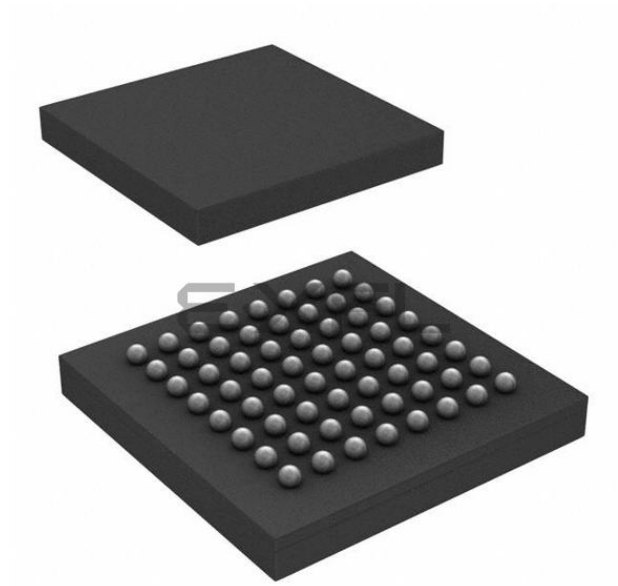


**Welcome to E-XFL.COM**

## What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

## Applications of "Embedded - Microcontrollers"

| Details | |
|---|---|
| Product Status | Active |
| Core Processor | ARM® Cortex®-M4 |
| Core Size | 32-Bit Single-Core |
| Speed | 80MHz |
| Connectivity | CANbus, I²C, IrDA, LINbus, MMC/SD, QSPI, SAI, SPI, SWPMI, UART/USART, USB |
| Peripherals | Brown-out Detect/Reset, DMA, LCD, PWM, WDT |
| Number of I/O | 52 |
| Program Memory Size | 256KB (256K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 64K x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.71V ~ 3.6V |
| Data Converters | A/D 16x12b; D/A 2x12b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 64-UFBGA |
| Supplier Device Package | 64-UFBGA (5x5) |
| Purchase URL | https://www.e-xfl.com/product-detail/stmicroelectronics/stm32l443rci6tr |

**Table 2. STM32L43xxx/44xxx/45xxx/46xxx memory map and peripheral register boundary addresses (continued)**

| Bus | Boundary address | Size (bytes) | Peripheral | Peripheral register map |
|---|---|---|---|---|
| APB1 | 0x4000 5400 - 0x4000 57FF | 1 KB | I2C1 | *Section 35.7.12: I2C register map* |
| | 0x4000 5000 - 0x4000 53FF | 1 KB | Reserved | - |
| | 0x4000 4C00 - 0x4000 4FFF | 1 KB | USART4 | *Section 36.8.12: USART register map* |
| | 0x4000 4800 - 0x4000 4BFF | 1 KB | USART3 | *Section 36.8.12: USART register map* |
| | 0x4000 4400 - 0x4000 47FF | 1 KB | USART2 | *Section 36.8.12: USART register map* |
| | 0x4000 4000 - 0x4000 43FF | 1 KB | Reserved | - |
| | 0x4000 3C00 - 0x4000 3FFF | 1 KB | SPI3 | *Section 38.6.8: SPI register map* |
| | 0x4000 3800 - 0x4000 3BFF | 1 KB | SPI2 | *Section 38.6.8: SPI register map* |
| | 0x4000 3400 - 0x4000 37FF | 1 KB | Reserved | - |
| | 0x4000 3000 - 0x4000 33FF | 1 KB | IWDG | *Section 32.4.6: IWDG register map* |
| | 0x4000 2C00 - 0x4000 2FFF | 1 KB | WWDG | *Section 33.4.4: WWDG register map* |
| | 0x4000 2800 - 0x4000 2BFF | 1 KB | RTC | *Section 34.6.21: RTC register map* |
| | 0x4000 2400 - 0x4000 27FF | 1 KB | LCD(6) | *Section 22.6.6: LCD register map* |
| | 0x4000 1800 - 0x4000 2400 | 3 KB | Reserved | - |
| | 0x4000 1400 - 0x4000 17FF | 1 KB | TIM7 | *Section 29.4.9: TIM6/TIM7 register map* |
| | 0x4000 1000 - 0x4000 13FF | 1 KB | TIM6 | *Section 29.4.9: TIM6/TIM7 register map* |
| | 0x4000 0800 - 0x4000 0FFF | 2 KB | Reserved | - |
| | 0x4000 0400 - 0x4000 07FF | 1 KB | TIM3 | *Section 27.4.21: TIMx register map* |
| | 0x4000 0000 - 0x4000 03FF | 1 KB | TIM2 | *Section 27.4.21: TIMx register map* |

1. Available on STM32L44xxx and STM32L46xxx devices only.
2. Not available on STM32L432xx and STM32L442xx devices.
3. Available on STM32L43xxx and STM32L44xxx devices only.
4. Available on STM32L45xxx and STM32L46xxx devices only.
5. Available on STM32L4x2xx and STM32L4x3xx devices only.
6. Available on STM32L4x3xx devices only.

