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®-M0
Core Size	32-Bit Single-Core
Speed	48MHz
Connectivity	CANbus, I²C, IrDA, LINbus, SPI, UART/USART
Peripherals	DMA, I²S, POR, PWM, WDT
Number of I/O	38
Program Memory Size	128KB (128K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	32K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 3.6V
Data Converters	A/D 13x12b; D/A 2x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	48-LQFP
Supplier Device Package	48-LQFP (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/stm32f091cbt7tr

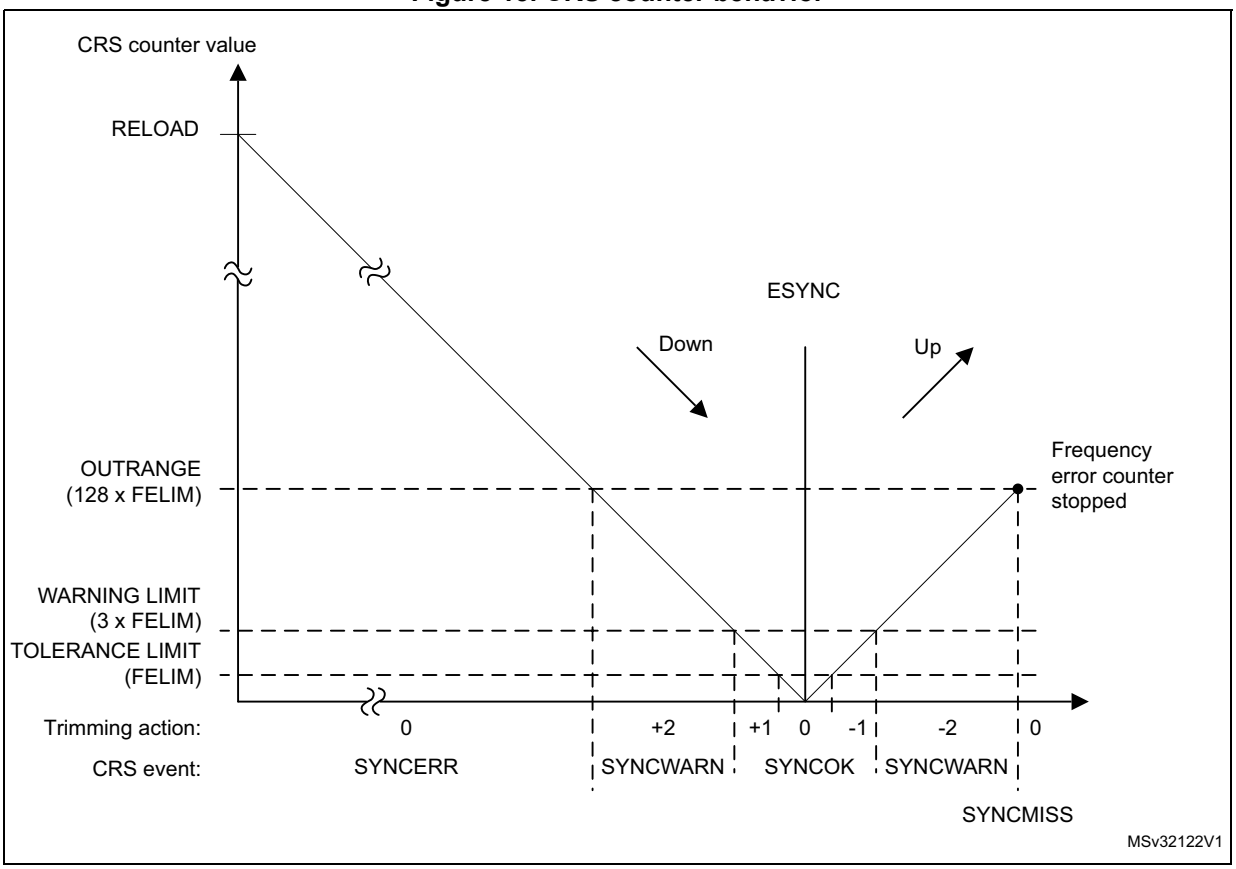
It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS_CR register.

