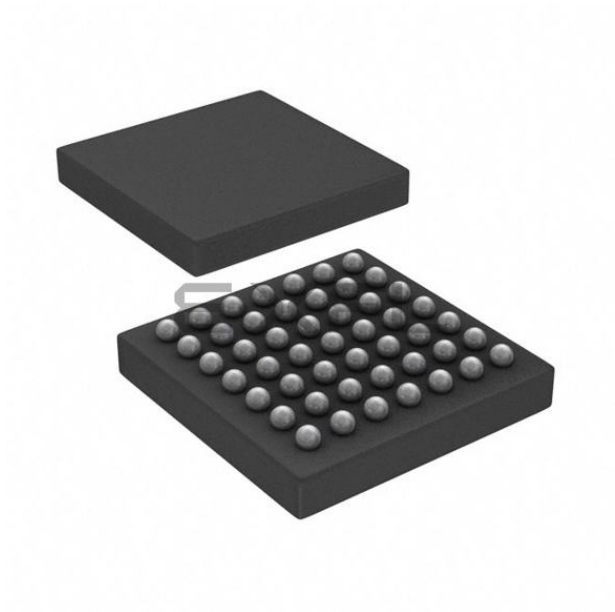
STMicroelectronics - STM32L433CBY6TR Datasheet



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	39
Program Memory Size	128KB (128K 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 10x12b; D/A 2x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	49-UFBGA, WLCSP
Supplier Device Package	49-WLCSP (3.14x3.13)
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/stm32l433cby6tr

The parity bits are computed and stored when writing into the SRAM2. Then, they are automatically checked when reading. If one bit fails, an NMI is generated. The same error can also be linked to the BRK_IN Break input of TIM1/TIM15/TIM16, with the SPL control bit in the *SYSCFG configuration register 2 (SYSCFG_CFGR2)*. The SRAM2 Parity Error flag (SPF) is available in the *SYSCFG configuration register 2 (SYSCFG_CFGR2)*.

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

2.4.2 SRAM2 Write protection

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

Table 3. SRAM2 organization

Page number	Start address	End address
Page 0	0x1000 0000	0x1000 03FF
Page 1	0x1000 0400	0x1000 07FF
Page 2	0x1000 0800	0x1000 0BFF
Page 3	0x1000 0C00	0x1000 0FFF
Page 4	0x1000 1000	0x1000 13FF
Page 5	0x1000 1400	0x1000 17FF
Page 6	0x1000 1800	0x1000 1BFF
Page 7	0x1000 1C00	0x1000 1FFF
Page 8	0x1000 2000	0x1000 23FF
Page 9	0x1000 2400	0x1000 27FF
Page 10	0x1000 2800	0x1000 2BFF
Page 11	0x1000 2C00	0x1000 2FFF
Page 12	0x1000 3000	0x1000 33FF
Page 13	0x1000 3400	0x1000 37FF
Page 14	0x1000 3800	0x1000 3BFF
Page 15	0x1000 3C00	0x1000 3FFF

Table 4. SRAM2 organization (continuation for STM32L45x and STM32L46x devices only)

Page number	Start address	End address
Page 16	0x1000 4000	0x1000 43FF
Page 17	0x1000 4400	0x1000 47FF
Page 18	0x1000 4800	0x1000 4BFF
Page 19	0x1000 4C00	0x1000 4FFF
Page 20	0x1000 5000	0x1000 53FF

Reset value: 0xF7E6 CE31

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
LPTIM1 SMEN	OPAMP SMEN	DAC1 SMEN	PWR SMEN	Res.	USB FSSM EN(1)	CAN1 SMEN	CRSS MEN	I2C3 SMEN	I2C2 SMEN (2)	I2C1 SMEN	Res.	UART4 SMEN(3)	USART3 SMEN(2)	USART2 SMEN	Res.
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw		rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 SMEN	SPI2 SMEN (2)	Res.	Res.	WWDG SMEN	RTCA PBSM EN	LCD SMEN (4)	Res.	Res.	Res.	TIM7 SMEN (5)	TIM6 SMEN	Res.	Res.	TIM3 SMEN(3)	TIM2 SMEN
rw	rw			rw	rw	rw				rw	rw			rw	rw

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

Bit 31 **LPTIM1SMEN**: Low power timer 1 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: LPTIM1 clocks disabled by the clock gating(1) during Sleep and Stop modes
1: LPTIM1 clocks enabled by the clock gating(1) during Sleep and Stop modes

Bit 30 **OPAMPSMEN**: OPAMP interface clocks enable during Sleep and Stop modes
Set and cleared by software.
0: OPAMP interface clocks disabled by the clock gating(1) during Sleep and Stop modes
1: OPAMP interface clocks enabled by the clock gating(1) during Sleep and Stop modes

Bit 29 **DAC1SMEN**: DAC1 interface clocks enable during Sleep and Stop modes
Set and cleared by software.
0: DAC1 interface clocks disabled by the clock gating(1) during Sleep and Stop modes
1: DAC1 interface clocks enabled by the clock gating(1) during Sleep and Stop modes

Bit 28 **PWRSMEN**: Power interface clocks enable during Sleep and Stop modes
Set and cleared by software.
0: Power interface clocks disabled by the clock gating(1) during Sleep and Stop modes
1: Power interface clocks enabled by the clock gating(1) during Sleep and Stop modes

Bit 27 Reserved, must be kept at reset value.

Bit 26 **USBFSSMEN**(2): USB FS clock enable during Sleep and Stop modes
Set and cleared by software.
0: USB FS clock disabled by the clock gating(1) during Sleep and Stop modes
1: USB FS clock enabled by the clock gating(1) during Sleep and Stop modes

Bit 25 **CAN1SMEN:** CAN1 clocks enable during Sleep and Stop modes
Set and cleared by software.
0: CAN1 clocks disabled by the clock gating(1) during Sleep and Stop modes
1: CAN1 clocks enabled by the clock gating(1) during Sleep and Stop modes

SDR mode

By default, the DDRM bit (QUADSPI_CCR[31]) is 0 and the QUADSPI operates in single data rate (SDR) mode.

