2C slave mode*.

The application has to control the various SMBus protocols by software.

- SMB Device Default Address acknowledged if ENARP=1 and SMBTYPE=0
- SMB Host Header acknowledged if ENARP=1 and SMBTYPE=1
- SMB Alert Response Address acknowledged if SMBALERT=1

25.3.7 DMA requests

DMA requests (when enabled) are generated only for data transfer. DMA requests are generated by Data Register becoming empty in transmission and Data Register becoming full in reception. The DMA must be initialized and enabled before the I2C data transfer. The DMAEN bit must be set in the I2C_CR2 register before the ADDR event. In master mode or in slave mode when clock stretching is enabled, the DMAEN bit can also be set during the ADDR event, before clearing the ADDR flag. The DMA request must be served before the end of the current byte transfer. When the number of data transfers which has been programmed for the corresponding DMA channel is reached, the DMA controller sends an End of Transfer EOT signal to the I^{2}C interface and generates a Transfer Complete interrupt if enabled:

- Master transmitter: In the interrupt routine after the EOT interrupt, disable DMA requests then wait for a BTF event before programming the Stop condition.
- Master receiver
  - When the number of bytes to be received is equal to or greater than two, the DMA controller sends a hardware signal, EOT_1, corresponding to the last but one data byte (number_of_bytes – 1). If, in the I2C_CR2 register, the LAST bit is set, I^{2}C automatically sends a NACK after the next byte following EOT_1. The user can generate a Stop condition in the DMA Transfer Complete interrupt routine if enabled.
  - When a single byte must be received: the NACK must be programmed during EV6 event, i.e. program ACK=0 when ADDR=1, before clearing ADDR flag. Then the user can program the STOP condition either after clearing ADDR flag, or in the DMA Transfer Complete interrupt routine.

Transmission using DMA

DMA mode can be enabled for transmission by setting the DMAEN bit in the I2C_CR2 register. Data will be loaded from a Memory area configured using the DMA peripheral (refer to the DMA specification) to the I2C_DR register whenever the TxE bit is set. To map a DMA channel for I^{2}C transmission, perform the following sequence. Here x is the channel number.

1. Set the I2C_DR register address in the DMA_CPARx register. The data will be moved to this address from the memory after each TxE event.
2. Set the memory address in the DMA_CMARx register. The data will be loaded into I2C_DR from this memory after each TxE event.
3. Configure the total number of bytes to be transferred in the DMA_CNDTRx register. After each TxE event, this value will be decremented.
4. Configure the channel priority using the PL[0:1] bits in the DMA_CCRx register
5. Set the DIR bit and, in the DMA_CCRx register, configure interrupts after half transfer or full transfer depending on application requirements.
6. Activate the channel by setting the EN bit in the DMA_CCRx register.

When the number of data transfers which has been programmed in the DMA Controller registers is reached, the DMA controller sends an End of Transfer EOT/ EOT_1 signal to the I^{2}C interface and the DMA generates an interrupt, if enabled, on the DMA channel interrupt vector.

Note: Do not enable the ITBUFEN bit in the I2C_CR2 register if DMA is used for transmission.

Reception using DMA

DMA mode can be enabled for reception by setting the DMAEN bit in the I2C_CR2 register. Data will be loaded from the I2C_DR register to a Memory area configured using the DMA peripheral (refer to the DMA specification) whenever a data byte is received. To map a DMA channel for I^{2}C reception, perform the following sequence. Here x is the channel number.

1. Set the I2C_DR register address in DMA_CPARx register. The data will be moved from this address to the memory after each RxNE event.
2. Set the memory address in the DMA_CMARx register. The data will be loaded from the I2C_DR register to this memory area after each RxNE event.
3. Configure the total number of bytes to be transferred in the DMA_CNDTRx register. After each RxNE event, this value will be decremented.
4. Configure the channel priority using the PL[0:1] bits in the DMA_CCRx register
5. Reset the DIR bit and configure interrupts in the DMA_CCRx register after half transfer or full transfer depending on application requirements.
6. Activate the channel by setting the EN bit in the DMA_CCRx register.

When the number of data transfers which has been programmed in the DMA Controller registers is reached, the DMA controller sends an End of Transfer EOT/ EOT_1 signal to the I^{2}C interface and DMA generates an interrupt, if enabled, on the DMA channel interrupt vector.

Note: Do not enable the ITBUFEN bit in the I2C_CR2 register if DMA is used for reception.

25.3.8 Packet error checking

A PEC calculator has been implemented to improve the reliability of communication. The PEC is calculated by using the $C(x) = x^8 + x^2 + x + 1$ CRC-8 polynomial serially on each bit.

- PEC calculation is enabled by setting the ENPEC bit in the I2C_CR1 register. PEC is a CRC-8 calculated on all message bytes including addresses and R/W bits.
  - In transmission: set the PEC transfer bit in the I2C_CR1 register after the TxE event corresponding to the last byte. The PEC will be transferred after the last transmitted byte.
  - In reception: set the PEC bit in the I2C_CR1 register after the RxNE event corresponding to the last byte so that the receiver sends a NACK if the next received byte is not equal to the internally calculated PEC. In case of Master-Receiver, a NACK must follow the PEC whatever the check result. The PEC must be set before the ACK of the CRC reception in slave mode. It must be set when the ACK is set low in master mode.
- A PECERR error flag/interrupt is also available in the I2C_SR1 register.
- If DMA and PEC calculation are both enabled:-
  - In transmission: when the I^2C interface receives an EOT signal from the DMA controller, it automatically sends a PEC after the last byte.
  - In reception: when the I^2C interface receives an EOT_1 signal from the DMA controller, it will automatically consider the next byte as a PEC and will check it. A DMA request is generated after PEC reception.
- To allow intermediate PEC transfers, a control bit is available in the I2C_CR2 register (LAST bit) to determine if it is really the last DMA transfer or not. If it is the last DMA request for a master receiver, a NACK is automatically sent after the last received byte.
- PEC calculation is corrupted by an arbitration loss.

25.4 I^2C interrupts

The table below gives the list of I^2C interrupt requests.

Table 125. I^2C Interrupt requests

Interrupt event	Event flag	Enable control bit
Start bit sent (Master)	SB	ITEVFEN
Address sent (Master) or Address matched (Slave)	ADDR	
10-bit header sent (Master)	ADD10	
Stop received (Slave)	STOPF	
Data byte transfer finished	BTF	
Receive buffer not empty	RxNE	ITEVFEN and ITBUFEN
Transmit buffer empty	TxE	

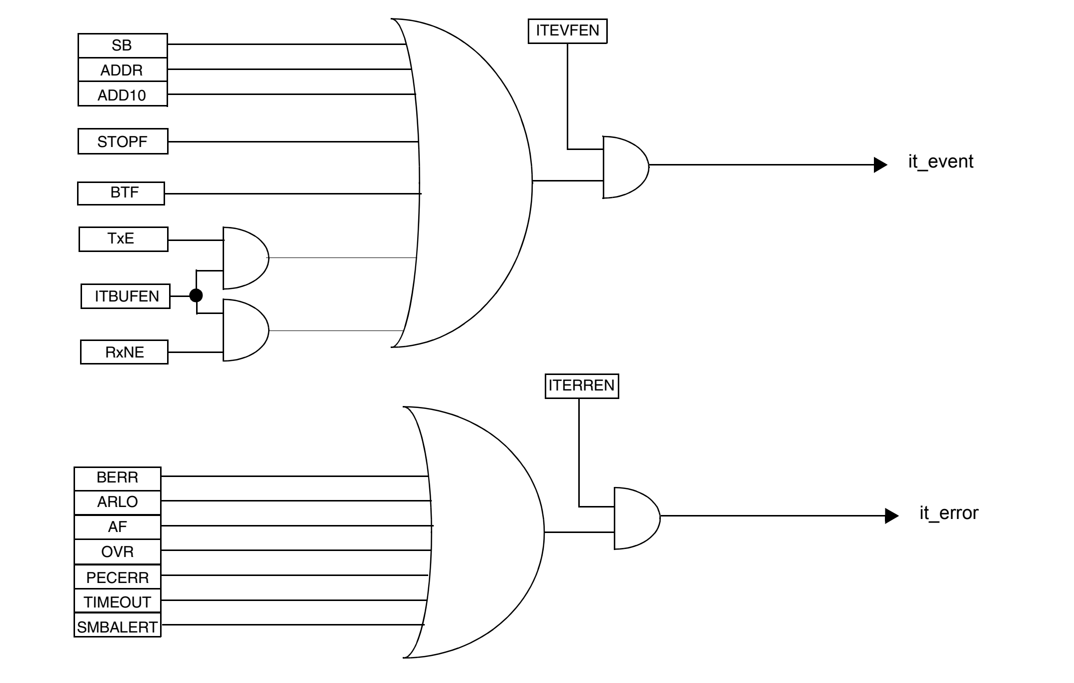
Table 125. I^2C Interrupt requests (continued)

Interrupt event	Event flag	Enable control bit
Bus error	BERR	ITERREN
Arbitration loss (Master)	ARLO	
Acknowledge failure	AF	
Overrun/Underrun	OVR	
PEC error	PECERR	
Timeout/Tlow error	TIMEOUT	
SMBus Alert	SMBALERT	

Note: SB, ADDR, ADD10, STOPF, BTF, RxNE and TxE are logically ORed on the same interrupt channel.

BERR, ARLO, AF, OVR, PECERR, TIMEOUT and SMBALERT are logically ORed on the same interrupt channel.

Figure 212. I^2C interrupt mapping diagram



25.5 I^{2}C debug mode

When the microcontroller enters the debug mode (Cortex™-M3 core halted), the SMBUS timeout either continues to work normally or stops, depending on the DBG_I2Cx_SMBUS_TIMEOUT configuration bits in the DBG module. For more details, refer to *Section 29.16.2: Debug support for timers, watchdog and I2C on page 833*.

25.6 I^{2}C registers

Refer to for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16 bits) or words (32 bits).

25.6.1 I^{2}C Control register 1 (I2C_CR1)

Address offset: 0x00
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWRST	Res.	ALERT	PEC	POS	ACK	STOP	START	NO STRETCH	ENGC	ENPEC	ENARP	SMB TYPE	Res.	SMBUS	PE
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit 15 **SWRST**: Software reset

When set, the I2C is under reset state. Before resetting this bit, make sure the I2C lines are released and the bus is free.

0: I^{2}C Peripheral not under reset

1: I^{2}C Peripheral under reset state

Note: This bit can be used to reinitialize the peripheral after an error or a locked state. As an example, if the BUSY bit is set and remains locked due to a glitch on the bus, the SWRST bit can be used to exit from this state.

Bit 14 Reserved, must be kept at reset value

Bit 13 **ALERT**: SMBus alert

This bit is set and cleared by software, and cleared by hardware when PE=0.

0: Releases SMBA pin high. Alert Response Address Header followed by NACK.

1: Drives SMBA pin low. Alert Response Address Header followed by ACK.

Bit 12 **PEC**: Packet error checking

This bit is set and cleared by software, and cleared by hardware when PEC is transferred or by a START or Stop condition or when PE=0.

0: No PEC transfer

1: PEC transfer (in Tx or Rx mode)

Note: PEC calculation is corrupted by an arbitration loss.

Bit 11 **POS**: Acknowledge/PEC Position (for data reception)

This bit is set and cleared by software and cleared by hardware when PE=0.

0: ACK bit controls the (N)ACK of the current byte being received in the shift register. The PEC bit indicates that current byte in shift register is a PEC.

1: ACK bit controls the (N)ACK of the next byte which will be received in the shift register. The PEC bit indicates that the next byte in the shift register is a PEC

Note: The POS bit must be used only in 2-byte reception configuration in master mode. It must be configured before data reception starts, as described in the 2-byte reception procedure recommended in Section : Master receiver on page 626.

Bit 10 **ACK**: Acknowledge enable

This bit is set and cleared by software and cleared by hardware when PE=0.
0: No acknowledge returned
1: Acknowledge returned after a byte is received (matched address or data)

Bit 9 **STOP**: Stop generation

The bit is set and cleared by software, cleared by hardware when a Stop condition is detected, set by hardware when a timeout error is detected.
In Master Mode:
0: No Stop generation.
1: Stop generation after the current byte transfer or after the current Start condition is sent.
In Slave mode:
0: No Stop generation.
1: Release the SCL and SDA lines after the current byte transfer.

Bit 8 **START**: Start generation

This bit is set and cleared by software and cleared by hardware when start is sent or PE=0.
In Master Mode:
0: No Start generation
1: Repeated start generation
In Slave mode:
0: No Start generation
1: Start generation when the bus is free

Bit 7 **NOSTRETCH**: Clock stretching disable (Slave mode)

This bit is used to disable clock stretching in slave mode when ADDR or BTF flag is set, until it is reset by software.
0: Clock stretching enabled
1: Clock stretching disabled

Bit 6 **ENGC**: General call enable

0: General call disabled. Address 00h is NACKed.
1: General call enabled. Address 00h is ACKed.

Bit 5 **ENPEC:** PEC enable

0: PEC calculation disabled
1: PEC calculation enabled

Bit 4 **ENARP**: ARP enable

0: ARP disable
1: ARP enable
SMBus Device default address recognized if SMBTYPE=0
SMBus Host address recognized if SMBTYPE=1

Bit 3 **SMBTYPE**: SMBus type

0: SMBus Device
1: SMBus Host

Bit 2 Reserved, must be kept at reset value

Bit 1 **SMBUS**: SMBus mode

0: I^2C mode
1: SMBus mode

Bit 0 **PE**: Peripheral enable

0: Peripheral disable
1: Peripheral enable

Note: If this bit is reset while a communication is on going, the peripheral is disabled at the end of the current communication, when back to IDLE state.
All bit resets due to PE=0 occur at the end of the communication.
In master mode, this bit must not be reset before the end of the communication.

Note: When the STOP, START or PEC bit is set, the software must not perform any write access to I2C_CR1 before this bit is cleared by hardware. Otherwise there is a risk of setting a second STOP, START or PEC request.

25.6.2 I^2C Control register 2 (I2C_CR2)

Address offset: 0x04
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			LAST	DMA EN	ITBUF EN	ITEVTE N	ITERR EN	Reserved		FREQ[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value

Bit 12 **LAST**: DMA last transfer

0: Next DMA EOT is not the last transfer
1: Next DMA EOT is the last transfer

Note: This bit is used in master receiver mode to permit the generation of a NACK on the last received data.

Bit 11 **DMAEN**: DMA requests enable

0: DMA requests disabled
1: DMA request enabled when TxE=1 or RxNE =1

Bit 10 **ITBUFEN**: Buffer interrupt enable

0: TxE = 1 or RxNE = 1 does not generate any interrupt.
1: TxE = 1 or RxNE = 1 generates Event Interrupt (whatever the state of DMAEN)

Bit 9 **ITEVTEN**: Event interrupt enable

0: Event interrupt disabled
1: Event interrupt enabled
This interrupt is generated when:

- SB = 1 (Master)
- ADDR = 1 (Master/Slave)
- ADD10= 1 (Master)
- STOPF = 1 (Slave)
- BTF = 1 with no TxE or RxNE event
- TxE event to 1 if ITBUFEN = 1
- RxNE event to 1if ITBUFEN = 1

ITERREN: Error interrupt enable

0: Error interrupt disabled
1: Error interrupt enabled
This interrupt is generated when:

- BERR = 1
- ARLO = 1
- AF = 1
- OVR = 1
- PECERR = 1
- TIMEOUT = 1
- SMBALERT = 1

Bits 7:6 Reserved, must be kept at reset value

Bits 5:0 **FREQ[5:0]**: Peripheral clock frequency

The peripheral clock frequency must be configured using the input APB clock frequency (I2C peripheral connected to APB1). The minimum allowed frequency is 2 MHz, the maximum frequency is limited by the maximum APB1 frequency (32 MHz) and an intrinsic limitation of 46 MHz.

0b000000: Not allowed
0b000001: Not allowed
0b000010: 2 MHz
...
0b100000: 32 MHz
Higher than 0b100000: Not allowed

25.6.3 I^2C Own address register 1 (I2C_OAR1)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD MODE	Reserved					ADD[9:8]		ADD[7:1]							ADD0
rw						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 **ADDMODE** Addressing mode (slave mode)

0: 7-bit slave address (10-bit address not acknowledged)

1: 10-bit slave address (7-bit address not acknowledged)

Bit 14 Should always be kept at 1 by software.

Bits 13:10 Reserved, must be kept at reset value

Bits 9:8 **ADD[9:8]**: Interface address

7-bit addressing mode: don't care

10-bit addressing mode: bits9:8 of address

Bits 7:1 **ADD[7:1]**: Interface address

bits 7:1 of address

Bit 0 **ADD0**: Interface address

7-bit addressing mode: don't care

10-bit addressing mode: bit 0 of address

25.6.4 I^2C Own address register 2 (I2C_OAR2)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ADD2[7:1]							ENDUAL
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:1 **ADD2[7:1]**: Interface address

bits 7:1 of address in dual addressing mode

Bit 0 **ENDUAL**: Dual addressing mode enable

0: Only OAR1 is recognized in 7-bit addressing mode

1: Both OAR1 and OAR2 are recognized in 7-bit addressing mode

25.6.5 I^{2}C Data register (I2C_DR)

Address offset: 0x10

Reset value: 0x0000

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

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

Bits 7:0 **DR[7:0]** 8-bit data register

Byte received or to be transmitted to the bus.

– Transmitter mode: Byte transmission starts automatically when a byte is written in the DR register. A continuous transmit stream can be maintained if the next data to be transmitted is put in DR once the transmission is started (TxE=1)

– Receiver mode: Received byte is copied into DR (RxNE=1). A continuous transmit stream can be maintained if DR is read before the next data byte is received (RxNE=1).

Note: In slave mode, the address is not copied into DR.

Note: Write collision is not managed (DR can be written if TxE=0).

Note: If an ARLO event occurs on ACK pulse, the received byte is not copied into DR and so cannot be read.

25.6.6 I^{2}C Status register 1 (I2C_SR1)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMB ALERT	TIME OUT	Res.	PEC ERR	OVR	AF	ARLO	BERR	TxE	RxNE	Res.	STOPF	ADD10	BTF	ADDR	SB
rc_w0	rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	r	r		r	r	r	r	r

Bit 15 **SMBALERT**: SMBus alert

In SMBus host mode:
0: no SMBALERT
1: SMBALERT event occurred on pin
In SMBus slave mode:
0: no SMBALERT response address header
1: SMBALERT response address header to SMBALERT LOW received

– Cleared by software writing 0, or by hardware when PE=0.

Bit 14 **TIMEOUT**: Timeout or Tlow error

0: No timeout error
1: SCL remained LOW for 25 ms (Timeout)
or
Master cumulative clock low extend time more than 10 ms (Tlow:mext)
or
Slave cumulative clock low extend time more than 25 ms (Tlow:sext)

– When set in slave mode: slave resets the communication and lines are released by hardware

– When set in master mode: Stop condition sent by hardware

– Cleared by software writing 0, or by hardware when PE=0.

Note: This functionality is available only in SMBus mode.

Bit 13 Reserved, must be kept at reset value

Bit 12 **PECERR**: PEC Error in reception

0: no PEC error: receiver returns ACK after PEC reception (if ACK=1)
1: PEC error: receiver returns NACK after PEC reception (whatever ACK)

– Cleared by software writing 0, or by hardware when PE=0.

– Note: When the received CRC is wrong, PECERR is not set in slave mode if the PEC control bit is not set before the end of the CRC reception. Nevertheless, reading the PEC value determines whether the received CRC is right or wrong.

Bit 11 **OVR**: Overrun/Underrun

0: No overrun/underrun
1: Overrun or underrun

– Set by hardware in slave mode when NOSTRETCH=1 and:

– In reception when a new byte is received (including ACK pulse) and the DR register has not been read yet. New received byte is lost.

– In transmission when a new byte should be sent and the DR register has not been written yet. The same byte is sent twice.

– Cleared by software writing 0, or by hardware when PE=0.

Note: If the DR write occurs very close to SCL rising edge, the sent data is unspecified and a hold timing error occurs

Bit 10 **AF**: Acknowledge failure

0: No acknowledge failure
1: Acknowledge failure

– Set by hardware when no acknowledge is returned.

– Cleared by software writing 0, or by hardware when PE=0.

Bit 9 **ARLO**: Arbitration lost (master mode)

0: No Arbitration Lost detected
1: Arbitration Lost detected

Set by hardware when the interface loses the arbitration of the bus to another master

– Cleared by software writing 0, or by hardware when PE=0.

After an ARLO event the interface switches back automatically to Slave mode (M/SL=0).

Note: In SMBUS, the arbitration on the data in slave mode occurs only during the data phase, or the acknowledge transmission (not on the address acknowledge).

Bit 8 **BERR**: Bus error

0: No misplaced Start or Stop condition
1: Misplaced Start or Stop condition

– Set by hardware when the interface detects an SDA rising or falling edge while SCL is high, occurring in a non-valid position during a byte transfer.

– Cleared by software writing 0, or by hardware when PE=0.

Bit 7 **TxE**: Data register empty (transmitters)

0: Data register not empty
1: Data register empty

– Set when DR is empty in transmission. TxE is not set during address phase.

– Cleared by software writing to the DR register or by hardware after a start or a stop condition or when PE=0.

TxE is not set if either a NACK is received, or if next byte to be transmitted is PEC (PEC=1)

Note: TxE is not cleared by writing the first data being transmitted, or by writing data when BTF is set, as in both cases the data register is still empty.

Bit 6 **RxNE**: Data register not empty (receivers)

0: Data register empty
1: Data register not empty

– Set when data register is not empty in receiver mode. RxNE is not set during address phase.

– Cleared by software reading or writing the DR register or by hardware when PE=0.

RxNE is not set in case of ARLO event.

Note: RxNE is not cleared by reading data when BTF is set, as the data register is still full.

Bit 5 Reserved, must be kept at reset value

Bit 4 **STOPF**: Stop detection (slave mode)

0: No Stop condition detected
1: Stop condition detected

– Set by hardware when a Stop condition is detected on the bus by the slave after an acknowledge (if ACK=1).

– Cleared by software reading the SR1 register followed by a write in the CR1 register, or by hardware when PE=0

Note: The STOPF bit is not set after a NACK reception.
It is recommended to perform the complete clearing sequence (READ SR1 then WRITE CR1) after the STOPF is set. Refer to Figure 209: Transfer sequence diagram for slave receiver on page 623.

Bit 3 **ADD10**: 10-bit header sent (Master mode)

0: No ADD10 event occurred.
1: Master has sent first address byte (header).

– Set by hardware when the master has sent the first byte in 10-bit address mode.

– Cleared by software reading the SR1 register followed by a write in the DR register of the second address byte, or by hardware when PE=0.

Note: ADD10 bit is not set after a NACK reception

Bit 2 **BTF**: Byte transfer finished

0: Data byte transfer not done
1: Data byte transfer succeeded

– Set by hardware when NOSTRETCH=0 and:

– In reception when a new byte is received (including ACK pulse) and DR has not been read yet (RxNE=1).

– In transmission when a new byte should be sent and DR has not been written yet (TxE=1).

– Cleared by software by either a read or write in the DR register or by hardware after a start or a stop condition in transmission or when PE=0.

Note: The BTF bit is not set after a NACK reception

The BTF bit is not set if next byte to be transmitted is the PEC (TRA=1 in I2C_SR2 register and PEC=1 in I2C_CR1 register)

Bit 1 **ADDR**: Address sent (master mode)/matched (slave mode)

This bit is cleared by software reading SR1 register followed reading SR2, or by hardware when PE=0.

Address matched (Slave)
0: Address mismatched or not received.
1: Received address matched.

– Set by hardware as soon as the received slave address matched with the OAR registers content or a general call or a SMBus Device Default Address or SMBus Host or SMBus Alert is recognized. (when enabled depending on configuration).

Note: In slave mode, it is recommended to perform the complete clearing sequence (READ SR1 then READ SR2) after ADDR is set. Refer to Figure 209: Transfer sequence diagram for slave receiver on page 623.

Address sent (Master)
0: No end of address transmission
1: End of address transmission

– For 10-bit addressing, the bit is set after the ACK of the 2nd byte.

– For 7-bit addressing, the bit is set after the ACK of the byte.

Note: ADDR is not set after a NACK reception

Bit 0 **SB**: Start bit (Master mode)

0: No Start condition
1: Start condition generated.

– Set when a Start condition generated.

– Cleared by software by reading the SR1 register followed by writing the DR register, or by hardware when PE=0

25.6.7 I^{2}C Status register 2 (I2C_SR2)

Address offset: 0x18
Reset value: 0x0000

Note: Reading I2C_SR2 after reading I2C_SR1 clears the ADDR flag, even if the ADDR flag was set after reading I2C_SR1. Consequently, I2C_SR2 must be read only when ADDR is found set in I2C_SR1 or when the STOPF bit is cleared.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEC[7:0]								DUALF	SMB HOST	SMBDE FAULT	GEN CALL	Res.	TRA	BUSY	MSL
r	r	r	r	r	r	r	r	r	r	r	r		r	r	r

Bits 15:8 **PEC[7:0]** Packet error checking register
This register contains the internal PEC when ENPEC=1.

Bit 7 **DUALF**: Dual flag (Slave mode)
0: Received address matched with OAR1
1: Received address matched with OAR2
– Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.

Bit 6 **SMBHOST**: SMBus host header (Slave mode)
0: No SMBus Host address
1: SMBus Host address received when SMBTYPE=1 and ENARP=1.
– Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.

Bit 5 **SMBDEFAULT**: SMBus device default address (Slave mode)
0: No SMBus Device Default address
1: SMBus Device Default address received when ENARP=1
– Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.

Bit 4 **GENCALL**: General call address (Slave mode)
0: No General Call
1: General Call Address received when ENGC=1
– Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.

Bit 3 Reserved, must be kept at reset value

Bit 2 **TRA**: Transmitter/receiver

0: Data bytes received
1: Data bytes transmitted

This bit is set depending on the R/W bit of the address byte, at the end of total address phase.

It is also cleared by hardware after detection of Stop condition (STOPF=1), repeated Start condition, loss of bus arbitration (ARLO=1), or when PE=0.

Bit 1 **BUSY**: Bus busy

0: No communication on the bus
1: Communication ongoing on the bus

– Set by hardware on detection of SDA or SCL low

– cleared by hardware on detection of a Stop condition.

It indicates a communication in progress on the bus. This information is still updated when the interface is disabled (PE=0).

Bit 0 **MSL**: Master/slave

0: Slave Mode
1: Master Mode

– Set by hardware as soon as the interface is in Master mode (SB=1).

– Cleared by hardware after detecting a Stop condition on the bus or a loss of arbitration (ARLO=1), or by hardware when PE=0.

Note: Reading I2C_SR2 after reading I2C_SR1 clears the ADDR flag, even if the ADDR flag was set after reading I2C_SR1. Consequently, I2C_SR2 must be read only when ADDR is found set in I2C_SR1 or when the STOPF bit is cleared.

25.6.8 I^{2}C Clock control register (I2C_CCR)

Address offset: 0x1C
Reset value: 0x0000

Note: f_{PCLK1} must be at least 2 MHz to achieve standard mode I²C frequencies. It must be at least 4 MHz to achieve fast mode I²C frequencies. It must be a multiple of 10MHz to reach the 400 kHz maximum I²C fast mode clock.

The CCR register must be configured only when the I2C is disabled (PE = 0).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F/S	DUTY	Reserved		CCR[11:0]											
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 **F/S:** I2C master mode selection

0: Standard Mode I2C
1: Fast Mode I2C

Bit 14 **DUTY:** Fast mode duty cycle

0: Fast Mode t_{low}/t_{high} = 2
1: Fast Mode t_{low}/t_{high} = 16/9 (see CCR)

Bits 13:12 Reserved, must be kept at reset value

Bits 11:0 **CCR[11:0]:** Clock control register in Fast/Standard mode (Master mode)

Controls the SCL clock in master mode.

Standard mode or SMBus:

T_{high} = CCR * T_{PCLK1}
T_{low} = CCR * T_{PCLK1}

Fast mode:

If DUTY = 0:
T_{high} = CCR * T_{PCLK1}
T_{low} = 2 * CCR * T_{PCLK1}

If DUTY = 1: (to reach 400 kHz)
T_{high} = 9 * CCR * T_{PCLK1}
T_{low} = 16 * CCR * T_{PCLK1}

For instance: in standard mode, to generate a 100 kHz SCL frequency:

If FREQR = 08, T_{PCLK1} = 125 ns so CCR must be programmed with 0x28
(0x28 <=> 40d x 125 ns = 5000 ns.)

Note: 1. The minimum allowed value is 0x04, except in FAST DUTY mode where the minimum allowed value is 0x01
2. These timings are without filters.
3. The CCR register must be configured only when the I^2C is disabled (PE = 0).

25.6.9 I^2C TRISE register (I2C_TRISE)

Address offset: 0x20
Reset value: 0x0002

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

Bits 15:6 Reserved, must be kept at reset value

Bits 5:0 **TRISE[5:0]**: Maximum rise time in Fast/Standard mode (Master mode)

These bits must be programmed with the maximum SCL rise time given in the I^2C bus specification, incremented by 1.

For instance: in standard mode, the maximum allowed SCL rise time is 1000 ns.

If, in the I2C_CR2 register, the value of FREQ[5:0] bits is equal to 0x08 and T_{PCLK1} = 125 ns therefore the TRISE[5:0] bits must be programmed with 09h.

(1000 ns / 125 ns = 8 + 1)

The filter value can also be added to TRISE[5:0].

If the result is not an integer, TRISE[5:0] must be programmed with the integer part, in order to respect the t_{HIGH} parameter.

Note: TRISE[5:0] must be configured only when the I2C is disabled (PE = 0).

25.6.10 I^{2}C register map

The table below provides the I^{2}C register map and reset values.

Refer to *Table 2 on page 42* for the register boundary addresses table.

Table 126. I^{2}C register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	I2C_CR1	Reserved																SWRST	Reserved	ALERT	PEC	POS	ACK	STOP	START	NOSTRETCH	ENGC	ENPEC	ENARP	SMBTYPE	Reserved	SMBUS	PE
	Reset value																	0		0	0	0	0	0	0	0	0	0	0	0		0	0
0x04	I2C_CR2	Reserved																			LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	Reserved		FREQ[5:0]					
	Reset value																				0	0	0	0	0			0	0	0	0	0	0
0x08	I2C_OAR1	Reserved																ADDMODE	Reserved					ADD[9:8]		ADD[7:1]							ADD0
	Reset value																	0						0	0	0	0	0	0	0	0	0	0
0x0C	I2C_OAR2	Reserved																								ADD2[7:1]							ENDUAL
	Reset value																									0	0	0	0	0	0	0	0
0x10	I2C_DR	Reserved																								DR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x14	I2C_SR1	Reserved																SMBALERT	TIMEOUT	Reserved	PECERR	OVR	AF	ARLO	BERR	TxE	RxNE	Reserved	STOPF	ADD10	BTF	ADDR	SB
	Reset value																	0	0		0	0	0	0	0	0	0		0	0	0	0	0
0x18	I2C_SR2	Reserved																PEC[7:0]								DUALF	SMBHOST	SMBDEFAULT	GENCALL	Reserved	TRA	BUSY	MSL
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x1C	I2C_CCR	Reserved																F/S	DUTY	Reserved		CCR[11:0]											
	Reset value																	0	0			0	0	0	0	0	0	0	0	0	0	0	0

Table 126. I^2C 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
0x20	**I2C_TRISE**	Reserved																										TRISE[5:0]					
	Reset value																											0	0	0	0	1	0

26 Universal synchronous asynchronous receiver transmitter (USART)

26.1 USART introduction

The universal synchronous asynchronous receiver transmitter (USART) offers a flexible means of full-duplex data exchange with external equipment requiring an industry standard NRZ asynchronous serial data format. The USART offers a very wide range of baud rates using a fractional baud rate generator.

It supports synchronous one-way communication and half-duplex single wire communication. It also supports the LIN (local interconnection network), Smartcard Protocol and IrDA (infrared data association) SIR ENDEC specifications, and modem operations (CTS/RTS). It allows multiprocessor communication.

High speed data communication is possible by using the DMA for multibuffer configuration.

26.2 USART main features

- Full duplex, asynchronous communications
- NRZ standard format (Mark/Space)
- Configurable oversampling method by 16 or by 8 to give flexibility between speed and clock tolerance
- Fractional baud rate generator systems
  - Common programmable transmit and receive baud rate of up to 4 Mbit/s when the APB frequency is 32 MHz and oversampling is by 8
- Programmable data word length (8 or 9 bits)
- Configurable stop bits - support for 1 or 2 stop bits
- 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
- Transmitter clock output for synchronous transmission
- IrDA SIR encoder decoder
  - Support for 3/16 bit duration for normal mode
- Smartcard emulation capability
  - The Smartcard interface supports the asynchronous protocol Smartcards as defined in the ISO 7816-3 standards
  - 0.5, 1.5 stop bits for Smartcard operation
- Single-wire half-duplex communication
- Configurable multibuffer communication using DMA (direct memory access)
  - Buffering of received/transmitted bytes in reserved SRAM using centralized DMA
- Separate enable bits for transmitter and receiver
- Transfer detection flags:
  - Receive buffer full
  - Transmit buffer empty
  - 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
- Ten interrupt sources with flags:
  - CTS changes
  - LIN break detection
  - Transmit data register empty
  - Transmission complete

- Receive data register full
- Idle line received
- Overrun error
- Framing error
- Noise error
- Parity error
- Multiprocessor communication - enter into mute mode if address match does not occur
- Wake up from mute mode (by idle line detection or address mark detection)
- Two receiver wakeup modes: Address bit (MSB, 9th bit), Idle line

26.3 USART functional description

The interface is externally connected to another device by three pins (see *Figure 213*). Any USART bidirectional communication requires a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

RX: Receive Data Input is the serial data input. Oversampling techniques are used for data recovery by discriminating 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 nothing is to be transmitted, the TX pin is at high level. In single-wire and smartcard modes, this I/O is used to transmit and receive the data (at USART level, data are then received on SW_RX).

Through these pins, serial data is transmitted and received in normal USART mode as frames comprising:

- An Idle Line prior to transmission or reception
- A start bit
- A data word (8 or 9 bits) least significant bit first
- 0.5,1, 1.5, 2 Stop bits indicating that the frame is complete
- This interface uses a fractional baud rate generator - with a 12-bit mantissa and 4-bit fraction
- A status register (USART_SR)
- Data Register (USART_DR)
- A baud rate register (USART_BRR) - 12-bit mantissa and 4-bit fraction.
- A Guardtime Register (USART_GTPR) in case of Smartcard mode.

Refer to *Section 26.6: USART registers on page 690* for the definitions of each bit.

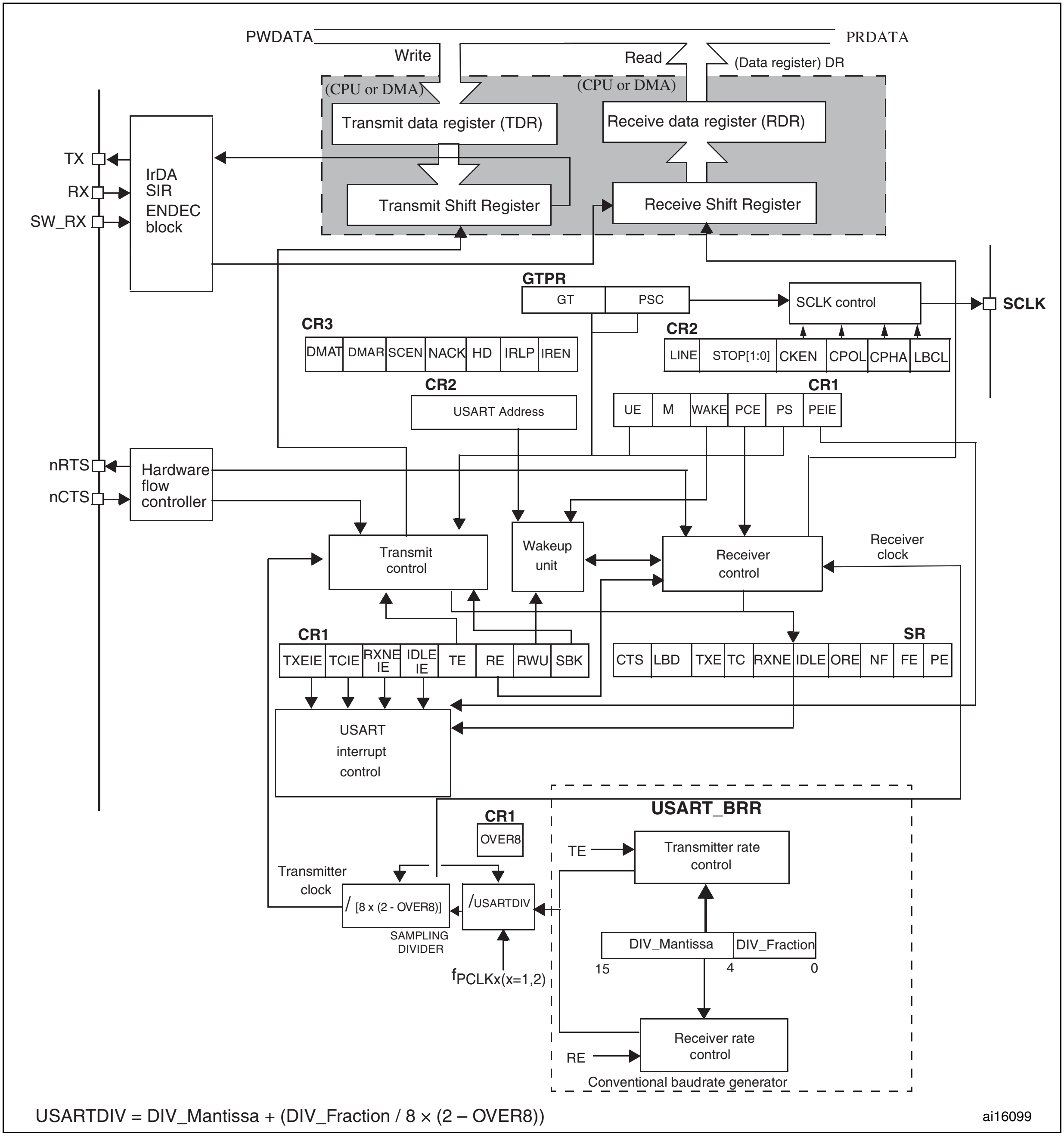
The following pin is required to interface in synchronous mode:

- **SCLK:** Transmitter clock output. 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. This can be used to control peripherals that have shift registers (e.g. LCD drivers). The clock phase and polarity are software programmable. In smartcard mode, SCLK can provide the clock to the smartcard.

The following pins are required in Hardware flow control mode:

- **nCTS:** Clear To Send blocks the data transmission at the end of the current transfer when high
- **nRTS:** Request to send indicates that the USART is ready to receive a data (when low).

Figure 213. USART block diagram



26.3.1 USART character description

Word length may be selected as being either 8 or 9 bits by programming the M bit in the USART_CR1 register (see *Figure 214*).

The TX pin is in low state during the start bit. It is in high state during the stop bit.

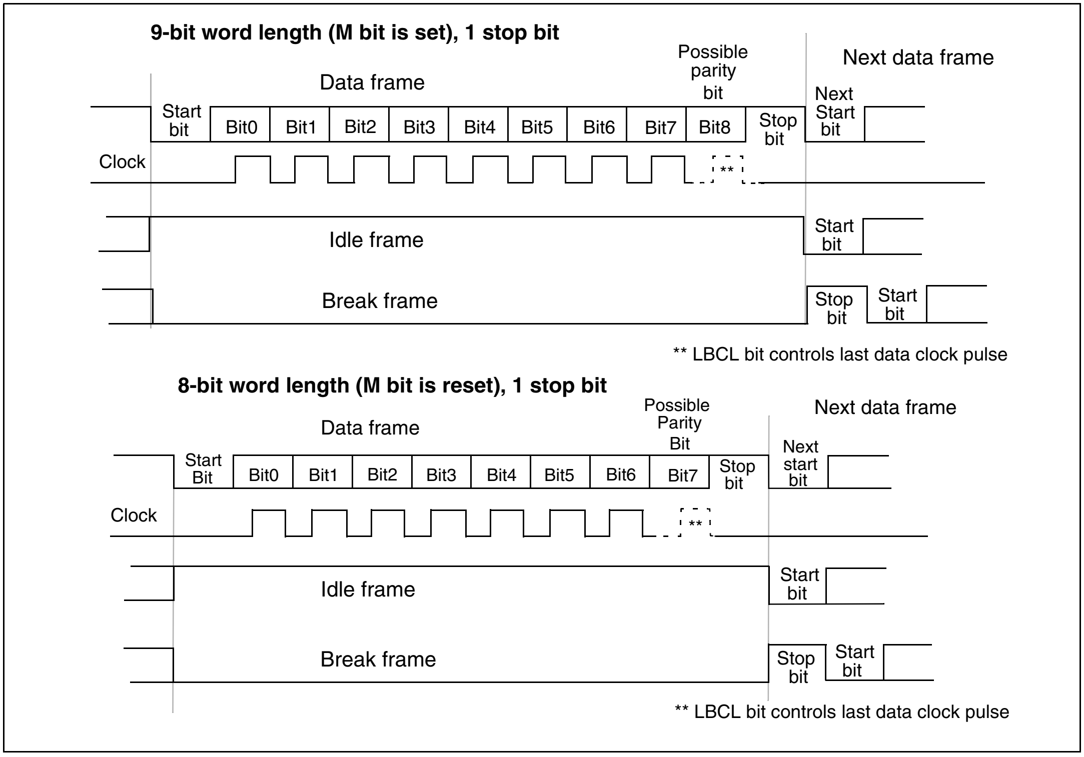
An ***Idle character*** is interpreted as an entire frame of “1”s followed by the start bit of the next frame which contains data (The number of “1” ‘s will include the number of stop bits).

A ***Break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame the transmitter inserts either 1 or 2 stop bits (logic “1” bit) to acknowledge the start bit.

Transmission and reception are driven by a common baud rate generator, the clock for each is generated when the enable bit is set respectively for the transmitter and receiver.

The details of each block is given below.

Figure 214. Word length programming



26.3.2 Transmitter

The transmitter can send data words of either 8 or 9 bits depending on the M bit status. When the transmit enable bit (TE) is set, the data in the transmit shift register is output on the TX pin and the corresponding clock pulses are output on the SCLK pin.

Character transmission

During an USART transmission, data shifts out least significant bit first on the TX pin. In this mode, the USART_DR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see *Figure 213*).

Every character is preceded by a start bit which is a logic level low for one bit period. The character is terminated by a configurable number of stop bits.

The following stop bits are supported by USART: 0.5, 1, 1.5 and 2 stop bits.

Note: The TE bit should not be reset during transmission of data. Resetting the TE bit during the transmission will corrupt the data on the TX pin as the baud rate counters will get frozen. The current data being transmitted will be lost.

An idle frame will be sent after the TE bit is enabled.

Configurable stop bits

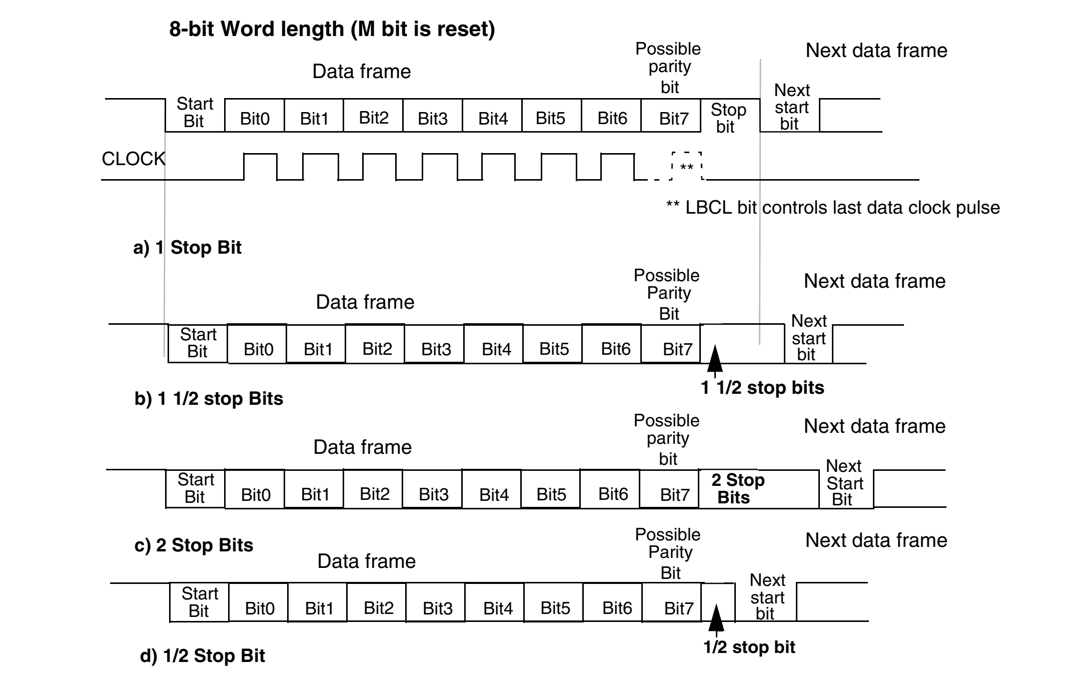
The number of stop bits to be transmitted with every character can be programmed in Control register 2, bits 13,12.

- ***1 stop bit***: This is the default value of number of stop bits.
- ***2 Stop bits***: This will be supported by normal USART, single-wire and modem modes.
- ***0.5 stop bit***: To be used when receiving data in Smartcard mode.
- ***1.5 stop bits***: To be used when transmitting and receiving data in Smartcard mode.

An idle frame transmission will include the stop bits.

A break transmission will be 10 low bits followed by the configured number of stop bits (when m = 0) and 11 low bits followed by the configured number of stop bits (when m = 1). It is not possible to transmit long breaks (break of length greater than 10/11 low bits).

Figure 215. Configurable stop bits



Procedure:

1. Enable the USART by writing the UE bit in USART_CR1 register to 1.
2. Program the M bit in USART_CR1 to define the word length.
3. Program the number of stop bits in USART_CR2.
4. Select DMA enable (DMAT) in USART_CR3 if Multi buffer Communication is to take place. Configure the DMA register as explained in multibuffer communication.
5. Select the desired baud rate using the USART_BRR register.
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_DR register (this clears the TXE bit). Repeat this for each data to be transmitted in case of single buffer.
8. After writing the last data into the USART_DR register, wait until TC=1. This indicates that the transmission of the last frame is complete. This is required for instance when the USART is disabled or enters the Halt mode to avoid corrupting the last transmission.

Single byte communication

Clearing the TXE bit is always performed by a write to the data register.

The TXE bit is set by hardware and it indicates:

- The data has been moved from TDR to the shift register and the data transmission has started.
- The TDR register is empty.
- The next data can be written in the USART_DR register without overwriting the previous data.

This flag generates an interrupt if the TXEIE bit is set.

When a transmission is taking place, a write instruction to the USART_DR register stores the data in the TDR register and which is copied in the shift register at the end of the current transmission.

When no transmission is taking place, a write instruction to the USART_DR register places the data directly in the shift register, the data transmission starts, and the TXE bit is immediately set.

If a frame is transmitted (after the stop bit) and the TXE bit is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the USART_CR1 register.

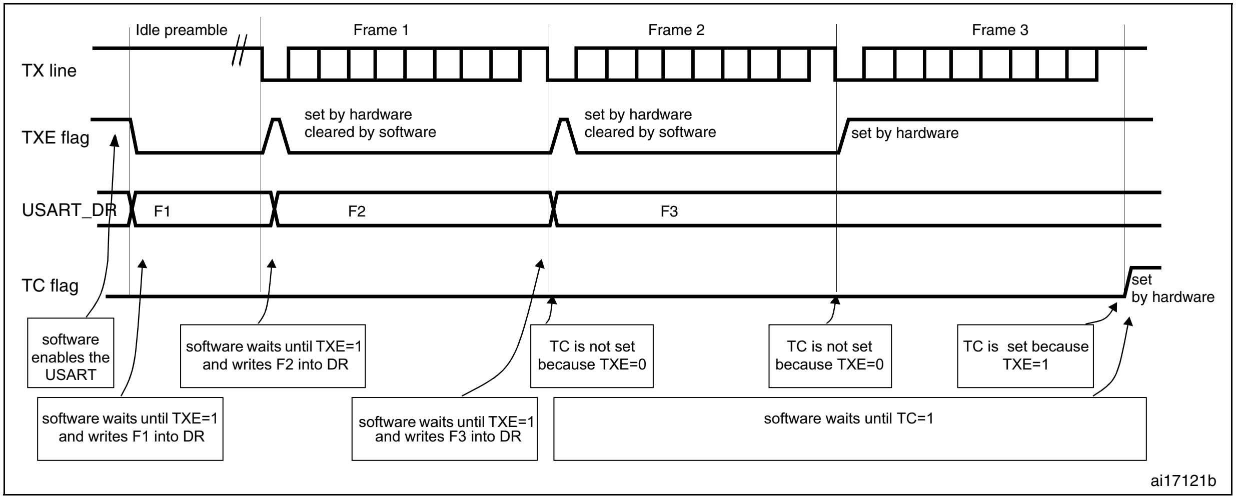
After writing the last data into the USART_DR register, it is mandatory to wait for TC=1 before disabling the USART or causing the microcontroller to enter the low power mode (see *Figure 216: TC/TXE behavior when transmitting*).

The TC bit is cleared by the following software sequence:

1. A read from the USART_SR register
2. A write to the USART_DR register

Note: The TC bit can also be cleared by writing a '0 to it. This clearing sequence is recommended only for Multibuffer communication.

Figure 216. TC/TXE behavior when transmitting


Idle preamble
Frame 1
Frame 2
Frame 3
TX line
TXE flag
set by hardware
cleared by software
set by hardware
cleared by software
set by hardware
USART_DR
F1
F2
F3
TC flag
set
by hardware
software enables the USART
software waits until TXE=1 and writes F2 into DR
TC is not set because TXE=0
TC is not set because TXE=0
TC is set because TXE=1
software waits until TXE=1 and writes F1 into DR
software waits until TXE=1 and writes F3 into DR
software waits until TC=1
ai17121b

Break characters

Setting the SBK bit transmits a break character. The break frame length depends on the M bit (see *Figure 214*).

If the SBK bit is set to '1 a break character is sent on the TX line after completing the current character transmission. This bit is reset by hardware when the break character is completed (during the stop bit of the break character). The USART inserts a logic 1 bit at the end of the last break frame to guarantee the recognition of the start bit of the next frame.

Note: If the software resets the SBK bit before the commencement of break transmission, the break character will not be transmitted. For two consecutive breaks, the SBK bit should be set after the stop bit of the previous break.

Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

26.3.3 Receiver

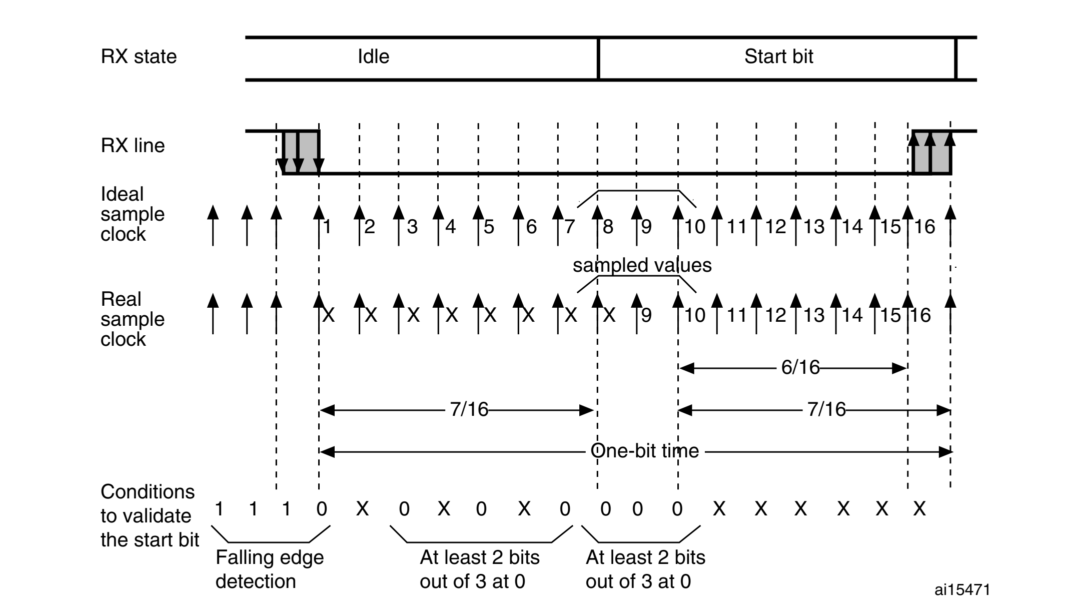
The USART can receive data words of either 8 or 9 bits depending on the M bit in the USART_CR1 register.

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 0 X 0 0 0 0.

Figure 217. Start bit detection when oversampling by 16 or 8


RX state
Idle
Start bit
RX line
Ideal sample clock
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
sampled values
Real sample clock
X X X X X X X X 9 10 11 12 13 14 15 16
6/16
7/16
7/16
One-bit time
Conditions to validate the start bit
1 1 1 0 X 0 X 0 X 0 0 0 0 X X X X X X
Falling edge detection
At least 2 bits out of 3 at 0
At least 2 bits out of 3 at 0
ai15471

Note: If the sequence is not complete, the start bit detection aborts and the receiver returns to the idle state (no flag is set) where it waits for a falling edge.

The start bit is confirmed (RXNE flag set, interrupt generated if RXNEIE=1) 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 (RXNE flag set, interrupt generated if RXNEIE=1) but the NE noise flag is set if, for both samplings, at least 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). If this condition is not met, the start detection aborts and the receiver returns to the idle state (no flag is set).

If, 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, the start bit is validated but the NE noise flag bit is set.

Character reception

During an USART reception, data shifts in least significant bit first through the RX pin. In this mode, the USART_DR register consists of a buffer (RDR) between the internal bus and the received shift register.

Procedure:

1. Enable the USART by writing the UE bit in USART_CR1 register to 1.
2. Program the M bit in USART_CR1 to define the word length.
3. Program the number of stop bits in USART_CR2.
4. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in multibuffer communication. STEP 3
5. Select the desired baud rate using the baud rate register USART_BRR
6. Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received

- 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).
- An interrupt is generated if the RXNEIE 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, RXNE is set after every byte received and is cleared by the DMA read to the Data Register.
- In single buffer mode, clearing the RXNE bit is performed by a software read to the USART_DR register. The RXNE flag can also be cleared by writing a zero to it. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.

Note: The RE bit should not be reset while receiving data. If the RE bit is disabled during reception, the reception of the current byte will be aborted.

Break character

When a break character is received, the USART handles it as a framing error.

Idle character

When an idle frame is detected, there is the same procedure as a data received character plus an interrupt if the IDLEIE bit is set.

Overrun error

An overrun error occurs when a character is received when RXNE has not been reset. Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared.

The RXNE flag is set after every byte received. An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

- The ORE bit is set.
- The RDR content will not be lost. The previous data is available when a read to USART_DR is performed.
- The shift register will be overwritten. After that point, any data received during overrun is lost.
- An interrupt is generated if either the RXNEIE bit is set or both the EIE and DMAR bits are set.
- The ORE bit is reset by a read to the USART_SR register followed by a USART_DR register read operation.

Note: The ORE bit, when set, indicates that at least 1 data has been lost. 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 it means that the last valid data has already been read and thus there is nothing 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. It may also occur when the new data is received during the reading sequence (between the USART_SR register read access and the USART_DR read access).

Selecting the proper oversampling method

The receiver implements different user-configurable oversampling techniques (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise.

The oversampling method can be selected by programming the OVER8 bit in the USART_CR1 register and can be either 16 or 8 times the baud rate clock (*Figure 218* and *Figure 219*).

Depending on the application:

- select oversampling by 8 (OVER8=1) to achieve higher speed (up to f_{PCLK}/8). In this case the maximum receiver tolerance to clock deviation is reduced (refer to *Section 26.3.5: USART receiver tolerance to clock deviation on page 672*)
- 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 f_{PCLK}/16

Programming the ONEBIT bit in the USART_CR3 register selects the method used to evaluate the logic level. There are two options:

- 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 NF bit is set
- a single sample in the center of the received bit

  Depending on the application:

  - select the three samples' majority vote method (ONEBIT=0) when operating in a noisy environment and reject the data when a noise is detected (refer to *Figure 127*) 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's tolerance to clock deviations (see *Section 26.3.5: USART*

receiver tolerance to clock deviation on page 672). In this case the NF bit will never be set.

When noise is detected in a frame:

- The NF bit is set at the rising edge of the RXNE bit.
- The invalid data is transferred from the Shift register to the USART_DR 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 will be issued if the EIE bit is set in the USART_CR3 register.

The NF bit is reset by a USART_SR register read operation followed by a USART_DR register read operation.

Note: 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 218. Data sampling when oversampling by 16

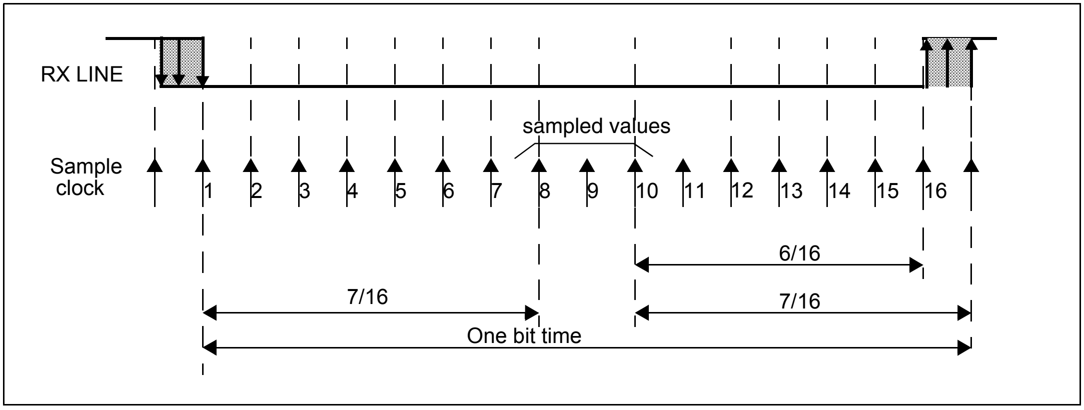


Figure 219. Data sampling when oversampling by 8

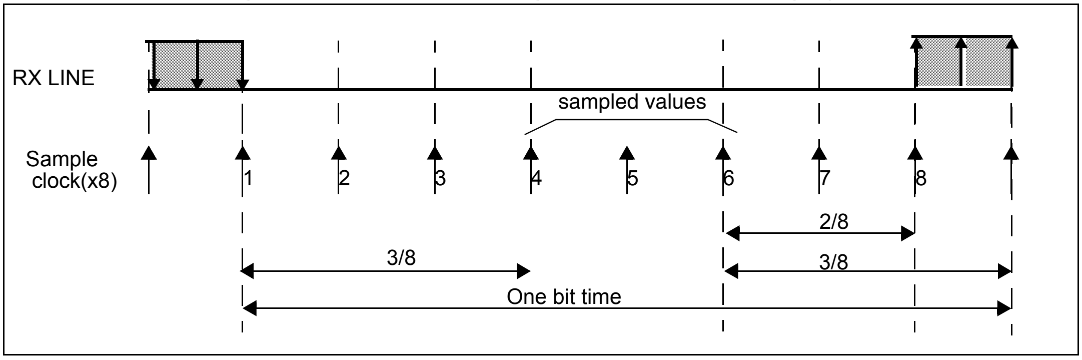


Table 127. Noise detection from sampled data

Sampled value	NE status	Received bit value
000	0	0
001	1	0
010	1	0
011	1	1

Table 127. Noise detection from sampled data (continued)

Sampled value	NE status	Received bit value
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_DR 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 will be issued if the EIE bit is set in the USART_CR3 register.

The FE bit is reset by a USART_SR register read operation followed by a USART_DR register read operation.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of Control Register 2 - it can be either 1 or 2 in normal mode and 0.5 or 1.5 in Smartcard mode.

1. ***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.
2. ***1 stop bit***: Sampling for 1 stop Bit is done on the 8th, 9th and 10th samples.
3. ***1.5 stop bits (Smartcard mode)***: When transmitting in smartcard mode, the device must check that the data is correctly sent. Thus the receiver block must be enabled (RE =1 in the USART_CR1 register) 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. Then, the FE flag is set with the RXNE 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 decomposed 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 26.3.11: Smartcard on page 681* for more details.
4. ***2 stop bits***: Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit. If a framing error is detected during the first stop bit the framing error flag will be set. The second stop bit is not checked for framing error. The RXNE flag will be set at the end of the first stop bit.

26.3.4 Fractional baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the same value as programmed in the Mantissa and Fraction values of USARTDIV.

Equation 1: Baud rate for standard USART (SPI mode included)

$$\text{Tx/Rx baud} = \frac{f_{CK}}{8 \times (2 - \text{OVER8}) \times \text{USARTDIV}}$$

Equation 2: Baud rate in Smartcard, LIN and IrDA modes

$$\text{Tx/Rx baud} = \frac{f_{CK}}{16 \times \text{USARTDIV}}$$

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

- When OVER8=0, the fractional part is coded on 4 bits and programmed by the DIV_fraction[3:0] bits in the USART_BRR register
- When OVER8=1, the fractional part is coded on 3 bits and programmed by the DIV_fraction[2:0] bits in the USART_BRR register, and bit DIV_fraction[3] must be kept cleared.

Note: The baud counters are updated to the new value in the baud registers after a write operation to USART_BRR. Hence the baud rate register value should not be changed during communication.

How to derive USARTDIV from USART_BRR register values when OVER8=0

Example 1:

If DIV_Mantissa = 0d27 and DIV_Fraction = 0d12 (USART_BRR = 0x1BC), then

Mantissa (USARTDIV) = 0d27

Fraction (USARTDIV) = 12/16 = 0d0.75

Therefore USARTDIV = 0d27.75

Example 2:

To program USARTDIV = 0d25.62

This leads to:

DIV_Fraction = 16*0d0.62 = 0d9.92

The nearest real number is 0d10 = 0xA

DIV_Mantissa = mantissa (0d25.620) = 0d25 = 0x19

Then, USART_BRR = 0x19A hence USARTDIV = 0d25.625

Example 3:

To program USARTDIV = 0d50.99

This leads to:

DIV_Fraction = 16*0d0.99 = 0d15.84

The nearest real number is 0d16 = 0x10 => overflow of DIV_frac[3:0] => carry must be added up to the mantissa

DIV_Mantissa = mantissa (0d50.990 + carry) = 0d51 = 0x33

Then, USART_BRR = 0x330 hence USARTDIV = 0d51.000

How to derive USARTDIV from USART_BRR register values when OVER8=1

Example 1:

If DIV_Mantissa = 0x27 and DIV_Fraction[2:0]= 0d6 (USART_BRR = 0x1B6), then

Mantissa (USARTDIV) = 0d27

Fraction (USARTDIV) = 6/8 = 0d0.75

Therefore USARTDIV = 0d27.75

Example 2:

To program USARTDIV = 0d25.62

This leads to:

DIV_Fraction = 8*0d0.62 = 0d4.96

The nearest real number is 0d5 = 0x5

DIV_Mantissa = mantissa (0d25.620) = 0d25 = 0x19

Then, USART_BRR = 0x195 => USARTDIV = 0d25.625

Example 3:

To program USARTDIV = 0d50.99

This leads to:

DIV_Fraction = 8*0d0.99 = 0d7.92

The nearest real number is 0d8 = 0x8 => overflow of the DIV_frac[2:0] => carry must be added up to the mantissa

DIV_Mantissa = mantissa (0d50.990 + carry) = 0d51 = 0x33

Then, USART_BRR = 0x0330 => USARTDIV = 0d51.000

Table 128. Error calculation for programmed baud rates at f_{PCLK} = 8 MHz or f_{PCLK} = 12 MHz, oversampling by 16[(1)]

Oversampling by 16 (OVER8=0)							
Baud rate7		f_{PCLK} = 8 MHz			f_{PCLK} = 12 MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate	Actual	Value programmed in the baud rate register	% Error
1	1.2 KBps	1.2 KBps	416.6875	0	1.2 KBps	625	0
2	2.4 KBps	2.4 KBps	208.3125	0.01	2.4 KBps	312.5	0

Table 128. Error calculation for programmed baud rates at f_{PCLK} = 8 MHz or f_{PCLK} = 12 MHz, oversampling by 16[(1)] (continued)

Oversampling by 16 (OVER8=0)							
Baud rate7		f_{PCLK} = 8 MHz			f_{PCLK} = 12 MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate	Actual	Value programmed in the baud rate register	% Error
3	9.6 KBps	9.604 KBps	52.0625	0.04	9.6 KBps	78.125	0
4	19.2 KBps	19.185 KBps	26.0625	0.08	19.2 KBps	39.0625	0
5	38.4 KBps	38.462 KBps	13	0.16	38.339 KBps	19.5625	0.16
6	57.6 KBps	57.554 KBps	8.6875	0.08	57.692 KBps	13	0.16
7	115.2 KBps	115.942 KBps	4.3125	0.64	115.385 KBps	6.5	0.16
8	230.4 KBps	228.571 KBps	2.1875	0.79	230.769 KBps	3.25	0.16
9	460.8 KBps	470.588 KBps	1.0625	2.12	461.538 KBps	1.625	0.16
10	921.6 KBps	NA	NA	NA	NA	NA	NA
11	2 MBps	NA	NA	NA	NA	NA	NA
12	3 MBps	NA	NA	NA	NA	NA	NA

1. The lower the CPU clock the lower the accuracy for a particular baud rate. The upper limit of the achievable baud rate can be fixed with these data.

Table 129. Error calculation for programmed baud rates at f_{PCLK} = 8 MHz or f_{PCLK} =12 MHz, oversampling by 8[(1)]

Oversampling by 8 (OVER8 = 1)							
Baud rate		f_{PCLK} = 8 MHz			f_{PCLK} = 12 MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate	Actual	Value programmed in the baud rate register	% Error
1	1.2 KBps	1.2 KBps	833.375	0	1.2 KBps	1250	0
2	2.4 KBps	2.4 KBps	416.625	0.01	2.4 KBps	625	0
3	9.6 KBps	9.604 KBps	104.125	0.04	9.6 KBps	156.25	0
4	19.2 KBps	19.185 KBps	52.125	0.08	19.2 KBps	78.125	0
5	38.4 KBps	38.462 KBps	26	0.16	38.339 KBps	39.125	0.16
6	57.6 KBps	57.554 KBps	17.375	0.08	57.692 KBps	26	0.16
7	115.2 KBps	115.942 KBps	8.625	0.64	115.385 KBps	13	0.16
8	230.4 KBps	228.571 KBps	4.375	0.79	230.769 KBps	6.5	0.16
9	460.8 KBps	470.588 KBps	2.125	2.12	461.538 KBps	3.25	0.16

Table 129. Error calculation for programmed baud rates at f_{PCLK} = 8 MHz or f_{PCLK} =12 MHz, oversampling by 8[1] (continued)

Oversampling by 8 (OVER8 = 1)							
Baud rate		f_{PCLK} = 8 MHz			f_{PCLK} = 12 MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate	Actual	Value programmed in the baud rate register	% Error
10	921.6 KBps	888.889 KBps	1.125	3.55	923.077 KBps	1.625	0.16
11	2 MBps	NA	NA	NA	NA	NA	NA
12	3 MBps	NA	NA	NA	NA	NA	NA

1. The lower the CPU clock the lower the accuracy for a particular baud rate. The upper limit of the achievable baud rate can be fixed with these data.

Table 130. Error calculation for programmed baud rates at f_{PCLK} = 16 MHz or f_{PCLK} = 24 MHz, oversampling by 16[1]

Oversampling by 16 (OVER8 = 0)							
Baud rate		f_{PCLK} = 16 MHz			f_{PCLK} = 24 MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate	Actual	Value programmed in the baud rate register	% Error
1	1.2 KBps	1.2 KBps	833.3125	0	1.2	1250	0
2	2.4 KBps	2.4 KBps	416.6875	0	2.4	625	0
3	9.6 KBps	9.598 KBps	104.1875	0.02	9.6	156.25	0
4	19.2 KBps	19.208 KBps	52.0625	0.04	19.2	78.125	0
5	38.4 KBps	38.369 KBps	26.0625	0.08	38.4	39.0625	0
6	57.6 KBps	57.554 KBps	17.375	0.08	57.554	26.0625	0.08
7	115.2 KBps	115.108 KBps	8.6875	0.08	115.385	13	0.16
8	230.4 KBps	231.884 KBps	4.3125	0.64	230.769	6.5	0.16
9	460.8 KBps	457.143 KBps	2.1875	0.79	461.538	3.25	0.16
10	921.6 KBps	941.176 KBps	1.0625	2.12	923.077	1.625	0.16
11	2 MBps	NA	NA	NA	NA	NA	NA
12	3 MBps	NA	NA	NA	NA	NA	NA

1. The lower the CPU clock the lower the accuracy for a particular baud rate. The upper limit of the achievable baud rate can be fixed with these data.

Table 131. Error calculation for programmed baud rates at f_{PCLK} = 16 MHz or f_{PCLK} = 24 MHz, oversampling by 8[1]

Oversampling by 8 (OVER8=1)							
Baud rate		f_{PCLK} = 16 MHz			f_{PCLK} = 24 MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired) B.rate / Desired B.rate	Actual	Value programmed in the baud rate register	% Error
1	1.2 KBps	1.2 KBps	1666.625	0	1.2 KBps	2500	0
2	2.4 KBps	2.4 KBps	833.375	0	2.4 KBps	1250	0
3	9.6 KBps	9.598 KBps	208.375	0.02	9.6 KBps	312.5	0
4	19.2 KBps	19.208 KBps	104.125	0.04	19.2 KBps	156.25	0
5	38.4 KBps	38.369 KBps	52.125	0.08	38.4 KBps	78.125	0
6	57.6 KBps	57.554 KBps	34.75	0.08	57.554 KBps	52.125	0.08
7	115.2 KBps	115.108 KBps	17.375	0.08	115.385 KBps	26	0.16
8	230.4 KBps	231.884 KBps	8.625	0.64	230.769 KBps	13	0.16
9	460.8 KBps	457.143 KBps	4.375	0.79	461.538 KBps	6.5	0.16
10	921.6 KBps	941.176 KBps	2.125	2.12	923.077 KBps	3.25	0.16
11	2 MBps	2000 KBps	1	0	2000 KBps	1.5	0
12	3 MBps	NA	NA	NA	3000 KBps	1	0

1. The lower the CPU clock the lower the accuracy for a particular baud rate. The upper limit of the achievable baud rate can be fixed with these data.

Table 132. Error calculation for programmed baud rates at f_{PCLK} = 1 MHz or f_{PCLK} = 8 MHz), oversampling by 16[1]

Oversampling by 16 (OVER8 = 0)							
Baud rate		f_{PCLK} = 1 MHz			f_{PCLK} = 8 MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired)B.Rate / Desired B.Rate	Actual	Value programmed in the baud rate register	% Error
1	1.2 KBps	1.2 KBps	52.0625	0.04	1.2 KBps	416.6875	0
2	2.4 KBps	2.398 KBps	26.0625	0.08	2.4 KBps	208.3125	0.01
3	9.6 KBps	9.615 KBps	6.5	0.16	9.604 KBps	52.0625	0.04
4	19.2 KBps	19.231 KBps	3.25	0.16	19.185 KBps	26.0625	0.08
5	38.4 KBps	38.462 KBps	1.625	0.16	38.462 KBps	13	0.16
6	57.6 KBps	58.824 KBps	1.0625	2.12	57.554 KBps	8.6875	0.08
7	115.2 KBps	NA	NA	NA	115.942 KBps	4.3125	0.64
8	230.4 KBps	NA	NA	NA	228.571 KBps	2.1875	0.79

Table 132. Error calculation for programmed baud rates at f_{PCLK} = 1 MHz or f_{PCLK} = 8 MHz), oversampling by 16[(1)] (continued)

Oversampling by 16 (OVER8 = 0)							
Baud rate		f_{PCLK} = 1 MHz			f_{PCLK} = 8 MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired)B.Rate / Desired B.Rate	Actual	Value programmed in the baud rate register	% Error
9	460.8 KBps	NA	NA	NA	470.588 KBps	1.0625	2.12
10	921.6 KBps	NA	NA	NA	NA	NA	NA
11	2 MBps	NA	NA	NA	NA	NA	NA
12	4 MBps	NA	NA	NA	NA	NA	NA

1. The lower the CPU clock the lower the accuracy for a particular baud rate. The upper limit of the achievable baud rate can be fixed with these data.

Table 133. Error calculation for programmed baud rates at f_{PCLK} = 1 MHz or f_{PCLK} = 8 MHz), oversampling by 8[(1)]

Oversampling by 8 (OVER8 = 1)							
Baud rate		f_{PCLK} = 1 MHz			f_{PCLK} = 8 MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired)B.Rate / Desired B.Rate	Actual	Value programmed in the baud rate register	% Error
1	1.2 KBps	1.2 KBps	104.125	0.04	1.2 KBps	833.375	0
2	2.4 KBps	2.398 KBps	52.125	0.08	2.4 KBps	416.625	0.01
3	9.6 KBps	9.615 KBps	13	-0.16	9.604 KBps	104.125	0.04
4	19.2 KBps	19.231 KBps	6.5	0.16	19.185 KBps	52.125	0.08
5	38.4 KBps	38.462 KBps	3.25	0.16	38.462 KBps	26	0.16
6	57.6 KBps	58.824 KBps	2.125	2.12	57.554 KBps	17.375	0.08
7	115.2 KBps	111.111 KBps	1.125	3.55	115.942 KBps	8.625	0.64
8	230.4 KBps	NA	NA	NA	228.571 KBps	4.375	0.79
9	460.8 KBps	NA	NA	NA	470.588 KBps	2.125	2.12
10	921.6 KBps	NA	NA	NA	888.889 KBps	1.125	3.55
11	2 MBps	NA	NA	NA	NA	NA	NA
12	4 MBps	NA	NA	NA	NA	NA	NA

1. The lower the CPU clock the lower the accuracy for a particular baud rate. The upper limit of the achievable baud rate can be fixed with these data.

Table 134. Error calculation for programmed baud rates at f_{PCLK} = 16 MHz or f_{PCLK} = 32 MHz), oversampling by 16[1]

Oversampling by 16 (OVER8 = 0)							
Baud rate		f_{PCLK} = 16 MHz			f_{PCLK} = 32 MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired)B.Rate / Desired B.Rate	Actual	Value programmed in the baud rate register	% Error
1	1.2 KBps	1.2 KBps	833.3125	0	1.2 KBps	1666.6875	0
2	2.4 KBps	2.4 KBps	416.6875	0	2.4 KBps	833.3125	0
3	9.6 KBps	9.598 KBps	104.1875	0.02	9.601 KBps	208.3125	0.01
4	19.2 KBps	19.208 KBps	52.0625	0.04	19.196 KBps	104.1875	0.02
5	38.4 KBps	38.369 KBps	26.0625	0.08	38.415 KBps	52.0625	0.04
6	57.6 KBps	57.554 KBps	17.375	0.08	57.554 KBps	34.75	0.08
7	115.2 KBps	115.108 KBps	8.6875	0.08	115.108 KBps	17.375	0.08
8	230.4 KBps	231.884 KBps	4.3125	0.64	230.216 KBps	8.6875	0.08
9	460.8 KBps	457.143 KBps	2.1875	0.79	463.768 KBps	4.3125	0.64
10	921.6 KBps	941.176 KBps	1.0625	2.12	914.286 KBps	2.1875	0.79
11	2 MBps	NA	NA	NA	2000 KBps	1	0
12	4 MBps	NA	NA	NA	NA	NA	NA

1. The lower the CPU clock the lower the accuracy for a particular baud rate. The upper limit of the achievable baud rate can be fixed with these data.

Table 135. Error calculation for programmed baud rates at f_{PCLK} = 16 MHz or f_{PCLK} = 32 MHz), oversampling by 8[1]

Oversampling by 8 (OVER8 = 1)							
Baud rate		f_{PCLK} = 16 MHz			f_{PCLK} = 32 MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired)B.Rate / Desired B.Rate	Actual	Value programmed in the baud rate register	% Error
1	1.2 KBps	1.2 KBps	1666.625	0	1.2 KBps	3333.375	0
2	2.4 KBps	2.4 KBps	833.375	0	2.4 KBps	1666.625	0
3	9.6 KBps	9.598 KBps	208.375	0.02	9.601 KBps	416.625	0.01
4	19.2 KBps	19.208 KBps	104.125	0.04	19.196 KBps	208.375	0.02
5	38.4 KBps	38.369 KBps	52.125	0.08	38.415 KBps	104.125	0.04
6	57.6 KBps	57.554 KBps	34.75	0.08	57.554 KBps	69.5	0.08
7	115.2 KBps	115.108 KBps	17.375	0.08	115.108 KBps	34.75	0.08
8	230.4 KBps	231.884 KBps	8.625	0.64	230.216 KBps	17.375	0.08

Table 135. Error calculation for programmed baud rates at f_{PCLK} = 16 MHz or f_{PCLK} = 32 MHz), oversampling by 8[1] (continued)

Oversampling by 8 (OVER8 = 1)							
Baud rate		f_{PCLK} = 16 MHz			f_{PCLK} = 32 MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired)B.Rate / Desired B.Rate	Actual	Value programmed in the baud rate register	% Error
9	460.8 KBps	457.143 KBps	4.375	0.79	463.768 KBps	8.625	0.64
10	921.6 KBps	941.176 KBps	2.125	2.12	914.286 KBps	4.375	0.79
11	2 MBps	2000 KBps	1	0	2000 KBps	2	0
12	4 MBps	NA	NA	NA	4000 KBps	1	0

1. The lower the CPU clock the lower the accuracy for a particular baud rate. The upper limit of the achievable baud rate can be fixed with these data.

Table 136. Error calculation for programmed baud rates at f_{PCLK} = 1 MHz or f_{PCLK} = 8 MHz), oversampling by 8[1]

Oversampling by 8 (OVER8 = 1)							
Baud rate		f_{PCLK} = 1 MHz			f_{PCLK} = 8 MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired)B.Rate / Desired B.Rate	Actual	Value programmed in the baud rate register	% Error
1	1.2 KBps	1.2 KBps	104.125	0.04	1.2 KBps	833.375	0
2	2.4 KBps	2.398 KBps	52.125	0.08	2.4 KBps	416.625	0.01
3	9.6 KBps	9.615 KBps	13	-0.16	9.604 KBps	104.125	0.04
4	19.2 KBps	19.231 KBps	6.5	0.16	19.185 KBps	52.125	0.08
5	38.4 KBps	38.462 KBps	3.25	0.16	38.462 KBps	26	0.16
6	57.6 KBps	58.824 KBps	2.125	2.12	57.554 KBps	17.375	0.08
7	115.2 KBps	111.111 KBps	1.125	3.55	115.942 KBps	8.625	0.64
8	230.4 KBps	NA	NA	NA	228.571 KBps	4.375	0.79
9	460.8 KBps	NA	NA	NA	470.588 KBps	2.125	2.12
10	921.6 KBps	NA	NA	NA	888.889 KBps	1.125	3.55
11	2 MBps	NA	NA	NA	NA	NA	NA
12	4 MBps	NA	NA	NA	NA	NA	NA

1. The lower the CPU clock the lower the accuracy for a particular baud rate. The upper limit of the achievable baud rate can be fixed with these data.

Table 137. Error calculation for programmed baud rates at f_{PCLK} = 16 MHz or f_{PCLK} = 32 MHz), oversampling by 16[1]

Oversampling by 16 (OVER8 = 0)							
Baud rate		f_{PCLK} = 16 MHz			f_{PCLK} = 32 MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired)B.Rate / Desired B.Rate	Actual	Value programmed in the baud rate register	% Error
1	1.2 KBps	1.2 KBps	833.3125	0	1.2 KBps	1666.6875	0
2	2.4 KBps	2.4 KBps	416.6875	0	2.4 KBps	833.3125	0
3	9.6 KBps	9.598 KBps	104.1875	0.02	9.601 KBps	208.3125	0.01
4	19.2 KBps	19.208 KBps	52.0625	0.04	19.196 KBps	104.1875	0.02
5	38.4 KBps	38.369 KBps	26.0625	0.08	38.415 KBps	52.0625	0.04
6	57.6 KBps	57.554 KBps	17.375	0.08	57.554 KBps	34.75	0.08
7	115.2 KBps	115.108 KBps	8.6875	0.08	115.108 KBps	17.375	0.08
8	230.4 KBps	231.884 KBps	4.3125	0.64	230.216 KBps	8.6875	0.08
9	460.8 KBps	457.143 KBps	2.1875	0.79	463.768 KBps	4.3125	0.64
10	921.6 KBps	941.176 KBps	1.0625	2.12	914.286 KBps	2.1875	0.79
11	2 MBps	NA	NA	NA	2000 KBps	1	0
12	4 MBps	NA	NA	NA	NA	NA	NA

1. The lower the CPU clock the lower the accuracy for a particular baud rate. The upper limit of the achievable baud rate can be fixed with these data.

Table 138. Error calculation for programmed baud rates at f_{PCLK} = 16 MHz or f_{PCLK} = 32 MHz), oversampling by 8[1]

Oversampling by 8 (OVER8 = 1)							
Baud rate		f_{PCLK} = 16 MHz			f_{PCLK} = 32 MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired)B.Rate / Desired B.Rate	Actual	Value programmed in the baud rate register	% Error
1	1.2 KBps	1.2 KBps	1666.625	0	1.2 KBps	3333.375	0
2	2.4 KBps	2.4 KBps	833.375	0	2.4 KBps	1666.625	0
3	9.6 KBps	9.598 KBps	208.375	0.02	9.601 KBps	416.625	0.01
4	19.2 KBps	19.208 KBps	104.125	0.04	19.196 KBps	208.375	0.02
5	38.4 KBps	38.369 KBps	52.125	0.08	38.415 KBps	104.125	0.04
6	57.6 KBps	57.554 KBps	34.75	0.08	57.554 KBps	69.5	0.08
7	115.2 KBps	115.108 KBps	17.375	0.08	115.108 KBps	34.75	0.08
8	230.4 KBps	231.884 KBps	8.625	0.64	230.216 KBps	17.375	0.08

Table 138. Error calculation for programmed baud rates at f_{PCLK} = 16 MHz or f_{PCLK} = 32 MHz), oversampling by 8[(1)] (continued)

Oversampling by 8 (OVER8 = 1)							
Baud rate		f_{PCLK} = 16 MHz			f_{PCLK} = 32 MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired)B.Rate / Desired B.Rate	Actual	Value programmed in the baud rate register	% Error
9	460.8 KBps	457.143 KBps	4.375	0.79	463.768 KBps	8.625	0.64
10	921.6 KBps	941.176 KBps	2.125	2.12	914.286 KBps	4.375	0.79
11	2 MBps	2000 KBps	1	0	2000 KBps	2	0
12	4 MBps	NA	NA	NA	4000 KBps	1	0

1. The lower the CPU clock the lower the accuracy for a particular baud rate. The upper limit of the achievable baud rate can be fixed with these data.

26.3.5 USART receiver tolerance to clock deviation

The USART asynchronous receiver works correctly only if the total clock system deviation is smaller than the USART receiver’s tolerance. 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’s local oscillator
- DTCL: Deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

DTRA + DQUANT + DREC + DTCL < USART receiver’s tolerance

The USART receiver’s tolerance to properly receive data is equal to the maximum tolerated deviation and depends on the following choices:

- 10- or 11-bit character length defined by the M bit in the USART_CR1 register
- oversampling by 8 or 16 defined by the OVER8 bit in the USART_CR1 register
- use of fractional baud rate or not
- 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 139. USART receiver’s tolerance when DIV fraction is 0

M bit	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
0	3.75%	4.375%	2.50%	3.75%
1	3.41%	3.97%	2.27%	3.41%

Table 140: USART receiver’s tolerance when DIV_Fraction is different from 0

M bit	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
0	3.33%	3.88%	2%	3%
1	3.03%	3.53%	1.82%	2.73%

Note: The figures specified in Table 139 and Table may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M=0 (11-bit times when M=1).

26.3.6 Multiprocessor communication

There is a possibility of performing multiprocessor communication with the USART (several USARTs connected in a network). For instance one of the USARTs can be the master, its TX output is connected to the RX input of the other USART. The others are slaves, 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 should actively receive the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non addressed devices may be placed in mute mode by means of the muting function. In mute mode:

- None of the reception status bits can be set.
- All the receive interrupts are inhibited.
- The RWU bit in USART_CR1 register is set to 1. RWU can be controlled automatically by hardware or written by the software 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.

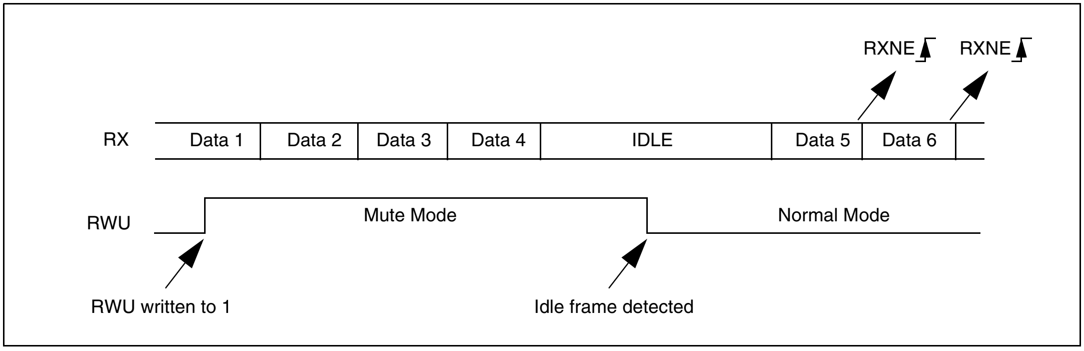
Idle line detection (WAKE=0)

The USART enters mute mode when the RWU bit is written to 1.

It wakes up when an Idle frame is detected. Then the RWU bit is cleared by hardware but the IDLE bit is not set in the USART_SR register. RWU can also be written to 0 by software.

An example of mute mode behavior using Idle line detection is given in Figure 220.

Figure 220. Mute mode using Idle line detection



Address mark detection (WAKE=1)

In this mode, bytes are recognized as addresses if their MSB is a ‘1 else they are considered as data. In an address byte, the address of the targeted receiver is put on the 4 LSB. This 4-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART_CR2 register.

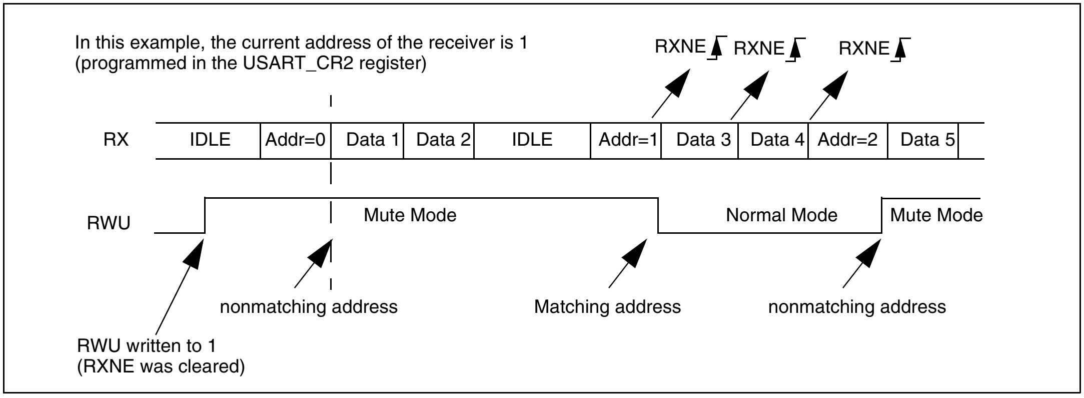
The USART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt nor DMA request is issued as the USART would have entered mute mode.

It 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 bit is set for the address character since the RWU bit has been cleared.

The RWU bit can be written to as 0 or 1 when the receiver buffer contains no data (RXNE=0 in the USART_SR register). Otherwise the write attempt is ignored.

An example of mute mode behavior using address mark detection is given in *Figure 221*.

Figure 221. Mute mode using address mark detection



26.3.7 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 bit, the possible USART frame formats are as listed in *Table* .

Table 141. Frame formats

M bit	PCE bit	USART frame[1]
0	0	| SB | 8 bit data | STB |
0	1	| SB | 7-bit data | PB | STB |
1	0	| SB | 9-bit data | STB |
1	1	| SB | 8-bit data PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit.

Even parity

The parity bit is calculated to obtain an even number of "1s" inside the frame made of the 7 or 8 LSB bits (depending on whether M is equal to 0 or 1) and the parity bit.

E.g.: data=00110101; 4 bits set => parity bit will be 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 7 or 8 LSB bits (depending on whether M is equal to 0 or 1) and the parity bit.

E.g.: data=00110101; 4 bits set => parity bit will be 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_SR register and an interrupt is generated if PEIE is set in the USART_CR1 register. The PE flag is cleared by a software sequence (a read from the status register followed by a read or write access to the USART_DR data register).

Note: *In case of wakeup by an address mark: the MSB bit of the data is taken into account to identify an address but not the parity bit. And the receiver does not check the parity of the address data (PE is not set in case of a parity error).*

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)).

Note: *The software routine that manages the transmission can activate the software sequence which clears the PE flag (a read from the status register followed by a read or write access to the data register). When operating in half-duplex mode, depending on the software, this can cause the PE flag to be unexpectedly cleared.*

26.3.8 LIN (local interconnection network) mode

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 same procedure explained in *Section 26.3.2* has to be applied for LIN Master transmission than 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 SBK bit sends 13 ‘0 bits as a break character. Then a bit of value ‘1 is sent to allow the next start detection.

LIN reception

A break detection circuit is implemented on the USART interface. 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 LBD flag is set in USART_SR. If the LBDIE bit=1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a ‘1 is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN=0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN=1), as soon as a framing error occurs (i.e. stop bit detected at ‘0, which will be the case for any break frame), the receiver stops until the break detection circuit receives either a ‘1, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown on the *Figure 222: Break detection in LIN mode (11-bit break length - LBDL bit is set) on page 677*.

Examples of break frames are given on *Figure 223: Break detection in LIN mode vs. Framing error detection on page 678*.

Figure 222. Break detection in LIN mode (11-bit break length - LBDL bit is set)

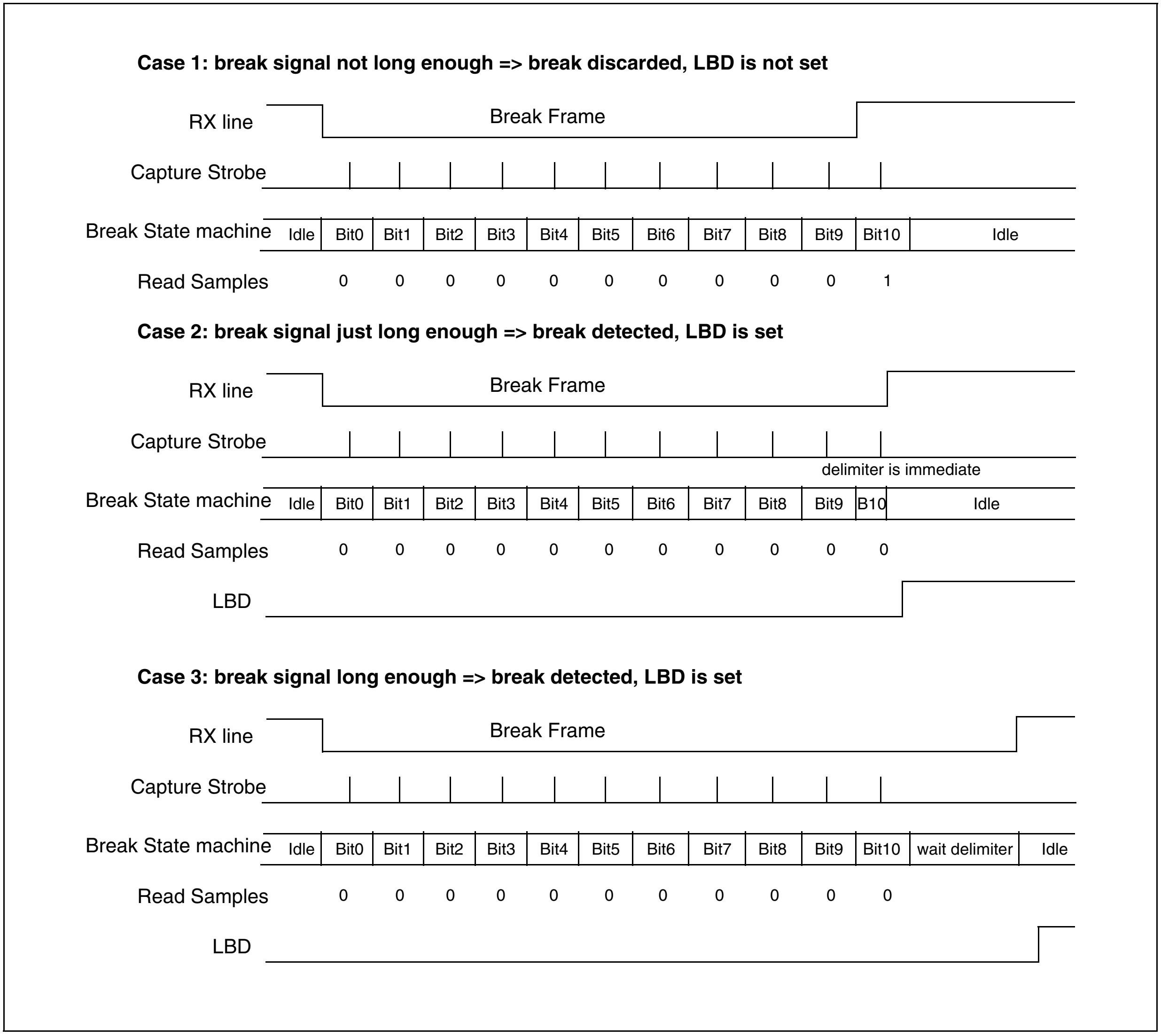
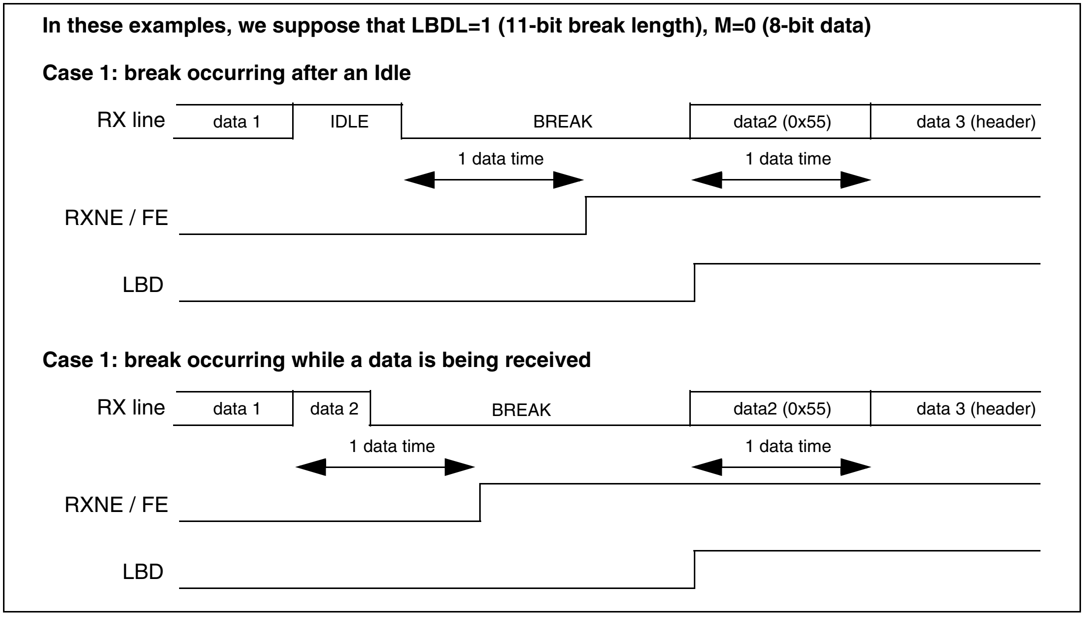


Figure 223. Break detection in LIN mode vs. Framing error detection


In these examples, we suppose that LBDL=1 (11-bit break length), M=0 (8-bit data)
Case 1: break occurring after an Idle
RX line
data 1
IDLE
BREAK
data2 (0x55)
data 3 (header)
1 data time
1 data time
RXNE / FE
LBD
Case 1: break occurring while a data is being received
RX line
data 1
data 2
BREAK
data2 (0x55)
data 3 (header)
1 data time
1 data time
RXNE / FE
LBD

26.3.9 USART synchronous mode

The synchronous mode is selected by writing 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.

The USART allows the user to control a bidirectional synchronous serial communications in master mode. The SCLK pin is the output of the USART transmitter clock. No clock pulses are sent to the SCLK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART_CR2 register clock pulses will or will not be generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register allows the user to select the clock polarity, and the CPHA bit in the USART_CR2 register allows the user to select the phase of the external clock (see *Figure 224*, *Figure 225* & *Figure 226*).

During the Idle state, preamble and send break, the external SCLK clock is not activated.

In synchronous mode the USART transmitter works exactly like in asynchronous mode. But as SCLK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In this mode the USART receiver works in a different manner compared to the asynchronous mode. If RE=1, the data is sampled on SCLK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).

Note: The SCLK pin works in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE=1) and a data is being transmitted (the data register USART_DR *has been written). This means that it is not possible to receive a synchronous data without transmitting data.*

The LBCL, CPOL and CPHA bits have to be selected when both the transmitter and the receiver are disabled (TE=RE=0) to ensure that the clock pulses function correctly. These bits should not be changed while the transmitter or the receiver is enabled.

It is advised that TE and RE are set in the same instruction in order to minimize the setup and the hold time of the receiver.

The USART supports master mode only: it cannot receive or send data related to an input clock (SCLK is always an output).

Figure 224. USART example of synchronous transmission

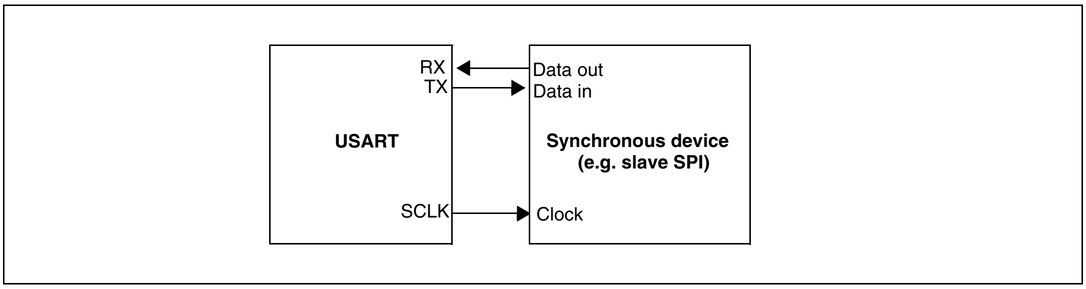


Figure 225. USART data clock timing diagram (M=0)

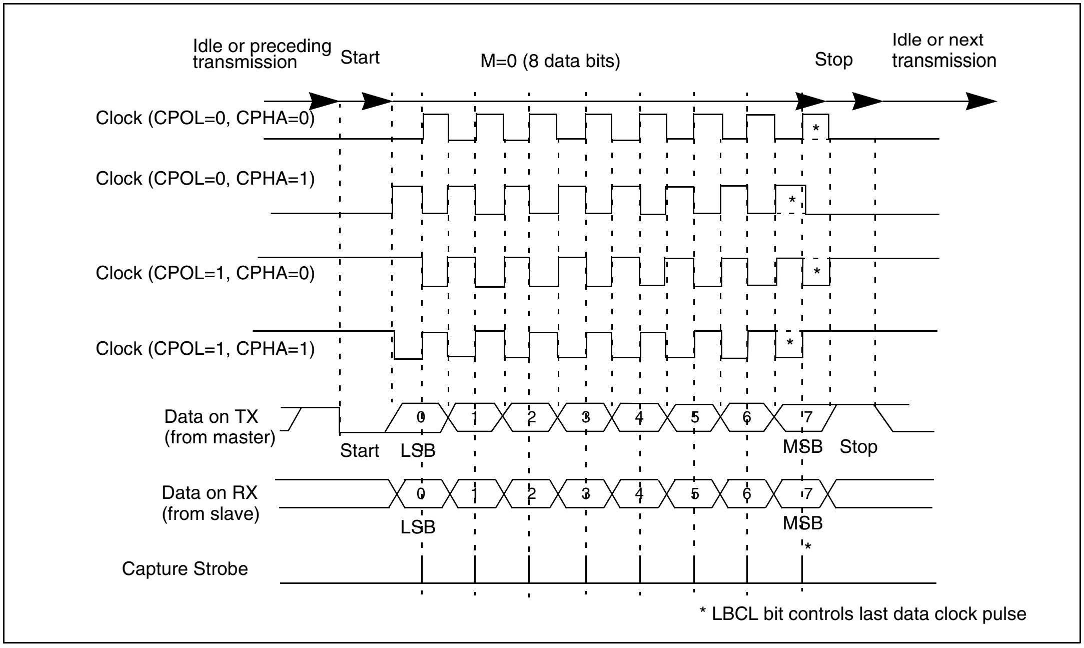


Figure 226. USART data clock timing diagram (M=1)

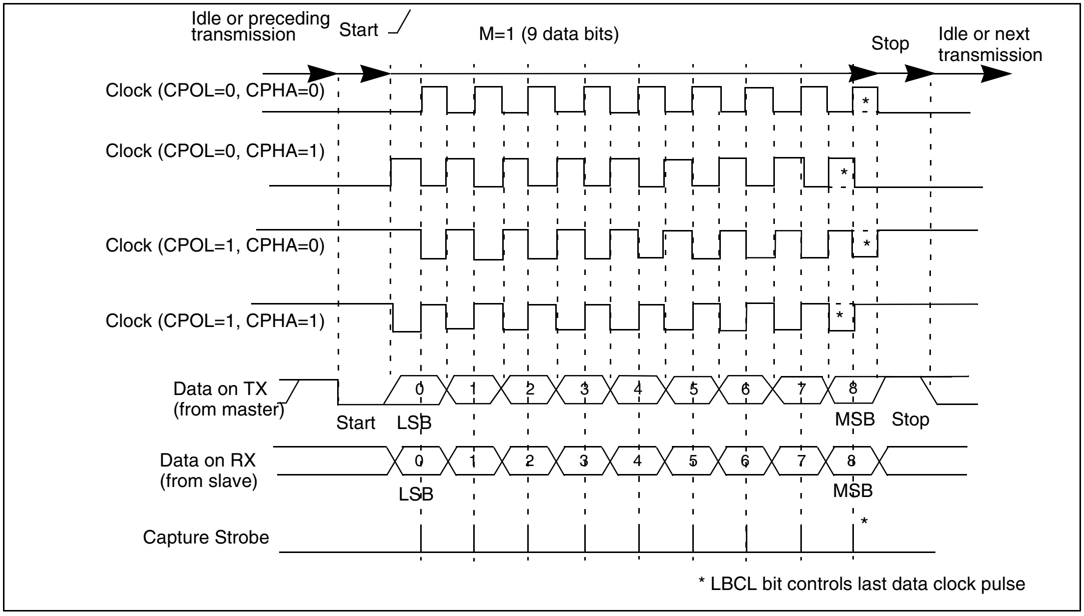
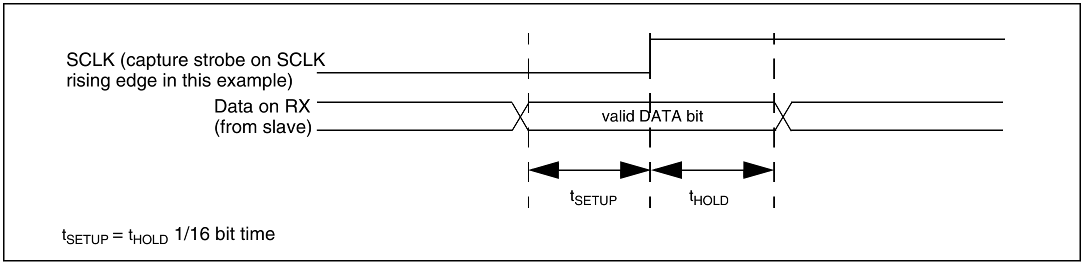


Figure 227. RX data setup/hold time


SCLK (capture strobe on SCLK rising edge in this example)
Data on RX (from slave)
valid DATA bit
t_{SETUP}
t_{HOLD}
$t_{SETUP} = t_{HOLD}$ 1/16 bit time

Note: The function of SCLK is different in Smartcard mode. Refer to the Smartcard mode chapter for more details.

26.3.10 Single-wire half-duplex communication

The 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 ‘HALF DUPLEX SEL’ (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 floating input (or output high open-drain) when not driven by the USART.

Apart from this, the communications are similar to what is done in normal USART mode. The conflicts on the line must be managed by the software (by the use of a centralized arbiter, for instance). In particular, the transmission is never blocked by hardware and continue to occur as soon as a data is written in the data register while the TE bit is set.

26.3.11 Smartcard

The 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.

Moreover, the CLKEN bit may be set in order to provide a clock to the smartcard.

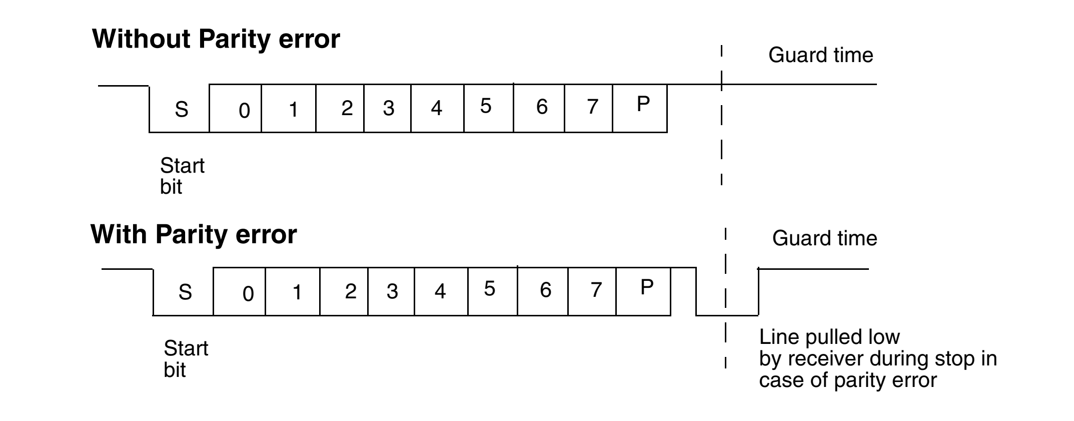
The Smartcard interface is designed to support asynchronous protocol Smartcards as defined in the ISO 7816-3 standard. The USART should be configured as:

- 8 bits plus parity: where M=1 and PCE=1 in the USART_CR1 register
- 1.5 stop bits when transmitting and receiving: where STOP=11 in the USART_CR2 register.

Note: It is also possible to choose 0.5 stop bit for receiving but it is recommended to use 1.5 stop bits for both transmitting and receiving to avoid switching between the two configurations.

Figure 228 shows examples of what can be seen on the data line with and without parity error.

Figure 228. 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 is 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 will start shifting on the next baud clock edge. In Smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- If a parity error is detected during reception of a frame programmed with a 0.5 or 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 USART has not been correctly received. This NACK signal (pulling transmit line low for 1 baud clock) will cause a framing error on the transmitter side

(configured with 1.5 stop bits). The application can handle re-sending of data according to the protocol. A parity error is 'NACK'ed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted.

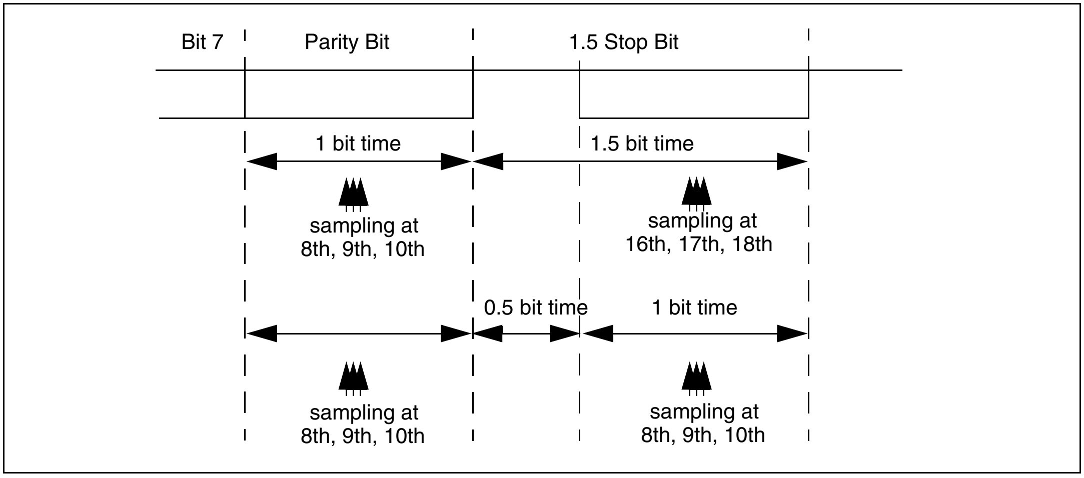
- The assertion of the TC flag can be delayed by programming the Guard Time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In Smartcard mode an empty transmit shift register triggers the guard time counter to count 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 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 will not be 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 will not detect the NACK as a start bit.

Note: A break character is not significant in Smartcard mode. A 0x00 data with a framing error will be 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 229 details how the NACK signal is sampled by the USART. In this example the USART is transmitting a 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 229. Parity error detection using the 1.5 stop bits



The USART can provide a clock to the smartcard through the SCLK output. In smartcard mode, SCLK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the prescaler register USART_GTPR. SCLK frequency can be programmed from $f_{CK}/2$ to $f_{CK}/62$, where f_{CK} is the peripheral input clock.

26.3.12 IrDA SIR ENDEC block

The IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the USART_CR2 register,
- SCEN and HDSEL bits in the USART_CR3 register.

The IrDA SIR physical layer specifies use of a Return to Zero, Inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see *Figure 230*).

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 115.2Kbps 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 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 (i.e. the USART is sending data to the IrDA encoder), any data on the IrDA receive line will be ignored by the IrDA decoder and if the Receiver is busy (USART is receiving decoded data from the USART), data on the TX from the USART to IrDA will not be encoded by IrDA. While receiving data, transmission should be avoided as the data to be transmitted could be corrupted.
- A ‘0 is transmitted as a high pulse and a ‘1 is transmitted as a ‘0. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see *Figure 231*).
- 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 us. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than 2 PSC periods (PSC is the prescaler value programmed in the IrDA low-power Baud Register, USART_GTPR). Pulses of width less than 1 PSC period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than 2 periods will be accepted as a pulse. The IrDA encoder/decoder doesn’t work when PSC=0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the STOP bits in the USART_CR2 register must be configured to “1 stop bit”.

IrDA low-power mode

Transmitter:

In low-power mode the pulse width is not maintained at 3/16 of the bit period. Instead, the width of the pulse is 3 times the low-power baud rate which can be a minimum of 1.42 MHz. Generally this value is 1.8432 MHz (1.42 MHz < PSC< 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.

Receiver:

Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART should discard pulses of duration shorter than 1/PSC. A valid low is accepted only if its duration is greater than 2 periods of the IrDA low-power Baud clock (PSC value in USART_GTPR).

Note: A pulse of width less than two and greater than one PSC period(s) may or may not be rejected.

The receiver set up time should be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception (IrDA is a half duplex protocol).

Figure 230. IrDA SIR ENDEC- block diagram

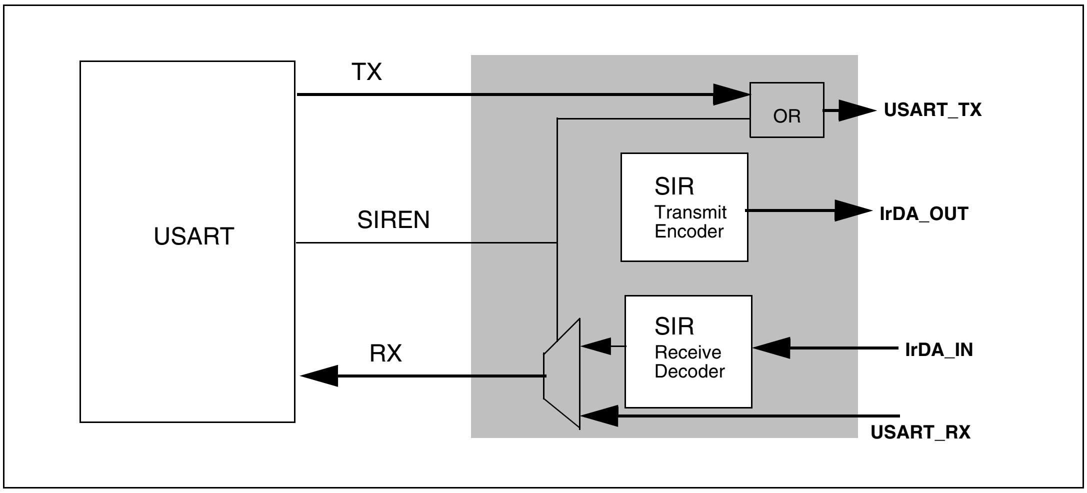
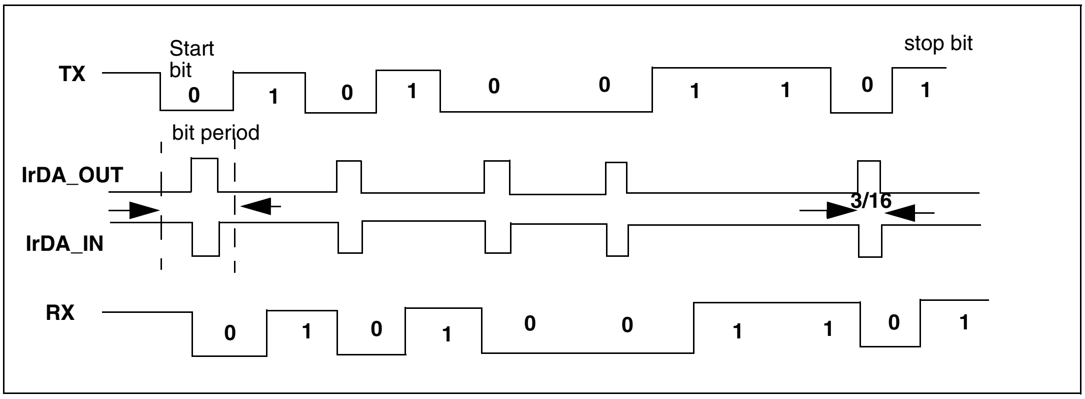


Figure 231. IrDA data modulation (3/16) -Normal mode



26.3.13 Continuous communication using DMA

The USART is capable of continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: You should refer to product specs for availability of the DMA controller. If DMA is not available in the product, you should use the USART as explained in Section 26.3.2 or 26.3.3. In the USART_SR register, you can clear the TXE/ RXNE flags to achieve continuous communication.

Transmission using DMA

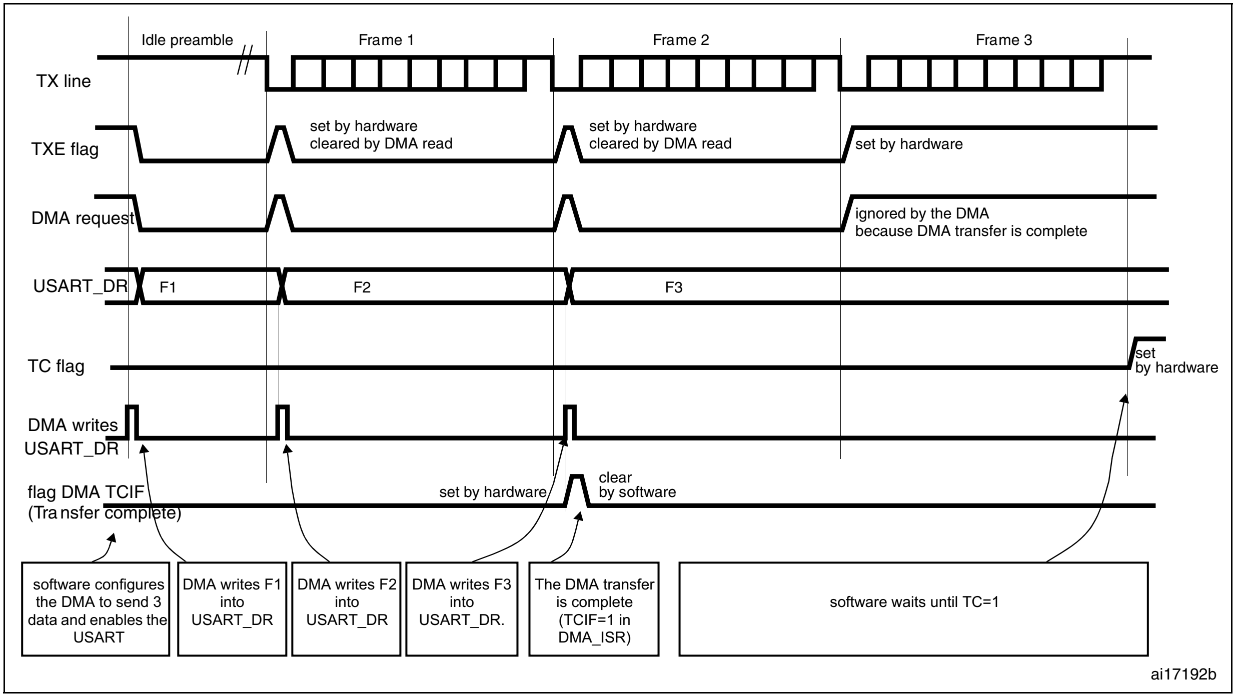
DMA mode can be enabled for transmission by setting DMAT bit in the USART_CR3 register. Data is loaded from a SRAM area configured using the DMA peripheral (refer to the DMA specification) to the USART_DR register whenever the TXE bit is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

1. Write the USART_DR register address in the DMA control register to configure it as the destination of the transfer. The data will be moved to this address from memory after each TXE event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data will be loaded into the USART_DR register from this memory area after each TXE 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 bit in the SR register by writing 0 to it.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the USART communication is complete. This is required to avoid corrupting the last transmission before disabling the USART or entering the Stop mode. The software must wait until TC=1. The TC flag remains cleared during all data transfers and it is set by hardware at the last frame’s end of transmission.

Figure 232. Transmission using DMA



Reception using DMA

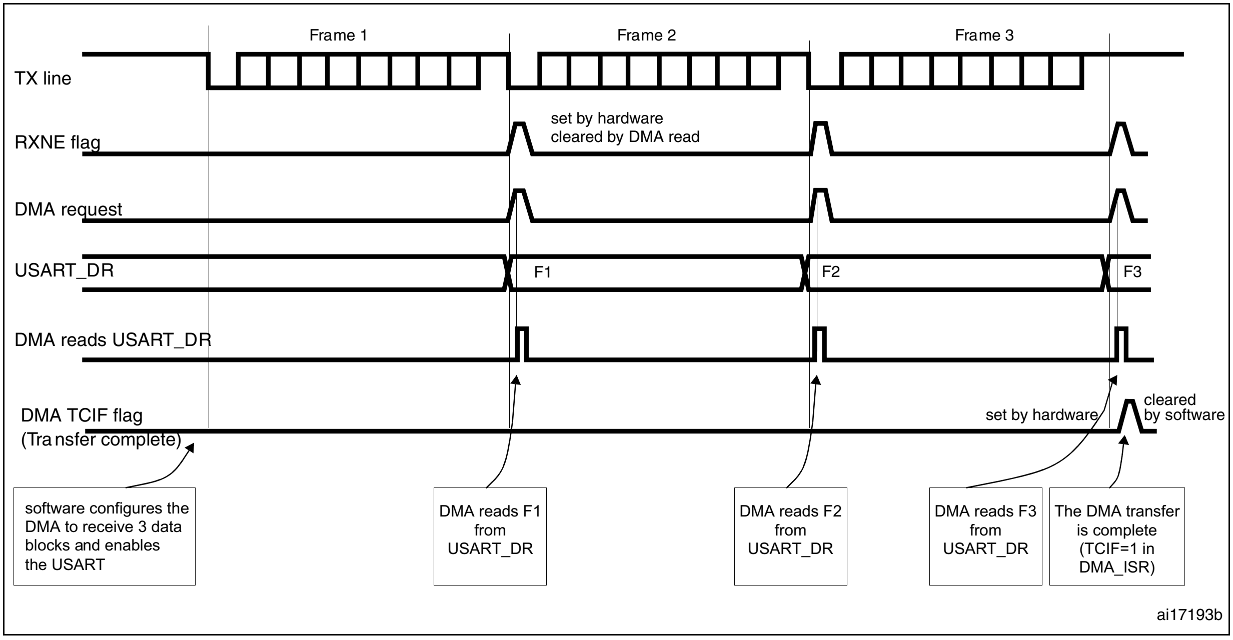
DMA mode can be enabled for reception by setting the DMAR bit in USART_CR3 register. Data is loaded from the USART_DR register to a SRAM area configured using the DMA peripheral (refer to the DMA specification) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_DR register address in the DMA control register to configure it as the source of the transfer. The data will be moved from this address to the memory after each RXNE event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data will be loaded from USART_DR to this memory area after each RXNE event.
3. Configure the total number of bytes to be transferred in 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. The DMAR bit should be cleared by software in the USART_CR3 register during the interrupt subroutine.

Note: If DMA is used for reception, do not enable the RXNEIE bit.

Figure 233. Reception using DMA


Frame 1
Frame 2
Frame 3
TX line
RXNE flag
set by hardware
cleared by DMA read
DMA request
USART_DR
F1
F2
F3
DMA reads USART_DR
DMA TCIF flag
(Transfer complete)
set by hardware
cleared
by software
software configures the DMA to receive 3 data blocks and enables the USART
DMA reads F1 from USART_DR
DMA reads F2 from USART_DR
DMA reads F3 from USART_DR
The DMA transfer is complete (TCIF=1 in DMA_ISR)
ai17193b

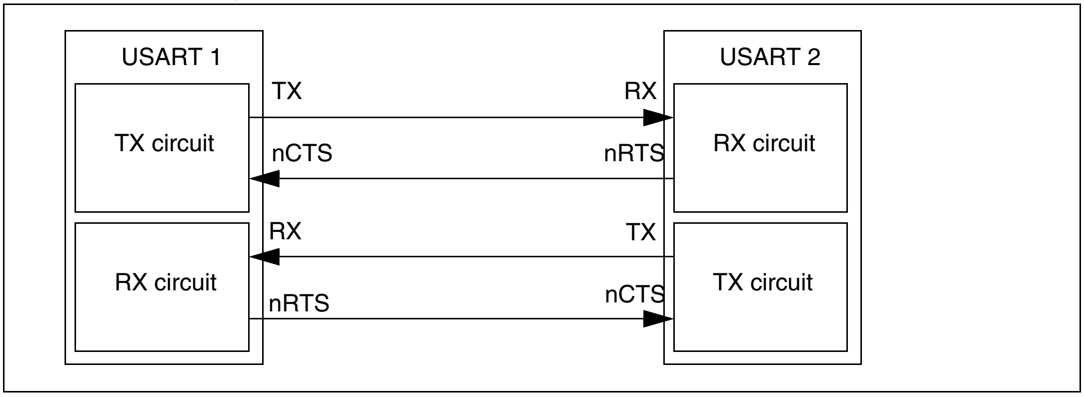
Error flagging and interrupt generation in multibuffer communication

In case of multibuffer communication if any error occurs during the transaction the error flag will be asserted after the current byte. An interrupt will be generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE in case of single byte reception, there will be separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which if set will issue an interrupt after the current byte with either of these errors.

26.3.14 Hardware flow control

It is possible to control the serial data flow between 2 devices by using the nCTS input and the nRTS output. The *Figure 234* shows how to connect 2 devices in this mode:

Figure 234. Hardware flow control between 2 USARTs

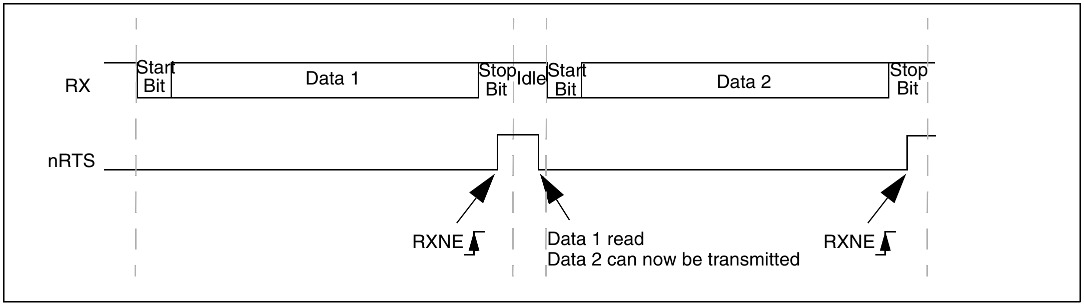


RTS and CTS flow control can be enabled independently by writing respectively RTSE and CTSE bits to 1 (in the USART_CR3 register).

RTS flow control

If the RTS flow control is enabled (RTSE=1), then nRTS is asserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, nRTS is deasserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 235* shows an example of communication with RTS flow control enabled.

Figure 235. RTS flow control

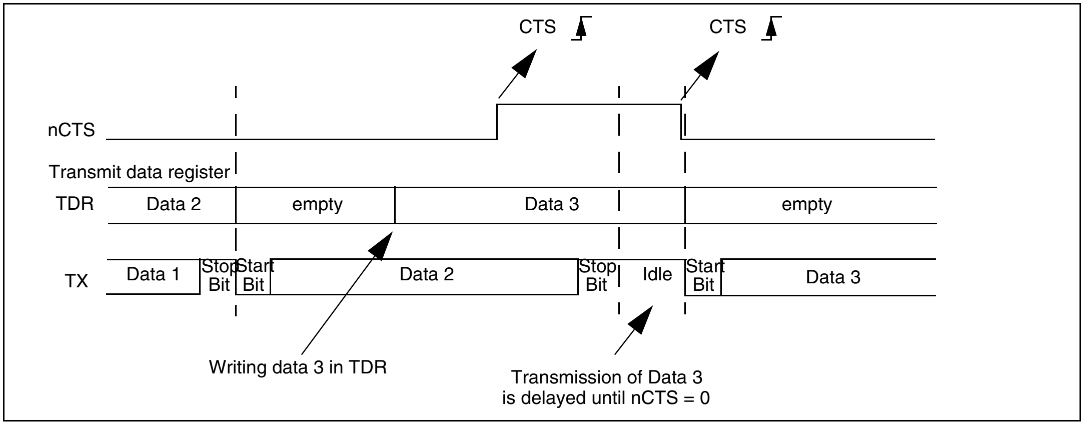


CTS flow control

If the CTS flow control is enabled (CTSE=1), then the transmitter checks the nCTS input before transmitting the next frame. If nCTS is asserted (tied low), then the next data is transmitted (assuming that a data is to be transmitted, in other words, if TXE=0), else the transmission does not occur. When nCTS is deasserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE=1, the CTSIF status bit is automatically set by hardware as soon as the nCTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. The figure below shows an example of communication with CTS flow control enabled.

Figure 236. CTS flow control



Note: ***Special behavior of break frames:*** *when the CTS flow is enabled, the transmitter does not check the nCTS input state to send a break.*

26.4 USART interrupts

Table 142. USART interrupt requests

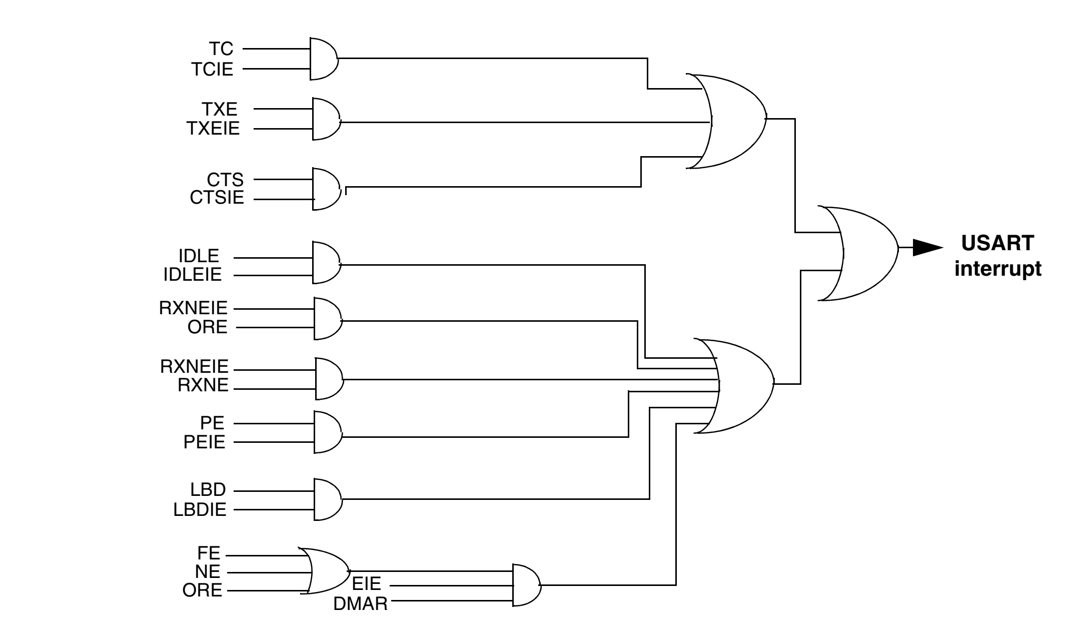
Interrupt event	Event flag	Enable control bit
Transmit Data Register Empty	TXE	TXEIE
CTS flag	CTS	CTSIE
Transmission Complete	TC	TCIE
Received Data Ready to be Read	RXNE	RXNEIE
Overrun Error Detected	ORE	
Idle Line Detected	IDLE	IDLEIE
Parity Error	PE	PEIE
Break Flag	LBD	LBDIE
Noise Flag, Overrun error and Framing Error in multibuffer communication	NF or ORE or FE	EIE

The USART interrupt events are connected to the same interrupt vector (see *Figure 237*).

- During transmission: Transmission Complete, Clear to Send or Transmit Data Register empty interrupt.
- While receiving: Idle Line detection, Overrun error, Receive Data register not empty, Parity error, LIN break detection, Noise Flag (only in multi buffer communication) and Framing Error (only in multi buffer communication).

These events generate an interrupt if the corresponding Enable Control Bit is set.

Figure 237. USART interrupt mapping diagram



26.5 USART mode configuration

Table 143. USART mode configuration[1]

USART modes	USART1	USART2	USART3	UART4	UART5
Asynchronous mode	X	X	X	X	X
Hardware flow control	X	X	X	NA	NA
Multibuffer communication (DMA)	X	X	X	X	X
Multiprocessor communication	X	X	X	X	X
Synchronous	X	X	X	NA	NA
Smartcard	X	X	X	NA	NA
Half-duplex (single-wire mode)	X	X	X	X	X
IrDA	X	X	X	X	X
LIN	X	X	X	X	X

1. X = supported; NA = not applicable.

26.6 USART registers

Refer to *Section 1.1 on page 37* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16 bits) or words (32 bits).

26.6.1 Status register (USART_SR)

Address offset: 0x00

Reset value: 0x00C0 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
						rc_w0	rc_w0	r	rc_w0	rc_w0	r	r	r	r	r

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

Bit 9 **CTS**: CTS flag

This bit is set by hardware when the nCTS input toggles, if the CTSE bit is set. It is cleared by software (by writing it to 0). An interrupt is generated if CTSIE=1 in the USART_CR3 register.

0: No change occurred on the nCTS status line

1: A change occurred on the nCTS status line

Bit 8 **LBD**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software (by writing it to 0). An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: An interrupt is generated when LBD=1 if LBDIE=1

Bit 7 **TXE**: Transmit data register empty

This bit is set by hardware when the content of the TDR register has been transferred into the shift register. An interrupt is generated if the TXEIE bit =1 in the USART_CR1 register. It is cleared by a write to the USART_DR register.

0: Data is not transferred to the shift register

1: Data is transferred to the shift register)

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit is set by hardware if the transmission of a frame containing data is complete and if TXE is set. An interrupt is generated if TCIE=1 in the USART_CR1 register. It is cleared by a software sequence (a read from the USART_SR register followed by a write to the USART_DR register). The TC bit can also be cleared by writing a '0' to it. This clearing sequence is recommended only for multibuffer communication.

0: Transmission is not complete

1: Transmission is complete

Bit 5 **RXNE**: Read data register not empty

This bit is set by hardware when the content of the RDR shift register has been transferred to the USART_DR register. An interrupt is generated if RXNEIE=1 in the USART_CR1 register. It is cleared by a read to the USART_DR register. The RXNE flag can also be cleared by writing a zero to it. This clearing sequence is recommended only for multibuffer communication.

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 the IDLEIE=1 in the USART_CR1 register. It is cleared by a software sequence (an read to the USART_SR register followed by a read to the USART_DR register).

0: No Idle Line is detected

1: Idle Line is detected

Note: The IDLE bit will not be set again until the RXNE bit has been set itself (i.e. a new idle line occurs).

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the word currently being received in the shift register is ready to be transferred into the RDR register while RXNE=1. An interrupt is generated if RXNEIE=1 in the USART_CR1 register. It is cleared by a software sequence (an read to the USART_SR register followed by a read to the USART_DR register).

0: No Overrun error

1: Overrun error is detected

Note: When this bit is set, the RDR register content will not be lost but the shift register will be overwritten. An interrupt is generated on ORE flag in case of Multi Buffer communication if the EIE bit is set.

Bit 2 **NF**: Noise detected flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by a software sequence (an read to the USART_SR register followed by a read to the USART_DR register).

0: No noise is detected

1: Noise is detected

Note: This bit does not generate interrupt as it appears at the same time as the RXNE bit which itself generates an interrupting interrupt is generated on NF flag in case of Multi Buffer communication if the EIE bit is set.

Note: When the line is noise-free, the NF flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 26.3.5: USART receiver tolerance to clock deviation on page 672).

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 a software sequence (an read to the USART_SR register followed by a read to the USART_DR register).

0: No Framing error is detected

1: Framing error or break character is detected

Note: This bit does not generate interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. If the word currently being transferred causes both frame error and overrun error, it will be transferred and only the ORE bit will be set.

An interrupt is generated on FE flag in case of Multi Buffer communication if the EIE bit is set.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by a software sequence (a read from the status register followed by a read or write access to the USART_DR data register). The software must wait for the RXNE flag to be set before clearing the PE bit.

An interrupt is generated if PEIE = 1 in the USART_CR1 register.

0: No parity error

1: Parity error

26.6.2 Data register (USART_DR)

Address offset: 0x04

Reset value: 0xXXXX XXXX

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

Bits 8:0 **DR[8:0]**: Data value

Contains the Received or Transmitted data character, depending on whether it is read from or written to.

The Data register performs a double function (read and write) since it is composed of two registers, one for transmission (TDR) and one for reception (RDR)

The TDR register provides the parallel interface between the internal bus and the output shift register (see Figure 1).

The RDR register provides the parallel interface between the input shift register and the internal bus.

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.

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

26.6.3 Baud rate register (USART_BRR)

Note: The baud counters stop counting if the TE or RE bits are disabled respectively.

Address offset: 0x08

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value

Bits 15:4 **DIV_Mantissa[11:0]**: mantissa of USARTDIV

These 12 bits define the mantissa of the USART Divider (USARTDIV)

Bits 3:0 **DIV_Fraction[3:0]**: fraction of USARTDIV

These 4 bits define the fraction of the USART Divider (USARTDIV). When OVER8=1, the DIV_Fraction3 bit is not considered and must be kept cleared.

26.6.4 Control register 1 (USART_CR1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	Reserved	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
rw	Res.	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value

Bit 15 **OVER8**: Oversampling mode

0: oversampling by 16
1: oversampling by 8

Note: Oversampling by 8 is not available in the Smartcard, IrDA and LIN modes: when SCEN=1,IREN=1 or LINEN=1 then OVER8 is forced to '0 by hardware.

Bit 14 Reserved, must be kept at reset value

Bit 13 **UE**: USART enable

When this bit is cleared the USART prescalers and outputs are stopped and the end of the current

byte transfer in order to reduce power consumption. This bit is set and cleared by software.

0: USART prescaler and outputs disabled

1: USART enabled

Bit 12 **M**: Word length

This bit determines the word length. It is set or cleared by software.

0: 1 Start bit, 8 Data bits, n Stop bit

1: 1 Start bit, 9 Data bits, n Stop bit

Note: The M bit must not be modified during a data transfer (both transmission and reception)

Bit 11 **WAKE**: Wakeup method

This bit determines the USART wakeup method, it is set or cleared by software.

0: Idle Line

1: Address Mark

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

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 will be selected after the current byte.

0: Even parity

1: Odd parity

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated whenever PE=1 in the USART_SR register

Bit 7 **TXEIE**: TXE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated whenever TXE=1 in the USART_SR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated whenever TC=1 in the USART_SR register

Bit 5 **RXNEIE**: RXNE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated whenever ORE=1 or RXNE=1 in the USART_SR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated whenever IDLE=1 in the USART_SR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: 1: During transmission, a “0” pulse on the TE bit (“0” followed by “1”) sends a preamble (idle line) after the current word, except in smartcard mode.

2: 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 **RWU**: Receiver wakeup

This bit determines if the USART is in mute mode or not. It is set and cleared by software and can be cleared by hardware when a wakeup sequence is recognized.

0: Receiver in active mode

1: Receiver in mute mode

Note: 1: Before selecting Mute mode (by setting the RWU bit) the USART must first receive a data byte, otherwise it cannot function in Mute mode with wakeup by Idle line detection.

2: In Address Mark Detection wakeup configuration (WAKE bit=1) the RWU bit cannot be modified by software while the RXNE bit is set.

Bit 0 **SBK**: Send break

This bit set is used to send break characters. It can be set and cleared by software. It should be set by software, and will be reset by hardware during the stop bit of break.

0: No break character is transmitted

1: Break character will be transmitted

26.6.5 Control register 2 (USART_CR2)

Address offset: 0x10

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	Res.	ADD[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

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

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 Synch Breaks (13 low bits) using the SBK bit in the USART_CR1 register, and to detect LIN Sync breaks.

Bits 13:12 **STOP**: 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 bit

Bit 11 **CLKEN**: Clock enable

This bit allows the user to enable the SCLK pin.

0: SCLK pin disabled

1: SCLK pin enabled

Bit 10 **CPOL**: Clock polarity

This bit allows the user to select the polarity of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on SCLK pin outside transmission window.

1: Steady high value on SCLK pin outside transmission window.

Bit 9 **CPHA**: Clock phase

This bit allows the user to select the phase of the clock output on the SCLK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see figures 225 to 226)

Note: 0: The first clock transition is the first data capture edge
1: The second clock transition is the first data capture edge

Bit 8 **LBCL**: Last bit clock pulse

This bit allows the user to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the SCLK pin in synchronous mode.

0: The clock pulse of the last data bit is not output to the SCLK pin

1: The clock pulse of the last data bit is output to the SCLK pin

1: The last bit is the 8th or 9th data bit transmitted depending on the 8 or 9 bit format selected by the M bit in the USART_CR1 register.

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 LBD=1 in the USART_SR register

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

Bit 4 Reserved, must be kept at reset value

Bits 3:0 **ADD[3:0]**: Address of the USART node

This bit-field gives the address of the USART node.
This is used in multiprocessor communication during mute mode, for wake up with address mark detection.

Note: These 3 bits (CPOL, CPHA, LBCL) should not be written while the transmitter is enabled.

26.6.6 Control register 3 (USART_CR3)

Address offset: 0x14

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value

Bit 11 **ONEBIT**: One sample bit method enable

This bit allows the user to select the sample method. When the one sample bit method is selected the noise detection flag (NF) is disabled.
0: Three sample bit method
1: One sample bit method

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited
1: An interrupt is generated whenever CTS=1 in the USART_SR register

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled
1: CTS mode enabled, data is only transmitted when the nCTS input is asserted (tied to 0). If the nCTS input is deasserted while a data is being transmitted, then the transmission is completed before stopping. If a data is written into the data register while nCTS is asserted, the transmission is postponed until nCTS is asserted.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled
1: RTS interrupt enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The nRTS output is asserted (tied to 0) when a data can be received.

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

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

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

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

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

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 NF=1 in the USART_SR register) in case of Multi Buffer Communication (DMAR=1 in the USART_CR3 register).

0: Interrupt is inhibited

1: An interrupt is generated whenever DMAR=1 in the USART_CR3 register and FE=1 or ORE=1 or NF=1 in the USART_SR register.

26.6.7 Guard time and prescaler register (USART_GTPR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
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 bit-field gives the Guard time value in terms of number of baud clocks.

This is used in Smartcard mode. The Transmission Complete flag is set after this guard time value.

Bits 7:0 **PSC[7:0]**: Prescaler value

– **In IrDA Low-power mode:**

PSC[7:0] = IrDA Low-Power Baud Rate
Used for programming the prescaler for dividing the system 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
...

– **In normal IrDA mode:** PSC must be set to 00000001.

– In smartcard mode:

PSC[4:0]: Prescaler value
Used for programming the prescaler for dividing the system 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
...

Note: 1: Bits [7:5] have no effect if Smartcard mode is used.

26.6.8 USART register map

The table below gives the USART register map and reset values.

Table 144. USART register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USART_SR	Reserved																						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
	Reset value																							0	0	1	1	0	0	0	0	0	0
0x04	USART_DR	Reserved																							DR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x08	USART_BRR	Reserved																DIV_Mantissa[15:4]												DIV_Fraction [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	USART_CR1	Reserved																OVER8	Reserved	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
	Reset value																	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USART_CR2	Reserved																	LINEN	STOP [1:0]		CLKEN	CPOL	CPHA	LBCL	Reserved	LBDIE	LBDL	Reserved	ADD[3:0]			
	Reset value																		0	0	0	0	0	0	0		0	0		0	0	0	0
0x14	USART_CR3	Reserved																				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
0x18	USART_GTPR	Reserved																GT[7:0]								PSC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Table 2 on page 42* for the register boundary addresses.

27 Serial peripheral interface (SPI)

27.1 SPI introduction

The SPI interface provides two main functions, supporting either the SPI protocol or the I^{2}S audio protocol. By default, it is the SPI function that is selected. It is possible to switch the interface from SPI to I^{2}S by software.
In low and medium density devices, the I^{2}S protocol is not available.

The serial peripheral interface (SPI) allows half/ full-duplex, synchronous, serial communication with external devices. The interface can be configured as the master and in this case it provides the communication clock (SCK) to the external slave device. The interface is also capable of operating in multimaster configuration.

It may be used for a variety of purposes, including simplex synchronous transfers on two lines with a possible bidirectional data line or reliable communication using CRC checking.

The I^{2}S is also a synchronous serial communication interface. It can address four different audio standards including the I^{2}S Philips standard, the MSB- and LSB-justified standards, and the PCM standard. It can operate as a slave or a master device in full-duplex mode (using 4 pins) or in half-duplex mode (using 6 pins). Master clock can be provided by the interface to an external slave component when the I^{2}S is configured as the communication master.

Warning: Since some SPI1 pins may be mapped onto some pins used by the JTAG interface (SPI1_NSS onto JTDI, SPI1_SCK onto JTDO and SPI1_MISO onto NJTRST), you may either:
– map SPI1 onto other pins
– disable the JTAG and use the SWD interface prior to configuring the pins listed as SPI IOs (when debugging the application) or
– disable both JTAG/SWD interfaces (for standalone applications).
For more information on the configuration of the JTAG/SWD interface pins, please refer to *Section 6.3.2: I/O pin multiplexer and mapping*.

27.2 SPI and I^{2}S main features

27.2.1 SPI features

- Full-duplex synchronous transfers on three lines
- Simplex synchronous transfers on two lines with or without a bidirectional data line
- 8- or 16-bit transfer frame format selection
- Master or slave operation
- Multimaster mode capability
- 8 master mode baud rate prescalers (f_{PCLK}/2 max.)
- Slave mode frequency (f_{PCLK}/2 max)
- Faster communication for both master and slave
- NSS management by hardware or software for both master and slave: dynamic change of master/slave operations
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Dedicated transmission and reception flags with interrupt capability
- SPI bus busy status flag
- SPI TI mode (in high and medium+ density devices)
- Hardware CRC feature for reliable communication:
  - CRC value can be transmitted as last byte in Tx mode
  - Automatic CRC error checking for last received byte
- Master mode fault, overrun and CRC error flags with interrupt capability
- 1-byte transmission and reception buffer with DMA capability: Tx and Rx requests

27.2.2 I^{2}S features

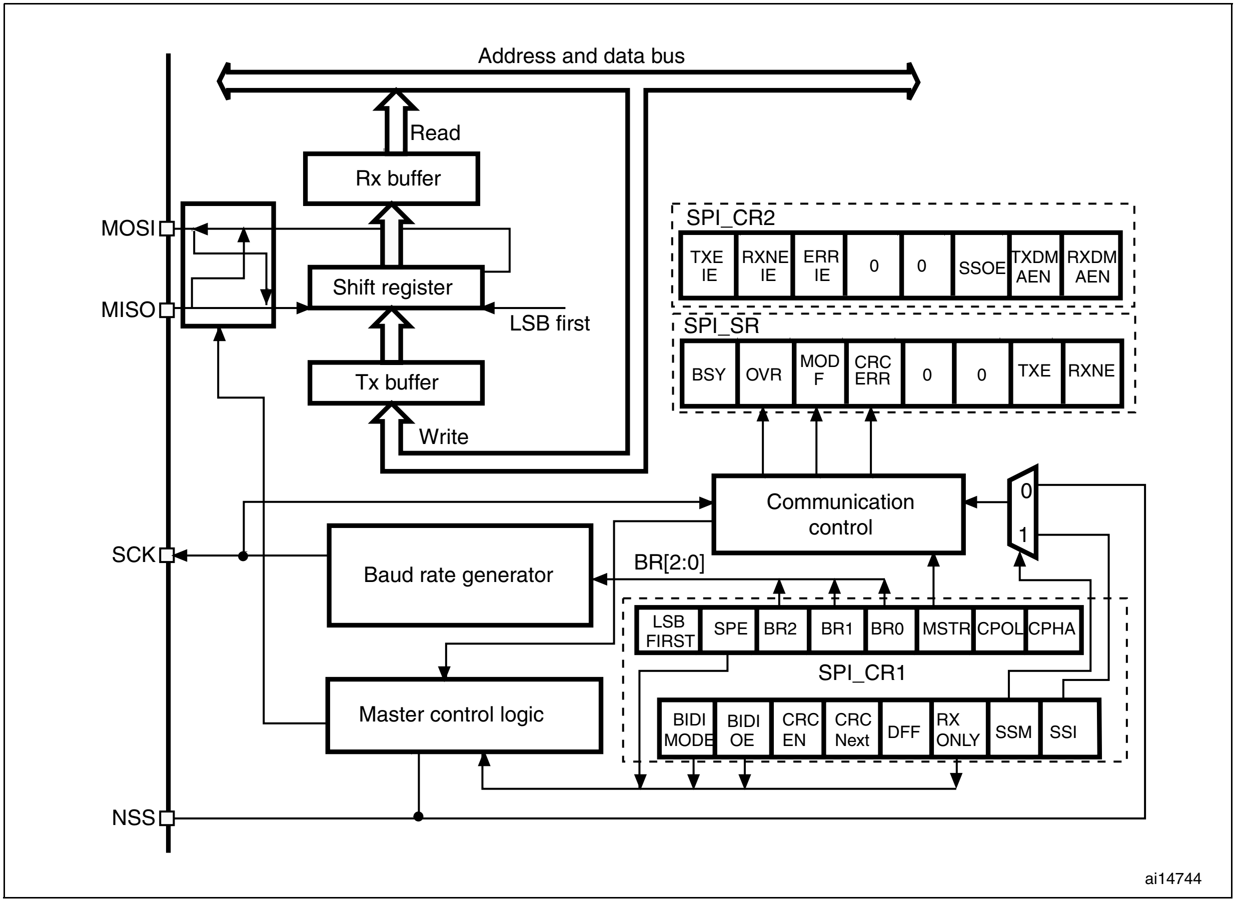
- Half-duplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler to reach accurate audio sample frequencies (from 8 kHz to 192 kHz)
- Data format may be 16-bit, 24-bit or 32-bit
- Packet frame is fixed to 16-bit (16-bit data frame) or 32-bit (16-bit, 24-bit, 32-bit data frame) by audio channel
- Programmable clock polarity (steady state)
- Underrun flag in slave transmission mode and Overrun flag in reception mode (master and slave)
- 16-bit register for transmission and reception with one data register for both channel sides
- Supported I^{2}S protocols:
  - I^{2}S Phillps standard
  - MSB-justified standard (left-justified)
  - LSB-justified standard (right-justified)
  - PCM standard (with short and long frame synchronization on 16-bit channel frame or 16-bit data frame extended to 32-bit channel frame)
- Data direction is always MSB first
- DMA capability for transmission and reception (16-bit wide)
- Master clock may be output to drive an external audio component. Ratio is fixed at 256 × F_S (where F_S is the audio sampling frequency)
- Frame error flag which detects a frame resynchronization between the external master device and the I2S slave device.

27.3 SPI functional description

27.3.1 General description

The block diagram of the SPI is shown in *Figure 238*.

Figure 238. SPI block diagram



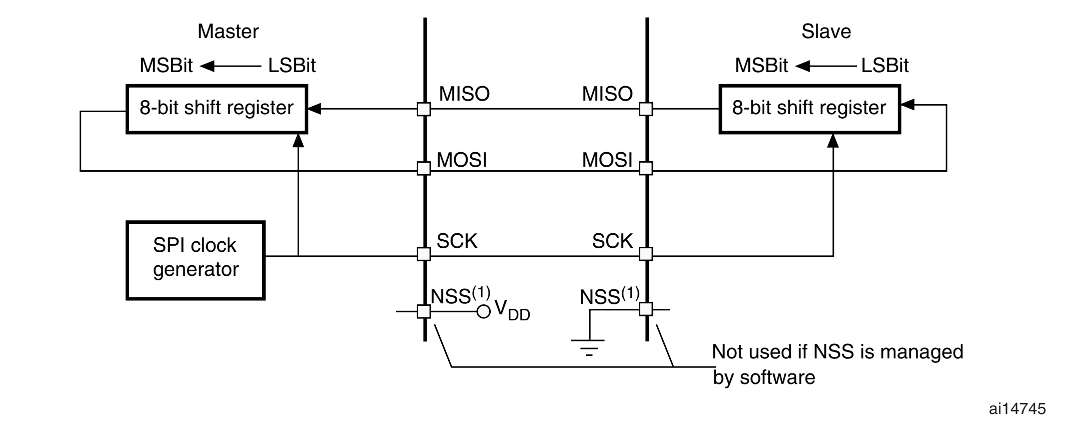
Usually, the SPI is connected to external devices through 4 pins:

- MISO: Master In / Slave Out data. This pin can be used to transmit data in slave mode and receive data in master mode.
- MOSI: Master Out / Slave In data. This pin can be used to transmit data in master mode and receive data in slave mode.
- SCK: Serial Clock output for SPI masters and input for SPI slaves.
- NSS: Slave select. This is an optional pin to select a slave device. This pin acts as a ‘chip select’ to let the SPI master communicate with slaves individually and to avoid contention on the data lines. Slave NSS inputs can be driven by standard IO ports on the master device. The NSS pin may also be used as an output if enabled (SSOE bit) and driven low if the SPI is in master configuration. In this manner, all NSS pins from devices connected to the Master NSS pin see a low level and become slaves when they are configured in NSS hardware mode. When configured in master mode with NSS configured as an input (MSTR=1 and SSOE=0) and if NSS is pulled low, the SPI

enters the master mode fault state: the MSTR bit is automatically cleared and the device is configured in slave mode (refer to *Section 27.3.10: Error flags on page 725*).

A basic example of interconnections between a single master and a single slave is illustrated in *Figure 239*.

Figure 239. Single master/ single slave application



1. Here, the NSS pin is configured as an input.

The MOSI pins are connected together and the MISO pins are connected together. In this way data is transferred serially between master and slave (most significant bit first).

The communication is always initiated by the master. When the master device transmits data to a slave device via the MOSI pin, the slave device responds via the MISO pin. This implies full-duplex communication with both data out and data in synchronized with the same clock signal (which is provided by the master device via the SCK pin).

Slave select (NSS) pin management

Hardware or software slave select management can be set using the SSM bit in the SPI_CR1 register.

- Software NSS management (SSM = 1)

  The slave select information is driven internally by the value of the SSI bit in the SPI_CR1 register. The external NSS pin remains free for other application uses.
- Hardware NSS management (SSM = 0)

  Two configurations are possible depending on the NSS output configuration (SSOE bit in register SPI_CR1).
  - NSS output enabled (SSM = 0, SSOE = 1)

    This configuration is used only when the device operates in master mode. The NSS signal is driven low when the master starts the communication and is kept low until the SPI is disabled.
  - NSS output disabled (SSM = 0, SSOE = 0)

    This configuration allows multimaster capability for devices operating in master mode. For devices set as slave, the NSS pin acts as a classical NSS input: the slave is selected when NSS is low and deselected when NSS high.

Clock phase and clock polarity

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPI_CR1 register. The CPOL (clock polarity) bit controls the steady 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 (clock phase) bit is set, the second edge on the SCK pin (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set) is the MSBit capture strobe. Data are latched on the occurrence of the second clock transition. If the CPHA bit is reset, the first edge on the SCK pin (falling edge if CPOL bit is set, rising edge if CPOL bit is reset) is the MSBit capture strobe. Data are latched on the occurrence of the first clock transition.

The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.

Figure 240, shows an SPI transfer with the four combinations of the CPHA and CPOL bits. The diagram may be interpreted as a master or slave timing diagram where the SCK pin, the MISO pin, the MOSI pin are directly connected between the master and the slave device.

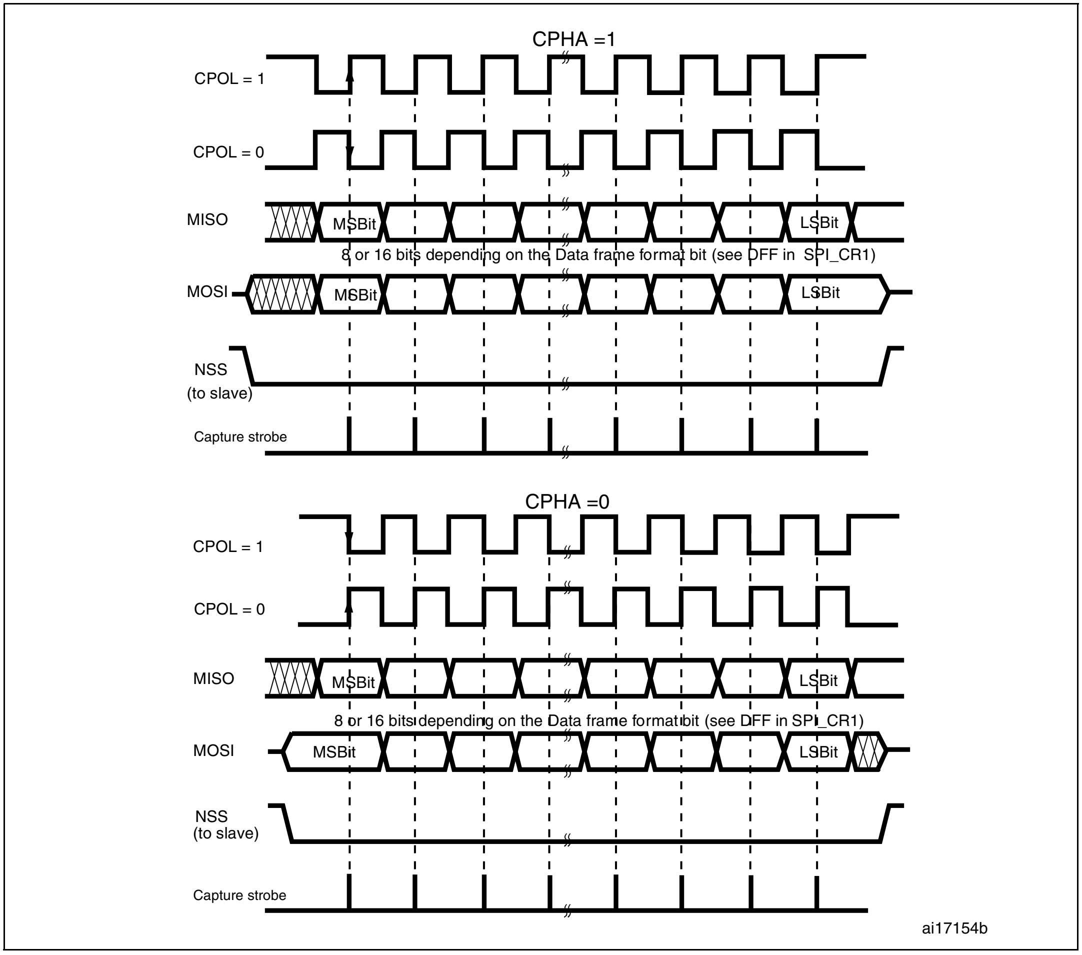
Note: Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit.

Master and slave must be programmed with the same timing mode.

The idle state of SCK must correspond to the polarity selected in the SPI_CR1 register (by pulling up SCK if CPOL=1 or pulling down SCK if CPOL=0).

The Data Frame Format (8- or 16-bit) is selected through the DFF bit in SPI_CR1 register, and determines the data length during transmission/reception.

Figure 240. Data clock timing diagram



1. These timings are shown with the LSBFIRST bit reset in the SPI_CR1 register.

Data frame format

Data can be shifted out either MSB-first or LSB-first depending on the value of the LSBFIRST bit in the SPI_CR1 Register.

Each data frame is 8 or 16 bits long depending on the size of the data programmed using the DFF bit in the SPI_CR1 register. The selected data frame format is applicable for transmission and/or reception.

27.3.2 Configuring the SPI in slave mode

In the slave configuration, the serial clock is received on the SCK pin from the master device. The value set in the BR[2:0] bits in the SPI_CR1 register, does not affect the data transfer rate.

Note: It is recommended to enable the SPI slave before the master sends the clock. If not, undesired data transmission might occur. The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication. It is mandatory to have the polarity of the communication clock set to the steady state value before the slave and the master are enabled.

Follow the procedure below to configure the SPI in slave mode:

Procedure

1. Set the DFF bit to define 8- or 16-bit data frame format
2. Select the CPOL and CPHA bits to define one of the four relationships between the data transfer and the serial clock (see *Figure 240*). For correct data transfer, the CPOL and CPHA bits must be configured in the same way in the slave device and the master device. This step is not required when the TI mode is selected through the FRF bit in the SPI_CR2 register.
3. The frame format (MSB-first or LSB-first depending on the value of the LSBFIRST bit in the SPI_CR1 register) must be the same as the master device. This step is not required when TI mode is selected.
4. In Hardware mode (refer to *Slave select (NSS) pin management on page 705*), the NSS pin must be connected to a low level signal during the complete byte transmit sequence. In NSS software mode, set the SSM bit and clear the SSI bit in the SPI_CR1 register. This step is not required when TI mode is selected.
5. Set the FRF bit in the SPI_CR2 register to select the TI mode protocol for serial communications.
6. Clear the MSTR bit and set the SPE bit (both in the SPI_CR1 register) to assign the pins to alternate functions.

In this configuration the MOSI pin is a data input and the MISO pin is a data output.

Transmit sequence

The data byte is parallel-loaded into the Tx buffer during a write cycle.

The transmit sequence begins when the slave device receives the clock signal and the most significant bit of the data on its MOSI pin. The remaining bits (the 7 bits in 8-bit data frame format, and the 15 bits in 16-bit data frame format) are loaded into the shift-register. The TXE flag in the SPI_SR register is set on the transfer of data from the Tx Buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPI_CR2 register is set.

Receive sequence

For the receiver, when data transfer is complete:

- The Data in shift register is transferred to Rx Buffer and the RXNE flag (SPI_SR register) is set
- An Interrupt is generated if the RXNEIE bit is set in the SPI_CR2 register.

After the last sampling clock edge the RXNE bit is set, a copy of the data byte received in the shift register is moved to the Rx buffer. When the SPI_DR register is read, the SPI peripheral returns this buffered value.

Clearing of the RXNE bit is performed by reading the SPI_DR register.

SPI TI protocol in slave mode

In slave mode, the SPI interface is compatible with the TI protocol. The FRF bit of the SPI_CR2 register can be used to configure the slave SPI serial communications to be compliant with this protocol.

The clock polarity and phase are forced to conform to the TI protocol requirements whatever the values set in the SPI_CR1 register. NSS management is also specific to the TI protocol which makes the configuration of NSS management through the SPI_CR1 and SPI_CR2 registers (such as SSM, SSI, SSOE) transparent for the user.

In Slave mode (*Figure 241: TI mode - Slave mode, single transfer* and *Figure 242: TI mode - Slave mode, continuous transfer*), the SPI baud rate prescaler is used to control the moment when the MISO pin state changes to HiZ. Any baud rate can be used thus allowing to determine this moment with optimal flexibility. However, the baud rate is generally set to the external master clock baud rate. The time for the MISO signal to become HiZ ($t_{release}$) depends on internal resynchronizations and on the baud rate value set in through BR[2:0] of SPI_CR1 register. It is given by the formula:

$$\frac{t_{baud\_rate}}{2} + 4 \times t_{pclk} < t_{release} < \frac{t_{baud\_rate}}{2} + 6 \times t_{pclk}$$

Note: This feature is not available for Motorola SPI communications (FRF bit set to 0).

To detect TI frame errors in Slave transmitter only mode by using the Error interrupt (ERRIE = 1), the SPI must be configured in 2-line unidirectional mode by setting BIDIMODE and BIDIOE to 1 in the SPI_CR1 register. When BIDIMODE is set to 0, OVR is set to 1 because the data register is never read and error interrupt are always generated, while when BIDIMODE is set to 1, data are not received and OVR is never set.

Figure 241. TI mode - Slave mode, single transfer

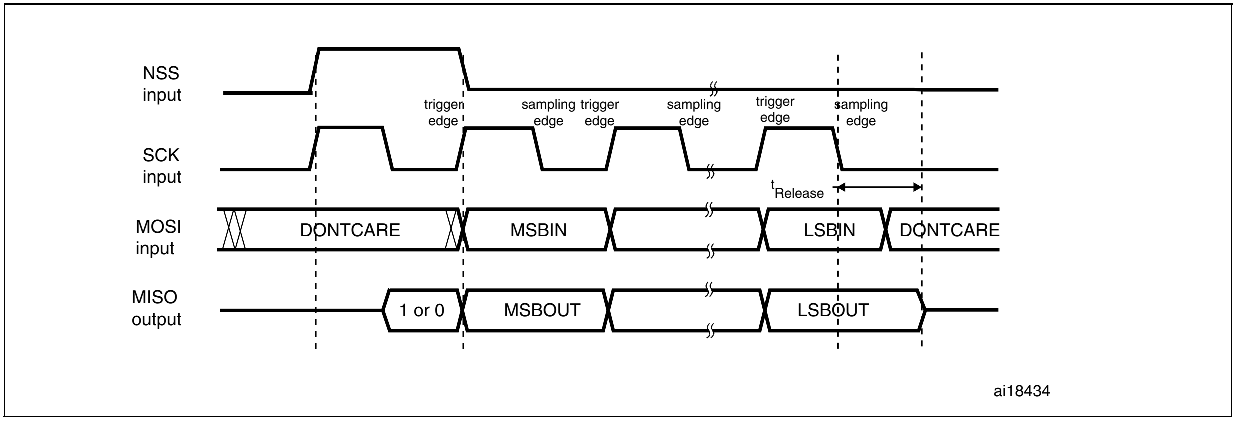
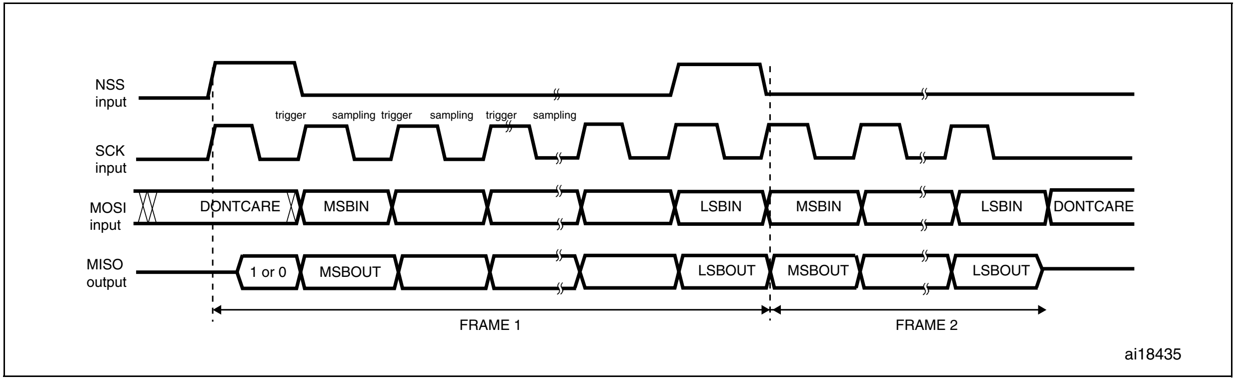


Figure 242. TI mode - Slave mode, continuous transfer



27.3.3 Configuring the SPI in master mode

In the master configuration, the serial clock is generated on the SCK pin.

Procedure

1. Select the BR[2:0] bits to define the serial clock baud rate (see SPI_CR1 register).
2. Select the CPOL and CPHA bits to define one of the four relationships between the data transfer and the serial clock (see *Figure 240*). This step is not required when the TI mode is selected.
3. Set the DFF bit to define 8- or 16-bit data frame format
4. Configure the LSBFIRST bit in the SPI_CR1 register to define the frame format. This step is not required when the TI mode is selected.
5. If the NSS pin is required in input mode, in hardware mode, connect the NSS pin to a high-level signal during the complete byte transmit sequence. In NSS software mode, set the SSM and SSI bits in the SPI_CR1 register. If the NSS pin is required in output mode, the SSOE bit only should be set. This step is not required when the TI mode is selected.
6. Set the FRF bit in SPI_CR2 to select the TI protocol for serial communications.
7. The MSTR and SPE bits must be set (they remain set only if the NSS pin is connected to a high-level signal).

In this configuration the MOSI pin is a data output and the MISO pin is a data input.

Transmit sequence

The transmit sequence begins when a byte is written in the Tx Buffer.

The data byte is parallel-loaded into the shift register (from the internal bus) during the first bit transmission and then shifted out serially to the MOSI pin MSB first or LSB first depending on the LSBFIRST bit in the SPI_CR1 register. The TXE flag is set on the transfer of data from the Tx Buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPI_CR2 register is set.

Receive sequence

For the receiver, when data transfer is complete:

- The data in the shift register is transferred to the RX Buffer and the RXNE flag is set
- An interrupt is generated if the RXNEIE bit is set in the SPI_CR2 register

At the last sampling clock edge the RXNE bit is set, a copy of the data byte received in the shift register is moved to the Rx buffer. When the SPI_DR register is read, the SPI peripheral returns this buffered value.

Clearing the RXNE bit is performed by reading the SPI_DR register.

A continuous transmit stream can be maintained if the next data to be transmitted is put in the Tx buffer once the transmission is started. Note that TXE flag should be ‘1 before any attempt to write the Tx buffer is made.

Note: When a master is communicating with SPI slaves which need to be de-selected between transmissions, the NSS pin must be configured as GPIO or another GPIO must be used and toggled by software.

SPI TI protocol in master mode

In master mode, the SPI interface is compatible with the TI protocol. The FRF bit of the SPI_CR2 register can be used to configure the master SPI serial communications to be compliant with this protocol.

The clock polarity and phase are forced to conform to the TI protocol requirements whatever the values set in the SPI_CR1 register. NSS management is also specific to the TI protocol which makes the configuration of NSS management through the SPI_CR1 and SPI_CR2 registers (SSM, SSI, SSOE) transparent for the user.

Figure 243: TI mode - master mode, single transfer and *Figure 244: TI mode - master mode, continuous transfer*) show the SPI master communication waveforms when the TI mode is selected in master mode.

Figure 243. TI mode - master mode, single transfer

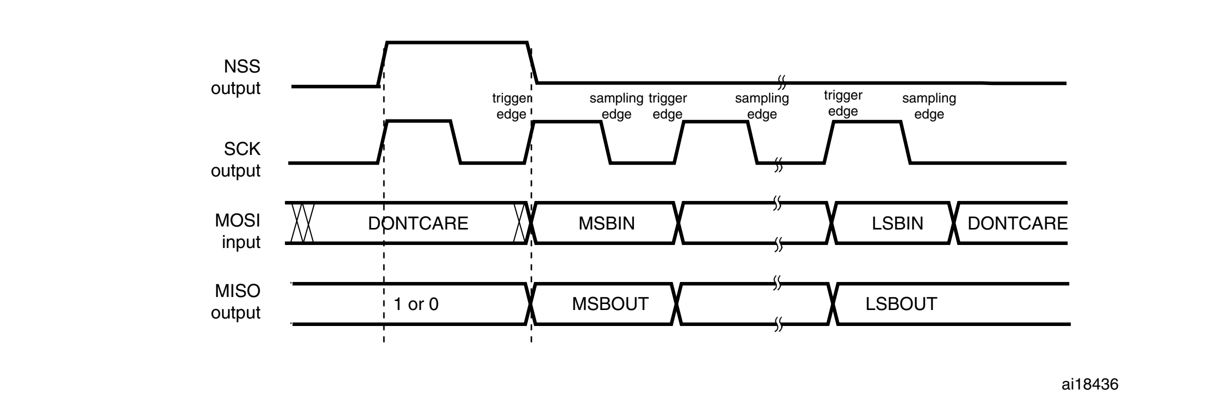
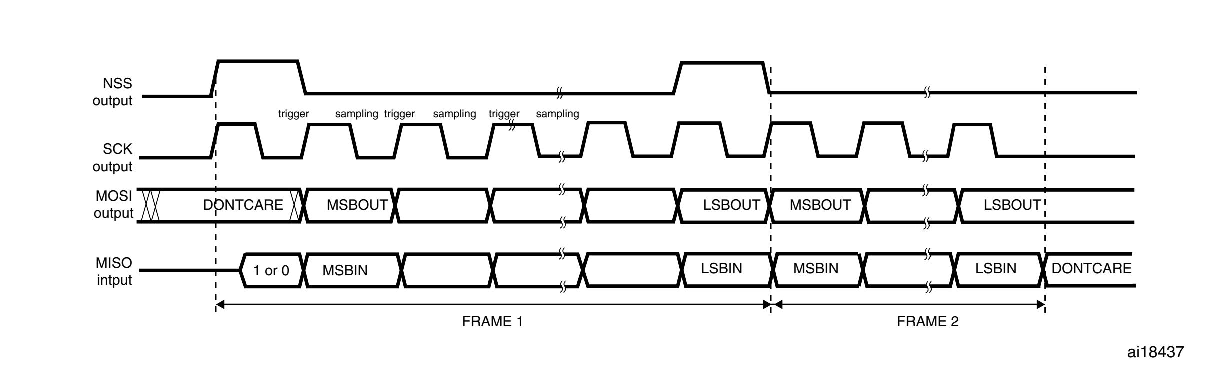


Figure 244. TI mode - master mode, continuous transfer



27.3.4 Configuring the SPI for half-duplex communication

The SPI is capable of operating in half-duplex mode in 2 configurations.

- 1 clock and 1 bidirectional data wire
- 1 clock and 1 data wire (receive-only or transmit-only)

1 clock and 1 bidirectional data wire (BIDIMODE=1)

This mode is enabled by setting the BIDIMODE bit in the SPI_CR1 register. In this mode SCK is used for the clock and MOSI in master or MISO in slave mode is used for data communication. The transfer direction (Input/Output) is selected by the BIDIOE bit in the SPI_CR1 register. When this bit is 1, the data line is output otherwise it is input.

1 clock and 1 unidirectional data wire (BIDIMODE=0)

In this mode, the application can use the SPI either in transmit-only mode or in receive-only mode.

- Transmit-only mode is similar to full-duplex mode (BIDIMODE=0, RXONLY=0): the data are transmitted on the transmit pin (MOSI in master mode or MISO in slave mode) and the receive pin (MISO in master mode or MOSI in slave mode) can be used as a general-purpose IO. In this case, the application just needs to ignore the Rx buffer (if the data register is read, it does not contain the received value).
- In receive-only mode, the application can disable the SPI output function by setting the RXONLY bit in the SPI_CR2 register. In this case, it frees the transmit IO pin (MOSI in master mode or MISO in slave mode), so it can be used for other purposes.

To start the communication in receive-only mode, configure and enable the SPI:

- In master mode, the communication starts immediately and stops when the SPE bit is cleared and the current reception stops. There is no need to read the BSY flag in this mode. It is always set when an SPI communication is ongoing.
- In slave mode, the SPI continues to receive as long as the NSS is pulled down (or the SSI bit is cleared in NSS software mode) and the SCK is running.

27.3.5 Data transmission and reception procedures

Rx and Tx buffers

In reception, data are received and then stored into an internal Rx buffer while In transmission, data are first stored into an internal Tx buffer before being transmitted.

A read access of the SPI_DR register returns the Rx buffered value whereas a write access to the SPI_DR stores the written data into the Tx buffer.

Start sequence in master mode

- In full-duplex (BIDIMODE=0 and RXONLY=0)
  - The sequence begins when data are written into the SPI_DR register (Tx buffer).
  - The data are then parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission and then shifted out serially to the MOSI pin.
  - At the same time, the received data on the MISO pin is shifted in serially to the 8-bit shift register and then parallel loaded into the SPI_DR register (Rx buffer).
- In unidirectional receive-only mode (BIDIMODE=0 and RXONLY=1)
  - The sequence begins as soon as SPE=1
  - Only the receiver is activated and the received data on the MISO pin are shifted in serially to the 8-bit shift register and then parallel loaded into the SPI_DR register (Rx buffer).
- In bidirectional mode, when transmitting (BIDIMODE=1 and BIDIOE=1)
  - The sequence begins when data are written into the SPI_DR register (Tx buffer).
  - The data are then parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission and then shifted out serially to the MOSI pin.
  - No data are received.
- In bidirectional mode, when receiving (BIDIMODE=1 and BIDIOE=0)
  - The sequence begins as soon as SPE=1 and BIDIOE=0.
  - The received data on the MOSI pin are shifted in serially to the 8-bit shift register and then parallel loaded into the SPI_DR register (Rx buffer).
  - The transmitter is not activated and no data are shifted out serially to the MOSI pin.

Start sequence in slave mode

- In full-duplex mode (BIDIMODE=0 and RXONLY=0)
  - The sequence begins when the slave device receives the clock signal and the first bit of the data on its MOSI pin. The 7 remaining bits are loaded into the shift register.
  - At the same time, the data are parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission, and then shifted out serially to the MISO pin. The software must have written the data to be sent before the SPI master device initiates the transfer.
- In unidirectional receive-only mode (BIDIMODE=0 and RXONLY=1)
  - The sequence begins when the slave device receives the clock signal and the first bit of the data on its MOSI pin. The 7 remaining bits are loaded into the shift register.
  - The transmitter is not activated and no data are shifted out serially to the MISO pin.
- In bidirectional mode, when transmitting (BIDIMODE=1 and BIDIOE=1)
  - The sequence begins when the slave device receives the clock signal and the first bit in the Tx buffer is transmitted on the MISO pin.
  - The data are then parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission and then shifted out serially to the MISO pin. The

software must have written the data to be sent before the SPI master device initiates the transfer.

  - No data are received.

- In bidirectional mode, when receiving (BIDIMODE=1 and BIDIOE=0)
  - The sequence begins when the slave device receives the clock signal and the first bit of the data on its MISO pin.
  - The received data on the MISO pin are shifted in serially to the 8-bit shift register and then parallel loaded into the SPI_DR register (Rx buffer).
  - The transmitter is not activated and no data are shifted out serially to the MISO pin.

Handling data transmission and reception

The TXE flag (Tx buffer empty) is set when the data are transferred from the Tx buffer to the shift register. It indicates that the internal Tx buffer is ready to be loaded with the next data. An interrupt can be generated if the TXEIE bit in the SPI_CR2 register is set. Clearing the TXE bit is performed by writing to the SPI_DR register.

Note: The software must ensure that the TXE flag is set to 1 before attempting to write to the Tx buffer. Otherwise, it overwrites the data previously written to the Tx buffer.

The RXNE flag (Rx buffer not empty) is set on the last sampling clock edge, when the data are transferred from the shift register to the Rx buffer. It indicates that data are ready to be read from the SPI_DR register. An interrupt can be generated if the RXNEIE bit in the SPI_CR2 register is set. Clearing the RXNE bit is performed by reading the SPI_DR register.

For some configurations, the BSY flag can be used during the last data transfer to wait until the completion of the transfer.

Full-duplex transmit and receive procedure in master or slave mode (BIDIMODE=0 and RXONLY=0)

The software has to follow this procedure to transmit and receive data (see *Figure 245* and *Figure 246*):

1. Enable the SPI by setting the SPE bit to 1.
2. Write the first data item to be transmitted into the SPI_DR register (this clears the TXE flag).
3. Wait until TXE=1 and write the second data item to be transmitted. Then wait until RXNE=1 and read the SPI_DR to get the first received data item (this clears the RXNE bit). Repeat this operation for each data item to be transmitted/received until the n–1 received data.
4. Wait until RXNE=1 and read the last received data.
5. Wait until TXE=1 and then wait until BSY=0 before disabling the SPI.

This procedure can also be implemented using dedicated interrupt subroutines launched at each rising edges of the RXNE or TXE flag.

Figure 245. TXE/RXNE/BSY behavior in Master / full-duplex mode (BIDIMODE=0 and RXONLY=0) in the case of continuous transfers

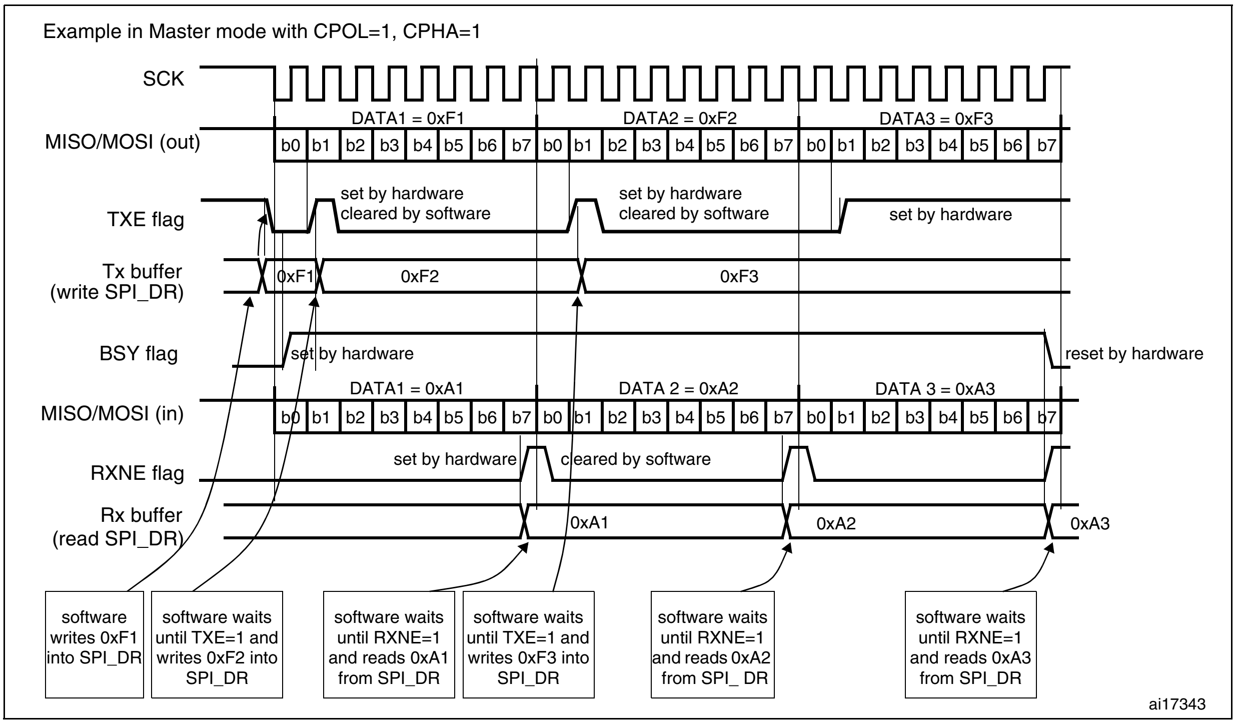
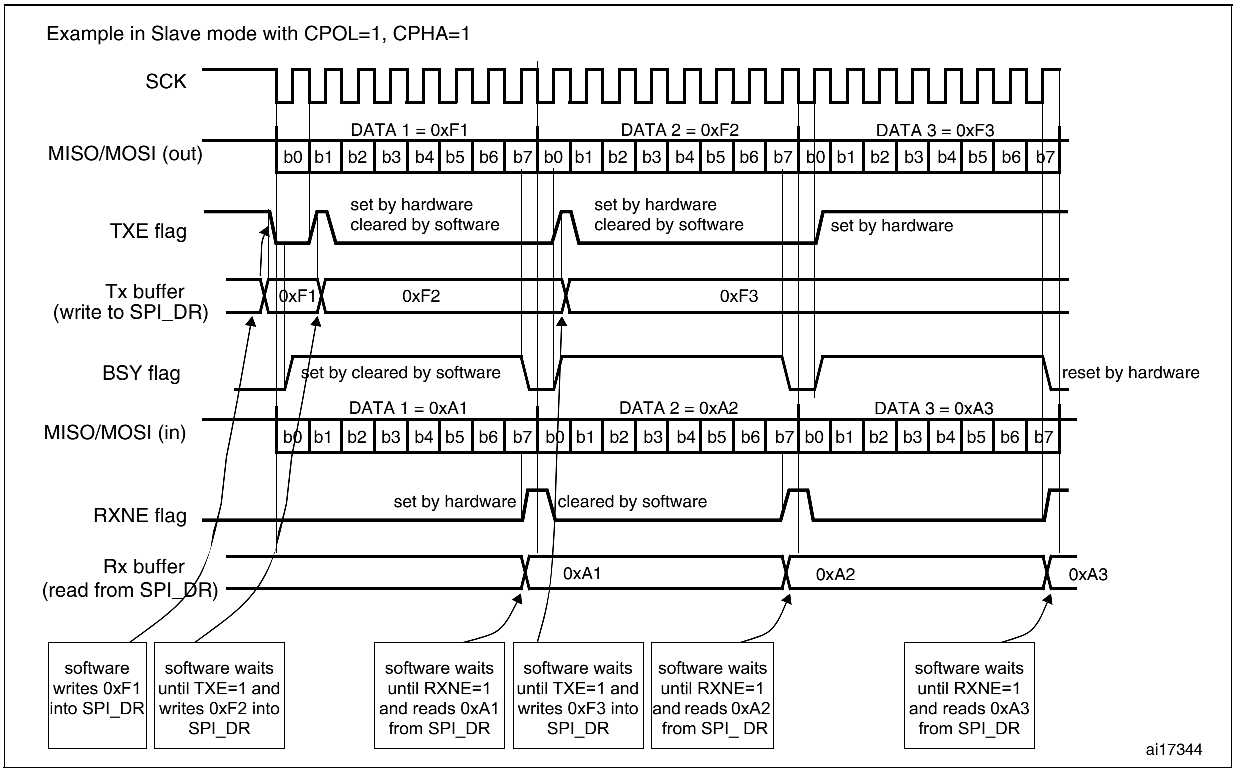


Figure 246. TXE/RXNE/BSY behavior in Slave / full-duplex mode (BIDIMODE=0, RXONLY=0) in the case of continuous transfers


Example in Slave mode with CPOL=1, CPHA=1
SCK
MISO/MOSI (out)
DATA 1 = 0xF1
DATA 2 = 0xF2
DATA 3 = 0xF3
b0 b1 b2 b3 b4 b5 b6 b7
TXE flag
set by hardware
cleared by software
set by hardware
cleared by software
set by hardware
Tx buffer (write to SPI_DR)
0xF1
0xF2
0xF3
BSY flag
set by cleared by software
reset by hardware
MISO/MOSI (in)
DATA 1 = 0xA1
DATA 2 = 0xA2
DATA 3 = 0xA3
RXNE flag
set by hardware
cleared by software
Rx buffer (read from SPI_DR)
0xA1
0xA2
0xA3
software writes 0xF1 into SPI_DR
software waits until TXE=1 and writes 0xF2 into SPI_DR
software waits until RXNE=1 and reads 0xA1 from SPI_DR
software waits until TXE=1 and writes 0xF3 into SPI_DR
software waits until RXNE=1 and reads 0xA2 from SPI_ DR
software waits until RXNE=1 and reads 0xA3 from SPI_DR
ai17344

Transmit-only procedure (BIDIMODE=0 RXONLY=0)

In this mode, the procedure can be reduced as described below and the BSY bit can be used to wait until the completion of the transmission (see *Figure 247* and *Figure 248*).

1. Enable the SPI by setting the SPE bit to 1.
2. Write the first data item to send into the SPI_DR register (this clears the TXE bit).
3. Wait until TXE=1 and write the next data item to be transmitted. Repeat this step for each data item to be transmitted.
4. After writing the last data item into the SPI_DR register, wait until TXE=1, then wait until BSY=0, this indicates that the transmission of the last data is complete.

This procedure can be also implemented using dedicated interrupt subroutines launched at each rising edge of the TXE flag.

Note: During discontinuous communications, there is a 2 APB clock period delay between the write operation to SPI_DR and the BSY bit setting. As a consequence, in transmit-only mode, it is mandatory to wait first until TXE is set and then until BSY is cleared after writing the last data.

After transmitting two data items in transmit-only mode, the OVR flag is set in the SPI_SR register since the received data are never read.

Figure 247. TXE/BSY behavior in Master transmit-only mode (BIDIMODE=0 and RXONLY=0) in case of continuous transfers

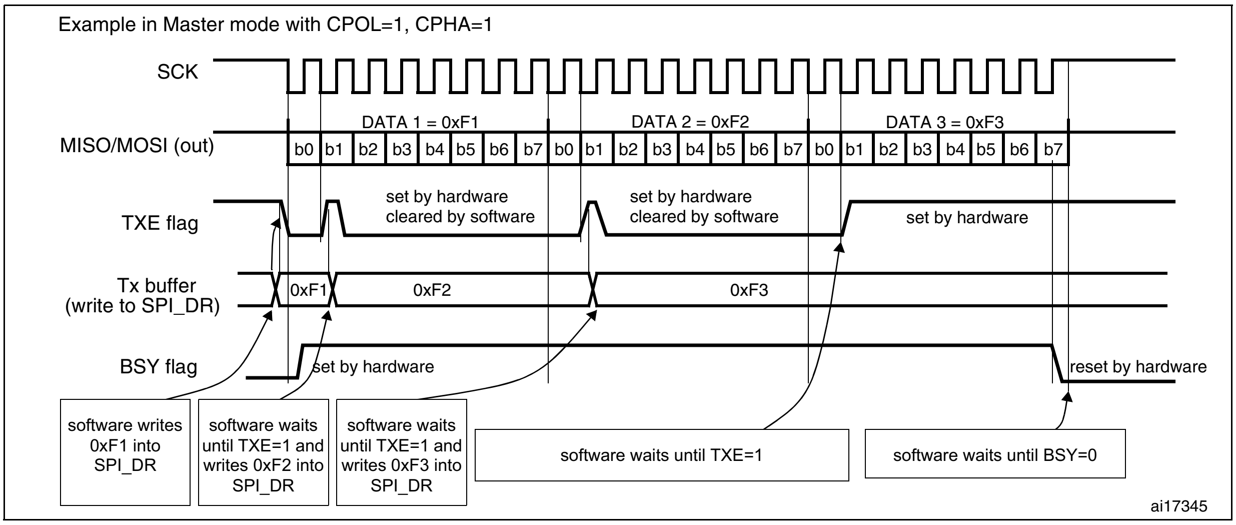
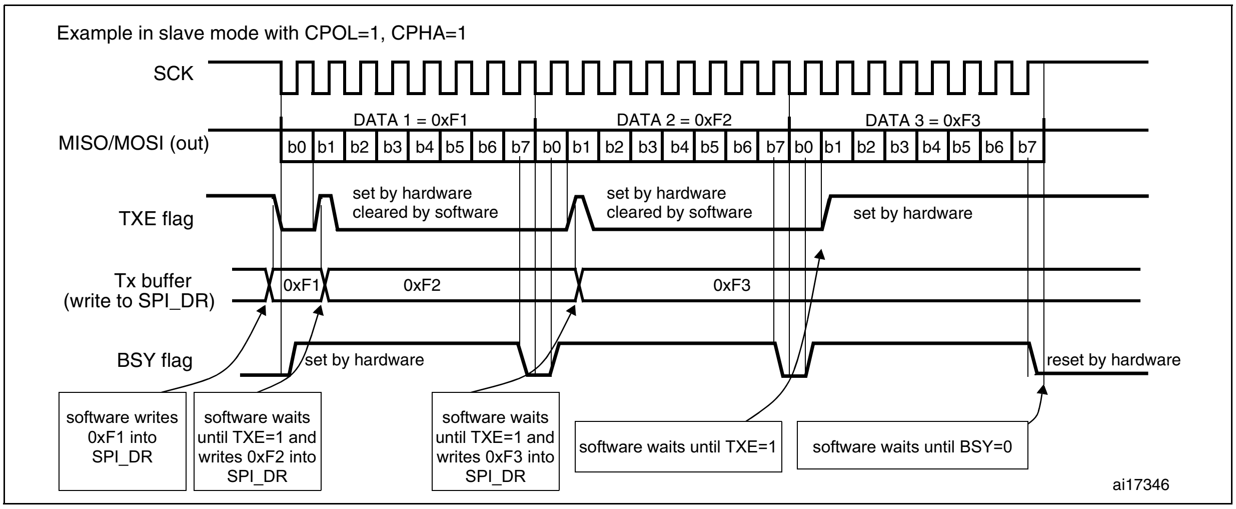

Example in Master mode with CPOL=1, CPHA=1
SCK
MISO/MOSI (out)
DATA 1 = 0xF1
DATA 2 = 0xF2
DATA 3 = 0xF3
b0 b1 b2 b3 b4 b5 b6 b7 b0 b1 b2 b3 b4 b5 b6 b7 b0 b1 b2 b3 b4 b5 b6 b7
TXE flag
set by hardware
cleared by software
set by hardware
cleared by software
set by hardware
Tx buffer
(write to SPI_DR)
0xF1
0xF2
0xF3
BSY flag
set by hardware
reset by hardware
software writes 0xF1 into SPI_DR
software waits until TXE=1 and writes 0xF2 into SPI_DR
software waits until TXE=1 and writes 0xF3 into SPI_DR
software waits until TXE=1
software waits until BSY=0
ai17345

Figure 248. TXE/BSY in Slave transmit-only mode (BIDIMODE=0 and RXONLY=0) in the case of continuous transfers


Example in slave mode with CPOL=1, CPHA=1
SCK
MISO/MOSI (out)
DATA 1 = 0xF1
DATA 2 = 0xF2
DATA 3 = 0xF3
b0 b1 b2 b3 b4 b5 b6 b7 b0 b1 b2 b3 b4 b5 b6 b7 b0 b1 b2 b3 b4 b5 b6 b7
TXE flag
set by hardware
cleared by software
set by hardware
cleared by software
set by hardware
Tx buffer
(write to SPI_DR)
0xF1
0xF2
0xF3
BSY flag
set by hardware
reset by hardware
software writes 0xF1 into SPI_DR
software waits until TXE=1 and writes 0xF2 into SPI_DR
software waits until TXE=1 and writes 0xF3 into SPI_DR
software waits until TXE=1
software waits until BSY=0
ai17346

Bidirectional transmit procedure (BIDIMODE=1 and BIDIOE=1)

In this mode, the procedure is similar to the procedure in Transmit-only mode except that the BIDIMODE and BIDIOE bits both have to be set in the SPI_CR2 register before enabling the SPI.

Unidirectional receive-only procedure (BIDIMODE=0 and RXONLY=1)

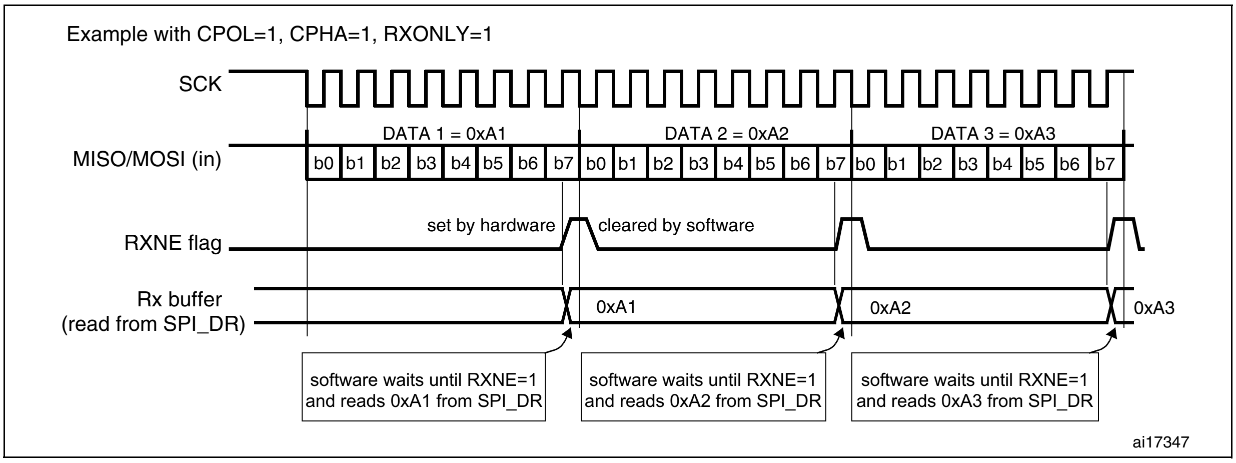
In this mode, the procedure can be reduced as described below (see *Figure 249*):

1. Set the RXONLY bit in the SPI_CR2 register.
2. Enable the SPI by setting the SPE bit to 1:
   a) In master mode, this immediately activates the generation of the SCK clock, and data are serially received until the SPI is disabled (SPE=0).
   b) In slave mode, data are received when the SPI master device drives NSS low and generates the SCK clock.
3. Wait until RXNE=1 and read the SPI_DR register to get the received data (this clears the RXNE bit). Repeat this operation for each data item to be received.

This procedure can also be implemented using dedicated interrupt subroutines launched at each rising edge of the RXNE flag.

Note: *If it is required to disable the SPI after the last transfer, follow the recommendation described in Section 27.3.8: Disabling the SPI on page 722.*

Figure 249. RXNE behavior in receive-only mode (BIDIRMODE=0 and RXONLY=1) in the case of continuous transfers


Example with CPOL=1, CPHA=1, RXONLY=1
SCK
MISO/MOSI (in)
DATA 1 = 0xA1
DATA 2 = 0xA2
DATA 3 = 0xA3
b0 b1 b2 b3 b4 b5 b6 b7
RXNE flag
set by hardware
cleared by software
Rx buffer
(read from SPI_DR)
0xA1
0xA2
0xA3
software waits until RXNE=1 and reads 0xA1 from SPI_DR
software waits until RXNE=1 and reads 0xA2 from SPI_DR
software waits until RXNE=1 and reads 0xA3 from SPI_DR
ai17347

Bidirectional receive procedure (BIDIMODE=1 and BIDIOE=0)

In this mode, the procedure is similar to the Receive-only mode procedure except that the BIDIMODE bit has to be set and the BIDIOE bit cleared in the SPI_CR2 register before enabling the SPI.

Continuous and discontinuous transfers

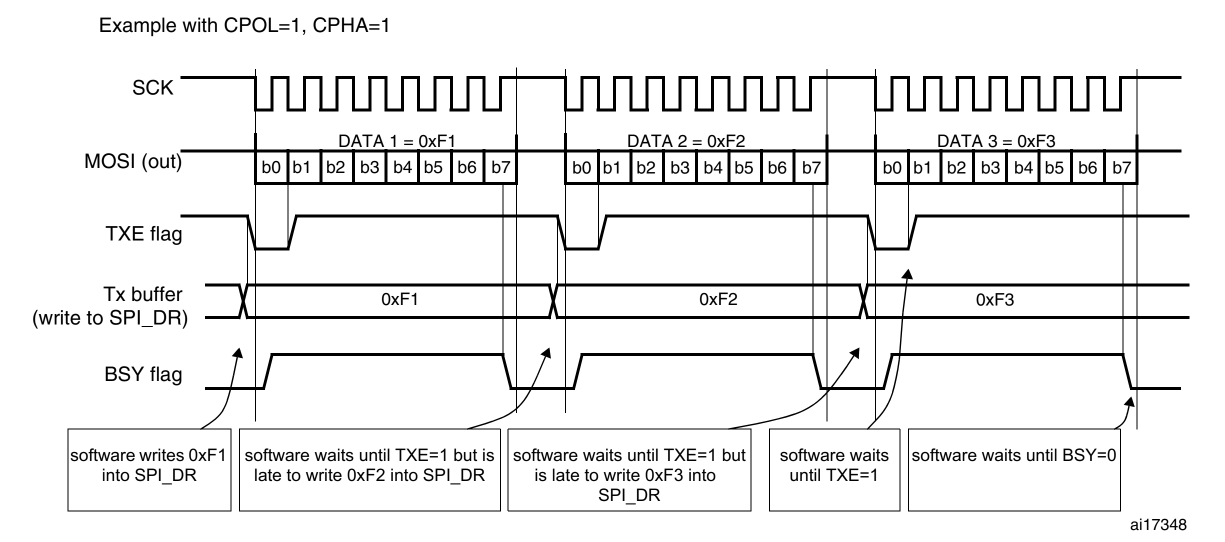
When transmitting data in master mode, if the software is fast enough to detect each rising edge of TXE (or TXE interrupt) and to immediately write to the SPI_DR register before the ongoing data transfer is complete, the communication is said to be continuous. In this case, there is no discontinuity in the generation of the SPI clock between each data item and the BSY bit is never cleared between each data transfer.

On the contrary, if the software is not fast enough, this can lead to some discontinuities in the communication. In this case, the BSY bit is cleared between each data transmission (see *Figure 250*).

In Master receive-only mode (RXONLY=1), the communication is always continuous and the BSY flag is always read at 1.

In slave mode, the continuity of the communication is decided by the SPI master device. In any case, even if the communication is continuous, the BSY flag goes low between each transfer for a minimum duration of one SPI clock cycle (see *Figure 248*).

Figure 250. TXE/BSY behavior when transmitting (BIDIRMODE=0 and RXONLY=0) in the case of discontinuous transfers



Example with CPOL=1, CPHA=1
SCK
MOSI (out)
DATA 1 = 0xF1
DATA 2 = 0xF2
DATA 3 = 0xF3
b0 b1 b2 b3 b4 b5 b6 b7
TXE flag
Tx buffer (write to SPI_DR)
0xF1
0xF2
0xF3
BSY flag
software writes 0xF1 into SPI_DR
software waits until TXE=1 but is late to write 0xF2 into SPI_DR
software waits until TXE=1 but is late to write 0xF3 into SPI_DR
software waits until TXE=1
software waits until BSY=0
ai17348

27.3.6 CRC calculation

A CRC calculator has been implemented for communication reliability. Separate CRC calculators are implemented for transmitted data and received data. The CRC is calculated using a programmable polynomial serially on each bit. It is calculated on the sampling clock edge defined by the CPHA and CPOL bits in the SPI_CR1 register.

Note: This SPI offers two kinds of CRC calculation standard which depend directly on the data frame format selected for the transmission and/or reception: 8-bit data (CR8) and 16-bit data (CRC16).

CRC calculation is enabled by setting the CRCEN bit in the SPI_CR1 register. This action resets the CRC registers (SPI_RXCRCR and SPI_TXCRCR). In full duplex or transmitter only mode, when the transfers are managed by the software (CPU mode), it is necessary to write the bit CRCNEXT immediately after the last data to be transferred is written to the SPI_DR. At the end of this last data transfer, the SPI_TXCRCR value is transmitted.

In receive only mode and when the transfers are managed by software (CPU mode), it is necessary to write the CRCNEXT bit after the second last data has been received. The CRC is received just after the last data reception and the CRC check is then performed.

At the end of data and CRC transfers, the CRCERR flag in the SPI_SR register is set if corruption occurs during the transfer.

If data are present in the TX buffer, the CRC value is transmitted only after the transmission of the data byte. During CRC transmission, the CRC calculator is switched off and the register value remains unchanged.

SPI communication using the CRC is possible through the following procedure:

1. Program the CPOL, CPHA, LSBFirst, BR, SSM, SSI and MSTR values.
2. Program the polynomial in the SPI_CRCPR register.
3. Enable the CRC calculation by setting the CRCEN bit in the SPI_CR1 register. This also clears the SPI_RXCRCR and SPI_TXCRCR registers.
4. Enable the SPI by setting the SPE bit in the SPI_CR1 register.
5. Start the communication and sustain the communication until all but one byte or half-word have been transmitted or received.
   - In full duplex or transmitter-only mode, when the transfers are managed by software, when writing the last byte or half word to the Tx buffer, set the CRCNEXT bit in the SPI_CR1 register to indicate that the CRC will be transmitted after the transmission of the last byte.
   - In receiver only mode, set the bit CRCNEXT just after the reception of the second to last data to prepare the SPI to enter in CRC Phase at the end of the reception of the last data. CRC calculation is frozen during the CRC transfer.
6. After the transfer of the last byte or half word, the SPI enters the CRC transfer and check phase. In full duplex mode or receiver-only mode, the received CRC is compared to the SPI_RXCRCR value. If the value does not match, the CRCERR flag in SPI_SR is set and an interrupt can be generated when the ERRIE bit in the SPI_CR2 register is set.

Note: *When the SPI is in slave mode, be careful to enable CRC calculation only when the clock is stable, that is, when the clock is in the steady state. If not, a wrong CRC calculation may be done. In fact, the CRC is sensitive to the SCK slave input clock as soon as CRCEN is set, and this, whatever the value of the SPE bit.*

With high bitrate frequencies, be careful when transmitting the CRC. As the number of used CPU cycles has to be as low as possible in the CRC transfer phase, it is forbidden to call software functions in the CRC transmission sequence to avoid errors in the last data and CRC reception. In fact, CRCNEXT bit has to be written before the end of the transmission/reception of the last data.

For high bit rate frequencies, it is advised to use the DMA mode to avoid the degradation of the SPI speed performance due to CPU accesses impacting the SPI bandwidth.

When the devices are configured as slaves and the NSS hardware mode is used, the NSS pin needs to be kept low between the data phase and the CRC phase.

When the SPI is configured in slave mode with the CRC feature enabled, CRC calculation takes place even if a high level is applied on the NSS pin. This may happen for example in case of a multislave environment where the communication master addresses slaves alternately.

Between a slave deselection (high level on NSS) and a new slave selection (low level on NSS), the CRC value should be cleared on both master and slave sides in order to resynchronize the master and slave for their respective CRC calculation.

To clear the CRC, follow the procedure below:

1. Disable SPI (SPE = 0)
2. Clear the CRCEN bit
3. Set the CRCEN bit
4. Enable the SPI (SPE = 1)

27.3.7 Status flags

Three status flags are provided for the application to completely monitor the state of the SPI bus.

Tx buffer empty flag (TXE)

When it is set, this flag indicates that the Tx buffer is empty and the next data to be transmitted can be loaded into the buffer. The TXE flag is cleared when writing to the SPI_DR register.

Rx buffer not empty (RXNE)

When set, this flag indicates that there are valid received data in the Rx buffer. It is cleared when SPI_DR is read.

BUSY flag

This BSY flag is set and cleared by hardware (writing to this flag has no effect). The BSY flag indicates the state of the communication layer of the SPI.

When BSY is set, it indicates that the SPI is busy communicating. There is one exception in master mode / bidirectional receive mode (MSTR=1 and BDM=1 and BDOE=0) where the BSY flag is kept low during reception.

The BSY flag is useful to detect the end of a transfer if the software wants to disable the SPI and enter Halt mode (or disable the peripheral clock). This avoids corrupting the last transfer. For this, the procedure described below must be strictly respected.

The BSY flag is also useful to avoid write collisions in a multimaster system.

The BSY flag is set when a transfer starts, with the exception of master mode / bidirectional receive mode (MSTR=1 and BDM=1 and BDOE=0).

It is cleared:

- when a transfer is finished (except in master mode if the communication is continuous)
- when the SPI is disabled
- when a master mode fault occurs (MODF=1)

When communication is not continuous, the BSY flag is low between each communication.

When communication is continuous:

- in master mode, the BSY flag is kept high during all the transfers
- in slave mode, the BSY flag goes low for one SPI clock cycle between each transfer

Note: Do not use the BSY flag to handle each data transmission or reception. It is better to use the TXE and RXNE flags instead.

27.3.8 Disabling the SPI

When a transfer is terminated, the application can stop the communication by disabling the SPI peripheral. This is done by clearing the SPE bit.

For some configurations, disabling the SPI and entering the Halt mode while a transfer is ongoing can cause the current transfer to be corrupted and/or the BSY flag might become unreliable.

To avoid any of those effects, it is recommended to respect the following procedure when disabling the SPI:

In master or slave full-duplex mode (BIDIMODE=0, RXONLY=0)

1. Wait until RXNE=1 to receive the last data
2. Wait until TXE=1
3. Then wait until BSY=0
4. Disable the SPI (SPE=0) and, eventually, enter the Halt mode (or disable the peripheral clock)

In master or slave unidirectional transmit-only mode (BIDIMODE=0, RXONLY=0) or bidirectional transmit mode (BIDIMODE=1, BIDIOE=1)

After the last data is written into the SPI_DR register:

1. Wait until TXE=1
2. Then wait until BSY=0
3. Disable the SPI (SPE=0) and, eventually, enter the Halt mode (or disable the peripheral clock)

In master unidirectional receive-only mode (MSTR=1, BIDIMODE=0, RXONLY=1) or bidirectional receive mode (MSTR=1, BIDIMODE=1, BIDIOE=0)

This case must be managed in a particular way to ensure that the SPI does not initiate a new transfer. The sequence below is valid only for SPI Motorola configuration (FRF bit set to 0):

1. Wait for the second to last occurrence of RXNE=1 (n–1)
2. Then wait for one SPI clock cycle (using a software loop) before disabling the SPI (SPE=0)
3. Then wait for the last RXNE=1 before entering the Halt mode (or disabling the peripheral clock)

When the SPI is configured in TI mode (Bit FRF set to 1), the following procedure has to be respected to avoid generating an undesired pulse on NSS when the SPI is disabled:

1. Wait for the second to last occurrence of RXNE = 1 (n-1).
2. Disable the SPI (SPE = 0) in the following window frame using a software loop:
   – After at least one SPI clock cycle,
   – Before the beginning of the LSB data transfer.

Note: *In master bidirectional receive mode (MSTR=1 and BDM=1 and BDOE=0), the BSY flag is kept low during transfers.*

In slave receive-only mode (MSTR=0, BIDIMODE=0, RXONLY=1) or bidirectional receive mode (MSTR=0, BIDIMODE=1, BIDOE=0)

1. You can disable the SPI (write SPE=1) at any time: the current transfer will complete before the SPI is effectively disabled
2. Then, if you want to enter the Halt mode, you must first wait until BSY = 0 before entering the Halt mode (or disabling the peripheral clock).

27.3.9 SPI communication using DMA (direct memory addressing)

To operate at its maximum speed, the SPI needs to be fed with the data for transmission and the data received on the Rx buffer should be read to avoid overrun. To facilitate the transfers, the SPI features a DMA capability implementing a simple request/acknowledge protocol.

A DMA access is requested when the enable bit in the SPI_CR2 register is enabled. Separate requests must be issued to the Tx and Rx buffers (see *Figure 251* and *Figure 252*):

- In transmission, a DMA request is issued each time TXE is set to 1. The DMA then writes to the SPI_DR register (this clears the TXE flag).
- In reception, a DMA request is issued each time RXNE is set to 1. The DMA then reads the SPI_DR register (this clears the RXNE flag).

When the SPI is used only to transmit data, it is possible to enable only the SPI Tx DMA channel. In this case, the OVR flag is set because the data received are not read.

When the SPI is used only to receive data, it is possible to enable only the SPI Rx DMA channel.

In transmission mode, when the DMA has written all the data to be transmitted (flag TCIF is set in the DMA_ISR register), the BSY flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or entering the Stop mode. The software must first wait until TXE=1 and then until BSY=0.

Note: During discontinuous communications, there is a 2 APB clock period delay between the write operation to SPI_DR and the BSY bit setting. As a consequence, it is mandatory to wait first until TXE=1 and then until BSY=0 after writing the last data.

Figure 251. Transmission using DMA

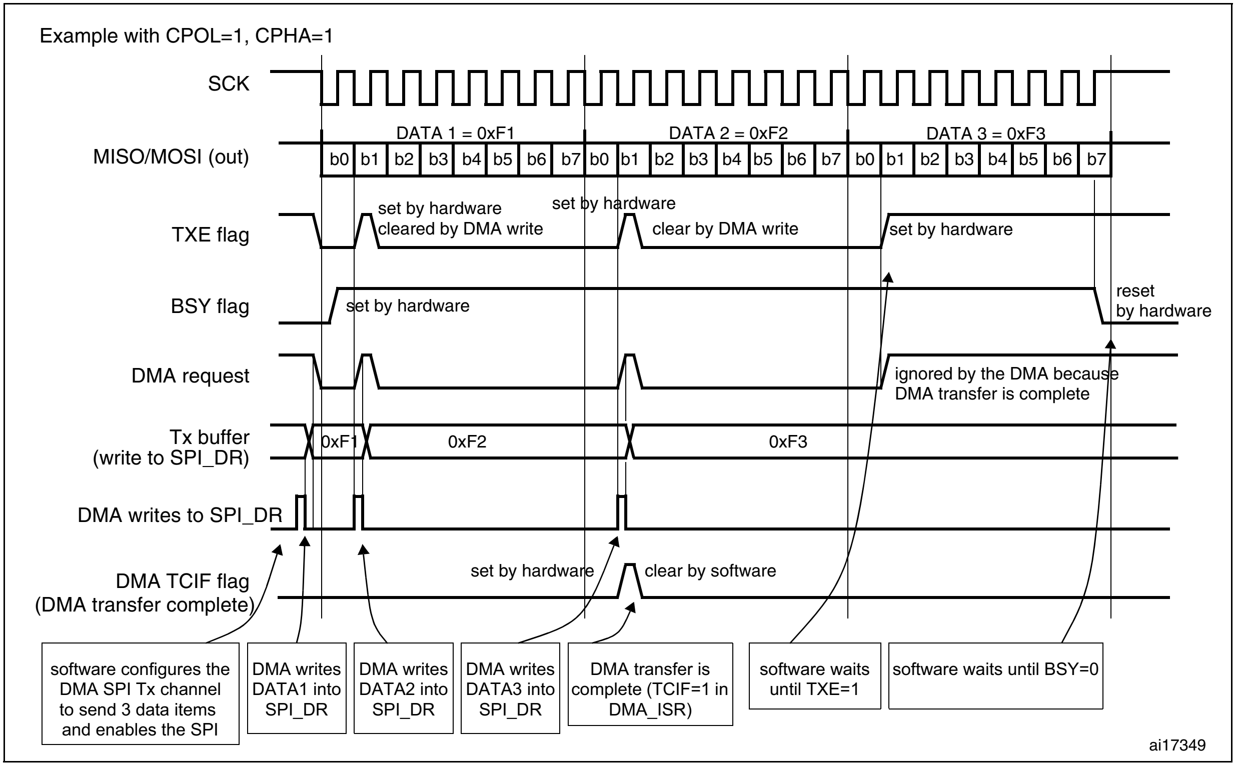
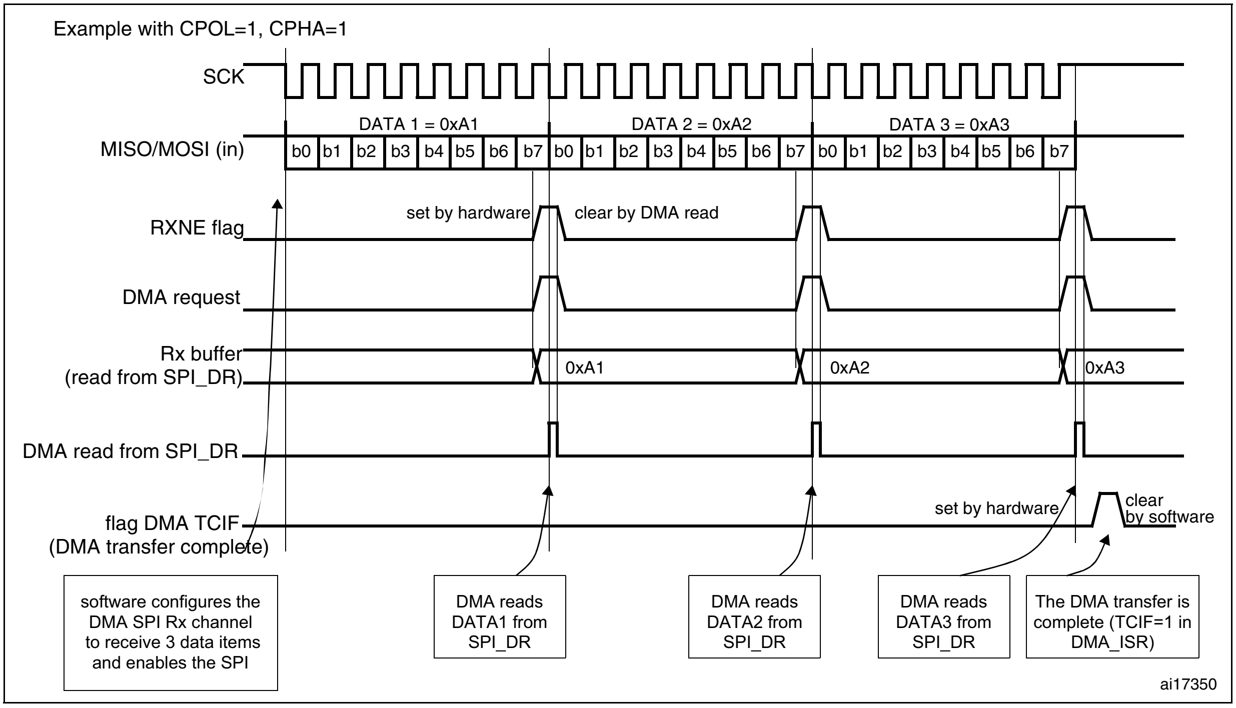

Example with CPOL=1, CPHA=1
SCK
MISO/MOSI (out)
DATA 1 = 0xF1
DATA 2 = 0xF2
DATA 3 = 0xF3
b0 b1 b2 b3 b4 b5 b6 b7
TXE flag
set by hardware
cleared by DMA write
set by hardware
clear by DMA write
set by hardware
BSY flag
set by hardware
reset by hardware
DMA request
ignored by the DMA because DMA transfer is complete
Tx buffer (write to SPI_DR)
0xF1
0xF2
0xF3
DMA writes to SPI_DR
DMA TCIF flag (DMA transfer complete)
set by hardware
clear by software
software configures the DMA SPI Tx channel to send 3 data items and enables the SPI
DMA writes DATA1 into SPI_DR
DMA writes DATA2 into SPI_DR
DMA writes DATA3 into SPI_DR
DMA transfer is complete (TCIF=1 in DMA_ISR)
software waits until TXE=1
software waits until BSY=0
ai17349

Figure 252. Reception using DMA


Example with CPOL=1, CPHA=1
SCK
MISO/MOSI (in)
DATA 1 = 0xA1
DATA 2 = 0xA2
DATA 3 = 0xA3
b0 b1 b2 b3 b4 b5 b6 b7
RXNE flag
set by hardware
clear by DMA read
DMA request
Rx buffer (read from SPI_DR)
0xA1
0xA2
0xA3
DMA read from SPI_DR
flag DMA TCIF (DMA transfer complete)
set by hardware
clear by software
software configures the DMA SPI Rx channel to receive 3 data items and enables the SPI
DMA reads DATA1 from SPI_DR
DMA reads DATA2 from SPI_DR
DMA reads DATA3 from SPI_DR
The DMA transfer is complete (TCIF=1 in DMA_ISR)
ai17350

DMA capability with CRC

When SPI communication is enabled with CRC communication and DMA mode, the transmission and reception of the CRC at the end of communication are automatic that is without using the bit CRCNEXT. After the CRC reception, the CRC must be read in the SPI_DR register to clear the RXNE flag.

At the end of data and CRC transfers, the CRCERR flag in SPI_SR is set if corruption occurs during the transfer.

27.3.10 Error flags

Master mode fault (MODF)

Master mode fault occurs when the master device has its NSS pin pulled low (in NSS hardware mode) or SSI bit low (in NSS software mode), this automatically sets the MODF bit. Master mode fault affects the SPI peripheral in the following ways:

- The MODF bit is set and an SPI interrupt is generated if the ERRIE bit is set.
- The SPE bit is cleared. This blocks all output from the device and disables the SPI interface.
- The MSTR bit is cleared, thus forcing the device into slave mode.

Use the following software sequence to clear the MODF bit:

1. Make a read or write access to the SPI_SR register while the MODF bit is set.
2. Then write to the SPI_CR1 register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled high during the MODF bit clearing sequence. The SPE and MSTR bits can be restored to their original state after this clearing sequence.

As a security, hardware does not allow the setting of the SPE and MSTR bits while the MODF bit is set.

In a slave device the MODF bit cannot be set. However, in a multimaster configuration, the device can be in slave mode with this MODF bit set. In this case, the MODF bit indicates that there might have been a multimaster conflict for system control. An interrupt routine can be used to recover cleanly from this state by performing a reset or returning to a default state.

Overrun condition

An overrun condition occurs when the master device has sent data bytes and the slave device has not cleared the RXNE bit resulting from the previous data byte transmitted. When an overrun condition occurs:

- the OVR bit is set and an interrupt is generated if the ERRIE bit is set.

In this case, the receiver buffer contents will not be updated with the newly received data from the master device. A read from the SPI_DR register returns this byte. All other subsequently transmitted bytes are lost.

Clearing the OVR bit is done by a read from the SPI_DR register followed by a read access to the SPI_SR register.

CRC error

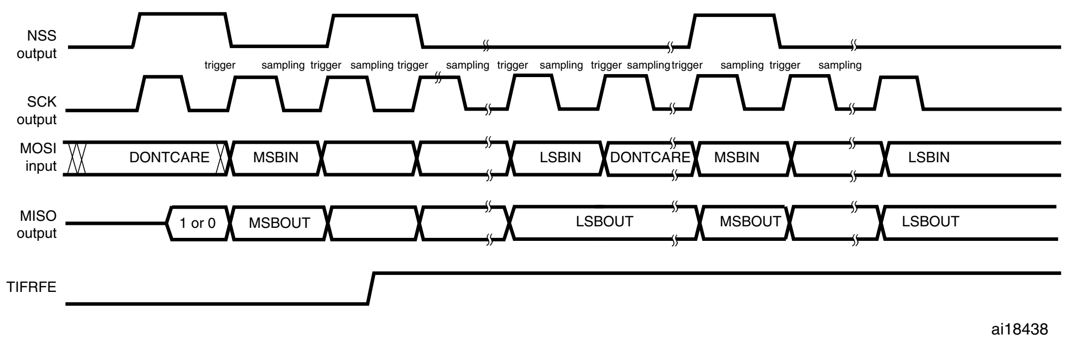
This flag is used to verify the validity of the value received when the CRCEN bit in the SPI_CR1 register is set. The CRCERR flag in the SPI_SR register is set if the value received in the shift register does not match the receiver SPI_RXCRCR value.

TI mode frame format error

A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI is acting in slave mode and configured to conform to the TI mode protocol. When this error occurs, the TIFRFE 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 lost of two data bytes.

The TIFRFE flag is cleared when SPI_SR register is read. If the bit ERRIE is set, an interrupt is generated on the NSS error detection. In this case, the SPI should be disabled because data consistency is no more guaranteed and communications should be reinitiated by the master when the slave SPI is re-enabled.

Figure 253. TI mode frame format error detection



27.3.11 SPI interrupts

Table 145. SPI interrupt requests

Interrupt event	Event flag	Enable Control bit
Transmit buffer empty flag	TXE	TXEIE
Receive buffer not empty flag	RXNE	RXNEIE
Master Mode fault event	MODF	ERRIE
Overrun error	OVR	
CRC error flag	CRCERR	
TI frame format error	TIFRFE	ERRIE

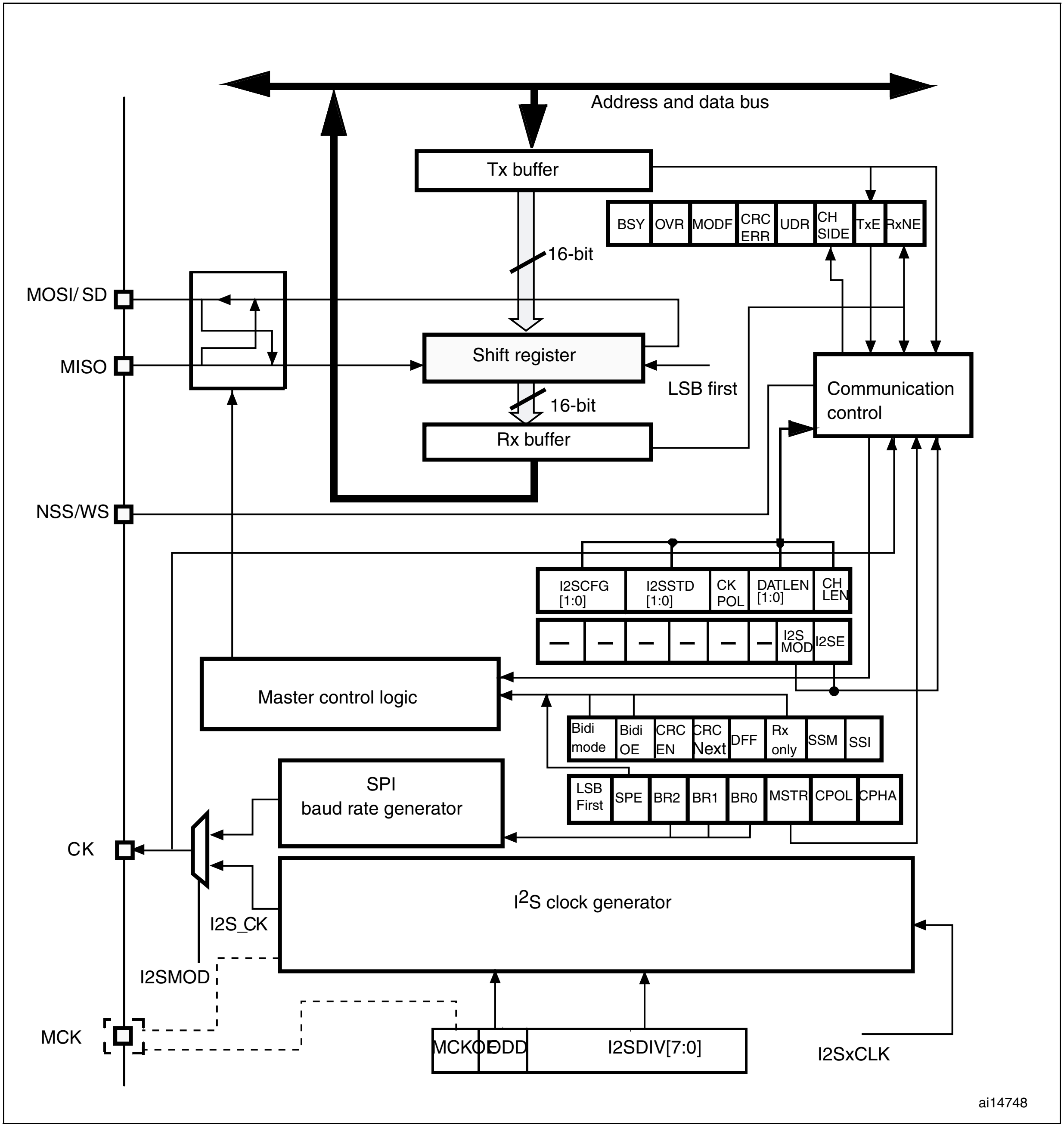
27.4 I^2S functional description

The I^2S audio protocol is not available in low and medium density devices.

27.4.1 I^2S general description

The block diagram of the I^2S is shown in *Figure 254*.

Figure 254. I^2S block diagram



The SPI could function as an audio I^2S interface when the I^2S capability is enabled (by setting the I2SMOD bit in the SPI_I2SCFGR register). This interface uses almost the same pins, flags and interrupts as the SPI.

The I^2S shares three common pins with the SPI:

- SD: Serial Data (mapped on the MOSI pin) to transmit or receive the two time-multiplexed data channels (in half-duplex mode only).
- WS: Word Select (mapped on the NSS pin) is the data control signal output in master mode and input in slave mode.
- CK: Serial Clock (mapped on the SCK pin) is the serial clock output in master mode and serial clock input in slave mode.

An additional pin could be used when a master clock output is needed for some external audio devices:

- MCK: Master Clock (mapped separately) is used, when the I^2S is configured in master mode (and when the MCKOE bit in the SPI_I2SPR register is set), to output this additional clock generated at a preconfigured frequency rate equal to 256 × F_S, where F_S is the audio sampling frequency.

The I^2S uses its own clock generator to produce the communication clock when it is set in master mode. This clock generator is also the source of the master clock output. Two additional registers are available in I^2S mode. One is linked to the clock generator configuration SPI_I2SPR and the other one is a generic I^2S configuration register SPI_I2SCFGR (audio standard, slave/master mode, data format, packet frame, clock polarity, etc.).

The SPI_CR1 register and all CRC registers are not used in the I^2S mode. Likewise, the SSOE bit in the SPI_CR2 register and the MODF and CRCERR bits in the SPI_SR are not used.

The I^2S uses the same SPI register for data transfer (SPI_DR) in 16-bit wide mode.

27.4.2 Supported audio protocols

The three-line bus has to handle only audio data generally time-multiplexed on two channels: the right channel and the left channel. However there is only one 16-bit register for the transmission and the reception. So, it is up to the software to write into the data register the adequate value corresponding to the considered channel side, or to read the data from the data register and to identify the corresponding channel by checking the CHSIDE bit in the SPI_SR register. Channel Left is always sent first followed by the channel right (CHSIDE has no meaning for the PCM protocol).

Four data and packet frames are available. Data may be sent with a format of:

- 16-bit data packed in 16-bit frame
- 16-bit data packed in 32-bit frame
- 24-bit data packed in 32-bit frame
- 32-bit data packed in 32-bit frame

When using 16-bit data extended on 32-bit packet, the first 16 bits (MSB) are the significant bits, the 16-bit LSB is forced to 0 without any need for software action or DMA request (only one read/write operation).

The 24-bit and 32-bit data frames need two CPU read or write operations to/from the SPI_DR or two DMA operations if the DMA is preferred for the application. For 24-bit data frame specifically, the 8 nonsignificant bits are extended to 32 bits with 0-bits (by hardware).

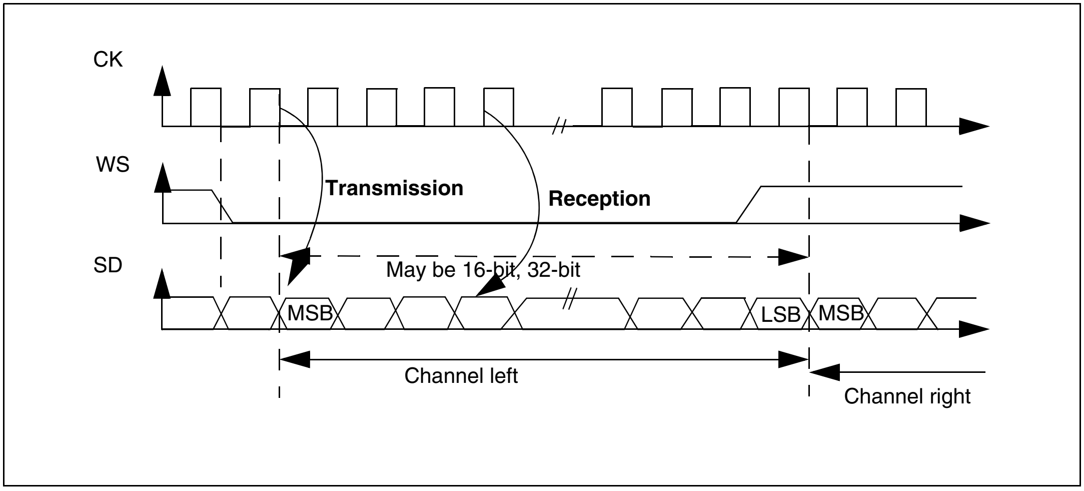
For all data formats and communication standards, the most significant bit is always sent first (MSB first).

The I^2S interface supports four audio standards, configurable using the I2SSTD[1:0] and PCMSYNC bits in the SPI_I2SCFGR register.

I^2S Philips standard

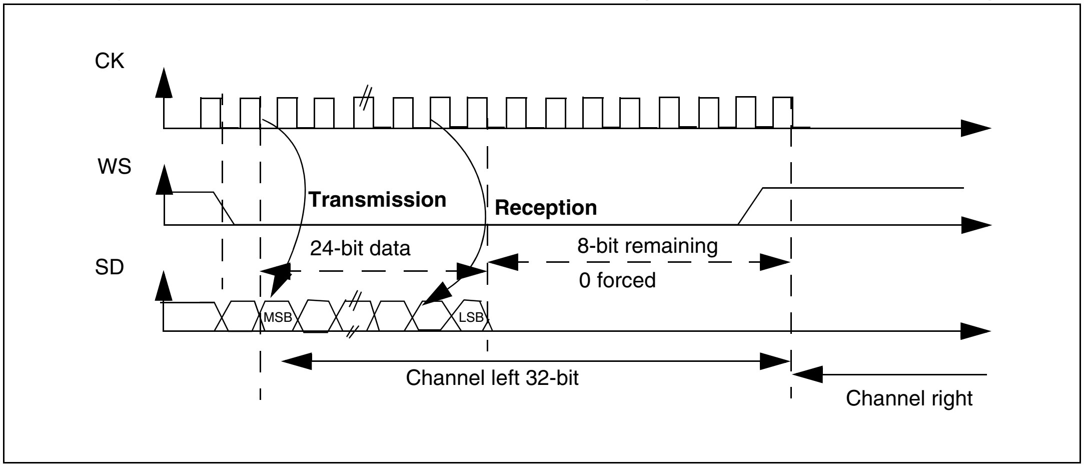
For this standard, the WS signal is used to indicate which channel is being transmitted. It is activated one CK clock cycle before the first bit (MSB) is available.

Figure 255. I^2S Philips protocol waveforms (16/32-bit full accuracy, CPOL = 0)



Data are latched on the falling edge of CK (for the transmitter) and are read on the rising edge (for the receiver). The WS signal is also latched on the falling edge of CK.

Figure 256. I^2S Philips standard waveforms (24-bit frame with CPOL = 0)

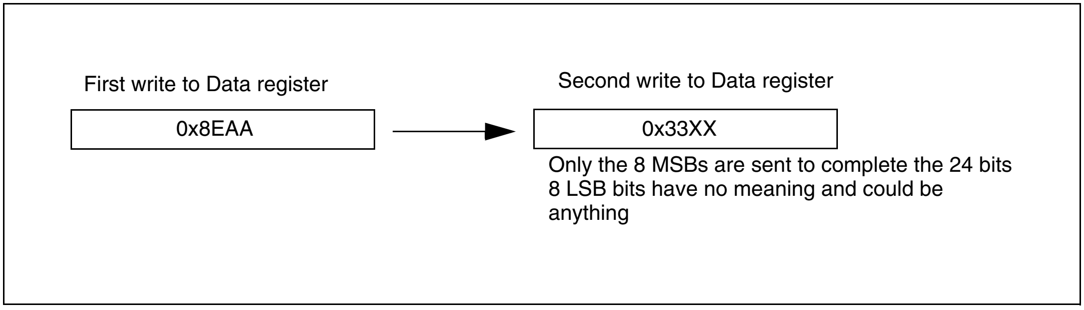


This mode needs two write or read operations to/from the SPI_DR.

- In transmission mode:

  if 0x8EAA33 has to be sent (24-bit):

Figure 257. Transmitting 0x8EAA33


First write to Data register
0x8EAA
Second write to Data register
0x33XX
Only the 8 MSBs are sent to complete the 24 bits
8 LSB bits have no meaning and could be anything

- In reception mode:

  if data 0x8EAA33 is received:

Figure 258. Receiving 0x8EAA33

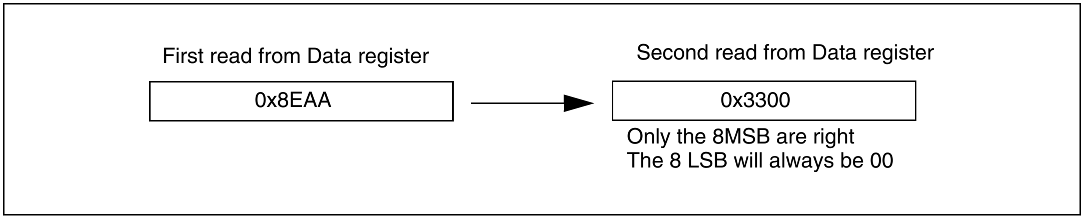
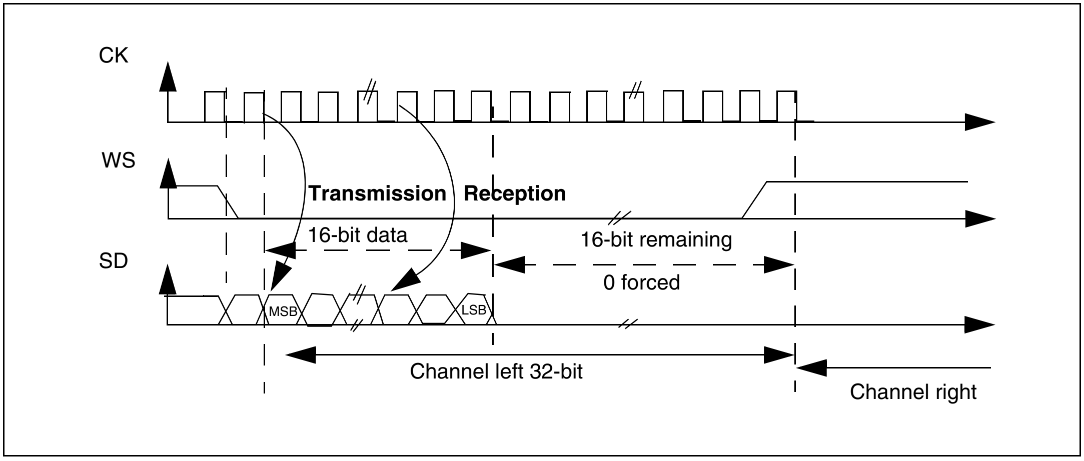

First read from Data register
0x8EAA
Second read from Data register
0x3300
Only the 8MSB are right
The 8 LSB will always be 00

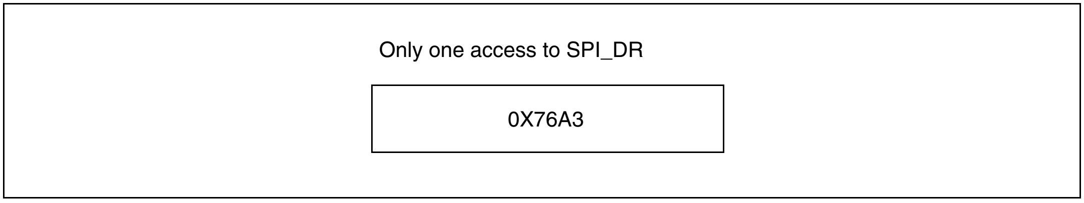
Figure 259. I^{2}S Philips standard (16-bit extended to 32-bit packet frame with CPOL = 0)



When 16-bit data frame extended to 32-bit channel frame is selected during the I^{2}S configuration phase, only one access to SPI_DR is required. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format.

If the data to transmit or the received data are 0x76A3 (0x76A30000 extended to 32-bit), the operation shown in *Figure 260* is required.

Figure 260. Example



For transmission, each time an MSB is written to SPI_DR, the TXE flag is set and its interrupt, if allowed, is generated to load SPI_DR with the new value to send. This takes place even if 0x0000 have not yet been sent because it is done by hardware.

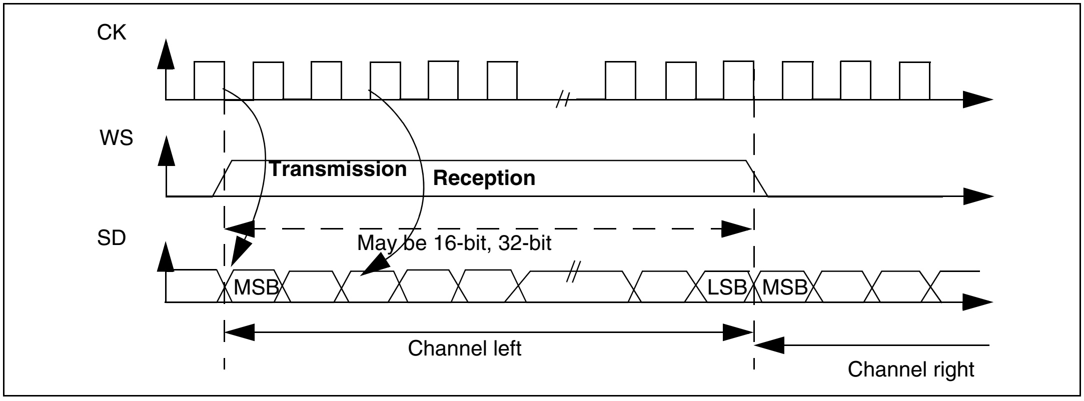
For reception, the RXNE flag is set and its interrupt, if allowed, is generated when the first 16 MSB half-word is received.

In this way, more time is provided between two write or read operations, which prevents underrun or overrun conditions (depending on the direction of the data transfer).

MSB justified standard

For this standard, the WS signal is generated at the same time as the first data bit, which is the MSBit.

Figure 261. MSB Justified 16-bit or 32-bit full-accuracy length with CPOL = 0



Data are latched on the falling edge of CK (for transmitter) and are read on the rising edge (for the receiver).

Figure 262. MSB Justified 24-bit frame length with CPOL = 0

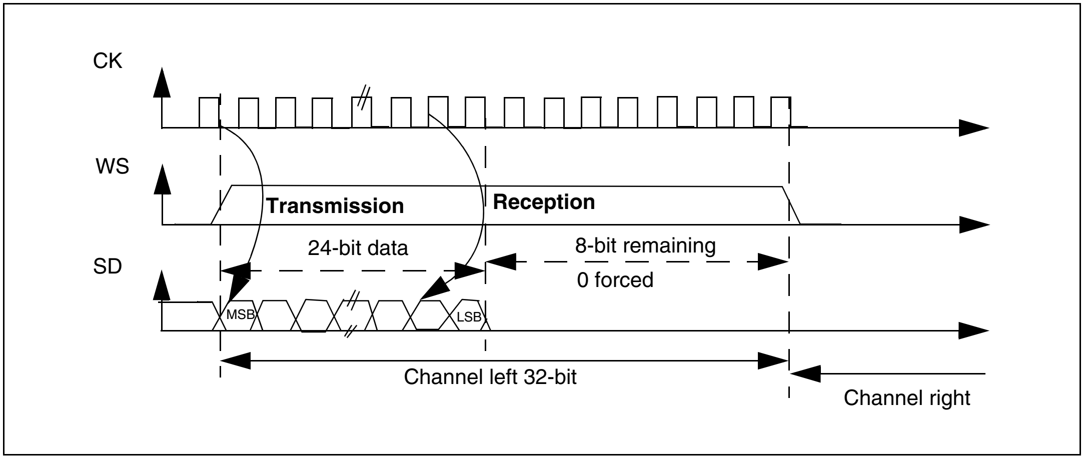


Figure 263. MSB Justified 16-bit extended to 32-bit packet frame with CPOL = 0

LSB justified standard

This standard is similar to the MSB justified standard (no difference for the 16-bit and 32-bit full-accuracy frame formats).

Figure 264. LSB justified 16-bit or 32-bit full-accuracy with CPOL = 0

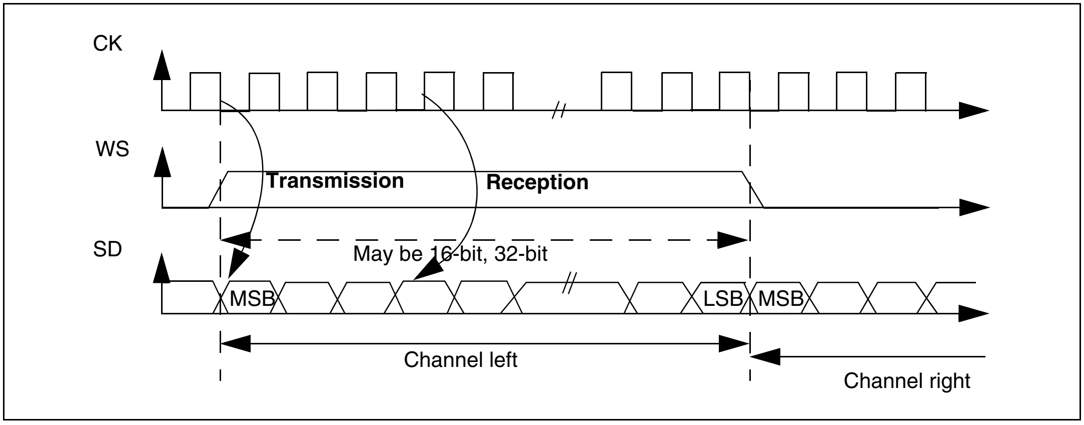
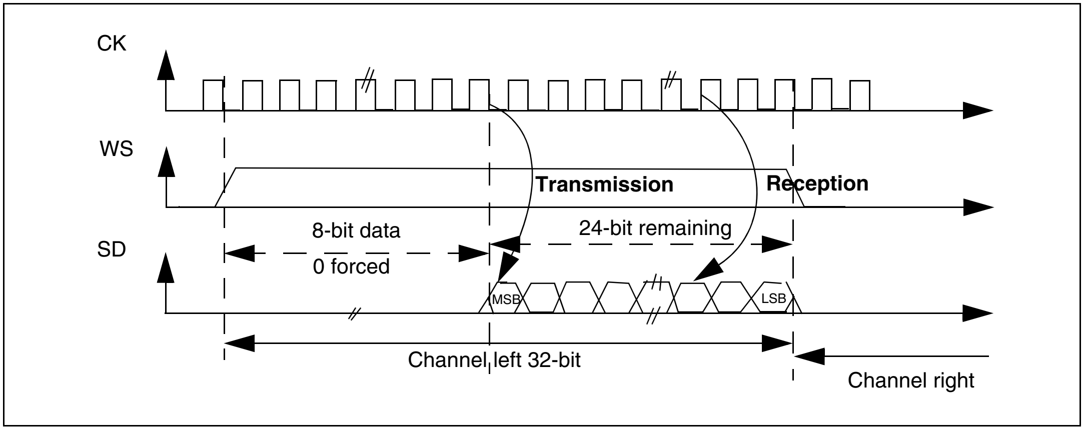


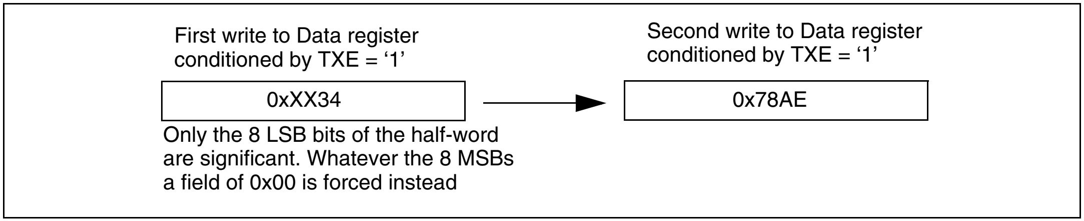
Figure 265. LSB Justified 24-bit frame length with CPOL = 0



- In transmission mode:

  If data 0x3478AE have to be transmitted, two write operations to the SPI_DR register are required from software or by DMA. The operations are shown below.

Figure 266. Operations required to transmit 0x3478AE



First write to Data register conditioned by TXE = ‘1’

0xXX34

Only the 8 LSB bits of the half-word are significant. Whatever the 8 MSBs a field of 0x00 is forced instead

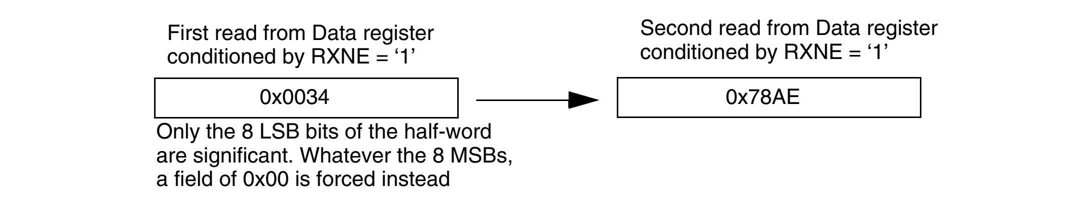
Second write to Data register conditioned by TXE = ‘1’

0x78AE

- In reception mode:

  If data 0x3478AE are received, two successive read operations from SPI_DR are required on each RXNE event.

Figure 267. Operations required to receive 0x3478AE



First read from Data register conditioned by RXNE = ‘1’

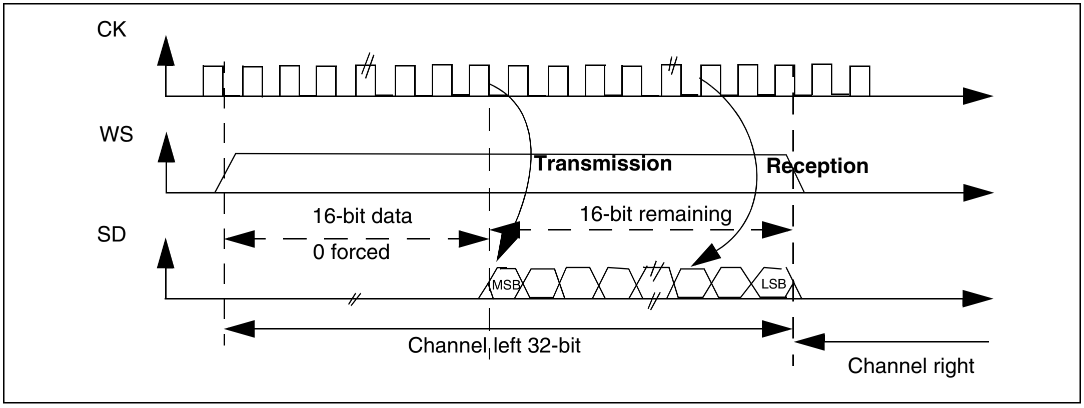
0x0034

Only the 8 LSB bits of the half-word are significant. Whatever the 8 MSBs, a field of 0x00 is forced instead

Second read from Data register conditioned by RXNE = ‘1’

0x78AE

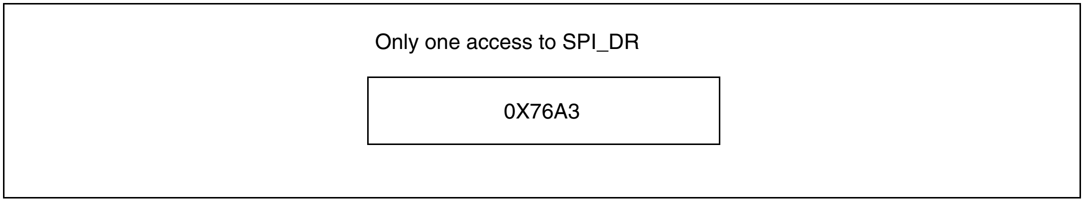
Figure 268. LSB justified 16-bit extended to 32-bit packet frame with CPOL = 0



When 16-bit data frame extended to 32-bit channel frame is selected during the I^{2}S configuration phase, Only one access to SPI_DR is required. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format. In this case it corresponds to the half-word MSB.

If the data to transmit or the received data are 0x76A3 (0x0000 76A3 extended to 32-bit), the operation shown in *Figure 269* is required.

Figure 269. Example of LSB justified 16-bit extended to 32-bit packet frame



In transmission mode, when TXE is asserted, the application has to write the data to be transmitted (in this case 0x76A3). The 0x000 field is transmitted first (extension on 32-bit). TXE is asserted again as soon as the effective data (0x76A3) is sent on SD.

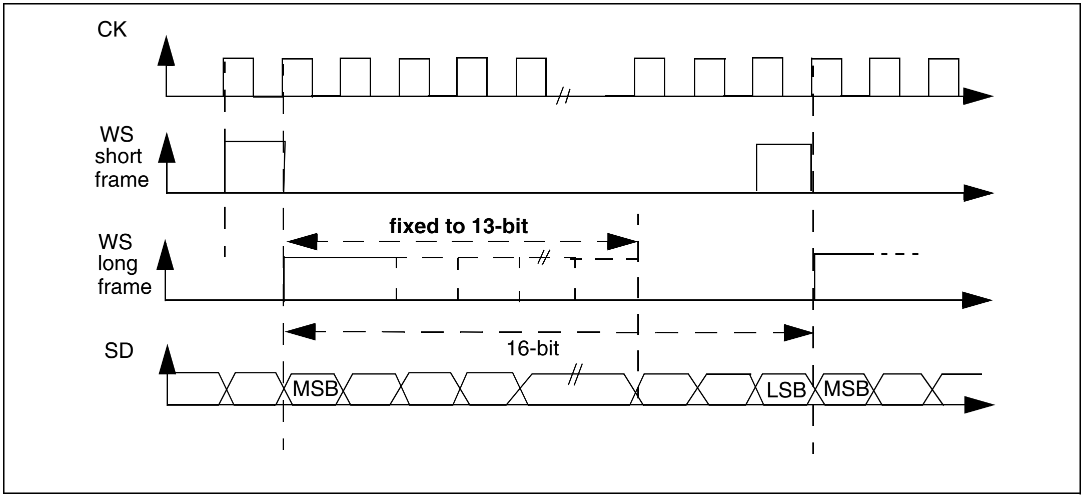
In reception mode, RXNE is asserted as soon as the significant half-word is received (and not the 0x0000 field).

In this way, more time is provided between two write or read operations to prevent underrun or overrun conditions.

PCM standard

For the PCM standard, there is no need to use channel-side information. The two PCM modes (short and long frame) are available and configurable using the PCMSYNC bit in SPI_I2SCFGR.

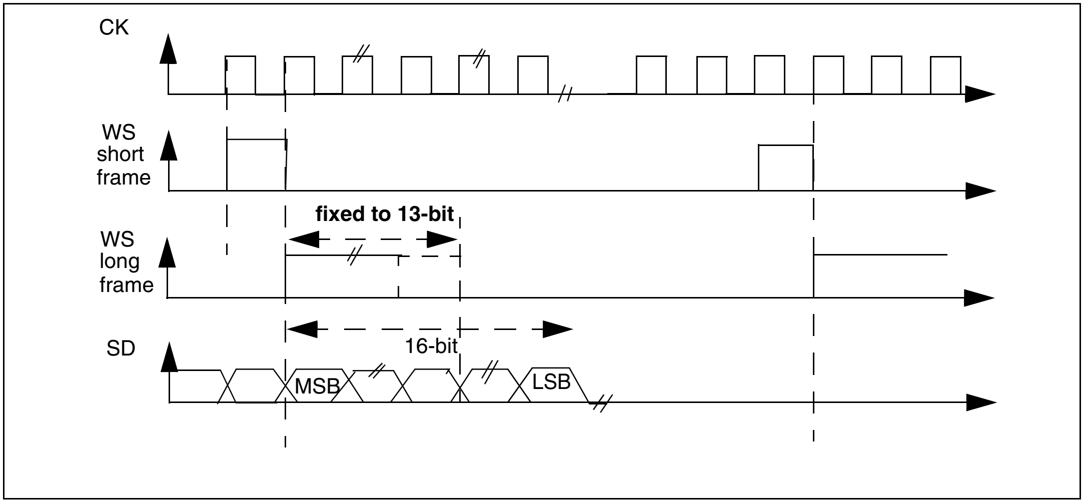
Figure 270. PCM standard waveforms (16-bit)



For long frame synchronization, the WS signal assertion time is fixed 13 bits in master mode.

For short frame synchronization, the WS synchronization signal is only one cycle long.

Figure 271. PCM standard waveforms (16-bit extended to 32-bit packet frame)



Note: For both modes (master and slave) and for both synchronizations (short and long), the number of bits between two consecutive pieces of data (and so two synchronization signals) needs to be specified (DATLEN and CHLEN bits in the SPI_I2SCFGR register) even in slave mode.

27.4.3 Clock generator

The I^2S bitrate determines the dataflow on the I^2S data line and the I^2S clock signal frequency.

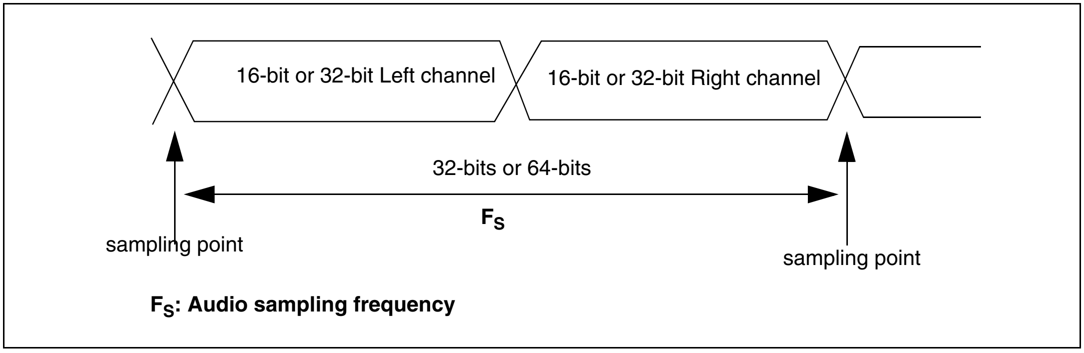
I^2S bitrate = number of bits per channel × number of channels × sampling audio frequency

For a 16-bit audio, left and right channel, the I^2S bitrate is calculated as follows:

I^2S bitrate = $16 \times 2 \times F_S$

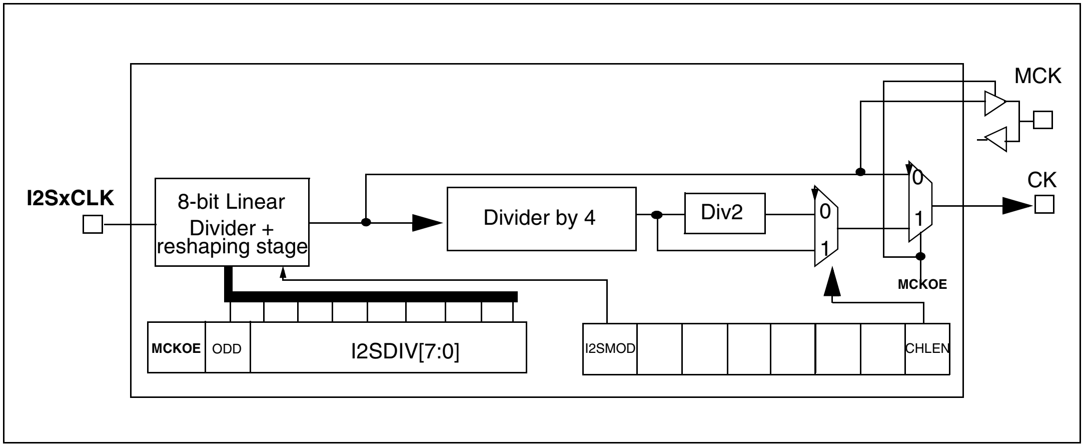
It will be: I^2S bitrate = 32 x 2 x F_S if the packet length is 32-bit wide.

Figure 272. Audio sampling frequency definition


16-bit or 32-bit Left channel
16-bit or 32-bit Right channel
32-bits or 64-bits
FS
sampling point
sampling point
FS: Audio sampling frequency

When the master mode is configured, a specific action needs to be taken to properly program the linear divider in order to communicate with the desired audio frequency.

Figure 273. I^2S clock generator architecture



1. Where x could be 2 or 3.

Figure 272 presents the communication clock architecture.. The I2SxCLK source is the system clock (provided by the HSI, the HSE or the PLL, and sourcing the AHB clock).

The audio sampling frequency can be 96 kHz, 48 kHz, 44.1 kHz, 32 kHz, 22.05 kHz, 16 kHz, 11.025 kHz or 8 kHz (or any other value within this range). In order to reach the desired frequency, the linear divider needs to be programmed according to the formulas below:

When the master clock is generated (MCKOE in the SPI_I2SPR register is set):

F_S = I2SxCLK / [(16*2)*((2*I2SDIV)+ODD)*8)] when the channel frame is 16-bit wide

F_S = I2SxCLK / [(32*2)*((2*I2SDIV)+ODD)*4)] when the channel frame is 32-bit wide

When the master clock is disabled (MCKOE bit cleared):

F_S = I2SxCLK / [(16*2)*((2*I2SDIV)+ODD))] when the channel frame is 16-bit wide

F_S = I2SxCLK / [(32*2)*((2*I2SDIV)+ODD))] when the channel frame is 32-bit wide

Table 146 provides example precision values for different clock configurations.

Note: Other configurations are possible that allow optimum clock precision.

Table 146. Audio-frequency precision using standard 8 MHz HSE (high and medium+ density devices only)

Data length	I2SDIV	I2SODD	MCLK	Target fs(Hz)	Real fs (kHz)	Error
16	5	0	No	96000	100	4.1667%
32	2	0	No	96000	100	4.1667%
16	10	1	No	48000	47.619	0.7937%
32	5	0	No	48000	50	4.1667%
16	11	1	No	44100	43.478	1.4098%
32	5	1	No	44100	45.454	3.0715%
16	15	1	No	32000	32.258	0.8065%
32	8	0	No	32000	31.25	2.3430%
16	22	1	No	22050	22.222	0.7811%
32	11	1	No	22050	21.739	1.4098%
16	31	1	No	16000	15.873	0.7937%
32	15	1	No	16000	16.129	0.8065%
16	45	1	No	11025	10.989	0.3264%
32	22	1	No	11025	11.111	0.7811%
16	62	1	No	8000	8	0.0000%
32	31	1	No	8000	7.936	0.7937%
16	2	0	Yes	32000	31.25	2.3430%
32	2	0	Yes	32000	31.25	2.3430%
16	3	0	Yes	22050	20.833	5.5170%
32	3	0	Yes	22050	20.833	5.5170%
16	4	0	Yes	16000	15.625	2.3428%
32	4	0	Yes	16000	15.625	2.3428%
16	5	1	Yes	11025	11.363	3.0715%
32	5	1	Yes	11025	11.363	3.0715%
16	8	0	Yes	8000	7.812	2.3428%
32	8	0	Yes	8000	7.812	2.3428%

Note: This table gives only example values for different clock configurations. Other configurations with a dedicated HSE clock value allow optimum clock precision.

To get 0 error precision, the I^{2}S sampling rate should be based on Real fs instead of Target fs.

27.4.4 I^{2}S master mode

The I^{2}S can be configured in master mode for transmission and reception. This means that the serial clock is generated on the CK pin as well as the Word Select signal WS. Master clock (MCK) may be output or not, thanks to the MCKOE bit in the SPI_I2SPR register.

Procedure

1. Select the I2SDIV[7:0] bits in the SPI_I2SPR register to define the serial clock baud rate to reach the proper audio sample frequency. The ODD bit in the SPI_I2SPR register also has to be defined.
2. Select the CKPOL bit to define the steady level for the communication clock. Set the MCKOE bit in the SPI_I2SPR register if the master clock MCK needs to be provided to the external DAC/ADC audio component (the I2SDIV and ODD values should be computed depending on the state of the MCK output, for more details refer to *Section 27.4.3: Clock generator*).
3. Set the I2SMOD bit in SPI_I2SCFGR to activate the I^{2}S functionalities and choose the I^{2}S standard through the I2SSTD[1:0] and PCMSYNC bits, the data length through the DATLEN[1:0] bits and the number of bits per channel by configuring the CHLEN bit. Select also the I^{2}S master mode and direction (Transmitter or Receiver) through the I2SCFG[1:0] bits in the SPI_I2SCFGR register.
4. If needed, select all the potential interruption sources and the DMA capabilities by writing the SPI_CR2 register.
5. The I2SE bit in SPI_I2SCFGR register must be set.

WS and CK are configured in output mode. MCK is also an output, if the MCKOE bit in SPI_I2SPR is set.

Transmission sequence

The transmission sequence begins when a half-word is written into the Tx buffer.

Assumedly, the first data written into the Tx buffer correspond to the channel Left data. When data are transferred from the Tx buffer to the shift register, TXE is set and data corresponding to the channel Right have to be written into the Tx buffer. The CHSIDE flag indicates which channel is to be transmitted. It has a meaning when the TXE flag is set because the CHSIDE flag is updated when TXE goes high.

A full frame has to be considered as a Left channel data transmission followed by a Right channel data transmission. It is not possible to have a partial frame where only the left channel is sent.

The data half-word is parallel loaded into the 16-bit shift register during the first bit transmission, and then shifted out, serially, to the MOSI/SD pin, MSB first. The TXE flag is set after each transfer from the Tx buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPI_CR2 register is set.

For more details about the write operations depending on the I^{2}S standard mode selected, refer to *Section 27.4.2: Supported audio protocols*).

To ensure a continuous audio data transmission, it is mandatory to write the SPI_DR with the next data to transmit before the end of the current transmission.

To switch off the I^{2}S, by clearing I2SE, it is mandatory to wait for TXE = 1 and BSY = 0.

Reception sequence

The operating mode is the same as for the transmission mode except for the point 3 (refer to the procedure described in *Section 27.4.4: I2S master mode*), where the configuration should set the master reception mode through the I2SCFG[1:0] bits.

Whatever the data or channel length, the audio data are received by 16-bit packets. This means that each time the Rx buffer is full, the RXNE flag is set and an interrupt is generated if the RXNEIE bit is set in SPI_CR2 register. Depending on the data and channel length configuration, the audio value received for a right or left channel may result from one or two receptions into the Rx buffer.

Clearing the RXNE bit is performed by reading the SPI_DR register.

CHSIDE is updated after each reception. It is sensitive to the WS signal generated by the I^{2}S cell.

For more details about the read operations depending on the I^{2}S standard mode selected, refer to *Section 27.4.2: Supported audio protocols*.

If data are received while the previously received data have not been read yet, an overrun is generated and the OVR flag is set. If the ERRIE bit is set in the SPI_CR2 register, an interrupt is generated to indicate the error.

To switch off the I^{2}S, specific actions are required to ensure that the I^{2}S completes the transfer cycle properly without initiating a new data transfer. The sequence depends on the configuration of the data and channel lengths, and on the audio protocol mode selected. In the case of:

- 16-bit data length extended on 32-bit channel length (DATLEN = 00 and CHLEN = 1) using the LSB justified mode (I2SSTD = 10)
  a) Wait for the second to last RXNE = 1 (n – 1)
  b) Then wait 17 I^{2}S clock cycles (using a software loop)
  c) Disable the I^{2}S (I2SE = 0)
- 16-bit data length extended on 32-bit channel length (DATLEN = 00 and CHLEN = 1) in MSB justified, I^{2}S or PCM modes (I2SSTD = 00, I2SSTD = 01 or I2SSTD = 11, respectively)
  a) Wait for the last RXNE
  b) Then wait 1 I^{2}S clock cycle (using a software loop)
  c) Disable the I^{2}S (I2SE = 0)
- For all other combinations of DATLEN and CHLEN, whatever the audio mode selected through the I2SSTD bits, carry out the following sequence to switch off the I^{2}S:
  a) Wait for the second to last RXNE = 1 (n – 1)
  b) Then wait one I^{2}S clock cycle (using a software loop)
  c) Disable the I^{2}S (I2SE = 0)

Note: The BSY flag is kept low during transfers.

27.4.5 I^{2}S slave mode

In slave mode, the I^{2}S can be configured in transmission or reception mode.The operating mode is following mainly the same rules as described for the I^{2}S master configuration. In slave mode, there is no clock to be generated by the I^{2}S interface. The clock and WS signals are input from the external master connected to the I^{2}S interface. There is then no need, for the user, to configure the clock.

The configuration steps to follow are listed below:

1. Set the I2SMOD bit in the SPI_I2SCFGR register to reach the I^{2}S functionalities and choose the I^{2}S standard through the I2SSTD[1:0] bits, the data length through the DATLEN[1:0] bits and the number of bits per channel for the frame configuring the CHLEN bit. Select also the mode (transmission or reception) for the slave through the I2SCFG[1:0] bits in SPI_I2SCFGR register.
2. If needed, select all the potential interrupt sources and the DMA capabilities by writing the SPI_CR2 register.
3. The I2SE bit in SPI_I2SCFGR register must be set.

Transmission sequence

The transmission sequence begins when the external master device sends the clock and when the WS signal requests the transfer of data. The slave has to be enabled before the external master starts the communication. The I^{2}S data register has to be loaded before the master initiates the communication.

For the I^{2}S, MSB justified and LSB justified modes, the first data item to be written into the data register corresponds to the data for the left channel. When the communication starts, the data are transferred from the Tx buffer to the shift register. The TXE flag is then set in order to request the right channel data to be written into the I^{2}S data register.

The CHSIDE flag indicates which channel is to be transmitted. Compared to the master transmission mode, in slave mode, CHSIDE is sensitive to the WS signal coming from the external master. This means that the slave needs to be ready to transmit the first data before the clock is generated by the master. WS assertion corresponds to left channel transmitted first.

Note: The I2SE has to be written at least two PCLK cycles before the first clock of the master comes on the CK line.

The data half-word is parallel-loaded into the 16-bit shift register (from the internal bus) during the first bit transmission, and then shifted out serially to the MOSI/SD pin MSB first. The TXE flag is set after each transfer from the Tx buffer to the shift register and an interrupt is generated if the TXEIE bit in the SPI_CR2 register is set.

Note that the TXE flag should be checked to be at 1 before attempting to write the Tx buffer.

For more details about the write operations depending on the I^{2}S standard mode selected, refer to *Section 27.4.2: Supported audio protocols*.

To secure a continuous audio data transmission, it is mandatory to write the SPI_DR register with the next data to transmit before the end of the current transmission. An underrun flag is set and an interrupt may be generated if the data are not written into the SPI_DR register before the first clock edge of the next data communication. This indicates to the software that the transferred data are wrong. If the ERRIE bit is set into the SPI_CR2 register, an interrupt is generated when the UDR flag in the SPI_SR register goes high. In this case, it is mandatory to switch off the I^{2}S and to restart a data transfer starting from the left channel.

To switch off the I^{2}S, by clearing the I2SE bit, it is mandatory to wait for TXE = 1 and BSY = 0.

Reception sequence

The operating mode is the same as for the transmission mode except for the point 1 (refer to the procedure described in *Section 27.4.5: I2S slave mode*), where the configuration should set the master reception mode using the I2SCFG[1:0] bits in the SPI_I2SCFGR register.

Whatever the data length or the channel length, the audio data are received by 16-bit packets. This means that each time the RX buffer is full, the RXNE flag in the SPI_SR register is set and an interrupt is generated if the RXNEIE bit is set in the SPI_CR2 register. Depending on the data length and channel length configuration, the audio value received for a right or left channel may result from one or two receptions into the RX buffer.

The CHSIDE flag is updated each time data are received to be read from SPI_DR. It is sensitive to the external WS line managed by the external master component.

Clearing the RXNE bit is performed by reading the SPI_DR register.

For more details about the read operations depending the I^{2}S standard mode selected, refer to *Section 27.4.2: Supported audio protocols*.

If data are received while the precedent received data have not yet been read, an overrun is generated and the OVR flag is set. If the bit ERRIE is set in the SPI_CR2 register, an interrupt is generated to indicate the error.

To switch off the I^{2}S in reception mode, I2SE has to be cleared immediately after receiving the last RXNE = 1.

Note: The external master components should have the capability of sending/receiving data in 16-bit or 32-bit packets via an audio channel.

27.4.6 Status flags

Three status flags are provided for the application to fully monitor the state of the I^{2}S bus.

Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing to this flag has no effect). It indicates the state of the communication layer of the I^{2}S.

When BSY is set, it indicates that the I^{2}S is busy communicating. There is one exception in master receive mode (I2SCFG = 11) where the BSY flag is kept low during reception.

The BSY flag is useful to detect the end of a transfer if the software needs to disable the I^{2}S. This avoids corrupting the last transfer. For this, the procedure described below must be strictly respected.

The BSY flag is set when a transfer starts, except when the I^{2}S is in master receiver mode.

The BSY flag is cleared:

- when a transfer completes (except in master transmit mode, in which the communication is supposed to be continuous)
- when the I^{2}S is disabled

When communication is continuous:

- In master transmit mode, the BSY flag is kept high during all the transfers
- In slave mode, the BSY flag goes low for one I^{2}S clock cycle between each transfer

Note: Do not use the BSY flag to handle each data transmission or reception. It is better to use the TXE and RXNE flags instead.

Tx buffer empty flag (TXE)

When set, this flag indicates that the Tx buffer is empty and the next data to be transmitted can then be loaded into it. The TXE flag is reset when the Tx buffer already contains data to be transmitted. It is also reset when the I^{2}S is disabled (I2SE bit is reset).

RX buffer not empty (RXNE)

When set, this flag indicates that there are valid received data in the RX Buffer. It is reset when SPI_DR register is read.

Channel Side flag (CHSIDE)

In transmission mode, this flag is refreshed when TXE goes high. It indicates the channel side to which the data to transfer on SD has to belong. In case of an underrun error event in slave transmission mode, this flag is not reliable and I^{2}S needs to be switched off and switched on before resuming the communication.

In reception mode, this flag is refreshed when data are received into SPI_DR. It indicates from which channel side data have been received. Note that in case of error (like OVR) this flag becomes meaningless and the I^{2}S should be reset by disabling and then enabling it (with configuration if it needs changing).

This flag has no meaning in the PCM standard (for both Short and Long frame modes).

When the OVR or UDR flag in the SPI_SR is set and the ERRIE bit in SPI_CR2 is also set, an interrupt is generated. This interrupt can be cleared by reading the SPI_SR status register (once the interrupt source has been cleared).

27.4.7 Error flags

There are three error flags for the I^{2}S cell.

Underrun flag (UDR)

In slave transmission mode this flag is set when the first clock for data transmission appears while the software has not yet loaded any value into SPI_DR. It is available when the I2SMOD bit in SPI_I2SCFGR is set. An interrupt may be generated if the ERRIE bit in SPI_CR2 is set.
The UDR bit is cleared by a read operation on the SPI_SR register.

Overrun flag (OVR)

This flag is set when data are received and the previous data have not yet been read from SPI_DR. As a result, the incoming data are lost. An interrupt may be generated if the ERRIE bit is set in SPI_CR2.

In this case, the receive buffer contents are not updated with the newly received data from the transmitter device. A read operation to the SPI_DR register returns the previous correctly received data. All other subsequently transmitted half-words are lost.

Clearing the OVR bit is done by a read operation on the SPI_DR register followed by a read access to the SPI_SR register.

Frame error flag (FRE)

This flag can be set by hardware only if the I2S is configured in Slave mode. It is set if the external master is changing the WS line at a moment when the slave is not expected this change. If the synchronization is lost, to recover from this state and resynchronize the external master device with the I2S slave device, follow the steps below:

1. Disable the I2S
2. Re-enable it when the correct level is detected on the WS line (WS line is high in I2S mode, or low for MSB- or LSB-justified or PCM modes).

Desynchronization between the master and slave device may be due to noisy environment on the SCK communication clock or on the WS frame synchronization line. An error interrupt can be generated if the ERRIE bit is set. The desynchronization flag (FRE) is cleared by software when the status register is read.

27.4.8 I^{2}S interrupts

Table 147 provides the list of I^{2}S interrupts.

Table 147. I^{2}S interrupt requests

Interrupt event	Event flag	Enable Control bit
Transmit buffer empty flag	TXE	TXEIE
Receive buffer not empty flag	RXNE	RXNEIE
Overrun error	OVR	ERRIE
Underrun error	UDR	
Frame error flag	FRE	FRE

27.5 SPI and I^{2}S registers

Refer to *Section 1.1 on page 37*for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16 bits) or words (32 bits).

27.5.1 SPI control register 1 (SPI_CR1)(not used in I^{2}S mode)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDI OE	CRC EN	CRC NEXT	DFF	RX ONLY	SSM	SSI	*LSB FIRST*	SPE	BR [2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 ***BIDIMODE:*** Bidirectional data mode enable

0: 2-line unidirectional data mode selected

1: 1-line bidirectional data mode selected

Note: ***Not used in I^{2}S mode***

Bit 14 **BIDIOE:** Output enable in bidirectional mode

This bit combined with the BIDImode bit selects the direction of transfer in bidirectional mode

0: Output disabled (receive-only mode)

1: Output enabled (transmit-only mode)

Note: In master mode, the MOSI pin is used and in slave mode, the MISO pin is used.

Not used in I^{2}S mode

Bit 13 **CRCEN:** Hardware CRC calculation enable

0: CRC calculation disabled

1: CRC calculation enabled

Note: This bit should be written only when SPI is disabled (SPE = ‘0’) for correct operation

Not used in I^{2}S mode

Bit 12 **CRCNEXT:** CRC transfer next

0: Data phase (no CRC phase)

1: Next transfer is CRC (CRC phase)

Note: When the SPI is configured in full duplex or transmitter only modes, CRCNEXT must be written as soon as the last data is written to the SPI_DR register.
When the SPI is configured in receiver only mode, CRCNEXT must be set after the second last data reception.
This bit should be kept cleared when the transfers are managed by DMA.

Not used in I^{2}S mode

Bit 11 **DFF:** Data frame format

0: 8-bit data frame format is selected for transmission/reception

1: 16-bit data frame format is selected for transmission/reception

Note: This bit should be written only when SPI is disabled (SPE = ‘0’) for correct operation

Not used in I^{2}S mode

Bit 10 **RXONLY:** Receive only

This bit combined with the BIDImode bit selects the direction of transfer in 2-line unidirectional mode. This bit is also useful in a multislave system in which this particular slave is not accessed, the output from the accessed slave is not corrupted.

0: Full duplex (Transmit and receive)

1: Output disabled (Receive-only mode)

Note: ***Not used in I^{2}S mode***

Bit 9 **SSM:** Software slave management

When the SSM bit is set, the NSS pin input is replaced with the value from the SSI bit.

0: Software slave management disabled

1: Software slave management enabled

Note: ***Not used in I^{2}S mode and SPI TI mode***

Bit 8 **SSI:** Internal slave select

This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the NSS pin and the IO value of the NSS pin is ignored.

Note: ***Not used in I^{2}S mode and SPI TI mode***

Bit 7 ***LSBFIRST:*** Frame format

0: MSB transmitted first

1: LSB transmitted first

Note: *This bit should not be changed when communication is ongoing.*

Not used in I^{2}S mode and SPI TI mode

Bit 6 **SPE:** SPI enable

0: Peripheral disabled

1: Peripheral enabled

Note: ***1- Not used in I^{2}S mode.***

Note: *2- When disabling the SPI, follow the procedure described in Section 27.3.8: Disabling the SPI.*

Bits 5:3 **BR[2:0]:** Baud rate control

000: f_{PCLK}/2100: f_{PCLK}/32
001: f_{PCLK}/4101: f_{PCLK}/64
010: f_{PCLK}/8110: f_{PCLK}/128
011: f_{PCLK}/16111: f_{PCLK}/256

Note: *These bits should not be changed when communication is ongoing.*

Not used in I^{2}S mode

Bit 2 **MSTR:** Master selection

0: Slave configuration

1: Master configuration

Note: This bit should not be changed when communication is ongoing.

Not used in I^{2}S mode

Bit1 **CPOL:** Clock polarity

0: CK to 0 when idle

1: CK to 1 when idle

Note: This bit should not be changed when communication is ongoing.

Not used in I^{2}S mode and SPI TI mode

Bit 0 **CPHA:** Clock phase

0: The first clock transition is the first data capture edge

1: The second clock transition is the first data capture edge

Note: This bit should not be changed when communication is ongoing.

Note: ***Not used in I^{2}S mode and SPI TI mode***

27.5.2 SPI control register 2 (SPI_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TXEIE	RXNEIE	ERRIE	FRF	Res.	*SSOE*	TXDMAEN	RXDMAEN
								rw	rw	rw	rw		rw	rw	rw

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

Bit 7 **TXEIE:** Tx buffer empty interrupt enable

0: TXE interrupt masked

1: TXE interrupt not masked. Used to generate an interrupt request when the TXE flag is set.

Bit 6 **RXNEIE:** RX buffer not empty interrupt enable

0: RXNE interrupt masked

1: RXNE interrupt not masked. Used to generate an interrupt request when the RXNE flag is set.

Bit 5 **ERRIE:** Error interrupt enable

This bit controls the generation of an interrupt when an error condition occurs (CRCERR, OVR, MODF, FRE in SPI mode and UDR, OVR, FRE in I^{2}S mode).

0: Error interrupt is masked

1: Error interrupt is enabled

Bit 4 **FRF**: Frame format

0: SPI Motorola mode

1 SPI TI mode (in high and medium+ density devices)

Note: Not used ***in I^{2}S mode***

Bit 3 Reserved. Forced to 0 by hardware.

Bit 2 **SSOE:** SS output enable

0: SS output is disabled in master mode and the cell can work in multimaster configuration

1: SS output is enabled in master mode and when the cell is enabled. The cell cannot work in a multimaster environment.

Note: ***Not used in I²S mode and SPI TI mode***

Bit 1 **TXDMAEN:** Tx buffer DMA enable

When this bit is set, the DMA request is made whenever the TXE flag is set.

0: Tx buffer DMA disabled

1: Tx buffer DMA enabled

Bit 0 **RXDMAEN:** Rx buffer DMA enable

When this bit is set, the DMA request is made whenever the RXNE flag is set.

0: Rx buffer DMA disabled

1: Rx buffer DMA enabled

27.5.3 SPI status register (SPI_SR)

Address offset: 0x08

Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							FRE	BSY	OVR	MODF	CRC ERR	UDR	CHSID E	TXE	RXNE
							r	r	r	r	rc_w0	r	r	r	r

Bits 15:9 Reserved. Forced to 0 by hardware.

Bit 8 **FRE**: Frame Error

0: No frame error

1: Frame error occurred.

This bit is set by hardware and cleared by software when the SPI_SR register is read.

This bit is used in SPI TI mode or in I2S mode whatever the audio protocol selected. It detects a change on NSS or WS line which takes place in slave mode at a non expected time, informing about a desynchronization between the external master device and the slave.

Bit 7 ***BSY:*** Busy flag

0: SPI(or I2S) not busy

1: SPI(or I2S) is busy in communication or Tx buffer is not empty

This flag is set and cleared by hardware.

Note: BSY flag must be used with caution: refer to Section 27.3.7: Status flags and Section 27.3.8: Disabling the SPI.

Bit 6 **OVR:** Overrun flag

0: No overrun occurred

1: Overrun occurred

This flag is set by hardware and reset by a software sequence.Refer to *Section 27.4.7 on page 742* for the software sequence.

Bit 5 **MODF:** Mode fault

0: No mode fault occurred

1: Mode fault occurred

This flag is set by hardware and reset by a software sequence. Refer to *Section 27.3.10 on page 725* for the software sequence.

Note: Not used in I^{2}S mode

Bit 4 **CRCERR:** CRC error flag

0: CRC value received matches the SPI_RXCRCR value

1: CRC value received does not match the SPI_RXCRCR value

This flag is set by hardware and cleared by software writing 0.

Note: Not used in I^{2}S mode

Bit 3 **UDR:** Underrun flag

0: No underrun occurred

1: Underrun occurred

This flag is set by hardware and reset by a software sequence. Refer to *Section 27.4.7 on page 742* for the software sequence.

Note: Not used in SPI mode

Bit 2 **CHSIDE**: Channel side

0: Channel Left has to be transmitted or has been received
1: Channel Right has to be transmitted or has been received

Note: Not used for SPI mode. No meaning in PCM mode

Bit 1 **TXE:** Transmit buffer empty

0: Tx buffer not empty
1: Tx buffer empty

Bit 0 **RXNE:** Receive buffer not empty

0: Rx buffer empty
1: Rx buffer not empty

27.5.4 SPI data register (SPI_DR)

Address offset: 0x0C

Reset value: 0x0000

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

Bits 15:0 **DR[15:0]:** Data register

Data received or to be transmitted.

The data register is split into 2 buffers - one for writing (Transmit Buffer) and another one for reading (Receive buffer). A write to the data register will write into the Tx buffer and a read from the data register will return the value held in the Rx buffer.

Notes for the SPI mode:

Depending on the data frame format selection bit (DFF in SPI_CR1 register), the data sent or received is either 8-bit or 16-bit. This selection has to be made before enabling the SPI to ensure correct operation.

For an 8-bit data frame, the buffers are 8-bit and only the LSB of the register (SPI_DR[7:0]) is used for transmission/reception. When in reception mode, the MSB of the register (SPI_DR[15:8]) is forced to 0.

For a 16-bit data frame, the buffers are 16-bit and the entire register, SPI_DR[15:0] is used for transmission/reception.

27.5.5 SPI CRC polynomial register (SPI_CRCPR)(not used in I^{2}S mode)

Address offset: 0x10

Reset value: 0x0007

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

Bits 15:0 **CRCPOLY[15:0]:** CRC polynomial register

This register contains the polynomial for the CRC calculation.

The CRC polynomial (0007h) is the reset value of this register. Another polynomial can be configured as required.

Note: Not used for the I^{2}S mode.

27.5.6 SPI RX CRC register (SPI_RXCRCR)(not used in I^{2}S mode)

Address offset: 0x14

Reset value: 0x0000

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

Bits 15:0 **RXCRC[15:0]:** Rx CRC register

When CRC calculation is enabled, the RxCRC[15:0] bits contain the computed CRC value of the subsequently received bytes. This register is reset when the CRCEN bit in SPI_CR1 register is written to 1. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPR register.

Only the 8 LSB bits are considered when the data frame format is set to be 8-bit data (DFF bit of SPI_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit data frame format is selected (DFF bit of the SPI_CR1 register is set). CRC calculation is done based on any CRC16 standard.

Note: A read to this register when the BSY Flag is set could return an incorrect value.
Not used for I^{2}S mode.

27.5.7 SPI TX CRC register (SPI_TXCRCR)(not used in I^{2}S mode)

Address offset: 0x18

Reset value: 0x0000

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

Bits 15:0 **TXCRC[15:0]:** Tx CRC register

When CRC calculation is enabled, the TxCRC[7:0] bits contain the computed CRC value of the subsequently transmitted bytes. This register is reset when the CRCEN bit of SPI_CR1 is written to 1. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPR register.

Only the 8 LSB bits are considered when the data frame format is set to be 8-bit data (DFF bit of SPI_CR1 is cleared). CRC calculation is done based on any CRC8 standard.

The entire 16-bits of this register are considered when a 16-bit data frame format is selected (DFF bit of the SPI_CR1 register is set). CRC calculation is done based on any CRC16 standard.

Note: A read to this register when the BSY flag is set could return an incorrect value.
Not used for I^{2}S mode.

27.5.8 SPI_I^{2}S configuration register (SPI_I2SCFGR)

Address offset: 0x1C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				I2SMOD	I2SE	I2SCFG		PCMSYNC	Res.	I2SSTD		CKPOL	DATLEN		CHLEN
				rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **I2SMOD**: I2S mode selection

0: SPI mode is selected

1: I2S mode is selected

Note: This bit should be configured when the SPI or I^{2}S is disabled

Bit 10 **I2SE**: I2S Enable

0: I^{2}S peripheral is disabled

1: I^{2}S peripheral is enabled

Note: Not used in SPI mode

Bits 9:8 **I2SCFG**: I2S configuration mode

00: Slave - transmit

01: Slave - receive

10: Master - transmit

11: Master - receive

Note: This bit should be configured when the I^{2}S is disabled.

Not used in SPI mode

Bit 7 **PCMSYNC**: PCM frame synchronization

0: Short frame synchronization

1: Long frame synchronization

Note: This bit has a meaning only if I2SSTD = 11 (PCM standard is used)

Not used in SPI mode

Bit 6 Reserved: forced at 0 by hardware

Bits 5:4 **I2SSTD**: I2S standard selection

00: I^{2}S Philips standard.

01: MSB justified standard (left justified)

10: LSB justified standard (right justified)

11: PCM standard

For more details on I^{2}S standards, refer to *Section 27.4.2 on page 728*. *Not used in SPI mode.*

Note: For correct operation, these bits should be configured when the I^{2}S is disabled.

Bit 3 **CKPOL**: Steady state clock polarity

0: I^{2}S clock steady state is low level

1: I^{2}S clock steady state is high level

Note: For correct operation, this bit should be configured when the I^{2}S is disabled.

Not used in SPI mode

Bits 2:1 **DATLEN**: Data length to be transferred

00: 16-bit data length

01: 24-bit data length

10: 32-bit data length

11: Not allowed

Note: For correct operation, these bits should be configured when the I^{2}S is disabled.

Not used in SPI mode.

Bit 0 **CHLEN**: Channel length (number of bits per audio channel)

0: 16-bit wide

1: 32-bit wide

The bit write operation has a meaning only if DATLEN = 00 otherwise the channel length is fixed to 32-bit by hardware whatever the value filled in. *Not used in SPI mode.*

Note: For correct operation, this bit should be configured when the I^{2}S is disabled.

27.5.9 SPI_I^{2}S prescaler register (SPI_I2SPR)

Address offset: 0x20

Reset value: 0000 0010 (0x0002)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						MCKOE	ODD	I2SDIV							
						rw	rw	rw							

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

Bit 9 **MCKOE**: Master clock output enable

0: Master clock output is disabled

1: Master clock output is enabled

Note: This bit should be configured when the I^{2}S is disabled. It is used only when the I^{2}S is in master mode.

Not used in SPI mode.

Bit 8 **ODD**: Odd factor for the prescaler

0: real divider value is = I2SDIV *2

1: real divider value is = (I2SDIV * 2)+1

Refer to Section 27.4.3 on page 735. *Not used in SPI mode.*

Note: This bit should be configured when the I^{2}S is disabled. It is used only when the I^{2}S is in master mode.

Bits 7:0 **I2SDIV**: I2S Linear prescaler

I2SDIV [7:0] = 0 or I2SDIV [7:0] = 1 are forbidden values.

Refer to Section 27.4.3 on page 735. *Not used in SPI mode.*

Note: These bits should be configured when the I^{2}S is disabled. It is used only when the I^{2}S is in master mode.

27.5.10 SPI register map

The table provides shows the SPI register map and reset values.

Table 148. SPI register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SPI_CR1	Reserved																BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXONLY	SSM	SSI	*LSBFIRST*	SPE	BR [2:0]			MSTR	CPOL	CPHA
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	SPI_CR2	Reserved																								TXEIE	RXNEIE	ERRIE	FRF	Reserved	*SSOE*	TXDMAEN	RXDMAEN
	Reset value																									0	0	0	0		0	0	0
0x08	SPI_SR	Reserved																							FRE	BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
	Reset value																								0	0	0	0	0	0	0	1	0
0x0C	SPI_DR	Reserved																DR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SPI_CRCPR	Reserved																CRCPOLY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x14	SPI_RXCRCR	Reserved																RxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	SPI_TXCRCR	Reserved																TxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	SPI_I2SCFGR	Reserved																				I2SMOD	I2SE	I2SCFG		PCMSYNC	Reserved	I2SSTD		CKPOL	DATLEN		CHLEN
	Reset value																					0	0	0	0	0		0	0	0	0	0	0
0x20	SPI_I2SPR	Reserved																						MCKOE	ODD	I2SDIV							
	Reset value																							0	0	0	0	0	0	0	0	1	0

Refer to *Table 2 on page 42* for the register boundary addresses.

28 Secure digital input/output interface (SDIO)

This section applies to high-density devices only.

28.1 SDIO main features

The SD/SDIO MMC card host interface (SDIO) provides an interface between the APB2 peripheral bus and MultiMediaCards (MMCs), SD memory cards, SDIO cards and CE-ATA devices.

The MultiMediaCard system specifications are available through the MultiMediaCard Association website at *www.mmca.org*, published by the MMCA technical committee.

SD memory card and SD I/O card system specifications are available through the SD card Association website at *www.sdcard.org*.

CE-ATA system specifications are available through the CE-ATA workgroup website at *www.ce-ata.org*.

The SDIO features include the following:

- Full compliance with *MultiMediaCard System Specification Version 4.2*. Card support for three different databus modes: 1-bit (default), 4-bit and 8-bit
- Full compatibility with previous versions of MultiMediaCards (forward compatibility)
- Full compliance with *SD Memory Card Specifications Version 2.0*
- Full compliance with *SD I/O Card Specification Version 2.0:* card support for two different databus modes: 1-bit (default) and 4-bit
- Full support of the CE-ATA features (full compliance with *CE-ATA digital protocol Rev1.1*)
- Data transfer up to 48 MHz for the 8 bit mode
- Data and command output enable signals to control external bidirectional drivers.

Note: The SDIO does not have an SPI-compatible communication mode.

The SD memory card protocol is a superset of the MultiMediaCard protocol as defined in the MultiMediaCard system specification V2.11. Several commands required for SD memory devices are not supported by either SD I/O-only cards or the I/O portion of combo cards. Some of these commands have no use in SD I/O devices, such as erase commands, and thus are not supported in the SDIO. In addition, several commands are different between SD memory cards and SD I/O cards and thus are not supported in the SDIO. For details refer to SD I/O card Specification Version 1.0. CE-ATA is supported over the MMC electrical interface using a protocol that utilizes the existing MMC access primitives. The interface electrical and signaling definition is as defined in the MMC reference.

The MultiMediaCard/SD bus connects cards to the controller.

The current version of the SDIO supports only one SD/SDIO/MMC4.2 card at any one time and a stack of MMC4.1 or previous.

28.2 SDIO bus topology

Communication over the bus is based on command and data transfers.

The basic transaction on the MultiMediaCard/SD/SD I/O bus is the command/response transaction. These types of bus transaction transfer their information directly within the command or response structure. In addition, some operations have a data token.

Data transfers to/from SD/SDIO memory cards are done in data blocks. Data transfers to/from MMC are done data blocks or streams. Data transfers to/from the CE-ATA Devices are done in data blocks.

Figure 274. SDIO “no response” and “no data” operations

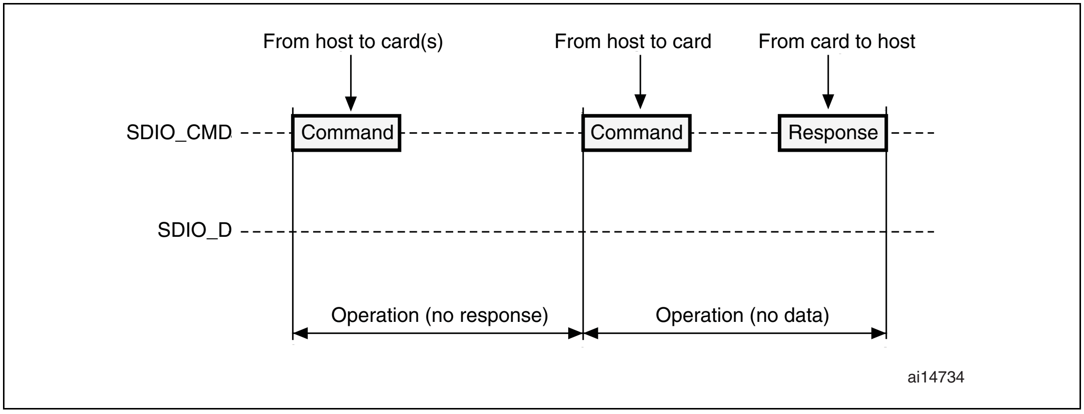


Figure 275. SDIO (multiple) block read operation

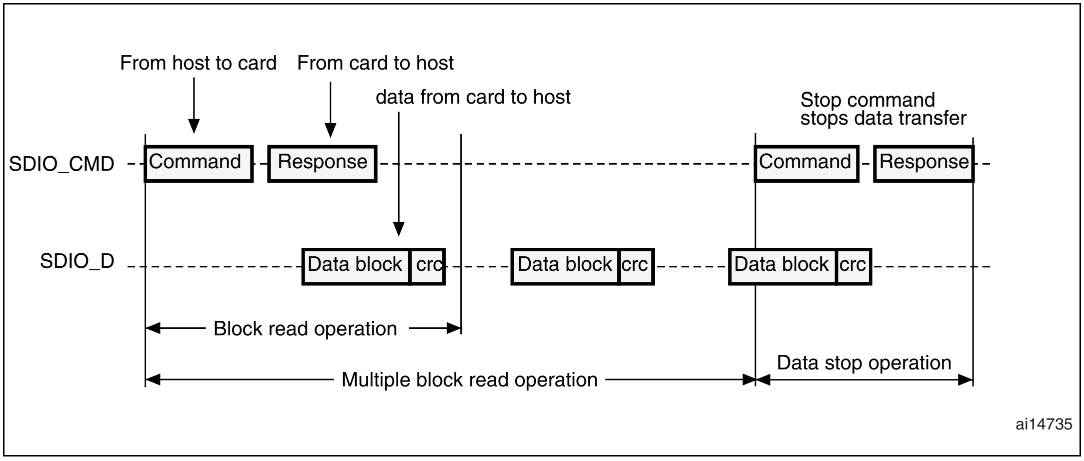
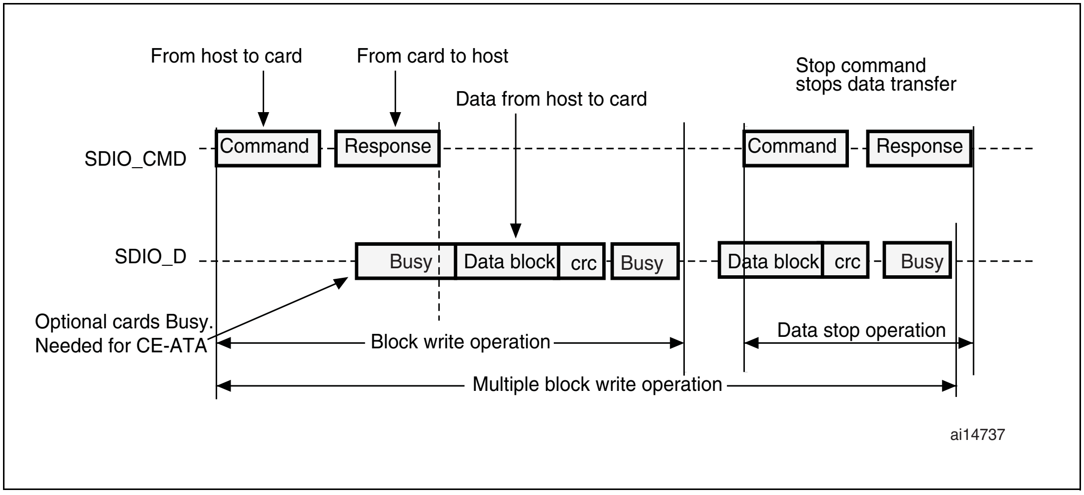


Figure 276. SDIO (multiple) block write operation



Note: The SDIO will not send any data as long as the Busy signal is asserted (SDIO_D0 pulled low).

Figure 277. SDIO sequential read operation

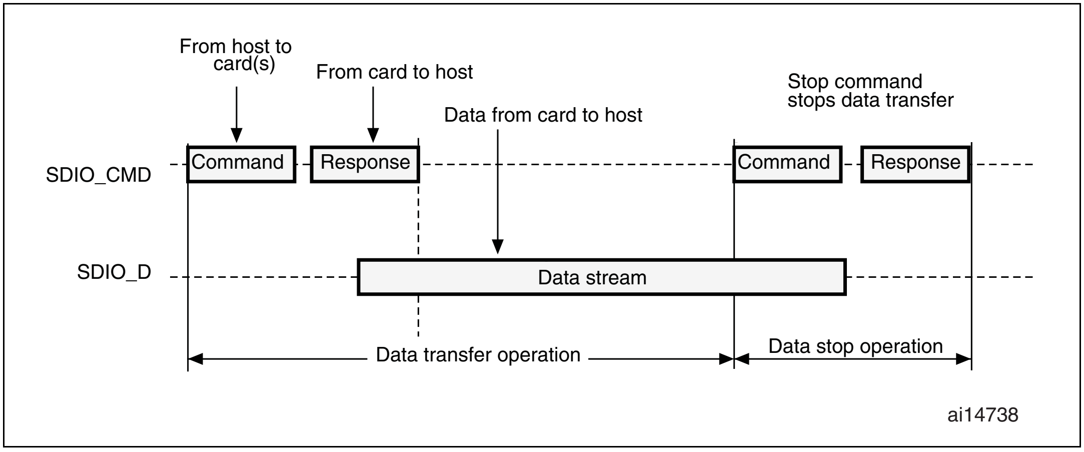
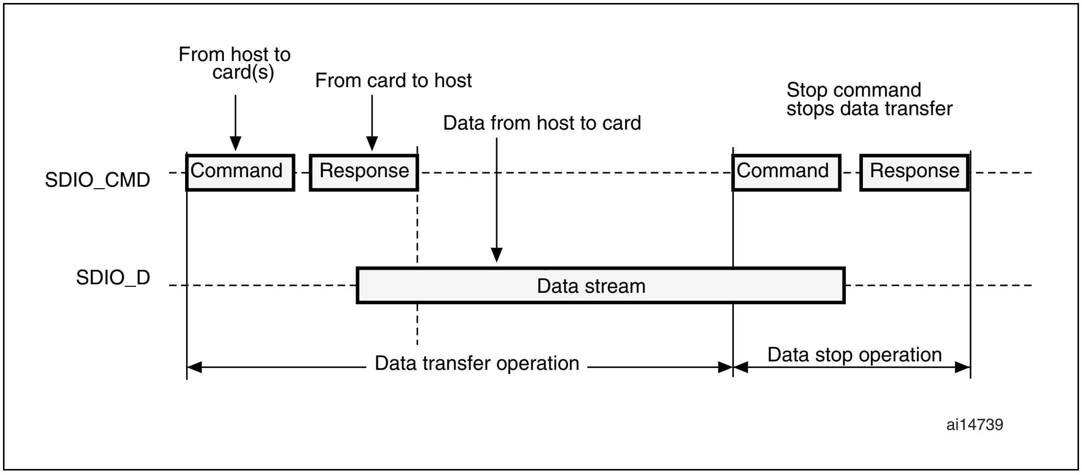


Figure 278. SDIO sequential write operation

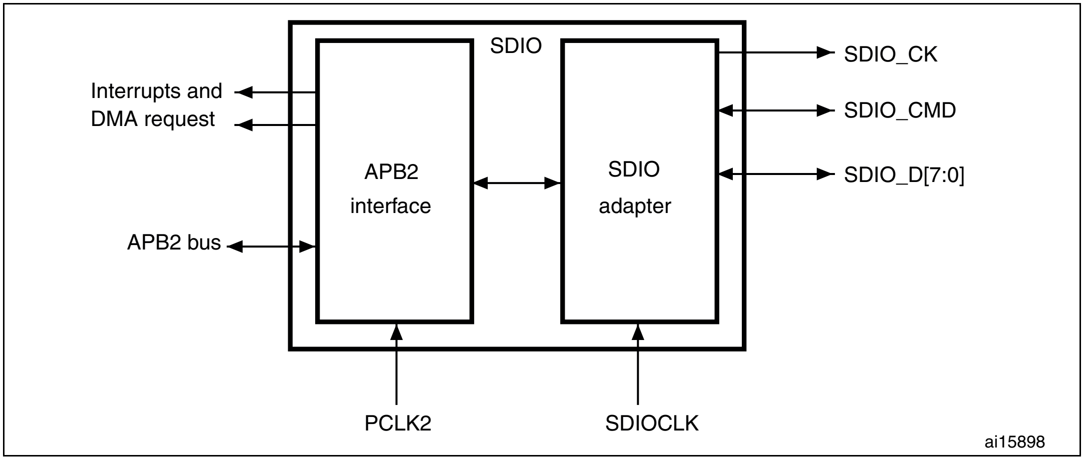


28.3 SDIO functional description

The SDIO consists of two parts:

- The SDIO adapter block provides all functions specific to the MMC/SD/SD I/O card such as the clock generation unit, command and data transfer.
- The APB2 interface accesses the SDIO adapter registers, and generates interrupt and DMA request signals.

Figure 279. SDIO block diagram



By default SDIO_D0 is used for data transfer. After initialization, the host can change the databus width.

If a MultiMediaCard is connected to the bus, SDIO_D0, SDIO_D[3:0] or SDIO_D[7:0] can be used for data transfer. MMC V3.31 or previous, supports only 1 bit of data so only SDIO_D0 can be used.

If an SD or SD I/O card is connected to the bus, data transfer can be configured by the host to use SDIO_D0 or SDIO_D[3:0]. All data lines are operating in push-pull mode.

SDIO_CMD has two operational modes:

- Open-drain for initialization (only for MMCV3.31 or previous)
- Push-pull for command transfer (SD/SD I/O card MMC4.2 use push-pull drivers also for initialization)

SDIO_CK is the clock to the card: one bit is transferred on both command and data lines with each clock cycle. The clock frequency can vary between 0 MHz and 20 MHz (for a MultiMediaCard V3.31), between 0 and 48 MHz for a MultiMediaCard V4.0/4.2, or between 0 and 25 MHz (for an SD/SD I/O card).

The SDIO uses two clock signals:

- SDIO adapter clock (SDIOCLK = 48 MHz)
- APB2 bus clock (PCLK2)

PCLK2 and SDIO_CK clock frequencies must respect the following condition:

$$\text{Frequenc}(\text{PCLK2}) \geq 3 / 8 \times \text{Frequency}(\text{SDIO\_CK})$$

The signals shown in *Table 149* are used on the MultiMediaCard/SD/SD I/O card bus.

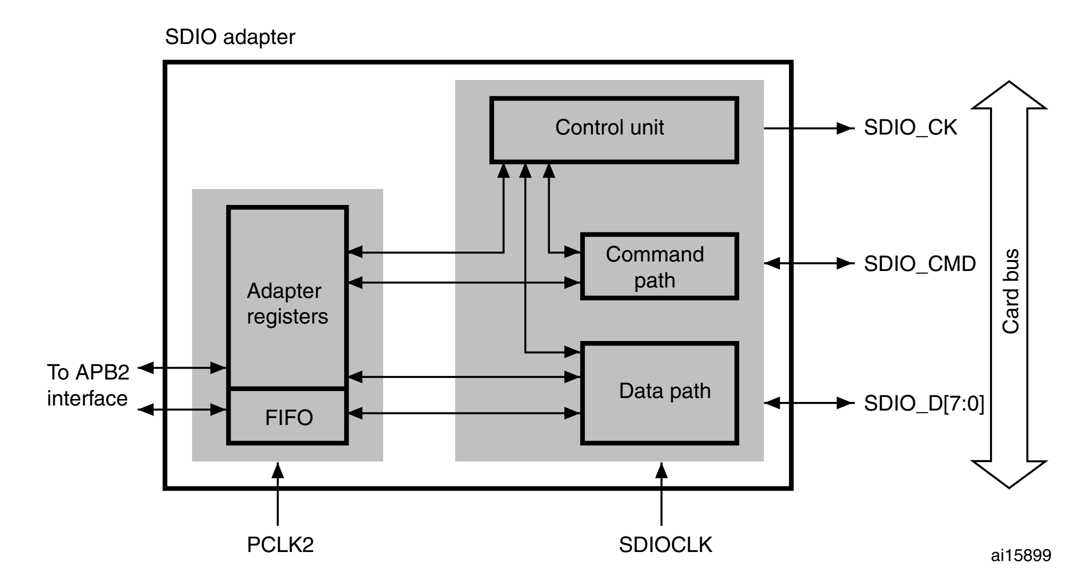
Table 149. SDIO I/O definitions

Pin	Direction	Description
SDIO_CK	Output	MultiMediaCard/SD/SDIO card clock. This pin is the clock from host to card.
SDIO_CMD	Bidirectional	MultiMediaCard/SD/SDIO card command. This pin is the bidirectional command/response signal.
SDIO_D[7:0]	Bidirectional	MultiMediaCard/SD/SDIO card data. These pins are the bidirectional databus.

28.3.1 SDIO adapter

Figure 280 shows a simplified block diagram of an SDIO adapter.

Figure 280. SDIO adapter



The SDIO adapter is a multimedia/secure digital memory card bus master that provides an interface to a multimedia card stack or to a secure digital memory card. It consists of five subunits:

- Adapter register block
- Control unit
- Command path
- Data path
- Data FIFO

*Note: The adapter registers and FIFO use the APB2 bus clock domain (*PCLK2*). The control unit, command path and data path use the SDIO adapter clock domain (SDIOCLK).*

Adapter register block

The adapter register block contains all system registers. This block also generates the signals that clear the static flags in the multimedia card. The clear signals are generated when 1 is written into the corresponding bit location in the SDIO Clear register.

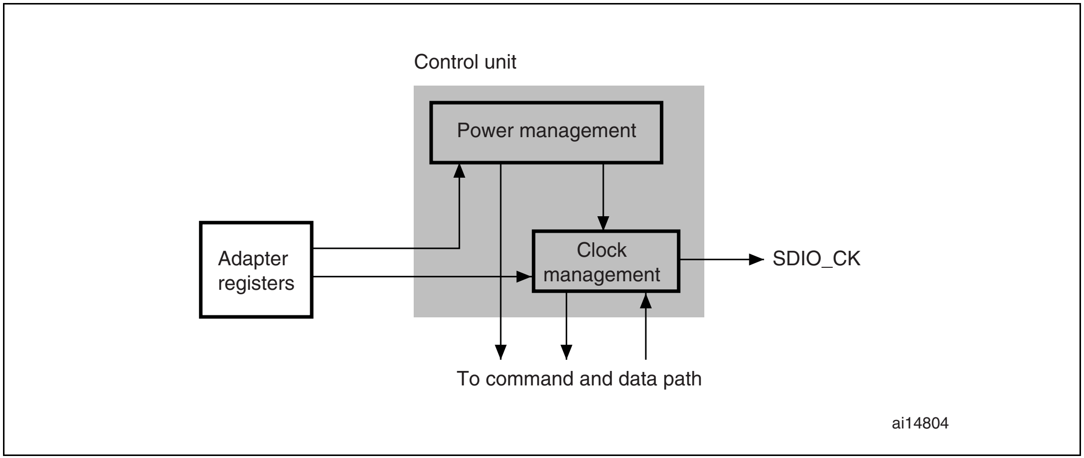
Control unit

The control unit contains the power management functions and the clock divider for the memory card clock.

There are three power phases:

- power-off
- power-up
- power-on

Figure 281. Control unit



The control unit is illustrated in *Figure 281*. It consists of a power management subunit and a clock management subunit.

The power management subunit disables the card bus output signals during the power-off and power-up phases.

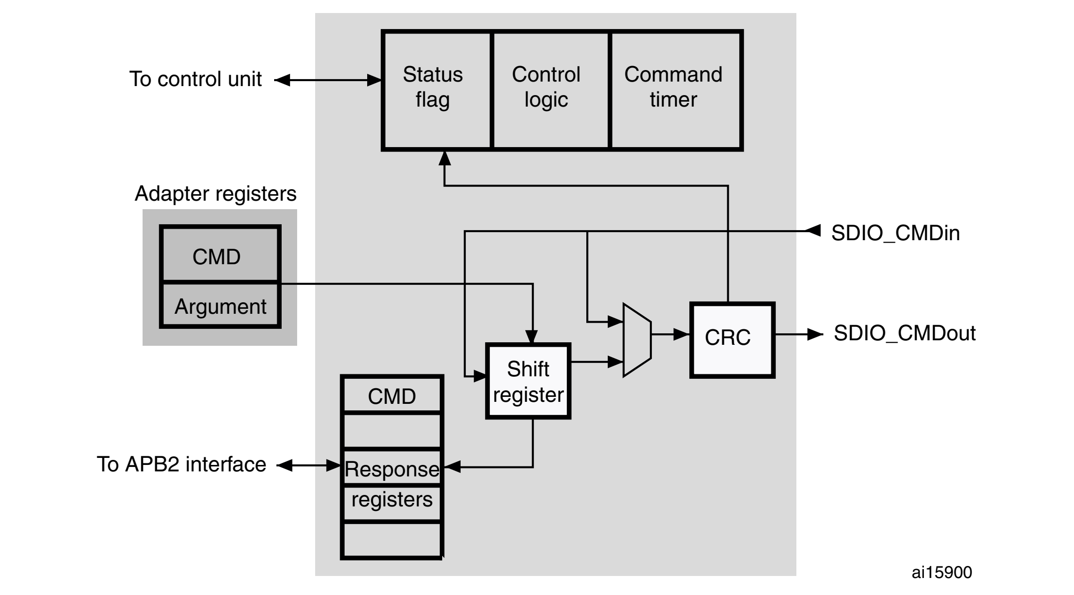
The clock management subunit generates and controls the SDIO_CK signal. The SDIO_CK output can use either the clock divide or the clock bypass mode. The clock output is inactive:

- after reset
- during the power-off or power-up phases
- if the power saving mode is enabled and the card bus is in the Idle state (eight clock periods after both the command and data path subunits enter the Idle phase)

Command path

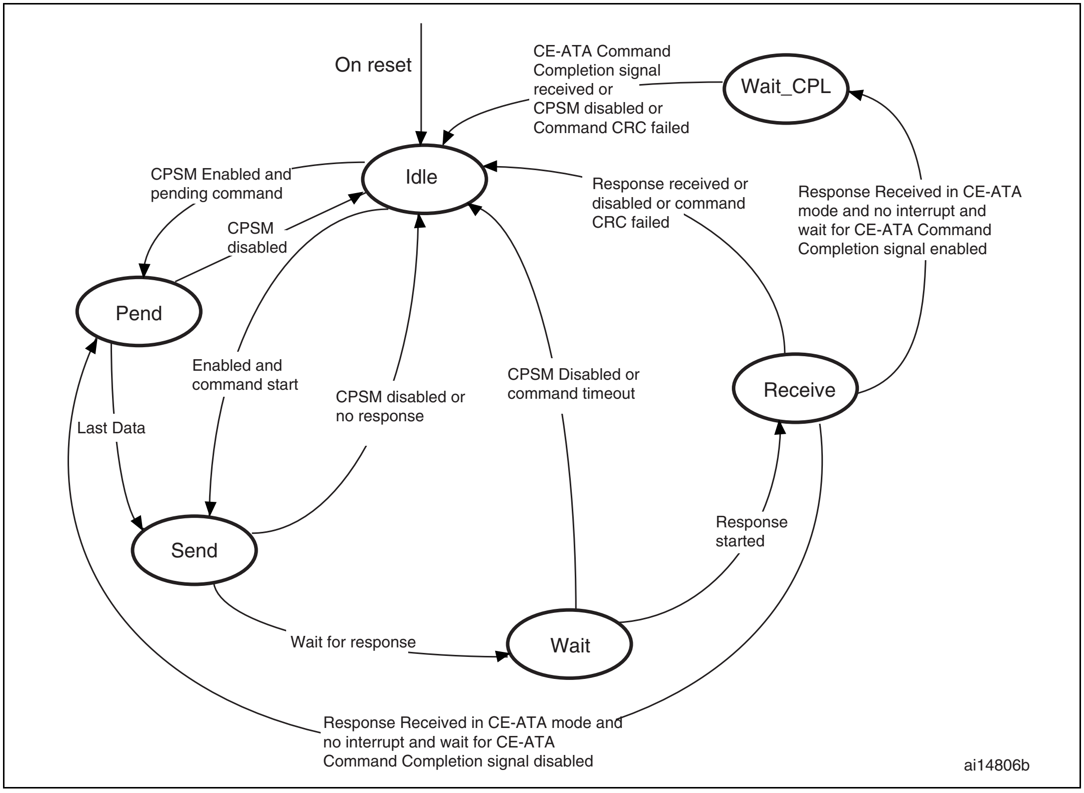
The command path unit sends commands to and receives responses from the cards.

Figure 282. SDIO adapter command path



- Command path state machine (CPSM)
  - When the command register is written to and the enable bit is set, command transfer starts. When the command has been sent, the command path state machine (CPSM) sets the status flags and enters the Idle state if a response is not required. If a response is required, it waits for the response (see *Figure 283 on page 762*). When the response is received, the received CRC code and the internally generated code are compared, and the appropriate status flags are set.

Figure 283. Command path state machine (CPSM)


On reset
CE-ATA Command Completion signal received or CPSM disabled or Command CRC failed
Wait_CPL
Idle
CPSM Enabled and pending command
CPSM disabled
Response received or disabled or command CRC failed
Response Received in CE-ATA mode and no interrupt and wait for CE-ATA Command Completion signal enabled
Pend
Enabled and command start
CPSM disabled or no response
CPSM Disabled or command timeout
Receive
Last Data
Response started
Send
Wait for response
Wait
Response Received in CE-ATA mode and no interrupt and wait for CE-ATA Command Completion signal disabled
ai14806b

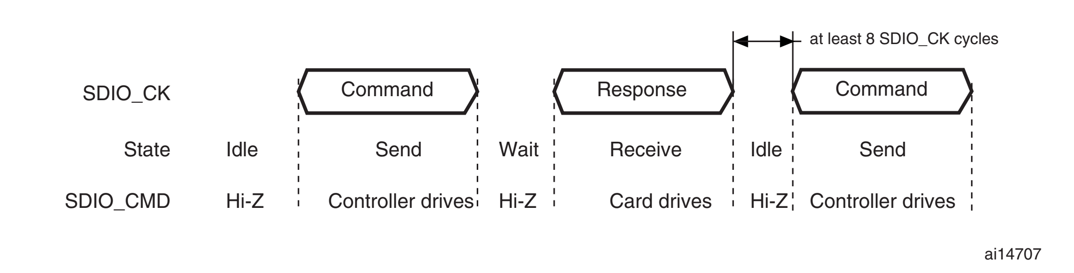
When the Wait state is entered, the command timer starts running. If the timeout is reached before the CPSM moves to the Receive state, the timeout flag is set and the Idle state is entered.

Note: The command timeout has a fixed value of 64 SDIO_CK clock periods.

If the interrupt bit is set in the command register, the timer is disabled and the CPSM waits for an interrupt request from one of the cards. If a pending bit is set in the command register, the CPSM enters the Pend state, and waits for a CmdPend signal from the data path subunit. When CmdPend is detected, the CPSM moves to the Send state. This enables the data counter to trigger the stop command transmission.

Note: The CPSM remains in the Idle state for at least eight SDIO_CK periods to meet the N_{CC} and N_{RC} timing constraints. N_{CC} is the minimum delay between two host commands, and N_{RC} is the minimum delay between the host command and the card response.

Figure 284. SDIO command transfer



- Command format
  – Command: a command is a token that starts an operation. Command are sent from the host either to a single card (addressed command) or to all connected cards (broadcast command are available for MMC V3.31 or previous). Commands are transferred serially on the CMD line. All commands have a fixed length of 48 bits. The general format for a command token for MultiMediaCards, SD-Memory cards and SDIO-Cards is shown in *Table 150*. CE-ATA commands are an extension of MMC commands V4.2, and so have the same format.

    The command path operates in a half-duplex mode, so that commands and responses can either be sent or received. If the CPSM is not in the Send state, the SDIO_CMD output is in the Hi-Z state, as shown in *Figure 284 on page 763*. Data on SDIO_CMD are synchronous with the rising edge of SDIO_CK. *Table* shows the command format.

Table 150. Command format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	1	Transmission bit
[45:40]	6	-	Command index
[39:8]	32	-	Argument
[7:1]	7	-	CRC7
0	1	1	End bit

  – Response: a response is a token that is sent from an addressed card (or synchronously from all connected cards for MMC V3.31 or previous), to the host as an answer to a previously received command. Responses are transferred serially on the CMD line.

The SDIO supports two response types. Both use CRC error checking:

- 48 bit short response
- 136 bit long response

Note: If the response does not contain a CRC (CMD1 response), the device driver must ignore the CRC failed status.

Table 151. Short response format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	-	Command index
[39:8]	32	-	Argument
[7:1]	7	-	CRC7(or 1111111)
0	1	1	End bit

Table 152. Long response format

Bit position	Width	Value	Description
135	1	0	Start bit
134	1	0	Transmission bit
[133:128]	6	111111	Reserved
[127:1]	127	-	CID or CSD (including internal CRC7)
0	1	1	End bit

The command register contains the command index (six bits sent to a card) and the command type. These determine whether the command requires a response, and whether the response is 48 or 136 bits long (see *Section 28.9.4 on page 799*). The command path implements the status flags shown in *Table 153*:

Table 153. Command path status flags

Flag	Description
CMDREND	Set if response CRC is OK.
CCRCFAIL	Set if response CRC fails.
CMDSENT	Set when command (that does not require response) is sent
CTIMEOUT	Response timeout.
CMDACT	Command transfer in progress.

The CRC generator calculates the CRC checksum for all bits before the CRC code. This includes the start bit, transmitter bit, command index, and command argument (or card status). The CRC checksum is calculated for the first 120 bits of CID or CSD for the long response format. Note that the start bit, transmitter bit and the six reserved bits are not used in the CRC calculation.

The CRC checksum is a 7-bit value:

$$\text{CRC}[6:0] = \text{Remainder } [(M(x) * x^7) / G(x)]$$

$$G(x) = x^7 + x^3 + 1$$

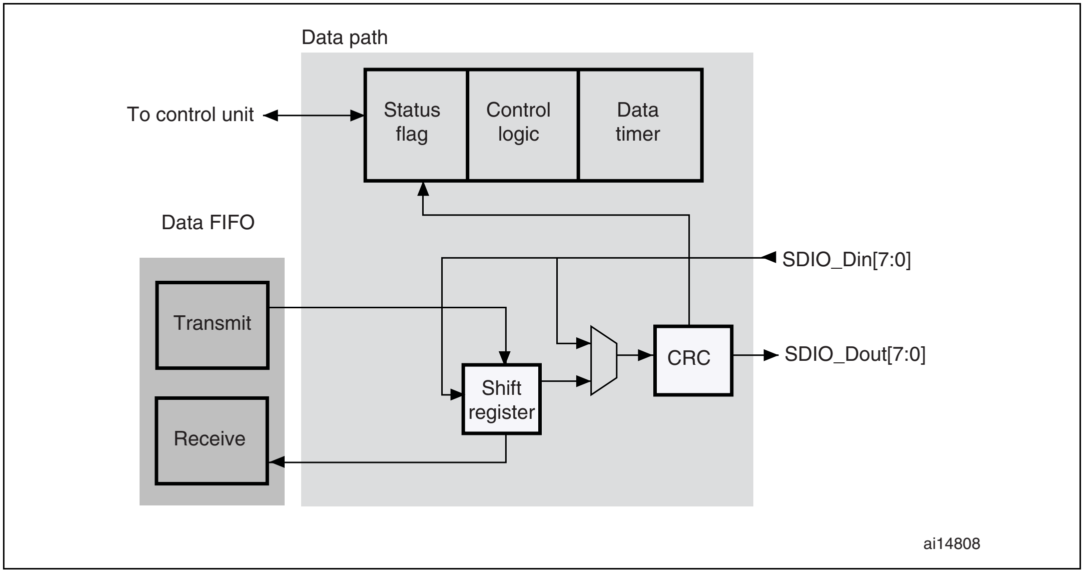
$$M(x) = (\text{start bit}) * x^{39} + \ldots + (\text{last bit before CRC}) * x^0\text{, or}$$

$$M(x) = (\text{start bit}) * x^{119} + \ldots + (\text{last bit before CRC}) * x^0$$

Data path

The data path subunit transfers data to and from cards. *Figure 285* shows a block diagram of the data path.

Figure 285. Data path



The card databus width can be programmed using the clock control register. If the 4-bit wide bus mode is enabled, data is transferred at four bits per clock cycle over all four data signals (SDIO_D[3:0]). If the 8-bit wide bus mode is enabled, data is transferred at eight bits per clock cycle over all eight data signals (SDIO_D[7:0]). If the wide bus mode is not enabled, only one bit per clock cycle is transferred over SDIO_D0.

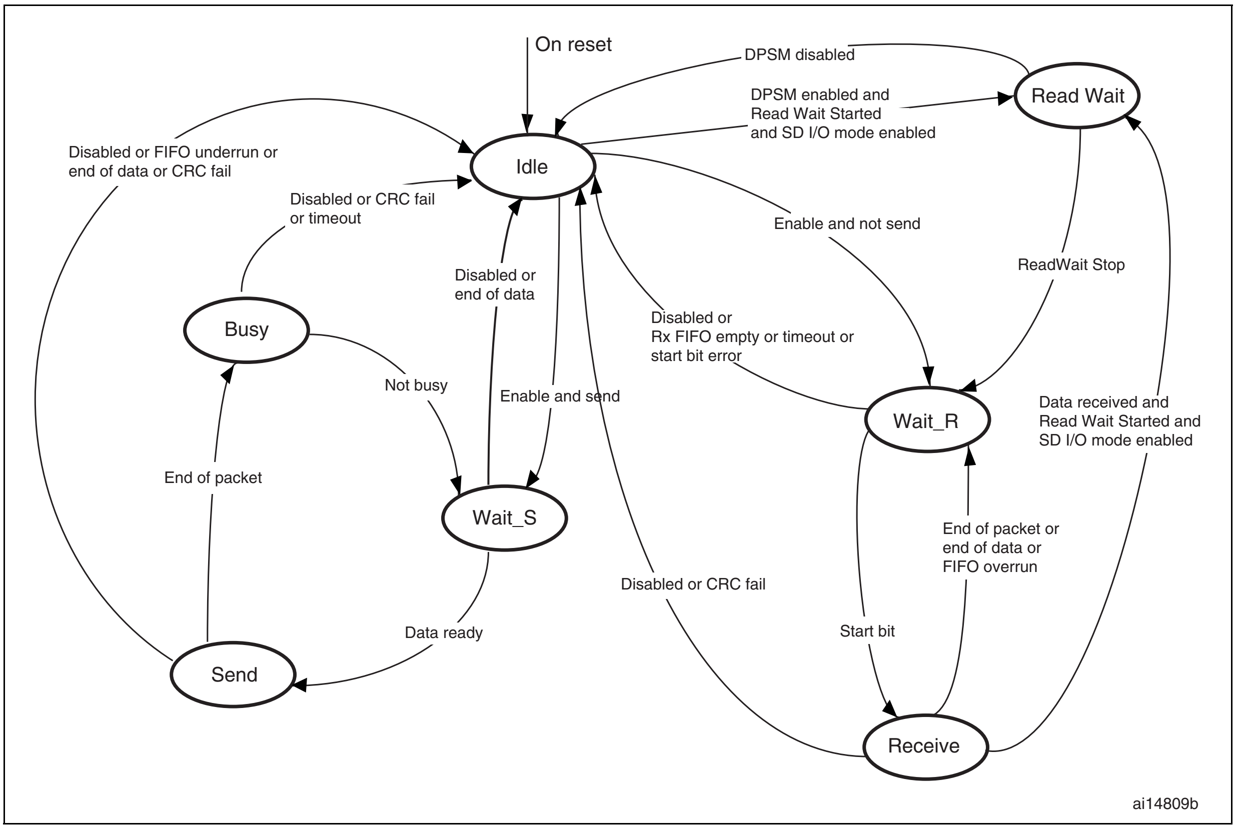
Depending on the transfer direction (send or receive), the data path state machine (DPSM) moves to the Wait_S or Wait_R state when it is enabled:

- Send: the DPSM moves to the Wait_S state. If there is data in the transmit FIFO, the DPSM moves to the Send state, and the data path subunit starts sending data to a card.
- Receive: the DPSM moves to the Wait_R state and waits for a start bit. When it receives a start bit, the DPSM moves to the Receive state, and the data path subunit starts receiving data from a card.

Data path state machine (DPSM)

The DPSM operates at SDIO_CK frequency. Data on the card bus signals is synchronous to the rising edge of SDIO_CK. The DPSM has six states, as shown in *Figure 286: Data path state machine (DPSM)*.

Figure 286. Data path state machine (DPSM)



- Idle: the data path is inactive, and the SDIO_D[7:0] outputs are in Hi-Z. When the data control register is written and the enable bit is set, the DPSM loads the data counter with a new value and, depending on the data direction bit, moves to either the Wait_S or the Wait_R state.
- Wait_R: if the data counter equals zero, the DPSM moves to the Idle state when the receive FIFO is empty. If the data counter is not zero, the DPSM waits for a start bit on SDIO_D. The DPSM moves to the Receive state if it receives a start bit before a timeout, and loads the data block counter. If it reaches a timeout before it detects a start bit, or a start bit error occurs, it moves to the Idle state and sets the timeout status flag.
- Receive: serial data received from a card is packed in bytes and written to the data FIFO. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block or stream:
  - In block mode, when the data block counter reaches zero, the DPSM waits until it receives the CRC code. If the received code matches the internally generated CRC code, the DPSM moves to the Wait_R state. If not, the CRC fail status flag is set and the DPSM moves to the Idle state.
  - In stream mode, the DPSM receives data while the data counter is not zero. When the counter is zero, the remaining data in the shift register is written to the data FIFO, and the DPSM moves to the Wait_R state.

  If a FIFO overrun error occurs, the DPSM sets the FIFO error flag and moves to the Idle state:
- Wait_S: the DPSM moves to the Idle state if the data counter is zero. If not, it waits until the data FIFO empty flag is deasserted, and moves to the Send state.

Note: The DPSM remains in the Wait_S state for at least two clock periods to meet the N_{WR} timing requirements, where N_{WR} is the number of clock cycles between the reception of the card response and the start of the data transfer from the host.

- Send: the DPSM starts sending data to a card. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block or stream:
  - In block mode, when the data block counter reaches zero, the DPSM sends an internally generated CRC code and end bit, and moves to the Busy state.
  - In stream mode, the DPSM sends data to a card while the enable bit is high and the data counter is not zero. It then moves to the Idle state.

  If a FIFO underrun error occurs, the DPSM sets the FIFO error flag and moves to the Idle state.
- Busy: the DPSM waits for the CRC status flag:
  - If it does not receive a positive CRC status, it moves to the Idle state and sets the CRC fail status flag.
  - If it receives a positive CRC status, it moves to the Wait_S state if SDIO_D0 is not low (the card is not busy).

  If a timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag and moves to the Idle state.

  The data timer is enabled when the DPSM is in the Wait_R or Busy state, and generates the data timeout error:
  - When transmitting data, the timeout occurs if the DPSM stays in the Busy state for longer than the programmed timeout period
  - When receiving data, the timeout occurs if the end of the data is not true, and if the DPSM stays in the Wait_R state for longer than the programmed timeout period.
- **Data:** data can be transferred from the card to the host or vice versa. Data is transferred via the data lines. They are stored in a FIFO of 32 words, each word is 32 bits wide.

Table 154. Data token format

Description	Start bit	Data	CRC16	End bit
Block Data	0	-	yes	1
Stream Data	0	-	no	1

Data FIFO

The data FIFO (first-in-first-out) subunit is a data buffer with a transmit and receive unit.

The FIFO contains a 32-bit wide, 32-word deep data buffer, and transmit and receive logic. Because the data FIFO operates in the APB2 clock domain (PCLK2), all signals from the subunits in the SDIO clock domain (SDIOCLK) are resynchronized.

Depending on the TXACT and RXACT flags, the FIFO can be disabled, transmit enabled, or receive enabled. TXACT and RXACT are driven by the data path subunit and are mutually exclusive:

- The transmit FIFO refers to the transmit logic and data buffer when TXACT is asserted
- The receive FIFO refers to the receive logic and data buffer when RXACT is asserted

- Transmit FIFO:

  Data can be written to the transmit FIFO through the APB2 interface when the SDIO is enabled for transmission.

  The transmit FIFO is accessible via 32 sequential addresses. The transmit FIFO contains a data output register that holds the data word pointed to by the read pointer. When the data path subunit has loaded its shift register, it increments the read pointer and drives new data out.

  If the transmit FIFO is disabled, all status flags are deasserted. The data path subunit asserts TXACT when it transmits data.

Table 155. Transmit FIFO status flags

Flag	Description
TXFIFOF	Set to high when all 32 transmit FIFO words contain valid data.
TXFIFOE	Set to high when the transmit FIFO does not contain valid data.
TXFIFOHE	Set to high when 8 or more transmit FIFO words are empty. This flag can be used as a DMA request.
TXDAVL	Set to high when the transmit FIFO contains valid data. This flag is the inverse of the TXFIFOE flag.
TXUNDERR	Set to high when an underrun error occurs. This flag is cleared by writing to the SDIO Clear register.

- Receive FIFO

  When the data path subunit receives a word of data, it drives the data on the write databus. The write pointer is incremented after the write operation completes. On the read side, the contents of the FIFO word pointed to by the current value of the read pointer is driven onto the read databus. If the receive FIFO is disabled, all status flags are deasserted, and the read and write pointers are reset. The data path subunit asserts RXACT when it receives data. *Table 156* lists the receive FIFO status flags. The receive FIFO is accessible via 32 sequential addresses.

Table 156. Receive FIFO status flags

Flag	Description
RXFIFOF	Set to high when all 32 receive FIFO words contain valid data
RXFIFOE	Set to high when the receive FIFO does not contain valid data.
RXFIFOHF	Set to high when 8 or more receive FIFO words contain valid data. This flag can be used as a DMA request.
RXDAVL	Set to high when the receive FIFO is not empty. This flag is the inverse of the RXFIFOE flag.
RXOVERR	Set to high when an overrun error occurs. This flag is cleared by writing to the SDIO Clear register.

28.3.2 SDIO APB2 interface

The APB2 interface generates the interrupt and DMA requests, and accesses the SDIO adapter registers and the data FIFO. It consists of a data path, register decoder, and interrupt/DMA logic.

SDIO interrupts

The interrupt logic generates an interrupt request signal that is asserted when at least one of the selected status flags is high. A mask register is provided to allow selection of the conditions that will generate an interrupt. A status flag generates the interrupt request if a corresponding mask flag is set.

SDIO/DMA interface - procedure for data transfers between the SDIO and memory

In the example shown, the transfer is from the SDIO host controller to an MMC (512 bytes using CMD24 (WRITE_BLOCK). The SDIO FIFO is filled by data stored in a memory using the DMA controller.

1. Do the card identification process
2. Increase the SDIO_CK frequency
3. Select the card by sending CMD7
4. Configure the DMA2 as follows:
   a) Enable DMA2 controller and clear any pending interrupts.
   b) Program the DMA2_Stream3 or DMA2_Stream6 Channel4 source address register with the memory location’s base address and DMA2_Stream3 or

DMA2_Stream6 Channel4 destination address register with the SDIO_FIFO register address.

c) Program DMA2_Stream3 or DMA2_Stream6 Channel4 control register (memory increment, not peripheral increment, peripheral and source width is word size).

d) Program DMA2_Stream3 or DMA2_Stream6 Channel4 to select the peripheral as flow controller (set PFCTRL bit in DMA_S3CR or DMA_S6CR configuration register).

e) Configure the incremental burst transfer to 4 beats (at least from peripheral side) in DMA2_Stream3 or DMA2_Stream6 Channel4.

f) Enable DMA2_Stream3 or DMA2_Stream6 Channel4

5. Send CMD24 (WRITE_BLOCK) as follows:

   a) Program the SDIO data length register (SDIO data timer register should be already programmed before the card identification process).

   b) Program the SDIO argument register with the address location of the card where data is to be transferred.

   c) Program the SDIO command register: CmdIndex with 24 (WRITE_BLOCK); WaitResp with ‘1’ (SDIO card host waits for a response); CPSMEN with ‘1’ (SDIO card host enabled to send a command). Other fields are at their reset value.

   d) Wait for SDIO_STA[6] = CMDREND interrupt, then program the SDIO data control register: DTEN with ‘1’ (SDIO card host enabled to send data); DTDIR with ‘0’ (from controller to card); DTMODE with ‘0’ (block data transfer); DMAEN with ‘1’ (DMA enabled); DBLOCKSIZE with 0x9 (512 bytes). Other fields are don’t care.

   e) Wait for SDIO_STA[10] = DBCKEND.

6. Check that no channels are still enabled by polling the DMA Enabled Channel Status register.

28.4 Card functional description

28.4.1 Card identification mode

While in card identification mode the host resets all cards, validates the operation voltage range, identifies cards and sets a relative card address (RCA) for each card on the bus. All data communications in the card identification mode use the command line (CMD) only.

28.4.2 Card reset

The `GO_IDLE_STATE` command (CMD0) is the software reset command and it puts the MultiMediaCard and SD memory in the Idle state. The `IO_RW_DIRECT` command (CMD52) resets the SD I/O card. After power-up or CMD0, all cards output bus drivers are in the high-impedance state and the cards are initialized with a default relative card address (RCA=0x0001) and with a default driver stage register setting (lowest speed, highest driving current capability).

28.4.3 Operating voltage range validation

All cards can communicate with the SDIO card host using any operating voltage within the specification range. The supported minimum and maximum V_{DD} values are defined in the operation conditions register (OCR) on the card.

Cards that store the card identification number (CID) and card specific data (CSD) in the payload memory are able to communicate this information only under data-transfer V_{DD} conditions. When the SDIO card host module and the card have incompatible V_{DD} ranges, the card is not able to complete the identification cycle and cannot send CSD data. For this purpose, the special commands, SEND_OP_COND (CMD1), SD_APP_OP_COND (ACMD41 for SD Memory), and IO_SEND_OP_COND (CMD5 for SD I/O), are designed to provide a mechanism to identify and reject cards that do not match the V_{DD} range desired by the SDIO card host. The SDIO card host sends the required V_{DD} voltage window as the operand of these commands. Cards that cannot perform data transfer in the specified range disconnect from the bus and go to the inactive state.

By using these commands without including the voltage range as the operand, the SDIO card host can query each card and determine the common voltage range before placing out-of-range cards in the inactive state. This query is used when the SDIO card host is able to select a common voltage range or when the user requires notification that cards are not usable.

28.4.4 Card identification process

The card identification process differs for MultiMediaCards and SD cards. For MultiMediaCard cards, the identification process starts at clock rate F_{od}. The SDIO_CMD line output drivers are open-drain and allow parallel card operation during this process. The registration process is accomplished as follows:

1. The bus is activated.
2. The SDIO card host broadcasts SEND_OP_COND (CMD1) to receive operation conditions.
3. The response is the wired AND operation of the operation condition registers from all cards.
4. Incompatible cards are placed in the inactive state.
5. The SDIO card host broadcasts ALL_SEND_CID (CMD2) to all active cards.
6. The active cards simultaneously send their CID numbers serially. Cards with outgoing CID bits that do not match the bits on the command line stop transmitting and must wait for the next identification cycle. One card successfully transmits a full CID to the SDIO card host and enters the Identification state.
7. The SDIO card host issues SET_RELATIVE_ADDR (CMD3) to that card. This new address is called the relative card address (RCA); it is shorter than the CID and addresses the card. The assigned card changes to the Standby state, it does not react to further identification cycles, and its output switches from open-drain to push-pull.
8. The SDIO card host repeats steps 5 through 7 until it receives a timeout condition.

For the SD card, the identification process starts at clock rate F_{od}, and the SDIO_CMD line output drives are push-pull drivers instead of open-drain. The registration process is accomplished as follows:

1. The bus is activated.
2. The SDIO card host broadcasts SD_APP_OP_COND (ACMD41).
3. The cards respond with the contents of their operation condition registers.
4. The incompatible cards are placed in the inactive state.
5. The SDIO card host broadcasts ALL_SEND_CID (CMD2) to all active cards.
6. The cards send back their unique card identification numbers (CIDs) and enter the Identification state.
7. The SDIO card host issues SET_RELATIVE_ADDR (CMD3) to an active card with an address. This new address is called the relative card address (RCA); it is shorter than the CID and addresses the card. The assigned card changes to the Standby state. The SDIO card host can reissue this command to change the RCA. The RCA of the card is the last assigned value.
8. The SDIO card host repeats steps 5 through 7 with all active cards.

For the SD I/O card, the registration process is accomplished as follows:

1. The bus is activated.
2. The SDIO card host sends IO_SEND_OP_COND (CMD5).
3. The cards respond with the contents of their operation condition registers.
4. The incompatible cards are set to the inactive state.
5. The SDIO card host issues SET_RELATIVE_ADDR (CMD3) to an active card with an address. This new address is called the relative card address (RCA); it is shorter than the CID and addresses the card. The assigned card changes to the Standby state. The SDIO card host can reissue this command to change the RCA. The RCA of the card is the last assigned value.

28.4.5 Block write

During block write (CMD24 - 27) one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host. A card supporting block write is always able to accept a block of data defined by WRITE_BL_LEN. If the CRC fails, the card indicates the failure on the SDIO_D line and the transferred data are discarded and not written, and all further transmitted blocks (in multiple block write mode) are ignored.

If the host uses partial blocks whose accumulated length is not block aligned and, block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set), the card will detect the block misalignment error before the beginning of the first misaligned block. (ADDRESS_ERROR error bit is set in the status register). The write operation will also be aborted if the host tries to write over a write-protected area. In this case, however, the card will set the WP_VIOLATION bit.

Programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in ROM, then this unchangeable part must match the corresponding part of the receive buffer. If this match fails, then the card reports an error and does not change any register contents. Some cards may require long and unpredictable times to write a block of data. After receiving a block of data and completing the CRC check, the card begins writing and holds the SDIO_D line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) at any time, and the card will respond with its status. The READY_FOR_DATA status bit indicates whether the card can accept new data or whether the write process is still in progress. The host may deselect the card by issuing CMD7 (to select a different card), which will place the card in the Disconnect state and release the SDIO_D line(s) without interrupting the write operation. When selecting the card again, it will reactivate busy indication by pulling SDIO_D to low if programming is still in progress and the write buffer is unavailable.

28.4.6 Block read

In Block read mode the basic unit of data transfer is a block whose maximum size is defined in the CSD (READ_BL_LEN). If READ_BL_PARTIAL is set, smaller blocks whose start and end addresses are entirely contained within one physical block (as defined by READ_BL_LEN) may also be transmitted. A CRC is appended to the end of each block, ensuring data transfer integrity. CMD17 (`READ_SINGLE_BLOCK`) initiates a block read and after completing the transfer, the card returns to the Transfer state.

CMD18 (`READ_MULTIPLE_BLOCK`) starts a transfer of several consecutive blocks.

The host can abort reading at any time, within a multiple block operation, regardless of its type. Transaction abort is done by sending the stop transmission command.

If the card detects an error (for example, out of range, address misalignment or internal error) during a multiple block read operation (both types) it stops the data transmission and remains in the data state. The host must than abort the operation by sending the stop transmission command. The read error is reported in the response to the stop transmission command.

If the host sends a stop transmission command after the card transmits the last block of a multiple block operation with a predefined number of blocks, it is responded to as an illegal command, since the card is no longer in the data state. If the host uses partial blocks whose accumulated length is not block-aligned and block misalignment is not allowed, the card detects a block misalignment error condition at the beginning of the first misaligned block (ADDRESS_ERROR error bit is set in the status register).

28.4.7 Stream access, stream write and stream read (MultiMediaCard only)

In stream mode, data is transferred in bytes and no CRC is appended at the end of each block.

Stream write (MultiMediaCard only)

`WRITE_DAT_UNTIL_STOP` (CMD20) starts the data transfer from the SDIO card host to the card, beginning at the specified address and continuing until the SDIO card host issues a stop command. When partial blocks are allowed (CSD parameter WRITE_BL_PARTIAL is set), the data stream can start and stop at any address within the card address space, otherwise it can only start and stop at block boundaries. Because the amount of data to be transferred is not determined in advance, a CRC cannot be used. When the end of the memory range is reached while sending data and no stop command is sent by the SD card host, any additional transferred data are discarded.

The maximum clock frequency for a stream write operation is given by the following equation fields of the card-specific data register:

$$\mathrm{Maximumspeed} = \mathrm{MIN}\left(\mathrm{TRANSPEED}, \frac{(8 \times 2^{\mathrm{writebllen}})(-\mathrm{NSAC})}{\mathrm{TAAC} \times \mathrm{R2WFACTOR}}\right)$$

- Maximumspeed = maximum write frequency
- TRANSPEED = maximum data transfer rate
- writebllen = maximum write data block length
- NSAC = data read access time 2 in CLK cycles
- TAAC = data read access time 1
- R2WFACTOR = write speed factor

If the host attempts to use a higher frequency, the card may not be able to process the data and stop programming, set the OVERRUN error bit in the status register, and while ignoring all further data transfer, wait (in the receive data state) for a stop command. The write operation is also aborted if the host tries to write over a write-protected area. In this case, however, the card sets the WP_VIOLATION bit.

Stream read (MultiMediaCard only)

READ_DAT_UNTIL_STOP (CMD11) controls a stream-oriented data transfer.

This command instructs the card to send its data, starting at a specified address, until the SDIO card host sends STOP_TRANSMISSION (CMD12). The stop command has an execution delay due to the serial command transmission and the data transfer stops after the end bit of the stop command. When the end of the memory range is reached while sending data and no stop command is sent by the SDIO card host, any subsequent data sent are considered undefined.

The maximum clock frequency for a stream read operation is given by the following equation and uses fields of the card specific data register.

$$\text{Maximumspeed} = \text{MIN}\left(\text{TRANSPEED}, \frac{(8 \times 2^{\text{readbllen}})(-\text{NSAC})}{\text{TAAC} \times \text{R2WFACTOR}}\right)$$

- Maximumspeed = maximum read frequency
- TRANSPEED = maximum data transfer rate
- readbllen = maximum read data block length
- writebllen = maximum write data block length
- NSAC = data read access time 2 in CLK cycles
- TAAC = data read access time 1
- R2WFACTOR = write speed factor

If the host attempts to use a higher frequency, the card is not able to sustain data transfer. If this happens, the card sets the UNDERRUN error bit in the status register, aborts the transmission and waits in the data state for a stop command.

28.4.8 Erase: group erase and sector erase

The erasable unit of the MultiMediaCard is the erase group. The erase group is measured in write blocks, which are the basic writable units of the card. The size of the erase group is a card-specific parameter and defined in the CSD.

The host can erase a contiguous range of Erase Groups. Starting the erase process is a three-step sequence.

First the host defines the start address of the range using the ERASE_GROUP_START (CMD35) command, next it defines the last address of the range using the ERASE_GROUP_END (CMD36) command and, finally, it starts the erase process by issuing the ERASE (CMD38) command. The address field in the erase commands is an Erase Group address in byte units. The card ignores all LSBs below the Erase Group size, effectively rounding the address down to the Erase Group boundary.

If an erase command is received out of sequence, the card sets the ERASE_SEQ_ERROR bit in the status register and resets the whole sequence.

If an out-of-sequence (neither of the erase commands, except SEND_STATUS) command received, the card sets the ERASE_RESET status bit in the status register, resets the erase sequence and executes the last command.

If the erase range includes write protected blocks, they are left intact and only unprotected blocks are erased. The WP_ERASE_SKIP status bit in the status register is set.

The card indicates that an erase is in progress by holding SDIO_D low. The actual erase time may be quite long, and the host may issue CMD7 to deselect the card.

28.4.9 Wide bus selection or deselection

Wide bus (4-bit bus width) operation mode is selected or deselected using SET_BUS_WIDTH (ACMD6). The default bus width after power-up or GO_IDLE_STATE (CMD0) is 1 bit. SET_BUS_WIDTH (ACMD6) is only valid in a transfer state, which means that the bus width can be changed only after a card is selected by SELECT/DESELECT_CARD (CMD7).

28.4.10 Protection management

Three write protection methods for the cards are supported in the SDIO card host module:

1. internal card write protection (card responsibility)
2. mechanical write protection switch (SDIO card host module responsibility only)
3. password-protected card lock operation

Internal card write protection

Card data can be protected against write and erase. By setting the permanent or temporary write-protect bits in the CSD, the entire card can be permanently write-protected by the manufacturer or content provider. For cards that support write protection of groups of sectors by setting the WP_GRP_ENABLE bit in the CSD, portions of the data can be protected, and the write protection can be changed by the application. The write protection is in units of WP_GRP_SIZE sectors as specified in the CSD. The SET_WRITE_PROT and CLR_WRITE_PROT commands control the protection of the addressed group. The SEND_WRITE_PROT command is similar to a single block read command. The card sends a data block containing 32 write protection bits (representing 32 write protect groups starting at the specified address) followed by 16 CRC bits. The address field in the write protect commands is a group address in byte units.

The card ignores all LSBs below the group size.

Mechanical write protect switch

A mechanical sliding tab on the side of the card allows the user to set or clear the write protection on a card. When the sliding tab is positioned with the window open, the card is write-protected, and when the window is closed, the card contents can be changed. A matched switch on the socket side indicates to the SDIO card host module that the card is write-protected. The SDIO card host module is responsible for protecting the card. The position of the write protect switch is unknown to the internal circuitry of the card.

Password protect

The password protection feature enables the SDIO card host module to lock and unlock a card with a password. The password is stored in the 128-bit PWD register and its size is set in the 8-bit PWD_LEN register. These registers are nonvolatile so that a power cycle does not erase them. Locked cards respond to and execute certain commands. This means that the SDIO card host module is allowed to reset, initialize, select, and query for status, however it is not allowed to access data on the card. When the password is set (as indicated by a nonzero value of PWD_LEN), the card is locked automatically after power-up. As with the CSD and CID register write commands, the lock/unlock commands are available in the transfer state only. In this state, the command does not include an address argument and the card must be selected before using it. The card lock/unlock commands have the structure and bus transaction types of a regular single-block write command. The transferred data block includes all of the required information for the command (the password setting mode, the PWD itself, and card lock/unlock). The command data block size is defined by the SDIO card host module before it sends the card lock/unlock command, and has the structure shown in *Table 170*.

The bit settings are as follows:

- ERASE: setting it forces an erase operation. All other bits must be zero, and only the command byte is sent
- LOCK_UNLOCK: setting it locks the card. LOCK_UNLOCK can be set simultaneously with SET_PWD, however not with CLR_PWD
- CLR_PWD: setting it clears the password data
- SET_PWD: setting it saves the password data to memory
- PWD_LEN: it defines the length of the password in bytes
- PWD: the password (new or currently used, depending on the command)

The following sections list the command sequences to set/reset a password, lock/unlock the card, and force an erase.

Setting the password

1. Select a card (SELECT/DESELECT_CARD, CMD7), if none is already selected.
2. Define the block length (SET_BLOCKLEN, CMD16) to send, given by the 8-bit card lock/unlock mode, the 8-bit PWD_LEN, and the number of bytes of the new password. When a password replacement is done, the block size must take into account that both the old and the new passwords are sent with the command.
3. Send LOCK/UNLOCK (CMD42) with the appropriate data block size on the data line including the 16-bit CRC. The data block indicates the mode (SET_PWD = 1), the length (PWD_LEN), and the password (PWD) itself. When a password replacement is done, the length value (PWD_LEN) includes the length of both passwords, the old and the new one, and the PWD field includes the old password (currently used) followed by the new password.
4. When the password is matched, the new password and its size are saved into the PWD and PWD_LEN fields, respectively. When the old password sent does not correspond (in size and/or content) to the expected password, the LOCK_UNLOCK_FAILED error bit is set in the card status register, and the password is not changed.

The password length field (PWD_LEN) indicates whether a password is currently set. When this field is nonzero, there is a password set and the card locks itself after power-up. It is possible to lock the card immediately in the current power session by setting the LOCK_UNLOCK bit (while setting the password) or sending an additional command for card locking.

Resetting the password

1. Select a card (SELECT/DESELECT_CARD, CMD7), if none is already selected.
2. Define the block length (SET_BLOCKLEN, CMD16) to send, given by the 8-bit card lock/unlock mode, the 8-bit PWD_LEN, and the number of bytes in the currently used password.
3. Send LOCK/UNLOCK (CMD42) with the appropriate data block size on the data line including the 16-bit CRC. The data block indicates the mode (CLR_PWD = 1), the length (PWD_LEN) and the password (PWD) itself. The LOCK_UNLOCK bit is ignored.
4. When the password is matched, the PWD field is cleared and PWD_LEN is set to 0. When the password sent does not correspond (in size and/or content) to the expected

password, the LOCK_UNLOCK_FAILED error bit is set in the card status register, and the password is not changed.

Locking a card

1. Select a card (SELECT/DESELECT_CARD, CMD7), if none is already selected.
2. Define the block length (SET_BLOCKLEN, CMD16) to send, given by the 8-bit card lock/unlock mode (byte 0 in *Table 170*), the 8-bit PWD_LEN, and the number of bytes of the current password.
3. Send LOCK/UNLOCK (CMD42) with the appropriate data block size on the data line including the 16-bit CRC. The data block indicates the mode (LOCK_UNLOCK = 1), the length (PWD_LEN), and the password (PWD) itself.
4. When the password is matched, the card is locked and the CARD_IS_LOCKED status bit is set in the card status register. When the password sent does not correspond (in size and/or content) to the expected password, the LOCK_UNLOCK_FAILED error bit is set in the card status register, and the lock fails.

It is possible to set the password and to lock the card in the same sequence. In this case, the SDIO card host module performs all the required steps for setting the password (see *Setting the password on page 777*), however it is necessary to set the LOCK_UNLOCK bit in Step 3 when the new password command is sent.

When the password is previously set (PWD_LEN is not 0), the card is locked automatically after power on reset. An attempt to lock a locked card or to lock a card that does not have a password fails and the LOCK_UNLOCK_FAILED error bit is set in the card status register.

Unlocking the card

1. Select a card (SELECT/DESELECT_CARD, CMD7), if none is already selected.
2. Define the block length (SET_BLOCKLEN, CMD16) to send, given by the 8-bit cardlock/unlock mode (byte 0 in *Table 170*), the 8-bit PWD_LEN, and the number of bytes of the current password.
3. Send LOCK/UNLOCK (CMD42) with the appropriate data block size on the data line including the 16-bit CRC. The data block indicates the mode (LOCK_UNLOCK = 0), the length (PWD_LEN), and the password (PWD) itself.
4. When the password is matched, the card is unlocked and the CARD_IS_LOCKED status bit is cleared in the card status register. When the password sent is not correct in size and/or content and does not correspond to the expected password, the LOCK_UNLOCK_FAILED error bit is set in the card status register, and the card remains locked.

The unlocking function is only valid for the current power session. When the PWD field is not clear, the card is locked automatically on the next power-up.

An attempt to unlock an unlocked card fails and the LOCK_UNLOCK_FAILED error bit is set in the card status register.

Forcing erase

If the user has forgotten the password (PWD content), it is possible to access the card after clearing all the data on the card. This forced erase operation erases all card data and all password data.

1. Select a card (SELECT/DESELECT_CARD, CMD7), if none is already selected.
2. Set the block length (SET_BLOCKLEN, CMD16) to 1 byte. Only the 8-bit card lock/unlock byte (byte 0 in *Table 170*) is sent.
3. Send LOCK/UNLOCK (CMD42) with the appropriate data byte on the data line including the 16-bit CRC. The data block indicates the mode (ERASE = 1). All other bits must be zero.
4. When the ERASE bit is the only bit set in the data field, all card contents are erased, including the PWD and PWD_LEN fields, and the card is no longer locked. When any other bits are set, the LOCK_UNLOCK_FAILED error bit is set in the card status register and the card retains all of its data, and remains locked.

An attempt to use a force erase on an unlocked card fails and the LOCK_UNLOCK_FAILED error bit is set in the card status register.

28.4.11 Card status register

The response format R1 contains a 32-bit field named card status. This field is intended to transmit the card status information (which may be stored in a local status register) to the host. If not specified otherwise, the status entries are always related to the previously issued command.

Table 157 defines the different entries of the status. The type and clear condition fields in the table are abbreviated as follows:

Type:

- E: error bit
- S: status bit
- R: detected and set for the actual command response
- X: detected and set during command execution. The SDIO card host must poll the card by issuing the status command to read these bits.

Clear condition:

- A: according to the card current state
- B: always related to the previous command. Reception of a valid command clears it (with a delay of one command)
- C: clear by read

Table 157. Card status

Bits	Identifier	Type	Value	Description	Clear condition
31	ADDRESS_ OUT_OF_RANGE	E R X	’0’= no error ’1’= error	The command address argument was out of the allowed range for this card. A multiple block or stream read/write operation is (although started in a valid address) attempting to read or write beyond the card capacity.	C
30	ADDRESS_MISALIGN		’0’= no error ’1’= error	The commands address argument (in accordance with the currently set block length) positions the first data block misaligned to the card physical blocks. A multiple block read/write operation (although started with a valid address/block-length combination) is attempting to read or write a data block which is not aligned with the physical blocks of the card.	C
29	BLOCK_LEN_ERROR		’0’= no error ’1’= error	Either the argument of a SET_BLOCKLEN command exceeds the maximum value allowed for the card, or the previously defined block length is illegal for the current command (e.g. the host issues a write command, the current block length is smaller than the maximum allowed value for the card and it is not allowed to write partial blocks)	C
28	ERASE_SEQ_ERROR		’0’= no error ’1’= error	An error in the sequence of erase commands occurred.	C
27	ERASE_PARAM	E X	’0’= no error ’1’= error	An invalid selection of erase groups for erase occurred.	C
26	WP_VIOLATION	E X	’0’= no error ’1’= error	Attempt to program a write-protected block.	C
25	CARD_IS_LOCKED	S R	‘0’ = card unlocked ‘1’ = card locked	When set, signals that the card is locked by the host	A
24	LOCK_UNLOCK_ FAILED	E X	’0’= no error ’1’= error	Set when a sequence or password error has been detected in lock/unlock card command	C
23	COM_CRC_ERROR	E R	’0’= no error ’1’= error	The CRC check of the previous command failed.	B
22	ILLEGAL_COMMAND	E R	’0’= no error ’1’= error	Command not legal for the card state	B
21	CARD_ECC_FAILED	E X	’0’= success ’1’= failure	Card internal ECC was applied but failed to correct the data.	C
20	CC_ERROR	E R	’0’= no error ’1’= error	(Undefined by the standard) A card error occurred, which is not related to the host command.	C

Table 157. Card status (continued)

Bits	Identifier	Type	Value	Description	Clear condition
19	ERROR	E X	’0’= no error ’1’= error	(Undefined by the standard) A generic card error related to the (and detected during) execution of the last host command (e.g. read or write failures).	C
18	Reserved				
17	Reserved				
16	CID/CSD_OVERWRITE	E X	’0’= no error ‘1’= error	Can be either of the following errors: – The CID register has already been written and cannot be overwritten – The read-only section of the CSD does not match the card contents – An attempt to reverse the copy (set as original) or permanent WP (unprotected) bits was made	C
15	WP_ERASE_SKIP	E X	’0’= not protected ’1’= protected	**Set when only partial address space was erased due to existing write**	C
14	CARD_ECC_DISABLED	S X	’0’= enabled ’1’= disabled	The command has been executed without using the internal ECC.	A
13	ERASE_RESET		’0’= cleared ’1’= set	An erase sequence was cleared before executing because an out of erase sequence command was received (commands other than CMD35, CMD36, CMD38 or CMD13)	C
12:9	CURRENT_STATE	S R	0 = Idle 1 = Ready 2 = Ident 3 = Stby 4 = Tran 5 = Data 6 = Rcv 7 = Prg 8 = Dis 9 = Btst 10-15 = reserved	The state of the card when receiving the command. If the command execution causes a state change, it will be visible to the host in the response on the next command. The four bits are interpreted as a binary number between 0 and 15.	B
8	READY_FOR_DATA	S R	’0’= not ready ‘1’ = ready	Corresponds to buffer empty signalling on the bus	
7	SWITCH_ERROR	E X	’0’= no error ’1’= switch error	If set, the card did not switch to the expected mode as requested by the SWITCH command	B
6	Reserved				
5	APP_CMD	S R	‘0’ = Disabled ‘1’ = Enabled	The card will expect ACMD, or an indication that the command has been interpreted as ACMD	C
4	Reserved for SD I/O Card				

Table 157. Card status (continued)

Bits	Identifier	Type	Value	Description	Clear condition
3	AKE_SEQ_ERROR	E R	’0’= no error ’1’= error	Error in the sequence of the authentication process	C
2	Reserved for application specific commands				
1	Reserved for manufacturer test mode				
0					

28.4.12 SD status register

The SD status contains status bits that are related to the SD memory card proprietary features and may be used for future application-specific usage. The size of the SD Status is one data block of 512 bits. The contents of this register are transmitted to the SDIO card host if ACMD13 is sent (CMD55 followed with CMD13). ACMD13 can be sent to a card in transfer state only (card is selected).

Table 158 defines the different entries of the SD status register. The type and clear condition fields in the table are abbreviated as follows:

Type:

- E: error bit
- S: status bit
- R: detected and set for the actual command response
- X: detected and set during command execution. The SDIO card Host must poll the card by issuing the status command to read these bits

Clear condition:

- A: according to the card current state
- B: always related to the previous command. Reception of a valid command clears it (with a delay of one command)
- C: clear by read

Table 158. SD status

Bits	Identifier	Type	Value	Description	Clear condition
511: 510	DAT_BUS_WIDTH	S R	’00’= 1 (default) ‘01’= reserved ‘10’= 4 bit width ‘11’= reserved	Shows the currently defined databus width that was defined by SET_BUS_WIDTH command	A
509	SECURED_MODE	S R	’0’= Not in the mode ’1’= In Secured Mode	Card is in Secured Mode of operation (refer to the “SD Security Specification”).	A
508: 496	Reserved				

Table 158. SD status (continued)

Bits	Identifier	Type	Value	Description	Clear condition
495: 480	SD_CARD_TYPE	S R	’00xxh’= SD Memory Cards as defined in Physical Spec Ver1.01-2.00 (’x’= don’t care). The following cards are currently defined: ’0000’= Regular SD RD/WR Card. ’0001’= SD ROM Card	In the future, the 8 LSBs will be used to define different variations of an SD memory card (each bit will define different SD types). The 8 MSBs will be used to define SD Cards that do not comply with current SD physical layer specification.	A
479: 448	SIZE_OF_PROTE CT ED_AREA	S R	Size of protected area (See below)	(See below)	A
447: 440	SPEED_CLASS	S R	Speed Class of the card (See below)	(See below)	A
439: 432	PERFORMANCE_ MOVE	S R	Performance of move indicated by 1 [MB/s] step. (See below)	(See below)	A
431:428	AU_SIZE	S R	Size of AU (See below)	(See below)	A
427:424	Reserved				
423:408	ERASE_SIZE	S R	Number of AUs to be erased at a time	(See below)	A
407:402	ERASE_TIMEOUT	S R	Timeout value for erasing areas specified by UNIT_OF_ERASE_AU	(See below)	A
401:400	ERASE_OFFSET	S R	Fixed offset value added to erase time.	(See below)	A
399:312	Reserved				
311:0	Reserved for Manufacturer				

SIZE_OF_PROTECTED_AREA

Setting this field differs between standard- and high-capacity cards. In the case of a standard-capacity card, the capacity of protected area is calculated as follows:

Protected area = SIZE_OF_PROTECTED_AREA_* MULT * BLOCK_LEN.

SIZE_OF_PROTECTED_AREA is specified by the unit in MULT*BLOCK_LEN.

In the case of a high-capacity card, the capacity of protected area is specified in this field:

Protected area = SIZE_OF_PROTECTED_AREA

SIZE_OF_PROTECTED_AREA is specified by the unit in bytes.

SPEED_CLASS

This 8-bit field indicates the speed class and the value can be calculated by $P_W/2$ (where P_W is the write performance).

Table 159. Speed class code field

SPEED_CLASS	Value definition
00h	Class 0
01h	Class 2
02h	Class 4
03h	Class 6
04h – FFh	Reserved

PERFORMANCE_MOVE

This 8-bit field indicates Pm (performance move) and the value can be set by 1 [MB/sec] steps. If the card does not move used RUs (recording units), Pm should be considered as infinity. Setting the field to FFh means infinity.

Table 160. Performance move field

PERFORMANCE_MOVE	Value definition
00h	Not defined
01h	1 [MB/sec]
02h	02h 2 [MB/sec]
---------	---------
FEh	254 [MB/sec]
FFh	Infinity

AU_SIZE

This 4-bit field indicates the AU size and the value can be selected in the power of 2 base from 16 KB.

Table 161. AU_SIZE field

AU_SIZE	Value definition
00h	Not defined
01h	16 KB
02h	32 KB
03h	64 KB
04h	128 KB
05h	256 KB
06h	512 KB
07h	1 MB
08h	2 MB

Table 161. AU_SIZE field (continued)

AU_SIZE	Value definition
09h	4 MB
Ah – Fh	Reserved

The maximum AU size, which depends on the card capacity, is defined in *Table 162*. The card can be set to any AU size between RU size and maximum AU size.

Table 162. Maximum AU size

Capacity	16 MB-64 MB	128 MB-256 MB	512 MB	1 GB-32 GB
Maximum AU Size	512 KB	1 MB	2 MB	4 MB

ERASE_SIZE

This 16-bit field indicates NERASE. When NERASE numbers of AUs are erased, the timeout value is specified by ERASE_TIMEOUT (Refer to *ERASE_TIMEOUT*). The host should determine the proper number of AUs to be erased in one operation so that the host can show the progress of the erase operation. If this field is set to 0, the erase timeout calculation is not supported.

Table 163. Erase size field

ERASE_SIZE	Value definition
0000h	Erase timeout calculation is not supported.
0001h	1 AU
0002h	2 AU
0003h	3 AU
---------	---------
FFFFh	65535 AU

ERASE_TIMEOUT

This 6-bit field indicates TERASE and the value indicates the erase timeout from offset when multiple AUs are being erased as specified by ERASE_SIZE. The range of ERASE_TIMEOUT can be defined as up to 63 seconds and the card manufacturer can choose any combination of ERASE_SIZE and ERASE_TIMEOUT depending on the implementation. Determining ERASE_TIMEOUT determines the ERASE_SIZE.

Table 164. Erase timeout field

ERASE_TIMEOUT	Value definition
00	Erase timeout calculation is not supported.
01	1 [sec]
02	2 [sec]
03	3 [sec]

Table 164. Erase timeout field (continued)

ERASE_TIMEOUT	Value definition
---------	---------
63	63 [sec]

ERASE_OFFSET

This 2-bit field indicates TOFFSET and one of four values can be selected. This field is meaningless if the ERASE_SIZE and ERASE_TIMEOUT fields are set to 0.

Table 165. Erase offset field

ERASE_OFFSET	Value definition
0h	0 [sec]
1h	1 [sec]
2h	2 [sec]
3h	3 [sec]

28.4.13 SD I/O mode

SD I/O interrupts

To allow the SD I/O card to interrupt the MultiMediaCard/SD module, an interrupt function is available on a pin on the SD interface. Pin 8, used as SDIO_D1 when operating in the 4-bit SD mode, signals the cards interrupt to the MultiMediaCard/SD module. The use of the interrupt is optional for each card or function within a card. The SD I/O interrupt is level-sensitive, which means that the interrupt line must be held active (low) until it is either recognized and acted upon by the MultiMediaCard/SD module or deasserted due to the end of the interrupt period. After the MultiMediaCard/SD module has serviced the interrupt, the interrupt status bit is cleared via an I/O write to the appropriate bit in the SD I/O card's internal registers. The interrupt output of all SD I/O cards is active low and the application must provide external pull-up resistors on all data lines (SDIO_D[3:0]). The MultiMediaCard/SD module samples the level of pin 8 (SDIO_D/IRQ) into the interrupt detector only during the interrupt period. At all other times, the MultiMediaCard/SD module ignores this value.

The interrupt period is applicable for both memory and I/O operations. The definition of the interrupt period for operations with single blocks is different from the definition for multiple-block data transfers.

SD I/O suspend and resume

Within a multifunction SD I/O or a card with both I/O and memory functions, there are multiple devices (I/O and memory) that share access to the MMC/SD bus. To share access to the MMC/SD module among multiple devices, SD I/O and combo cards optionally implement the concept of suspend/resume. When a card supports suspend/resume, the MMC/SD module can temporarily halt a data transfer operation to one function or memory (suspend) to free the bus for a higher-priority transfer to a different function or memory. After this higher-priority transfer is complete, the original transfer is resumed (restarted) where it left off. Support of suspend/resume is optional on a per-card basis. To perform the suspend/resume operation on the MMC/SD bus, the MMC/SD module performs the following steps:

1. Determines the function currently using the SDIO_D [3:0] line(s)
2. Requests the lower-priority or slower transaction to suspend
3. Waits for the transaction suspension to complete
4. Begins the higher-priority transaction
5. Waits for the completion of the higher priority transaction
6. Restores the suspended transaction

SD I/O ReadWait

The optional ReadWait (RW) operation is defined only for the SD 1-bit and 4-bit modes. The ReadWait operation allows the MMC/SD module to signal a card that it is reading multiple registers (IO_RW_EXTENDED, CMD53) to temporarily stall the data transfer while allowing the MMC/SD module to send commands to any function within the SD I/O device. To determine when a card supports the ReadWait protocol, the MMC/SD module must test capability bits in the internal card registers. The timing for ReadWait is based on the interrupt period.

28.4.14 Commands and responses

Application-specific and general commands

The SD card host module system is designed to provide a standard interface for a variety of applications types. In this environment, there is a need for specific customer/application features. To implement these features, two types of generic commands are defined in the standard: application-specific commands (ACMD) and general commands (GEN_CMD).

When the card receives the APP_CMD (CMD55) command, the card expects the next command to be an application-specific command. ACMDs have the same structure as regular MultiMediaCard commands and can have the same CMD number. The card recognizes it as ACMD because it appears after APP_CMD (CMD55). When the command immediately following the APP_CMD (CMD55) is not a defined application-specific command, the standard command is used. For example, when the card has a definition for SD_STATUS (ACMD13), and receives CMD13 immediately following APP_CMD (CMD55), this is interpreted as SD_STATUS (ACMD13). However, when the card receives CMD7 immediately following APP_CMD (CMD55) and the card does not have a definition for ACMD7, this is interpreted as the standard (SELECT/DESELECT_CARD) CMD7.

To use one of the manufacturer-specific ACMDs the SD card Host must perform the following steps:

1. Send APP_CMD (CMD55)
   The card responds to the MultiMediaCard/SD module, indicating that the APP_CMD bit is set and an ACMD is now expected.
2. Send the required ACMD
   The card responds to the MultiMediaCard/SD module, indicating that the APP_CMD bit is set and that the accepted command is interpreted as an ACMD. When a nonACMD is sent, it is handled by the card as a normal MultiMediaCard command and the APP_CMD bit in the card status register stays clear.

When an invalid command is sent (neither ACMD nor CMD) it is handled as a standard MultiMediaCard illegal command error.

The bus transaction for a GEN_CMD is the same as the single-block read or write commands (WRITE_BLOCK, CMD24 or READ_SINGLE_BLOCK,CMD17). In this case, the argument denotes the direction of the data transfer rather than the address, and the data block has vendor-specific format and meaning.

The card must be selected (in transfer state) before sending GEN_CMD (CMD56). The data block size is defined by SET_BLOCKLEN (CMD16). The response to GEN_CMD (CMD56) is in R1b format.

Command types

Both application-specific and general commands are divided into the four following types:

- **broadcast command (BC)**: sent to all cards; no responses returned.
- **broadcast command with response (BCR):** sent to all cards; responses received from all cards simultaneously.
- **addressed (point-to-point) command (AC):** sent to the card that is selected; does not include a data transfer on the SDIO_D line(s).
- **addressed (point-to-point) data transfer command (ADTC):** sent to the card that is selected; includes a data transfer on the SDIO_D line(s).

Command formats

See *Table 150 on page 763* for command formats.

Commands for the MultiMediaCard/SD module

Table 166. Block-oriented write commands

CMD index	Type	Argument	Response format	Abbreviation	Description
CMD23	ac	[31:16] set to 0 [15:0] number of blocks	R1	SET_BLOCK_COUNT	Defines the number of blocks which are going to be transferred in the multiple-block read or write command that follows.
CMD24	adtc	[31:0] data address	R1	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.
CMD25	adtc	[31:0] data address	R1	WRITE_MULTIPLE_BLOCK	Continuously writes blocks of data until a STOP_TRANSMISSION follows or the requested number of blocks has been received.
CMD26	adtc	[31:0] stuff bits	R1	PROGRAM_CID	Programming of the card identification register. This command must be issued only once per card. The card contains hardware to prevent this operation after the first programming. Normally this command is reserved for manufacturer.
CMD27	adtc	[31:0] stuff bits	R1	PROGRAM_CSD	Programming of the programmable bits of the CSD.

Table 167. Block-oriented write protection commands

CMD index	Type	Argument	Response format	Abbreviation	Description
CMD28	ac	[31:0] data address	R1b	`SET_WRITE_PROT`	If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card-specific data (WP_GRP_SIZE).
CMD29	ac	[31:0] data address	R1b	`CLR_WRITE_PROT`	If the card provides write protection features, this command clears the write protection bit of the addressed group.
CMD30	adtc	[31:0] write protect data address	R1	`SEND_WRITE_PROT`	If the card provides write protection features, this command asks the card to send the status of the write protection bits.
CMD31	Reserved				

Table 168. Erase commands

CMD index	Type	Argument	Response format	Abbreviation	Description
CMD32 ... CMD34	Reserved. These command indexes cannot be used in order to maintain backward compatibility with older versions of the MultiMediaCard.				
CMD35	ac	[31:0] data address	R1	`ERASE_GROUP_START`	Sets the address of the first erase group within a range to be selected for erase.
CMD36	ac	[31:0] data address	R1	`ERASE_GROUP_END`	Sets the address of the last erase group within a continuous range to be selected for erase.
CMD37	Reserved. This command index cannot be used in order to maintain backward compatibility with older versions of the MultiMediaCards				
CMD38	ac	[31:0] stuff bits	R1	`ERASE`	Erases all previously selected write blocks.

Table 169. I/O mode commands

CMD index	Type	Argument	Response format	Abbreviation	Description
CMD39	ac	[31:16] RCA [15:15] register write flag [14:8] register address [7:0] register data	R4	`FAST_IO`	Used to write and read 8-bit (register) data fields. The command addresses a card and a register and provides the data for writing if the write flag is set. The R4 response contains data read from the addressed register. This command accesses application-dependent registers that are not defined in the MultiMediaCard standard.

Table 169. I/O mode commands (continued)

CMD index	Type	Argument	Response format	Abbreviation	Description
CMD40	bcr	[31:0] stuff bits	R5	GO_IRQ_STATE	Places the system in the interrupt mode.
CMD41	Reserved				

Table 170. Lock card

CMD index	Type	Argument	Response format	Abbreviation	Description
CMD42	adtc	[31:0] stuff bits	R1b	LOCK_UNLOCK	Sets/resets the password or locks/unlocks the card. The size of the data block is set by the SET_BLOCK_LEN command.
CMD43 ... CMD54	Reserved				

Table 171. Application-specific commands

CMD index	Type	Argument	Response format	Abbreviation	Description
CMD55	ac	[31:16] RCA [15:0] stuff bits	R1	APP_CMD	Indicates to the card that the next command bits is an application specific command rather than a standard command
CMD56	adtc	[31:1] stuff bits [0]: RD/WR			Used either to transfer a data block to the card or to get a data block from the card for general purpose/application-specific commands. The size of the data block shall be set by the SET_BLOCK_LEN command.
CMD57 ... CMD59	Reserved.				
CMD60 ... CMD63	Reserved for manufacturer.				

28.5 Response formats

All responses are sent via the MCCMD command line SDIO_CMD. The response transmission always starts with the left bit of the bit string corresponding to the response code word. The code length depends on the response type.

A response always starts with a start bit (always 0), followed by the bit indicating the direction of transmission (card = 0). A value denoted by x in the tables below indicates a variable entry. All responses, except for the R3 response type, are protected by a CRC. Every command code word is terminated by the end bit (always 1).

There are five types of responses. Their formats are defined as follows:

28.5.1 R1 (normal response command)

Code length = 48 bits. The 45:40 bits indicate the index of the command to be responded to, this value being interpreted as a binary-coded number (between 0 and 63). The status of the card is coded in 32 bits.

Table 172. R1 response

Bit position	Width (bits	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	X	Command index
[39:8]	32	X	Card status
[7:1]	7	X	CRC7
0	1	1	End bit

28.5.2 R1b

It is identical to R1 with an optional busy signal transmitted on the data line. The card may become busy after receiving these commands based on its state prior to the command reception.

28.5.3 R2 (CID, CSD register)

Code length = 136 bits. The contents of the CID register are sent as a response to the CMD2 and CMD10 commands. The contents of the CSD register are sent as a response to CMD9. Only the bits [127...1] of the CID and CSD are transferred, the reserved bit [0] of these registers is replaced by the end bit of the response. The card indicates that an erase is in progress by holding MCDAT low. The actual erase time may be quite long, and the host may issue CMD7 to deselect the card.

Table 173. R2 response

Bit position	Width (bits	Value	Description
135	1	0	Start bit
134	1	0	Transmission bit
[133:128]	6	‘111111’	Command index
[127:1]	127	X	Card status
0	1	1	End bit

28.5.4 R3 (OCR register)

Code length: 48 bits. The contents of the OCR register are sent as a response to CMD1. The level coding is as follows: restricted voltage windows = low, card busy = low.

Table 174. R3 response

Bit position	Width (bits	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	‘111111’	Reserved
[39:8]	32	X	OCR register
[7:1]	7	‘1111111’	Reserved
0	1	1	End bit

28.5.5 R4 (Fast I/O)

Code length: 48 bits. The argument field contains the RCA of the addressed card, the register address to be read out or written to, and its content.

Table 175. R4 response

Bit position		Width (bits	Value	Description
47		1	0	Start bit
46		1	0	Transmission bit
[45:40]		6	‘100111’	CMD39
[39:8] Argument field	[31:16]	16	X	RCA
	[15:8]	8	X	register address
	[7:0]	8	X	read register contents
[7:1]		7	X	CRC7
0		1	1	End bit

28.5.6 R4b

For SD I/O only: an SDIO card receiving the CMD5 will respond with a unique SDIO response R4. The format is:

Table 176. R4b response

Bit position		Width (bits	Value	Description
47		1	0	Start bit
46		1	0	Transmission bit
[45:40]		6	x	Reserved
[39:8] Argument field	39	16	X	Card is ready
	[38:36]	3	X	Number of I/O functions
	35	1	X	Present memory
	[34:32]	3	X	Stuff bits
	[31:8]	24	X	I/O ORC

Table 176. R4b response (continued)

Bit position	Width (bits	Value	Description
[7:1]	7	X	Reserved
0	1	1	End bit

Once an SD I/O card has received a CMD5, the I/O portion of that card is enabled to respond normally to all further commands. This I/O enable of the function within the I/O card will remain set until a reset, power cycle or CMD52 with write to I/O reset is received by the card. Note that an SD memory-only card may respond to a CMD5. The proper response for a memory-only card would be *Present memory* = 1 and *Number of I/O functions* = 0. A memory-only card built to meet the SD Memory Card specification version 1.0 would detect the CMD5 as an illegal command and not respond. The I/O aware host will send CMD5. If the card responds with response R4, the host determines the card’s configuration based on the data contained within the R4 response.

28.5.7 R5 (interrupt request)

Only for MultiMediaCard. Code length: 48 bits. If the response is generated by the host, the RCA field in the argument will be 0x0.

Table 177. R5 response

Bit position		Width (bits	Value	Description
47		1	0	Start bit
46		1	0	Transmission bit
[45:40]		6	‘101000’	CMD40
[39:8] Argument field	[31:16]	16	X	RCA [31:16] of winning card or of the host
	[15:0]	16	X	Not defined. May be used for IRQ data
[7:1]		7	X	CRC7
0		1	1	End bit

28.5.8 R6

Only for SD I/O. The normal response to CMD3 by a memory device. It is shown in *Table 178*.

Table 178. R6 response

Bit position	Width (bits)	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	‘101000’	CMD40

Table 178. R6 response (continued)

Bit position		Width (bits)	Value	Description
[39:8] Argument field	[31:16]	16	X	RCA [31:16] of winning card or of the host
	[15:0]	16	X	Not defined. May be used for IRQ data
[7:1]		7	X	CRC7
0		1	1	End bit

The card [23:8] status bits are changed when CMD3 is sent to an I/O-only card. In this case, the 16 bits of response are the SD I/O-only values:

- Bit [15] COM_CRC_ERROR
- Bit [14] ILLEGAL_COMMAND
- Bit [13] ERROR
- Bits [12:0] Reserved

28.6 SDIO I/O card-specific operations

The following features are SD I/O-specific operations:

- SDIO read wait operation by SDIO_D2 signalling
- SDIO read wait operation by stopping the clock
- SDIO suspend/resume operation (write and read suspend)
- SDIO interrupts

The SDIO supports these operations only if the SDIO_DCTRL[11] bit is set, except for read suspend that does not need specific hardware implementation.

28.6.1 SDIO I/O read wait operation by SDIO_D2 signalling

It is possible to start the readwait interval before the first block is received: when the data path is enabled (SDIO_DCTRL[0] bit set), the SDIO-specific operation is enabled (SDIO_DCTRL[11] bit set), read wait starts (SDI0_DCTRL[10] =0 and SDI_DCTRL[8] =1) and data direction is from card to SDIO (SDIO_DCTRL[1] = 1), the DPSM directly moves from Idle to Readwait. In Readwait the DPSM drives SDIO_D2 to 0 after 2 SDIO_CK clock cycles. In this state, when you set the RWSTOP bit (SDIO_DCTRL[9]), the DPSM remains in Wait for two more SDIO_CK clock cycles to drive SDIO_D2 to 1 for one clock cycle (in accordance with SDIO specification). The DPSM then starts waiting again until it receives data from the card. The DPSM will not start a readwait interval while receiving a block even if read wait start is set: the readwait interval will start after the CRC is received. The RWSTOP bit has to be cleared to start a new read wait operation. During the readwait interval, the SDIO can detect SDIO interrupts on SDIO_D1.

28.6.2 SDIO read wait operation by stopping SDIO_CK

If the SDIO card does not support the previous read wait method, the SDIO can perform a read wait by stopping SDIO_CK (SDIO_DCTRL is set just like in the method presented in *Section 28.6.1*, but SDIO_DCTRL[10] =1): DSPM stops the clock two SDIO_CK cycles after the end bit of the current received block and starts the clock again after the read wait start bit is set.

As SDIO_CK is stopped, any command can be issued to the card. During a read/wait interval, the SDIO can detect SDIO interrupts on SDIO_D1.

28.6.3 SDIO suspend/resume operation

While sending data to the card, the SDIO can suspend the write operation. the SDIO_CMD[11] bit is set and indicates to the CPSM that the current command is a suspend command. The CPSM analyzes the response and when the ACK is received from the card (suspend accepted), it acknowledges the DPSM that goes Idle after receiving the CRC token of the current block.

The hardware does not save the number of the remaining block to be sent to complete the suspended operation (resume).

The write operation can be suspended by software, just by disabling the DPSM (SDIO_DCTRL[0] =0) when the ACK of the suspend command is received from the card. The DPSM enters then the Idle state.

To suspend a read: the DPSM waits in the Wait_r state as the function to be suspended sends a complete packet just before stopping the data transaction. The application continues reading RxFIFO until the FIF0 is empty, and the DPSM goes Idle automatically.

28.6.4 SDIO interrupts

SDIO interrupts are detected on the SDIO_D1 line once the SDIO_DCTRL[11] bit is set.

28.7 CE-ATA specific operations

The following features are CE-ATA specific operations:

- sending the command completion signal disable to the CE-ATA device
- receiving the command completion signal from the CE-ATA device
- signaling the completion of the CE-ATA command to the CPU, using the status bit and/or interrupt.

The SDIO supports these operations only for the CE-ATA CMD61 command, that is, if SDIO_CMD[14] is set.

28.7.1 Command completion signal disable

Command completion signal disable is sent 8 bit cycles after the reception of a **short** response if the ‘enable CMD completion’ bit, SDIO_CMD[12], is not set and the ‘not interrupt Enable’ bit, SDIO_CMD[13], is set.

The CPSM enters the Pend state, loading the command shift register with the disable sequence “00001” and, the command counter with 43. Eight cycles after, a trigger moves the CPSM to the Send state. When the command counter reaches 48, the CPSM becomes Idle as no response is awaited.

28.7.2 Command completion signal enable

If the ‘enable CMD completion’ bit SDIO_CMD[12] is set and the ‘not interrupt Enable’ bit SDIO_CMD[13] is set, the CPSM waits for the command completion signal in the Waitcpl state.

When ‘0’ is received on the CMD line, the CPSM enters the Idle state. No new command can be sent for 7 bit cycles. Then, for the last 5 cycles (out of the 7) the CMD line is driven to ‘1’ in push-pull mode.

28.7.3 CE-ATA interrupt

The command completion is signaled to the CPU by the status bit SDIO_STA[23]. This static bit can be cleared with the clear bit SDIO_ICR[23].

The SDIO_STA[23] status bit can generate an interrupt on each interrupt line, depending on the mask bit SDIO_MASKx[23].

28.7.4 Aborting CMD61

If the command completion disable signal has not been sent and CMD61 needs to be aborted, the command state machine must be disabled. It then becomes Idle, and the CMD12 command can be sent. No command completion disable signal is sent during the operation.

28.8 HW flow control

The HW flow control functionality is used to avoid FIFO underrun (TX mode) and overrun (RX mode) errors.

The behavior is to stop SDIO_CK and freeze SDIO state machines. The data transfer is stalled while the FIFO is unable to transmit or receive data. Only state machines clocked by SDIOCLK are frozen, the APB2 interface is still alive. The FIFO can thus be filled or emptied even if flow control is activated.

To enable HW flow control, the SDIO_CLKCR[14] register bit must be set to 1. After reset Flow Control is disabled.

28.9 SDIO registers

The device communicates to the system via 32-bit-wide control registers accessible via APB2.

The peripheral registers have to be accessed by words (32 bits).

28.9.1 SDIO power control register (SDIO_POWER)

Address offset: 0x00

Reset value: 0x0000 0000

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

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

Bits 1:0 **PWRCTRL:** Power supply control bits.

These bits are used to define the current functional state of the card clock:
00: Power-off: the clock to card is stopped.
01: Reserved
10: Reserved power-up
11: Power-on: the card is clocked.

Note: At least seven HCLK clock periods are needed between two write accesses to this register.

Note: After a data write, data cannot be written to this register for three SDIOCLK (48 MHz) clock periods plus two PCLK2 clock periods.

28.9.2 SDI clock control register (SDIO_CLKCR)

Address offset: 0x04

Reset value: 0x0000 0000

The SDIO_CLKCR register controls the SDIO_CK output clock.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																	HWFC_EN	NEGEDGE	WID BUS		BYPASS	PWRSAV	CLKEN	CLKDIV							
																	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 14 **HWFC_EN:** HW Flow Control enable

0b: HW Flow Control is disabled
1b: HW Flow Control is enabled
When HW Flow Control is enabled, the meaning of the TXFIFOE and RXFIFOF interrupt signals, please see SDIO Status register definition in *Section 28.9.11*.

Bit 13 **NEGEDGE:**SDIO_CK dephasing selection bit

0b: SDIO_CK generated on the rising edge of the master clock SDIOCLK
1b: SDIO_CK generated on the falling edge of the master clock SDIOCLK

Bits 12:11 **WIDBUS:** Wide bus mode enable bit

00: Default bus mode: SDIO_D0 used
01: 4-wide bus mode: SDIO_D[3:0] used
10: 8-wide bus mode: SDIO_D[7:0] used

Bit 10 **BYPASS:** Clock divider bypass enable bit

0: Disable bypass: SDIOCLK is divided according to the CLKDIV value before driving the SDIO_CK output signal.
1: Enable bypass: SDIOCLK directly drives the SDIO_CK output signal.

Bit 9 **PWRSAV:** Power saving configuration bit

For power saving, the SDIO_CK clock output can be disabled when the bus is idle by setting PWRSAV:

0: SDIO_CK clock is always enabled

1: SDIO_CK is only enabled when the bus is active

Bit 8 **CLKEN:** Clock enable bit

0: SDIO_CK is disabled

1: SDIO_CK is enabled

Bits 7:0 **CLKDIV:** Clock divide factor

This field defines the divide factor between the input clock (SDIOCLK) and the output clock (SDIO_CK): SDIO_CK frequency = SDIOCLK / [CLKDIV + 2].

Note: While the SD/SDIO card or MultiMediaCard is in identification mode, the SDIO_CK frequency must be less than 400 kHz.

The clock frequency can be changed to the maximum card bus frequency when relative card addresses are assigned to all cards.

After a data write, data cannot be written to this register for three SDIOCLK (48 MHz) clock periods plus two PCLK2 clock periods. SDIO_CK can also be stopped during the read wait interval for SD I/O cards: in this case the SDIO_CLKCR register does not control SDIO_CK.

28.9.3 SDIO argument register (SDIO_ARG)

Address offset: 0x08

Reset value: 0x0000 0000

The SDIO_ARG register contains a 32-bit command argument, which is sent to a card as part of a command message.

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

Bits 31:0 **CMDARG:** Command argument

Command argument sent to a card as part of a command message. If a command contains an argument, it must be loaded into this register before writing a command to the command register.

28.9.4 SDIO command register (SDIO_CMD)

Address offset: 0x0C

Reset value: 0x0000 0000

The SDIO_CMD register contains the command index and command type bits. The command index is sent to a card as part of a command message. The command type bits control the command path state machine (CPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																	CE-ATACMD	nIEN	ENCMDcompl	SDIOSuspend	CPSMEN	WAITPEND	WAITINT	WAITRESP		CMDINDEX					
																	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 14 **ATACMD:** CE-ATA command

If ATACMD is set, the CPSM transfers CMD61.

Bit 13 **nIEN:** not Interrupt Enable

if this bit is 0, interrupts in the CE-ATA device are enabled.

Bit 12 **ENCMDcompl:** Enable CMD completion

If this bit is set, the command completion signal is enabled.

Bit 11 **SDIOSuspend:** SD I/O suspend command

If this bit is set, the command to be sent is a suspend command (to be used only with SDIO card).

Bit 10 **CPSMEN:** Command path state machine (CPSM) Enable bit

If this bit is set, the CPSM is enabled.

Bit 9 **WAITPEND:** CPSM Waits for ends of data transfer (CmdPend internal signal).

If this bit is set, the CPSM waits for the end of data transfer before it starts sending a command.

Bit 8 **WAITINT:** CPSM waits for interrupt request

If this bit is set, the CPSM disables command timeout and waits for an interrupt request.

Bits 7:6 **WAITRESP**: Wait for response bits

They are used to configure whether the CPSM is to wait for a response, and if yes, which kind of response.
00: No response, expect CMDSENT flag
01: Short response, expect CMDREND or CCRCFAIL flag
10: No response, expect CMDSENT flag
11: Long response, expect CMDREND or CCRCFAIL flag

Bits 5:0 **CMDINDEX:** Command index

The command index is sent to the card as part of a command message.

Note: After a data write, data cannot be written to this register for three SDIOCLK (48 MHz) clock periods plus two PCLK2 clock periods.

MultiMediaCards can send two kinds of response: short responses, 48 bits long, or long responses,136 bits long. SD card and SD I/O card can send only short responses, the *argument can vary according to the type of response: the software will distinguish the type of response according to the sent command. CE-ATA devices send only short responses.*

28.9.5 SDIO command response register (SDIO_RESPCMD)

Address offset: 0x10

Reset value: 0x0000 0000

The SDIO_RESPCMD register contains the command index field of the last command response received. If the command response transmission does not contain the command index field (long or OCR response), the RESPCMD field is unknown, although it must contain 111111b (the value of the reserved field from the response).

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

Bits 31:6 Reserved, must be kept at reset value

Bits 5:0 **RESPCMD:** Response command index

Read-only bit field. Contains the command index of the last command response received.

28.9.6 SDIO response 1..4 register (SDIO_RESPx)

Address offset: (0x10 + (4 × x)); x = 1..4

Reset value: 0x0000 0000

The SDIO_RESP1/2/3/4 registers contain the status of a card, which is part of the received response.

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

Bits 31:0 **CARDSTATUSx:** see *Table 179*.

The Card Status size is 32 or 127 bits, depending on the response type.

Table 179. Response type and SDIO_RESPx registers

Register	Short response	Long response
SDIO_RESP1	Card Status[31:0]	Card Status [127:96]
SDIO_RESP2	Unused	Card Status [95:64]
SDIO_RESP3	Unused	Card Status [63:32]
SDIO_RESP4	Unused	Card Status [31:1]0b

The most significant bit of the card status is received first. The SDIO_RESP3 register LSB is always 0b.

28.9.7 SDIO data timer register (SDIO_DTIMER)

Address offset: 0x24

Reset value: 0x0000 0000

The SDIO_DTIMER register contains the data timeout period, in card bus clock periods.

A counter loads the value from the SDIO_DTIMER register, and starts decrementing when the data path state machine (DPSM) enters the Wait_R or Busy state. If the timer reaches 0 while the DPSM is in either of these states, the timeout status flag is set.

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

Bits 31:0 **DATATIME:** Data timeout period

Data timeout period expressed in card bus clock periods.

Note: A data transfer must be written to the data timer register and the data length register before being written to the data control register.

28.9.8 SDIO data length register (SDIO_DLEN)

Address offset: 0x28

Reset value: 0x0000 0000

The SDIO_DLEN register contains the number of data bytes to be transferred. The value is loaded into the data counter when data transfer starts.

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

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

Bits 24:0 **DATALENGTH:** Data length value

Number of data bytes to be transferred.

Note: For a block data transfer, the value in the data length register must be a multiple of the block size (see SDIO_DCTRL). A data transfer must be written to the data timer register and the data length register before being written to the data control register.

For an SDIO multibyte transfer the value in the data length register must be between 1 and 512.

28.9.9 SDIO data control register (SDIO_DCTRL)

Address offset: 0x2C

Reset value: 0x0000 0000

The SDIO_DCTRL register control the data path state machine (DPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				SDIOEN	RWMOD	RWSTOP	RWSTART	DBLOCKSIZE				DMAEN	DTMODE	DTDIR	DTEN
																				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value

Bit 11 **SDIOEN:** SD I/O enable functions

If this bit is set, the DPSM performs an SD I/O-card-specific operation.

Bit 10 **RWMOD:** Read wait mode

0: Read Wait control stopping SDIO_D2
1: Read Wait control using SDIO_CK

Bit 9 **RWSTOP:** Read wait stop

0: Read wait in progress if RWSTART bit is set
1: Enable for read wait stop if RWSTART bit is set

Bit 8 **RWSTART:** Read wait start

If this bit is set, read wait operation starts.

Bits 7:4 **DBLOCKSIZE:** Data block size

Define the data block length when the block data transfer mode is selected:

0000: (0 decimal) lock length = 2^0 = 1 byte
0001: (1 decimal) lock length = 2^1 = 2 bytes
0010: (2 decimal) lock length = 2^2 = 4 bytes
0011: (3 decimal) lock length = 2^3 = 8 bytes
0100: (4 decimal) lock length = 2^4 = 16 bytes
0101: (5 decimal) lock length = 2^5 = 32 bytes
0110: (6 decimal) lock length = 2^6 = 64 bytes
0111: (7 decimal) lock length = 2^7 = 128 bytes
1000: (8 decimal) lock length = 2^8 = 256 bytes
1001: (9 decimal) lock length = 2^9 = 512 bytes
1010: (10 decimal) lock length = 2^{10} = 1024 bytes
1011: (11 decimal) lock length = 2^{11} = 2048 bytes
1100: (12 decimal) lock length = 2^{12} = 4096 bytes
1101: (13 decimal) lock length = 2^{13} = 8192 bytes
1110: (14 decimal) lock length = 2^{14} = 16384 bytes
1111: (15 decimal) reserved

Bit 3 **DMAEN:** DMA enable bit

0: DMA disabled.
1: DMA enabled.

Bit 2 **DTMODE:** Data transfer mode selection 1: Stream or SDIO multibyte data transfer.

0: Block data transfer
1: Stream or SDIO multibyte data transfer

Bit 1 **DTDIR:** Data transfer direction selection

0: From controller to card.
1: From card to controller.

Bit 0 **DTEN:** Data transfer enabled bit

Data transfer starts if 1b is written to the DTEN bit. Depending on the direction bit, DTDIR, the DPSM moves to the Wait_S, Wait_R state or Readwait if RW Start is set immediately at the beginning of the transfer. It is not necessary to clear the enable bit after the end of a data transfer but the SDIO_DCTRL must be updated to enable a new data transfer

Note: After a data write, data cannot be written to this register for three SDIOCLK (48 MHz) clock periods plus two PCLK2 clock periods.

The meaning of the DTMODE bit changes according to the value of the SDIOEN bit. When SDIOEN=0 and DTMODE=1, the MultiMediaCard stream mode is enabled, and when SDIOEN=1 and DTMODE=1, the peripheral enables an SDIO multibyte transfer.

28.9.10 SDIO data counter register (SDIO_DCOUNT)

Address offset: 0x30

Reset value: 0x0000 0000

The SDIO_DCOUNT register loads the value from the data length register (see SDIO_DLEN) when the DPSM moves from the Idle state to the Wait_R or Wait_S state. As data is transferred, the counter decrements the value until it reaches 0. The DPSM then moves to the Idle state and the data status end flag, DATAEND, is set.

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

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

Bits 24:0 **DATACOUNT:** Data count value

When this bit is read, the number of remaining data bytes to be transferred is returned. Write has no effect.

Note: This register should be read only when the data transfer is complete.

28.9.11 SDIO status register (SDIO_STA)

Address offset: 0x34

Reset value: 0x0000 0000

The SDIO_STA register is a read-only register. It contains two types of flag:

- Static flags (bits [23:22,10:0]): these bits remain asserted until they are cleared by writing to the SDIO Interrupt Clear register (see SDIO_ICR)
- Dynamic flags (bits [21:11]): these bits change state depending on the state of the underlying logic (for example, FIFO full and empty flags are asserted and deasserted as data while written to the FIFO)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CEATAEND	SDIOIT	RXDAVL	TXDAVL	RXFIFOE	TXFIFOE	RXFIFOF	TXFIFOF	RXFIFOHF	TXFIFOHE	RXACT	TXACT	CMDACT	DBCKEND	STBITERR	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTIMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL
								r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bit 23 **CEATAEND:** CE-ATA command completion signal received for CMD61

Bit 22 **SDIOIT:** SDIO interrupt received

Bit 21 **RXDAVL:** Data available in receive FIFO

Bit 20 **TXDAVL:** Data available in transmit FIFO

Bit 19 **RXFIFOE:** Receive FIFO empty

Bit 18 **TXFIFOE:** Transmit FIFO empty

When HW Flow Control is enabled, TXFIFOE signals becomes activated when the FIFO contains 2 words.

Bit 17 **RXFIFOF:** Receive FIFO full

When HW Flow Control is enabled, RXFIFOF signals becomes activated 2 words before the FIFO is full.

Bit 16 **TXFIFOF:** Transmit FIFO full

Bit 15 **RXFIFOHF:** Receive FIFO half full: there are at least 8 words in the FIFO

Bit 14 **TXFIFOHE:** Transmit FIFO half empty: at least 8 words can be written into the FIFO

Bit 13 **RXACT:** Data receive in progress

Bit 12 **TXACT:** Data transmit in progress

Bit 11 **CMDACT:** Command transfer in progress

Bit 10 **DBCKEND:** Data block sent/received (CRC check passed)

Bit 9 **STBITERR:** Start bit not detected on all data signals in wide bus mode

Bit 8 **DATAEND:** Data end (data counter, SDIDCOUNT, is zero)

Bit 7 **CMDSENT:** Command sent (no response required)

Bit 6 **CMDREND:** Command response received (CRC check passed)

Bit 5 **RXOVERR:** Received FIFO overrun error

Bit 4 **TXUNDERR:** Transmit FIFO underrun error

Bit 3 **DTIMEOUT:** Data timeout

Bit 2 **CTIMEOUT:** Command response timeout

The Command TimeOut period has a fixed value of 64 SDIO_CK clock periods.

Bit 1 **DCRCFAIL:** Data block sent/received (CRC check failed)

Bit 0 **CCRCFAIL:** Command response received (CRC check failed)

28.9.12 SDIO interrupt clear register (SDIO_ICR)

Address offset: 0x38

Reset value: 0x0000 0000

The SDIO_ICR register is a write-only register. Writing a bit with 1b clears the corresponding bit in the SDIO_STA Status 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
Reserved								CEATAENDC	SDIOITC	Reserved											DBCKENDC	STBITERRC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTIMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC
								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 **CEATAENDC:** CEATAEND flag clear bit

Set by software to clear the CEATAEND flag.
0: CEATAEND not cleared
1: CEATAEND cleared

Bit 22 **SDIOITC:** SDIOIT flag clear bit

Set by software to clear the SDIOIT flag.
0: SDIOIT not cleared
1: SDIOIT cleared

Bits 21:11 Reserved, must be kept at reset value

Bit 10 **DBCKENDC:** DBCKEND flag clear bit

Set by software to clear the DBCKEND flag.
0: DBCKEND not cleared
1: DBCKEND cleared

Bit 9 **STBITERRC:** STBITERR flag clear bit

Set by software to clear the STBITERR flag.
0: STBITERR not cleared
1: STBITERR cleared

Bit 8 **DATAENDC:** DATAEND flag clear bit

Set by software to clear the DATAEND flag.
0: DATAEND not cleared
1: DATAEND cleared

Bit 7 **CMDSENTC:** CMDSENT flag clear bit

Set by software to clear the CMDSENT flag.
0: CMDSENT not cleared
1: CMDSENT cleared

Bit 6 **CMDRENDC:** CMDREND flag clear bit

Set by software to clear the CMDREND flag.
0: CMDREND not cleared
1: CMDREND cleared

Bit 5 **RXOVERRC:** RXOVERR flag clear bit

Set by software to clear the RXOVERR flag.
0: RXOVERR not cleared
1: RXOVERR cleared

Bit 4 **TXUNDERRC:** TXUNDERR flag clear bit

Set by software to clear TXUNDERR flag.
0: TXUNDERR not cleared
1: TXUNDERR cleared

Bit 3 **DTIMEOUTC:** DTIMEOUT flag clear bit

Set by software to clear the DTIMEOUT flag.
0: DTIMEOUT not cleared
1: DTIMEOUT cleared

Bit 2 **CTIMEOUTC:** CTIMEOUT flag clear bit

Set by software to clear the CTIMEOUT flag.
0: CTIMEOUT not cleared
1: CTIMEOUT cleared

Bit 1 **DCRCFAILC:** DCRCFAIL flag clear bit

Set by software to clear the DCRCFAIL flag.
0: DCRCFAIL not cleared
1: DCRCFAIL cleared

Bit 0 **CCRCFAILC:** CCRCFAIL flag clear bit

Set by software to clear the CCRCFAIL flag.
0: CCRCFAIL not cleared
1: CCRCFAIL cleared

28.9.13 SDIO mask register (SDIO_MASK)

Address offset: 0x3C

Reset value: 0x0000 0000

The interrupt mask register determines which status flags generate an interrupt request by setting the corresponding bit to 1b.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CEATAENDIE	SDIOITIE	RXDAVLIE	TXDAVLIE	RXFIFOEIE	TXFIFOEIE	RXFIFOFIE	TXFIFOFIE	RXFIFOHFIE	TXFIFOHEIE	RXACTIE	TXACTIE	CMDACTIE	DBCKENDIE	STBITERRIE	DATAENDIE	CMDSENTIE	CMDRENDIE	RXOVERRIE	TXUNDERRIE	DTIMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE
								rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 23 **CEATAENDIE:** CE-ATA command completion signal received interrupt enable

Set and cleared by software to enable/disable the interrupt generated when receiving the CE-ATA command completion signal.

0: CE-ATA command completion signal received interrupt disabled

1: CE-ATA command completion signal received interrupt enabled

Bit 22 **SDIOITIE:** SDIO mode interrupt received interrupt enable

Set and cleared by software to enable/disable the interrupt generated when receiving the SDIO mode interrupt.

0: SDIO Mode Interrupt Received interrupt disabled

1: SDIO Mode Interrupt Received interrupt enabled

Bit 21 **RXDAVLIE:** Data available in Rx FIFO interrupt enable

Set and cleared by software to enable/disable the interrupt generated by the presence of data available in Rx FIFO.

0: Data available in Rx FIFO interrupt disabled

1: Data available in Rx FIFO interrupt enabled

Bit 20 **TXDAVLIE:** Data available in Tx FIFO interrupt enable

Set and cleared by software to enable/disable the interrupt generated by the presence of data available in Tx FIFO.

0: Data available in Tx FIFO interrupt disabled

1: Data available in Tx FIFO interrupt enabled

Bit 19 **RXFIFOEIE:** Rx FIFO empty interrupt enable

Set and cleared by software to enable/disable interrupt caused by Rx FIFO empty.

0: Rx FIFO empty interrupt disabled

1: Rx FIFO empty interrupt enabled

Bit 18 **TXFIFOEIE:** Tx FIFO empty interrupt enable

Set and cleared by software to enable/disable interrupt caused by Tx FIFO empty.

0: Tx FIFO empty interrupt disabled

1: Tx FIFO empty interrupt enabled

Bit 17 **RXFIFOFIE:** Rx FIFO full interrupt enable

Set and cleared by software to enable/disable interrupt caused by Rx FIFO full.

0: Rx FIFO full interrupt disabled

1: Rx FIFO full interrupt enabled

Bit 16 **TXFIFOFIE:** Tx FIFO full interrupt enable

Set and cleared by software to enable/disable interrupt caused by Tx FIFO full.
0: Tx FIFO full interrupt disabled
1: Tx FIFO full interrupt enabled

Bit 15 **RXFIFOHFIE:** Rx FIFO half full interrupt enable

Set and cleared by software to enable/disable interrupt caused by Rx FIFO half full.
0: Rx FIFO half full interrupt disabled
1: Rx FIFO half full interrupt enabled

Bit 14 **TXFIFOHEIE:** Tx FIFO half empty interrupt enable

Set and cleared by software to enable/disable interrupt caused by Tx FIFO half empty.
0: Tx FIFO half empty interrupt disabled
1: Tx FIFO half empty interrupt enabled

Bit 13 **RXACTIE:** Data receive acting interrupt enable

Set and cleared by software to enable/disable interrupt caused by data being received (data receive acting).
0: Data receive acting interrupt disabled
1: Data receive acting interrupt enabled

Bit 12 **TXACTIE:** Data transmit acting interrupt enable

Set and cleared by software to enable/disable interrupt caused by data being transferred (data transmit acting).
0: Data transmit acting interrupt disabled
1: Data transmit acting interrupt enabled

Bit 11 **CMDACTIE:** Command acting interrupt enable

Set and cleared by software to enable/disable interrupt caused by a command being transferred (command acting).
0: Command acting interrupt disabled
1: Command acting interrupt enabled

Bit 10 **DBCKENDIE:** Data block end interrupt enable

Set and cleared by software to enable/disable interrupt caused by data block end.
0: Data block end interrupt disabled
1: Data block end interrupt enabled

Bit 9 **STBITERRIE:** Start bit error interrupt enable

Set and cleared by software to enable/disable interrupt caused by start bit error.
0: Start bit error interrupt disabled
1: Start bit error interrupt enabled

Bit 8 **DATAENDIE:** Data end interrupt enable

Set and cleared by software to enable/disable interrupt caused by data end.
0: Data end interrupt disabled
1: Data end interrupt enabled

Bit 7 **CMDSENTIE:** Command sent interrupt enable

Set and cleared by software to enable/disable interrupt caused by sending command.
0: Command sent interrupt disabled
1: Command sent interrupt enabled

Bit 6 **CMDRENDIE:** Command response received interrupt enable

Set and cleared by software to enable/disable interrupt caused by receiving command response.
0: Command response received interrupt disabled
1: command Response Received interrupt enabled

Bit 5 **RXOVERRIE:** Rx FIFO overrun error interrupt enable

Set and cleared by software to enable/disable interrupt caused by Rx FIFO overrun error.
0: Rx FIFO overrun error interrupt disabled
1: Rx FIFO overrun error interrupt enabled

Bit 4 **TXUNDERRIE:** Tx FIFO underrun error interrupt enable

Set and cleared by software to enable/disable interrupt caused by Tx FIFO underrun error.
0: Tx FIFO underrun error interrupt disabled
1: Tx FIFO underrun error interrupt enabled

Bit 3 **DTIMEOUTIE:** Data timeout interrupt enable

Set and cleared by software to enable/disable interrupt caused by data timeout.
0: Data timeout interrupt disabled
1: Data timeout interrupt enabled

Bit 2 **CTIMEOUTIE:** Command timeout interrupt enable

Set and cleared by software to enable/disable interrupt caused by command timeout.
0: Command timeout interrupt disabled
1: Command timeout interrupt enabled

Bit 1 **DCRCFAILIE:** Data CRC fail interrupt enable

Set and cleared by software to enable/disable interrupt caused by data CRC failure.
0: Data CRC fail interrupt disabled
1: Data CRC fail interrupt enabled

Bit 0 **CCRCFAILIE:** Command CRC fail interrupt enable

Set and cleared by software to enable/disable interrupt caused by command CRC failure.
0: Command CRC fail interrupt disabled
1: Command CRC fail interrupt enabled

28.9.14 SDIO FIFO counter register (SDIO_FIFOCNT)

Address offset: 0x48

Reset value: 0x0000 0000

The SDIO_FIFOCNT register contains the remaining number of words to be written to or read from the FIFO. The FIFO counter loads the value from the data length register (see SDIO_DLEN) when the data transfer enable bit, DTEN, is set in the data control register (SDIO_DCTRL register) and the DPSM is at the Idle state. If the data length is not word-aligned (multiple of 4), the remaining 1 to 3 bytes are regarded as a word.

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

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

Bits 23:0 **FIFOCOUNT:** Remaining number of words to be written to or read from the FIFO.

28.9.15 SDIO data FIFO register (SDIO_FIFO)

Address offset: 0x80

Reset value: 0x0000 0000

The receive and transmit FIFOs can be read or written as 32-bit wide registers. The FIFOs contain 32 entries on 32 sequential addresses. This allows the CPU to use its load and store multiple operands to read from/write to the FIFO.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIF0Data																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

bits 31:0 **FIFOData:** Receive and transmit FIFO data

The FIFO data occupies 32 entries of 32-bit words, from address:
SDIO base + 0x080 to SDIO base + 0xFC.

28.9.16 SDIO register map

The following table summarizes the SDIO registers.

Table 180. SDIO register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**SDIO_POWER**	Reserved																														PWRCTRL	
0x04	**SDIO_CLKCR**	Reserved																	HWFC_EN	NEGEDGE	WIDBUS		BYPASS	PWRSAV	CLKEN	CLKDIV							
0x08	**SDIO_ARG**	CMDARG																															
0x0C	**SDIO_CMD**	Reserved																	CE-ATACMD	nIEN	ENCMDcompl	SDIOSuspend	CPSMEN	WAITPEND	WAITINT	WAITRESP		CMDINDEX					
0x10	**SDIO_RESPCMD**	Reserved																										RESPCMD					
0x14	**SDIO_RESP1**	CARDSTATUS1																															
0x18	**SDIO_RESP2**	CARDSTATUS2																															
0x1C	**SDIO_RESP3**	CARDSTATUS3																															
0x20	**SDIO_RESP4**	CARDSTATUS4																															
0x24	**SDIO_DTIMER**	DATATIME																															
0x28	**SDIO_DLEN**	Reserved							DATALENGTH																								

Table 180. SDIO register map (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	**SDIO_DCTRL**	Reserved																				SDIOEN	RWMOD	RWSTOP	RWSTART	DBLOCKSIZE				DMAEN	DTMODE	DTDIR	DTEN
0x30	**SDIO_DCOUNT**	Reserved							DATACOUNT																								
0x34	**SDIO_STA**	Reserved								CEATAEND	SDIOIT	RXDAVL	TXDAVL	RXFIFOE	TXFIFOE	RXFIFOF	TXFIFOF	RXFIFOHF	TXFIFOHE	RXACT	TXACT	CMDACT	DBCKEND	STBITERR	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTIMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL
0x38	**SDIO_ICR**	Reserved								CEATAENDC	SDIOITC	Reserved											DBCKENDC	STBITERRC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTIMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC
0x3C	**SDIO_MASK**	Reserved								CEATAENDIE	SDIOITIE	RXDAVLIE	TXDAVLIE	RXFIFOEIE	TXFIFOEIE	RXFIFOFIE	TXFIFOFIE	RXFIFOHFIE	TXFIFOHEIE	RXACTIE	TXACTIE	CMDACTIE	DBCKENDIE	STBITERRIE	DATAENDIE	CMDSENTIE	CMDRENDIE	RXOVERRIE	TXUNDERRIE	DTIMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE
0x48	**SDIO_FIFOCNT**	Reserved								FIFOCOUNT																							
0x80	**SDIO_FIFO**	FIF0Data																															

Refer to *Table 2 on page 42* for the register boundary addresses.

29 Debug support (DBG)

29.1 Overview

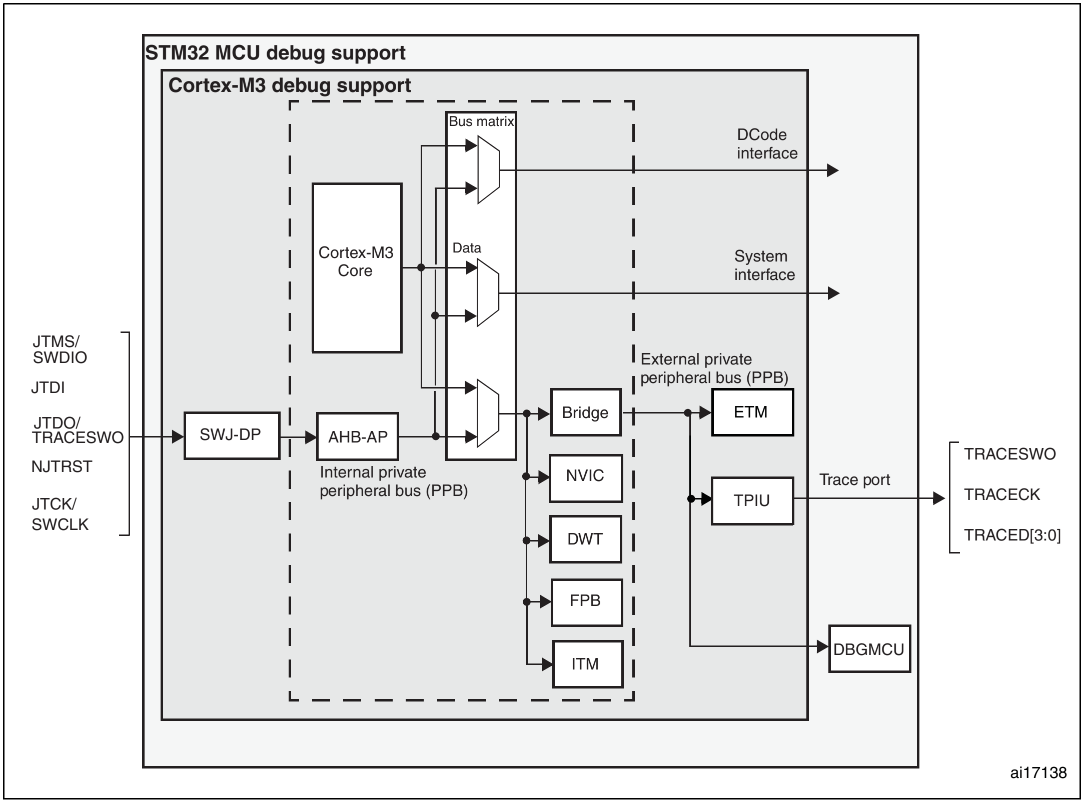
The STM32L1xxxx are built around a Cortex™-M3 core which contains hardware extensions for advanced debugging features. The debug extensions allow the core to be stopped either on a given instruction fetch (breakpoint) or data access (watchpoint). When stopped, the core's internal state and the system's external state may be examined. Once examination is complete, the core and the system may be restored and program execution resumed.

The debug features are used by the debugger host when connecting to and debugging the STM32L1xxxx MCUs.

Two interfaces for debug are available:

- Serial wire
- JTAG debug port

Figure 287. Block diagram of STM32 MCU and Cortex™-M3-level debug support



Note: The debug features embedded in the Cortex™-M3 *core are a subset of the ARM CoreSight Design Kit.*

The ARM Cortex™-M3 core provides integrated on-chip debug support. It is comprised of:

- SWJ-DP: Serial wire / JTAG debug port
- AHP-AP: AHB access port
- ITM: Instrumentation trace macrocell
- FPB: Flash patch breakpoint
- DWT: Data watchpoint trigger
- TPUI: Trace port unit interface (available on larger packages, where the corresponding pins are mapped)
- ETM: Embedded Trace Macrocell (available on larger packages, where the corresponding pins are mapped)

It also includes debug features dedicated to the STM32L1xxxx:

- Flexible debug pinout assignment
- MCU debug box (support for low-power modes, control over peripheral clocks, etc.)

Note: For further information on debug functionality supported by the ARM Cortex™-M3 core, refer to the Cortex™-M3-r2p0 Technical Reference Manual and to the CoreSight Design Kit-r2p0 TRM (see Section 29.2: Reference ARM documentation).

29.2 Reference ARM documentation

- Cortex™-M3 r2p0 Technical Reference Manual (TRM)
  (see Related documents on page 1)
- ARM Debug Interface V5
- ARM CoreSight Design Kit revision r2p0 Technical Reference Manual

29.3 SWJ debug port (serial wire and JTAG)

The STM32L1xxxx core integrates the Serial Wire / JTAG Debug Port (SWJ-DP). It is an ARM standard CoreSight debug port that combines a JTAG-DP (5-pin) interface and a SW-DP (2-pin) interface.

- The JTAG Debug Port (JTAG-DP) provides a 5-pin standard JTAG interface to the AHP-AP port.
- The Serial Wire Debug Port (SW-DP) provides a 2-pin (clock + data) interface to the AHP-AP port.

In the SWJ-DP, the two JTAG pins of the SW-DP are multiplexed with some of the five JTAG pins of the JTAG-DP.

Figure 288. SWJ debug port

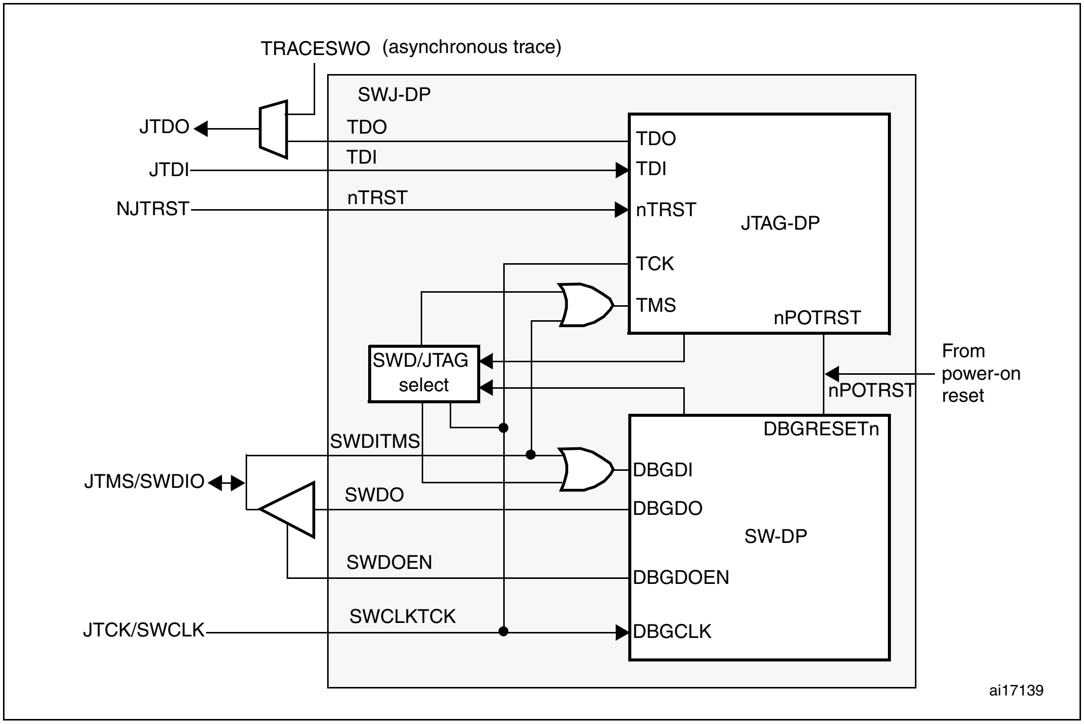


Figure 288 shows that the asynchronous TRACE output (TRACESWO) is multiplexed with TDO. This means that the asynchronous trace can only be used with SW-DP, not JTAG-DP.

29.3.1 Mechanism to select the JTAG-DP or the SW-DP

By default, the JTAG-Debug Port is active.

If the debugger host wants to switch to the SW-DP, it must provide a dedicated JTAG sequence on TMS/TCK (respectively mapped to SWDIO and SWCLK) which disables the JTAG-DP and enables the SW-DP. This way it is possible to activate the SWDP using only the SWCLK and SWDIO pins.

This sequence is:

1. Send more than 50 TCK cycles with TMS (SWDIO) =1
2. Send the 16-bit sequence on TMS (SWDIO) = 0111100111100111 (MSB transmitted first)
3. Send more than 50 TCK cycles with TMS (SWDIO) =1

29.4 Pinout and debug port pins

The STM32L1xxxx MCUs are available in various packages with different numbers of available pins. As a result, some functionality (ETM) related to pin availability may differ between packages.

29.4.1 SWJ debug port pins

Five pins are used as outputs from the STM32L1xxxx for the SWJ-DP as *alternate functions* of general-purpose I/Os. These pins are available on all packages.

Table 181. SWJ debug port pins

SWJ-DP pin name	JTAG debug port		SW debug port		Pin assign ment
	Type	Description	Type	Debug assignment	
JTMS/SWDIO	I	JTAG Test Mode Selection	IO	Serial Wire Data Input/Output	PA13
JTCK/SWCLK	I	JTAG Test Clock	I	Serial Wire Clock	PA14
JTDI	I	JTAG Test Data Input	-	-	PA15
JTDO/TRACESWO	O	JTAG Test Data Output	-	TRACESWO if async trace is enabled	PB3
NJTRST	I	JTAG Test nReset	-	-	PB4

29.4.2 Flexible SWJ-DP pin assignment

After RESET (SYSRESETn or PORESETn), all five pins used for the SWJ-DP are assigned as dedicated pins immediately usable by the debugger host (note that the trace outputs are not assigned except if explicitly programmed by the debugger host).

However, the STM32L1xxxx MCU offers the possibility of disabling some or all of the SWJ-DP ports and so, of releasing the associated pins for general-purpose IO (GPIO) usage.

Table 182. Flexible SWJ-DP pin assignment

Available debug ports	SWJ IO pin assigned				
	PA13 / JTMS / SWDIO	PA14 / JTCK / SWCLK	PA15 / JTDI	PB3 / JTDO	PB4 / NJTRST
Full SWJ (JTAG-DP + SW-DP) - Reset State	X	X	X	X	X
Full SWJ (JTAG-DP + SW-DP) but without NJTRST	X	X	X	X	
JTAG-DP Disabled and SW-DP Enabled	X	X			
JTAG-DP Disabled and SW-DP Disabled	Released				

Note: When the APB bridge write buffer is full, it takes one extra APB cycle when writing the GPIO_AFR register. This is because the deactivation of the JTAGSW pins is done in two cycles to guarantee a clean level on the nTRST and TCK input signals of the core.

- Cycle 1: the JTAGSW input signals to the core are tied to 1 or 0 (to 1 for nTRST, TDI and TMS, to 0 for TCK)
- Cycle 2: the GPIO controller takes the control signals of the SWJTAG IO pins (like controls of direction, pull-up/down, Schmitt trigger activation, etc.).

29.4.3 Internal pull-up and pull-down on JTAG pins

It is necessary to ensure that the JTAG input pins are not floating since they are directly connected to flip-flops to control the debug mode features. Special care must be taken with the SWCLK/TCK pin which is directly connected to the clock of some of these flip-flops.

To avoid any uncontrolled IO levels, the device embeds internal pull-ups and pull-downs on the JTAG input pins:

- NJTRST: Internal pull-up
- JTDI: Internal pull-up
- JTMS/SWDIO: Internal pull-up
- TCK/SWCLK: Internal pull-down

Once a JTAG IO is released by the user software, the GPIO controller takes control again. The reset states of the GPIO control registers put the I/Os in the equivalent state:

- NJTRST: AF input pull-up
- JTDI: AF input pull-up
- JTMS/SWDIO: AF input pull-up
- JTCK/SWCLK: AF input pull-down
- JTDO: AF output floating

The software can then use these I/Os as standard GPIOs.

Note: The JTAG IEEE standard recommends to add pull-ups on TDI, TMS and nTRST but there is no special recommendation for TCK. However, for JTCK, the device needs an integrated pull-down.

Having embedded pull-ups and pull-downs removes the need to add external resistors.

29.4.4 Using serial wire and releasing the unused debug pins as GPIOs

To use the serial wire DP to release some GPIOs, the user software must change the GPIO (PA15, PB3 and PB4) configuration mode in the GPIO_MODER register. This releases PA15, PB3 and PB4 which now become available as GPIOs.

When debugging, the host performs the following actions:

- Under system reset, all SWJ pins are assigned (JTAG-DP + SW-DP).
- Under system reset, the debugger host sends the JTAG sequence to switch from the JTAG-DP to the SW-DP.
- Still under system reset, the debugger sets a breakpoint on vector reset.
- The system reset is released and the Core halts.
- All the debug communications from this point are done using the SW-DP. The other JTAG pins can then be reassigned as GPIOs by the user software.

Note: For user software designs, note that:

To release the debug pins, remember that they will be first configured either in input-pull-up (nTRST, TMS, TDI) or pull-down (TCK) or output tristate (TDO) for a certain duration after reset until the instant when the user software releases the pins.

When debug pins (JTAG or SW or TRACE) are mapped, changing the corresponding IO pin configuration in the IOPORT controller has no effect.

29.5 STM32L1xxxx JTAG TAP connection

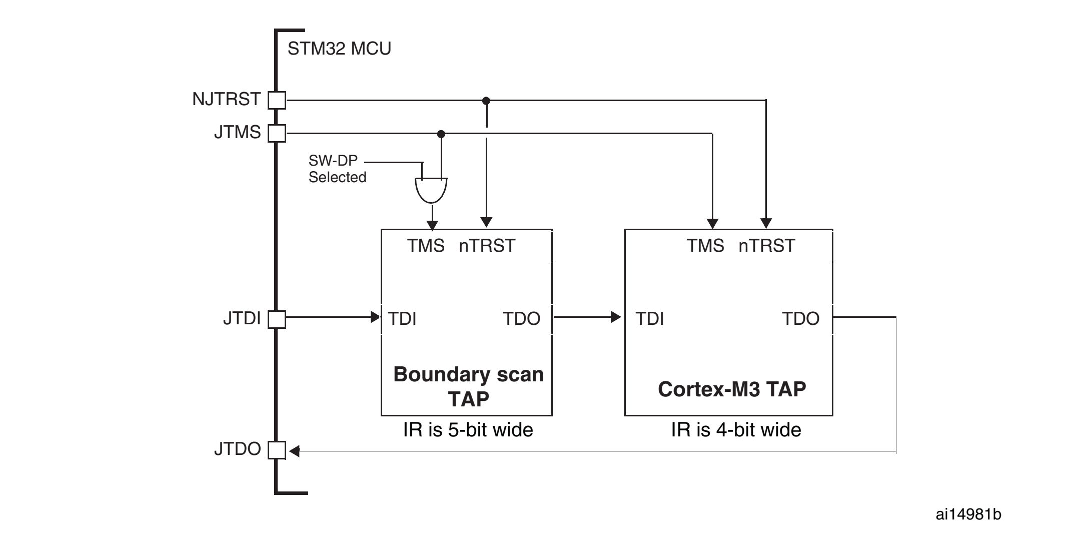
The STM32L1xxxx MCUs integrate two serially connected JTAG TAPs, the boundary scan TAP (IR is 5-bit wide) and the Cortex™-M3 TAP (IR is 4-bit wide).

To access the TAP of the Cortex™-M3 for debug purposes:

1. First, it is necessary to shift the BYPASS instruction of the boundary scan TAP.
2. Then, for each IR shift, the scan chain contains 9 bits (=5+4) and the unused TAP instruction must be shifted in using the BYPASS instruction.
3. For each data shift, the unused TAP, which is in BYPASS mode, adds 1 extra data bit in the data scan chain.

*Note: **Important**: Once Serial-Wire is selected using the dedicated ARM JTAG sequence, the boundary scan TAP is automatically disabled (JTMS forced high).*

Figure 289. JTAG TAP connections



29.6 ID codes and locking mechanism

There are several ID codes inside the STM32L1xxxx MCUs. ST strongly recommends tools designers to lock their debuggers using the MCU DEVICE ID code located in the external PPB memory map at address 0xE0042000.

29.6.1 MCU device ID code

The STM32L1xxxx MCUs integrate an MCU ID code. This ID identifies the ST MCU part-number and the die revision. It is part of the DBG_MCU component and is mapped on the external PPB bus (see *Section 29.16 on page 832*). This code is accessible using the JTAG debug port (4 to 5 pins) or the SW debug port (two pins) or by the user software. It is even accessible while the MCU is under system reset.

DBGMCU_IDCODE

Address: 0xE004 2000

Only 32-bits access supported. Read-only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID															
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
Reserved				DEV_ID											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **REV_ID(15:0)** Revision identifier

This field indicates the revision of the device:

Value	Low and Medium density devices	Medium density+ devices	High density devices
0x1000:	Rev A		Rev A
0x1008:	Rev Y		Rev Z
0x1018		Rev A	Rev Y
0x1038:	Rev W		
0x1078:	Rev V		

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV_ID(11:0)**: Device identifier

This field indicates the device ID.

0x416: Low and medium density device

0x427: Medium+ density device except devices in CSP63, BGA132 and LQFP144 packages

0x436: Medium+ density device for devices in CSP64, BGA132 and LQFP144 packages

0x436: High density device

29.6.2 Boundary scan TAP

JTAG ID code

The TAP of the STM32L1xxxx BSC (boundary scan) integrates a JTAG ID code equal to 0x4BA00477.

29.6.3 Cortex™-M3 TAP

The TAP of the ARM Cortex™-M3 integrates a JTAG ID code. This ID code is the ARM default one and has not been modified. This code is only accessible by the JTAG Debug Port.
This code is 0x4BA00477 (corresponds to Cortex™-M3 r2p0, see *Section 29.2: Reference ARM documentation*).

Only the DEV_ID(11:0) should be used for identification by the debugger/programmer tools.

29.6.4 Cortex™-M3 JEDEC-106 ID code

The ARM Cortex™-M3 integrates a JEDEC-106 ID code. It is located in the 4KB ROM table mapped on the internal PPB bus at address 0xE00FF000_0xE00FFFFF.

This code is accessible by the JTAG Debug Port (4 to 5 pins) or by the SW Debug Port (two pins) or by the user software.

29.7 JTAG debug port

A standard JTAG state machine is implemented with a 4-bit instruction register (IR) and five data registers (for full details, refer to the Cortex™-M3 r2p0 Technic*al Reference Manual (TRM), for references, please see Section 29.2: Reference ARM documentation)*.

Table 183. JTAG debug port data registers

IR(3:0)	Data register	Details
1111	BYPASS [1 bit]	
1110	IDCODE [32 bits]	ID CODE 0x4BA00477 (ARM Cortex™-M3 r2p0 ID Code)
1010	DPACC [35 bits]	Debug port access register This initiates a debug port and allows access to a debug port register. – When transferring data IN: Bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request Bits 2:1 = A[3:2] = 2-bit address of a debug port register. Bit 0 = RnW = Read request (1) or write request (0). – When transferring data OUT: Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: 010 = OK/FAULT 001 = WAIT OTHER = reserved Refer to *Table 184* for a description of the A(3:2) bits

Table 183. JTAG debug port data registers (continued)

IR(3:0)	Data register	Details
1011	APACC [35 bits]	Access port access register Initiates an access port and allows access to an access port register. – When transferring data IN: Bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request Bits 2:1 = A[3:2] = 2-bit address (sub-address AP registers). Bit 0 = RnW= Read request (1) or write request (0). – When transferring data OUT: Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: 010 = OK/FAULT 001 = WAIT OTHER = reserved There are many AP Registers (see AHB-AP) addressed as the combination of: – The shifted value A[3:2] – The current value of the DP SELECT register
1000	ABORT [35 bits]	Abort register – Bits 31:1 = Reserved – Bit 0 = DAPABORT: write 1 to generate a DAP abort.

Table 184. 32-bit debug port registers addressed through the shifted value A[3:2]

Address	A(3:2) value	Description
0x0	00	Reserved, must be kept at reset value.
0x4	01	DP CTRL/STAT register. Used to: – Request a system or debug power-up – Configure the transfer operation for AP accesses – Control the pushed compare and pushed verify operations. – Read some status flags (overrun, power-up acknowledges)
0x8	10	DP SELECT register: Used to select the current access port and the active 4-words register window. – Bits 31:24: APSEL: select the current AP – Bits 23:8: reserved – Bits 7:4: APBANKSEL: select the active 4-words register window on the current AP – Bits 3:0: reserved
0xC	11	DP RDBUFF register: Used to allow the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation)

29.8 SW debug port

29.8.1 SW protocol introduction

This synchronous serial protocol uses two pins:

- SWCLK: clock from host to target
- SWDIO: bidirectional

The protocol allows two banks of registers (DPACC registers and APACC registers) to be read and written to.

Bits are transferred LSB-first on the wire.

For SWDIO bidirectional management, the line must be pulled-up on the board (100 KΩ recommended by ARM).

Each time the direction of SWDIO changes in the protocol, a turnaround time is inserted where the line is not driven by the host nor the target. By default, this turnaround time is one bit time, however this can be adjusted by configuring the SWCLK frequency.

29.8.2 SW protocol sequence

Each sequence consist of three phases:

1. Packet request (8 bits) transmitted by the host
2. Acknowledge response (3 bits) transmitted by the target
3. Data transfer phase (33 bits) transmitted by the host or the target

Table 185. Packet request (8-bits)

Bit	Name	Description
0	Start	Must be “1”
1	APnDP	0: DP Access 1: AP Access
2	RnW	0: Write Request 1: Read Request
4:3	A(3:2)	Address field of the DP or AP registers (refer to *Table 184*)
5	Parity	Single bit parity of preceding bits
6	Stop	0
7	Park	Not driven by the host. Must be read as “1” by the target because of the pull-up

Refer to the Cortex™-M3r2p0 *TRM* for a detailed description of DPACC and APACC registers.

The packet request is always followed by the turnaround time (default 1 bit) where neither the host nor target drive the line.

Table 186. ACK response (3 bits)

Bit	Name	Description
0..2	ACK	001: FAULT 010: WAIT 100: OK

The ACK Response must be followed by a turnaround time only if it is a READ transaction or if a WAIT or FAULT acknowledge has been received.

Table 187. DATA transfer (33 bits)

Bit	Name	Description
0..31	WDATA or RDATA	Write or Read data
32	Parity	Single parity of the 32 data bits

The DATA transfer must be followed by a turnaround time only if it is a READ transaction.

29.8.3 SW-DP state machine (reset, idle states, ID code)

The State Machine of the SW-DP has an internal ID code which identifies the SW-DP. It follows the JEP-106 standard. This ID code is the default ARM one and is set to 0x4BA00477 (corresponding to Cortex™-M3 r2p0).

Note: Note that the SW-DP state machine is inactive until the target reads this ID code.

- The SW-DP state machine is in RESET STATE either after power-on reset, or after the DP has switched from JTAG to SWD or after the line is high for more than 50 cycles
- The SW-DP state machine is in IDLE STATE if the line is low for at least two cycles after RESET state.
- After RESET state, it is **mandatory** to first enter into an IDLE state AND to perform a READ access of the DP-SW ID CODE register. Otherwise, the target will issue a FAULT acknowledge response on another transactions.

Further details of the SW-DP state machine can be found in the *Cortex™-M3 r2p0 TRM* and the *CoreSight Design Kit r2p0 TRM*.

29.8.4 DP and AP read/write accesses

- Read accesses to the DP are not posted: the target response can be immediate (if ACK=OK) or can be delayed (if ACK=WAIT).
- Read accesses to the AP are posted. This means that the result of the access is returned on the next transfer. If the next access to be done is NOT an AP access, then the DP-RDBUFF register must be read to obtain the result.
  The READOK flag of the DP-CTRL/STAT register is updated on every AP read access or RDBUFF read request to know if the AP read access was successful.
- The SW-DP implements a write buffer (for both DP or AP writes), that enables it to accept a write operation even when other transactions are still outstanding. If the write buffer is full, the target acknowledge response is “WAIT”. With the exception of IDCODE read or CTRL/STAT read or ABORT write which are accepted even if the write buffer is full.
- Because of the asynchronous clock domains SWCLK and HCLK, two extra SWCLK cycles are needed after a write transaction (after the parity bit) to make the write effective internally. These cycles should be applied while driving the line low (IDLE state)
  This is particularly important when writing the CTRL/STAT for a power-up request. If the next transaction (requiring a power-up) occurs immediately, it will fail.

29.8.5 SW-DP registers

Access to these registers are initiated when APnDP=0

Table 188. SW-DP registers

A(3:2)	R/W	CTRLSEL bit of SELECT register	Register	Notes
00	Read		IDCODE	The manufacturer code is not set to ST code. 0x4BA00477 (identifies the SW-DP)
00	Write		ABORT	

Table 188. SW-DP registers (continued)

A(3:2)	R/W	CTRLSEL bit of SELECT register	Register	Notes
01	Read/Write	0	DP-CTRL/STAT	Purpose is to: – request a system or debug power-up – configure the transfer operation for AP accesses – control the pushed compare and pushed verify operations. – read some status flags (overrun, power-up acknowledges)
01	Read/Write	1	WIRE CONTROL	Purpose is to configure the physical serial port protocol (like the duration of the turnaround time)
10	Read		READ RESEND	Enables recovery of the read data from a corrupted debugger transfer, without repeating the original AP transfer.
10	Write		SELECT	The purpose is to select the current access port and the active 4-words register window
11	Read/Write		READ BUFFER	This read buffer is useful because AP accesses are posted (the result of a read AP request is available on the next AP transaction). This read buffer captures data from the AP, presented as the result of a previous read, without initiating a new transaction

29.8.6 SW-AP registers

Access to these registers are initiated when APnDP=1

There are many AP Registers (see AHB-AP) addressed as the combination of:

- The shifted value A[3:2]
- The current value of the DP SELECT register

29.9 AHB-AP (AHB access port) - valid for both JTAG-DP and SW-DP

Features:

- System access is independent of the processor status.
- Either SW-DP or JTAG-DP accesses AHB-AP.
- The AHB-AP is an AHB master into the Bus Matrix. Consequently, it can access all the data buses (Dcode Bus, System Bus, internal and external PPB bus) but the ICode bus.
- Bitband transactions are supported.
- AHB-AP transactions bypass the FPB.

The address of the 32-bits AHP-AP resisters are 6-bits wide (up to 64 words or 256 bytes) and consists of:

f) Bits [7:4] = the bits [7:4] APBANKSEL of the DP SELECT register

g) Bits [3:2] = the 2 address bits of A(3:2) of the 35-bit packet request for SW-DP.

The AHB-AP of the Cortex™-M3 includes 9 x 32-bits registers:

Table 189. Cortex™-M3 AHB-AP registers

Address offset	Register name	Notes
0x00	AHB-AP Control and Status Word	Configures and controls transfers through the AHB interface (size, hprot, status on current transfer, address increment type
0x04	AHB-AP Transfer Address	
0x0C	AHB-AP Data Read/Write	
0x10	AHB-AP Banked Data 0	Directly maps the 4 aligned data words without rewriting the Transfer Address Register.
0x14	AHB-AP Banked Data 1	
0x18	AHB-AP Banked Data 2	
0x1C	AHB-AP Banked Data 3	
0xF8	AHB-AP Debug ROM Address	Base Address of the debug interface
0xFC	AHB-AP ID Register	

Refer to the *Cortex™-M3* r2p0 *TRM* for further details.

29.10 Core debug

Core debug is accessed through the core debug registers. Debug access to these registers is by means of the *Advanced High-performance Bus* (AHB-AP) port. The processor can access these registers directly over the internal *Private Peripheral Bus* (PPB).

It consists of 4 registers:

Table 190. Core debug registers

Register	Description
DHCSR	The 32-bit Debug Halting Control and Status Register This provides status information about the state of the processor enable core debug halt and step the processor
DCRSR	The 17-bit Debug Core Register Selector Register: This selects the processor register to transfer data to or from.
DCRDR	The 32-bit Debug Core Register Data Register: This holds data for reading and writing registers to and from the processor selected by the DCRSR (Selector) register.
DEMCR	The 32-bit Debug Exception and Monitor Control Register: This provides Vector Catching and Debug Monitor Control. This register contains a bit named ***TRCENA*** which enable the use of a TRACE.

Note: ***Important****: these registers are not reset by a system reset. They are only reset by a power-on reset.*

Refer to the *Cortex™-M3 r2p0 TRM* for further details.

To Halt on reset, it is necessary to:

- enable the bit0 (VC_CORRESET) of the Debug and Exception Monitor Control Register
- enable the bit0 (C_DEBUGEN) of the Debug Halting Control and Status Register.

29.11 Capability of the debugger host to connect under system reset

The STM32L1xxxx MCUs’ reset system comprises the following reset sources:

- POR (power-on reset) which asserts a RESET at each power-up.
- Internal watchdog reset
- Software reset
- External reset

The Cortex™-M3 differentiates the reset of the debug part (generally PORRESETn) and the other one (SYSRESETn)

This way, it is possible for the debugger to connect under System Reset, programming the Core Debug Registers to halt the core when fetching the reset vector. Then the host can release the system reset and the core will immediately halt without having executed any instructions. In addition, it is possible to program any debug features under System Reset.

Note: It is highly recommended for the debugger host to connect (set a breakpoint in the reset vector) under system reset.

29.12 FPB (Flash patch breakpoint)

The FPB unit:

- implements hardware breakpoints
- patches code and data from code space to system space. This feature gives the possibility to correct software bugs located in the Code Memory Space.

The use of a Software Patch or a Hardware Breakpoint is exclusive.

The FPB consists of:

- 2 literal comparators for matching against literal loads from Code Space and remapping to a corresponding area in the System Space.
- 6 instruction comparators for matching against instruction fetches from Code Space. They can be used either to remap to a corresponding area in the System Space or to generate a Breakpoint Instruction to the core.

29.13 DWT (data watchpoint trigger)

The DWT unit consists of four comparators. They are configurable as:

- a hardware watchpoint or
- a trigger to an ETM or
- a PC sampler or
- a data address sampler

The DWT also provides some means to give some profiling informations. For this, some counters are accessible to give the number of:

- Clock cycle
- Folded instructions
- Load store unit (LSU) operations
- Sleep cycles
- CPI (clock per instructions)
- Interrupt overhead

29.14 ITM (instrumentation trace macrocell)

29.14.1 General description

The ITM is an application-driven trace source that supports *printf* style debugging to trace *Operating System* (OS) and application events, and emits diagnostic system information. The ITM emits trace information as packets which can be generated as:

- **Software trace.** Software can write directly to the ITM stimulus registers to emit packets.
- **Hardware trace.** The DWT generates these packets, and the ITM emits them.
- **Time stamping.** Timestamps are emitted relative to packets. The ITM contains a 21-bit counter to generate the timestamp. The Cortex™-M3 clock or the bit clock rate of the *Serial Wire Viewer* (SWV) output clocks the counter.

The packets emitted by the ITM are output to the TPIU (Trace Port Interface Unit). The formatter of the TPIU adds some extra packets (refer to TPIU) and then output the complete packets sequence to the debugger host.

The bit TRCEN of the Debug Exception and Monitor Control Register must be enabled before you program or use the ITM.

The SysTick timer clock is not stopped during the Stop mode debug (DBG_STOP bit set). The counter keeps on being decremented and can generate interrupts if they are enabled

29.14.2 Time stamp packets, synchronization and overflow packets

Time stamp packets encode time stamp information, generic control and synchronization. It uses a 21-bit timestamp counter (with possible prescalers) which is reset at each time stamp packet emission. This counter can be either clocked by the CPU clock or the SWV clock.

A synchronization packet consists of 6 bytes equal to 0x80_00_00_00_00_00 which is emitted to the TPIU as 00 00 00 00 00 80 (LSB emitted first).

A synchronization packet is a timestamp packet control. It is emitted at each DWT trigger.

For this, the DWT must be configured to trigger the ITM: the bit CYCCNTENA (bit0) of the DWT Control Register must be set. In addition, the bit2 (SYNCENA) of the ITM Trace Control Register must be set.

Note: If the SYNENA bit is not set, the DWT generates Synchronization triggers to the TPIU which will send only TPIU synchronization packets and not ITM synchronization packets.

An overflow packet consists is a special timestamp packets which indicates that data has been written but the FIFO was full.

Table 191. Main ITM registers

Address	Register	Details
@E0000FB0	ITM lock access	Write 0xC5ACCE55 to unlock Write Access to the other ITM registers
@E0000E80	ITM trace control	Bits 31-24 = Always 0
		Bits 23 = Busy
		Bits 22-16 = 7-bits ATB ID which identifies the source of the trace data.
		Bits 15-10 = Always 0
		Bits 9:8 = TSPrescale = Time Stamp Prescaler
		Bits 7-5 = Reserved
		Bit 4 = SWOENA = Enable SWV behavior (to clock the timestamp counter by the SWV clock).
		Bit 3 = DWTENA: Enable the DWT Stimulus
		Bit 2 = SYNCENA: this bit must be to 1 to enable the DWT to generate synchronization triggers so that the TPIU can then emit the synchronization packets.
		Bit 1 = TSENA (Timestamp Enable)
		Bit 0 = ITMENA: Global Enable Bit of the ITM
@E0000E40	ITM trace privilege	Bit 3: mask to enable tracing ports31:24
		Bit 2: mask to enable tracing ports23:16
		Bit 1: mask to enable tracing ports15:8
		Bit 0: mask to enable tracing ports7:0
@E0000E00	ITM trace enable	Each bit enables the corresponding Stimulus port to generate trace.
@E0000000- E000007C	Stimulus port registers 0-31	Write the 32-bits data on the selected Stimulus Port (32 available) to be traced out.

Example of configuration

To output a simple value to the TPIU:

- Configure the TPIU and assign TRACE I/Os by configuring the DBGMCU_CR (refer to *Section 29.17.2: TRACE pin assignment* and *Section 29.16.3: Debug MCU configuration register*)
- Write 0xC5ACCE55 to the ITM Lock Access Register to unlock the write access to the ITM registers
- Write 0x00010005 to the ITM Trace Control Register to enable the ITM with Sync enabled and an ATB ID different from 0x00
- Write 0x1 to the ITM Trace Enable Register to enable the Stimulus Port 0
- Write 0x1 to the ITM Trace Privilege Register to unmask stimulus ports 7:0
- Write the value to output in the Stimulus Port Register 0: this can be done by software (using a printf function)

29.15 ETM (Embedded trace macrocell)

29.15.1 General description

The ETM enables the reconstruction of program execution. Data are traced using the Data Watchpoint and Trace (DWT) component or the Instruction Trace Macrocell (ITM) whereas instructions are traced using the Embedded Trace Macrocell (ETM).

The ETM transmits information as packets and is triggered by embedded resources. These resources must be programmed independently and the trigger source is selected using the Trigger Event Register (0xE0041008). An event could be a simple event (address match from an address comparator) or a logic equation between 2 events. The trigger source is one of the fourth comparators of the DWT module, The following events can be monitored:

- Clock cycle matching
- Data address matching

For more informations on the trigger resources refer to *Section 29.13: DWT (data watchpoint trigger)*.

The packets transmitted by the ETM are output to the TPIU (Trace Port Interface Unit). The formatter of the TPIU adds some extra packets (refer to *Section 29.17: TPIU (trace port interface unit)*) and then outputs the complete packet sequence to the debugger host.

29.15.2 Signal protocol, packet types

This part is described in the chapter 7 ETMv3 Signal Protocol of the ARM IHI 0014N document.

29.15.3 Main ETM registers

For more information on registers refer to the chapter 3 of the ARM IHI 0014N specification.

Table 192. Main ETM registers

Address	Register	Details
0xE0041FB0	ETM Lock Access	Write 0xC5ACCE55 to unlock the write access to the other ETM registers.
0xE0041000	ETM Control	This register controls the general operation of the ETM, for instance how tracing is enabled.
0xE0041010	ETM Status	This register provides information about the current status of the trace and trigger logic.
0xE0041008	ETM Trigger Event	This register defines the event that will control trigger.
0xE004101C	ETM Trace Enable Control	This register defines which comparator is selected.
0xE0041020	ETM Trace Enable Event	This register defines the trace enabling event.
0xE0041024	ETM Trace Start/Stop	This register defines the traces used by the trigger source to start and stop the trace, respectively.

29.15.4 Configuration example

To output a simple value to the TPIU:

- Configure the TPIU and enable the I/IO_TRACEN to assign TRACE I/Os in the debug configuration register.
- Write 0xC5ACCE55 to the ETM Lock Access Register to unlock the write access to the ITM registers
- Write 0x00001D1E to the control register (configure the trace)
- Write 0000406F to the Trigger Event register (define the trigger event)
- Write 0000006F to the Trace Enable Event register (define an event to start/stop)
- Write 00000001 to the Trace Start/stop register (enable the trace)
- Write 0000191E to the ETM Control Register (end of configuration)

29.16 MCU debug component (DBGMCU)

The MCU debug component helps the debugger provide support for:

- Low-power modes
- Clock control for timers, watchdog (WWDG and IWDG) and I2Cs
- Control of the trace pins assignment

29.16.1 Debug support for low-power modes

To enter low-power mode, the instruction WFI or WFE must be executed.

The MCU implements several low-power modes which can either deactivate the CPU clock or reduce the power of the CPU.

The core does not allow FCLK or HCLK to be turned off during a debug session. As these are required for the debugger connection, during a debug, they must remain active. The MCU integrates special means to allow the user to debug software in low-power modes.

For this, the debugger host must first set some debug configuration registers to change the low-power mode behavior:

- In Sleep mode, DBG_SLEEP bit of DBGMCU_CR register must be previously set by the debugger. This will feed HCLK with the same clock that is provided to FCLK (system clock previously configured by the software).
- In Stop mode, the bit DBG_STOP must be previously set by the debugger. This will enable the internal RC oscillator clock to feed FCLK and HCLK in STOP mode.

29.16.2 Debug support for timers, watchdog and I^2C

During a breakpoint, it is necessary to choose how the counter of timers and watchdog should behave:

- They can continue to count inside a breakpoint. This is usually required when a PWM is controlling a motor, for example.
- They can stop to count inside a breakpoint. This is required for watchdog purposes.

For the I^2C, the user can choose to block the SMBUS timeout during a breakpoint.

29.16.3 Debug MCU configuration register

This register allows the configuration of the MCU under DEBUG. This concerns:

- Low-power mode support: Sleep, Stop and Standby modes
- Trace pin assignment

This DBGMCU_CR is mapped on the External PPB bus at address 0xE0042004

It is asynchronously reset by the PORESET (and not the system reset). It can be written by the debugger under system reset.

If the debugger host does not support these features, it is still possible for the user software to write to these registers.

DBGMCU_CR

Address: 0xE004 2004

Only 32-bit access supported

POR Reset: 0x0000 0000 (not reset by system reset)

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

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

Bits 7:5 **TRACE_MODE[1:0] and TRACE_IOEN**: Trace pin assignment control

– With TRACE_IOEN=0:

TRACE_MODE=xx: TRACE pins not assigned (default state)

– With TRACE_IOEN=1:

- TRACE_MODE=00: TRACE pin assignment for Asynchronous Mode
- TRACE_MODE=01: TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 1
- TRACE_MODE=10: TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 2
- TRACE_MODE=11: TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 4

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **DBG_STANDBY:** Debug Standby mode

0: (FCLK=Off, HCLK=Off) The whole digital part is unpowered.

From software point of view, exiting from Standby is identical than fetching reset vector (except a few status bit indicated that the MCU is resuming from Standby)

1: (FCLK=On, HCLK=On) In this case, the digital part is not unpowered and FCLK and HCLK are provided by the internal RC oscillator which remains active. In addition, the MCU generate a system reset during Standby mode so that exiting from Standby is identical than fetching from reset

Bit 1 **DBG_STOP:** Debug Stop mode

0: (FCLK=Off, HCLK=Off) In STOP mode, the clock controller disables all clocks (including HCLK and FCLK). When exiting from STOP mode, the clock configuration is identical to the one after RESET (CPU clocked by the 8 MHz internal RC oscillator (HSI)). Consequently, the software must reprogram the clock controller to enable the PLL, the Xtal, etc.

1: (FCLK=On, HCLK=On) In this case, when entering STOP mode, FCLK and HCLK are provided by the internal RC oscillator which remains active in STOP mode. When exiting STOP mode, the software must reprogram the clock controller to enable the PLL, the Xtal, etc. (in the same way it would do in case of DBG_STOP=0)

Bit 0 **DBG_SLEEP:** Debug Sleep mode

0: (FCLK=On, HCLK=Off) In Sleep mode, FCLK is clocked by the system clock as previously configured by the software while HCLK is disabled.

In Sleep mode, the clock controller configuration is not reset and remains in the previously programmed state. Consequently, when exiting from Sleep mode, the software does not need to reconfigure the clock controller.

1: (FCLK=On, HCLK=On) In this case, when entering Sleep mode, HCLK is fed by the same clock that is provided to FCLK (system clock as previously configured by the software).

29.16.4 Debug MCU APB1 freeze register (DBGMCU_APB1_FZ)

The DBGMCU_APB1_FZ register is used to configure the MCU under DEBUG. It concerns the APB1 peripherals:

- Timer clock counter freeze
- I2C SMBUS timeout freeze
- Window watchdog and independent watchdog counter freeze support

This DBGMCU_APB1_FZ is mapped on the external PPB bus at address 0xE0042008.

The register is asynchronously reset by the POR (and not the system reset). It can be written by the debugger under system reset.

Address: 0xE004 2008

Only 32-bit access are supported.

Power on reset (POR): 0x0000 0000 (not reset by system reset)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									DBG_I2C2_SMBUS_TIMEOUT	DBG_I2C1_SMBUS_TIMEOUT	Reserved				
									rw	rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_WWDG_STOP	Reserved				DBG_TIM7_STOP	DBG_TIM6_STOP	Reserved	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
			rw	rw	rw					rw	rw		rw	rw	rw

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

Bit 22 **DBG_I2C2_SMBUS_TIMEOUT:** SMBUS timeout mode stopped when core is halted

0: Same behavior as in normal mode
1: The SMBUS timeout is frozen

Bit 21 **DBG_I2C1_SMBUS_TIMEOUT:** SMBUS timeout mode stopped when core is halted

0: Same behavior as in normal mode
1: The SMBUS timeout is frozen

Bits 20:13 Reserved, must be kept at reset value.

Bit 12 **DBG_IWDG_STOP:** Debug independent watchdog stopped when core is halted

0: The independent watchdog counter clock continues even if the core is halted
1: The independent watchdog counter clock is stopped when the core is halted

Bit 11 **DBG_WWDG_STOP:** Debug window watchdog stopped when core is halted

0: The window watchdog counter clock continues even if the core is halted
1: The window watchdog counter clock is stopped when the core is halted

Bit 10 **DBG_RTC_STOP**: Debug RTC stopped when core is halted

0: The clock of the RTC counter is fed even if the core is halted
1: The clock of the RTC counter is stopped when the core is halted

Note: This bit is available only in high and medium+ density devices.

Bits 9:6 Reserved, must be kept at reset value.

Bit 5 **DBG_TIM7_STOP**: TIM7 counter stopped when core is halted

0: The counter clock of TIM7 is fed even if the core is halted
1: The counter clock of TIM7 is stopped when the core is halted

Bit 4 **DBG_TIM6_STOP**: TIM6 counter stopped when core is halted

0: The counter clock of TIM6 is fed even if the core is halted
1: The counter clock of TIM6 is stopped when the core is halted

Bit 3 **DBG_TIM5_STOP**: TIM5 counter stopped when core is halted

0: The counter clock of TIM5 is fed even if the core is halted
1: The counter clock of TIM5 is stopped when the core is halted

Note: This bit is available only in high and medium+ density devices.

Bit 2 **DBG_TIM4_STOP**: TIM4 counter stopped when core is halted

0: The counter clock of TIM4 is fed even if the core is halted
1: The counter clock of TIM4 is stopped when the core is halted

Bit 1 **DBG_TIM3_STOP**: TIM3 counter stopped when core is halted

0: The counter clock of TIM3 is fed even if the core is halted
1: The counter clock of TIM3 is stopped when the core is halted

Bit 0 **DBG_TIM2_STOP:** TIM2 counter stopped when core is halted

0: The counter clock of TIM2 is fed even if the core is halted
1: The counter clock of TIM2 is stopped when the core is halted

29.16.5 Debug MCU APB2 freeze register (DBGMCU_APB2_FZ)

The DBGMCU_APB2_FZ register is used to configure the MCU under DEBUG. It concerns APB2 peripherals:

- Timer clock counter freeze

This register is mapped on the external PPB bus at address 0xE004 200C

It is asynchronously reset by the POR (and not the system reset). It can be written by the debugger under system reset.

Address: 0xE004 200C

Only 32-bit access is supported.

POR: 0x0000 0000 (not reset by system reset)

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											DBG_TIM11_STOP	DBG_TIM10_STOP	DBG_TIM9_STOP	Reserved	
											rw	rw	rw		

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:2 **DBG_TIMx_STOP:** TIMx counter stopped when core is halted (x=9..11)
0: The clock of the involved timer counter is fed even if the core is halted
1: The clock of the involved timer counter is stopped when the core is halted

Bits 1:0 Reserved, must be kept at reset value.

29.17 TPIU (trace port interface unit)

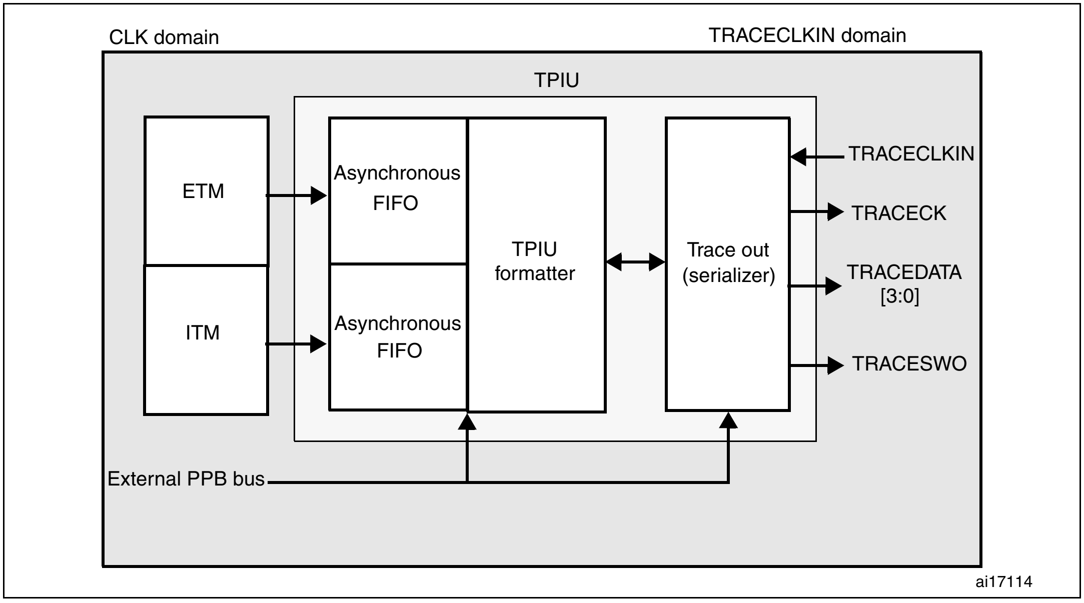
29.17.1 Introduction

The TPIU acts as a bridge between the on-chip trace data from the ITM and the ETM.

The output data stream encapsulates the trace source ID, that is then captured by a *trace port analyzer* (TPA).

The core embeds a simple TPIU, especially designed for low-cost debug (consisting of a special version of the CoreSight TPIU).

Figure 290. TPIU block diagram



29.17.2 TRACE pin assignment

- Asynchronous mode

  The asynchronous mode requires 1 extra pin and is available on all packages. It is only available if using Serial Wire mode (not in JTAG mode).

Table 193. Asynchronous TRACE pin assignment

TPUI pin name	Trace synchronous mode		STM32L1xxxx pin assignment
	Type	Description	
TRACESWO	O	TRACE Async Data Output	PB3

- Synchronous mode

  The synchronous mode requires from 2 to 6 extra pins depending on the data trace size and is only available in the larger packages. In addition it is available in JTAG mode and in Serial Wire mode and provides better bandwidth output capabilities than asynchronous trace.

Table 194. Synchronous TRACE pin assignment

TPUI pin name	Trace synchronous mode		STM32L1xxxx pin assignment
	Type	Description	
TRACECK	O	TRACE Clock	PE2
TRACED[3:0]	O	TRACE Sync Data Outputs Can be 1, 2 or 4.	PE[6:3]

TPUI TRACE pin assignment

By default, these pins are NOT assigned. They can be assigned by setting the TRACE_IOEN and TRACE_MODE bits in the ***MCU Debug component configuration register***. This configuration has to be done by the debugger host.

In addition, the number of pins to assign depends on the trace configuration (asynchronous or synchronous).

- **Asynchronous mode**: 1 extra pin is needed
- **Synchronous mode**: from 2 to 5 extra pins are needed depending on the size of the data trace port register (1, 2 or 4):
  - TRACECK
  - TRACED(0) if port size is configured to 1, 2 or 4
  - TRACED(1) if port size is configured to 2 or 4
  - TRACED(2) if port size is configured to 4
  - TRACED(3) if port size is configured to 4

To assign the TRACE pin, the debugger host must program the bits TRACE_IOEN and TRACE_MODE[1:0] of the Debug MCU configuration Register (DBGMCU_CR). By default the TRACE pins are not assigned.

This register is mapped on the external PPB and is reset by the PORESET (and not by the SYSTEM reset). It can be written by the debugger under SYSTEM reset.

Table 195. Flexible TRACE pin assignment

DBGMCU_CR register		**Pins assigned for:**	**TRACE IO pin assigned**					
TRACE_IOEN	**TRACE_MODE[1:0]**		**PB3** / JTDO/ TRACESWO	**PE2** / TRACECK	**PE3** / TRACED[0]	**PE4** / TRACED[1]	**PE5** / TRACED[2]	**PE6** / TRACED[3]
0	XX	No Trace (default state)	Released [1]					
1	00	Asynchronous Trace	TRACESWO			Released (usable as GPIO)		
1	01	Synchronous Trace 1 bit	Released [1]	TRACECK	TRACED[0]			
1	10	Synchronous Trace 2 bit		TRACECK	TRACED[0]	TRACED[1]		
1	11	Synchronous Trace 4 bit		TRACECK	TRACED[0]	TRACED[1]	TRACED[2]	TRACED[3]

1. When Serial Wire mode is used, it is released. But when JTAG is used, it is assigned to JTDO.

Note: By default, the TRACECLKIN input clock of the TPIU is tied to GND. It is assigned to HCLK two clock cycles after the bit TRACE_IOEN has been set.

The debugger must then program the Trace Mode by writing the PROTOCOL[1:0] bits in the SPP_R (Selected Pin Protocol) register of the TPIU.

- PROTOCOL=00: Trace Port Mode (synchronous)
- PROTOCOL=01 or 10: Serial Wire (Manchester or NRZ) Mode (asynchronous mode). Default state is 01

It then also configures the TRACE port size by writing the bits [3:0] in the CPSPS_R (Current Sync Port Size Register) of the TPIU:

- 0x1 for 1 pin (default state)
- 0x2 for 2 pins
- 0x8 for 4 pins

29.17.3 TPUI formatter

The formatter protocol outputs data in 16-byte frames:

- seven bytes of data
- eight bytes of mixed-use bytes consisting of:
  - 1 bit (LSB) to indicate it is a DATA byte (‘0) or an ID byte (‘1).
  - 7 bits (MSB) which can be data or change of source ID trace.
- one byte of auxiliary bits where each bit corresponds to one of the eight mixed-use bytes:
  - if the corresponding byte was a data, this bit gives bit0 of the data.
  - if the corresponding byte was an ID change, this bit indicates when that ID change takes effect.

Note: Refer to the ARM CoreSight Architecture Specification v1.0 (ARM IHI 0029B) for further information

29.17.4 TPUI frame synchronization packets

The TPUI can generate two types of synchronization packets:

- The Frame Synchronization packet (or Full Word Synchronization packet)

  It consists of the word: 0x7F_FF_FF_FF (LSB emitted first). This sequence can not occur at any other time provided that the ID source code 0x7F has not been used.

  It is output periodically ***between*** frames.

  In continuous mode, the TPA must discard all these frames once a synchronization frame has been found.
- The Half-Word Synchronization packet

  It consists of the half word: 0x7F_FF (LSB emitted first).

  It is output periodically ***between or within*** frames.

  These packets are only generated in continuous mode and enable the TPA to detect that the TRACE port is in IDLE mode (no TRACE to be captured). When detected by the TPA, it must be discarded.

29.17.5 Transmission of the synchronization frame packet

There is no Synchronization Counter register implemented in the TPIU of the core. Consequently, the synchronization trigger can only be generated by the **DWT**. Refer to the registers DWT Control Register (bits SYNCTAP[11:10]) and the DWT Current PC Sampler Cycle Count Register.

The TPUI Frame synchronization packet (0x7F_FF_FF_FF) is emitted:

- after each TPIU reset release. This reset is synchronously released with the rising edge of the TRACECLKIN clock. This means that this packet is transmitted when the TRACE_IOEN bit in the DBGMCU_CFG register is set. In this case, the word 0x7F_FF_FF_FF is not followed by any formatted packet.
- at each DWT trigger (assuming DWT has been previously configured). Two cases occur:
  - If the bit SYNENA of the ITM is reset, only the word 0x7F_FF_FF_FF is emitted without any formatted stream which follows.
  - If the bit SYNENA of the ITM is set, then the ITM synchronization packets will follow (0x80_00_00_00_00_00), formatted by the TPUI (trace source ID added).

29.17.6 Synchronous mode

The trace data output size can be configured to 4, 2 or 1 pin: TRACED(3:0)

The output clock is output to the debugger (TRACECK)

Here, TRACECLKIN is driven internally and is connected to HCLK only when TRACE is used.

Note: In this synchronous mode, it is not required to provide a stable clock frequency.

The TRACE I/Os (including TRACECK) are driven by the rising edge of TRACLKIN (equal to HCLK). Consequently, the output frequency of TRACECK is equal to HCLK/2.

29.17.7 Asynchronous mode

This is a low cost alternative to output the trace using only 1 pin: this is the asynchronous output pin TRACESWO. Obviously there is a limited bandwidth.

TRACESWO is multiplexed with JTDO when using the SW-DP pin. This way, this functionality is available in all STM32L1xxxx packages.

This asynchronous mode requires a constant frequency for TRACECLKIN. For the standard UART (NRZ) capture mechanism, 5% accuracy is needed. The Manchester encoded version is tolerant up to 10%.

29.17.8 TRACECLKIN connection inside the STM32L1xxxx

In the STM32L1xxxx, this TRACECLKIN input is internally connected to HCLK. This means that when in asynchronous trace mode, the application is restricted to use to time frames where the CPU frequency is stable.

Note: ***Important:*** *when using asynchronous trace: it is important to be aware that:*

The default clock of the STM32L1xxxx MCUs is the internal RC oscillator. Its frequency under reset is different from the one after reset release. This is because the RC calibration is the default one under system reset and is updated at each system reset release.

Consequently, the trace port analyzer (TPA) should not enable the trace (with the TRACE_IOEN bit) under system reset, because a Synchronization Frame Packet will be issued with a different bit time than trace packets which will be transmitted after reset release.

29.17.9 TPIU registers

The TPIU APB registers can be read and written only if the bit TRCENA of the Debug Exception and Monitor Control Register (DEMCR) is set. Otherwise, the registers are read as zero (the output of this bit enables the PCLK of the TPIU).

Table 196. Important TPIU registers

Address	Register	Description
0xE0040004	Current port size	Allows the trace port size to be selected: Bit 0: Port size = 1 Bit 1: Port size = 2 Bit 2: Port size = 3, not supported Bit 3: Port Size = 4 Only 1 bit must be set. By default, the port size is one bit. (0x00000001)
0xE00400F0	Selected pin protocol	Allows the Trace Port Protocol to be selected: Bit1:0= 00: Sync Trace Port Mode 01: Serial Wire Output - manchester (default value) 10: Serial Wire Output - NRZ 11: reserved
0xE0040304	Formatter and flush control	Bit 31-9 = always ‘0 Bit 8 = TrigIn = always ‘1 to indicate that triggers are indicated Bit 7-4 = always 0 Bit 3-2 = always 0 Bit 1 = EnFCont. In Sync Trace mode (Select_Pin_Protocol register bit1:0=00), this bit is forced to ‘1: the formatter is automatically enabled in continuous mode. In asynchronous mode (Select_Pin_Protocol register bit1:0 <> 00), this bit can be written to activate or not the formatter. Bit 0 = always 0 The resulting default value is 0x102 **Note:** In synchronous mode, because the TRACECTL pin is not mapped outside the chip, the formatter is always enabled in continuous mode -this way the formatter inserts some control packets to identify the source of the trace packets).
0xE0040300	Formatter and flush status	Not used in Cortex™-M3, always read as 0x00000008

29.17.10 Example of configuration

- Set the bit TRCENA in the Debug Exception and Monitor Control Register (DEMCR)
- Write the TPIU Current Port Size Register to the desired value (default is 0x1 for a 1-bit port size)
- Write TPIU Formatter and Flush Control Register to 0x102 (default value)
- Write the TPIU Select Pin Protocol to select the sync or async mode. Example: 0x2 for async NRZ mode (UART like)
- Write the DBGMCU control register to 0x20 (bit IO_TRACEN) to assign TRACE I/Os for async mode. A TPIU Sync packet is emitted at this time (FF_FF_FF_7F)
- Configure the ITM and write the ITM Stimulus register to output a value

29.18 DBG register map

The following table summarizes the Debug registers

Table 197. DBG register map and reset values

Addr.	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xE004 2000	DBGMCU_IDCODE	REV_ID																Reserved				DEV_ID											
	Reset value[1]	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0xE004 2004	DBGMCU_CR	Reserved																								TRACE_MODE [1:0]		TRACE_IOEN	Reserved		DBG_	DBG_	DBG_
	Reset value																									0	0	0			0	0	0
0xE004 2008	DBGMCU_APB1_FZ	Reserved									DBG_I2C2_SMBUS_TIMEOUT	DBG_I2C1_SMBUS_TIMEOUT	Reserved								DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_RTC_STOP	Reserved				DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
	Reset value										0	0									0	0	0					0	0	0	0	0	0

Table 197. DBG register map and reset values (continued)

Addr.	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xE004 200C	DBGMCU_APB2_FZ	Reserved																											DBG_TIM11_STOP	DBG_TIM10_STOP	DBG_TIM9_STOP	Reserved	
	Reset value																												0	0	0		

1. The reset value is product dependent. For more information, refer to *Section 29.6.1: MCU device ID code*.

30 Device electronic signature

This section applies to all STM32L1xxxx devices, unless otherwise specified.

The electronic signature is stored in the System memory area in the Flash memory module, and can be read using the JTAG/SWD or the CPU. It contains factory-programmed identification data that allow the user firmware or other external devices to automatically match its interface to the characteristics of the STM32L1xxxx microcontroller.

30.1 Memory size register

30.1.1 Flash size register

Base address: 0x1FF8004C for low and medium density devices

Base address: 0x1FF800CC for medium+ and high density devices

Read only = 0xXXXX where X is factory-programmed

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F_SIZE															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **F_SIZE:** Flash memory size

For DEV_ID = 0x416 or 0x427, this field value indicates the Flash memory size of the device in Kbytes.
Example: 0x0080 = 128 Kbytes.
For DEV_ID = 0x436, the field value can be ‘0’ or ‘1’, with ‘0’ for 384 Kbytes and ‘1’ for 256 Kbytes.

30.2 Unique device ID registers (96 bits)

The unique device identifier is ideally suited:

- for use as serial numbers
- for use as security keys in order to increase the security of code in Flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the internal Flash memory
- to activate secure boot processes, etc.

The 96-bit unique device identifier provides a reference number which is unique for any device and in any context. These bits can never be altered by the user.

The 96-bit unique device identifier can also be read in single bytes/half-words/words in different ways and then be concatenated using a custom algorithm.

Base address: 0x1FF80050 for low and medium density devices and 0x1FF800D0 for medium+ and high density devices

Address offset: 0x00

Read only = 0xXXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_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
U_ID(15:0)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **U_ID(31:0):** 31:0 unique ID bits

Address offset: 0x04

Read only = 0xXXXX where X is factory-programmed

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
U_ID(63:48)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
U_ID(47:32)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 63:32 **U_ID(63:32):** 63:32 unique ID bits

Address offset: 0x14

Read only = 0xXXXX XXXX where X is factory-programmed

95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80
U_ID(95:80)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
U_ID(79:64)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 95:64 **U_ID(95:64):** 95:64 unique ID bits

31 Revision history

Table 198. Document revision history

Date	Revision	Changes
02-Jul-2010	1	Initial release.
01-Oct-2010	2	Modified note in *Section 6.3.2* after *Section Table 24.* Updated *Figure 12: Clock tree on page 90* Modified *Table 19: Standby mode on page 78* (wakeup latency) Updated *Section 11.12: Temperature sensor on page 244* Updated SOF and SOFC bit descriptions in *Section 15.5: LCD registers on page 332* Updated *RTC register write protection on page 468* Updated I2C *Master receiver on page 626*
29-Nov-2010	3	Modified *Section 4.3.1: Behavior of clocks in low power modes* (65 kHz instead of 64 KHz) Modified *Section 4.3.9: Waking up the device from Stop and Standby modes using the RTC and comparators on page 78* Modified sequence orders in *RTC auto-wakeup (AWU) from the Stop mode on page 79* and *Section 4.4.1: PWR power control register (PWR_CR) on page 81* Modified *Section 5.2.3: MSI clock on page 92* Modified MSIRANGE bit description in *Section 5.3.2: Internal clock sources calibration register (RCC_ICSCR) on page 101* Modified PLS[2:0] bit description in *Section 4.4.1: PWR power control register (PWR_CR)*Modified *Section 5.2.6: LSI clock on page 94* Modified *Section 7.4.7: RI Hysteresis control register (RI_HYSCR4) on page 167* (“SCM” instead of “ST”) Modified *Section 7.5.7: SYSCFG register map on page 181* (“SYSCFG_MEMRMP” instead of “SYSCFG_MEMRM”) Updated JSQ bit description and added note in *Section 11.15.15: ADC injected sequence register (ADC_JSQR) on page 261* Modified *Figure 63: COMP2 interconnections (low and medium density devices) on page 294* odified*Section 13.4: Comparator 1 (COMP1) on page 292* and *Section 13.9.1: COMP comparator control and status register (COMP_CSR) on page 297* Modified *Section 15.2: LCD main features on page 312* Modified content of *Section 19: Real-time clock (RTC) on page 463* and changed bit and register names, added note on APB vs RTCCLK frequency in *Section 19.3.6 on page 470*.

Table 198. Document revision history (continued)

Date	Revision	Changes
29-Nov-2010	3 (continued)	Modified *Table 78: Min/max IWDG timeout period at 37 kHz (LSI) on page 507* Modified *Section : LIN reception on page 676* Modified note in *Structure and usage of packet buffers on page 549* Modified *Section : on page 819* Modified *Figure 66: Comparators in Window mode on page 296* Modified REV_ID(15:0) description in *Section 29.6.1: MCU device ID code on page 819* Added *Section 30: Device electronic signature on page 845* Added *Section 30.1.1: Flash size register on page 845*
24-Feb-2010	4	Modified *Table 35: Vector table (low and medium density devices) on page 189* (TIM9 and LCD) Modified *Figure 70: LCD controller block diagram on page 314* Modified PON[2:0], CC[2:0] and PS[3:0] bit description in *Section 15.5.2: LCD frame control register (LCD_FCR) on page 333* Modified *Section 4.3: Low-power modes* Modified *Section 6.3.13: Using the OSC32_IN/OSC32_OUT pins as GPIO PC14/PC15 port pins* and *Section 6.3.14: Using the OSC_IN/OSC_OUT pins as GPIO PH0/PH1 port pins* Modified *Section 12.1: DAC introduction on page 268* Added note 2 to *Section 15.2: LCD main features on page 312* Modified bit descriptions in *Section 16.4.17: TIMx DMA control register (TIMx_DCR) on page 393* Modified DMAB[15:0] bit description in *Section 16.4.18: TIMx DMA address for full transfer (TIMx_DMAR) on page 393* Modified *Section 25.3.7: DMA requests on page 632* Added note below *Figure 209: Transfer sequence diagram for slave receiver on page 623* Modified *Section : Closing slave communication on page 623* Modified *Section 25.6.6: I2C Status register 1 (I2C_SR1) on page 641* Added note to *Section 25.6.7: I2C Status register 2 (I2C_SR2) on page 645* Modified note in *Section 25.6.8: I2C Clock control register (I2C_CCR) on page 646* Modified *Section 27: Serial peripheral interface (SPI) on page 701* Added note below *Figure 209: Transfer sequence diagram for slave receiver on page 623*
17-Jan-2012	5	Moved *CRC calculation unit* to *Section 3* Added *Section 8: Touch sensing I/Os*, *Section 14: Operational amplifiers (OPAMP)*, *Section 22: Advanced encryption standard hardware accelerator (AES)*, *Section 24: Flexible static memory controller (FSMC)*, *Section 28: Secure digital input/output interface (SDIO)* for high density devices. Modified *Section 5.2.9: Clock security system (CSS) on page 95* Modified *Section 5.3.1: Clock control register (RCC_CR) on page 99*

Table 198. Document revision history (continued)

Date	Revision	Changes
13-Jul-2012	6	Updated for medium+ devices Added *Figure 2: System architecture (medium+ density devices) on page 39* Added *Table 4: Flash module organization (medium+ density devices) on page 47* Added *Figure 23: Routing interface (RI) block diagram for medium+ density devices on page 152* Added *Table 36: Vector table (medium+ density devices) on page 191* Added *Figure 32: DMA block diagram in medium+ density STM32L1xxxx devices on page 207* Removed V_{DDA} in *Section 4.2.3: Programmable voltage detector (PVD)* Replaced “pulse or pending” with “event or interrupt” in *Section 9.2: External interrupt/event controller (EXTI)* Replaced “simplex communication” and “simplex mode” with “half-duplex communication” and “half-duplex” mode in *Section 27: Serial peripheral interface (SPI)* Corrected *Figure 22: Routing interface (RI) block diagram for medium density devices on page 151* and *Figure 24: Routing interface (RI) block diagram for high-density devices on page 153* In *Section 8: Touch sensing I/Os*: replaced ‘36’ with ‘34’ in “supports up to 36 capacitive sensing channels” , replaced ‘6’ with ‘4’ in “One sampling capacitor for up to 6 capacitive sensing channels”, replaced “Compatible with touch” with “compatible with touchkey”, replaced “STM32 touch sensing” with “STM32L1xx STMTouch”, replaced “STM32 touch sensing library STM32L1xx” with STMTouch sensing library” Corrected connection to G1_IO2 pin in *Figure 26: Surface charge transfer analog IO group structure on page 184* Corrected display of the ETF[3:0] bit-field description in *Section 16.4.3: TIMx slave mode control register (TIMx_SMCR)* Added *Figure 211: Transfer sequence diagram for master receiver on page 627* Added a line with value = "0x1018” in *Section 29.6.1: MCU device ID code* Added line “Clear WUF bit...” in *Table 18: Stop mode on page 77* Modified RTCSEL bit-field description in *Section 5.3.14: Control/status register (RCC_CSR)* and changed “Reset value” to "Power-on reset value” Moved ‘Rev A’ label to line 0x1018 in the REV_ID bit field *Section 29.6.1: MCU device ID code on page 819* and added ‘Medium+’ in the description

Table 198. Document revision history (continued)

Date	Revision	Changes
13-Jul-2012	6	Corrected ‘2728 conversions’ in *Section 11.3.3: Channel selection on page 229* and changed ‘Due to internal connections...’ note in *Section 11.3: ADC functional description on page 225*. Updated arrow between OC2 mux and NOR gate in *Figure 28: Timer mode acquisition logic on page 187* Corrected ‘7.9370%’ in *Figure 146: Audio-frequency precision using standard 8 MHz HSE (high and medium+ density devices only) on page 737* Updated bit description for all CMR5 registers in *Section 7.4: RI registers on page 160* Replaced ‘CH31 GR7-1’, ‘COMP1_SW1’ and ‘CH31 GR11-5’ in *Section 7.4.2: RI analog switches control register (RI_ASCR1) on page 162* Modified description of bit 25:22, bit 5 and bit 4 in *Section 7.4.2: RI analog switches control register (RI_ASCR1) on page 162*, Modified description of Bit 5 SW1 in *Section 13.9.1: COMP comparator control and status register (COMP_CSR) on page 297* Modified cross reference to RI(RI_ASRCR1) section in *Section 14.3.2: Using the OPAMP outputs as ADC inputs on page 303* Updated *Table 31: RI register map and reset values on page 175* and *Table 57: COMP register map and reset values on page 300* Added bit 29 "GR5-4" and bit 15 "GR4-4" in *Section 7.4.3: RI analog switch control register 2 (RI_ASCR2) on page 164* Added ‘f_MSI range1’ in *Section 4.3.4: Low power run mode (LP run) on page 72*

Table 198. Document revision history (continued)

Date	Revision	Changes
14-Mar-2013	7	Updated description of OSPEEDR bits in *Section 6: General-purpose I/Os (GPIO)*. Updated max. input frequency in *Section 19.3.1: Clock and prescalers*. In *Section 24: Flexible static memory controller (FSMC)*: – Updated *Figure 186: FSMC block diagram*. – Updated *Section 24.5.4: NOR Flash/PSRAM controller asynchronous transactions*. – Modified differences between Mode B and mode 1 in *Section : Mode 2/B - NOR Flash*. – Modified differences between Mode C and mode 1 in *Section : Mode C - NOR Flash - OE toggling*. – Modified differences between Mode D and mode 1 in *Section : Mode D - asynchronous access with extended address*. – Updated NWAIT signal in *Figure 201: Asynchronous wait during a read access*, *Figure 202: Asynchronous wait during a write access*, *Figure 203: Wait configurations*, *Figure 204: Synchronous multiplexed read mode - NOR, PSRAM (CRAM)*, and *Figure 205: Synchronous multiplexed write mode - PSRAM (CRAM)*. – Updated *Section : SRAM/NOR-Flash chip-select control registers 1..4 (FSMC_BCR1..4)*. – Updated WAITEN definition in *Table 119: FSMC_BCRx bit fields* and *Table 121: FSMC_BCRx bit fields*. – Updated *Table 103: FSMC_BCRx bit fields* to *Table 122: FSMC_BTRx bit fields*. Updated Bits 11:0 description in *Section 29.6.1: MCU device ID code*. Updated *Section 30.1.1: Flash size register* and *Section 30.2: Unique device ID registers (96 bits)*.

Table 198. Document revision history (continued)

Date	Revision	Changes
19-Apr-2013	8	Added STM32L100xx value line Added reference to PM0056 in cover page Updated *Section 4.1.3: RTC and RTC backup registers* Updated 4.2 MHz in *Section 4.1.7: Voltage regulator and clock management when VDD drops below 2.0 V*, *Section 4.3.1: Behavior of clocks in low power modes* and *Section 5.2.8: System clock source frequency versus voltage range* Updated "...for each internal rest source" in *Section 5.1.2: Power reset* Removed first paragraph in *Section 7: System configuration controller (SYSCFG) and routing interface (RI)* Removed GR5-4 and GR4-4 bits and corrected bits 28:16 definition in *Section 7.4.3: RI analog switch control register 2 (RI_ASCR2)* Updated *Section 7.5.1: SYSCFG memory remap register (SYSCFG_MEMRMP)*, added FSMC to bit MEM_MODE Added LCD_CAPA to *Section 7.5.2: SYSCFG peripheral mode configuration register (SYSCFG_PMC)* Updated *Section 7.5.3: SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1)*, *Section 7.5.4: SYSCFG external interrupt configuration register 2 (SYSCFG_EXTICR2)*, *Section 7.5.5: SYSCFG external interrupt configuration register 3 (SYSCFG_EXTICR3)* and *Section 7.5.6: SYSCFG external interrupt configuration register 4 (SYSCFG_EXTICR4)* Updated *Section 8.1: Introduction* for value line devices Replaced 23 by 24 lines in *Section 9: Interrupts and events* Updated *Section 9.1: Nested vectored interrupt controller (NVIC)* Updated the number of GPIOs in *Section 9.2.5: External interrupt/event line mapping* Updated *Figure 30: External interrupt/event GPIO mapping* Added bit 23 to *Section 9.3: EXTI registers* Added *Section 17.4.14: Encoder interface mode (only for TIM9)* Replaced OPAMP_OPTR and OPAMP_CSR with OPAMP_OTR in *Section 14.3.3: Calibration* Updated bit 31 OT_USER in *Section 14.4.2: OPAMP offset trimming register for normal mode (OPAMP_OTR)* Removed first paragraph in *Section 11: Analog-to-digital converter (ADC)*, *Section 13: Comparators (COMP)*, *Section 14: Operational amplifiers (OPAMP)*, *Section 16: General-purpose timers (TIM2 to TIM5)*, *Section 19: Real-time clock (RTC)*, *Section 23: Universal serial bus full-speed device interface (USB)*, *Section 24: Flexible static memory controller (FSMC)*, *Section 27: Serial peripheral interface (SPI)* and *Section 28: Secure digital input/output interface (SDIO)* Added "VLCD rails..." bullet in *Section 15.2: LCD main features* Updated *Figure 71: 1/3 bias, 1/4 duty* Added *External decoupling* to *Section 15.4.5: Voltage generator*

Table 198. Document revision history (continued)

Date	Revision	Changes
19-Apr-2013	8	Updated *Figure 78: LCD voltage control* Added *Section 15.4.7: Double buffer memory* Removed Pulse mode in *Section 17.3: TIM10/TIM11 main features* Updated *Section 19.3.13: Tamper detection* Updated system reset value in *Section 19.6.4: RTC initialization and status register (RTC_ISR)* Changed min. value for address set to 0 in *Table 106*, *Table 107*, *Table 109*, *Table 110*, *Table 112* Updated DEV_ID = 0x436 in *Section 30.1.1: Flash size register*

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

ST PRODUCTS ARE NOT AUTHORIZED FOR USE IN WEAPONS. NOR ARE ST PRODUCTS DESIGNED OR AUTHORIZED FOR USE IN: (A) SAFETY CRITICAL APPLICATIONS SUCH AS LIFE SUPPORTING, ACTIVE IMPLANTED DEVICES OR SYSTEMS WITH PRODUCT FUNCTIONAL SAFETY REQUIREMENTS; (B) AERONAUTIC APPLICATIONS; (C) AUTOMOTIVE APPLICATIONS OR ENVIRONMENTS, AND/OR (D) AEROSPACE APPLICATIONS OR ENVIRONMENTS. WHERE ST PRODUCTS ARE NOT DESIGNED FOR SUCH USE, THE PURCHASER SHALL USE PRODUCTS AT PURCHASER’S SOLE RISK, EVEN IF ST HAS BEEN INFORMED IN WRITING OF SUCH USAGE, UNLESS A PRODUCT IS EXPRESSLY DESIGNATED BY ST AS BEING INTENDED FOR “AUTOMOTIVE, AUTOMOTIVE SAFETY OR MEDICAL” INDUSTRY DOMAINS ACCORDING TO ST PRODUCT DESIGN SPECIFICATIONS. PRODUCTS FORMALLY ESCC, QML OR JAN QUALIFIED ARE DEEMED SUITABLE FOR USE IN AEROSPACE BY THE CORRESPONDING GOVERNMENTAL AGENCY.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.
Information in this document supersedes and replaces all information previously supplied.
The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan -
Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com