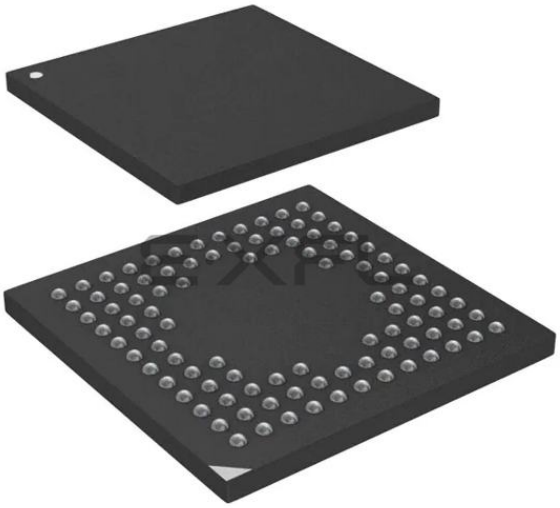


Welcome to E-XFL.COM

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	83
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	100-UFBGA
Supplier Device Package	100-UFBGA (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/stm32l433vci6

5.1.4 Battery backup domain

To retain the content of the Backup registers and supply the RTC function when V_{DD} is turned off, the VBAT pin can be connected to an optional backup voltage supplied by a battery or by another source.

VBAT pin is not available on low pin-count packages, V_{BAT} is internally connected to V_{DD}.

The VBAT pin powers the RTC unit, the LSE oscillator and the PC13 to PC15 I/Os, allowing the RTC to operate even when the main power supply is turned off. The switch to the V_{BAT} supply is controlled by the power-down reset embedded in the Reset block.

Warning: **During $t_{RSTTEMPO}$ (temporization at V_{DD} startup) or after a PDR has been detected, the power switch between V_{BAT} and V_{DD} remains connected to V_{BAT}.**
During the startup phase, if V_{DD} is established in less than $t_{RSTTEMPO}$ (refer to the datasheet for the value of $t_{RSTTEMPO}$) and $V_{DD} > V_{BAT} + 0.6$ V, a current may be injected into V_{BAT} through an internal diode connected between V_{DD} and the power switch (V_{BAT}).
If the power supply/battery connected to the VBAT pin cannot support this current injection, it is strongly recommended to connect an external low-drop diode between this power supply and the VBAT pin.

If no external battery is used in the application, it is recommended to connect V_{BAT} externally to V_{DD} with a 100 nF external ceramic decoupling capacitor.

When the backup domain is supplied by V_{DD} (analog switch connected to V_{DD}), the following pins are available:

- PC13, PC14 and PC15, which can be used as GPIO pins
- PC13, PC14 and PC15, which can be configured by RTC or LSE (refer to *Section 34.3: RTC functional description on page 969*)
- PA0/RTC_TAMP2 and PE6/RTC_TAMP3 when they are configured by the RTC as tamper pins

Note: Due to the fact that the analog switch can transfer only a limited amount of current (3 mA), the use of GPIO PC13 to PC15 in output mode is restricted: the speed has to be limited to 2 MHz with a maximum load of 30 pF and these I/Os must not be used as a current source (e.g. to drive a LED).

When the backup domain is supplied by V_{BAT} (analog switch connected to V_{BAT} because V_{DD} is not present), the following functions are available:

- PC13, PC14 and PC15 can be controlled only by RTC or LSE (refer to *Section 34.3: RTC functional description*)
- PA0/RTC_TAMP2 and PE6/RTC_TAMP3 when they are configured by the RTC as tamper pins

input events) for both regular and injected conversions

- Conversion modes
  - The ADC can convert a single channel or can scan a sequence of channels
  - Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode
- Interrupt generation at ADC ready, the end of sampling, the end of conversion (regular or injected), end of sequence conversion (regular or injected), analog watchdog 1, 2 or 3 or overrun events
- 3 analog watchdogs
- ADC supply requirements: 1.62 to 3.6 V
- ADC input range: $V_{REF-} \leq V_{IN} \leq V_{REF+}$

Figure 39 shows the block diagram of one ADC.

16.3 ADC implementation

Table 53. Main ADC features

References	ADC1
Dual mode	-
DFSDM interface(1)	X
SMPPLUS control	-

1. Available only on STM32L451xx/452xx/462xx.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HT1[11:0]											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LT1[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **HT1[11:0]**: Analog watchdog 1 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 1.

Refer to *Section 16.4.29: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

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

Bits 11:0 **LT1[11:0]**: Analog watchdog 1 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 1.

Refer to *Section 16.4.29: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Note: Software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

16.6.9 ADC watchdog threshold register 2 (ADC_TR2)

Address offset: 0x24

Reset value: 0x00FF 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.	HT2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LT2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

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

19.6 COMP registers

19.6.1 Comparator 1 control and status register (COMP1_CSR)

The COMP1_CSR is the Comparator 1 control/status register. It contains all the bits /flags related to comparator1.