In SDR mode, when the QUADSPI is driving the IO0/SO, IO1, IO2, IO3 signals, these signals transition only with the falling edge of CLK.

When receiving data in SDR mode, the QUADSPI assumes that the Flash memories also send the data using CLK's falling edge. By default (when SSHIFT = 0), the signals are sampled using the following (rising) edge of CLK.

DDR mode

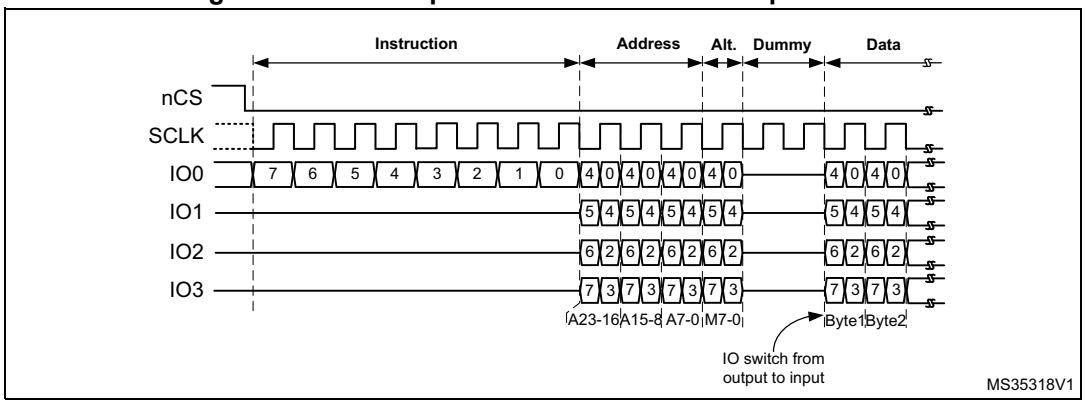
When the DDRM bit (QUADSPI_CCR[31]) is set to 1, the QUADSPI operates in double data rate (DDR) mode.

In DDR mode, when the QUADSPI is driving the IO0/SO, IO1, IO2, IO3 signals in the address/alternate-byte/data phases, a bit is sent on each of the falling and rising edges of CLK.

The instruction phase is not affected by DDRM. The instruction is always sent using CLK's falling edge.

When receiving data in DDR mode, the QUADSPI assumes that the Flash memories also send the data using both rising and falling CLK edges. When DDRM = 1, firmware must clear SSHIFT bit (bit 4 of QUADSPI_CR). Thus, the signals are sampled one half of a CLK cycle later (on the following, opposite edge).

Figure 34. An example of a DDR command in quad mode



Dual-flash mode

When the DFM bit (bit 6 of QUADSPI_CR) is 1, the QUADSPI is in dual-flash mode, where two external quad SPI Flash memories (FLASH 1 and FLASH 2) are used in order to send/receive 8 bits (or 16 bits in DDR mode) every cycle, effectively doubling the throughput as well as the capacity.

Each of the Flash memories use the same CLK and optionally the same nCS signals, but each have separate IO0, IO1, IO2, and IO3 signals.

Dual-flash mode can be used in conjunction with single-bit, dual-bit, and quad-bit modes, as well as with either SDR or DDR mode.

Note: In the above formula the settling to the desired code value with ½ LSB or accuracy requires 10 constant time for 12 bits resolution. For 8 bits resolution, the settling time is 7 constant time.

*The tolerated voltage drop during the hold phase "Vd" is represented by the number of LSBs after the capacitor discharging with the output leakage current. The settling back to the desired value with ½ LSB error accuracy requires ln(2*Nlsb) constant time of the DAC.*

The parameters "$T_{stab-BON}$","$T_{stab-BOFF}$", "R_{BON}" and "R_{BOFF}" are specified in the datasheet

Example of the sample and refresh time calculation with output buffer on

Note: The values used in the example below are provided as indication only. Please refer to the product datasheet for product data.

C_{load} = 100 nF

V_{DDA} = 3.0 V

Sampling phase:

$t_{sampling}$ = 7 µs + (10 * 2000 * 100 * 10^{-9}) = 2.007 ms
(where $T_{stab-BON}$ = 7 µs, RBON = 2 kΩ)

Refresh phase:

$t_{refresh}$ = 7 µs + (2000 * 100 * 10^{-9}) * ln(2*10) = 606.1 µs
(where N_{lsb} = 10 (10 LSB drop during the hold phase)

Hold phase:

D_v = i_{leak} * t_{hold} / C_{load} = 0.0073 V (10 LSB of 12bit at 3 V)
i_{leak} = 150 nA (worst case on the IO leakage on all the temperature range)
t_{hold} = 0.0073 * 100 * 10^{-9} / (150 * 10^{-9}) = 4.867 ms

19 Comparator (COMP)

19.1 Introduction

The device embeds two ultra-low power comparators COMP1, and COMP2

The comparators can be used for a variety of functions including:

- Wake-up from low-power mode triggered by an analog signal,
- Analog signal conditioning,
- Cycle-by-cycle current control loop when combined with a PWM output from a timer.

19.2 COMP main features

- Each comparator has configurable plus and minus inputs used for flexible voltage selection:
  - Multiplexed I/O pins
  - DAC Channel1 and Channel2
  - Internal reference voltage and three submultiple values (1/4, 1/2, 3/4) provided by scaler (buffered voltage divider)
- Programmable hysteresis
- Programmable speed / consumption
- The outputs can be redirected to an I/O or to timer inputs for triggering:
  - Break events for fast PWM shutdowns
- Comparator outputs with blanking source
- The two comparators can be combined in a window comparator
- Each comparator has interrupt generation capability with wake-up from Sleep and Stop modes (through the EXTI controller)

Bit 2 **CCUS**: Capture/compare control update selection
0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only
1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI
Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on TRGI, depending on the CCUS bit).
Note: This bit acts only on channels that have a complementary output.

