Timer & Pulse-Width Modulation

Microcontroller Application and Development 2564

Sorayut Glomglome

Outline

1. Basic Block Diagram
2. Advanced-control Timer
3. Pulse-width Modulation
4. General Purpose Timer
5. SysTick Timer
6. Watchdog Timer

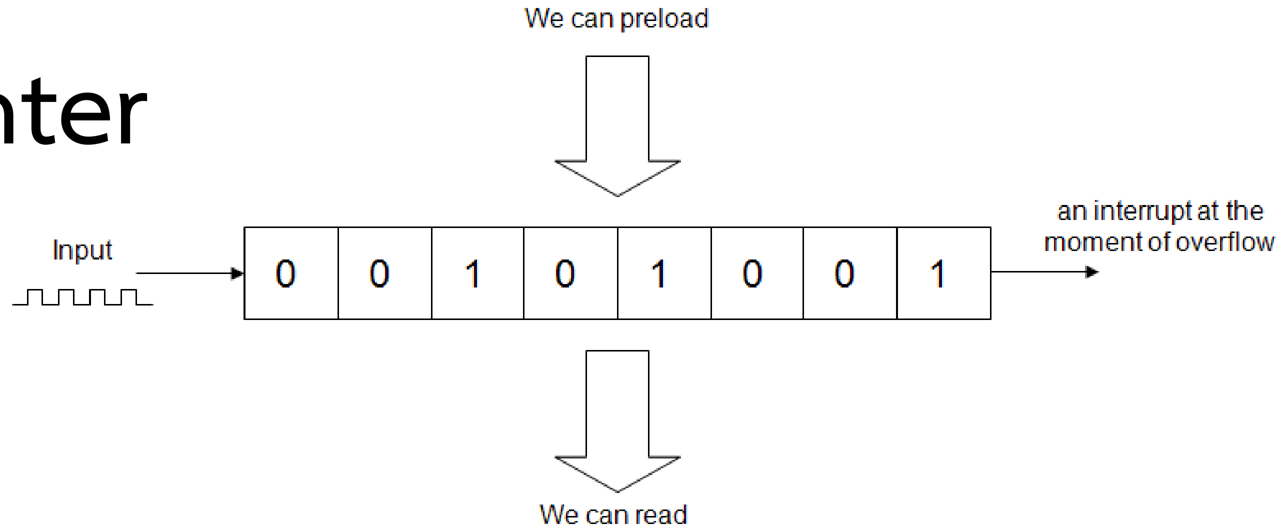
Learning Outcomes

1. Explain the use of timer
2. Config timing interrupt for specific period
3. Generate pulse-width modulation signal with specific duty cycle

Applications

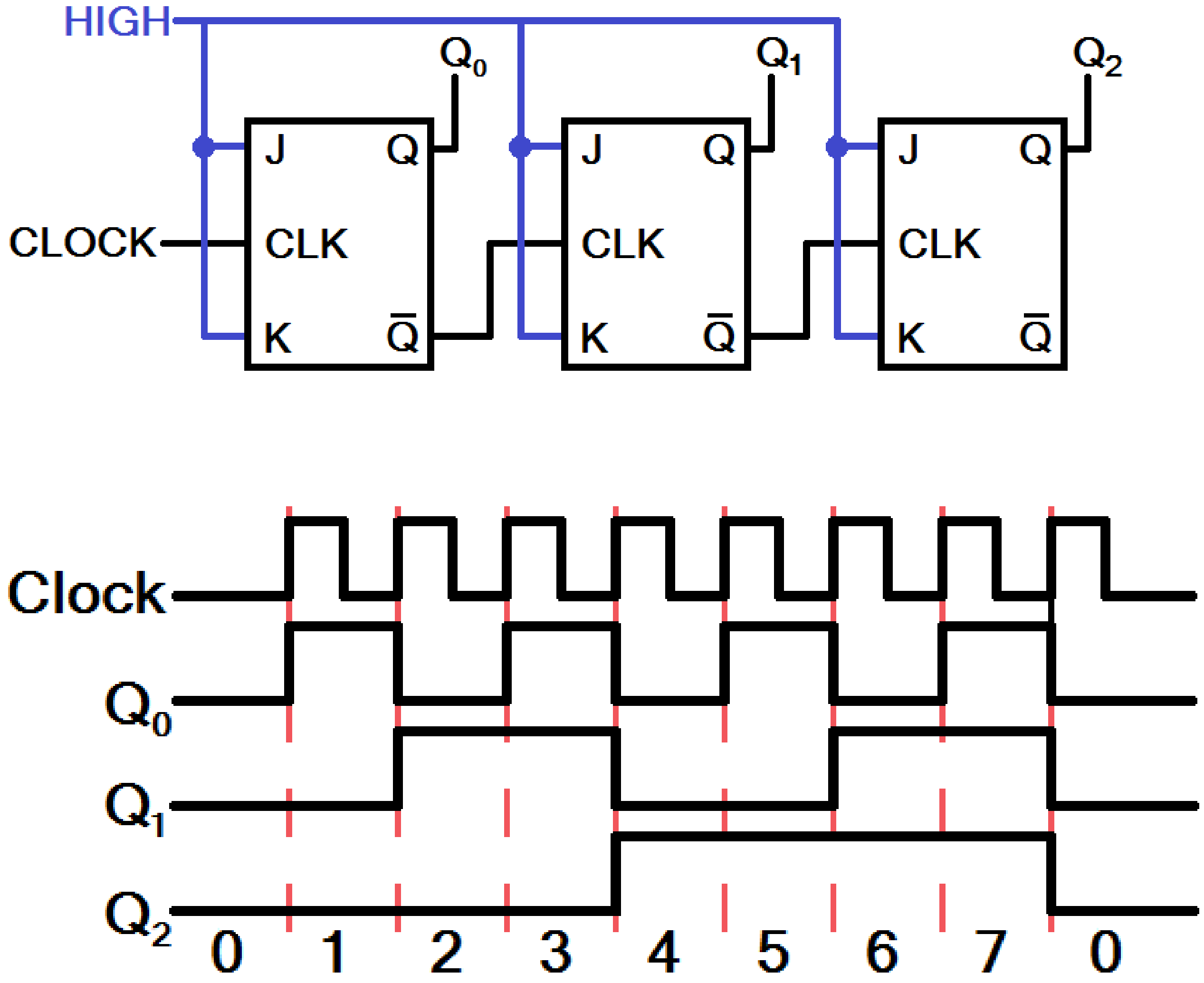
- Interval Timer for counting internal events.
- Pulse Width Demodulator via Capture inputs.
- Free running timer.

The Digital Counter

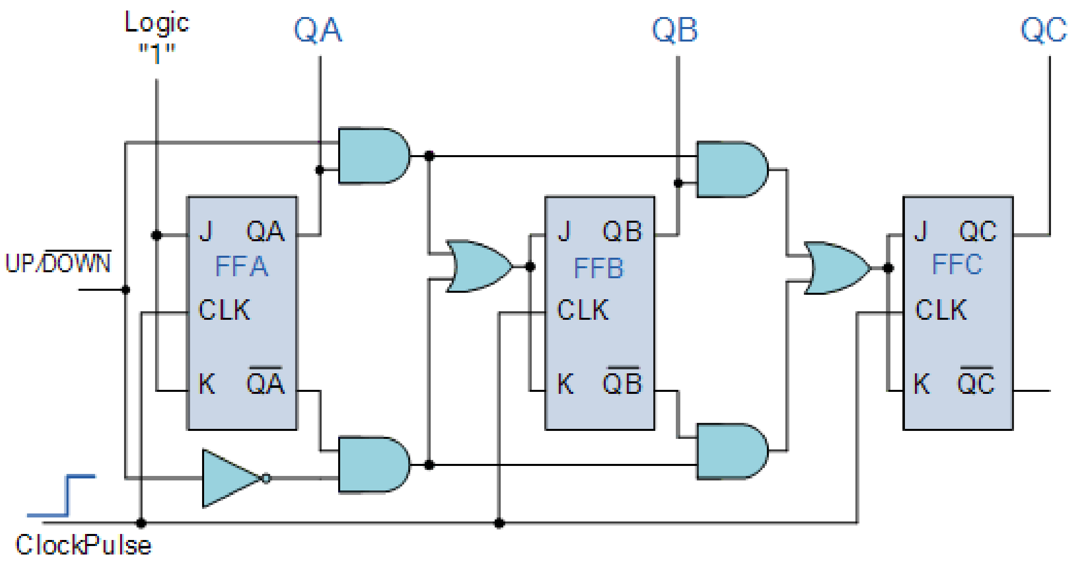


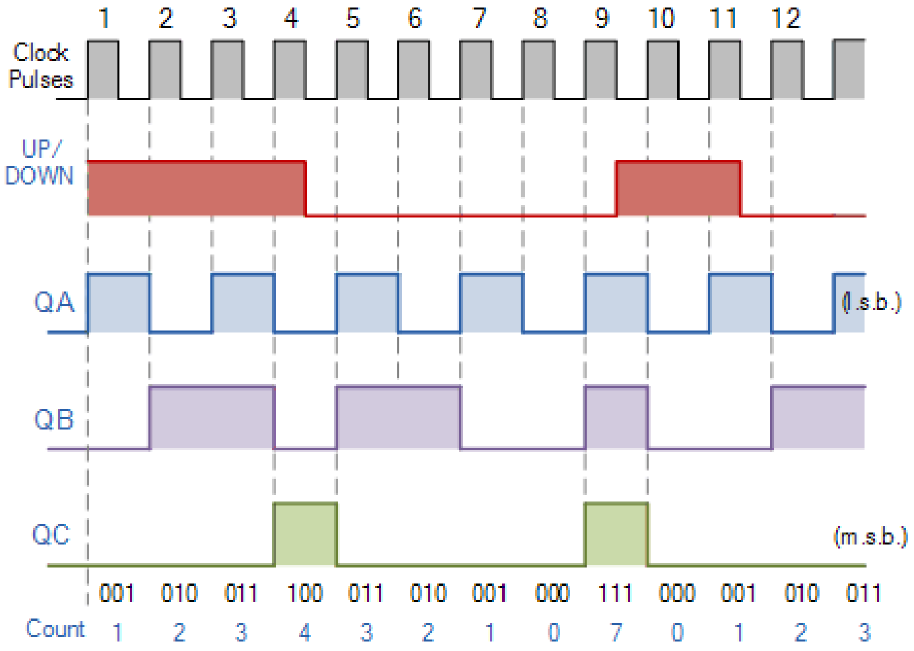
- n-bit counter can count from 0 to $(2^n - 1)$
- 8-bit counter can count from 0000 0000 to 1111 1111 or 0 to 255
- Overflow then start over; 0 ➔ 255 ➔0
- Counting is according to clock pulses

3-bit Up Counter



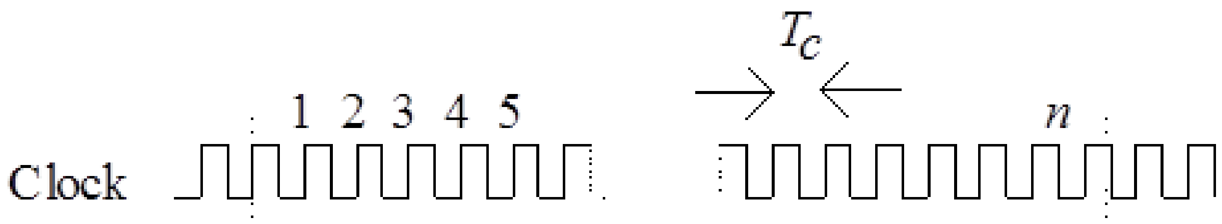
3-bit Synchronous Up/Down Counter





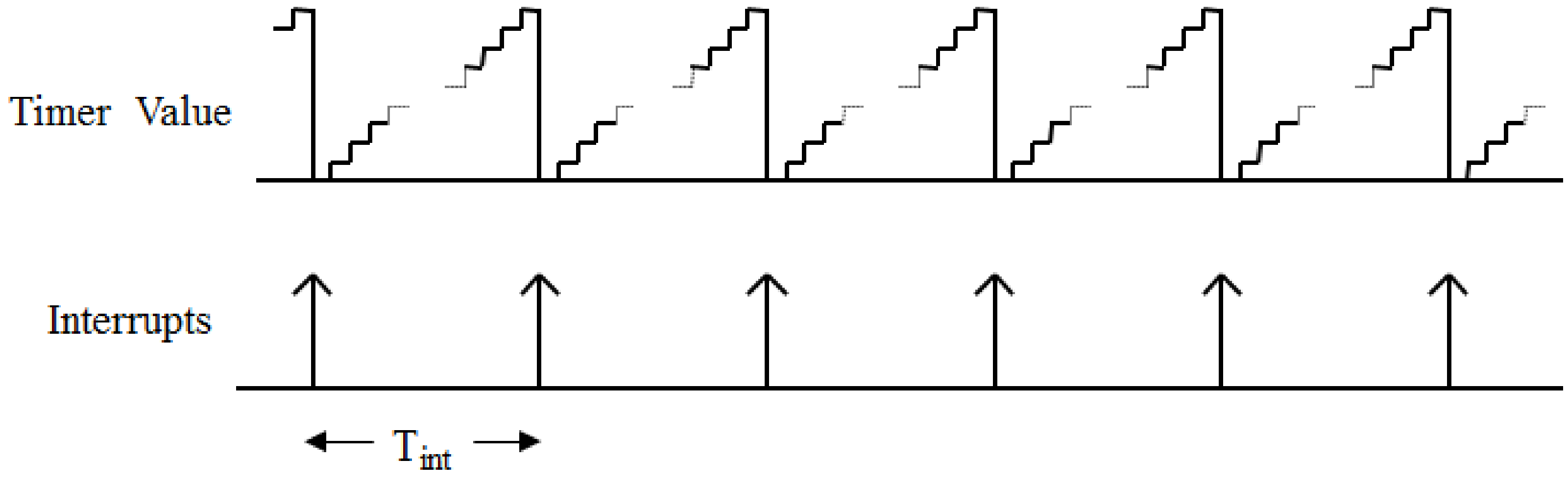
Counting and Timing

- An 8 bit counter with 1 MHz clock frequency.
- Clock period is 1 μs.
- Counting from 0➔255 ➔0 takes 256*1 μs

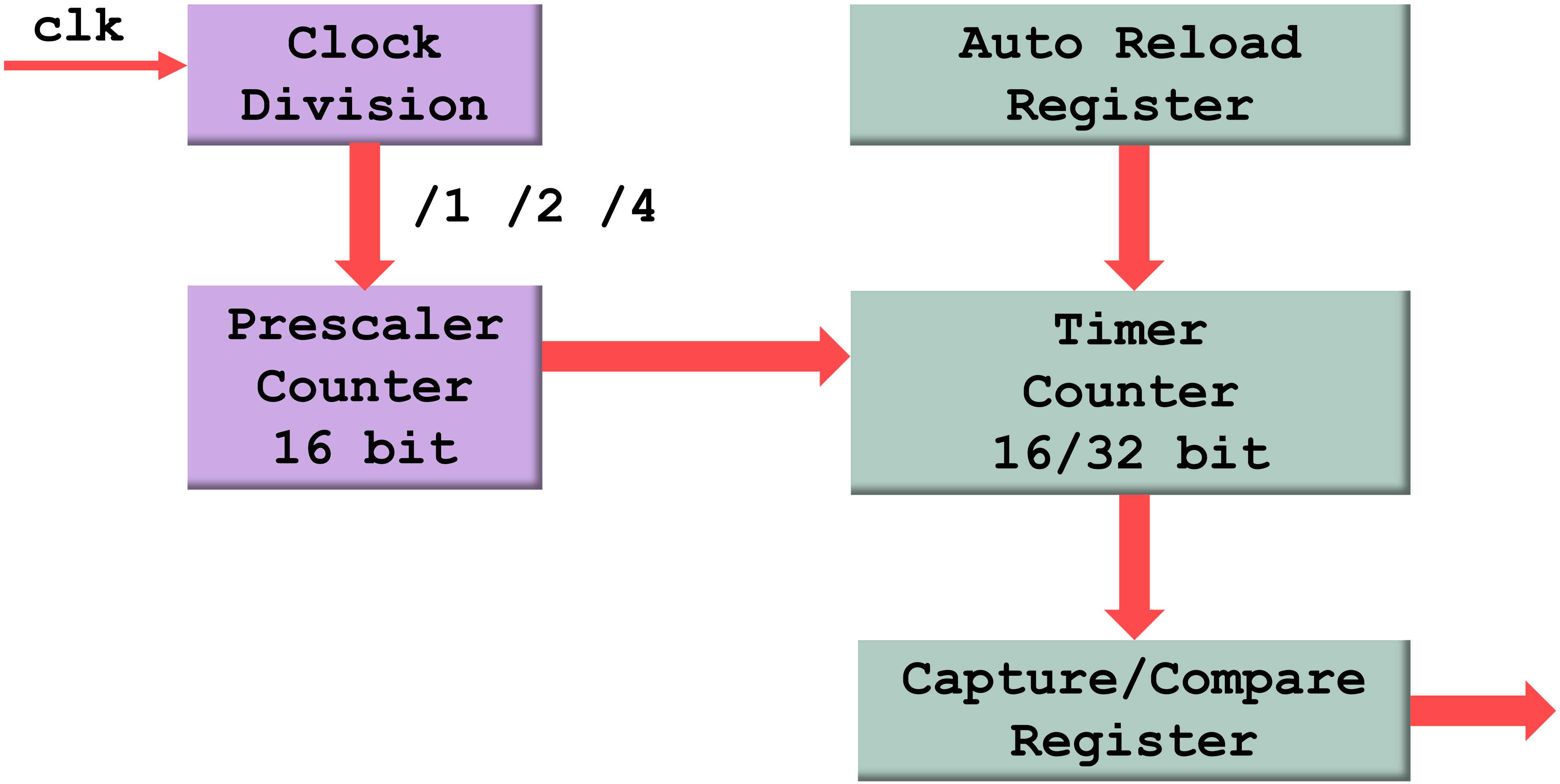


Counting and Timing – Interrupt on Overflow

- A counter overflow (0➔255 ➔0) can generate an interrupt signal.
- If a counter continues counting after an overflow, then interrupt signals will occur repeatedly.



