# Timer & Pulse-Width Modulation

Microcontroller Application and Development

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.
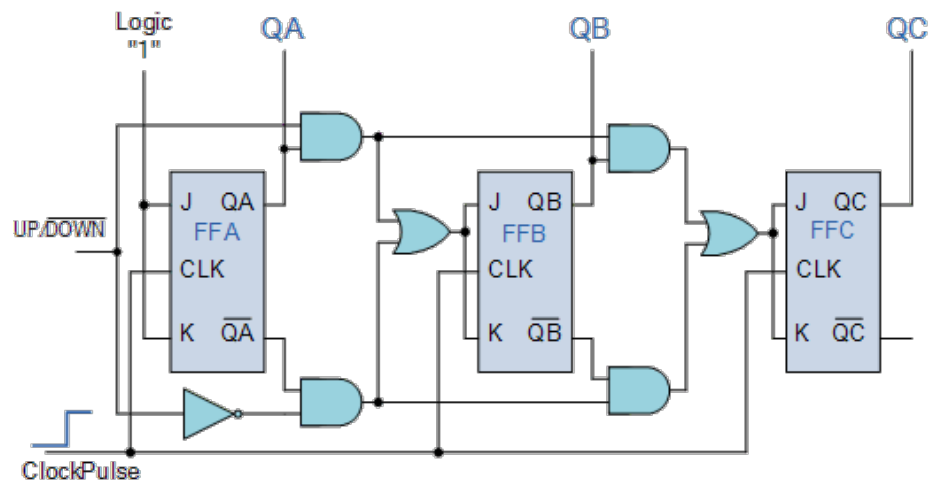
- Free running timer.

# The Digital Counter

We can preload

Input

| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |

an interrupt at the moment of overflow

We can read
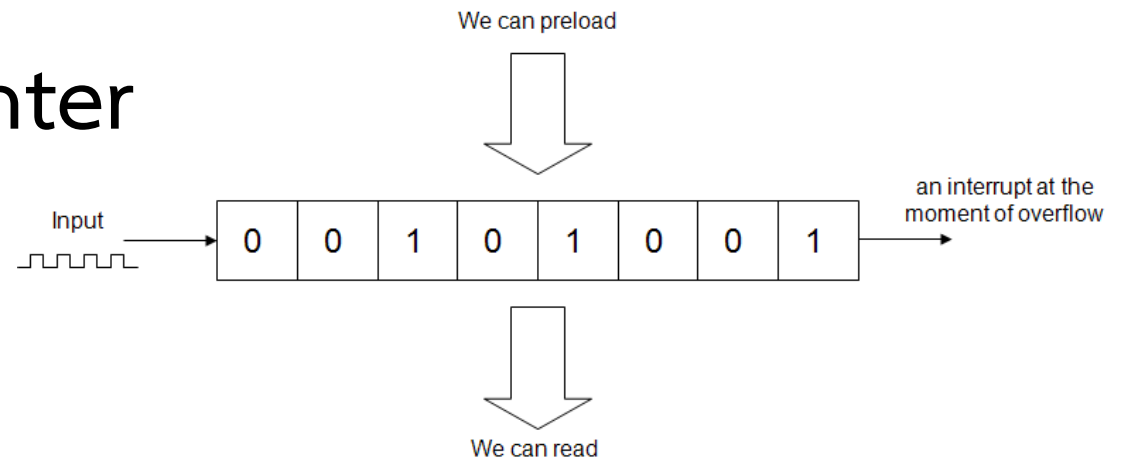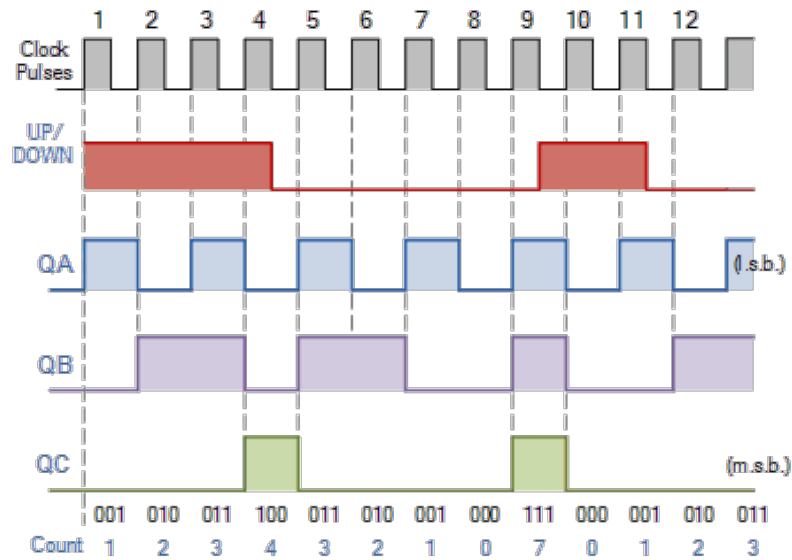
- n-bit counter can count from 0 to $(2^n - 1)$

- 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
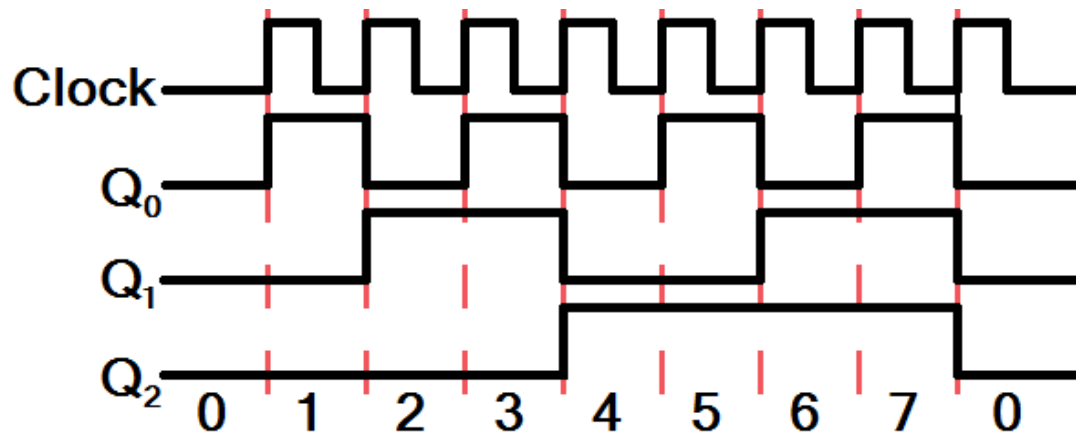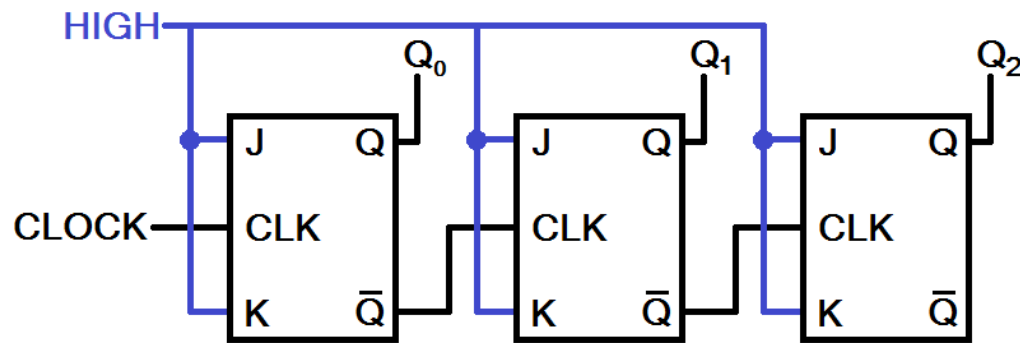
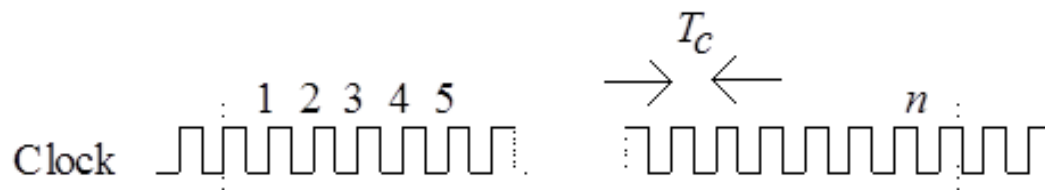# 3-bit Synchronous Up/Down Counter

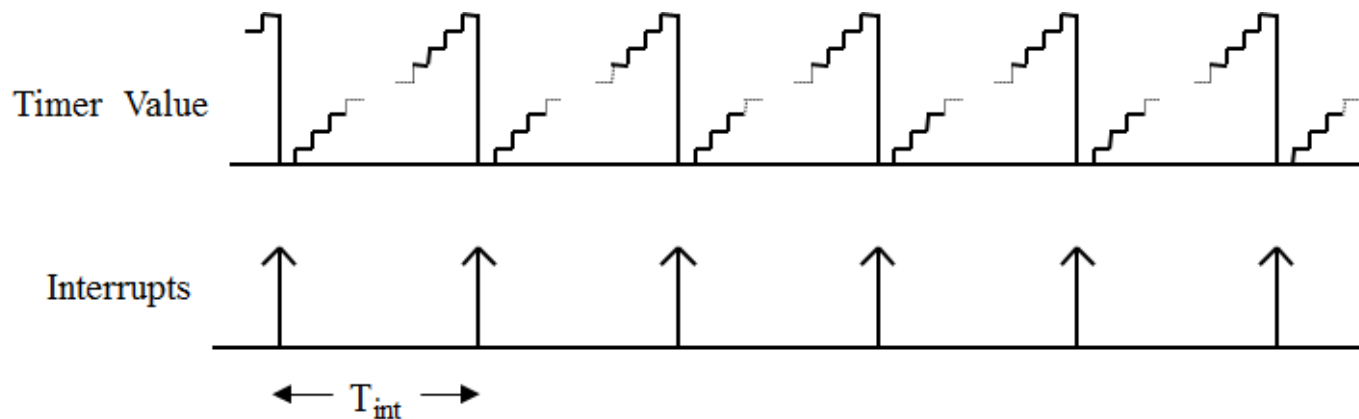# Counting and Timing

- 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 ($0 \rightarrow 255 \rightarrow 0$) can generate an interrupt signal.

- If a counter continues counting after an overflow, then interrupt signals will occur repeatedly.