Address offset: 0x00

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	VALUE	Res.	Res.	Res.	INMESEL		Res.	SCAL EN	BRG EN	Res.	BLANKING			HYST	
rs	r				rw			rw	rw		rw			rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POLA RITY	Res.	Res.	Res.	Res.	Res.	Res.	INP SEL.	INMSEL				PWRMODE		Res.	EN
rw							rw	rw				rw			rw

Bit 31 **LOCK:** COMP1_CSR register lock bit

This bit is set by software and cleared by a hardware system reset. It locks the whole content of the comparator 1 control register, COMP1_CSR[31:0].

0: COMP1_CSR[31:0] for comparator 1 are read/write

1: COMP1_CSR[31:0] for comparator 1 are read-only

Bit 30 **VALUE:** Comparator 1 output status bit

This bit is read-only. It reflects the current comparator 1 output taking into account POLARITY bit effect.

Bits 29:27 Reserved, must be kept at reset value.

Bits 26:25 **INMESEL:** comparator 1 input minus extended selection bits.

These bits are set and cleared by software (only if LOCK is not set). They select which extended GPIO input is connected to the input minus of comparator if INMSEL = 111.

00: PC4

01: PA0

10: PA4

11: PA5

Bit 24 Reserved, must be kept at reset value.

Bit 23 **SCALEN**: Voltage scaler enable bit

This bit is set and cleared by software. This bit enable the outputs of the V_{REFINT} divider available on the minus input of the Comparator 1.

0: Bandgap scaler disable (if SCALEN bit of COMP2_CSR register is also reset)

1: Bandgap scaler enable

Bits 6:4 **INMSEL**: Comparator 2 input minus selection bits

These bits are set and cleared by software (only if LOCK not set). They select which input is connected to the input minus of comparator 2.

000 = 1/4 V_{REFINT}
001 = 1/2 V_{REFINT}
010 = 3/4 V_{REFINT}
011 = V_{REFINT}
100 = DAC Channel1
101 = DAC Channel2
110 = PB3
111: GPIOx selected by INMESEL bits

Bits 3:2 **PWRMODE[1:0]:** Power Mode of the comparator 2

These bits are set and cleared by software (only if LOCK not set). They control the power/speed of the Comparator 2.

00: High speed
01 or 10: Medium speed
11: Ultra low power

Bit 1 Reserved, must be kept cleared.

Bit 0 **EN:** Comparator 2 enable bit

This bit is set and cleared by software (only if LOCK not set). It switches oncomparator2.

0: Comparator 2 switched OFF
1: Comparator 2 switched ON

23.6.11 TSC register map

Table 110. TSC 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
0x0000	**TSC_CR**	CTPH[3:0]				CTPL[3:0]				SSD[6:0]							SSE	SSPSC	PGPSC[2:0]			Res.	Res.	Res.	Res.	MCV [2:0]			IODEF	SYNCPOL	AM	START	TSCE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0
0x0004	**TSC_IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCEIE	EOAIE
	Reset value																															0	0
0x0008	**TSC_ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCEIC	EOAIC
	Reset value																															0	0
0x000C	**TSC_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCEF	EOAF
	Reset value																															0	0
0x0010	**TSC_IOHCR**	Res.	Res.	Res.	Res.	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0014	Reserved																																
0x0018	**TSC_IOASCR**	Res.	Res.	Res.	Res.	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x001C	Reserved																																
0x0020	**TSC_IOSCR**	Res.	Res.	Res.	Res.	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0024	Reserved																																
0x0028	**TSC_IOCCR**	Res.	Res.	Res.	Res.	G7_IO4	G7_IO3	G7_IO2	G7_IO1	G6_IO4	G6_IO3	G6_IO2	G6_IO1	G5_IO4	G5_IO3	G5_IO2	G5_IO1	G4_IO4	G4_IO3	G4_IO2	G4_IO1	G3_IO4	G3_IO3	G3_IO2	G3_IO1	G2_IO4	G2_IO3	G2_IO2	G2_IO1	G1_IO4	G1_IO3	G1_IO2	G1_IO1
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	Reserved																																
0x0030	**TSC_IOGCSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G7S	G6S	G5S	G4S	G3S	G2S	G1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	G7E	G6E	G5E	G4E	G3E	G2E	G1E
	Reset value										0	0	0	0	0	0	0										0	0	0	0	0	0	0
0x0034	**TSC_IOG1CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0038	**TSC_IOG2CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 112. RNG interrupt requests

Interrupt event	Event flag	Enable control bit
Data ready flag	DRDY	IE
Seed error flag	SEIS	IE
Clock error flag	CEIS	IE

The user can enable or disable the above interrupt sources individually by changing the mask bits or the general interrupt control bit IE in the RNG_CR register. The status of the individual interrupt sources can be read from the RNG_SR register.