Timer Basic Block Diagram



Time Interval

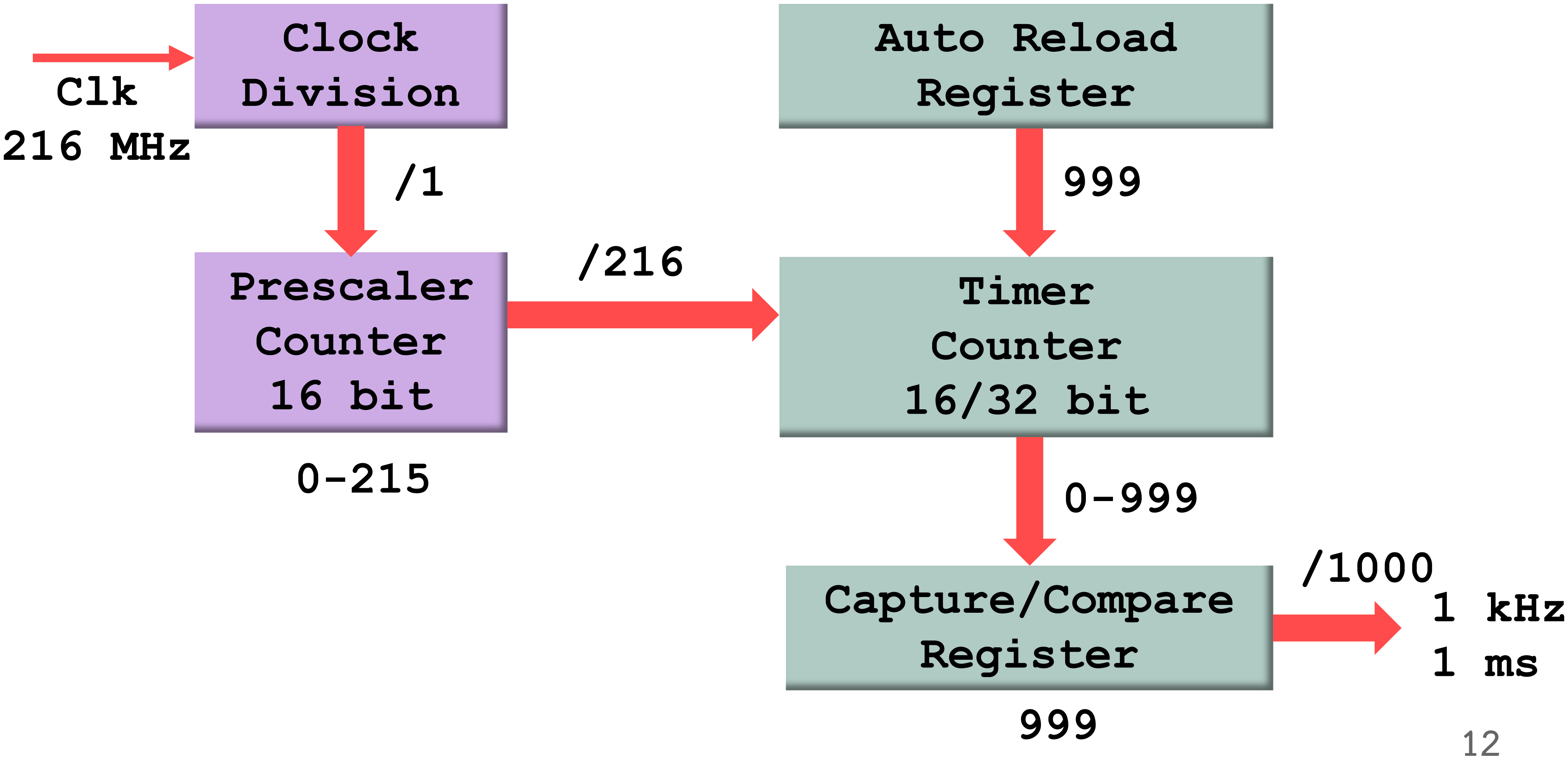
$$\frac{Timer\ Clock\ Speed\ (Hz)}{Clock\ Division \times Prescaler \times Period} = Interrupt\ Frequency\ (Hz)$$

1, 2, 4 16 bit 16/32 bit

$$\frac{216\ MHz}{1 \times 216 \times 1000} = 1\ kHz$$

$$Time\ Interval = 1\ ms$$

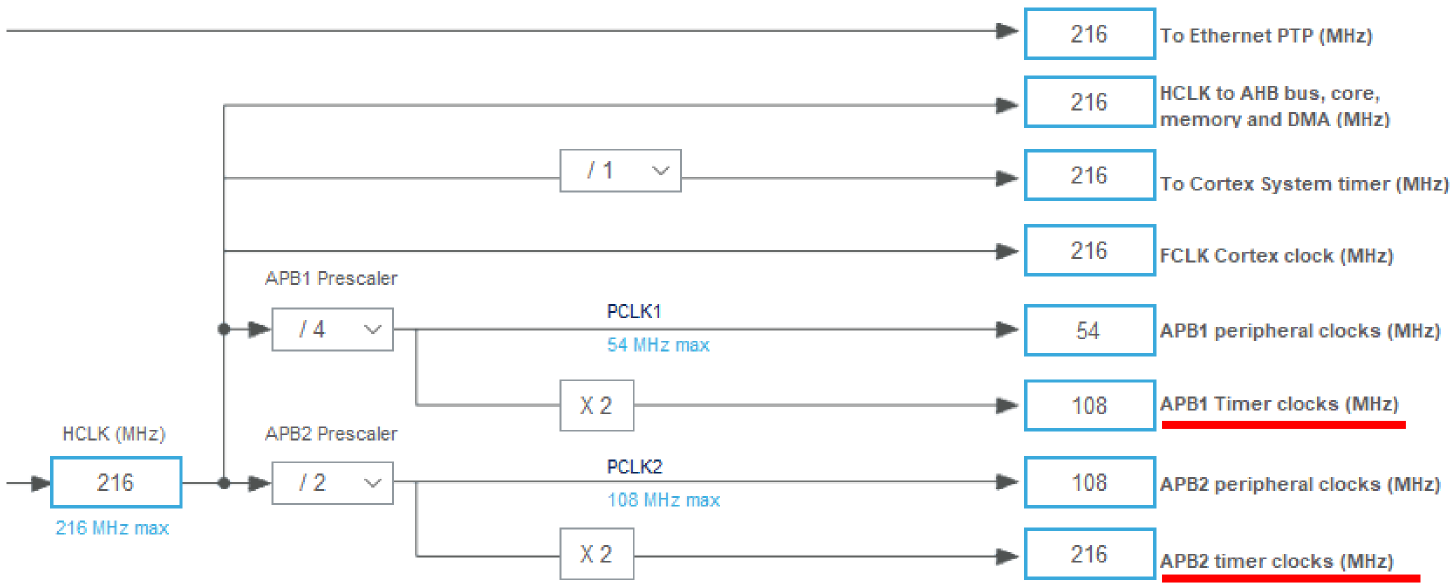
Timer Basic Block Diagram



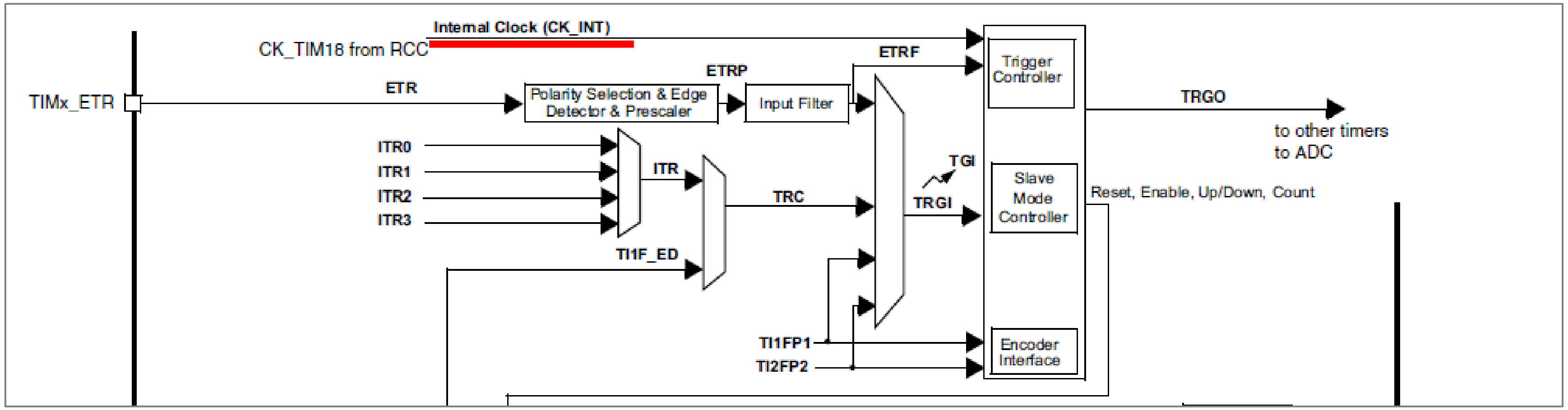
TIMx and APB clock Speed

Bus	MAX Bus Frequency (MHz)	MAX Timer Frequency (MHz)	TIMER Module
APB1	54	108	TIM2 TIM3 TIM4 TIM5 TIM12 TIM13 TIM14
APB2	108	216	TIM1 TIM8 TIM9 TIM10 TIM11

Clock Input



216
To Ethernet PTP (MHz)
216
HCLK to AHB bus, core, memory and DMA (MHz)
/ 1
216
To Cortex System timer (MHz)
216
FCLK Cortex clock (MHz)
APB1 Prescaler
/ 4
PCLK1
54 MHz max
54
APB1 peripheral clocks (MHz)
X 2
108
APB1 Timer clocks (MHz)
HCLK (MHz)
216
216 MHz max
APB2 Prescaler
/ 2
PCLK2
108 MHz max
108
APB2 peripheral clocks (MHz)
X 2
216
APB2 timer clocks (MHz)

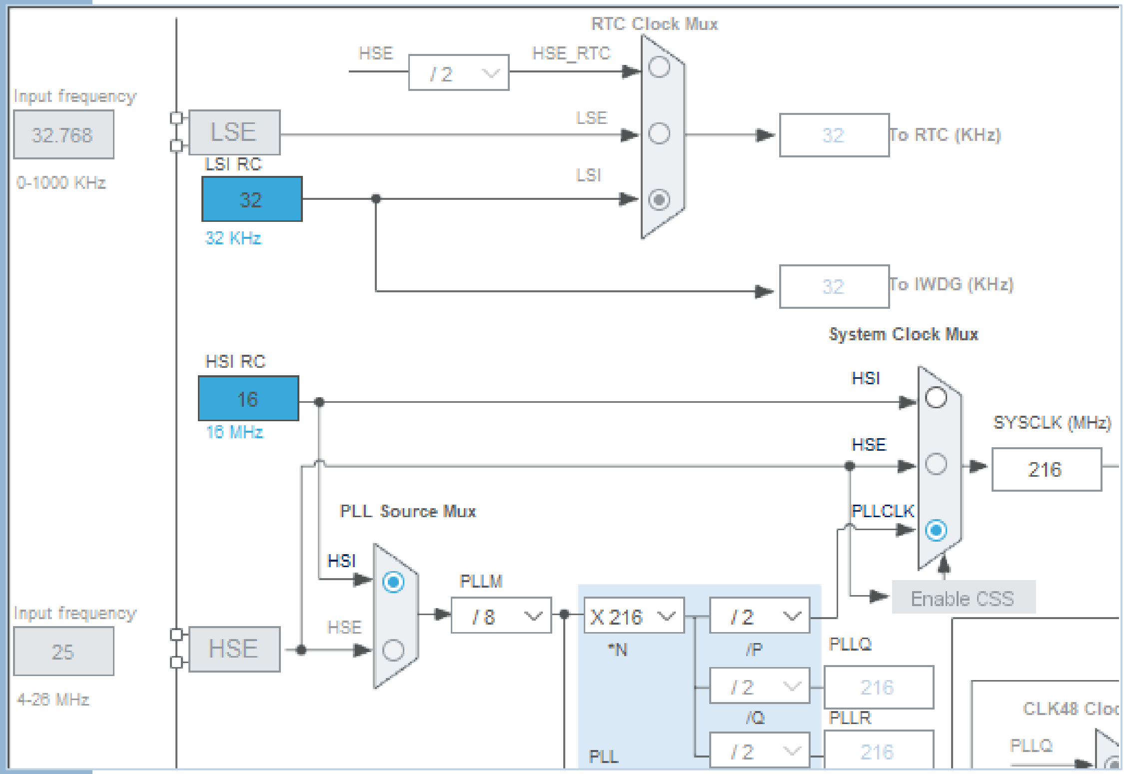


Internal Clock (CK_INT)
CK_TIM18 from RCC
ETRF
ETRP
ETR
TIMx_ETR
Polarity Selection & Edge Detector & Prescaler
Input Filter
Trigger Controller
TRGO
to other timers
to ADC
ITR0
ITR1
ITR2
ITR3
ITR
TRC
TGI
TRGI
Slave Mode Controller
Reset, Enable, Up/Down, Count
TI1F_ED
TI1FP1
TI2FP2
Encoder Interface