## 2.3 Bit banding

The Cortex®-M4 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 STM32L43xxx/44xxx/45xxx/46xxx devices both the peripheral registers and the SRAM1 are mapped to a bit-band region, so that single bit-band write and read operations are allowed. The operations are only available for Cortex®-M4 accesses, and not from other bus masters (e.g. DMA).

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

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

where:

– *bit_word_addr* is the address of the word in the alias memory region that maps to the targeted bit
– *bit_band_base* is the starting address of the alias region
– *byte_offset* is the number of the byte in the bit-band region that contains the targeted bit
– *bit_number* is the bit position (0-7) of the targeted bit

**Example**

The following example shows how to map bit 2 of the byte located at SRAM1 address 0x20000300 to the alias region:

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

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

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

For more information on bit-banding, refer to the *Cortex*®-M4 *programming manual* (see *Related documents on page 1*).

## 2.4 Embedded SRAM

The STM32L43xxx/44xxx/45xxx/46xxx devices feature up to 196 Kbyte SRAM:

- Up to 128 Kbyte SRAM1
- Up to 32 Kbyte SRAM2.

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

The CPU can access the SRAM1 through the system bus or through the ICode/DCode buses when boot from SRAM1 is selected or when physical remap is selected (*Section 9.2.1: SYSCFG memory remap register (SYSCFG_MEMRMP)* in the SYSCFG controller). To get the maximum performance on SRAM1 execution, physical remap should be selected (boot or software selection).

Execution can be performed from SRAM2 with maximum performance without any remap thanks to access through ICode bus.

### 2.4.1 SRAM2 Parity check

The user can enable the SRAM2 parity check using the option bit SRAM2_PE in the user option byte (refer to *Section 3.4.1: Option bytes description*).

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

Bit 11 Reserved, must be kept cleared

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

These bits contain the VDD supply level threshold that activates/releases the reset.
000: BOR Level 0. Reset level threshold is around 1.7 V
001: BOR Level 1. Reset level threshold is around 2.0 V
010: BOR Level 2. Reset level threshold is around 2.2 V
011: BOR Level 3. Reset level threshold is around 2.5 V
100: BOR Level 4. Reset level threshold is around 2.8 V

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

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

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

### 3.7.9 Flash PCROP Start address register (FLASH_PCROP1SR)

Address offset: 0x24

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

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

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PCROP1_STRT[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:16 Reserved, must be kept cleared

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

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

The enable bit of each PLL output clock (PLLPEN, PLLQEN, PLLREN, PLLSAI1PEN, PLLSAI1QEN, PLLSAI1REN) can be modified at any time without stopping the corresponding PLL. PLLREN cannot be cleared if PLLCLK is used as system clock.

### 6.2.6 LSE clock

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

The LSE crystal is switched on and off using the LSEON bit in *Backup domain control register (RCC_BDCR)*. The crystal oscillator driving strength can be changed at runtime using the LSEDRV[1:0] bits in the *Backup domain control register (RCC_BDCR)* to obtain the best compromise between robustness and short start-up time on one side and low-power-consumption on the other side. The LSE drive can be decreased to the lower drive capability (LSEDRV=00) when the LSE is ON. However, once LSEDRV is selected, the drive capability can not be increased if LSEON=1.

The LSERDY flag in the *Backup domain control register (RCC_BDCR)* indicates whether the LSE crystal is stable or not. At startup, the LSE crystal output clock signal is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *Clock interrupt enable register (RCC_CIER)*.

#### External source (LSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 1 MHz. You select this mode by setting the LSEBYP and LSEON bits in the *AHB1 peripheral clocks enable in Sleep and Stop modes register (RCC_AHB1SMENR)*. The external clock signal (square, sinus or triangle) with ~50 % duty cycle has to drive the OSC32_IN pin while the OSC32_OUT pin can be used as GPIO. See *Figure 14*.

### 6.2.7 LSI clock

The LSI RC acts as a low-power clock source that can be kept running in Stop and Standby mode for the independent watchdog (IWDG), RTC and LCD. The clock frequency is 32 kHz. For more details, refer to the electrical characteristics section of the datasheets.

The LSI RC can be switched on and off using the LSION bit in the *Control/status register (RCC_CSR)*.

The LSIRDY flag in the *Control/status register (RCC_CSR)* indicates if the LSI oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *Clock interrupt enable register (RCC_CIER)*.