Note: Interrupts are generated only when RNG is enabled.

24.6 RNG processing time

The RNG can produce one 32-bit random numbers every 42 RNG clock cycles.

After enabling or re-enabling the RNG using the RNGEN bit it takes 46 RNG clock cycles before random data are available.

24.7 Entropy source validation

24.7.1 Introduction

In order to assess of the amount of entropy available from the RNG, STMicroelectronics has tested the RNG against AIS-31 PTG.2 set of tests. The results can be provided on demand or the customer can reproduce the measurements using the AIS reference software. The customer could also test the RNG against an older NIST SP800-22 set of tests.

24.7.2 Validation conditions

STMicroelectronics has validated the RNG true random number generator in the following conditions:

- RNG clock rng_clk= 48 MHz
- AHB clock rng_hclk= 60 MHz

24.7.3 Data collection

If raw data needs to be read instead of pre-processed data the developer is invited to contact STMicroelectronics to receive the correct procedure to follow.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYR731:16]															
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
KEYR7[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **KEYR7[31:0]**: Data output register (MSB key [255:224])
Same description as AES_KEYR0 for the key[255:224].

Note: The key registers from 4 to 7 are used only when 256-bit key length is selected. These registers have no effect when 128-bit key length is selected (only key registers from 0 to 3 are used).

29 Basic timers (TIM6/TIM7)

Timer TIM7 is available on STM32L43xxx and STM32L44xxx devices only.

29.1 TIM6/TIM7 introduction

The basic timers TIM6 and TIM7 consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used as generic timers for time-base generation but they are also specifically used to drive the digital-to-analog converter (DAC). In fact, the timers are internally connected to the DAC and are able to drive it through their trigger outputs.

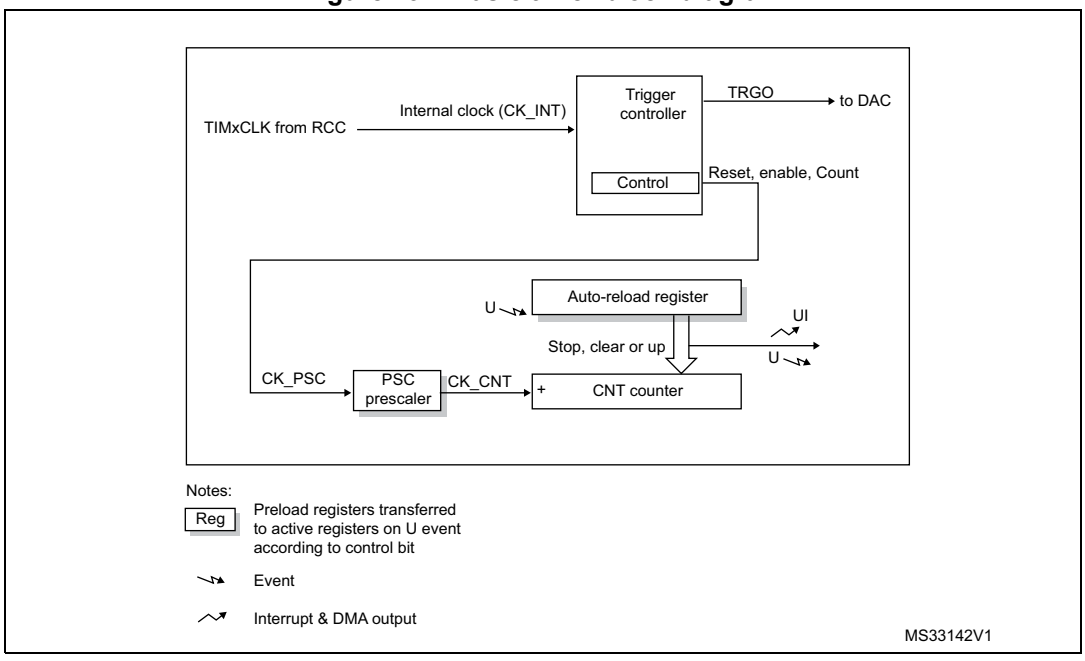
The timers are completely independent, and do not share any resources.