26.4.3 TIM1 slave mode control register (TIMx_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMS[3]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **SMS[3]**: Slave mode selection - bit 3
Refer to SMS description - bits 2:0

Bit 15 **ETP**: External trigger polarity
This bit selects whether ETR or $\overline{ETR}$ is used for trigger operations
0: ETR is non-inverted, active at high level or rising edge.
1: ETR is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable
This bit enables External clock mode 2.
0: External clock mode 2 disabled
1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.
*Note: **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.*

26.4.17 TIM1 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/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r

Bits 15:0 **CCR4[15:0]**: Capture/Compare value

If channel CC4 is configured as output: CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC4 output.

If channel CC4 is configured as input: CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx_CCR4 register is read-only and cannot be programmed.

26.4.18 TIM1 break and dead-time register (TIMx_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	BK2P	BK2E	BK2F[3:0]				BKF[3:0]			
						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
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the bits BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **BK2P**: Break 2 polarity

0: Break input BRK2 is active low

1: Break input BRK2 is active high

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

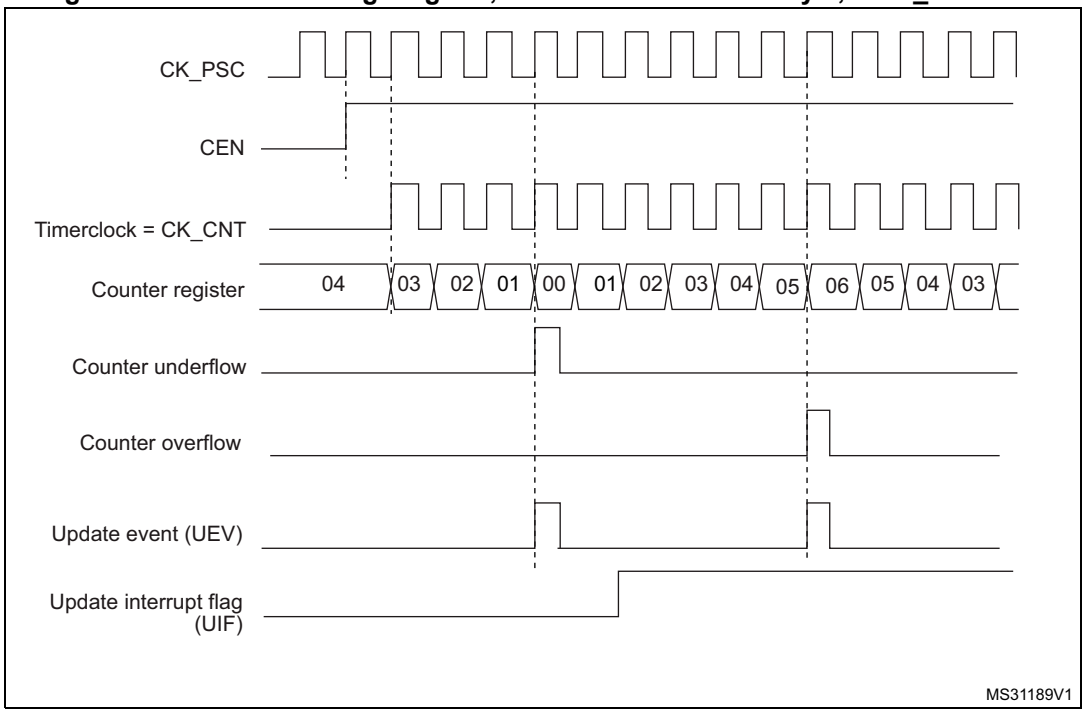
DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on the capture event.

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 227. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6



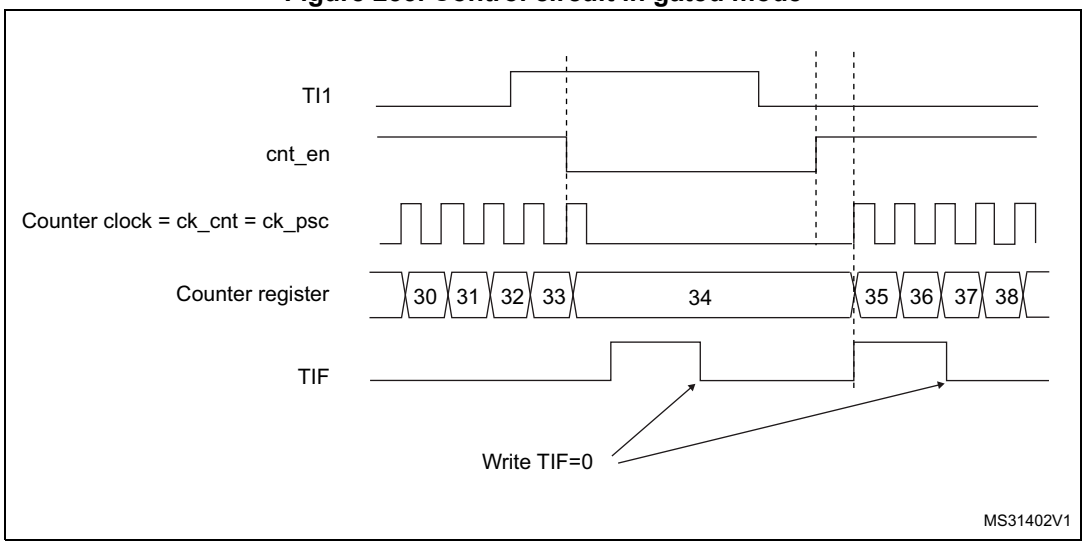
1. Here, center-aligned mode 1 is used (for more details refer to *Section 27.4.1: TIMx control register 1 (TIMx_CR1) on page 811*).