### 6.2.8 System clock (SYSCLK) selection

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

- MSI oscillator
- HSI16 oscillator
- HSE oscillator
- PLL

Bit 28 **SIOO**: Send instruction only once mode
See *Section 15.3.12: Sending the instruction only once on page 349*. This bit has no effect when IMODE = 00.
0: Send instruction on every transaction
1: Send instruction only for the first command
This field can be written only when BUSY = 0.

Bits 27:26 **FMODE[1:0]**: Functional mode
This field defines the QUADSPI functional mode of operation.
00: Indirect write mode
01: Indirect read mode
10: Automatic polling mode
11: Memory-mapped mode
If DMAEN = 1 already, then the DMA controller for the corresponding channel must be disabled before changing the FMODE value.
This field can be written only when BUSY = 0.

Bits 25:24 **DMODE[1:0]**: Data mode
This field defines the data phase's mode of operation:
00: No data
01: Data on a single line
10: Data on two lines
11: Data on four lines
This field also determines the dummy phase mode of operation.
This field can be written only when BUSY = 0.

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **DCYC[4:0]**: Number of dummy cycles
This field defines the duration of the dummy phase. In both SDR and DDR modes, it specifies a number of CLK cycles (0-31).
This field can be written only when BUSY = 0.

Bits 17:16 **ABSIZE[1:0]**: Alternate bytes size
This bit defines alternate bytes size:
00: 8-bit alternate byte
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes
This field can be written only when BUSY = 0.

Bits 15:14 **ABMODE[1:0]**: Alternate bytes mode
This field defines the alternate-bytes phase mode of operation:
00: No alternate bytes
01: Alternate bytes on a single line
10: Alternate bytes on two lines
11: Alternate bytes on four lines
This field can be written only when BUSY = 0.

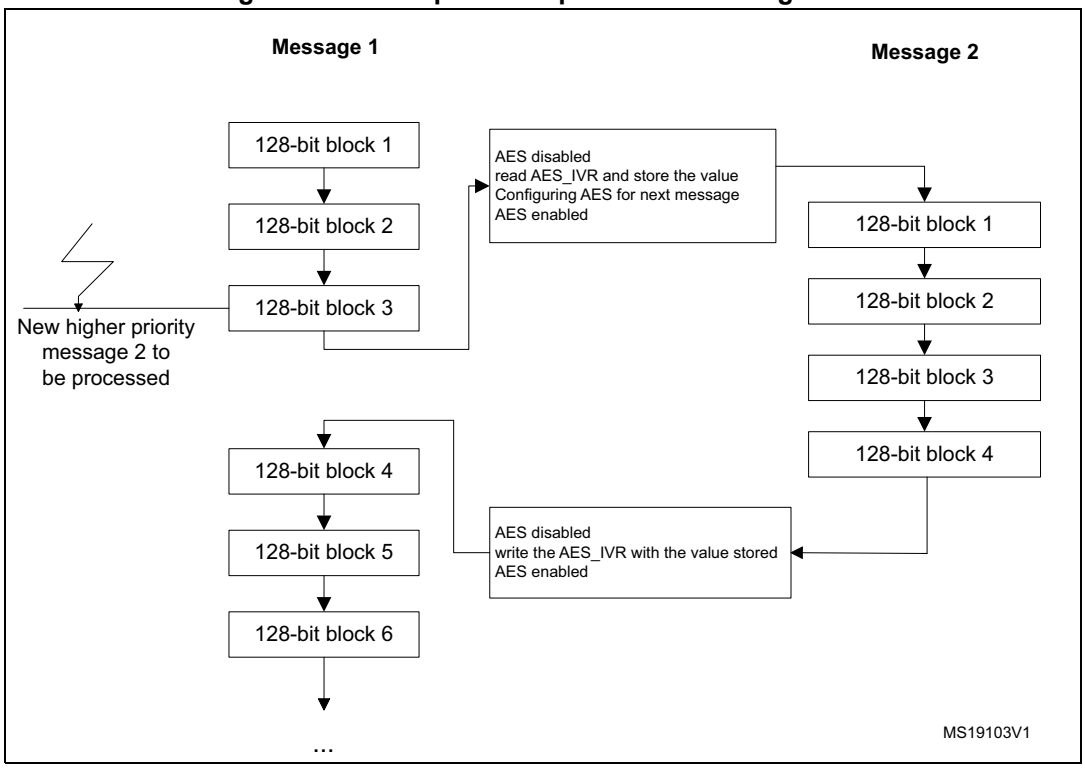
All the parameters of the context are defined into a single register ADC_JSQR and this register implements a queue of 2 buffers, allowing the bufferization of up to 2 sets of parameters:

- The JSQR register can be written at any moment even when injected conversions are ongoing.
- Each data written into the JSQR register is stored into the Queue of context.
- At the beginning, the Queue is empty and the first write access into the JSQR register immediately changes the context and the ADC is ready to receive injected triggers.
- Once an injected sequence is complete, the Queue is consumed and the context changes according to the next JSQR parameters stored in the Queue. This new context is applied for the next injected sequence of conversions.
- A Queue overflow occurs when writing into register JSQR while the Queue is full. This overflow is signaled by the assertion of the flag JQOVF. When an overflow occurs, the write access of JSQR register which has created the overflow is ignored and the queue of context is unchanged. An interrupt can be generated if bit JQOVFIE is set.
- Two possible behaviors are possible when the Queue becomes empty, depending on the value of the control bit JQM of register ADC_CFGR:
  - If JQM=0, the Queue is empty just after enabling the ADC, but then it can never be empty during run operations: the Queue always maintains the last active context and any further valid start of injected sequence will be served according to the last active context.
  - If JQM=1, the Queue can be empty after the end of an injected sequence or if the Queue is flushed. When this occurs, there is no more context in the queue and hardware triggers are disabled. Therefore, any further hardware injected triggers are ignored until the software re-writes a new injected context into JSQR register.
- Reading JSQR register returns the current JSQR context which is active at that moment. When the JSQR context is empty, JSQR is read as 0x0000.
- The Queue is flushed when stopping injected conversions by setting JADSTP=1 or when disabling the ADC by setting ADDIS=1:
  - If JQM=0, the Queue is maintained with the last active context.
  - If JQM=1, the Queue becomes empty and triggers are ignored.

*Note: When configured in discontinuous mode (bit JDISCEN=1), only the last trigger of the injected sequence changes the context and consumes the Queue.The 1st trigger only consumes the queue but others are still valid triggers as shown by the discontinuous mode example below (length = 3 for both contexts):*

- *1st trigger, discontinuous. Sequence 1: context 1 consumed, 1st conversion carried out*
- *2nd trigger, disc. Sequence 1: 2nd conversion.*
- *3rd trigger, discontinuous. Sequence 1: 3rd conversion.*
- *4th trigger, discontinuous. Sequence 2: context 2 consumed, 1st conversion carried out.*
- *5th trigger, discontinuous. Sequence 2: 2nd conversion.*
- *6th trigger, discontinuous. Sequence 2: 3rd conversion.*

**Figure 141. Example of suspend mode management**



Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred.

1: An active level has been detected on the break input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred.

1: Trigger interrupt pending.

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on COM event (when Capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.

0: No COM event occurred.

1: COM interrupt pending.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag

Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag

Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

Refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

**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 regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

– When CNT is reinitialized by a trigger event (refer to *Section 26.4.3: TIM1 slave mode control register (TIMx_SMCR)*), if URS=0 and UDIS=0 in the TIMx_CR1 register.

## 27.4 TIM2/TIM3 registers

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

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

### 27.4.1 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | UIF RE-MAP | Res. | CKD[1:0] | | ARPE | CMS | | DIR | OPM | URS | UDIS | CEN |
| | | | | rw | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD**: 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.*

### 30.7.8 LPTIM counter register (LPTIM_CNT)

Address offset: 0x1C

Reset value: 0x0000 0000

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

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT**: Counter value

When the LPTIM is running with an asynchronous clock, reading the LPTIM_CNT register may return unreliable values. So in this case it is necessary to perform two consecutive read accesses and verify that the two returned values are identical.

It should be noted that for a reliable LPTIM_CNT register read access, two consecutive read accesses must be performed and compared. A read access can be considered reliable when the values of the two consecutive read accesses are equal.

### 30.7.9 LPTIM1 option register (LPTIM1_OR)

Address offset: 0x20

Reset value: 0x0000 0000

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | OR_1 | OR_0 |
| | | | | | | | | | | | | | | rw | rw |

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

Bit 1 **OR_1**: Option register bit 1

0: LPTIM1 input 2 is connected to I/O

1: LPTIM1 input 2 is connected to COMP2_OUT

Bit 0 **OR_0**: Option register bit 0

0: LPTIM1 input 1 is connected to I/O

1: LPTIM1 input 1 is connected to COMP1_OUT

### 30.7.10 LPTIM2 option register (LPTIM2_OR)

Address offset: 0x20

Reset value: 0x0000 0000

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

Table 144. RTC functions over modes

| Pin | RTC functions | Functional in all low-power modes except Standby/Shutdown modes | Functional in Standby/Shutdown mode | Functional in $V_{BAT}$ mode |
|---|---|---|---|---|
| PC13 | RTC_TAMP1<br>RTC_TS<br>RTC_OUT | YES | YES | YES |
| PA0 | RTC_TAMP2 | YES | YES | YES |
| PE6 | RTC_TAMP3 | YES | YES | YES |
| PB2 | RTC_OUT | YES | NO | NO |
| PB15 | RTC_REFIN | YES | NO | NO |

### 34.3.3 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 6: 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 315: RTC block diagram*):

- A 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.
- A 15-bit synchronous prescaler configured through the PREDIV_S bits of the RTC_PRER register.