# Timer Basic Block Diagram

clk → **Clock Division** → /1 /2 /4 → **Prescaler Counter 16 bit** → **Timer Counter 16/32 bit**

**Auto Reload Register** → **Timer Counter 16/32 bit**

**Timer Counter 16/32 bit** → **Capture/Compare Register** →
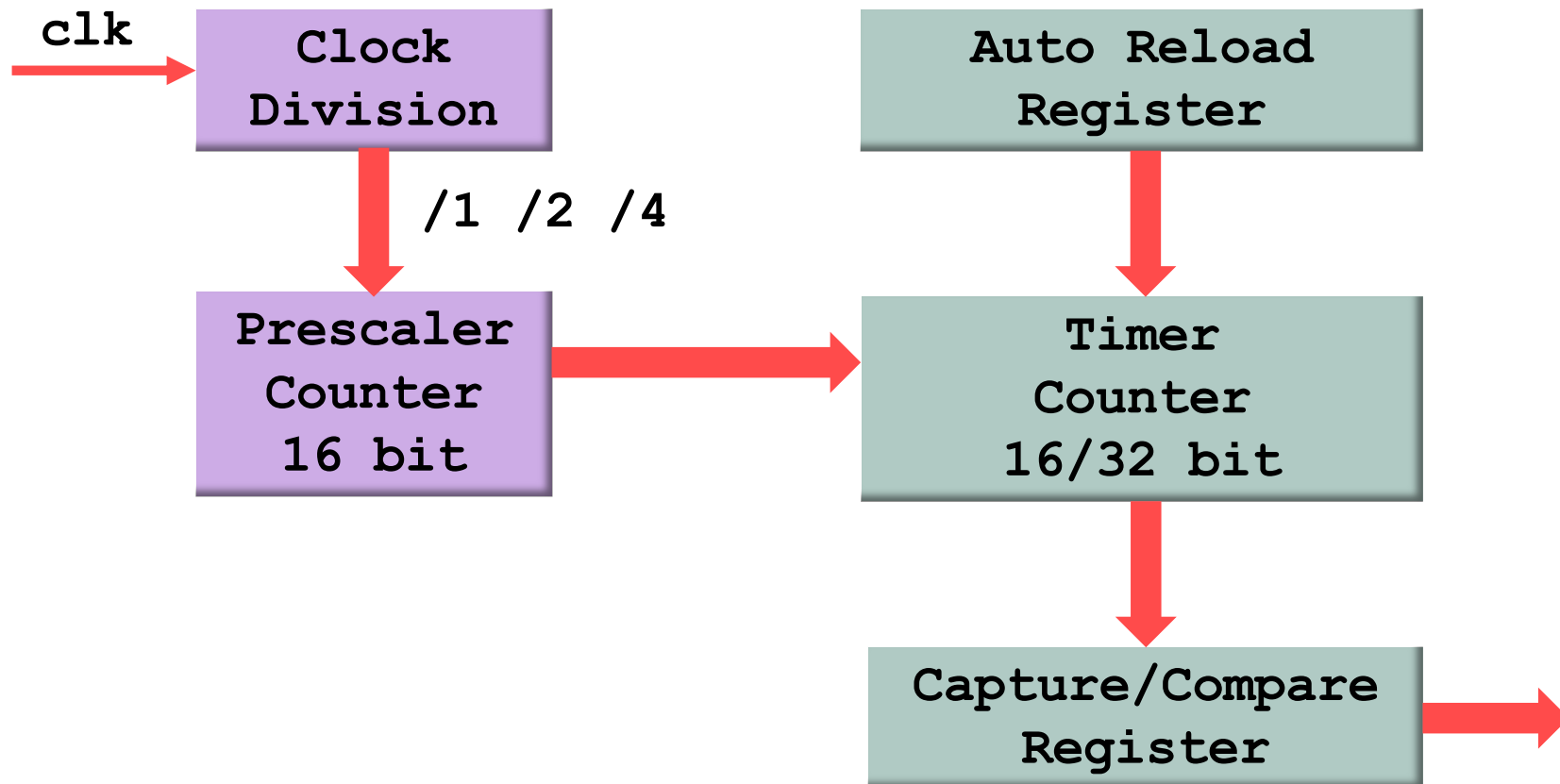
# Time Interval

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

<span style="color:red">1, 2, 4        16 bit     16/32 bit</span>

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

$$Time\ Interval = 1\ ms$$