29.2 TIM6/TIM7 main features

Basic timer (TIM6/TIM7) features include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Synchronization circuit to trigger the DAC
- Interrupt/DMA generation on the update event: counter overflow

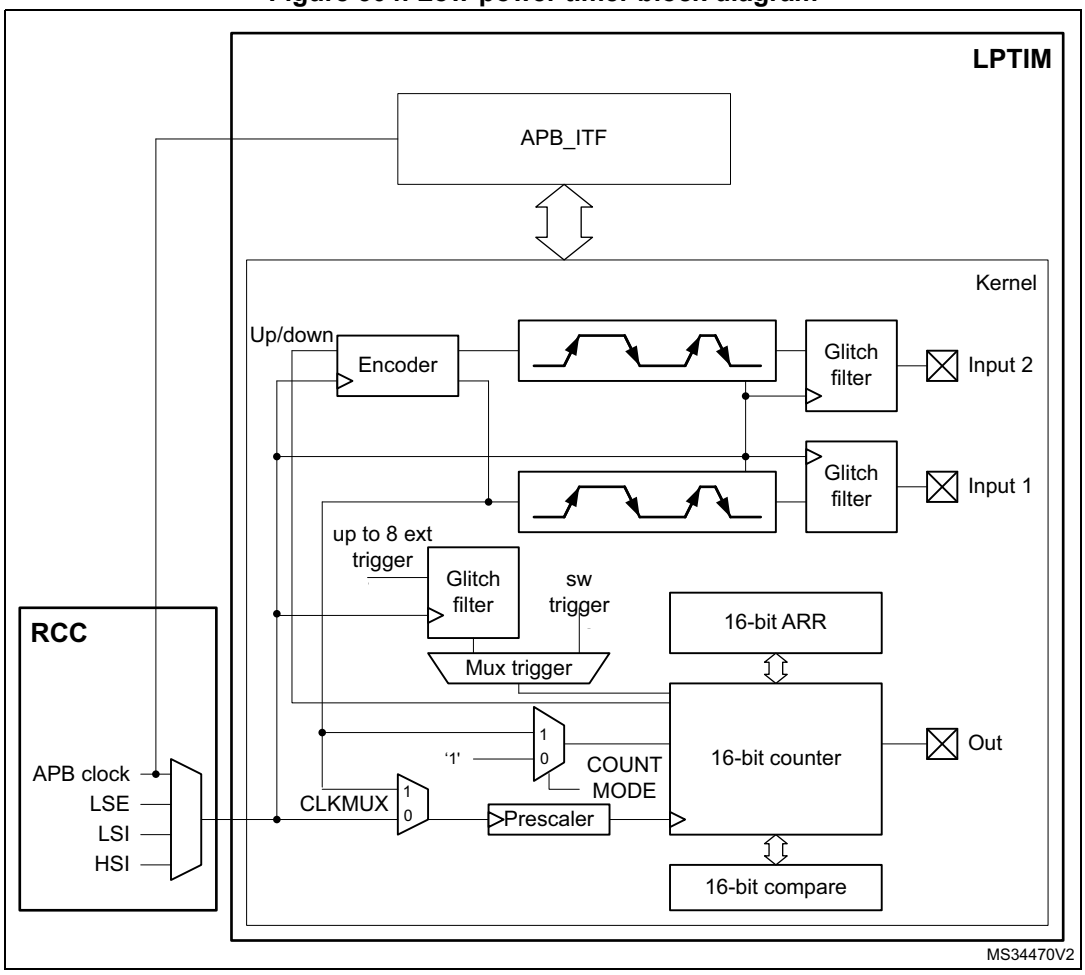
Figure 294. Basic timer block diagram



30.4 LPTIM functional description

30.4.1 LPTIM block diagram

Figure 304. Low-power timer block diagram



30.4.2 LPTIM reset and clocks

The LPTIM can be clocked using several clock sources. It can be clocked using an internal clock signal which can be chosen among APB, LSI, LSE or HSI16 sources through the Reset and Clock controller (RCC). Also, the LPTIM can be clocked using an external clock signal injected on its external Input1. When clocked with an external clock source, the LPTIM may run in one of these two possible configurations:

- The first configuration is when the LPTIM is clocked by an external signal but in the same time an internal clock signal is provided to the LPTIM either from APB or any other embedded oscillator including LSE, LSI and HSI16.
- The second configuration is when the LPTIM is solely clocked by an external clock source through its external Input1. This configuration is the one used to realize Timeout