*Note: When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize consumption.*

The asynchronous prescaler division factor is set to 128, and the synchronous division factor to 256, to obtain an internal clock frequency of 1 Hz (ck_spre) with an LSE frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is $2^{22}$.

This corresponds to a maximum input frequency of around 4 MHz.

$f_{ck\_apre}$ is given by the following formula:

$$f_{CK\_APRE} = \frac{f_{RTCCLK}}{PREDIV\_A + 1}$$

The ck_apre clock is used to clock the binary RTC_SSR subseconds downcounter. When it reaches 0, RTC_SSR is reloaded with the content of PREDIV_S.

$f_{ck\_spre}$ is given by the following formula:

$$f_{CK\_SPRE} = \frac{f_{RTCCLK}}{(PREDIV\_S + 1) \times (PREDIV\_A + 1)}$$

**Table 152. I²C-SMBUS specification clock timings**

| Symbol | Parameter | Standard-mode (Sm) | | Fast-mode (Fm) | | Fast-mode Plus (Fm+) | | SMBUS | | Unit |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Min | Max | Min | Max | Min | Max | Min | Max | |
| $f_{SCL}$ | SCL clock frequency | - | 100 | - | 400 | - | 1000 | - | 100 | kHz |
| $t_{HD:STA}$ | Hold time (repeated) START condition | 4.0 | - | 0.6 | - | 0.26 | - | 4.0 | - | µs |
| $t_{SU:STA}$ | Set-up time for a repeated START condition | 4.7 | - | 0.6 | - | 0.26 | - | 4.7 | - | µs |
| $t_{SU:STO}$ | Set-up time for STOP condition | 4.0 | - | 0.6 | - | 0.26 | - | 4.0 | - | µs |
| $t_{BUF}$ | Bus free time between a STOP and START condition | 4.7 | - | 1.3 | - | 0.5 | - | 4.7 | - | µs |
| $t_{LOW}$ | Low period of the SCL clock | 4.7 | - | 1.3 | - | 0.5 | - | 4.7 | - | µs |
| $t_{HIGH}$ | Period of the SCL clock | 4.0 | - | 0.6 | - | 0.26 | - | 4.0 | 50 | µs |
| $t_r$ | Rise time of both SDA and SCL signals | - | 1000 | - | 300 | - | 120 | - | 1000 | ns |
| $t_f$ | Fall time of both SDA and SCL signals | - | 300 | - | 300 | - | 120 | - | 300 | ns |

*Note: SCLL is also used to generate the $t_{BUF}$ and $t_{SU:STA}$ timings.*

*SCLH is also used to generate the $t_{HD:STA}$ and $t_{SU:STO}$ timings.*

Refer to *Section 35.4.9: I2C_TIMINGR register configuration examples* for examples of I2C_TIMINGR settings vs. I2CCLK frequency.

### Master communication initialization (address phase)

In order to initiate the communication, the user must program the following parameters for the addressed slave in the I2C_CR2 register:

- Addressing mode (7-bit or 10-bit): ADD10
- Slave address to be sent: SADD[9:0]
- Transfer direction: RD_WRN
- In case of 10-bit address read: HEAD10R bit. HEAD10R must be configure to indicate if the complete address sequence must be sent, or only the header in case of a direction change.
- The number of bytes to be transferred: NBYTES[7:0]. If the number of bytes is equal to or greater than 255 bytes, NBYTES[7:0] must initially be filled with 0xFF.

The user must then set the START bit in I2C_CR2 register. Changing all the above bits is not allowed when START bit is set.

Then the master automatically sends the START condition followed by the slave address as soon as it detects that the bus is free (BUSY = 0) and after a delay of $t_{BUF}$.

In case of an arbitration loss, the master automatically switches back to slave mode and can acknowledge its own address if it is addressed as a slave.

at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6.

In parallel, another check is performed for each intermediate transition of RX line. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating auto baud rate detection, the USART_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART_CR2 register. The USART will then wait for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag will be set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The RXNE interrupt will signal the end of the operation.

At any later time, the auto baud rate detection may be relaunched by resetting the ABRF flag (by writing a 0).

*Note: If the USART is disabled (UE=0) during an auto baud rate operation, the BRR value may be corrupted.*

### 36.5.7 Multiprocessor communication using USART

In multiprocessor communication, the following bits are to be kept cleared:

- LINEN bit in the USART_CR2 register,
- HDSEL, IREN and SCEN bits in the USART_CR3 register.

It is possible to perform multiprocessor communication with the USART (with several USARTs connected in a network). For instance one of the USARTs can be the master, its TX output connected to the RX inputs of the other USARTs. 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 order to use the mute mode feature, the MME bit must be set in the USART_CR1 register.