# Timer Basic Block Diagram



Clk
216 MHz

Clock Division

/1

Prescaler Counter 16 bit

0-215

/216

Auto Reload Register

999

Timer Counter 16/32 bit

0-999

Capture/Compare Register

999

/1000

1 kHz
1 ms
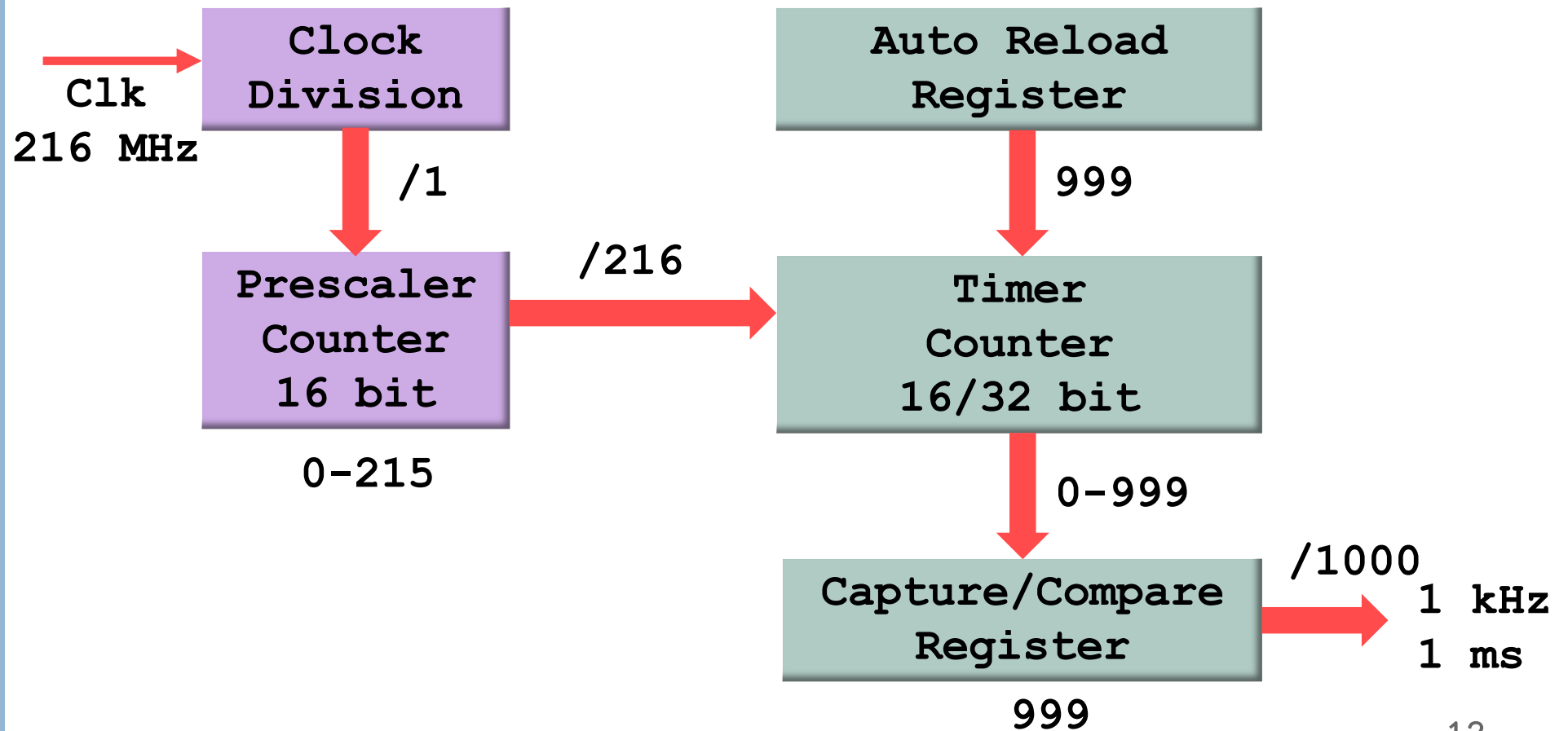
# TIMx and APB clock Speed

| 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
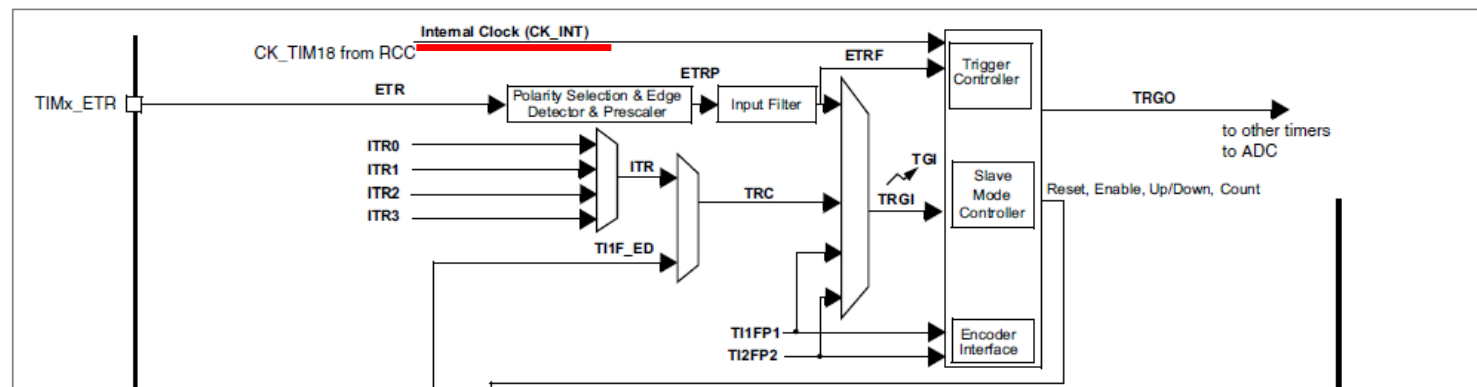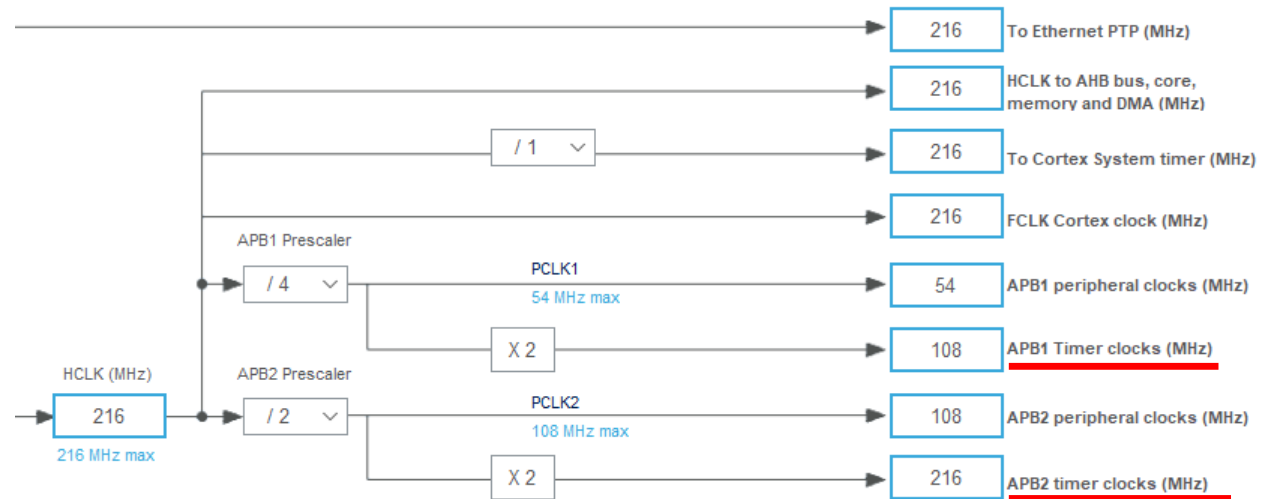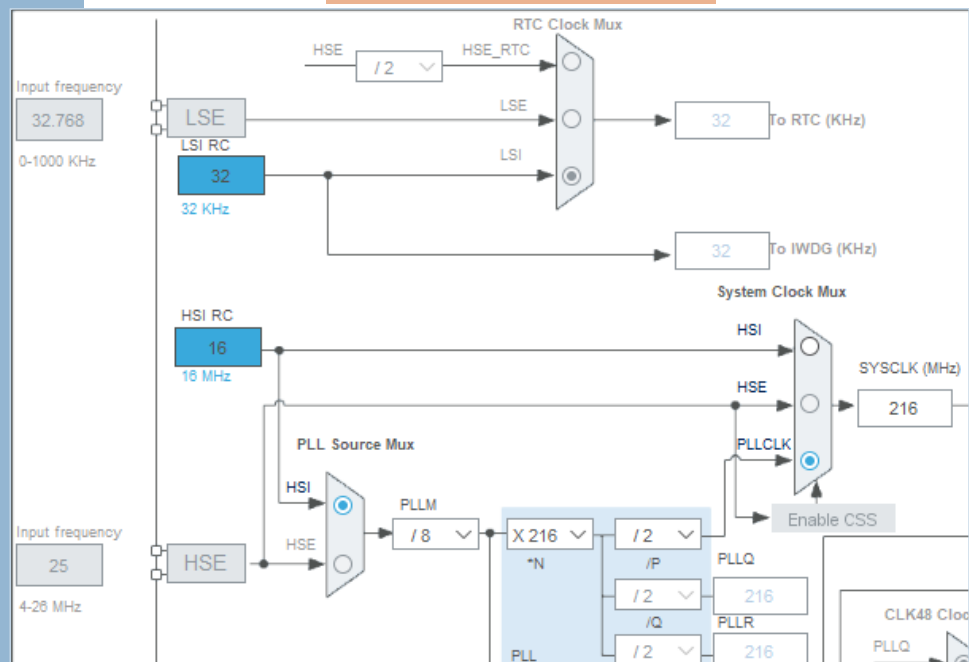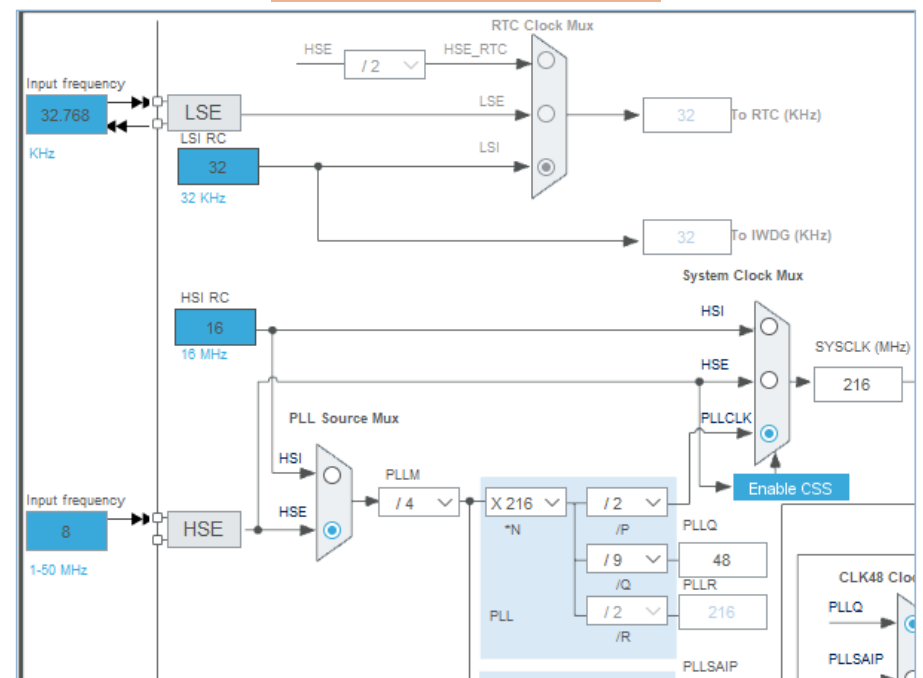
# Clock Sources