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. Write CC1P=1 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=101 in TIMx_SMCR register.
3. Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 253. Control circuit in gated mode



1. The configuration "CCxP=CCxNP=1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Note: The configuration "CCxP=CCxNP=1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we 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

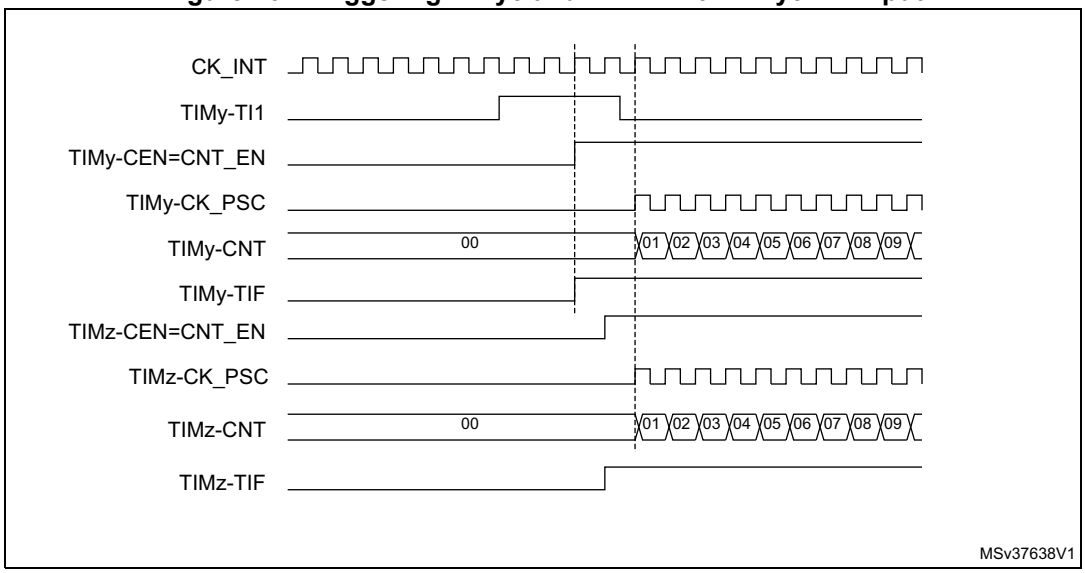
aligned, TIMy3 must be configured in Master/Slave mode (slave with respect to TI1, master with respect to TIMz2):

1. Configure TIMy3 master mode to send its Enable as trigger output (MMS=001 in the TIMy3_CR2 register).
2. Configure TIMy slave mode to get the input trigger from TI1 (TS=100 in the TIMy3_SMCR register).
3. Configure TIMy3 in trigger mode (SMS=110 in the TIMy3_SMCR register).
4. Configure the TIMy3 in Master/Slave mode by writing MSM=1 (TIMy3_SMCR register).
5. Configure TIMz2 to get the input trigger from TIMy (TS=000 in the TIMz2_SMCR register).
6. Configure TIMz2 in trigger mode (SMS=110 in the TIMz2_SMCR register).

When a rising edge occurs on TI1 (TIMy3), 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 TIMy3.

Figure 261. Triggering TIMy3 and TIMz2 with TIMy3 TI1 input



Note: The clock of the slave timer must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

27.3.20 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

Figure 270. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

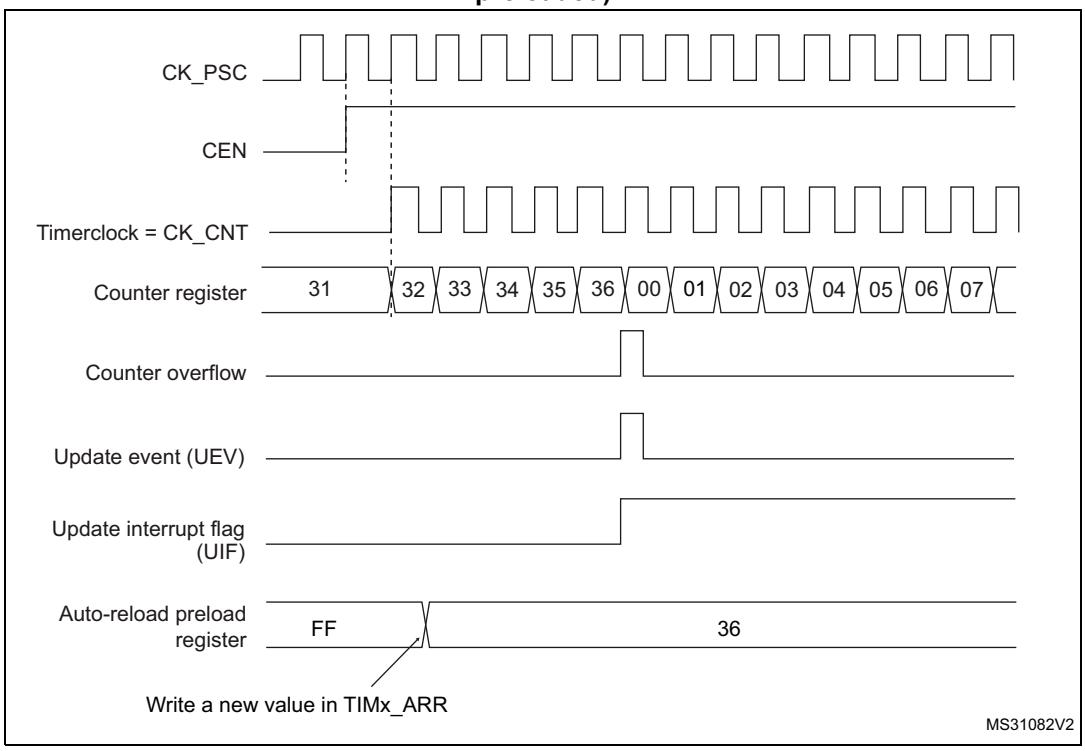


Figure 271. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)