In mute mode:

- None of the reception status bits can be set.
- All the receive interrupts are inhibited.
- The RWU bit in USART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USART_RQR register, under certain conditions.

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

Bit 4 **TXFRQ**: Transmit data flush request

Writing 1 to this bit sets the TXE flag.

This allows to discard the transmit data. This bit must be used only in Smartcard mode, when data has not been sent due to errors (NACK) and the FE flag is active in the USART_ISR register.

If the USART does not support Smartcard mode, this bit is reserved and forced by hardware to '0'. Please refer to *Section 36.4: USART implementation on page 1085*.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit clears the RXNE flag.

This allows to discard the received data without reading it, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the USART in mute mode and sets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

*Note: In the case the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software should wait for the TXE flag assertion before setting the SBKRQ bit.*

Bit 0 **ABRRQ**: Auto baud rate request

Writing 1 to this bit resets the ABRF flag in the USART_ISR and request an automatic baud rate measurement on the next received data frame.

*Note: If the USART does not support the auto baud rate feature, this bit is reserved and forced by hardware to '0'. Please refer to Section 36.4: USART implementation on page 1085.*

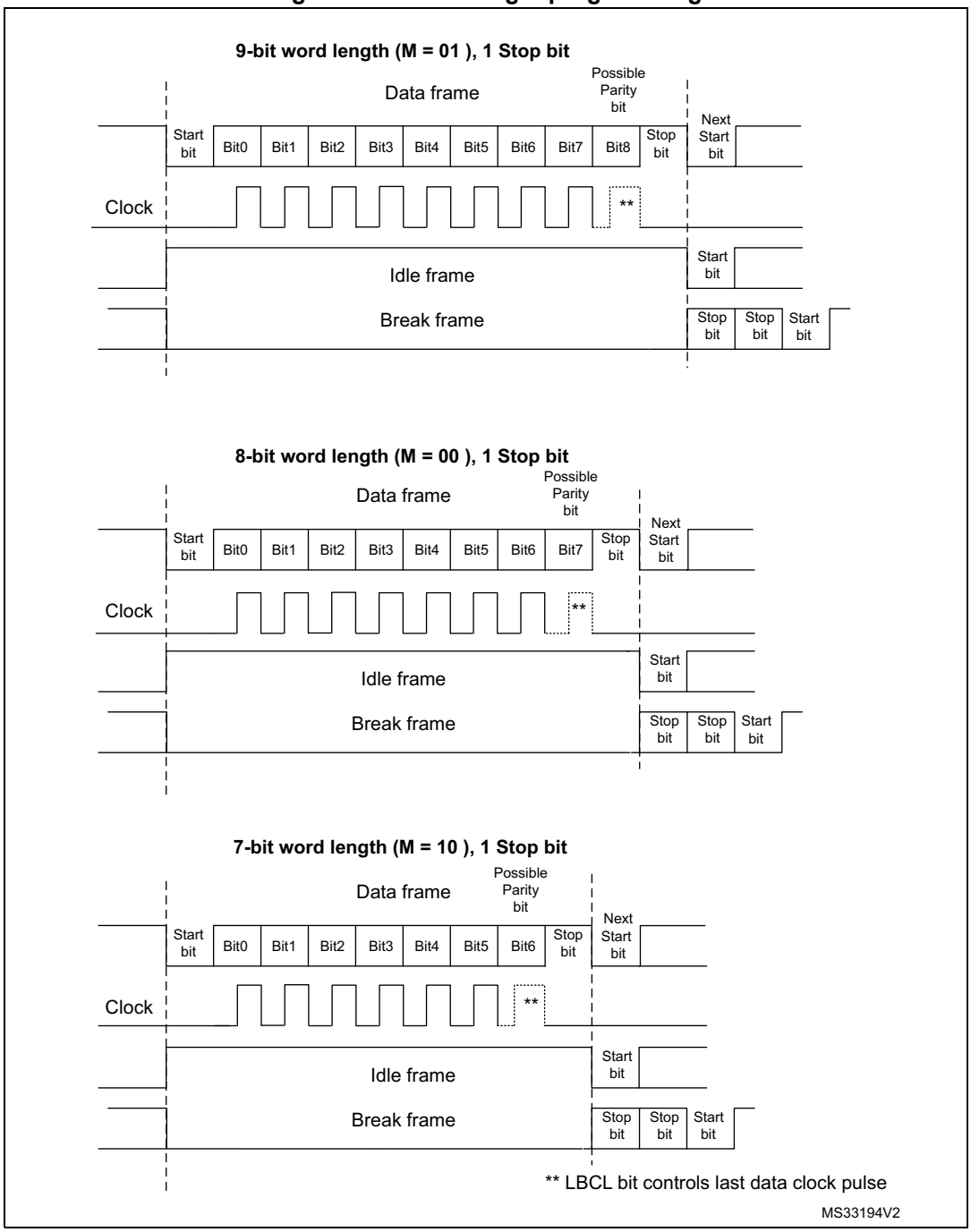
## 36.8.8 Interrupt and status register (USART_ISR)