MCU Selector
STM32F767ZI

Board Selector
Nucleo-767ZI

15

| 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 |

# STM32F767 Timers

- 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 |

# 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
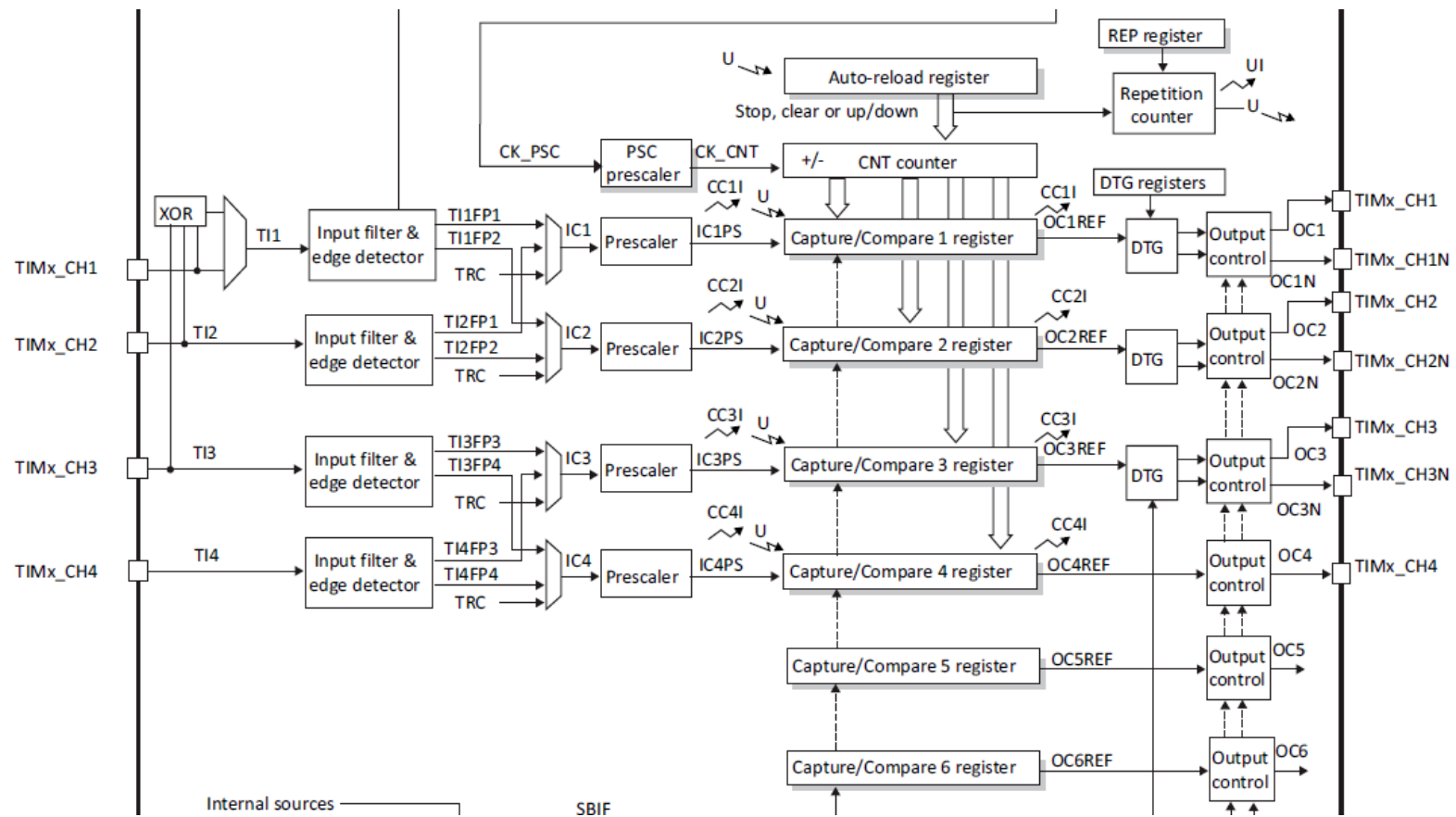
## Time-base unit

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.
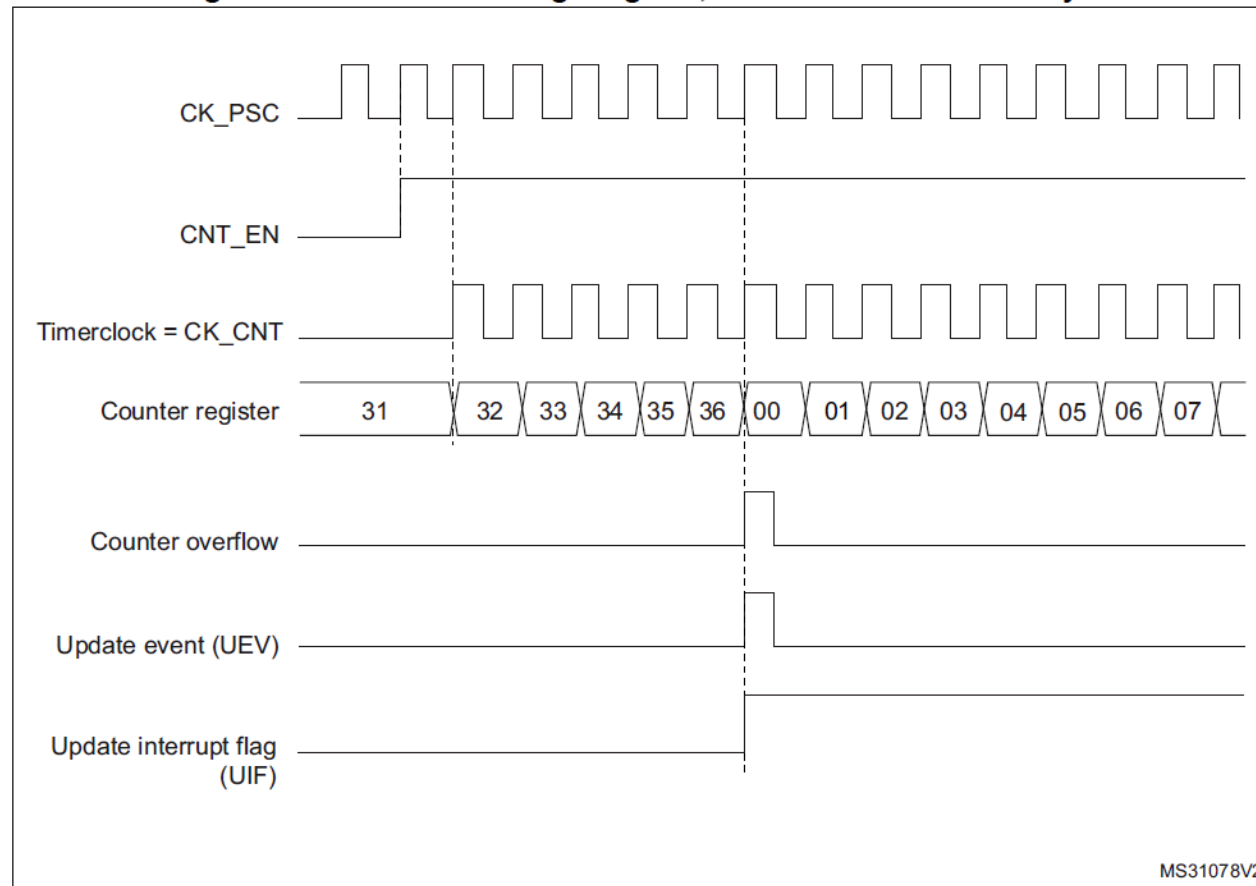
# Up Counting Mode

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.
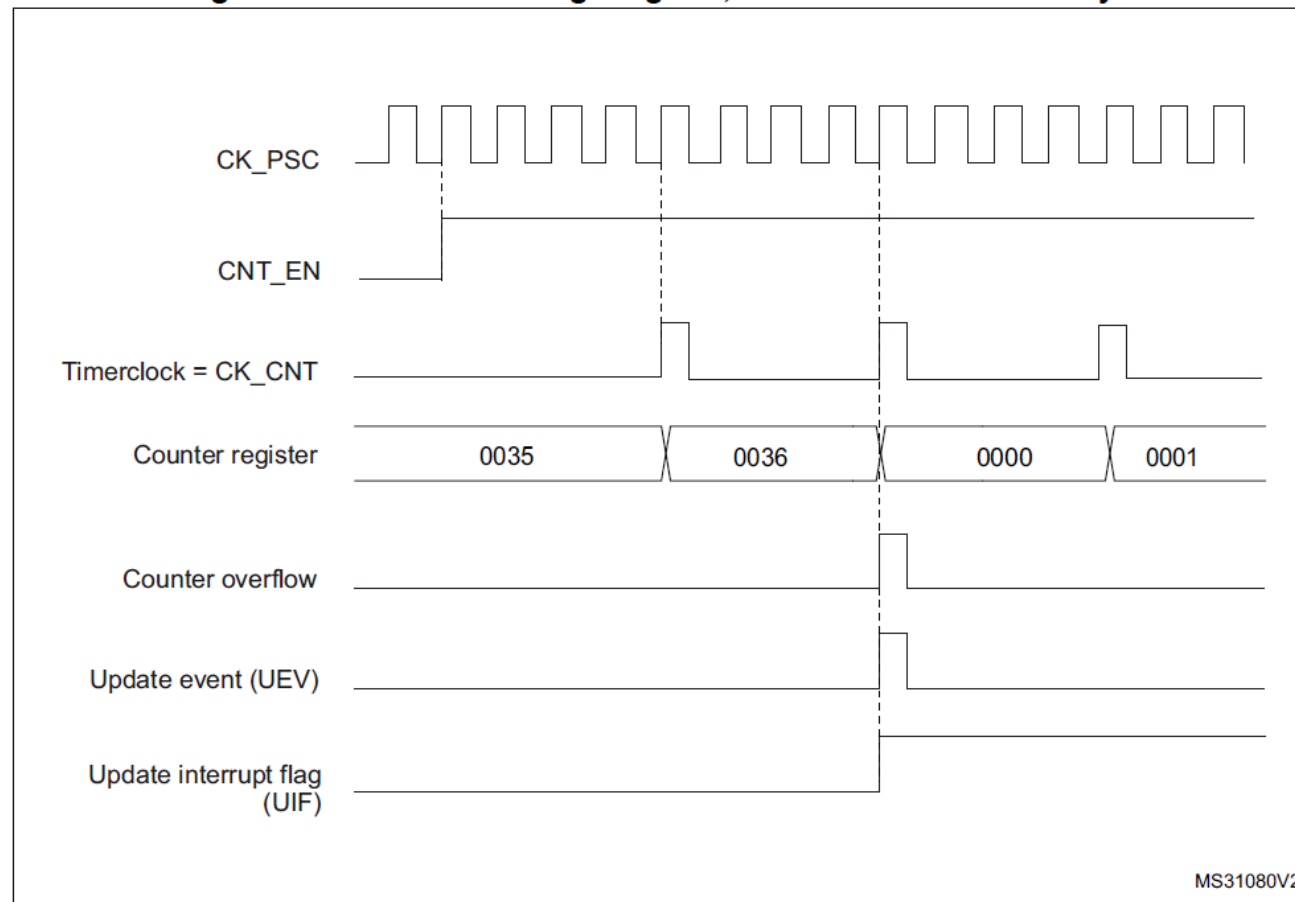
# Up counting Mode, TIMx_ARR = 0x36

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

24

# Up counting Mode, TIMx_ARR = 0x36

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



MS31080V2

# 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.
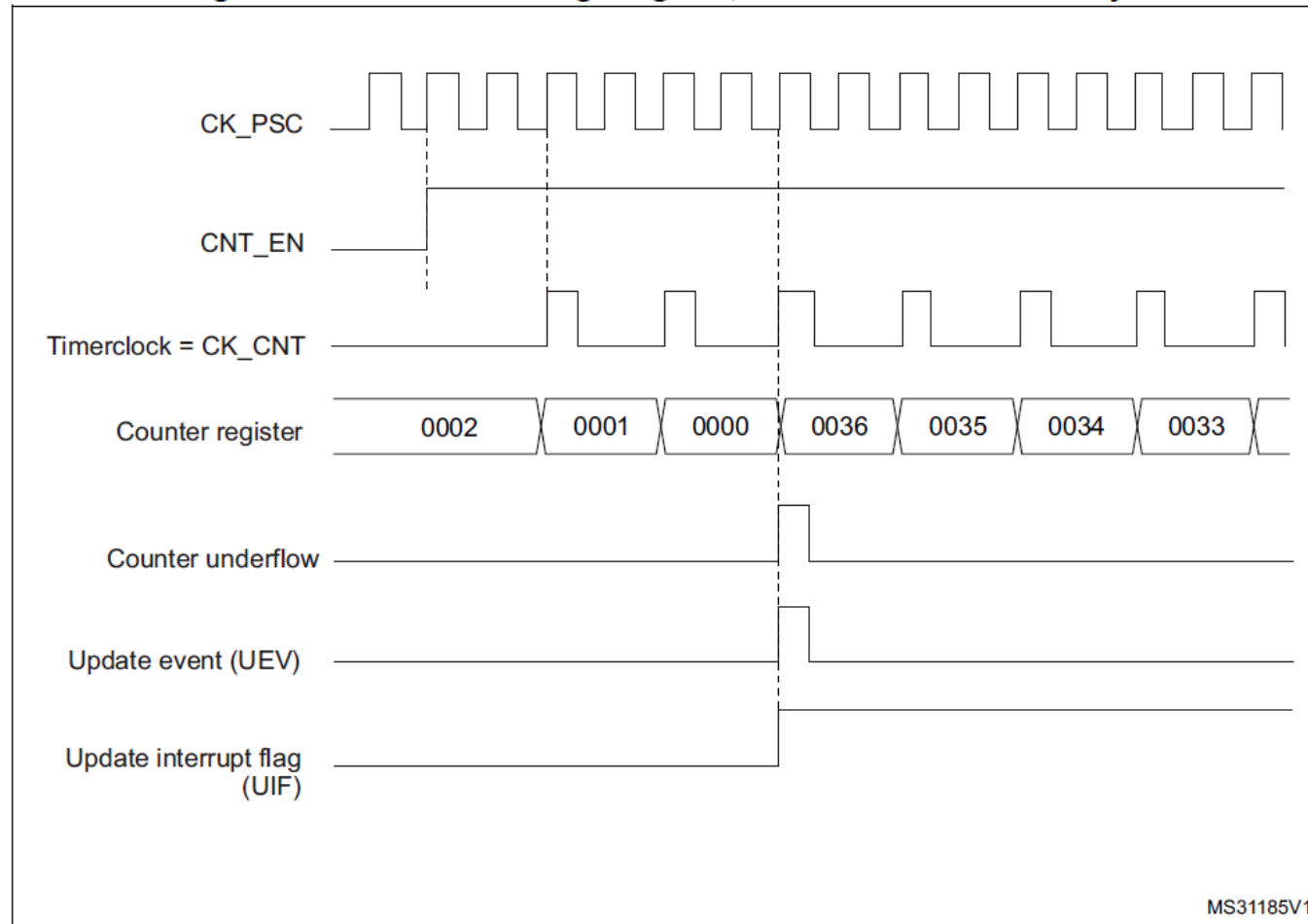
# Down Counting Mode, TIMx_ARR = 0x36



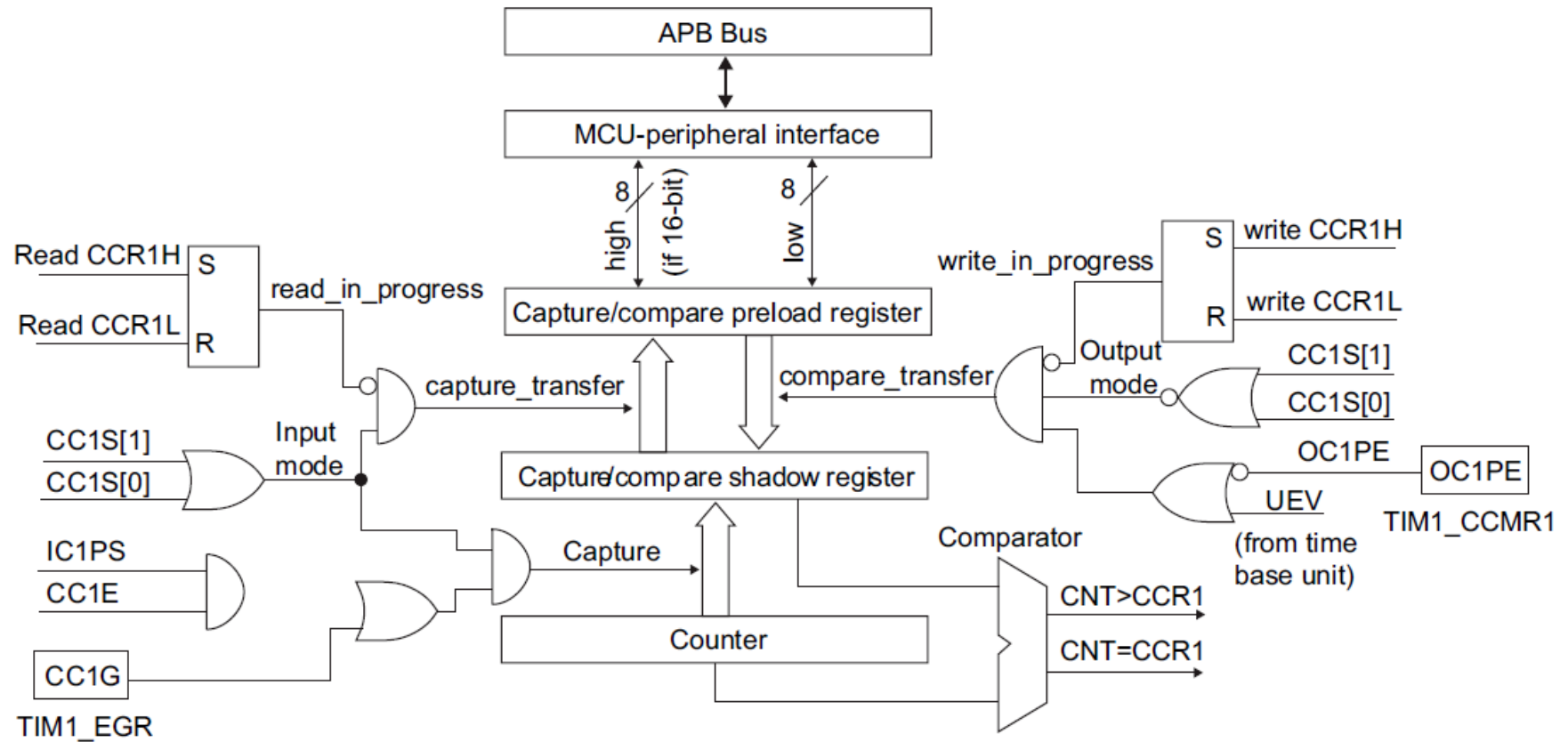Figure 204. Counter timing diagram, internal clock divided by 1
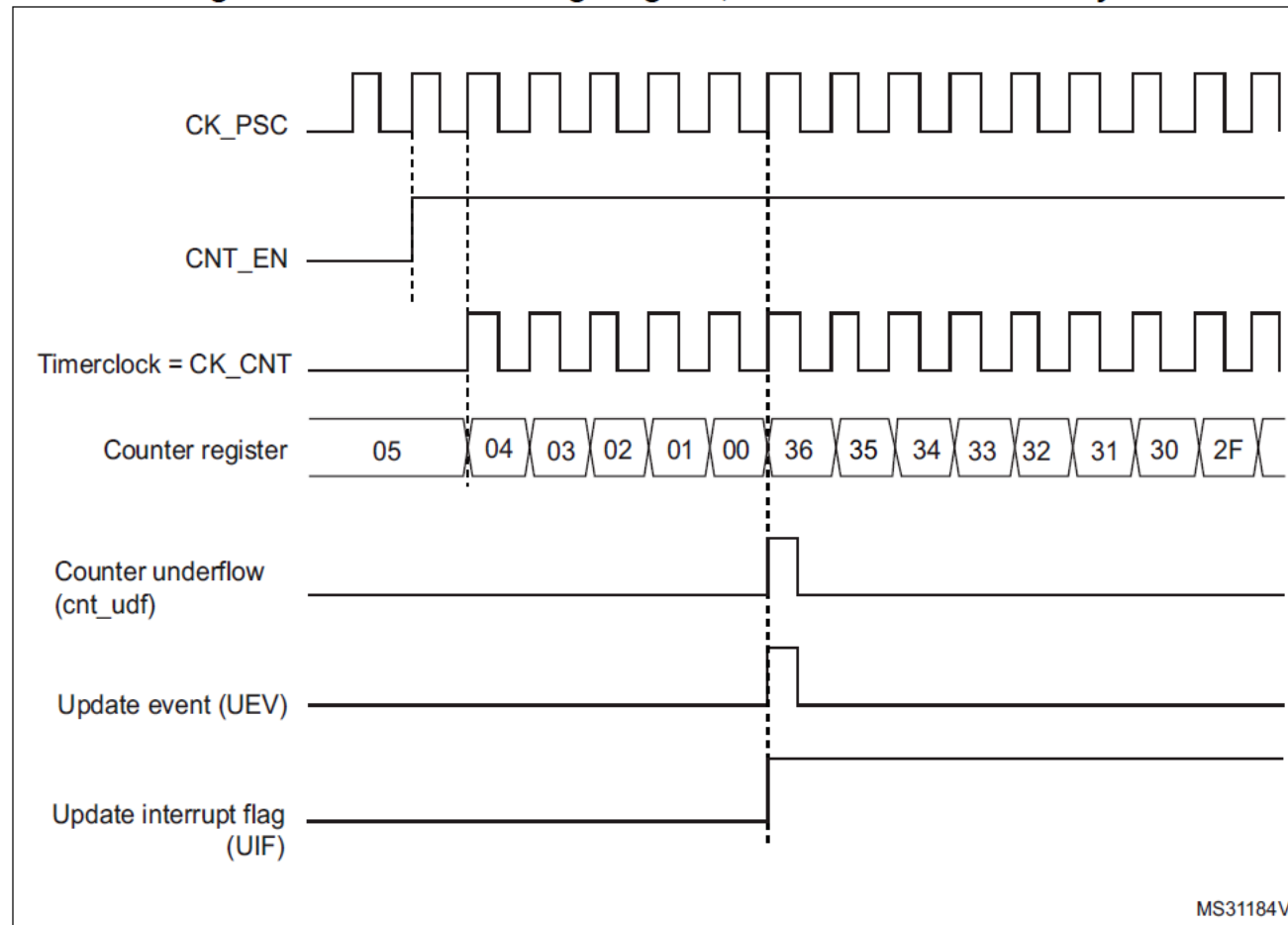
# Down Counting Mode, TIMx_ARR = 0x36
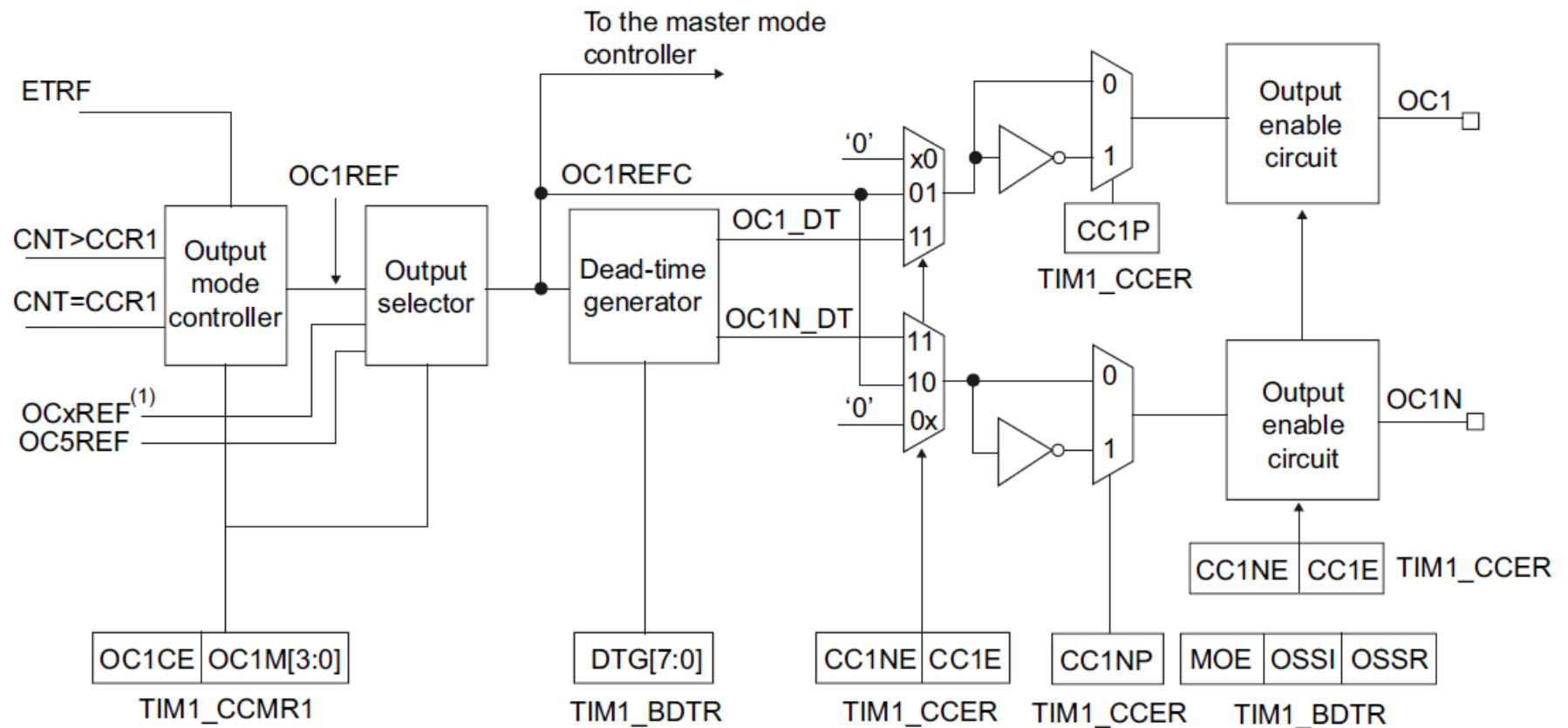


Figure 205. Counter timing diagram, internal clock divided by 2

# 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)**
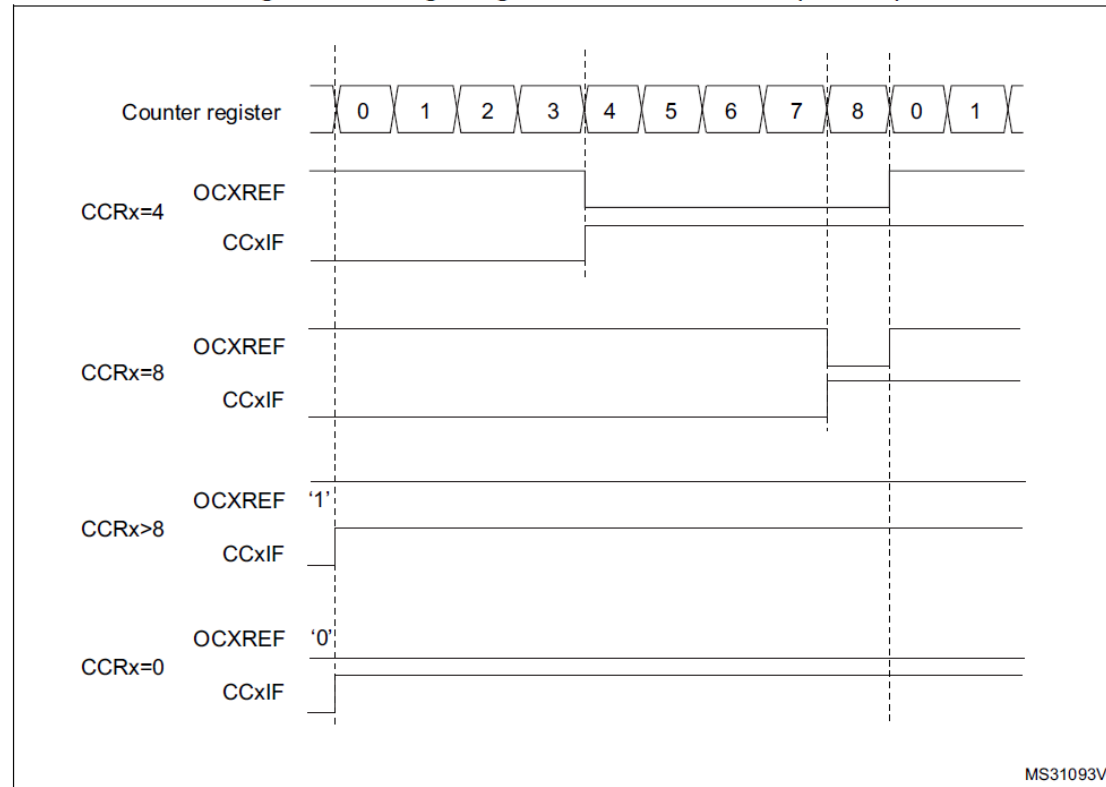
## Pulse Width Modulation PWM unit

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

On/off    (MLE)

DC Motor

Battery +
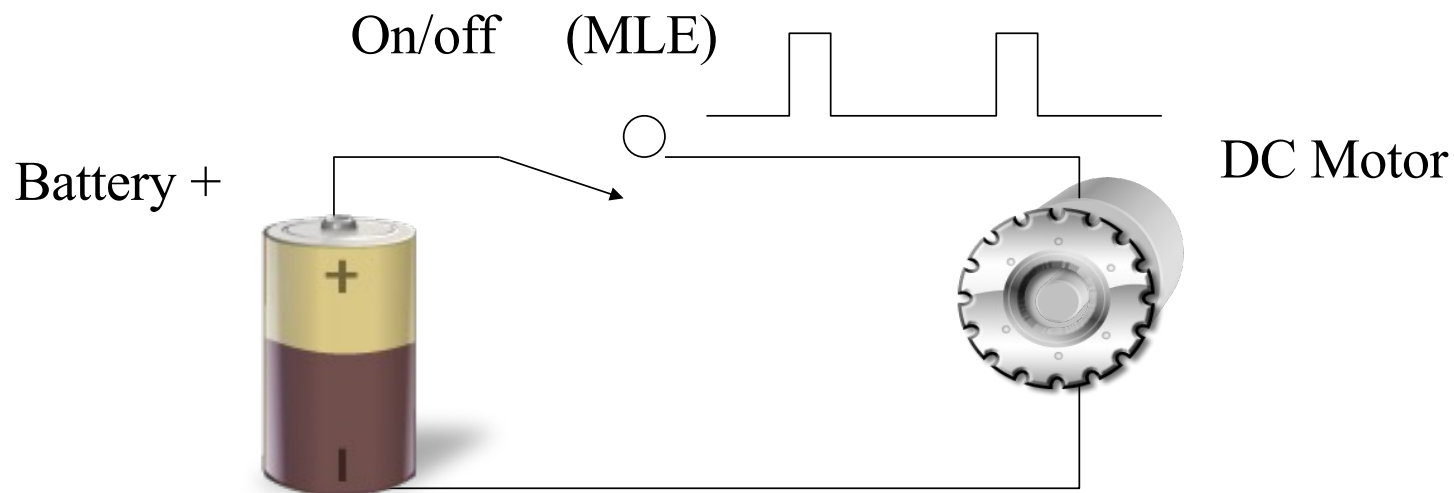
# Timing diagrams of pulse width modulation

- Comparing two pulse modulated signals S1,S2

T =Period 1ms

S1

Toff1

Ton1

S2

Toff2

Ton2

time
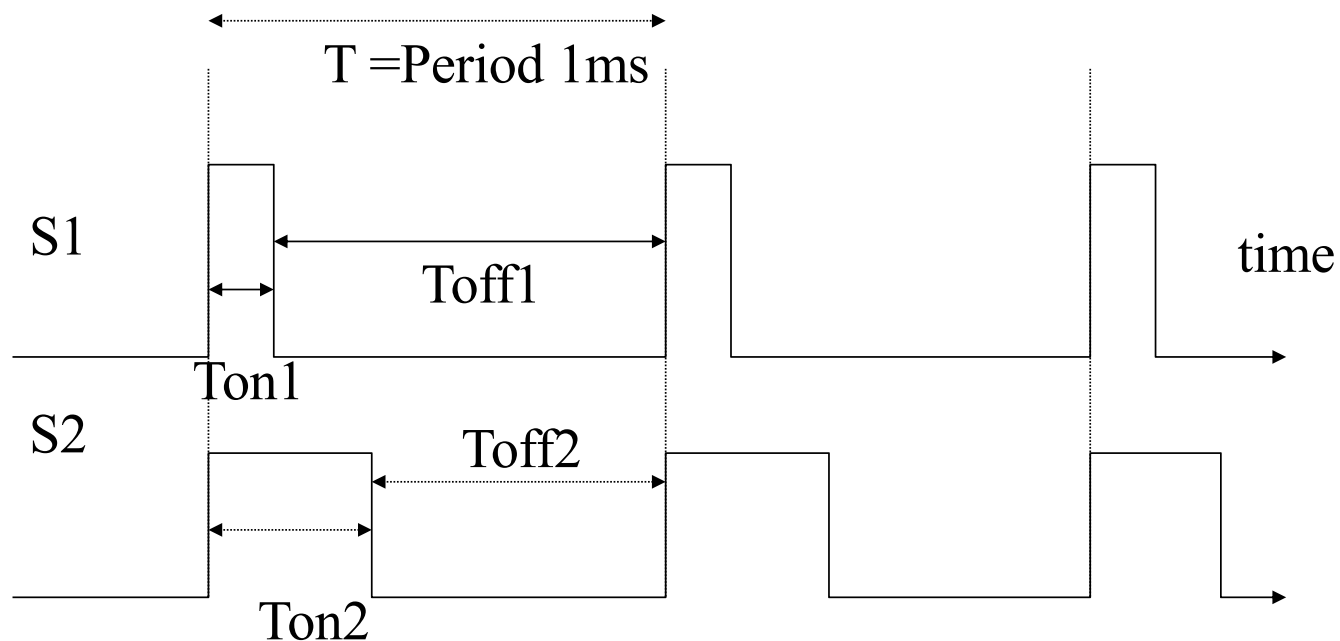
# Use L293 H bridge circuit

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

**PWMMR2**

L_DIR

**PWMMR5**

R_DIR

Left-motor

Right-motor

# L293D & RGB LED

# 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 SYS_AF | AF01 TIM1/TIM2 | AF02 TIM3/ TIM4/ TIM5 | AF03 TIM9/ TIM10/ TIM11 | AF04 I2C1/I2C2/ I2C3 | AF05 SPI1/I2S1S PI2/ I2S2/SPI3/ I2S3 | AF06 SPI2/I2S2/ SPI3/ I2S3/SPI4/ I2S4/SPI5/ I2S5 | AF07 SPI3/I2S3/ USART1/ USART2 | AF08 USART6 | AF09 I2C2/ I2C3 | AF10 OTG1_FS | AF11 | AF12 SDIO | AF13 | AF14 | AF15 |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| PA0 | - | TIM2_CH1/ TIM2_ETR | TIM5_CH1 | - | - | - | - | USART2_CTS | - | - | - | - | - | - | - | EVENT OUT |
| PA1 | - | TIM2_CH2 | TIM5_CH2 | - | - | SPI4_MOSI /I2S4_SD | - | USART2_RTS | - | - | - | - | - | - | - | EVENT OUT |
| PA2 | - | TIM2_CH3 | TIM5_CH3 | TIM9_CH1 | - | I2S2_CKIN | - | USART2_TX | - | - | - | - | - | - | - | EVENT OUT |
| PA3 | - | TIM2_CH4 | TIM5_CH4 | TIM9_CH2 | - | I2S2_MCK | - | USART2_RX | - | - | - | - | - | - | - | EVENT OUT |
| PA4 | - | - | - | - | - | SPI1_NSS/I 2S1_WS | SPI3_NSS/I2 S3_WS | USART2_CK | - | - | - | - | - | - | - | EVENT OUT |
| PA5 | - | TIM2_CH1/ TIM2_ETR | - | - | - | SPI1_SCK/I 2S1_CK | - | - | - | - | - | - | - | - | - | EVENT OUT |
| PA6 | - | TIM1_BKIN | TIM3_CH1 | - | - | SPI1_MISO | I2S2_MCK | - | - | - | - | - | SDIO_CMD | - | - | EVENT OUT |
| PA7 | - | TIM1_CH1N | TIM3_CH2 | - | - | SPI1_MOSI /I2S1_SD | - | - | - | - | - | - | - | - | - | EVENT OUT |
| PA8 | MCO_1 | TIM1_CH1 | - | - | I2C3_SCL | - | - | USART1_CK | - | - | USB_FS_SOF | - | SDIO_D1 | - | - | EVENT OUT |
| PA9 | - | TIM1_CH2 | - | - | I2C3_SMB A | - | - | USART1_TX | - | - | USB_FS_VBUS | - | SDIO_D2 | - | - | EVENT OUT |
| PA10 | - | TIM1_CH3 | - | - | - | - | SPI5_MOSI/I 2S5_SD | USART1_RX | - | - | USB_FS_I D | - | - | - | - | EVENT OUT |
| PA11 | - | TIM1_CH4 | - | - | - | - | SPI4_MISO | USART1_CTS | USART6_TX | - | USB_FS_DM | - | - | - | - | EVENT OUT |
| PA12 | - | TIM1_ETR | - | - | - | - | SPI5_MISO | USART1_RTS | USART6_RX | - | USB_FS_DP | - | - | - | - | EVENT OUT |

*Port A (label on left side of table)*

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

Our motor and
speed encoder

Each wheel rotation=
88 on/off changes

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

IR receiver output

Motor rotates

Motor gear box

A metal disk

holes

IR transmitter

wheel

IR receiver

☐ LWSensor

☐ RWSensor

IR receiver

Darkened part blocks light

IR light source

IR receiver
Speed Encoder
sensor

interrupts

1000 interrupts per second

time

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
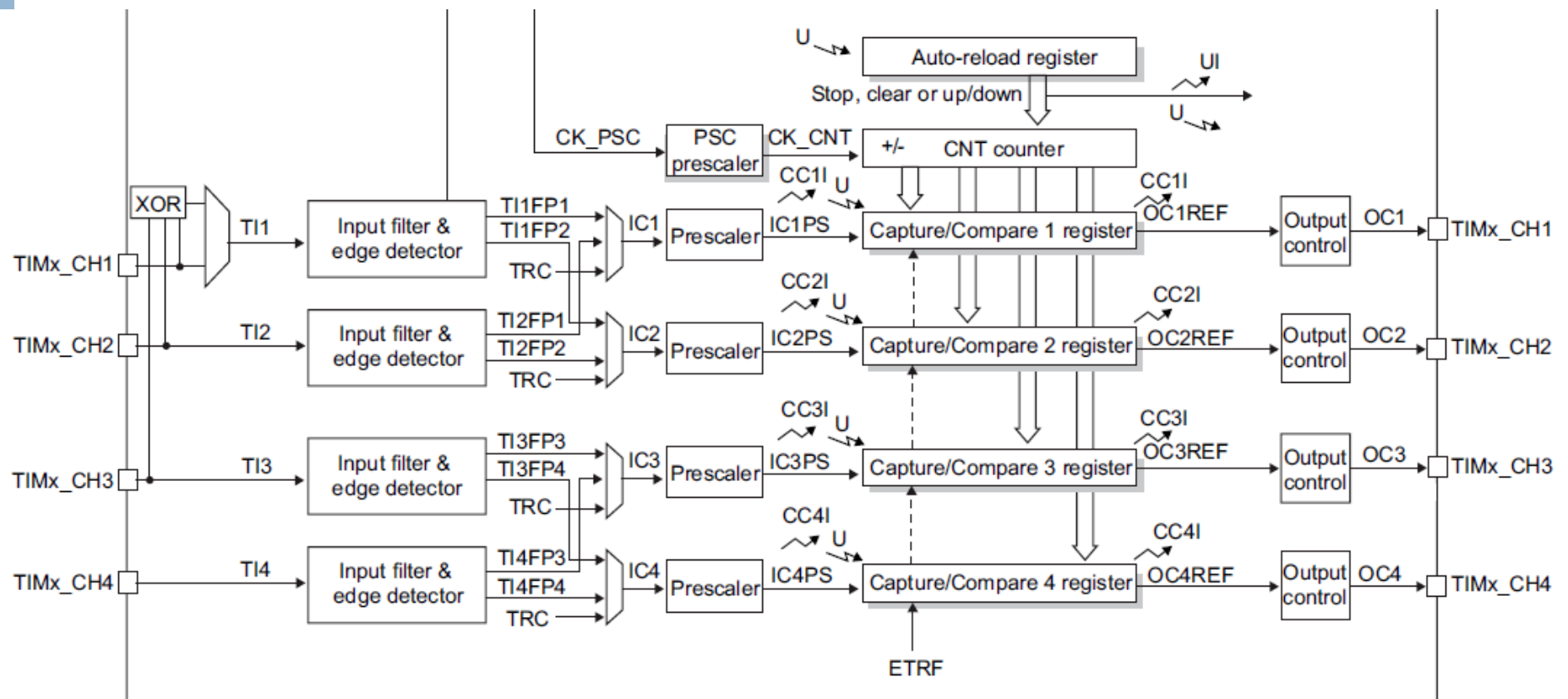
# TIM2-TIM5 Time-base unit

## 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
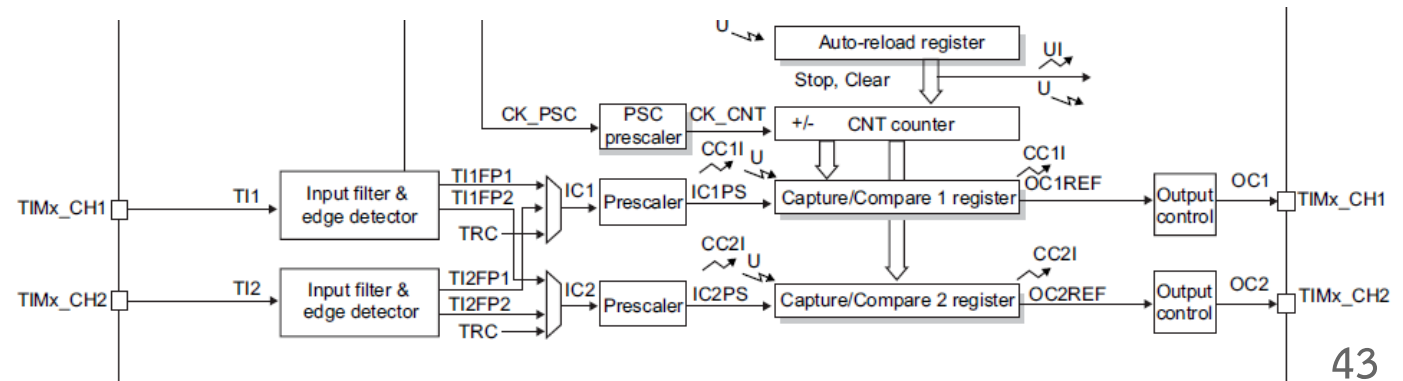
## TIM10/TIM11/TIM13/TIM14 main features

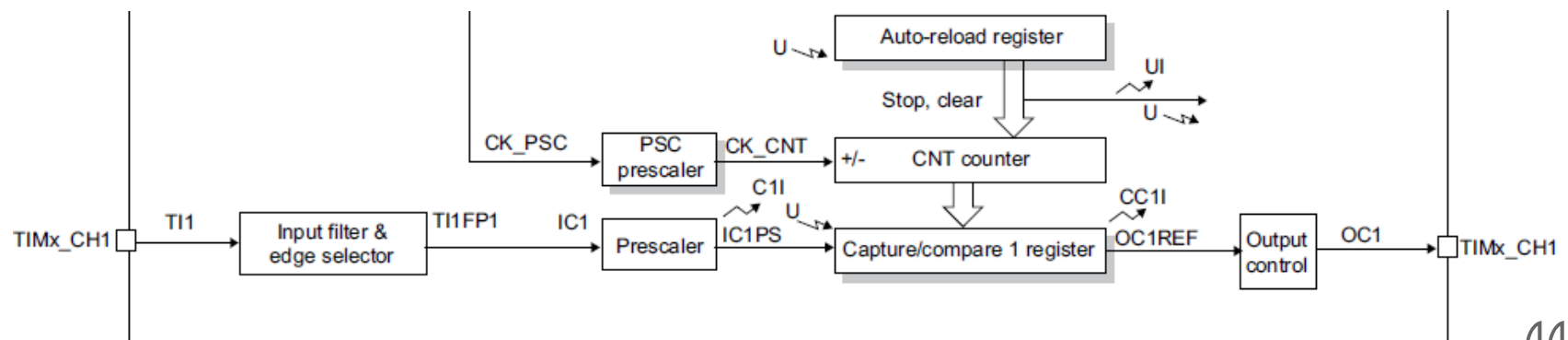The features of general-purpose timers TIM10/TIM11/TIM13/TIM14 include:

- 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
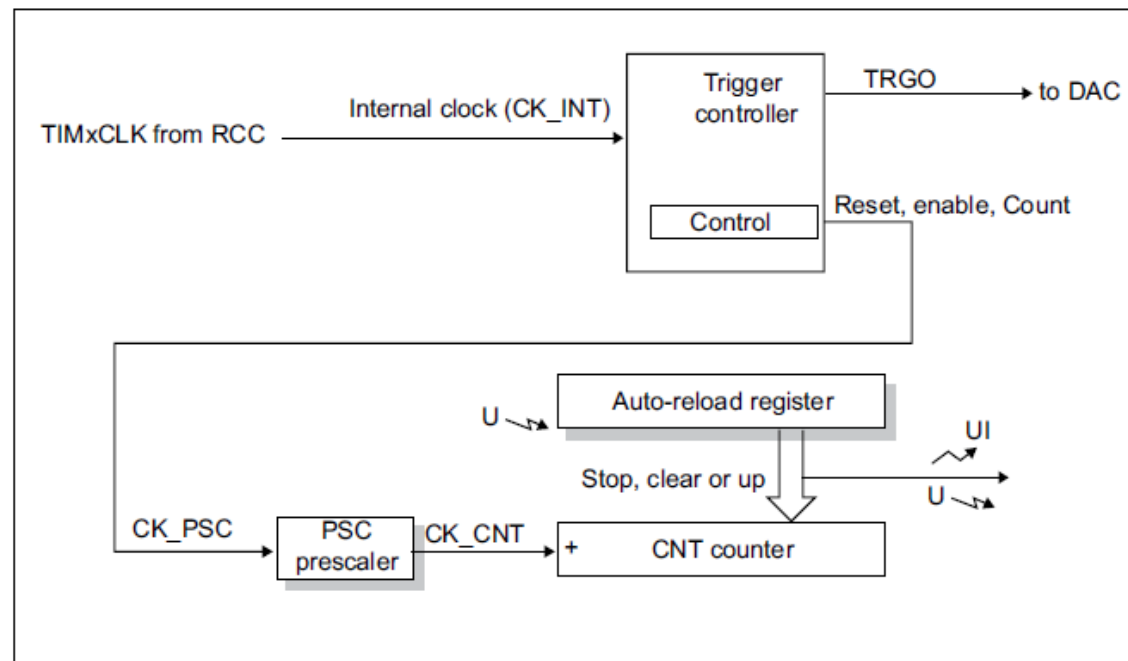
Basic timer (TIM6/TIM7) features include:

- 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
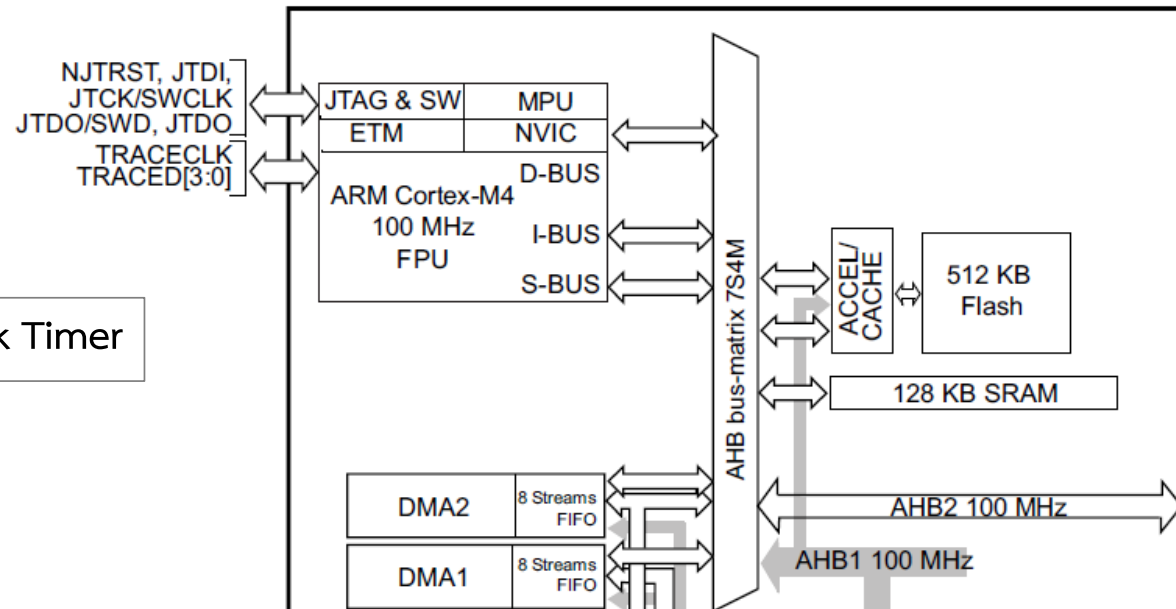


## SysTick timer

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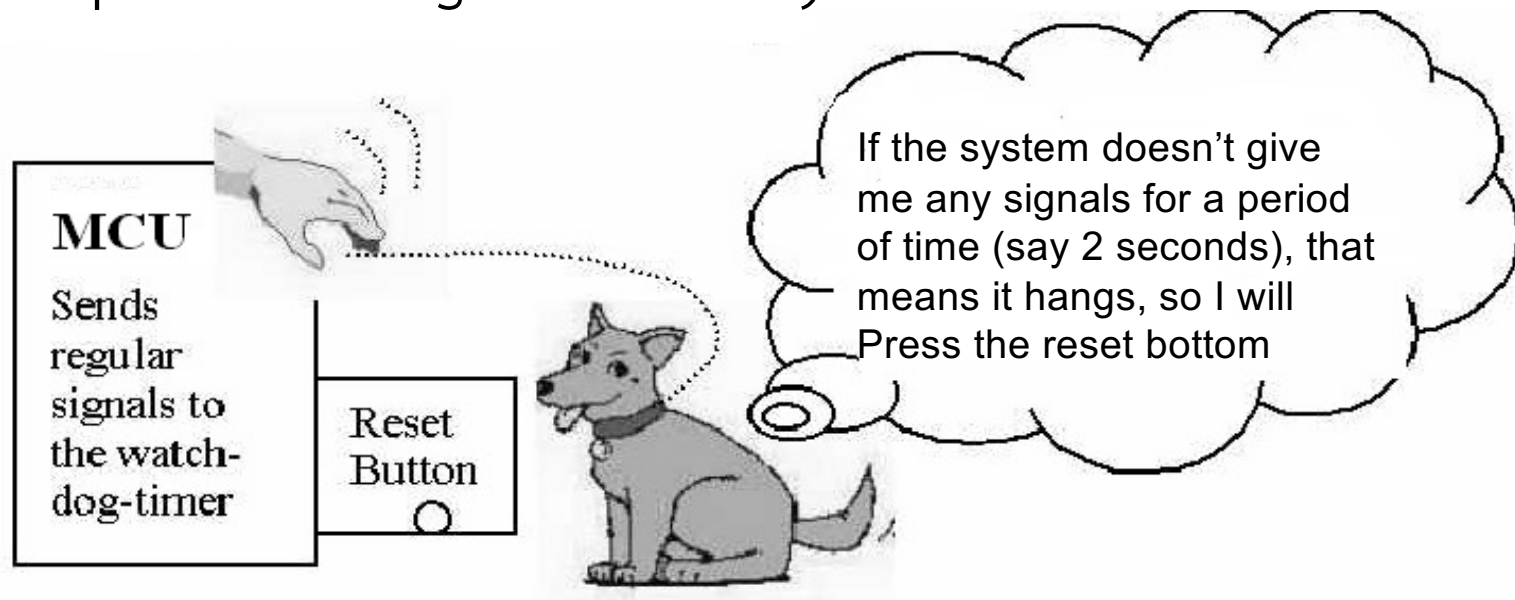
## Example, solar power wireless telephone (register setting , see appendix)



- 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



**MCU** Sends regular signals to the watch-dog-timer

Reset Button

If the system doesn't give me any signal for a period of time (say 2 seconds), that means it hangs, so I will Press the reset bottom
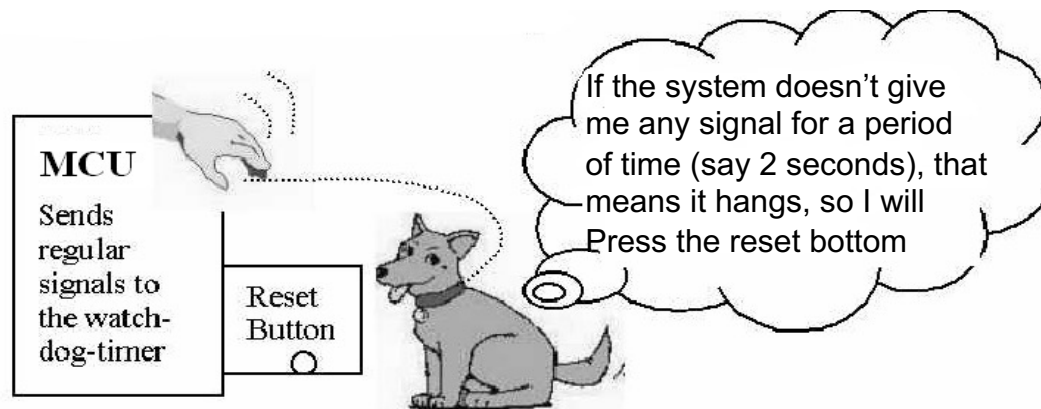
# Software

➢Main

{

   While(1)

   { Do_the _neccessary();

     Send_a_pulse_to_watch_dog();

   }

}



MCU

Sends regular signals to the watch-dog-timer

Reset Button

If the system doesn't give me any signal for a period of time (say 2 seconds), that means it hangs, so I will Press the reset bottom
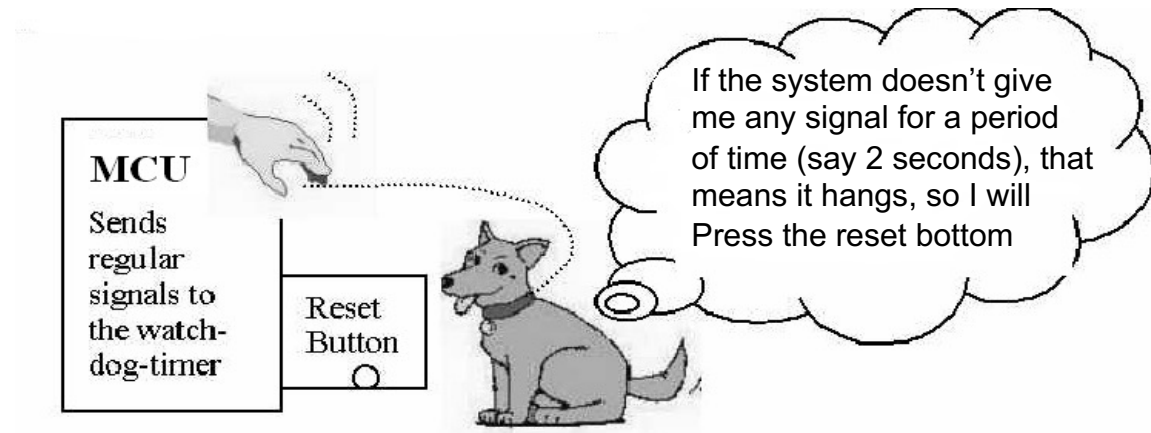
➢If the software hangs, it will not **Send_a_pulse_to_watch_dog()**;

➢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.
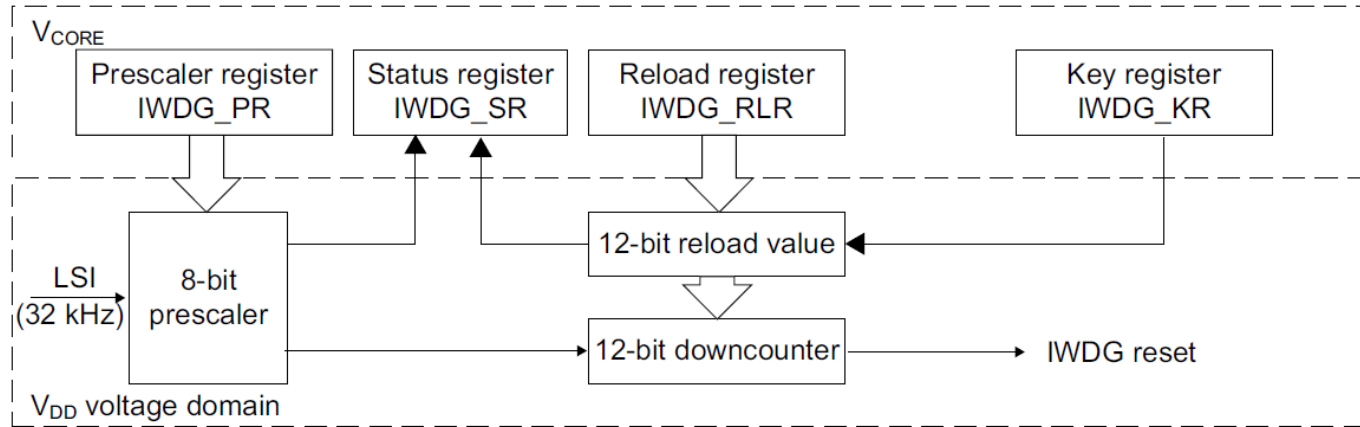
# Independent Watchdog



## 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.