7.3.3 Frequency error measurement

The frequency error counter is a 16-bit down/up counter which is reloaded with the RELOAD value on each SYNC event. It starts counting down till it reaches the zero value, where the ESYNC (expected synchronization) event is generated. Then it starts counting up to the OUTRANGE limit where it eventually stops (if no SYNC event is received) and generates a SYNCMISS event. The OUTRANGE limit is defined as the frequency error limit (FELIM field of the CRS_CFGR register) multiplied by 128.

When the SYNC event is detected, the actual value of the frequency error counter and its counting direction are stored in the FECAP (frequency error capture) field and in the FEDIR (frequency error direction) bit of the CRS_ISR register. When the SYNC event is detected during the downcounting phase (before reaching the zero value), it means that the actual frequency is lower than the target (and so, that the TRIM value should be incremented), while when it is detected during the upcounting phase it means that the actual frequency is higher (and that the TRIM value should be decremented).

Figure 15. CRS counter behavior



7.6 CRS registers

Refer to *Section 1.1 on page 42* of the reference manual for a list of abbreviations used in register descriptions.

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

7.6.1 CRS control register (CRS_CR)

Address offset: 0x00

Reset value: 0x0000 2000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	TRIM[5:0]						SWSYNC	AUTOTRIMEN	CEN	Res.	ESYNCIE	ERRIE	SYNCWARNIE	SYNCOKIE
		rw	rw	rw	rw	rw	rw	rt_w	rw	rw		rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **TRIM[5:0]**: HSI48 oscillator smooth trimming

These bits provide a user-programmable trimming value to the HSI48 oscillator. They can be programmed to adjust to variations in voltage and temperature that influence the frequency of the HSI48.

The default value is 32, which corresponds to the middle of the trimming interval. The trimming step is around 67 kHz between two consecutive TRIM steps. A higher TRIM value corresponds to a higher output frequency.

When the AUTOTRIMEN bit is set, this field is controlled by hardware and is read-only.

Bit 7 **SWSYNC**: Generate software SYNC event

This bit is set by software in order to generate a software SYNC event. It is automatically cleared by hardware.

0: No action

1: A software SYNC event is generated.

Bit 6 **AUTOTRIMEN**: Automatic trimming enable

This bit enables the automatic hardware adjustment of TRIM bits according to the measured frequency error between two SYNC events. If this bit is set, the TRIM bits are read-only. The TRIM value can be adjusted by hardware by one or two steps at a time, depending on the measured frequency error value. Refer to *Section 7.3.4: Frequency error evaluation and automatic trimming* for more details.

0: Automatic trimming disabled, TRIM bits can be adjusted by the user.

1: Automatic trimming enabled, TRIM bits are read-only and under hardware control.

Bit 5 **CEN**: Frequency error counter enable

This bit enables the oscillator clock for the frequency error counter.

0: Frequency error counter disabled

1: Frequency error counter enabled

When this bit is set, the CRS_CFGR register is write-protected and cannot be modified.

Bit 4 Reserved, must be kept at reset value.

7.6.5 CRS register map

Table 22. CRS register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**CRS_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]						SWSYNC	AUTOTRIMEN	CEN	Res.	ESYNCIE	ERRIE	SYNCWARNIE	SYNCOKIE
	Reset value																			1	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**CRS_CFGR**	SYNCPOL	Res.	SYNC SRC [1:0]		Res.	SYNC DIV [2:0]			FELIM[7:0]								RELOAD[15:0]															
	Reset value	0		1	0		0	0	0	0	0	1	0	0	0	1	0	1	0	1	1	1	0	1	1	0	1	1	1	1	1	1	1
0x08	**CRS_ISR**	FECAP[15:0]																FEDIR	Res.	Res.	Res.	Res.	TRIMOVF	SYNCMISS	SYNCERR	Res.	Res.	Res.	Res.	ESYNCF	ERRF	SYNCWARNF	SYNCOKF
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0					0	0	0	0
0x0C	**CRS_ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESYNCC	ERRC	SYNCWARNC	SYNCOKC
	Reset value																													0	0	0	0

Refer to *Section 2.2.2 on page 46* for the register boundary addresses.