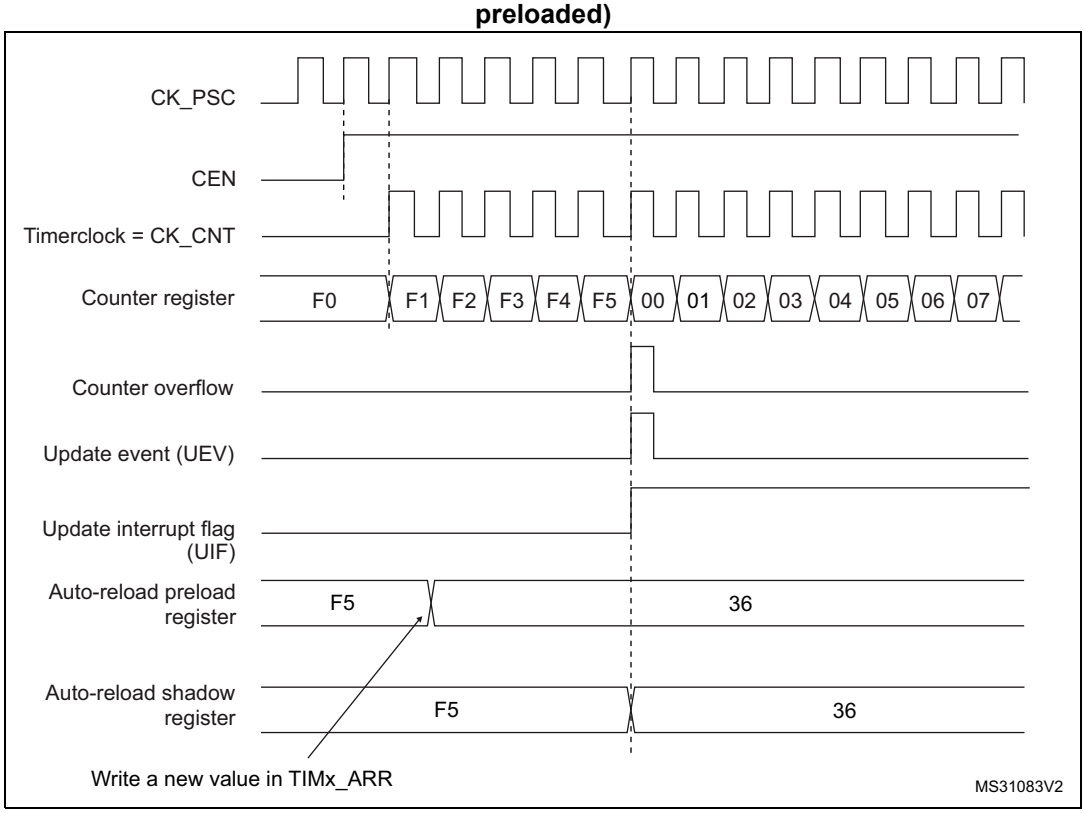
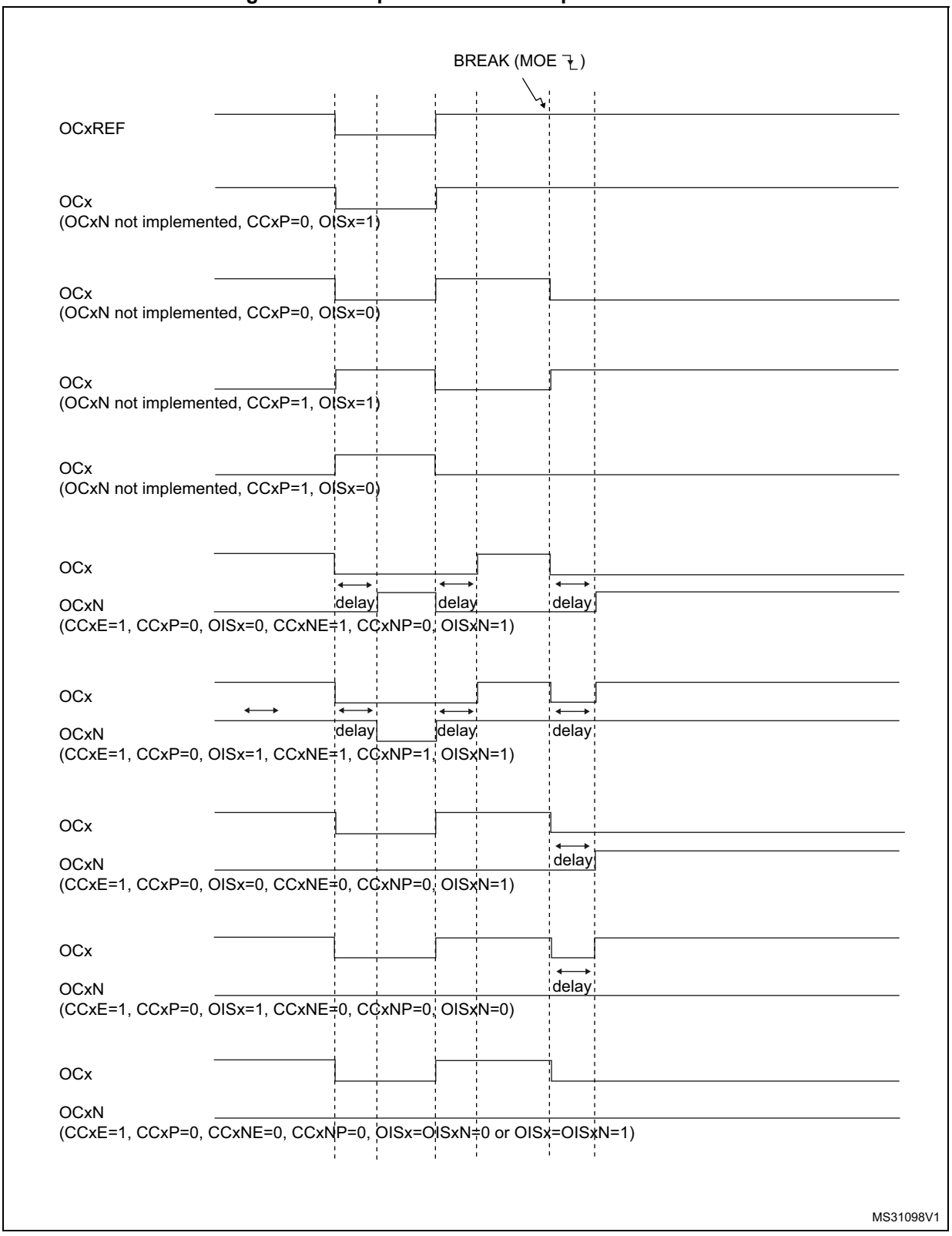