Clock Sources

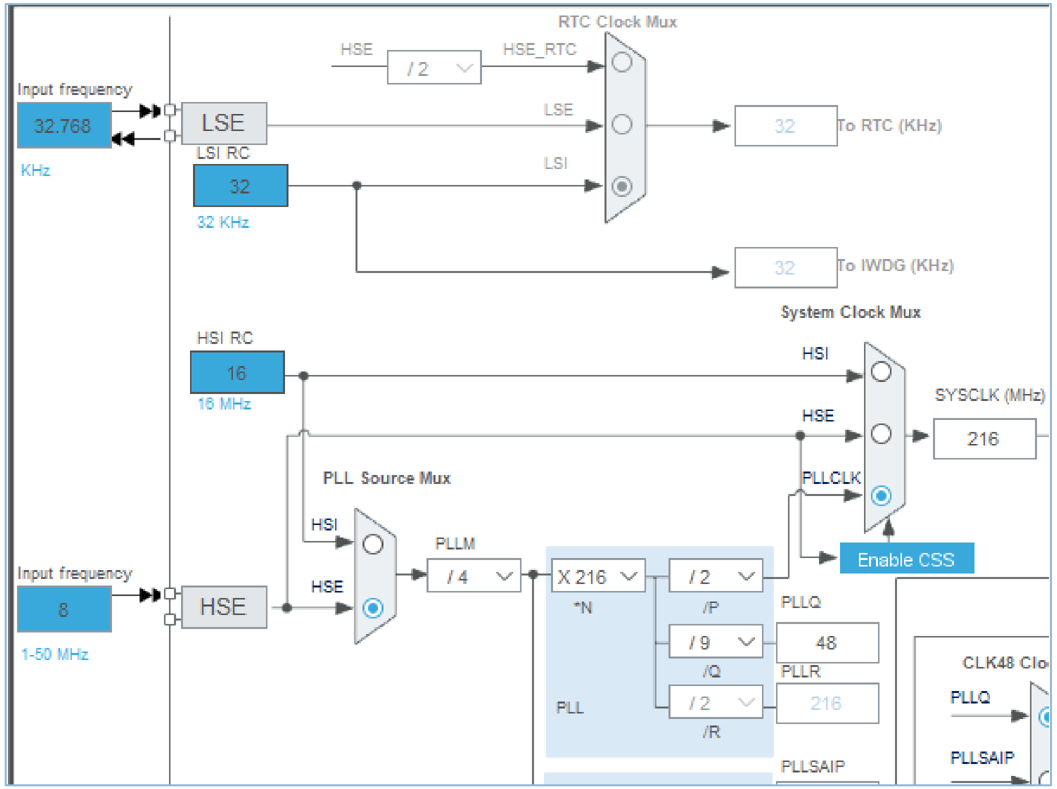
MCU Selector

STM32F767ZI



Board Selector

Nucleo-767ZI



STM32F767 Timers

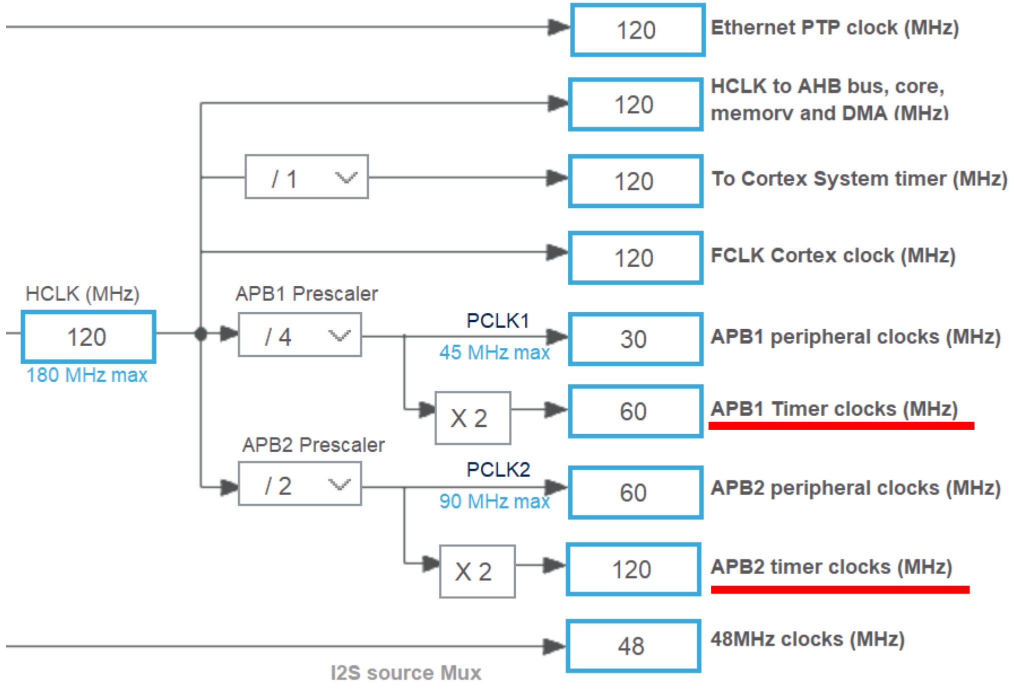
Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Complementary output	Max interface clock (MHz)	Max timer clock (MHz)[1]
Advanced -control	TIM1, TIM8	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	Yes	108	216
General purpose	TIM2, TIM5	32-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	54	108/216
	TIM3, TIM4	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	54	108/216
	TIM9	16-bit	Up	Any integer between 1 and 65536	No	2	No	108	216
	TIM10, TIM11	16-bit	Up	Any integer between 1 and 65536	No	1	No	108	216
	TIM12	16-bit	Up	Any integer between 1 and 65536	No	2	No	54	108/216
	TIM13, TIM14	16-bit	Up	Any integer between 1 and 65536	No	1	No	54	108/216
Basic	TIM6, TIM7	16-bit	Up	Any integer between 1 and 65536	Yes	0	No	54	108/216

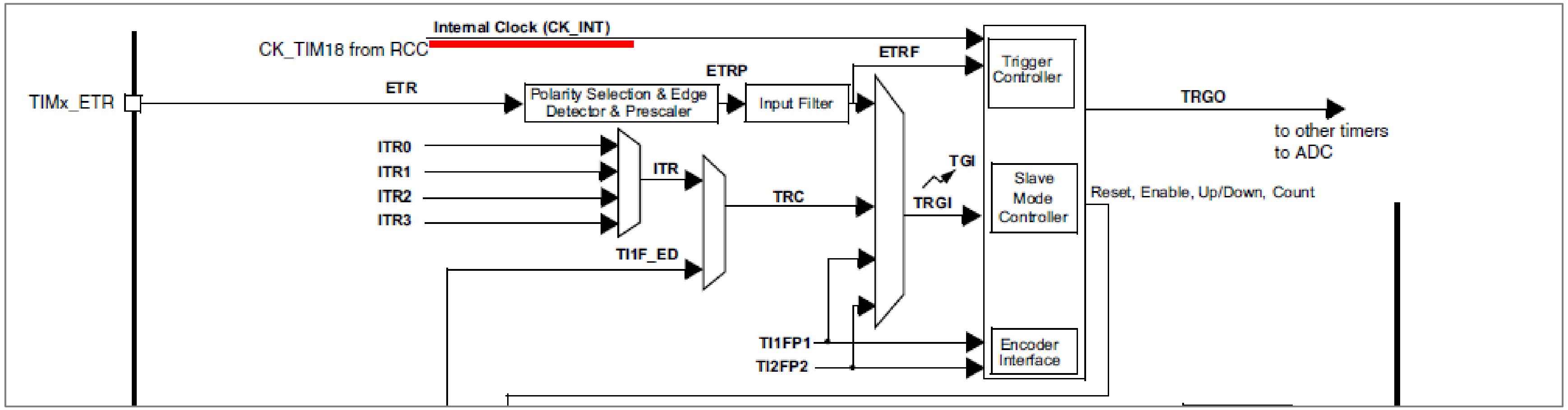
- 2 Advanced-control timer
- 10 General-purpose timers
- 2 Basic timers
- 1 Low-power timer
- 2 Watchdog timers
- 1 System Timer

TIMx and APB clock Speed – STM32F429ZI

Bus	MAX Bus Frequency (MHz)	MAX Timer Frequency (MHz)	TIMER Module
APB1	45	180	TIM2 TIM3 TIM4 TIM5 TIM12 TIM13 TIM14
APB2	90	180	TIM1 TIM8 TIM9 TIM10 TIM11

Clock Input


120
Ethernet PTP clock (MHz)
120
HCLK to AHB bus, core, memory and DMA (MHz)
/ 1
120
To Cortex System timer (MHz)
120
FCLK Cortex clock (MHz)
HCLK (MHz)
120
180 MHz max
APB1 Prescaler
/ 4
PCLK1
45 MHz max
30
APB1 peripheral clocks (MHz)
X 2
60
APB1 Timer clocks (MHz)
APB2 Prescaler
/ 2
PCLK2
90 MHz max
60
APB2 peripheral clocks (MHz)
X 2
120
APB2 timer clocks (MHz)
48
48MHz clocks (MHz)
I2S source Mux


Internal Clock (CK_INT)
CK_TIM18 from RCC
ETRF
ETRP
ETR
TIMx_ETR
Polarity Selection & Edge Detector & Prescaler
Input Filter
Trigger Controller
TRGO
to other timers
to ADC
ITR0
ITR1
ITR2
ITR3
ITR
TRC
TGI
TRGI
Slave Mode Controller
Reset, Enable, Up/Down, Count
TI1F_ED
TI1FP1
TI2FP2
Encoder Interface

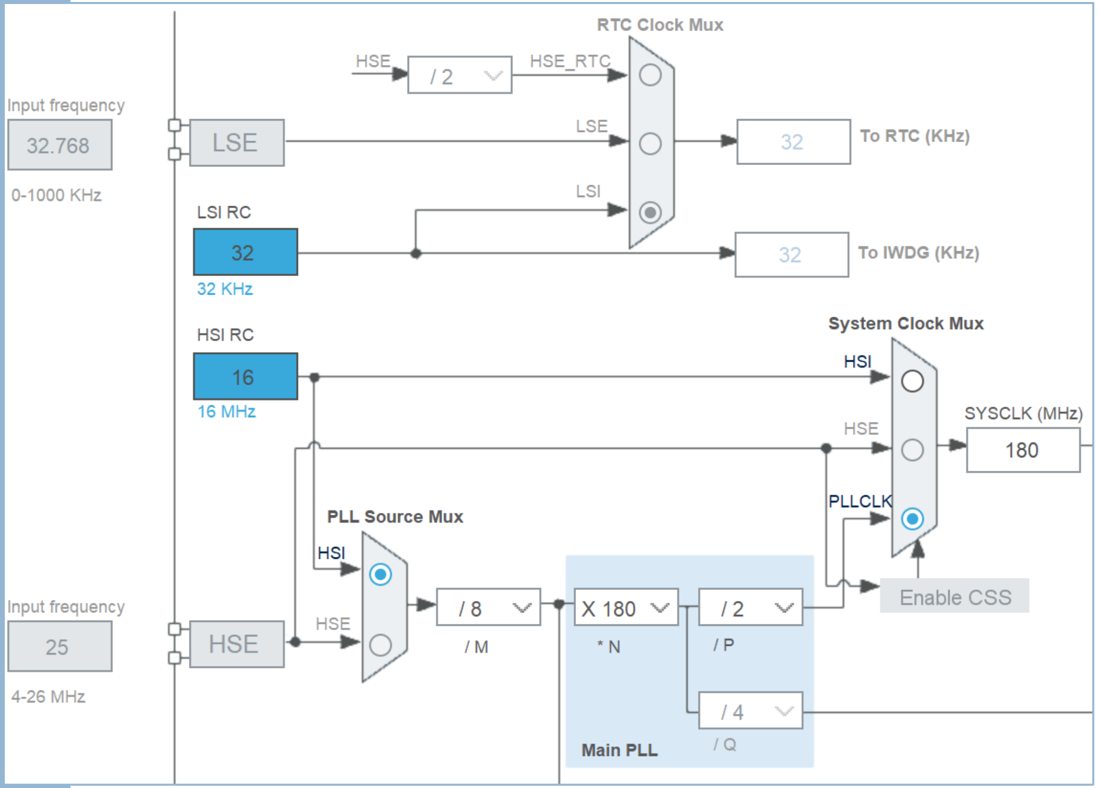
Clock Sources

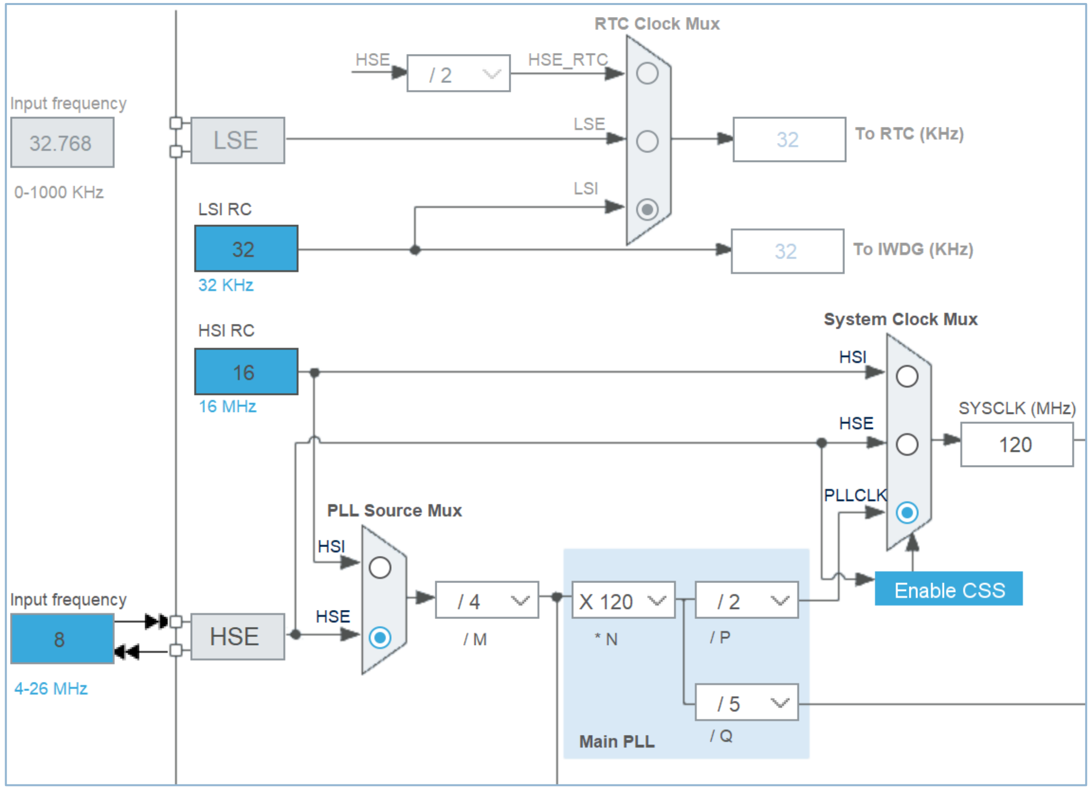
MCU Selector

STM32F429ZI

Board Selector

STM32F429-DISC1





STM32F429 Timers

Table 6. Timer feature comparison

Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Complementary output	Max interface clock (MHz)	Max timer clock (MHz)[1]
Advanced -control	TIM1, TIM8	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	Yes	90	180
General purpose	TIM2, TIM5	32-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	45	90/180
	TIM3, TIM4	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	45	90/180
	TIM9	16-bit	Up	Any integer between 1 and 65536	No	2	No	90	180
	TIM10, TIM11	16-bit	Up	Any integer between 1 and 65536	No	1	No	90	180
	TIM12	16-bit	Up	Any integer between 1 and 65536	No	2	No	45	90/180
	TIM13, TIM14	16-bit	Up	Any integer between 1 and 65536	No	1	No	45	90/180
Basic	TIM6, TIM7	16-bit	Up	Any integer between 1 and 65536	Yes	0	No	45	90/180

1. The maximum timer clock is either 90 or 180 MHz depending on TIMPRE bit configuration in the RCC_DCKCFGR register.

- 2 Advanced-control timer
- 10 General-purpose timers
- 2 Basic timers
- 2 Watchdog timers
- 1 System Timer

Advanced-control timer – TIM1/TIM8

TIM1/TIM8 introduction

The advanced-control timers (TIM1/TIM8) consist of a 16-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

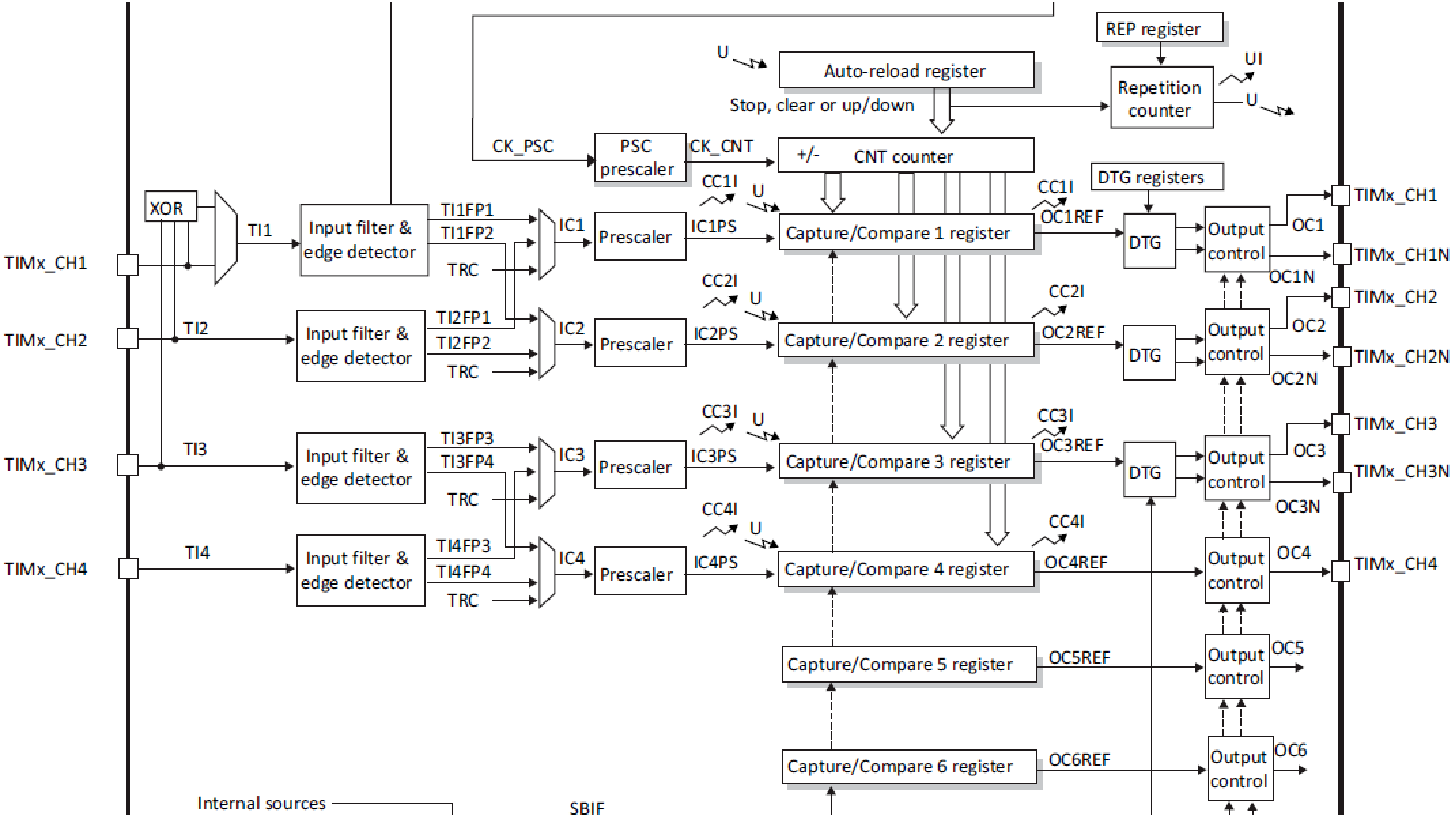
The advanced-control (TIM1/TIM8) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 25.3.25: Timer synchronization*.