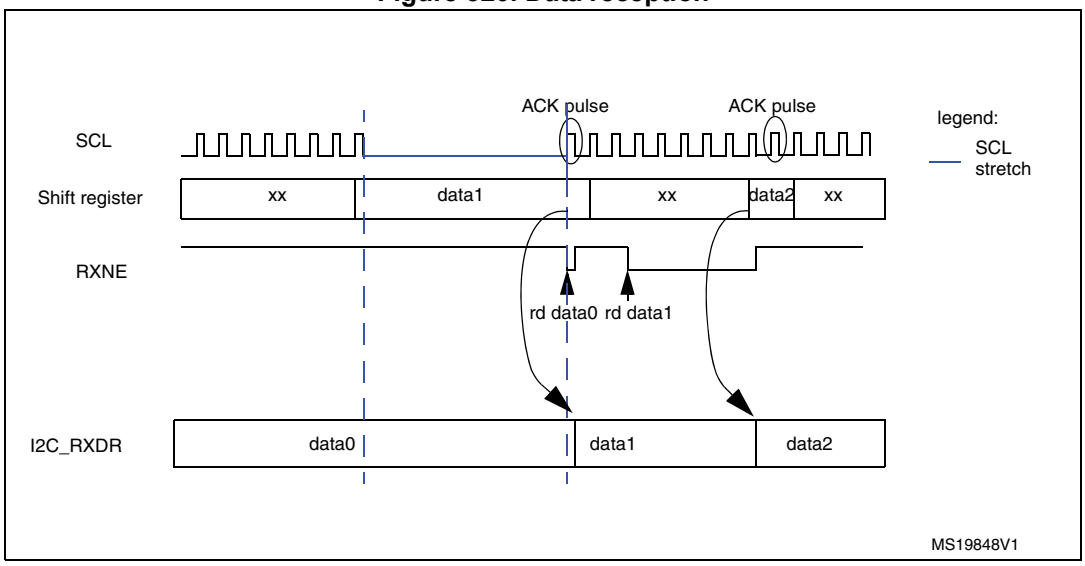
35.4.6 Data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

Reception

The SDA input fills the shift register. After the 8th SCL pulse (when the complete data byte is received), the shift register is copied into I2C_RXDR register if it is empty (RXNE=0). If RXNE=1, meaning that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch is inserted between the 8th and 9th SCL pulse (before the Acknowledge pulse).

Figure 320. Data reception



36.5.9 USART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART_CR1 register. Depending on the frame length defined by the M bits, the possible USART frame formats are as listed in *Table 169*.

Table 169. Frame formats

M bits	PCE bit	USART frame(1)
00	0	| SB | 8-bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data | PB | STB |
10	0	| SB | 7-bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (9th, 8th or 7th, depending on the M bits value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 6, 7 or 8 LSB bits (depending on M bits values) and the parity bit.

As an example, if data=00110101, and 4 bits are set, then the parity bit will be 0 if even parity is selected (PS bit in USART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bits values) and the parity bit.

As an example, if data=00110101 and 4 bits set, then the parity bit will be 1 if odd parity is selected (PS bit in USART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the USART_ISR register and an interrupt is generated if PEIE is set in the USART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART_ICR register.

Parity generation in transmission

If the PCE bit is set in USART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS=0) or an odd number of “1s” if odd parity is selected (PS=1)).

USART is not requesting it. The LSE clock is not OFF but there is a clock gating to avoid useless consumption.

When the USART clock source is configured to be f_{LSE} or f_{HSI}, it is possible to keep enabled this clock during STOP mode by setting the UCESM bit in USART_CR3 control register.

The MCU wakeup from Stop mode can be done using the standard RXNE interrupt. In this case, the RXNEIE bit must be set before entering Stop mode.

Alternatively, a specific interrupt may be selected through the WUS bit fields.

In order to be able to wake up the MCU from Stop mode, the UESM bit in the USART_CR1 control register must be set prior to entering Stop mode.

When the wakeup event is detected, the WUF flag is set by hardware and a wakeup interrupt is generated if the WUFIE bit is set.

Note: Before entering Stop mode, the user must ensure that the USART is not performing a transfer. BUSY flag cannot ensure that Stop mode is never entered during a running reception.

The WUF flag is set when a wakeup event is detected, independently of whether the MCU is in Stop or in an active mode.

When entering Stop mode just after having initialized and enabled the receiver, the REACK bit must be checked to ensure the USART is actually enabled.

When DMA is used for reception, it must be disabled before entering Stop mode and re-enabled upon exit from Stop mode.

The wakeup from Stop mode feature is not available for all modes. For example it doesn't work in SPI mode because the SPI operates in master mode only.

Using Mute mode with Stop mode

If the USART is put into Mute mode before entering Stop mode:

- Wakeup from Mute mode on idle detection must not be used, because idle detection cannot work in Stop mode.
- If the wakeup from Mute mode on address match is used, then the source of wake-up from Stop mode must also be the address match. If the RXNE flag is set when entering the Stop mode, the interface will remain in mute mode upon address match and wake up from Stop.
- If the USART is configured to wake up the MCU from Stop mode on START bit detection, the WUF flag is set, but the RXNE flag is not set.

Determining the maximum USART baud rate allowing to wakeup correctly from Stop mode when the USART clock source is the HSI clock

The maximum baud rate allowing to wakeup correctly from stop mode depends on:

- the parameter $t_{WUUSART}$ provided in the device datasheet
- the USART receiver tolerance provided in the *Section 36.5.5: Tolerance of the USART receiver to clock deviation*.