Bits 31:1 Reserved (read as ‘0’)

Bit 0 **DMA1_CH1**: DMA1 channel 1 interrupt request pending

9.1.17 SYSCFG interrupt line 10 status register (SYSCFG_ITLINE10)

Address offset: A8h

System reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res..	Res..	Res..	Res..	Res..	Res..	Res..	Res..	Res..	Res..	Res..	Res..	Res..	Res..	Res..	Res..
15	14	13	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..	DMA2_CH2	DMA2_CH1	DMA1_CH3	DMA1_CH2
												r	r	r	r

Bits 31:4 Reserved (read as ‘0’)

Bit 3 **DMA2_CH2**: DMA2 channel 2 interrupt request pending

Bit 2 **DMA2_CH1**: DMA2 channel 1 interrupt request pending

Bit 1 **DMA1_CH3**: DMA1 channel 3 interrupt request pending

Bit 0 **DMA1_CH2**: DMA1 channel 2 interrupt request pending

9.1.18 SYSCFG interrupt line 11 status register (SYSCFG_ITLINE11)

Address offset: ACh

System reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res..	Res..	Res..	Res..	Res..	Res..	Res..	Res..	Res..	Res..	Res..	Res..	Res..	Res..	Res..	Res..
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res..	Res..	Res..	Res..	Res..	Res..	Res..	Res..	Res..	DMA2_CH5	DMA2_CH4	DMA2_CH3	DMA1_CH7	DMA1_CH6	DMA1_CH5	DMA1_CH4
									r	r	r	r	r	r	r

Bits 31:7 Reserved (read as ‘0’)

Bit 6 **DMA2_CH5**: DMA2 channel 5 interrupt request pending

Bit 5 **DMA2_CH4**: DMA2 channel 4 interrupt request pending

Bit 4 **DMA2_CH3**: DMA2 channel 3 interrupt request pending

Bit 3 **DMA1_CH7**: DMA1 channel 7 interrupt request pending

Bit 2 **DMA1_CH6**: DMA1 channel 6 interrupt request pending

Bit 1 **DMA1_CH5**: DMA1 channel 5 interrupt request pending

Bit 0 **DMA1_CH4**: DMA1 channel 4 interrupt request pending

10.4.6 DMA channel x memory address register (DMA_CMARx and DMA2_CMARx) (x = 1..7 for DMA and x = 1..5 for DMA2, 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
MA [31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA [15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MA[31:0]:** Memory address

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.

Table 64. TIM1 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	**TIM1_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**TIM1_CNT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIM1_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIM1_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	**TIM1_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	**TIM1_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIM1_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**TIM1_CCR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	**TIM1_CCR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**TIM1_BDTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**TIM1_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	**TIM1_DMAR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.2.2 on page 46* for the register boundary addresses.

Bits 31:16 **CCR2[31:16]**: High Capture/Compare 2 value (on TIM2).

Bits 15:0 **CCR2[15:0]**: Low Capture/Compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC2 output.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (IC2).

18.4.15 TIM2 and TIM3 capture/compare register 3 (TIM2_CCR3 and TIM3_CCR3)

Address offset: 0x3C

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3[31:16] (TIM2 only)															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR3[31:16]**: High Capture/Compare 3 value (on TIM2).

Bits 15:0 **CCR3[15:0]**: Low Capture/Compare 3 value

If channel CC3 is configured as output:

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC3 output.

If channel CC3is configured as input:

CCR3 is the counter value transferred by the last input capture 3 event (IC3).

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

20.6.2 TIM16 and TIM17 control register 2 (TIM16_CR2 and TIM17_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
						rw	rw					rw	rw		rw

Bits 15:10 Reserved, always read as 0.

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0
1: OC1N=1 after a dead-time when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0
1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