TIM1/TIM8 main features

TIM1/TIM8 timer features include:

- 16-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 65536.
- Up to 6 independent channels for:
  - Input Capture (but channels 5 and 6)
  - Output Compare
  - PWM generation (Edge and Center-aligned Mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- 2 break inputs to put the timer’s output signals in a safe user selectable configuration.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare

Timer Block Diagram – Advanced Timer



Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

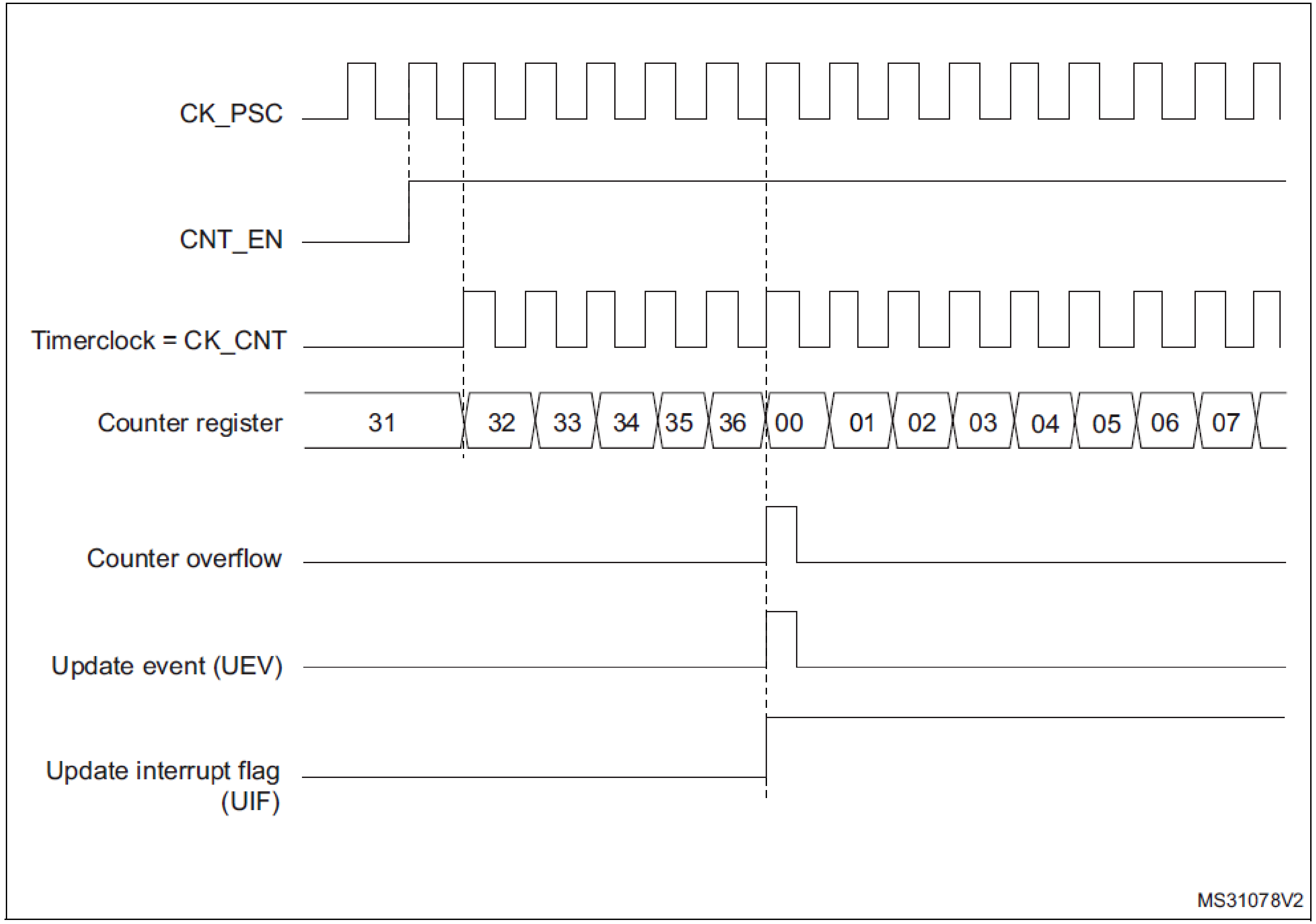
Up Counting Mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter overflow.

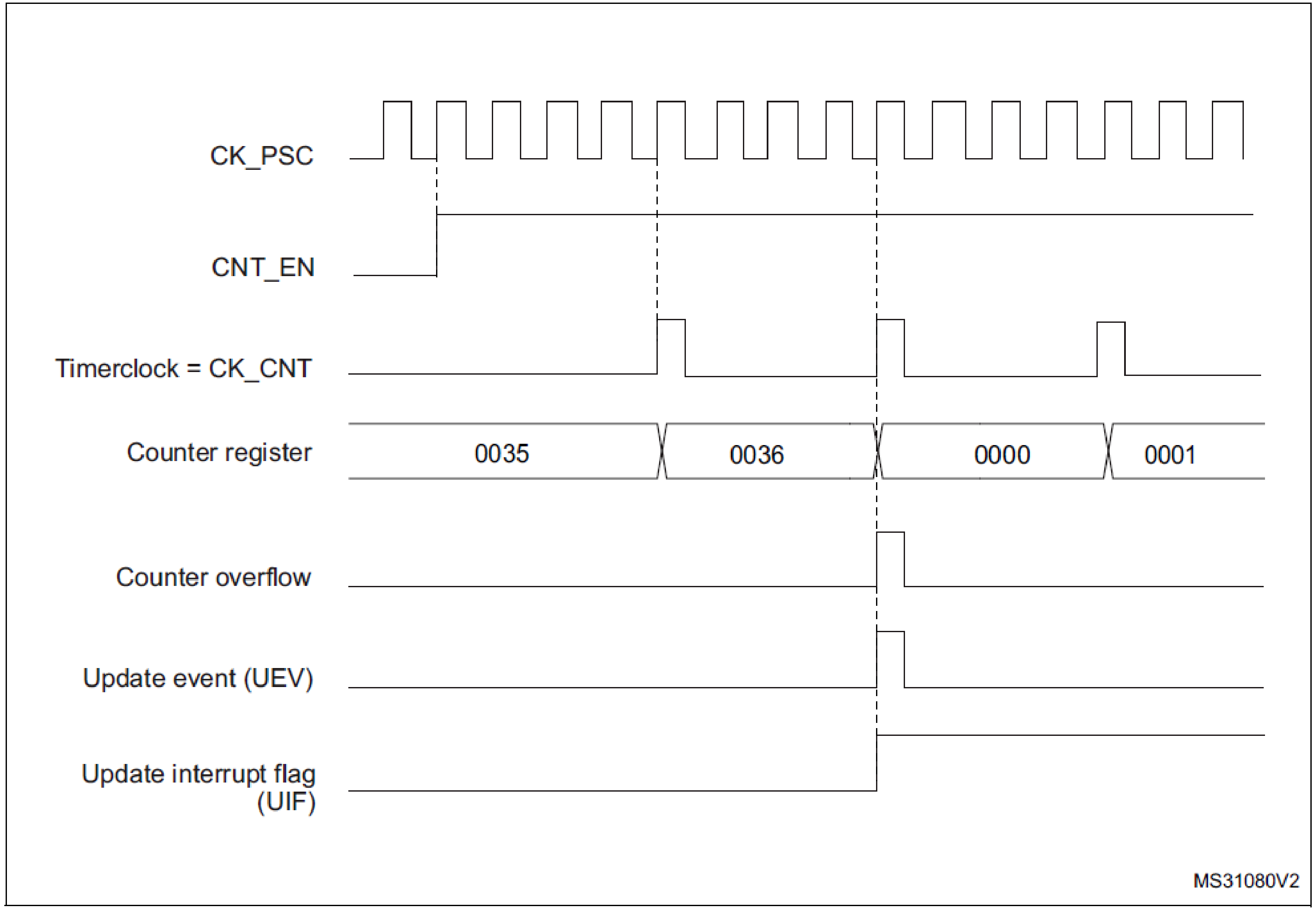
Up counting Mode, TIMx_ARR = 0x36

Figure 198. Counter timing diagram, internal clock divided by 1



Up counting Mode, TIMx_ARR = 0x36

Figure 200. Counter timing diagram, internal clock divided by 4



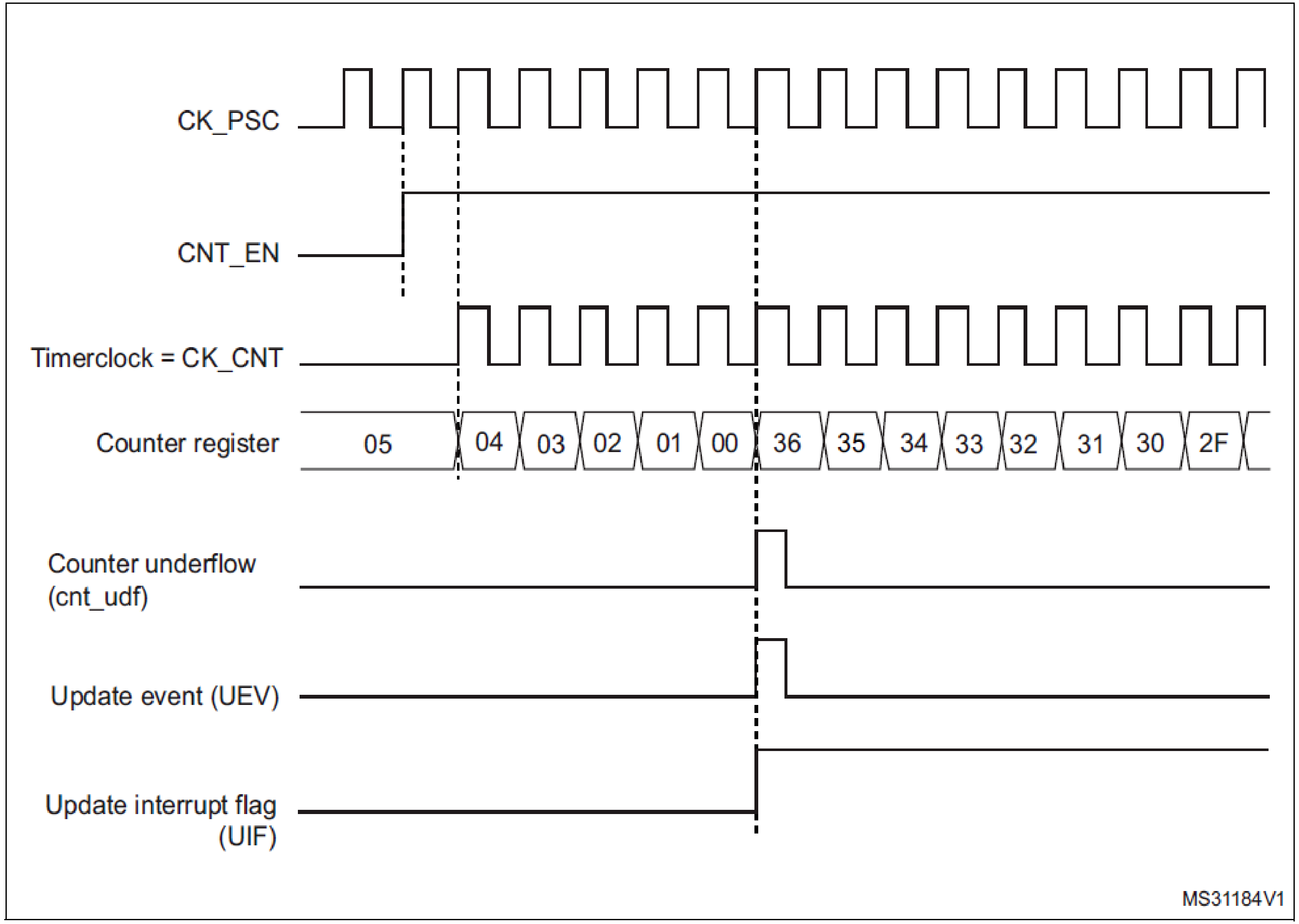
Down Counting Mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter underflow.