Figure 288. Output behavior in response to a break



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.

It is cleared either by software or by hardware in initialization mode.

0: Calendar shadow registers not yet synchronized

1: Calendar shadow registers synchronized

Bit 4 **INITS**: Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (Backup domain reset state).

0: Calendar has not been initialized

1: Calendar has been initialized

Bit 3 **SHPF**: Shift operation pending

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 register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

Figure 337. Transfer sequence flowchart for I2C master receiver for N >255 bytes

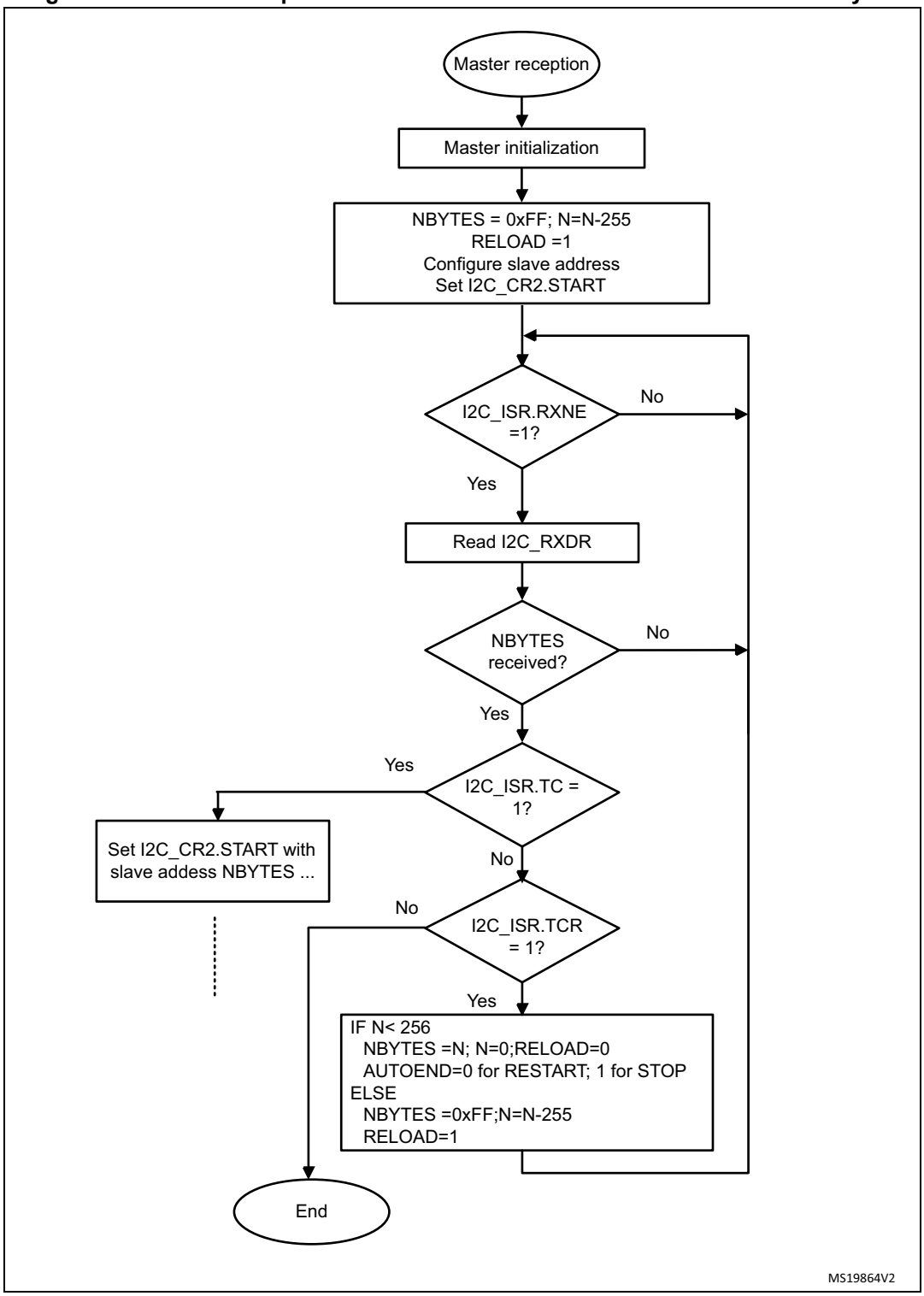


Table 162. I2C Interrupt requests

Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit
Receive buffer not empty	RXNE	Read I2C_RXDR register	RXIE
Transmit buffer interrupt status	TXIS	Write I2C_TXDR register	TXIE
Stop detection interrupt flag	STOPF	Write STOPCF=1	STOPIE
Transfer Complete Reload	TCR	Write I2C_CR2 with NBYTES[7:0] ≠ 0	TCIE
Transfer complete	TC	Write START=1 or STOP=1	
Address matched	ADDR	Write ADDRCF=1	ADDRIE
NACK reception	NACKF	Write NACKCF=1	NACKIE
Bus error	BERR	Write BERRCF=1	ERRIE
Arbitration loss	ARLO	Write ARLOCF=1	
Overrun/Underrun	OVR	Write OVRCF=1	
PEC error	PECERR	Write PECERRCF=1	
Timeout/t_{LOW} error	TIMEOUT	Write TIMEOUTCF=1	
SMBus Alert	ALERT	Write ALERTCF=1	