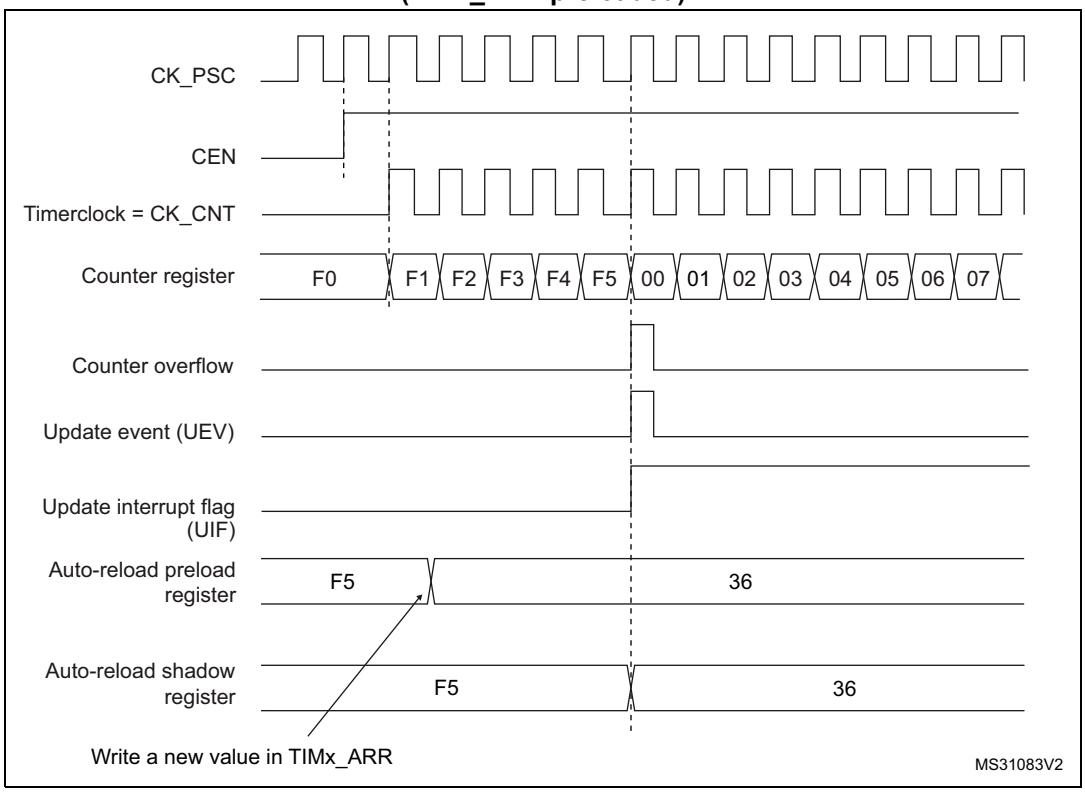
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bits 7:4 Reserved, always read as 0.

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

Figure 205. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



24.4.4 WWDG register map

The following table gives the WWDG register map and reset values.

Table 78. 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**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
	Reset value																									0	1	1	1	1	1	1	1
0x04	**WWDG_CFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWI	WDGTB1	WDGTB0	W[6:0]						
	Reset value																							0	0	0	1	1	1	1	1	1	1
0x08	**WWDG_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
	Reset value																																0

Refer to *Section 2.2.2: Memory map and register boundary addresses* for the register boundary addresses.

Bits 14:13 **TAMPPRCH[1:0]**: RTC_TAMPx 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 RTC_TAMPx inputs.

0x0: 1 RTCCLK cycle
0x1: 2 RTCCLK cycles
0x2: 4 RTCCLK cycles
0x3: 8 RTCCLK cycles

Bits 12:11 **TAMPFLT[1:0]**: RTC_TAMPx filter count

These bits determines the number of consecutive samples at the specified level (TAMP*TRG) needed to activate a Tamper event. TAMPFLT is valid for each of the RTC_TAMPx inputs.

0x0: Tamper event is activated on edge of RTC_TAMPx input transitions to the active level (no internal pull-up on RTC_TAMPx input).
0x1: Tamper event is activated after 2 consecutive samples at the active level.
0x2: Tamper event is activated after 4 consecutive samples at the active level.
0x3: Tamper event is activated after 8 consecutive samples at the active level.

Bits 10:8 **TAMPFREQ[2:0]**: Tamper sampling frequency

Determines the frequency at which each of the RTC_TAMPx 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)

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.

Bit 6 **TAMP3TRG**: Active level for RTC_TAMP3 input

if TAMPFLT != 00:
0: RTC_TAMP3 input staying low triggers a tamper detection event.
1: RTC_TAMP3 input staying high triggers a tamper detection event.
if TAMPFLT = 00:
0: RTC_TAMP3 input rising edge triggers a tamper detection event.
1: RTC_TAMP3input falling edge triggers a tamper detection event.