Address offset: 0x1C

Reset value: 0x0200 00C0

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | Res. | Res. | TCBGT | Res. | Res. | REACK | TEACK | WUF | RWU | SBKF | CMF | BUSY |
| | | | | | | r | | | r | r | r | r | r | r | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ABRF | ABRE | Res. | EOBF | RTOF | CTS | CTSIF | LBDF | TXE | TC | RXNE | IDLE | ORE | NF | FE | PE |
| r | r | | r | r | r | r | r | r | r | r | r | r | r | r | r |

**Figure 373. Word length programming**



In order to work with n reception buffers in RAM, the DMA channel or stream must be configured in following mode (refer to DMA section):

- memory to memory mode disabled,
- memory increment mode enabled,
- memory size set to 32-bit,
- peripheral size set to 32-bit,
- peripheral increment mode disabled,
- circular mode enabled,
- data transfer direction set to read from peripheral,
- the number of words to be transfered must be set to 8 x n (8 words per buffer),
- the source address is the SWPMI_TDR register,
- the destination address is the buffer1 address in RAM

Then the user must:

1. Set RXDMA in the SWPMI_CR register
2. Set RXBFIE in the SWPMI_IER register
3. Enable stream or channel in the DMA module.

In the SWPMI interrupt routine, the user must check RXBFF in the SWPMI_ISR register. If it is set, the user must set CRXBFF bit in the SWPMI_ICR register to clear RXBFF flag and the user can read the first frame payload received in the first buffer (at the RAM address set in DMA2_CMAR1).

The number of data bytes in the payload is available in bits [23:16] of the last 8th word.

In the next SWPMI interrupt routine occurrence, the user will read the second frame received in the second buffer (address set in DMA2_CMAR1 + 8), and so on (refer to *Figure 425: SWPMI Multi software buffer mode reception*).

In case the application software cannot ensure to handle the SMPMI interrupt before the next frame reception, each buffer status is available in the most significant byte of the 8th buffer word:

- The CRC error flag (equivalent to RXBERF flag in the SWPMI_ISR register) is available in bit 24 of the 8th word. Refer to *Section 40.3.9: Error management* for an CRC error description.
- The receive overrun flag (equivalent to RXOVRF flag in the SWPMI_ISR register) is available in bit 25 of the 8th word. Refer to *Section 40.3.9: Error management* for an overrun error description.
- The receive buffer full flag (equivalent to RXBFF flag in the SWPMI_ISR register) is available in bit 26 of the 8th word.

In case of a CRC error, both RXBFF and RXBERF flags are set, thus bit 24 and bit 26 are set.

In case of an overrun, an overrun flag is set, thus bit 25 is set. The receive buffer full flag is set only in case of an overrun during the last word reception; then, both bit 25 and bit 26 are set for the current and the next frame reception.

The software can also read the DMA counter (number of data to transfer) in the DMA registers in order to retrieve the frame which has already been received and transferred into the RAM memory through DMA. For example, if the software works with 4 reception buffers,

*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 SDMMC_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 202. Data token format**

| Description | Start bit | Data | CRC16 | End bit |
|---|---|---|---|---|
| Block Data | 0 | - | yes | 1 |
| Stream Data | 0 | - | no | 1 |

# 42 Controller area network (bxCAN)

## 42.1 Introduction

The **Basic Extended CAN** peripheral, named **bxCAN**, interfaces the CAN network. It supports the CAN protocols version 2.0A and B. It has been designed to manage a high number of incoming messages efficiently with a minimum CPU load. It also meets the priority requirements for transmit messages.

For safety-critical applications, the CAN controller provides all hardware functions for supporting the CAN Time Triggered Communication option.

## 42.2 bxCAN main features

- Supports CAN protocol version 2.0 A, B Active
- Bit rates up to 1 Mbit/s
- Supports the Time Triggered Communication option

Transmission

- Three transmit mailboxes
- Configurable transmit priority
- Time Stamp on SOF transmission

Reception

- Two receive FIFOs with three stages
- Scalable filter banks:
  - 14 filter banks
- Identifier list feature
- Configurable FIFO overrun
- Time Stamp on SOF reception

Time-triggered communication option

- Disable automatic retransmission mode
- 16-bit free running timer
- Time Stamp sent in last two data bytes

Management

- Maskable interrupts
- Software-efficient mailbox mapping at a unique address space

Bit 3 **FULL1**: FIFO 1 full
Set by hardware when three messages are stored in the FIFO.
This bit is cleared by software.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **FMP1[1:0]**: FIFO 1 message pending
These bits indicate how many messages are pending in the receive FIFO1.
FMP1 is increased each time the hardware stores a new message in to the FIFO1. FMP is decreased each time the software releases the output mailbox by setting the RFOM1 bit.