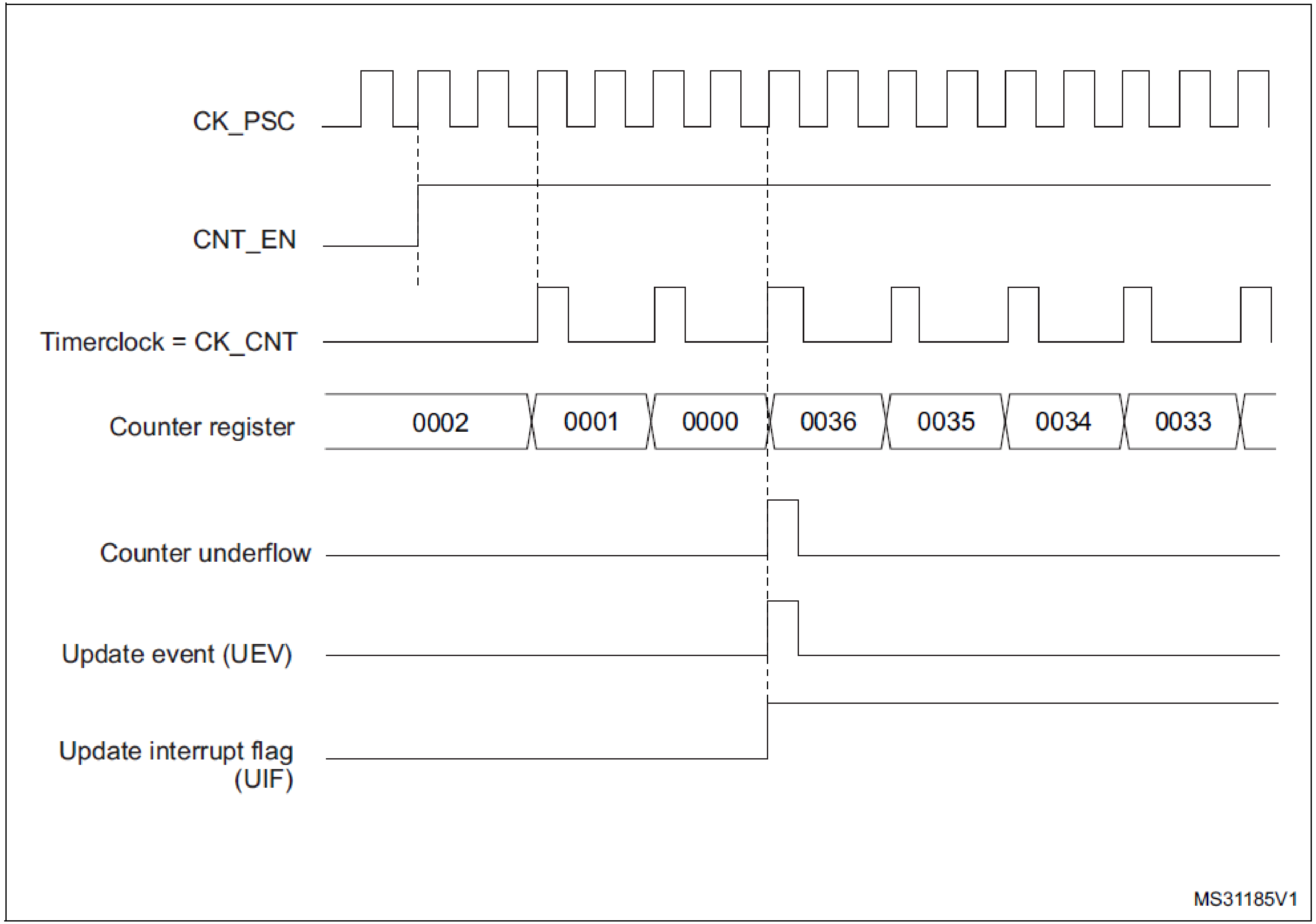
Down Counting Mode, TIMx_ARR = 0x36

Figure 204. Counter timing diagram, internal clock divided by 1

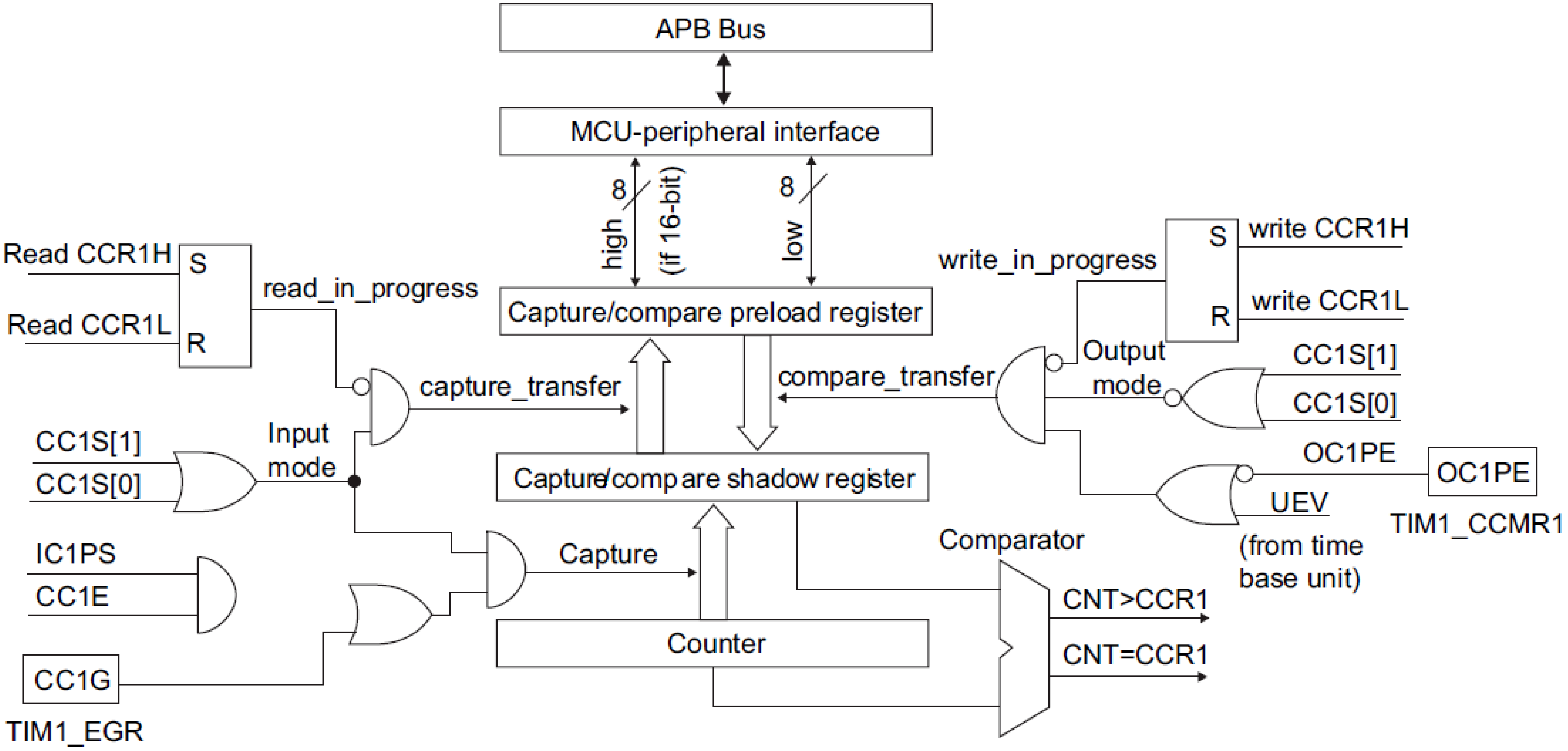


Down Counting Mode, TIMx_ARR = 0x36

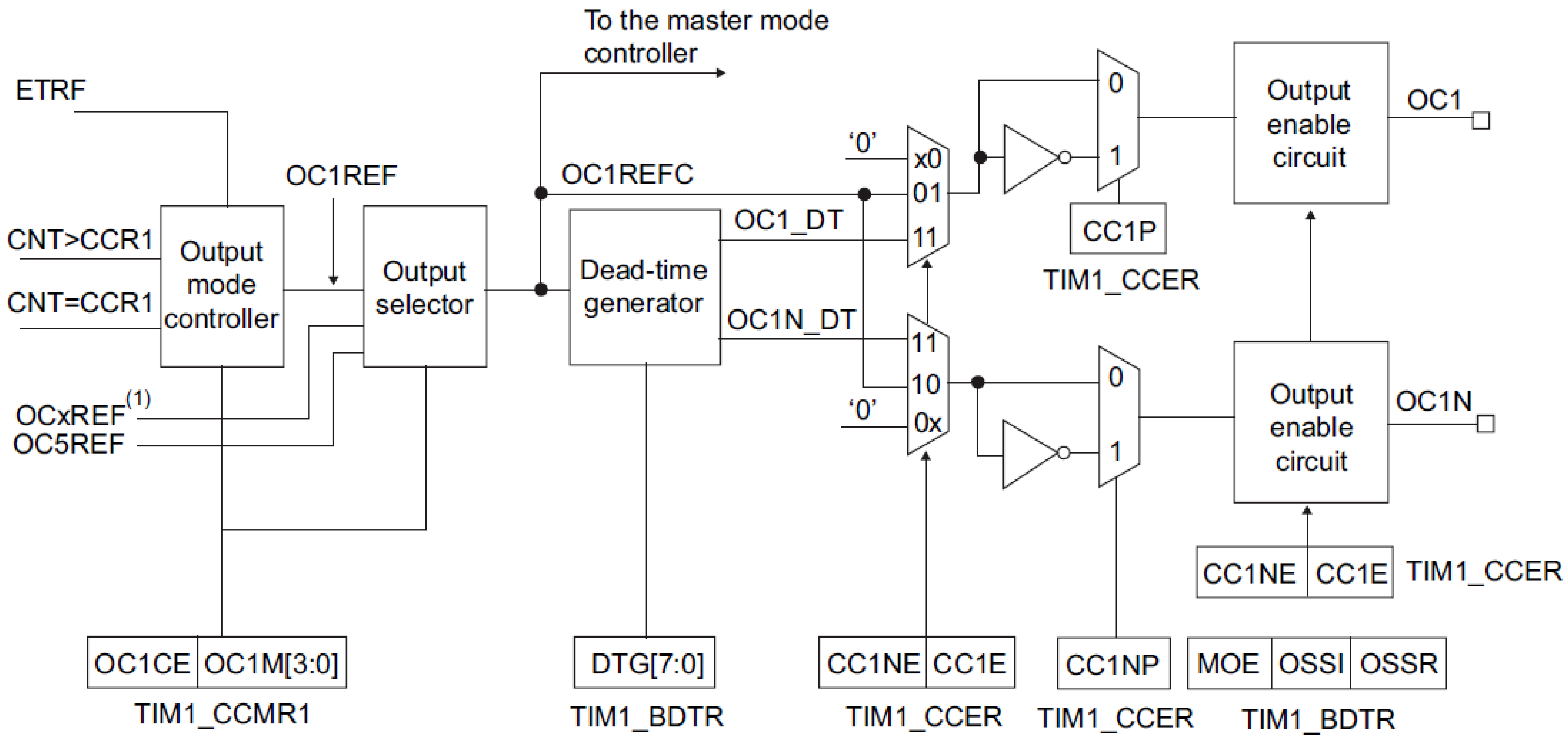
Figure 205. Counter timing diagram, internal clock divided by 2



Capture/Compare Circuit



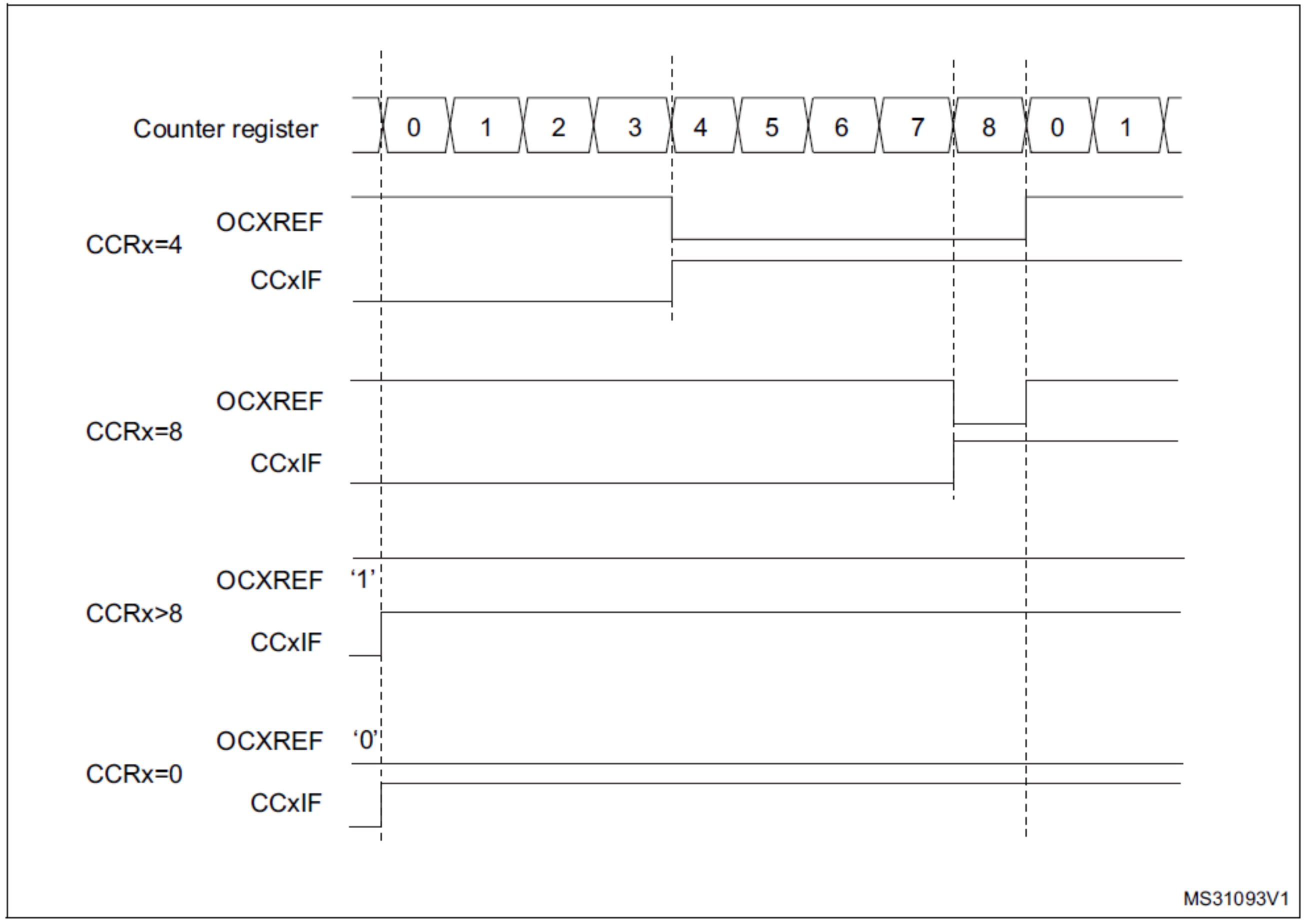
Capture/Compare Channel



PWM mode

Pulse Width Modulation mode allows you to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

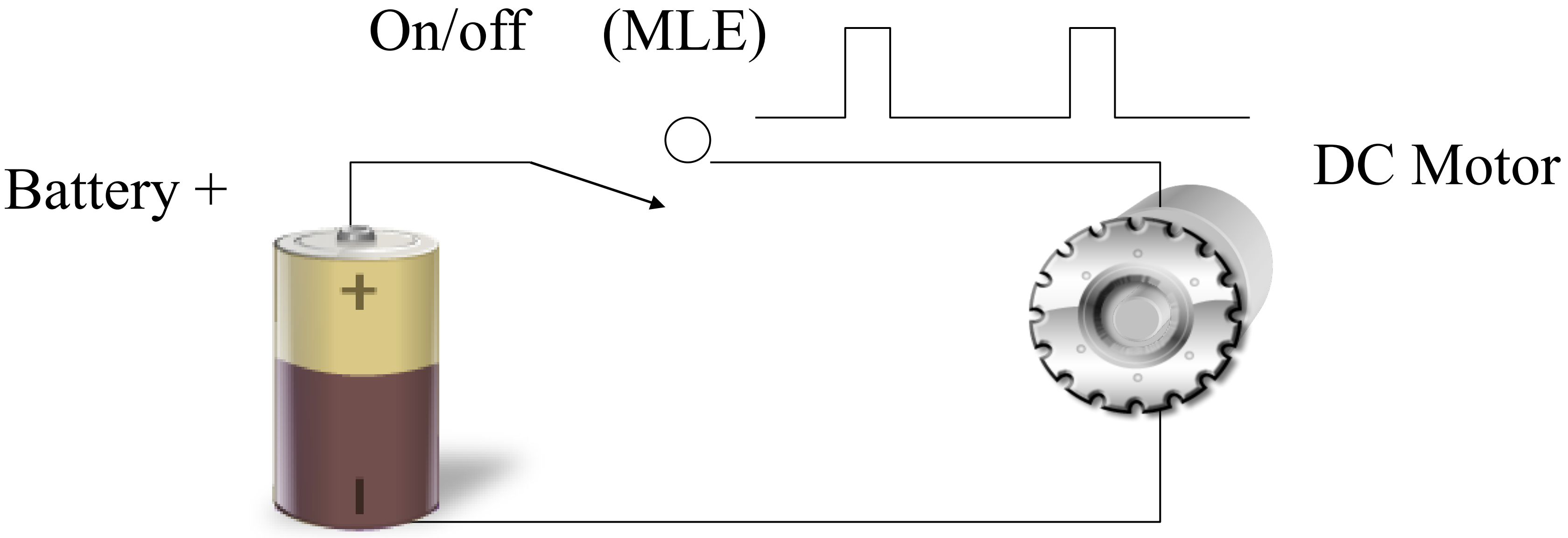
Figure 229. Edge-aligned PWM waveforms (ARR=8)



