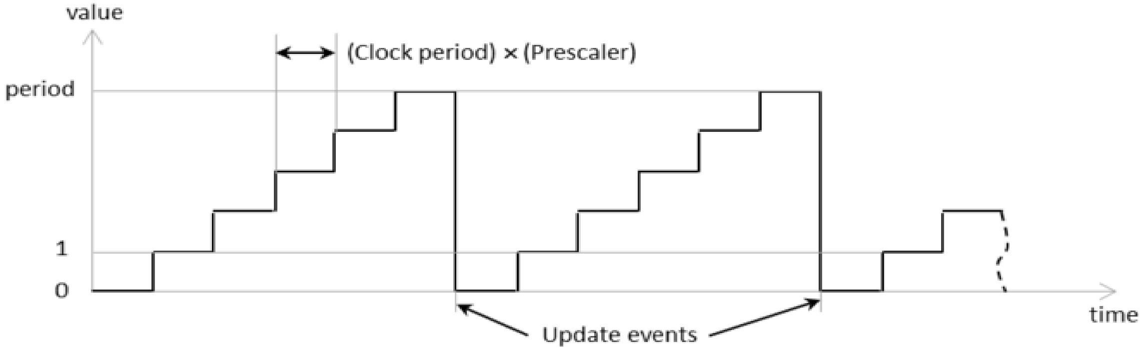
LAB 03
TIMERS

Prof. Davide Brunelli
Dept. of Industrial Engineering – DII
University of Trento, Italy
davide.brunelli@unitn.it

TIMERS

TIMERS

A hardware timer is essentially an **independent counter** that counts from zero to its maximum value at a given speed and generates various events. It runs in the background independently from your C/C++ program and its value typically follows the sequence depicted below:



It is basically a **global variable** (timer counter) that **increments** (or **decrements**) on the basis of a **programmable clock source**

TIMERS

The hardware of a timer is composed of **three basic programmable parts**:

- The **clock source**: the circuit that **generates the clock tick** for the timer
- The **time base**: the circuit that **derives the time granularity** from the clock source and contains the timer counter variable
- The **slave circuits**: provide **additional functions** (pulse measure, signal generation, etc.) by exploiting the timer variable

TIMERS

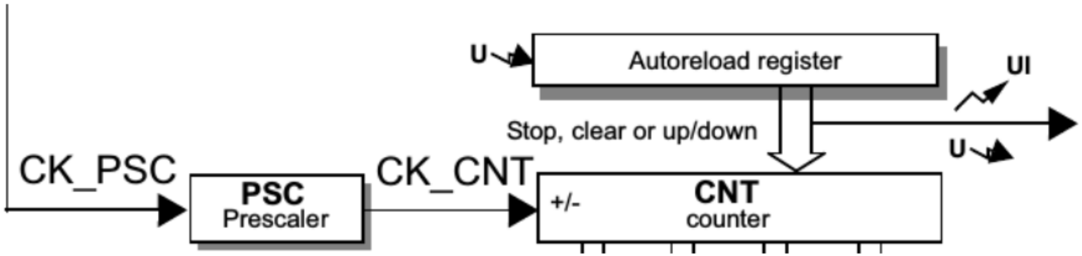
The **STM32 MCU** offers up to 11 different **timer/counters** with the following features:

- **Clock selection** (internal, external, other)
- **16/32-bit counter resolution**
- **Programmable prescaler**
- **Four independent channels** configurable as:
  - Input Capture
  - Output Compare
  - PWM Mode
  - One-pulse Output
- **Interrupt generation** on the basis of the various events that can occur

TIMERS

Counting is handled in the **time-base** by the following registers:

- **TIMx->PSC**: the prescaler register; it directly specifies the **division factor**
- **TIMx->CNT**: the counter register; it holds the counter value and increments (or decrements) it according to the input clock
- **TIMx->ARR**: the auto-reload register; **CNT** counts from 0 to **ARR**, then **CNT** is set to 0 again
- When **CNT** is **reloaded,** an **update event** is generated (the "**U**" in figure) which can trigger an **interrupt generation**

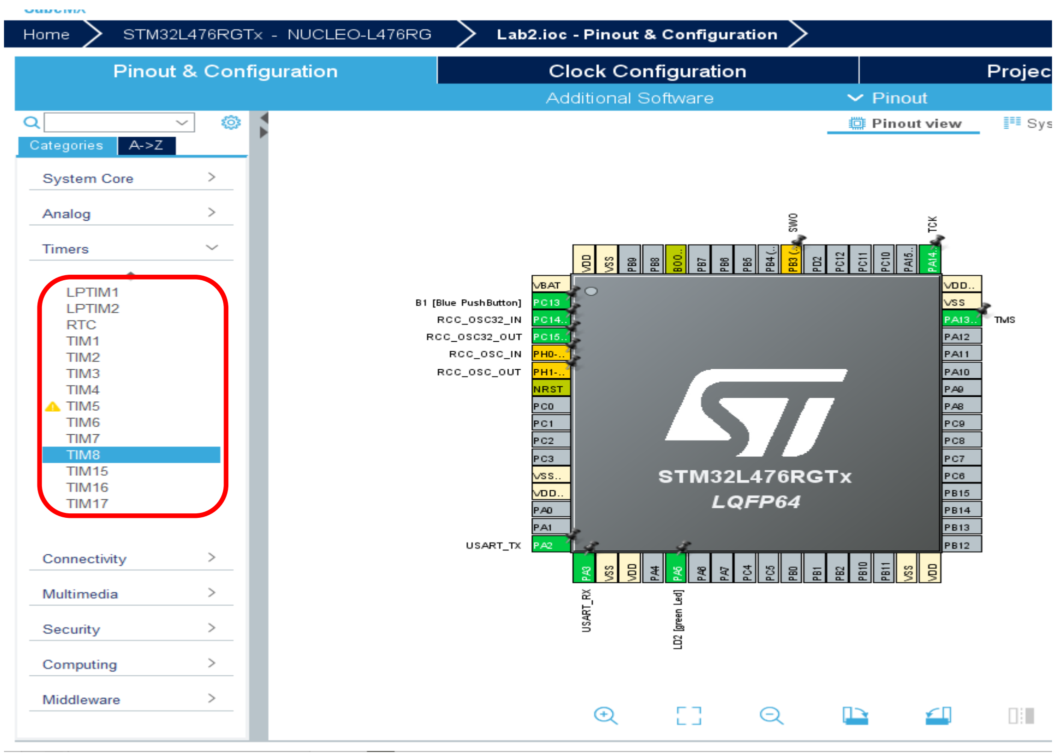


STM32 TIMERS

The STM32 embeds **multiple timers** providing timing resources for software or hardware tasks.

The software tasks mainly consist of providing **time bases, timeout event generation and time-triggers**.

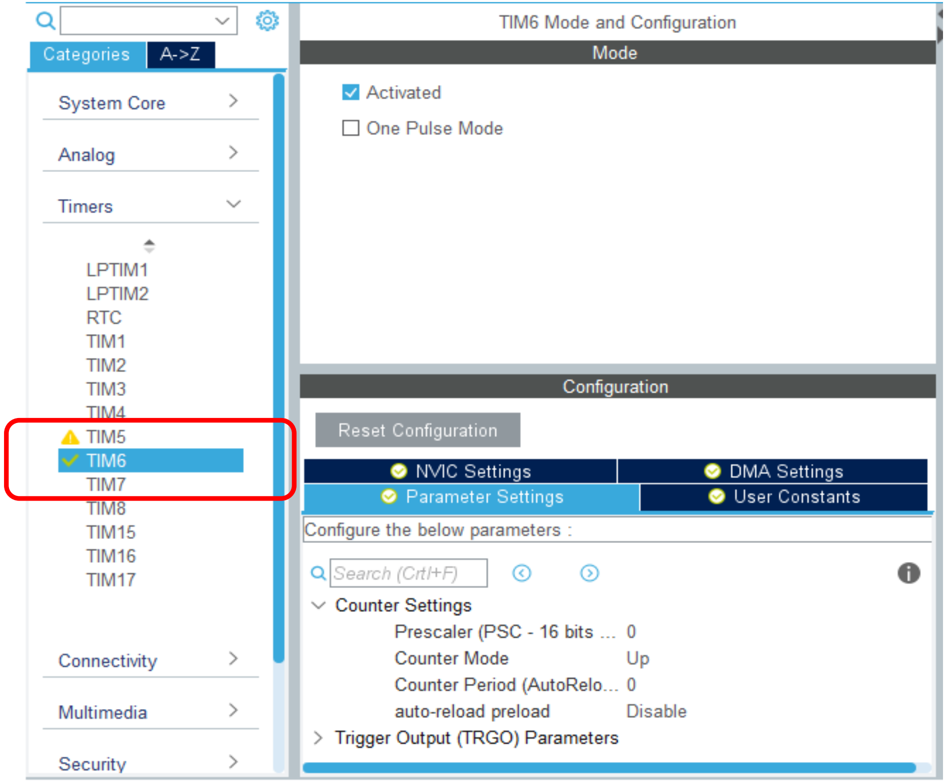
The hardware tasks are related to I/Os: the timers can **generate waveforms** on their outputs, **measure incoming signal** parameters and **react to external events** on their inputs.



STM32 TIMERS

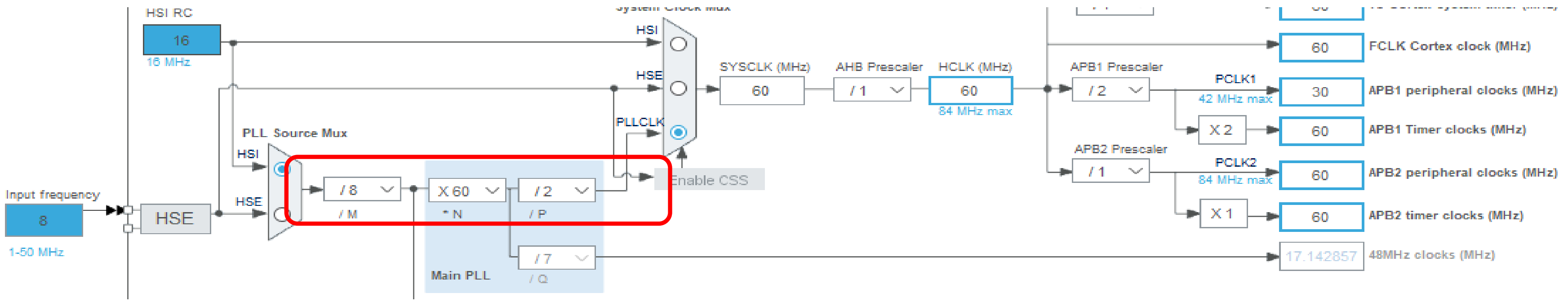
We will use **TIM6** for our board.

TIM6 is a simple Timer acting as a **Counter**. It just counts the clock cycles up to the maximum value set and then reset.



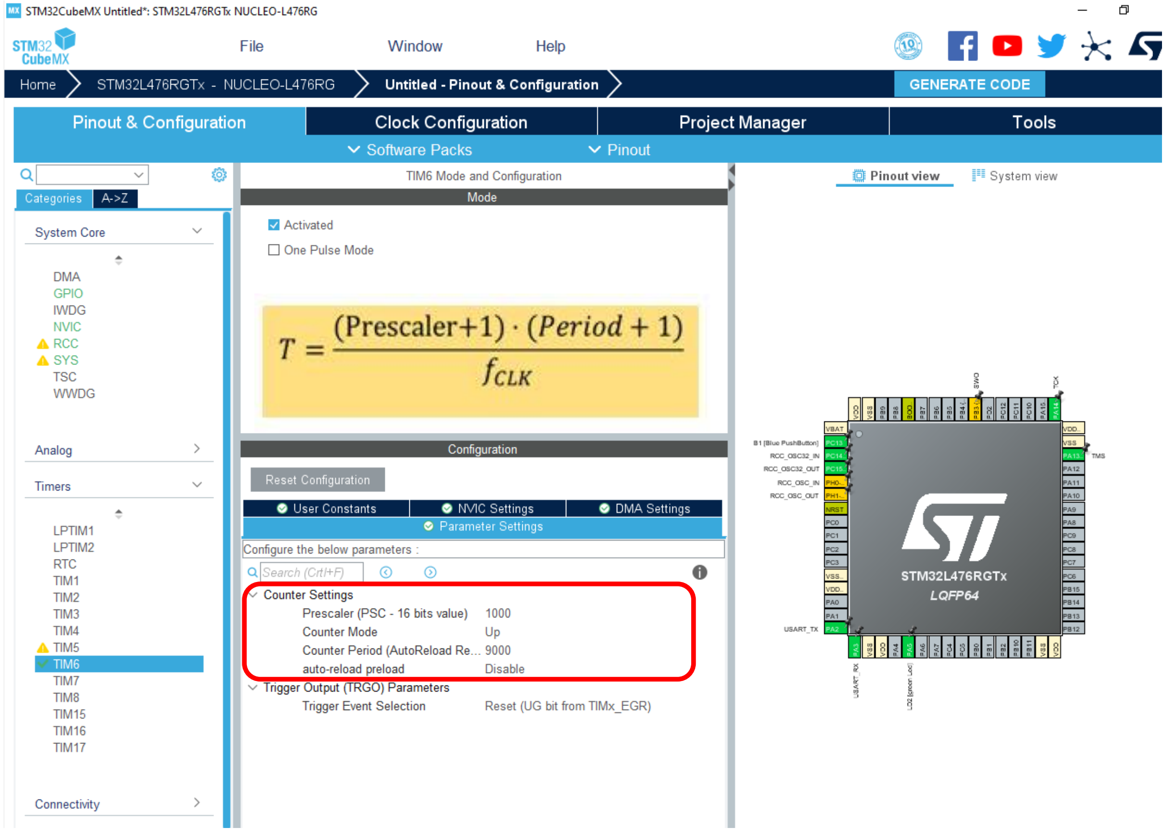
STM32 TIMERS

Timer tick time (or timer resolution) is base on **APBx** clock. It can be scaled using a **16 Bit prescaler.** We can modify the base clock, to simplify the pre-scaler division.



$$t_{TIMx} = \frac{\text{Prescaler} + 1}{APBx_{CLK}}$$

STM32 TIMERS

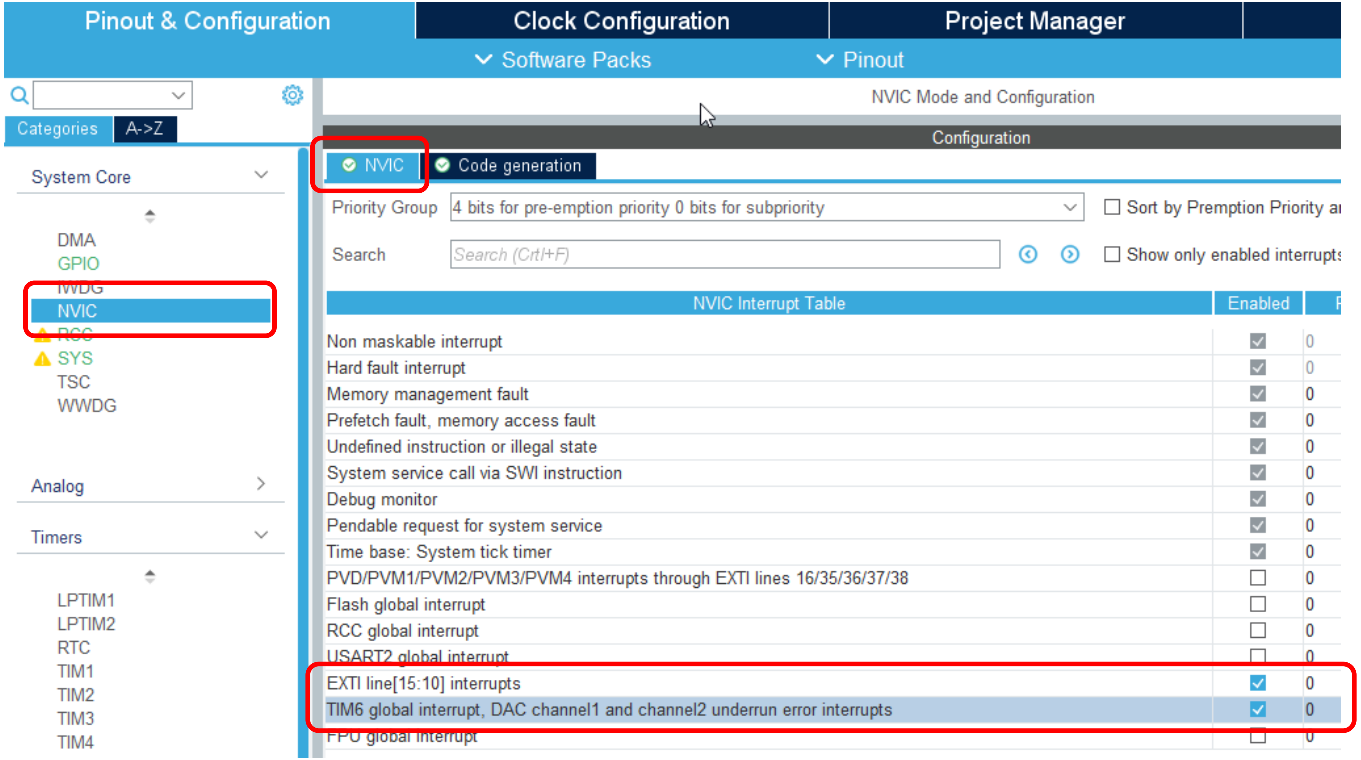


$$Period = \frac{T \cdot APBx_{CLK}}{Prescaler + 1} - 1$$

$$t_{TIMx} = \frac{Prescaler + 1}{APBx_{CLK}}$$

STM32 TIMERS – INTERRUPT

We can activate the interrupt using the NVIC configuration tab in CubeMX



STM32 TIMERS – INTERRUPT

Once the timer has reached its counting period, an interrupt will be fired. Using the proper callback function we can create timed tasks (like the blinking of a LED)

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    // The CUBEMX was set as: Prescaler: 1000, Counter Period (ARR): 9000, HCLK: 9 MHz
    HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
}

/* USER CODE END 4 */
```

After the Peripheral Initializations, remember to start TIM6 In Interrupt Mode.

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    HAL_TIM_Base_Start_IT(&htim6); // Start the timer

};
```

STM32 TIMERS – RESOURCES

- General-purpose timer cookbook for STM32 microcontrollers
  - https://www.stmicroelectronics.com.cn/content/ccc/resource/technical/document/application_note/group0/91/01/84/3f/7c/67/41/3f/DM00236305/files/DM00236305.pdf/jcr:content/translations/en.DM00236305.pdf

- STM32L4 Timers
  - https://www.st.com/content/ccc/resource/training/technical/product_training/c4/1b/56/83/3a/a1/47/64/STM32L4_WDG_TIMERS_GPTIM.pdf/files/STM32L4_WDG_TIMERS_GPTIM.pdf/jcr:content/translations/en.STM32L4_WDG_TIMERS_GPTIM.pdf

TOGGLE THE LED USING A TIMER

TOGGLE THE LED USING A TIMER

Use the previously configured timer to toggle the green LED every 1 second.

1. Use CubeMX to **configure a simple timer**
2. Configure the board clock to provide an **APB clock equal to 60 MHz**
3. If needed, pre-scale the clock to a suitable frequency for the timer
4. Set the timer counting period
5. Start the timer **using the blue button**
6. Using the interrupt callback, **toggle the green LED**

CHANGE THE TIMER COUNTING PERIOD

CHANGE THE TIMER COUNTING PERIOD

Using an interrupt generated by an external input, cycle the timer counting period between 0 – 0.25s – 0.5s – 1s – 2s

1. Use CubeMX to **configure a simple timer**
2. Configure the board clock to provide an **APB clock equal to 60 MHz**
3. If needed, pre-scale the clock to a suitable frequency for the timer
4. Set the timer counting period
5. Using the interrupt callback, **cycle the counting period** of the configured timer by updating the relative register.