Let us take this example: OVER8 = 0, M bits = 10, ONEBIT = 1, BRR [3:0] = 0000.

In these conditions, according to *Table 167: Tolerance of the USART receiver when BRR [3:0] = 0000*, the USART receiver tolerance is 4.86 %.

DTRA + DQUANT + DREC + DTCL + DWU < USART receiver's tolerance

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE=1 in the USART_CR1 register.

In Smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF will be set.

If the USART does not support the Receiver timeout feature, this bit is reserved and forced by hardware to '0'.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE=1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and forced by hardware to '0'.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

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

Bit 7 **TXE**: Transmit data register empty

This bit is set by hardware when the content of the USART_TDR register has been transferred into the shift register. It is cleared by a write to the USART_TDR register.

The TXE flag can also be cleared by writing 1 to the TXFRQ in the USART_RQR register, in order to discard the data (only in Smartcard T=0 mode, in case of transmission failure).

An interrupt is generated if the TXEIE bit =1 in the USART_CR1 register.

0: data is not transferred to the shift register

1: data is transferred to the shift register)

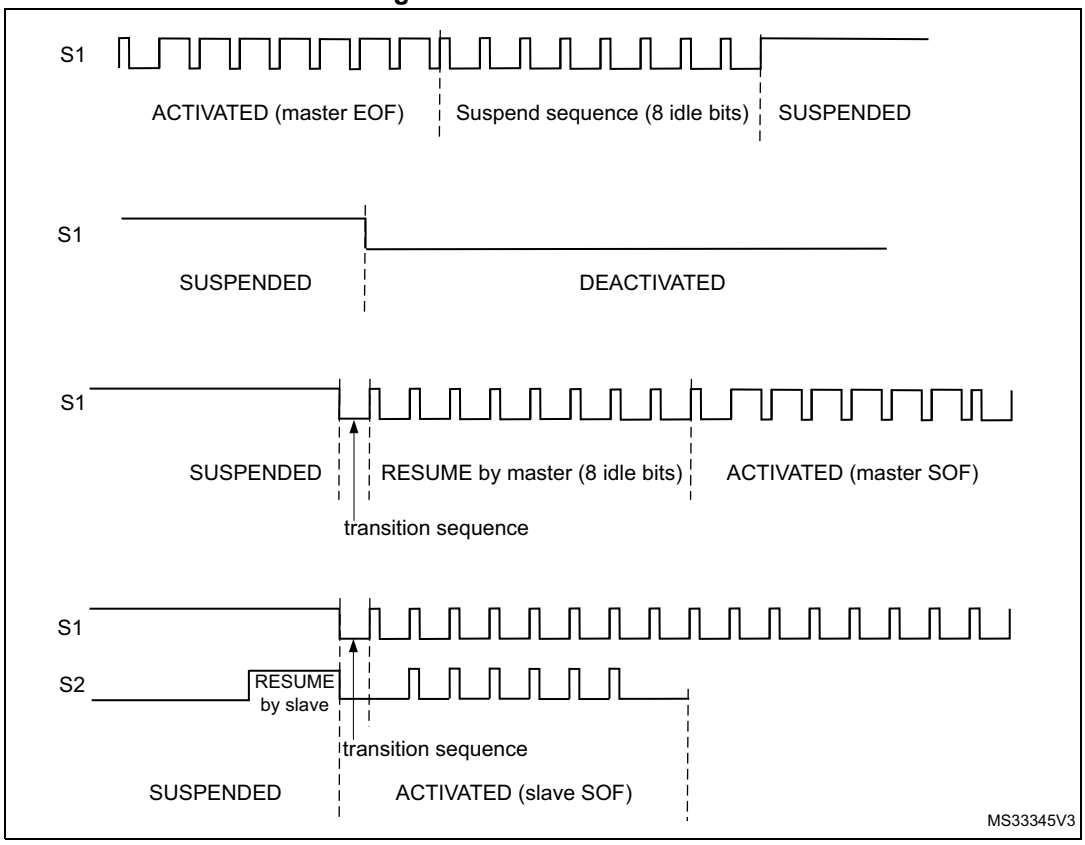
Note: This bit is used during single buffer transmission.

Deactivate mode

In order to switch the SWP to the DEACTIVATED mode immediately, ignoring any possible incoming RESUME by slave, the user must clear SWPACT bit in the SWPMI_CR register.

Note: In order to further reduce current consumption once SWPACT bit is cleared, configure the SWPMI_IO port as output push pull low in GPIO controller (refer to Section 8: General-purpose I/Os (GPIO)).

Figure 418. SWP bus states



40.3.4 SWPMI_IO (internal transceiver) bypass

A SWPMI_IO (transceiver), compliant with ETSI TS 102 613 technical specification, is embedded in the microcontroller. Nevertheless, this is possible to bypass it by setting SWP_TBYP bit in SWPMI_OR register. In this case, the SWPMI_IO is disabled and the SWPMI_RX, SWPMI_TX and SWPMI_SUSPEND signals are available as alternate functions on three GPIOs (refer to “Pinouts and pin description” in product datasheet). This configuration is selected to connect an external transceiver.

40.3.5 SWPMI Bit rate

The bit rate must be set in the SWPMI_BRR register, according to the following formula:

$F_{SWP} = F_{SWPCLK} / ((BR[5:0]+1)x4)$

Note: The maximum bitrate is 2 Mbit/s.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	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.	HWFC_EN	NEGEDGE	WIDBUS		BYPASS	PWRSAV	CLKEN	CLKDIV							
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 14 **HWFC_EN:** HW Flow Control enable

0b: HW Flow Control is disabled

1b: HW Flow Control is enabled

When HW Flow Control is enabled, the meaning of the TXFIFOE and RXFIFOF interrupt signals, see SDMMC Status register definition in *Section 41.8.11*.

Bit 13 **NEGEDGE:** SDMMC_CK dephasing selection bit

0b: Command and Data changed on the SDMMCCLK falling edge succeeding the rising edge of SDMMC_CK. (SDMMC_CK rising edge occurs on SDMMCCLK rising edge).

1b: Command and Data changed on the SDMMC_CK falling edge.

When BYPASS is active, the data and the command change on SDMMCCLK falling edge whatever NEGEDGE value.

Bits 12:11 **WIDBUS:** Wide bus mode enable bit

00: Default bus mode: SDMMC_D0 used

01: 4-wide bus mode: SDMMC_D[3:0] used

10: 8-wide bus mode: SDMMC_D[7:0] used

Bit 10 **BYPASS:** Clock divider bypass enable bit

0: Disable bypass: SDMMCCLK is divided according to the CLKDIV value before driving the SDMMC_CK output signal.

1: Enable bypass: SDMMCCLK directly drives the SDMMC_CK output signal.

Bit 9 **PWRSAV:** Power saving configuration bit

For power saving, the SDMMC_CK clock output can be disabled when the bus is idle by setting PWRSAV:

0: SDMMC_CK clock is always enabled

1: SDMMC_CK is only enabled when the bus is active

Bit 8 **CLKEN:** Clock enable bit

0: SDMMC_CK is disabled

1: SDMMC_CK is enabled

Bits 7:0 **CLKDIV:** Clock divide factor

This field defines the divide factor between the input clock (SDMMCCLK) and the output clock (SDMMC_CK): SDMMC_CK frequency = SDMMCCLK / [CLKDIV + 2].

Note: 1 While the SD/SDIO card or MultiMediaCard is in identification mode, the SDMMC_CK frequency must be less than 400 kHz.

2 The clock frequency can be changed to the maximum card bus frequency when relative card addresses are assigned to all cards.

3 After a data write, data cannot be written to this register for three SDMMCCLK clock periods plus two PCLK2 clock periods. SDMMC_CK can also be stopped during the read wait interval for SD I/O cards: in this case the SDMMC_CLKCR register does not control SDMMC_CK.

Each packet buffer is used either during reception or transmission starting from the bottom. The USB peripheral will never change the contents of memory locations adjacent to the allocated memory buffers; if a packet bigger than the allocated buffer length is received (buffer overrun condition) the data will be copied to the memory only up to the last available location.

Endpoint initialization

The first step to initialize an endpoint is to write appropriate values to the ADDRn_TX/ADDRn_RX registers so that the USB peripheral finds the data to be transmitted already available and the data to be received can be buffered. The EP_TYPE bits in the USB_EPnR register must be set according to the endpoint type, eventually using the EP_KIND bit to enable any special required feature. On the transmit side, the endpoint must be enabled using the STAT_TX bits in the USB_EPnR register and COUNTn_TX must be initialized. For reception, STAT_RX bits must be set to enable reception and COUNTn_RX must be written with the allocated buffer size using the BL_SIZE and NUM_BLOCK fields. Unidirectional endpoints, except Isochronous and double-buffered bulk endpoints, need to initialize only bits and registers related to the supported direction. Once the transmission and/or reception are enabled, register USB_EPnR and locations ADDRn_TX/ADDRn_RX, COUNTn_TX/COUNTn_RX (respectively), should not be modified by the application software, as the hardware can change their value on the fly. When the data transfer operation is completed, notified by a CTR interrupt event, they can be accessed again to re-enable a new operation.

IN packets (data transmission)

When receiving an IN token packet, if the received address matches a configured and valid endpoint, the USB peripheral accesses the contents of ADDRn_TX and COUNTn_TX locations inside the buffer descriptor table entry related to the addressed endpoint. The content of these locations is stored in its internal 16 bit registers ADDR and COUNT (not accessible by software). The packet memory is accessed again to read the first byte to be transmitted (Refer to *Structure and usage of packet buffers on page 1401*) and starts sending a DATA0 or DATA1 PID according to USB_EPnR bit DTOG_TX. When the PID is completed, the first byte, read from buffer memory, is loaded into the output shift register to be transmitted on the USB bus. After the last data byte is transmitted, the computed CRC is sent. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the data packet, according to STAT_TX bits in the USB_EPnR register.

The ADDR internal register is used as a pointer to the current buffer memory location while COUNT is used to count the number of remaining bytes to be transmitted. Each half-word read from the packet buffer memory is transmitted over the USB bus starting from the least significant byte. Transmission buffer memory is read starting from the address pointed by ADDRn_TX for COUNTn_TX/2 half-words. If a transmitted packet is composed of an odd number of bytes, only the lower half of the last half-word accessed will be used.

On receiving the ACK receipt by the host, the USB_EPnR register is updated in the following way: DTOG_TX bit is toggled, the endpoint is made invalid by setting STAT_TX=10 (NAK) and bit CTR_TX is set. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the EP_ID and DIR bits in the USB_ISTR register. Servicing of the CTR_TX event starts clearing the interrupt bit; the application software then prepares another buffer full of data to be sent, updates the COUNTn_TX table location with the number of byte to be transmitted during the next transfer, and finally sets STAT_TX to ‘11 (VALID) to re-enable transmissions. While the STAT_TX bits are equal to ‘10 (NAK), any IN request addressed to that endpoint is NAKed,

Bits 13:12 **STAT_RX [1:0]:** Status bits, for reception transfers

These bits contain information about the endpoint status, which are listed in *Table 238: Reception status encoding on page 1423*.These bits can be toggled by software to initialize their value. When the application software writes ‘0, the value remains unchanged, while writing ‘1 makes the bit value toggle. Hardware sets the STAT_RX bits to NAK when a correct transfer has occurred (CTR_RX=1) corresponding to a OUT or SETUP (control only) transaction addressed to this endpoint, so the software has the time to elaborate the received data before it acknowledge a new transaction

Double-buffered bulk endpoints implement a special transaction flow control, which control the status based upon buffer availability condition (Refer to *Section 43.5.3: Double-buffered endpoints*).

If the endpoint is defined as Isochronous, its status can be only “VALID” or “DISABLED”, so that the hardware cannot change the status of the endpoint after a successful transaction. If the software sets the STAT_RX bits to ‘STALL’ or ‘NAK’ for an Isochronous endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing ‘1.

Bit 11 **SETUP:** Setup transaction completed

This bit is read-only and it is set by the hardware when the last completed transaction is a SETUP. This bit changes its value only for control endpoints. It must be examined, in the case of a successful receive transaction (CTR_RX event), to determine the type of transaction occurred. To protect the interrupt service routine from the changes in SETUP bits due to next incoming tokens, this bit is kept frozen while CTR_RX bit is at 1; its state changes when CTR_RX is at 0. This bit is read-only.

Bits 10:9 **EP_TYPE[1:0]:** Endpoint type

These bits configure the behavior of this endpoint as described in *Table 239: Endpoint type encoding on page 1424*. Endpoint 0 must always be a control endpoint and each USB function must have at least one control endpoint which has address 0, but there may be other control endpoints if required. Only control endpoints handle SETUP transactions, which are ignored by endpoints of other kinds. SETUP transactions cannot be answered with NAK or STALL. If a control endpoint is defined as NAK, the USB peripheral will not answer, simulating a receive error, in the receive direction when a SETUP transaction is received. If the control endpoint is defined as STALL in the receive direction, then the SETUP packet will be accepted anyway, transferring data and issuing the CTR interrupt. The reception of OUT transactions is handled in the normal way, even if the endpoint is a control one.

Bulk and interrupt endpoints have very similar behavior and they differ only in the special feature available using the EP_KIND configuration bit.

The usage of Isochronous endpoints is explained in *Section 43.5.4: Isochronous transfers*

Bit 8 **EP_KIND:** Endpoint kind

The meaning of this bit depends on the endpoint type configured by the EP_TYPE bits. *Table 240* summarizes the different meanings.

DBL_BUF: This bit is set by the software to enable the double-buffering feature for this bulk endpoint. The usage of double-buffered bulk endpoints is explained in *Section 43.5.3: Double-buffered endpoints*.

STATUS_OUT: This bit is set by the software to indicate that a status out transaction is expected: in this case all OUT transactions containing more than zero data bytes are answered ‘STALL’ instead of ‘ACK’. This bit may be used to improve the robustness of the application to protocol errors during control transfers and its usage is intended for control endpoints only. When STATUS_OUT is reset, OUT transactions can have any number of bytes, as required.