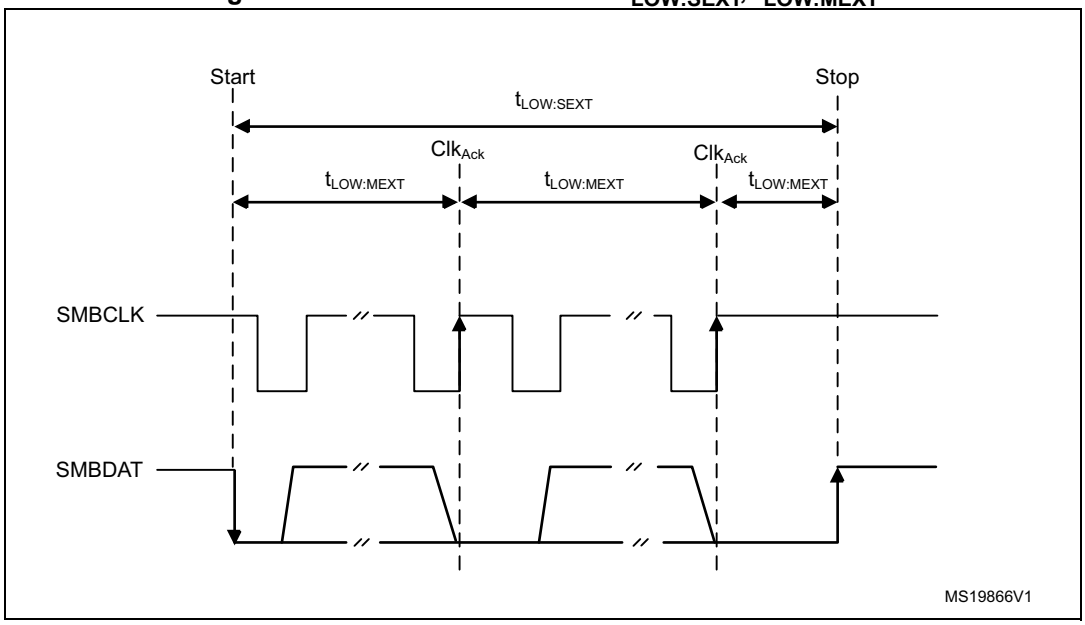
Bit 5 **TAMP3E**: RTC_TAMP3 detection enable

0: RTC_TAMP3 input detection disabled
1: RTC_TAMP3 input detection enabled

Bit 4 **TAMP2TRG**: Active level for RTC_TAMP2 input

if TAMPFLT != 00:
0: RTC_TAMP2 input staying low triggers a tamper detection event.
1: RTC_TAMP2 input staying high triggers a tamper detection event.
if TAMPFLT = 00:
0: RTC_TAMP2 input rising edge triggers a tamper detection event.
1: RTC_TAMP2 input falling edge triggers a tamper detection event.

Figure 237. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$



Bus idle detection

A master can assume that the bus is free if it detects that the clock and data signals have been high for t_{IDLE} greater than $t_{HIGH,MAX}$. (refer to *Table 90: I2C-SMBUS specification clock timings*)

This timing parameter covers the condition where a master has been dynamically added to the bus and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the master must wait long enough to ensure that a transfer is not currently in progress. The peripheral supports a hardware bus idle detection.

26.4.12 SMBus initialization

This section is relevant only when SMBus feature is supported. Please refer to *Section 26.3: I2C implementation*.

In addition to I2C initialization, some other specific initialization must be done in order to perform SMBus communication:

Received Command and Data Acknowledge control (Slave mode)

A SMBus receiver must be able to NACK each received command or data. In order to allow ACK control in slave mode, the Slave Byte Control mode must be enabled by setting the SBC bit in the I2C_CR1 register. Refer to *Slave Byte Control mode on page 631* for more details.

When a timeout violation is detected in master mode, a STOP condition is automatically sent.

When a timeout violation is detected in slave mode, SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

Alert (ALERT)

This section is relevant only when the SMBus feature is supported. Please refer to *Section 26.3: I2C implementation*.

The ALERT flag is set when the I2C interface is configured as a Host (SMBHEN=1), the alert pin detection is enabled (ALERTEN=1) and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit is set in the I2C_CR1 register.

26.4.17 DMA requests

Transmission using DMA

DMA (Direct Memory Access) can be enabled for transmission by setting the TXDMAEN bit in the I2C_CR1 register. Data is loaded from an SRAM area configured using the DMA peripheral (see *Section 10: Direct memory access controller (DMA) on page 188*) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

- In master mode: the initialization, the slave address, direction, number of bytes and START bit are programmed by software (the transmitted slave address cannot be transferred with DMA). When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to *Master transmitter on page 642*.

For code example refer to the Appendix section *A.14.8: I2C configured in master mode to transmit with DMA code example*.

- In slave mode:
  - With NOSTRETCH=0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
  - With NOSTRETCH=1, the DMA must be initialized before the address match event.
- For instances supporting SMBus: the PEC transfer is managed with NBYTES counter. Refer to *SMBus Slave transmitter on page 657* and *SMBus Master transmitter on page 661*.

Note: If DMA is used for transmission, the TXIE bit does not need to be enabled.

Reception using DMA

DMA (Direct Memory Access) can be enabled for reception by setting the RXDMAEN bit in the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured using the DMA peripheral (refer to *Section 10: Direct memory access controller (DMA) on page 188*) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

- In master mode, the initialization, the slave address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, the DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter..
- In slave mode with NOSTRETCH=0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.
- If SMBus is supported (see *Section 26.3: I2C implementation*): the PEC transfer is managed with the NBYTES counter. Refer to *SMBus Slave receiver on page 659* and *SMBus Master receiver on page 663*.

Note: If DMA is used for reception, the RXIE bit does not need to be enabled.

For code example refer to the Appendix section *A.14.9: I2C configured in slave mode to receive with DMA code example*.

26.4.18 Debug mode

When the microcontroller enters debug mode (core halted), the SMBus timeout either continues to work normally or stops, depending on the DBG_I2Cx_SMBUS_TIMEOUT configuration bits in the DBG module.

26.5 I2C low-power modes

Table 99. low-power modes

Mode	Description
Sleep	No effect I2C interrupts cause the device to exit the Sleep mode.
Stop	The contents of I2C registers are kept.
Standby	The I2C peripheral is powered down and must be reinitialized after exiting Standby.

26.6 I2C interrupts

The table below gives the list of I2C interrupt requests.