Pulse Width Modulation PWM unit

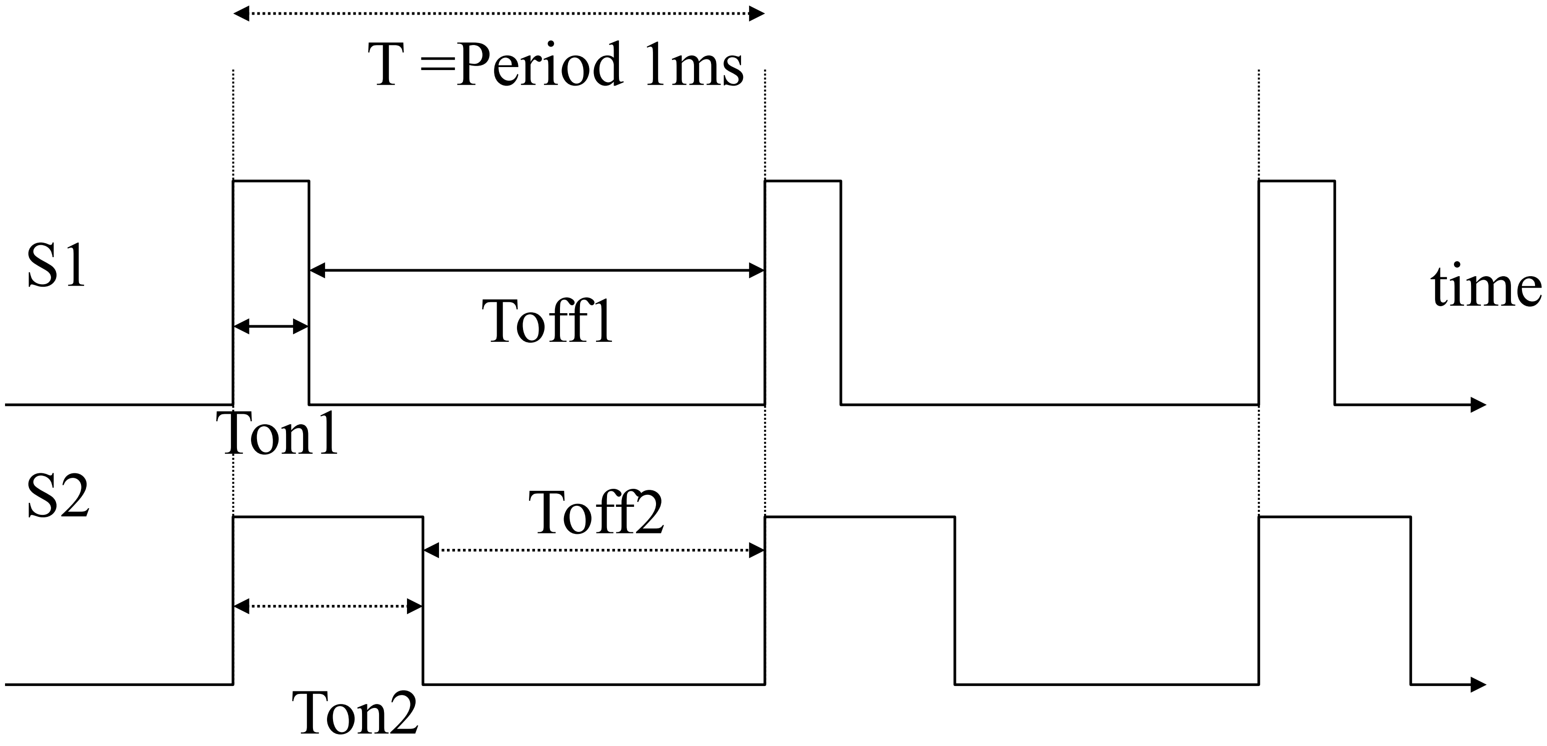
Use on-off time to control energy delivery

- The DC motor speed is determined by the on/off time of the motor enable signal MLE



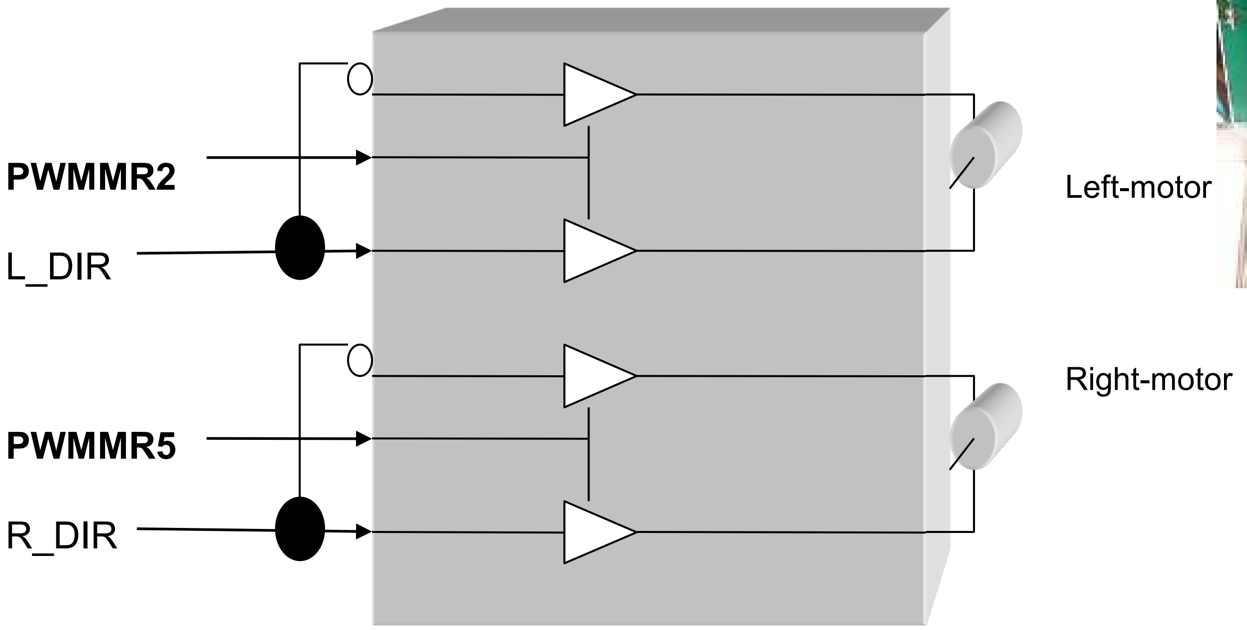
Timing diagrams of pulse width modulation

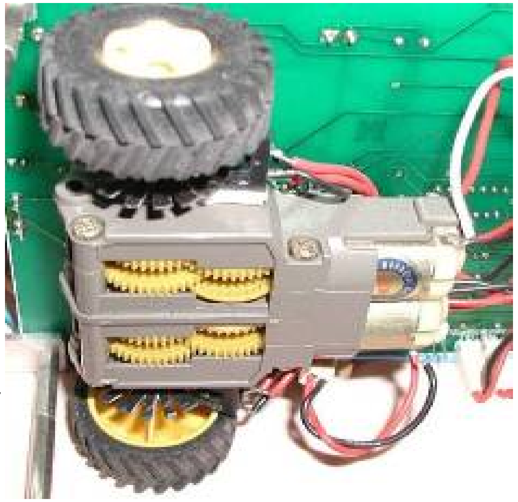
- Comparing two pulse modulated signals S1,S2



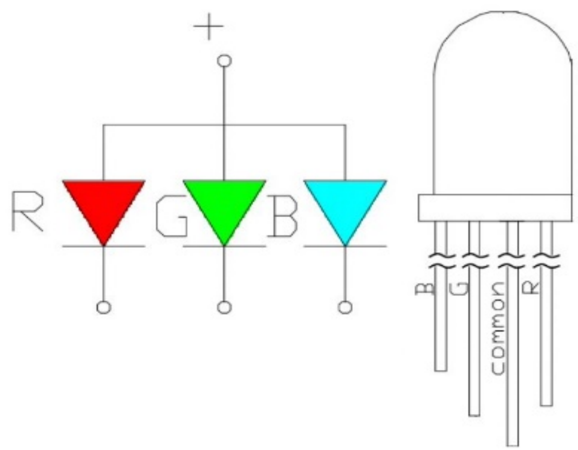
Use L293 H bridge circuit

- A chip for generating enough current to drive 2 motors controlled by 4 signals



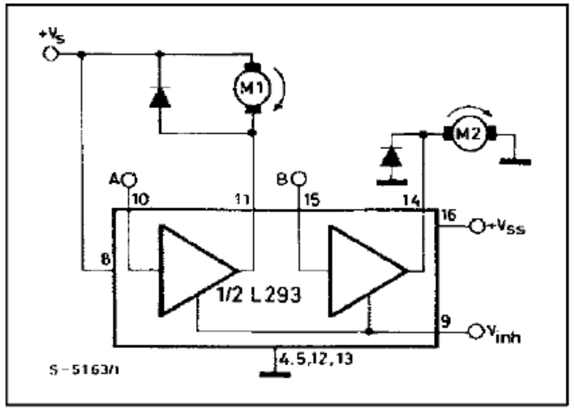


L293D & RGB LED



APPLICATION INFORMATION

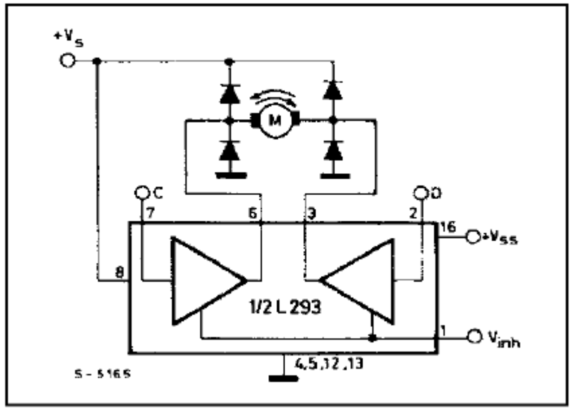
Figure 8 : DC Motor Controls (with connection to ground and to the supply voltage)



Vinh	A	M1	B	M2
H	H	Fast Motor Stop	H	Run
H	L	Run	L	Fast Motor Stop
L	X	Free Running Motor Stop	X	Free Running Motor Stop

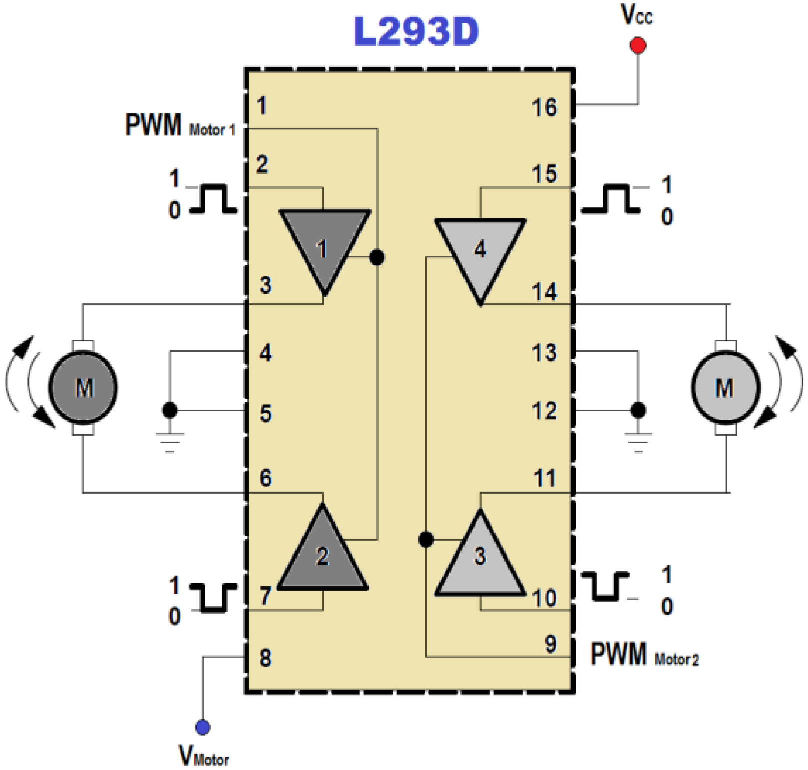
L = Low H = High X = Don't Care

Figure 9 : Bidirectional DC Motor Control



Inputs	Function	
Vinh = H	C = H ; D = L	Turn Right
	C = L ; D = H	Turn Left
	C = D	Fast Motor Stop
Vinh = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't Care



Timer Pin Definitions

Pin number					Pin name (function after reset)(1)	Pin type	I/O structure	Notes	Alternate functions	Additional functions
UQFN48	LQFP64	WLCSP49	LQFP100	UFBGA100L						
10	14	F6	23	L2	PA0-WKUP	I/O	TC	(5)	TIM2_CH1/TIM2_ET, TIM5_CH1, USART2_CTS, EVENTOUT	ADC1_0, WKUP1
11	15	G7	24	M2	PA1	I/O	FT	-	TIM2_CH2, TIM5_CH2, SPI4_MOSI/I2S4_SD, USART2_RTS, EVENTOUT	ADC1_1
12	16	E5	25	K3	PA2	I/O	FT	-	TIM2_CH3, TIM5_CH3, TIM9_CH1, I2S2_CKIN, USART2_TX, EVENTOUT	ADC1_2

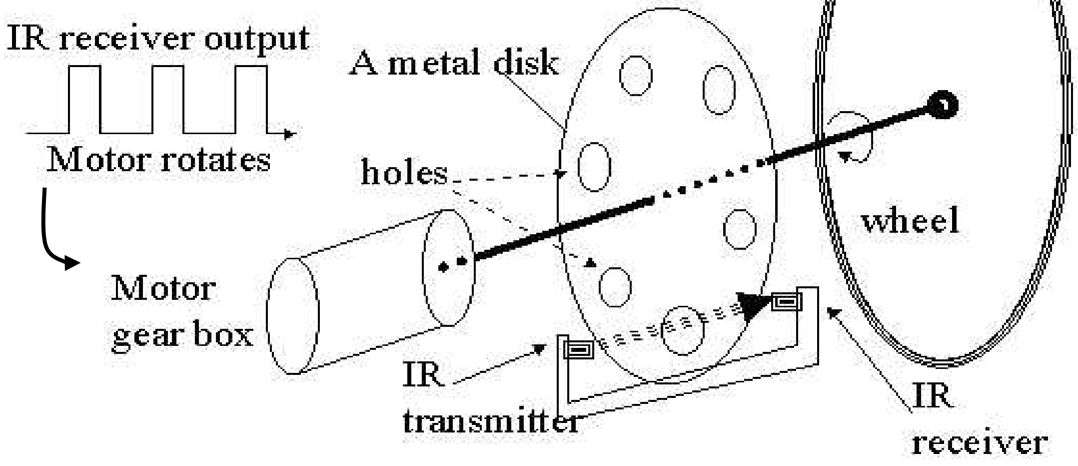
Alternate Function Mapping

Port	AF00	AF01	AF02	AF03	AF04	AF05	AF06	AF07	AF08	AF09	AF10	AF11	AF12	AF13	AF14	AF15
	SYS_AF	TIM1/TIM2	TIM3/ TIM4/ TIM5	TIM9/ TIM10/ TIM11	I2C1/I2C2/ I2C3	SPI1/I2S1SPI2/ I2S2/SPI3/ I2S3	SPI2/I2S2/ SPI3/ I2S3/SPI4/ I2S4/SPI5/ I2S5	SPI3/I2S3/ USART1/ USART2	USART6	I2C2/ I2C3	OTG1_FS		SDIO			
Port A PA0	-	TIM2_CH1/ TIM2_ETR	TIM5_CH1	-	-	-	-	USART2_CTS	-	-	-	-	-	-	-	EVENTOUT
Port A PA1	-	TIM2_CH2	TIM5_CH2	-	-	SPI4_MOSI /I2S4_SD	-	USART2_RTS	-	-	-	-	-	-	-	EVENTOUT
Port A PA2	-	TIM2_CH3	TIM5_CH3	TIM9_CH1	-	I2S2_CKIN	-	USART2_TX	-	-	-	-	-	-	-	EVENTOUT
Port A PA3	-	TIM2_CH4	TIM5_CH4	TIM9_CH2	-	I2S2_MCK	-	USART2_RX	-	-	-	-	-	-	-	EVENTOUT
Port A PA4	-	-	-	-	-	SPI1_NSS/I2S1_WS	SPI3_NSS/I2S3_WS	USART2_CK	-	-	-	-	-	-	-	EVENTOUT
Port A PA5	-	TIM2_CH1/ TIM2_ETR	-	-	-	SPI1_SCK/I2S1_CK	-	-	-	-	-	-	-	-	-	EVENTOUT
Port A PA6	-	TIM1_BKIN	TIM3_CH1	-	-	SPI1_MISO	I2S2_MCK	-	-	-	-	-	SDIO_CMD	-	-	EVENTOUT
Port A PA7	-	TIM1_CH1N	TIM3_CH2	-	-	SPI1_MOSI /I2S1_SD	-	-	-	-	-	-	-	-	-	EVENTOUT
Port A PA8	MCO_1	TIM1_CH1	-	-	I2C3_SCL	-	-	USART1_CK	-	-	USB_FS_SOF	-	SDIO_D1	-	-	EVENTOUT
Port A PA9	-	TIM1_CH2	-	-	I2C3_SMBA	-	-	USART1_TX	-	-	USB_FS_VBUS	-	SDIO_D2	-	-	EVENTOUT
Port A PA10	-	TIM1_CH3	-	-	-	-	SPI5_MOSI/I2S5_SD	USART1_RX	-	-	USB_FS_ID	-	-	-	-	EVENTOUT
Port A PA11	-	TIM1_CH4	-	-	-	-	SPI4_MISO	USART1_CTS	USART6_TX	-	USB_FS_DM	-	-	-	-	EVENTOUT
Port A PA12	-	TIM1_ETR	-	-	-	-	SPI5_MISO	USART1_RTS	USART6_RX	-	USB_FS_DP	-	-	-	-	EVENTOUT

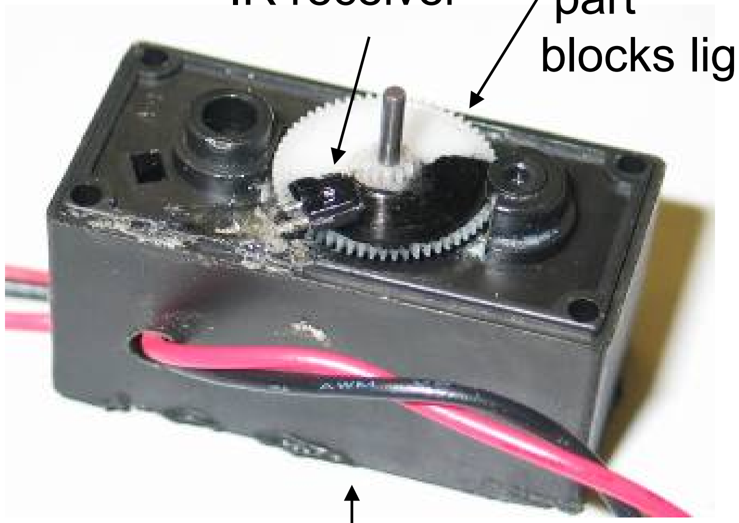
Left Wheel sensor – LWheelsen
(same for Right wheel sensor RWheelsen)

Our motor and
speed encoder

- The disk chops the IR light
- on and off as the wheel rotates

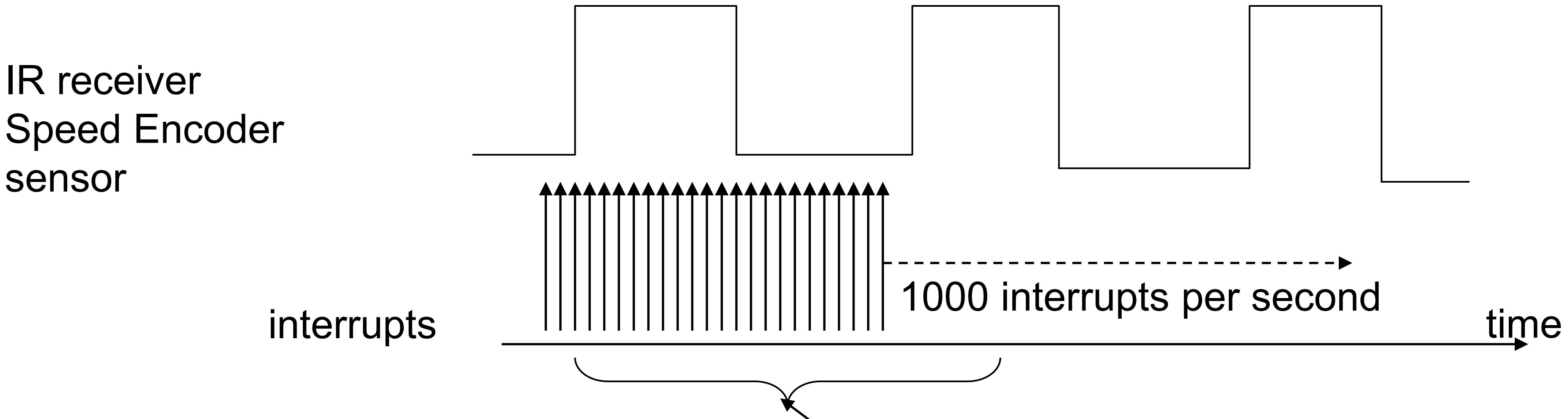


Each wheel rotation=
88 on/off changes



LWSensor

RWSensor



Read wheel count (lcount, rcount) using interrupts

```
Main( )
{
Setup( );
:
:
}
```

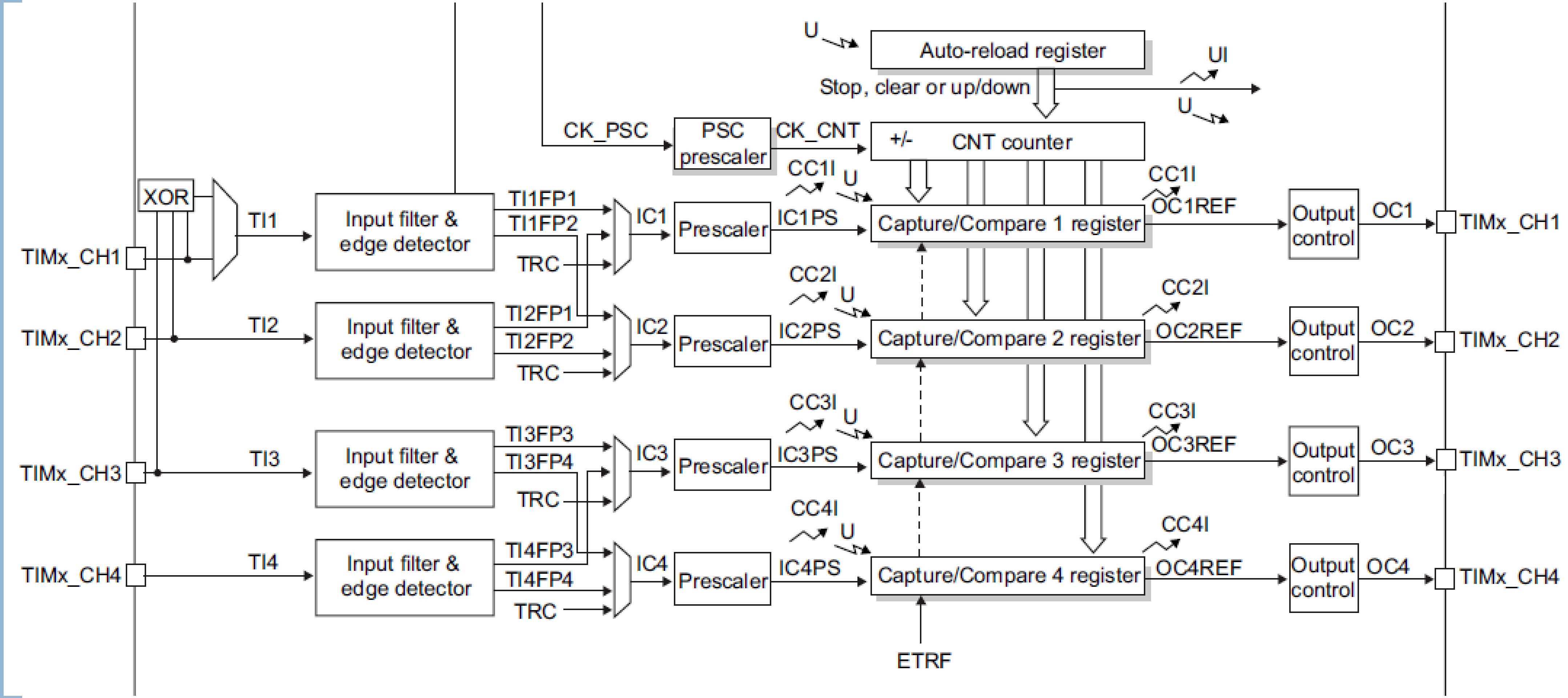
```
_IRQ exception() //1000Hz
{
:
read wheel speed
Update rcount
Update lcount
:
}
```

TIM2 to TIM5 main features

General-purpose TIMx timer features include:

- 16-bit (TIM3 and TIM4) or 32-bit (TIM2 and TIM5) up, down, up/down auto-reload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65536.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge- and Center-aligned modes)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

TIM2-TIM5 Block Diagram



TIM2-TIM5 Time-base unit

Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related auto-reload register. The counter can count up, down or both up and down but also down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

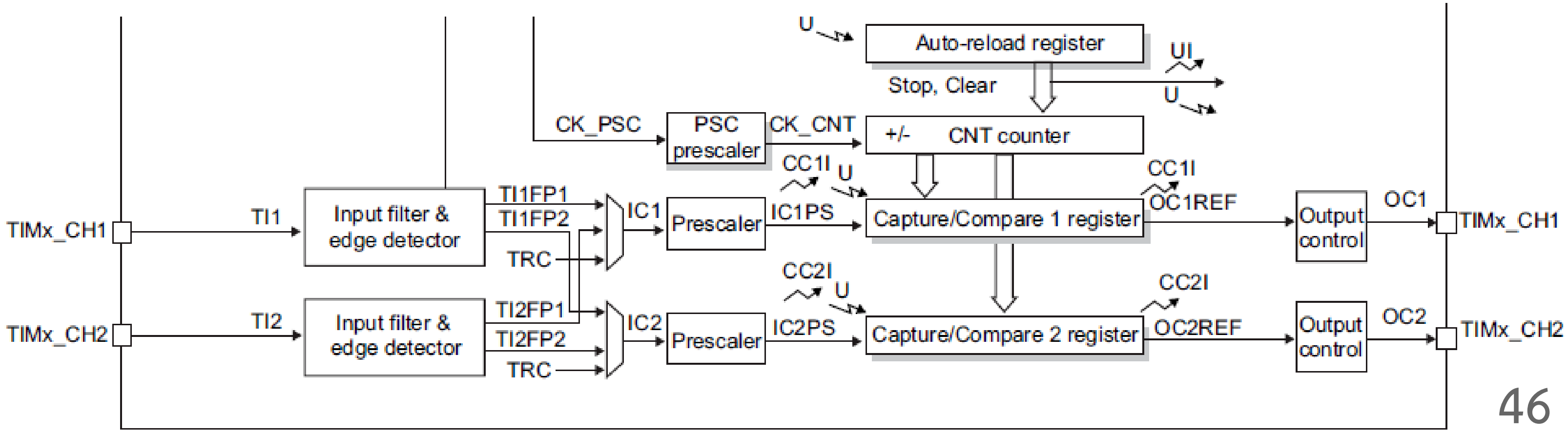
The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC):
- Auto-Reload Register (TIMx_ARR)

TIM9/TIM12 main features

The features of the TIM9/TIM12 general-purpose timers include:

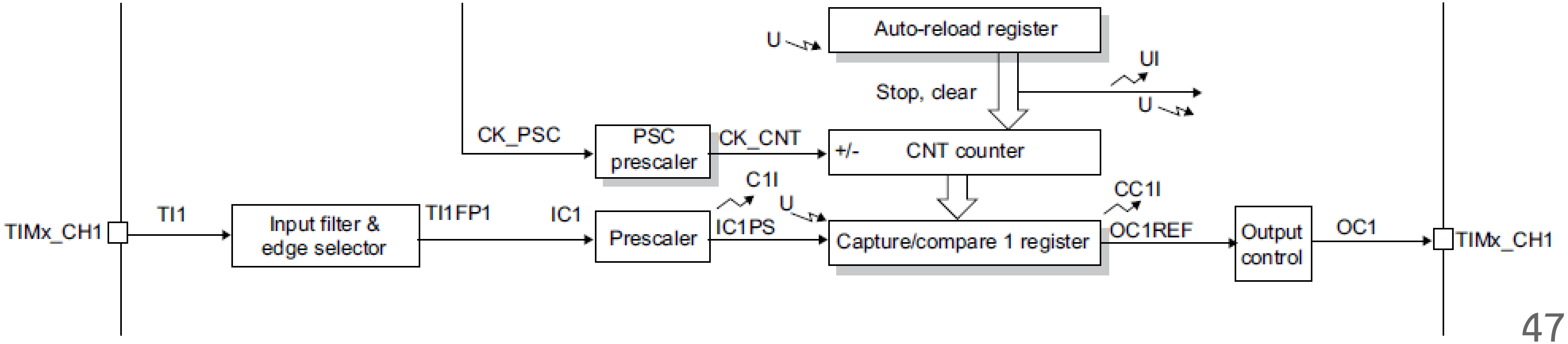
- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- Up to 2 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal trigger)
  - Trigger event (counter start, stop, initialization or count by internal trigger)
  - Input capture
  - Output compare



TIM10/TIM11/TIM13/TIM14 main features

The features of general-purpose timers TIM10/TIM11/TIM13/TIM14 include:

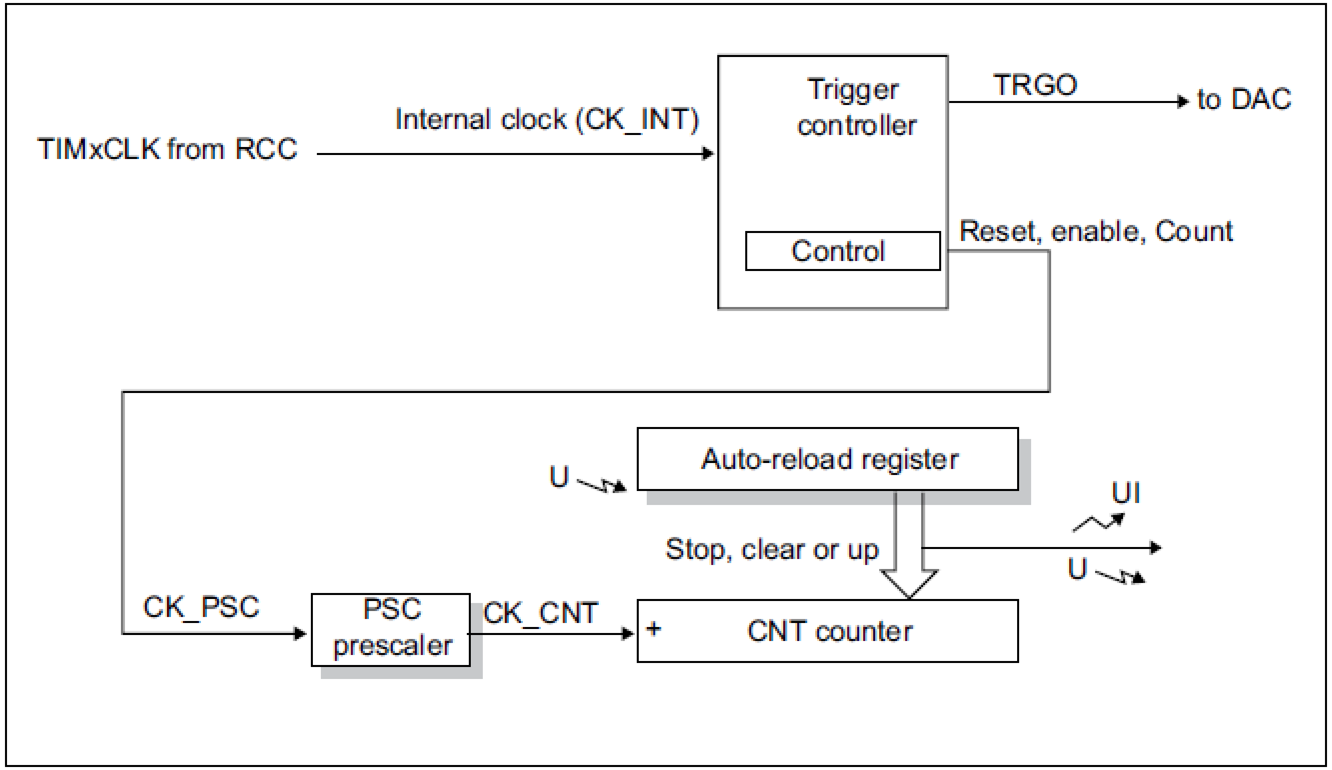
- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- independent channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software)
  - Input capture
  - Output compare



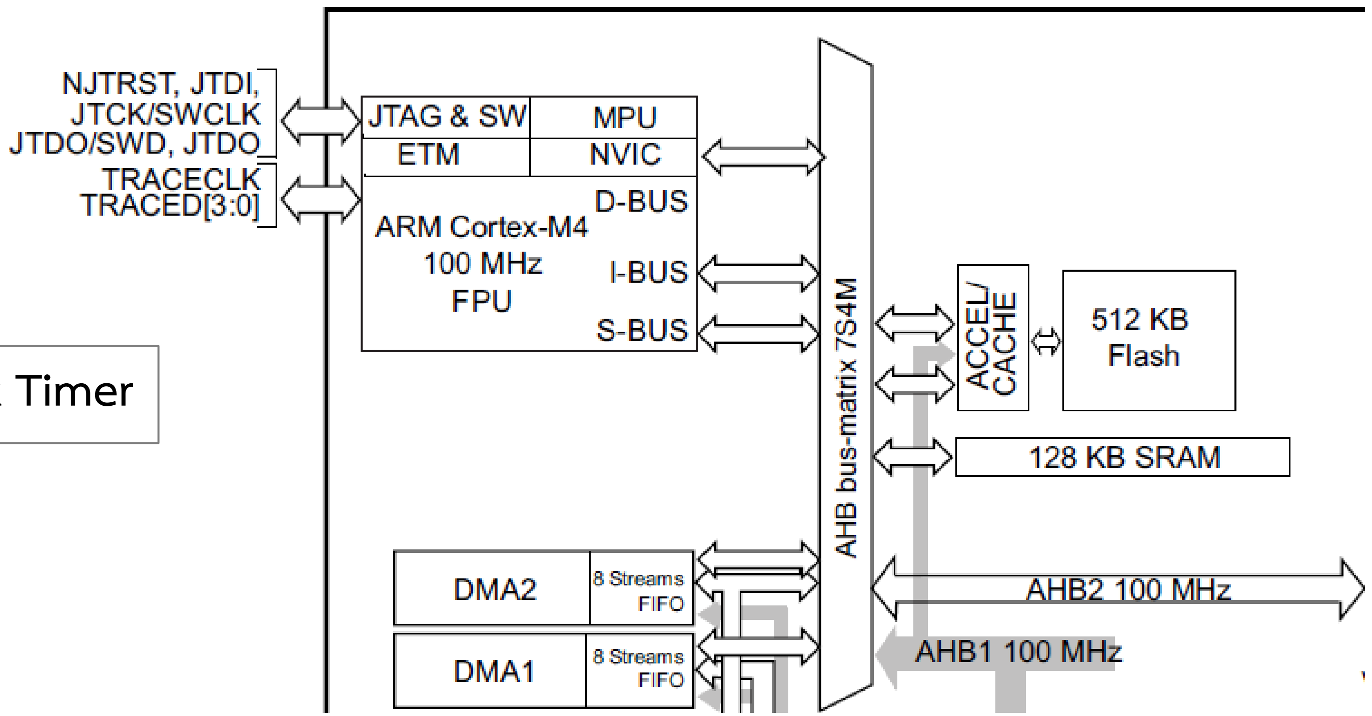
TIM6/TIM7 main features

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



SysTick timer



HAL_Delay function uses Systick Timer

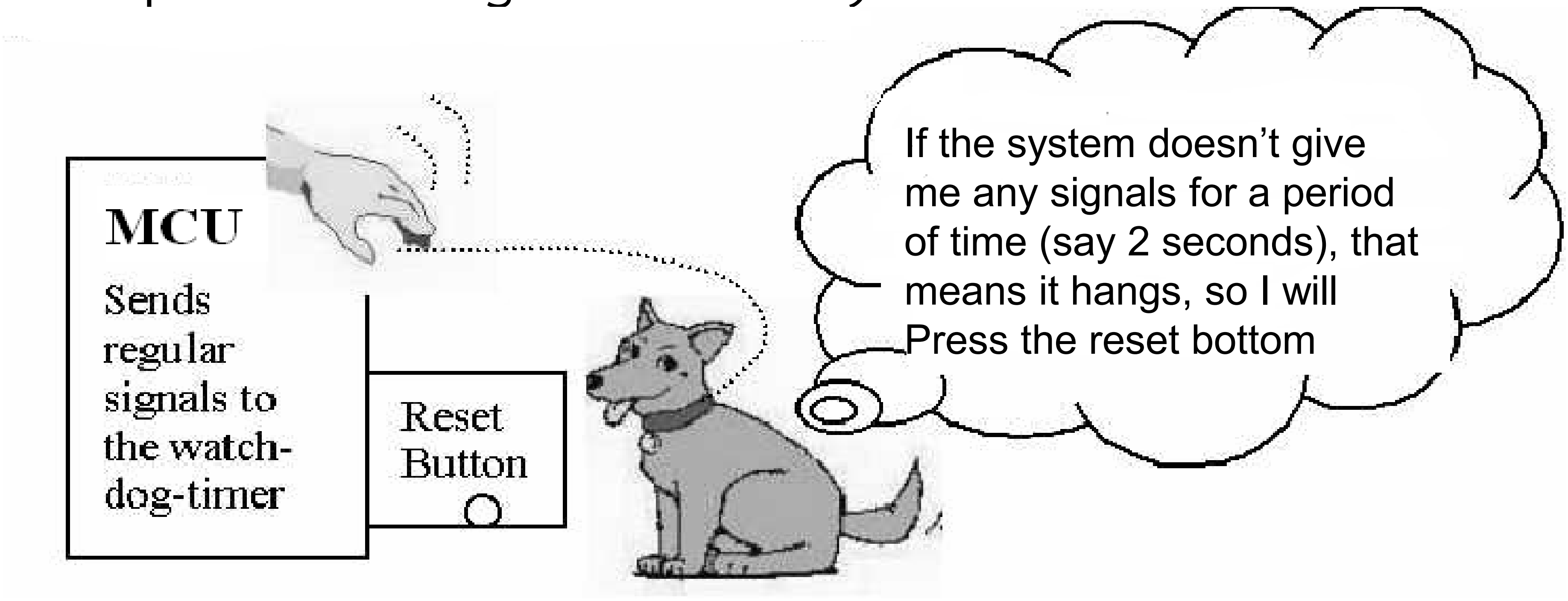
SysTick timer

This timer is dedicated to real-time operating systems, but could also be used as a standard downcounter. It features:

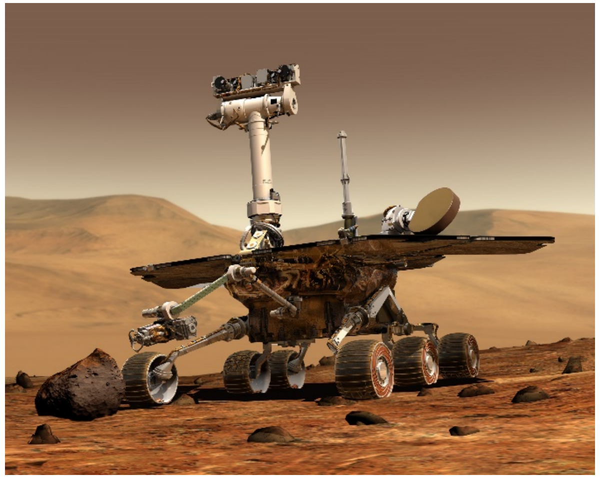
- A 24-bit downcounter
- Autoreload capability
- Maskable system interrupt generation when the counter reaches 0
- Programmable clock source

Watchdog timer

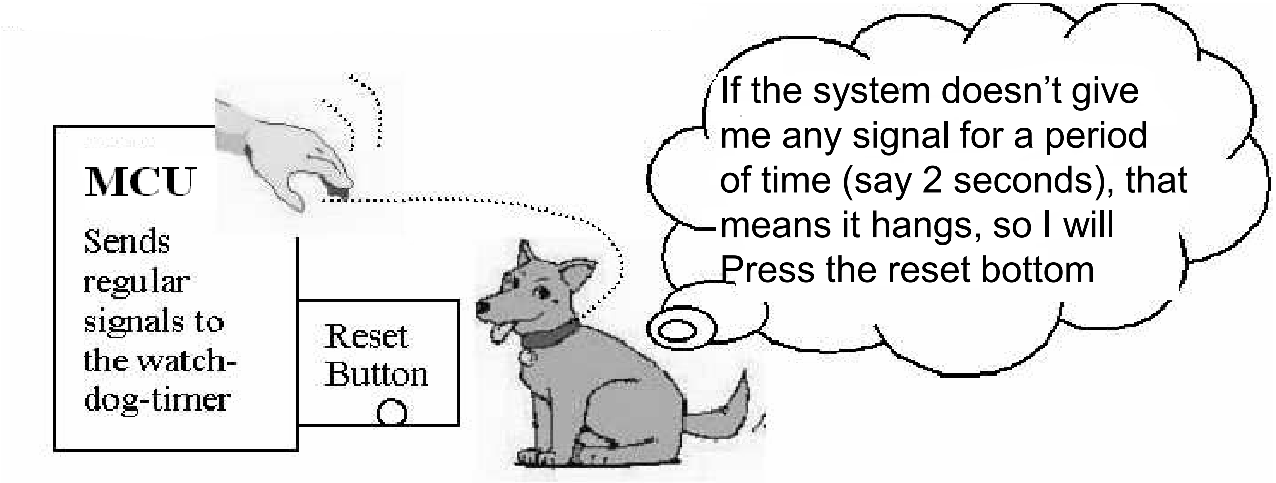
- For implementing fail safe systems



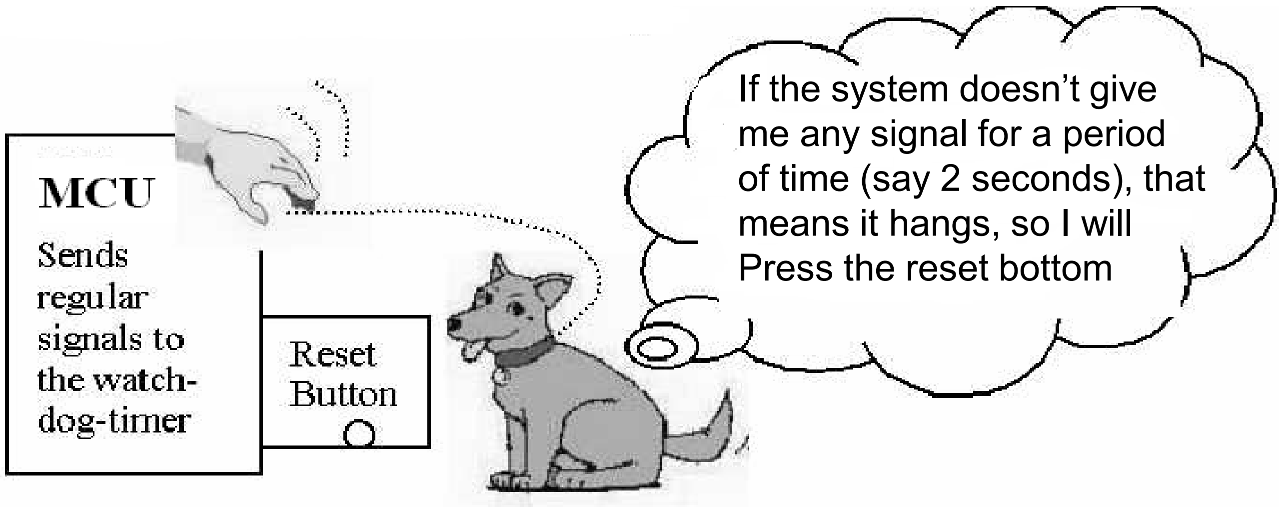
Example, solar power wireless telephone (register setting , see appendix)



- At remote area, maintenance is difficult
- If the software does not operate properly (hangs)
  - That means it sends no regular signals to the watch dog sensor
- Then
  - the watch-dog resets the system



Software



- Main

```
{
    While(1)
    {  Do_the _neccessary();
       Send_a_pulse_to_watch_dog();
    }
}
```

- If the software hangs, it will not **Send_a_pulse_to_watch_dog()**;
- so the system is reset by the watch_dog_hardware

STM32F767 Watchdog Timer

Independent watchdog

The independent watchdog is based on a 12-bit downcounter and 8-bit prescaler. It is clocked from an independent 32 kHz internal RC and as it operates independently from the main clock, it can operate in Stop and Standby modes. It can be used either as a watchdog to reset the device when a problem occurs, or as a free-running timer for application timeout management. It is hardware- or software-configurable through the option bytes.

Window watchdog

The window watchdog is based on a 7-bit downcounter that can be set as free-running. It can be used as a watchdog to reset the device when a problem occurs. It is clocked from the main clock. It has an early warning interrupt capability and the counter can be frozen in debug mode.

STM32F429 Watchdog Timer

21 Independent watchdog (IWDG)

This section applies to the whole STM32F4xx family, unless otherwise specified.

21.1 IWDG introduction

The devices have two embedded watchdog peripherals which offer a combination of high safety level, timing accuracy and flexibility of use. Both watchdog peripherals (Independent and Window) serve to detect and resolve malfunctions due to software failure, and to trigger system reset or an interrupt (window watchdog only) when the counter reaches a given timeout value.

The independent watchdog (IWDG) is clocked by its own dedicated low-speed clock (LSI) and thus stays active even if the main clock fails. The window watchdog (WWDG) clock is prescaled from the APB1 clock and has a configurable time-window that can be programmed to detect abnormally late or early application behavior.

The IWDG is best suited to applications which require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints. The WWDG is best suited to applications which require the watchdog to react within an accurate timing window. For further information on the window watchdog, refer to *Section 22 on page 713*.

STM32F429 Watchdog Timer

21.2 IWDG main features

- Free-running downcounter
- clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Reset (if watchdog activated) when the downcounter value of 0x000 is reached

21.3 IWDG functional description

Figure 213 shows the functional blocks of the independent watchdog module.

When the independent watchdog is started by writing the value 0xCCCC in the Key register (IWDG_KR), the counter starts counting down from the reset value of 0xFFF. When it reaches the end of count value (0x000) a reset signal is generated (IWDG reset).

Whenever the key value 0xAAAA is written in the IWDG_KR register, the IWDG_RLR value is reloaded in the counter and the watchdog reset is prevented.

STM32F429 Watchdog Timer

22 Window watchdog (WWDG)

22.1 WWDG introduction

The window watchdog is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the contents of the downcounter before the T6 bit becomes cleared. An MCU reset is also generated if the 7-bit downcounter value (in the control register) is refreshed before the downcounter has reached the window register value. This implies that the counter must be refreshed in a limited window.

Independent Watchdog

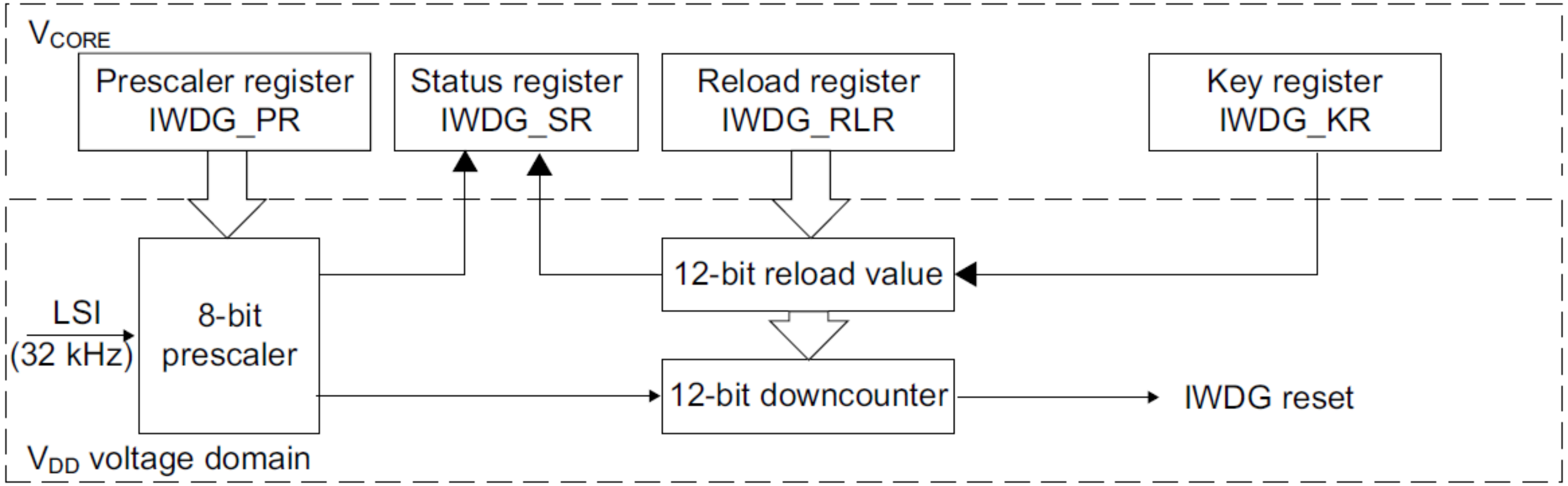


Table 61. Min/max IWDG timeout period at 32 kHz (LSI)(1)

Prescaler divider	PR[2:0] bits	Min timeout (ms) RL[11:0]= 0x000	Max timeout (ms) RL[11:0]= 0xFFF
/4	0	0.125	512
/8	1	0.25	1024
/16	2	0.5	2048
/32	3	1	4096
/64	4	2	8192
/128	5	4	16384
/256	6		32768

Start and Reset Independent Watch

15.4.1 Key register (IWDG_KR)

Address offset: 0x00

Reset value: 0x0000 0000 (reset by Standby mode)

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

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

Bits 15:0 **KEY[15:0]:** Key value (write only, read 0000h)

These bits must be written by software at regular intervals with the key value AAAAh, otherwise the watchdog generates a reset when the counter reaches 0.

Writing the key value 5555h to enable access to the IWDG_PR and IWDG_RLR registers (see *Section 15.3.2*)

Writing the key value CCCCh starts the watchdog (except if the hardware watchdog option is selected)

Summary

- Need to know clock speed and bit length for timer to operate correctly.
- Overflow/Underflow event can generate timing interrupt.
- Timers may connect to different APB buses and have different bit length.
- PWM signal generated from timer can drive LED and help drive motor.
- System timer is close to processor core.