Depending on the product implementation, all these interrupts events can either share the same interrupt vector (I2C global interrupt), or be grouped into 2 interrupt vectors (I2C event interrupt and I2C error interrupt). Refer to *Table 45: STM32L43xxx/44xxx/45xxx/46xxx vector table* for details.

In order to enable the I2C interrupts, the following sequence is required:

1. Configure and enable the I2C IRQ channel in the NVIC.
2. Configure the I2C to generate interrupts.

The I2C wakeup event is connected to the EXTI controller (refer to *Section 13: Extended interrupts and events controller (EXTI)*).

Example 2

To obtain 921.6 Kbaud with f_{CK} = 48 MHz.

- In case of oversampling by 16:
  USARTDIV = 48 000 000/921 600
  BRR = USARTDIV = 52d = 34h
- In case of oversampling by 8:
  USARTDIV = 2 * 48 000 000/921 600
  USARTDIV = 104 (104d = 68h)
  BRR[3:0] = USARTDIV[3:0] >> 1 = 8h >> 1 = 4h
  BRR = 0x64

Table 166. Error calculation for programmed baud rates at f_{CK} = 72MHz in both cases of oversampling by 16 or by 8(1)

Baud rate		Oversampling by 16 (OVER8 = 0)			Oversampling by 8 (OVER8 = 1)		
S.No	Desired	Actual	BRR	% Error = (Calculated - Desired)B.Rate / Desired B.Rate	Actual	BRR	% Error
1	2.4 KBps	2.4 KBps	0x7530	0	2.4 KBps	0xEA60	0
2	9.6 KBps	9.6 KBps	0x1D4C	0	9.6 KBps	0x3A94	0
3	19.2 KBps	19.2 KBps	0xEA6	0	19.2 KBps	0x1D46	0
4	38.4 KBps	38.4 KBps	0x753	0	38.4 KBps	0xEA3	0
5	57.6 KBps	57.6 KBps	0x4E2	0	57.6 KBps	0x9C2	0
6	115.2 KBps	115.2 KBps	0x271	0	115.2 KBps	0x4E1	0
7	230.4 KBps	230.03KBps	0x139	0.16	230.4 KBps	0x270	0
8	460.8 KBps	461.54KBps	0x9C	0.16	460.06KBps	0x134	0.16
9	921.6 KBps	923.08KBps	0x4E	0.16	923.07KBps	0x96	0.16
10	2 MBps	2 MBps	0x24	0	2 MBps	0x44	0
11	3 MBps	3 MBps	0x18	0	3 MBps	0x30	0
12	4MBps	4MBps	0x12	0	4MBps	0x22	0
13	5MBps	N.A	N.A	N.A	4965.51KBps	0x16	0.69
14	6MBps	N.A	N.A	N.A	6MBps	0x14	0
15	7MBps	N.A	N.A	N.A	6857.14KBps	0x12	2
16	9MBps	N.A	N.A	N.A	9MBps	0x10	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.

Character transmission procedure

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the LPUART_BRR register.
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAT) in LPUART_CR3 if multibuffer Communication is to take place. Configure the DMA register as explained in multibuffer communication.
6. Set the TE bit in LPUART_CR1 to send an idle frame as first transmission.
7. Write the data to send in the LPUART_TDR register (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 LPUART_TDR register, wait until TC=1. This indicates that the transmission of the last frame is complete. This is required for instance when the LPUART 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 transmit data register.

The TXE bit is set by hardware and it indicates:

- The data has been moved from the LPUART_TDR register to the shift register and the data transmission has started.
- The LPUART_TDR register is empty.
- The next data can be written in the LPUART_TDR 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 LPUART_TDR register stores the data in the TDR register; next, the data is copied in the shift register at the end of the currently ongoing transmission.

When no transmission is taking place, a write instruction to the LPUART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

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 LPUART_CR1 register.

After writing the last data in the LPUART_TDR register, it is mandatory to wait for TC=1 before disabling the LPUART or causing the microcontroller to enter the low-power mode (see *Figure 350: TC/TXE behavior when transmitting*).

37.4.5 Tolerance of the LPUART receiver to clock deviation

The asynchronous receiver of the LPUART works correctly only if the total clock system deviation is less than the tolerance of the LPUART receiver. The causes which contribute to the total deviation are:

- DTRA: Deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: Error due to the baud rate quantization of the receiver
- DREC: Deviation of the receiver'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)

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{LPUART receiver tolerance}$$

where

DWU is the error due to sampling point deviation when the wakeup from Stop mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WULPUART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WULPUART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WULPUART}}{9 \times Tbit}$$

$t_{WULPUART}$ is the time between detecting the wakeup event and both clock (requested by the peripheral) and regulator ready.

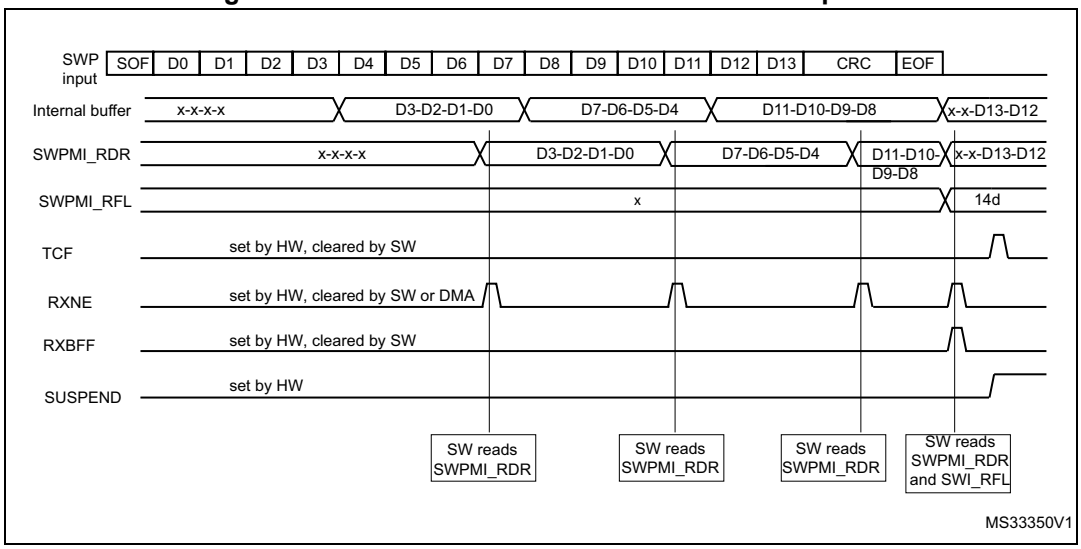
The LPUART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 175*:

- 7, 8 or 9-bit character length defined by the M bits in the LPUARTx_CR1 register
- 1 or 2 stop bits

Table 175. Tolerance of the LPUART receiver

M bits	768 ≤ BRR <1024	1024 ≤ BRR < 2048	2048 ≤ BRR < 4096	4096 ≤ BRR
8 bits (M=00), 1 stop bit	1.82%	2.56%	3.90%	4.42%
9 bits (M=01), 1 stop bit	1.69%	2.33%	2.53%	4.14%
7 bits (M=10), 1 stop bit	2.08%	2.86%	4.35%	4.42%
8 bits (M=00), 2 stop bit	2.08%	2.86%	4.35%	4.42%
9 bits (M=01), 2 stop bit	1.82%	2.56%	3.90%	4.42%
7 bits (M=10), 2stop bit	2.34%	3.23%	4.92%	4.42%

Figure 423. SWPMI No software buffer mode reception



Single software buffer mode

This mode allows to receive a complete SWP frame without any CPU intervention using the DMA. The DMA transfers received data from the 32-bit SWPMI_RDR register to the RAM memory, and the software can poll the end of the frame reception using the SWPMI_RBFF flag.

The Single software buffer mode is selected by setting RXDMA bit and clearing RXMODE bit in the SWPMI_CR register.

The DMA must be configured as follows:

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 disabled,
- data transfer direction set to read from peripheral,
- the number of words to be transfered must be set to 8,
- the source address is the SWPMI_RDR register,
- the destination address is the SWP frame buffer in RAM.

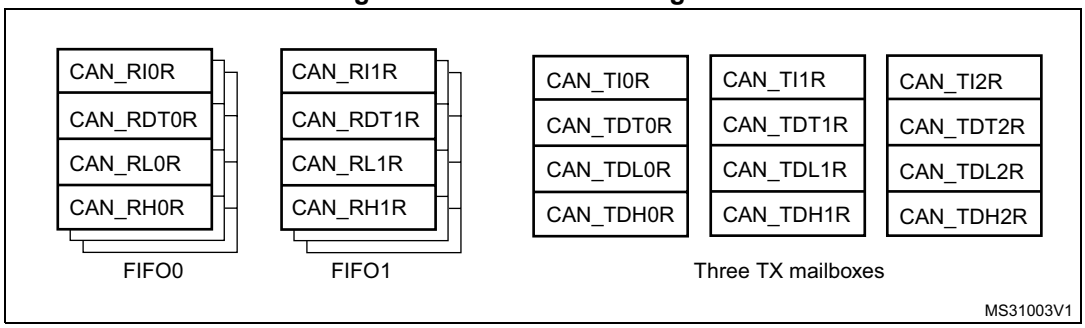
Then the user must:

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

A DMA request is issued by SWPMI when RXNE flag is set in SWPMI_ISR. The RXNE flag is cleared automatically when the DMA is reading the SWPMI_RDR register.

In the SWPMI interrupt routine, the user must check RXBFF bit in the SWPMI_ISR register. If it is set, the user must:

Figure 455. Can mailbox registers



CAN TX mailbox identifier register (CAN_TIxR) (x = 0..2)

Address offsets: 0x180, 0x190, 0x1A0
Reset value: 0xXXXX XXXX (except bit 0, TXRQ = 0)

All TX registers are write protected when the mailbox is pending transmission (TMEx reset).

This register also implements the TX request control (bit 0) - reset value 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STID[10:0]/EXID[28:18]											EXID[17:13]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXID[12:0]													IDE	RTR	TXRQ
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 **STID[10:0]/EXID[28:18]**: Standard identifier or extended identifier
The standard identifier or the MSBs of the extended identifier (depending on the IDE bit value).

Bit 20:3 **EXID[17:0]**: Extended identifier
The LSBs of the extended identifier.

Bit 2 **IDE**: Identifier extension
This bit defines the identifier type of message in the mailbox.
0: Standard identifier.
1: Extended identifier.

Bit 1 **RTR**: Remote transmission request
0: Data frame
1: Remote frame

Bit 0 **TXRQ**: Transmit mailbox request
Set by software to request the transmission for the corresponding mailbox.
Cleared by hardware when the mailbox becomes empty.