Figure 250. Data sampling when oversampling by 16

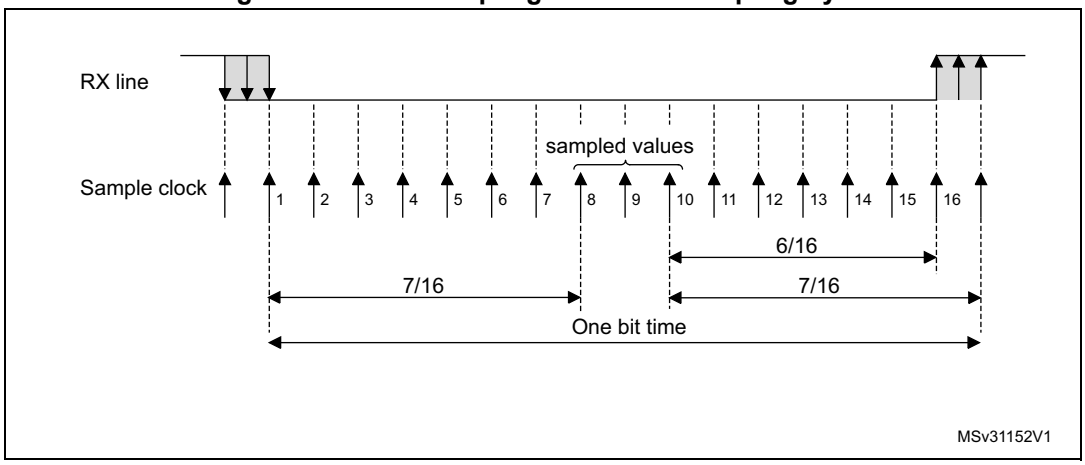


Figure 251. Data sampling when oversampling by 8

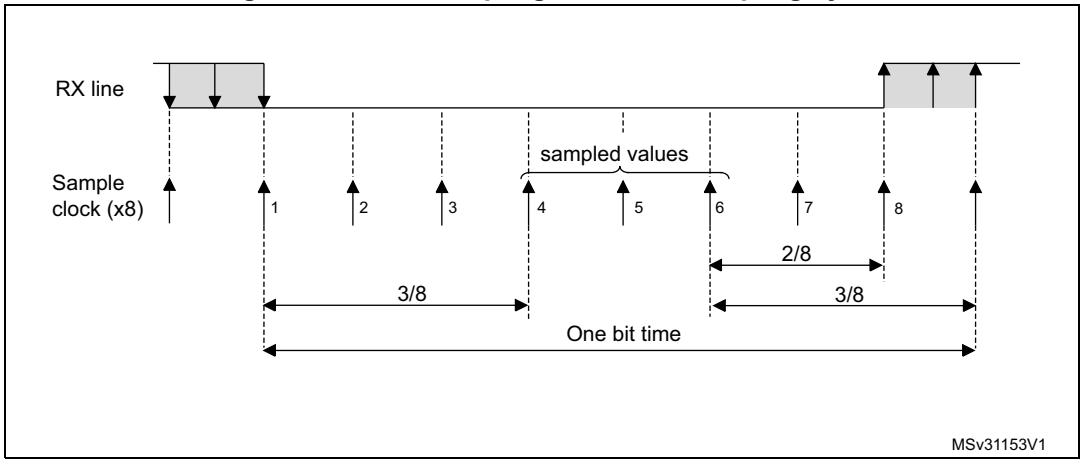


Table 103. Noise detection from sampled data

Sampled value	NE status	Received bit value
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

Block mode (T=1)

In T=1 (block) mode, the parity error transmission is deactivated, by clearing the NACK bit in the UART_CR3 register.

When requesting a read from the smartcard, in block mode, the software must enable the receiver Timeout feature by setting the RTOEN bit in the USART_CR2 register and program the RTO bits field in the RTOR register to the BWT (block wait time) - 11 value. If no answer is received from the card before the expiration of this period, the RTOF flag will be set and a timeout interrupt will be generated (if RTOIE bit in the USART_CR1 register is set). If the first character is received before the expiration of the period, it is signaled by the RXNE interrupt.

Note: The RXNE interrupt must be enabled even when using the USART in DMA mode to read from the smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.

After the reception of the first character (RXNE interrupt), the RTO bit fields in the RTOR register must be programmed to the CWT (character wait time) - 11 value, in order to allow the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baudtime units. If the smartcard does not send a new character in less than the CWT period after the end of the previous character, the USART signals this to the software through the RTOF flag and interrupt (when RTOIE bit is set).

Note: The RTO counter starts counting:

- From the end of the stop bit in case STOP = 00.

- From the end of the second stop bit in case of STOP = 10.

- 1 bit duration after the beginning of the STOP bit in case STOP = 11.

- From the beginning of the STOP bit in case STOP = 01.

As in the Smartcard protocol definition, the BWT/CWT values are defined from the beginning (start bit) of the last character. The RTO register must be programmed to BWT - 11 or CWT -11, respectively, taking into account the length of the last character itself.

A block length counter is used to count all the characters received by the USART. This counter is reset when the USART is transmitting (TXE=0). The length of the block is communicated by the smartcard in the third byte of the block (prologue field). This value must be programmed to the BLEN field in the USART_RTOR register. when using DMA mode, before the start of the block, this register field must be programmed to the minimum value (0x0). with this value, an interrupt is generated after the 4th received character. The software must read the LEN field (third byte), its value must be read from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BLEN value. However, before the start of the block, the maximum value of BLEN (0xFF) may be programmed. The real value will be programmed after the reception of the third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the BLEN=LEN. If the block is using the CRC mechanism (2 epilogue bytes), BLEN=LEN+1 must be programmed. The total block length (including prologue, epilogue and information fields) equals BLEN+4. The end of the block is signaled to the software through the EOBF flag and interrupt (when EOBIE bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character wait Time overflow).

CAN mailbox data low register (CAN_TDLxR) (x = 0..2)

All bits of this register are write protected when the mailbox is not in empty state.

Address offsets: 0x188, 0x198, 0x1A8
Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA1[7:0]								DATA0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **DATA3[7:0]**: Data byte 3
Data byte 3 of the message.

Bits 23:16 **DATA2[7:0]**: Data byte 2
Data byte 2 of the message.

Bits 15:8 **DATA1[7:0]**: Data byte 1
Data byte 1 of the message.

Bits 7:0 **DATA0[7:0]**: Data byte 0
Data byte 0 of the message.
A message can contain from 0 to 8 data bytes and starts with byte 0.

CAN mailbox data high register (CAN_TDHxR) (x = 0..2)

All bits of this register are write protected when the mailbox is not in empty state.

Address offsets: 0x18C, 0x19C, 0x1AC
Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA5[7:0]								DATA4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

CAN receive FIFO mailbox data length control and time stamp register (CAN_RDTxR) (x = 0..1)

Address offsets: 0x1B4, 0x1C4
Reset value: 0xXXXX XXXX

All RX registers are write protected.

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

Bits 31:16 **TIME[15:0]**: Message time stamp
This field contains the 16-bit timer value captured at the SOF detection.

Bits 15:8 **FMI[7:0]**: Filter match index
This register contains the index of the filter the message stored in the mailbox passed through. For more details on identifier filtering please refer to *Section 29.7.4: Identifier filtering on page 823* - **Filter Match Index** paragraph.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **DLC[3:0]**: Data length code
This field defines the number of data bytes a data frame contains (0 to 8). It is 0 in the case of a remote frame request.

A.9.7 Output compare configuration code example

```
/* (1) Set prescaler to 3, so APBCLK/4 i.e 12MHz */
/* (2) Set ARR = 12000 -1 */
/* (3) Set CCRx = ARR, as timer clock is 12MHz, an event occurs each 1 ms */
/* (4) Select toggle mode on OC1 (OC1M = 011),
       disable preload register on OC1 (OC1PE = 0, reset value) */
/* (5) Select active high polarity on OC1 (CC1P = 0, reset value),
       enable the output on OC1 (CC1E = 1)*/
/* (6) Enable output (MOE = 1)*/
/* (7) Enable counter */
TIMx->PSC |= 3; /* (1) */
TIMx->ARR = 12000 - 1; /* (2) */
TIMx->CCR1 = 12000 - 1; /* (3) */
TIMx->CCMR1 |= TIM_CCMR1_OC1M_0 | TIM_CCMR1_OC1M_1; /* (4) */
TIMx->CCER |= TIM_CCER_CC1E; /* (5)*/
TIMx->BDTR |= TIM_BDTR_MOE; /* (6) */
TIMx->CR1 |= TIM_CR1_CEN; /* (7) */
```

A.9.8 Edge-aligned PWM configuration example

```
/* (1) Set prescaler to 47, so APBCLK/48 i.e 1MHz */
/* (2) Set ARR = 8, as timer clock is 1MHz the period is 9 us */
/* (3) Set CCRx = 4, , the signal will be high during 4 us */
/* (4) Select PWM mode 1 on OC1 (OC1M = 110),
       enable preload register on OC1 (OC1PE = 1) */
/* (5) Select active high polarity on OC1 (CC1P = 0, reset value),
       enable the output on OC1 (CC1E = 1)*/
/* (6) Enable output (MOE = 1)*/
/* (7) Enable counter (CEN = 1)
       select edge aligned mode (CMS = 00, reset value)
       select direction as upcounter (DIR = 0, reset value) */
/* (8) Force update generation (UG = 1) */
TIMx->PSC = 47; /* (1) */
TIMx->ARR = 8; /* (2) */
TIMx->CCR1 = 4; /* (3) */
TIMx->CCMR1 |= TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1
             | TIM_CCMR1_OC1PE; /* (4) */
TIMx->CCER |= TIM_CCER_CC1E; /* (5) */
TIMx->BDTR |= TIM_BDTR_MOE; /* (6) */
TIMx->CR1 |= TIM_CR1_CEN; /* (7) */
TIMx->EGR |= TIM_EGR_UG; /* (8) */
```