### CAN interrupt enable register (CAN_IER)

Address offset: 0x14
Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | SLKIE | WKUIE |
| | | | | | | | | | | | | | | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ERRIE | Res. | Res. | Res. | LEC IE | BOF IE | EPV IE | EWG IE | Res. | FOV IE1 | FF IE1 | FMP IE1 | FOV IE0 | FF IE0 | FMP IE0 | TME IE |
| rw | | | | rw | rw | rw | rw | | rw | rw | rw | rw | rw | rw | rw |

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

Bit 17 **SLKIE**: Sleep interrupt enable
0: No interrupt when SLAKI bit is set.
1: Interrupt generated when SLAKI bit is set.

Bit 16 **WKUIE**: Wakeup interrupt enable
0: No interrupt when WKUI is set.
1: Interrupt generated when WKUI bit is set.

Bit 15 **ERRIE**: Error interrupt enable
0: No interrupt will be generated when an error condition is pending in the CAN_ESR.
1: An interrupt will be generation when an error condition is pending in the CAN_ESR.

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

Bit 11 **LECIE**: Last error code interrupt enable
0: ERRI bit will not be set when the error code in LEC[2:0] is set by hardware on error detection.
1: ERRI bit will be set when the error code in LEC[2:0] is set by hardware on error detection.

Bit 10 **BOFIE**: Bus-off interrupt enable
0: ERRI bit will not be set when BOFF is set.
1: ERRI bit will be set when BOFF is set.

Bit 9 **EPVIE**: Error passive interrupt enable
0: ERRI bit will not be set when EPVF is set.
1: ERRI bit will be set when EPVF is set.

**Table 232. bxCAN 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x184 | **CAN_TDT0R** | TIME[15:0] | | | | | | | | | | | | | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TGT | Res. | Res. | Res. | Res. | DLC[3:0] | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | - | - | - | - | - | - | - | x | - | - | - | - | x | x | x | x |
| 0x188 | **CAN_TDL0R** | DATA3[7:0] | | | | | | | | DATA2[7:0] | | | | | | | | DATA1[7:0] | | | | | | | | DATA0[7:0] | | | | | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 0x18C | **CAN_TDH0R** | DATA7[7:0] | | | | | | | | DATA6[7:0] | | | | | | | | DATA5[7:0] | | | | | | | | DATA4[7:0] | | | | | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 0x190 | **CAN_TI1R** | STID[10:0]/EXID[28:18] | | | | | | | | | | | EXID[17:0] | | | | | | | | | | | | | | | | | | IDE | RTR | TXRQ |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | 0 |
| 0x194 | **CAN_TDT1R** | TIME[15:0] | | | | | | | | | | | | | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TGT | Res. | Res. | Res. | Res. | DLC[3:0] | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | - | - | - | - | - | - | - | x | - | - | - | - | x | x | x | x |
| 0x198 | **CAN_TDL1R** | DATA3[7:0] | | | | | | | | DATA2[7:0] | | | | | | | | DATA1[7:0] | | | | | | | | DATA0[7:0] | | | | | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 0x19C | **CAN_TDH1R** | DATA7[7:0] | | | | | | | | DATA6[7:0] | | | | | | | | DATA5[7:0] | | | | | | | | DATA4[7:0] | | | | | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 0x1A0 | **CAN_TI2R** | STID[10:0]/EXID[28:18] | | | | | | | | | | | EXID[17:0] | | | | | | | | | | | | | | | | | | IDE | RTR | TXRQ |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | 0 |
| 0x1A4 | **CAN_TDT2R** | TIME[15:0] | | | | | | | | | | | | | | | | Res. | Res. | Res. | Res. | Res. | Res. | Res. | TGT | Res. | Res. | Res. | Res. | DLC[3:0] | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | - | - | - | - | - | - | - | x | - | - | - | - | x | x | x | x |
| 0x1A8 | **CAN_TDL2R** | DATA3[7:0] | | | | | | | | DATA2[7:0] | | | | | | | | DATA1[7:0] | | | | | | | | DATA0[7:0] | | | | | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 0x1AC | **CAN_TDH2R** | DATA7[7:0] | | | | | | | | DATA6[7:0] | | | | | | | | DATA5[7:0] | | | | | | | | DATA4[7:0] | | | | | | | |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| 0x1B0 | **CAN_RI0R** | STID[10:0]/EXID[28:18] | | | | | | | | | | | EXID[17:0] | | | | | | | | | | | | | | | | | | IDE | RTR | Res. |
| | Reset value | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | - |