LAB 04
PWM

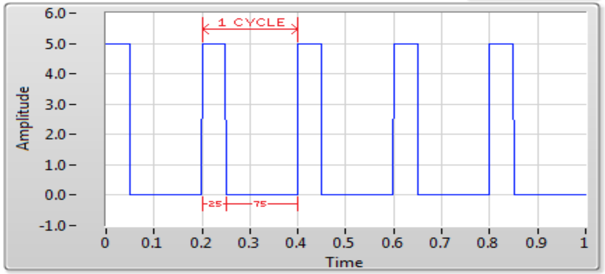
UNIVERSITY OF TRENTO

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

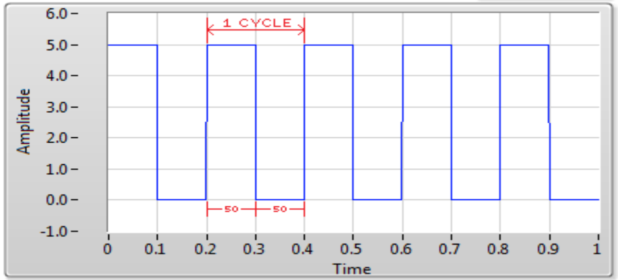
PWM

PWM

A Pulse Width Modulation (PWM) Signal is a method for generating an analog signal using a digital source. A PWM signal consists of two main components that define its behavior: a **duty cycle and a frequency**. The duty cycle describes the amount of time the signal is in a high (on) state as a percentage of the total time it takes to complete one cycle.



25% Duty Cycle

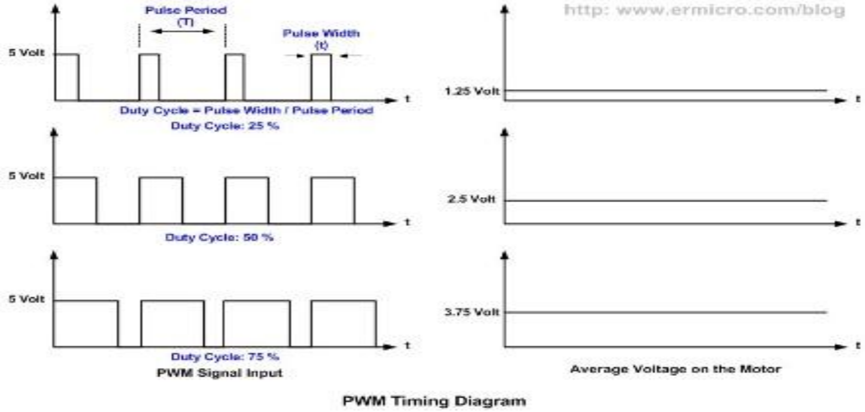


50% Duty Cycle

The frequency determines how fast the PWM completes a cycle (i.e. 1000 Hz would be 1000 cycles per second), and therefore how fast it switches between high and low states. By cycling a digital signal off and on at a fast enough rate, and with a certain duty cycle, the output will appear to behave like a constant voltage analog signal when providing power to devices.

PWM

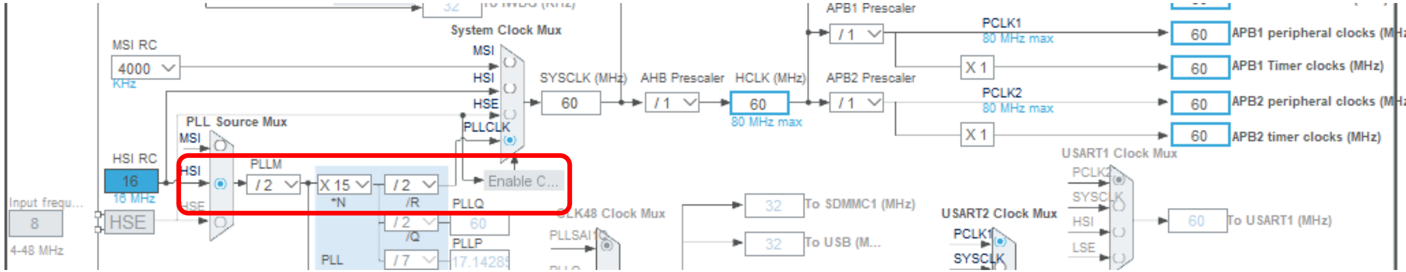
Example: To create a 3V signal given a digital source that can be either high (on) at 5V, or low (off) at 0V, you can use PWM with a duty cycle of 60% which outputs 5V 60% of the time. If the digital signal is cycled fast enough, then the voltage seen at the output appears to be the average voltage.



If the digital low is 0V (which is usually the case) then the average voltage can be calculated by taking the digital high voltage multiplied by the duty cycle, or 5V x 0.6 = 3V. Selecting a duty cycle of 50% would yield 2.5V, 25% would yield 1.25V, and so on.

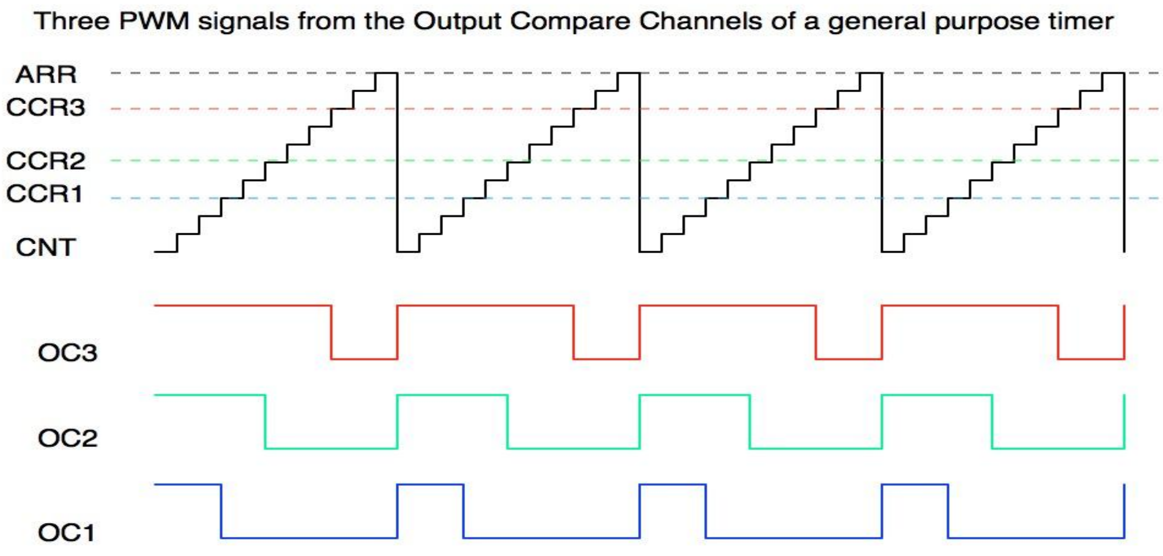
STM32 TIMERS

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.



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

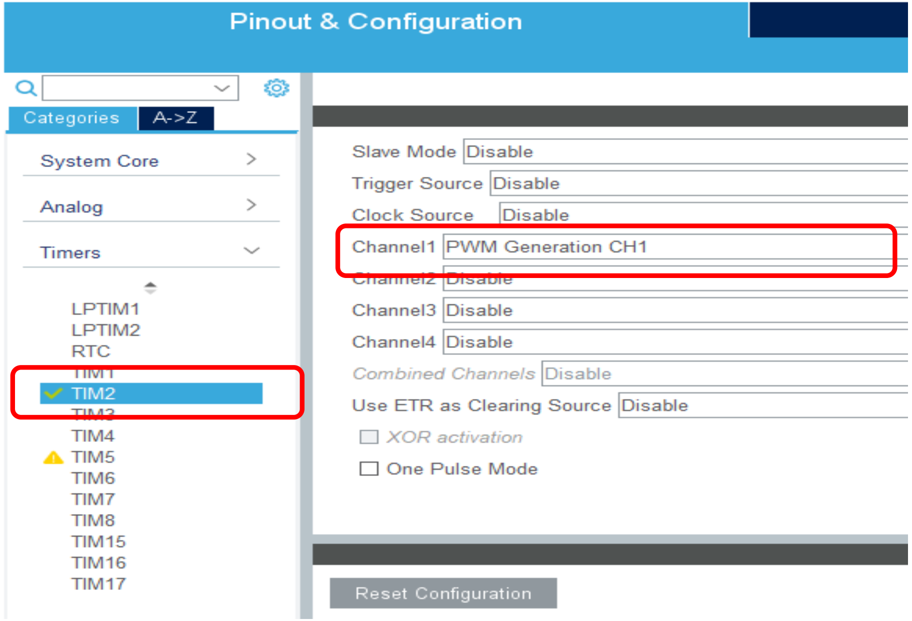
STM32 TIMERS – PWM



The output pin is cleared whenever there is a match between the CCRx and the CNT registers and then set again when the counter reloads.

STM32 TIMERS – PWM

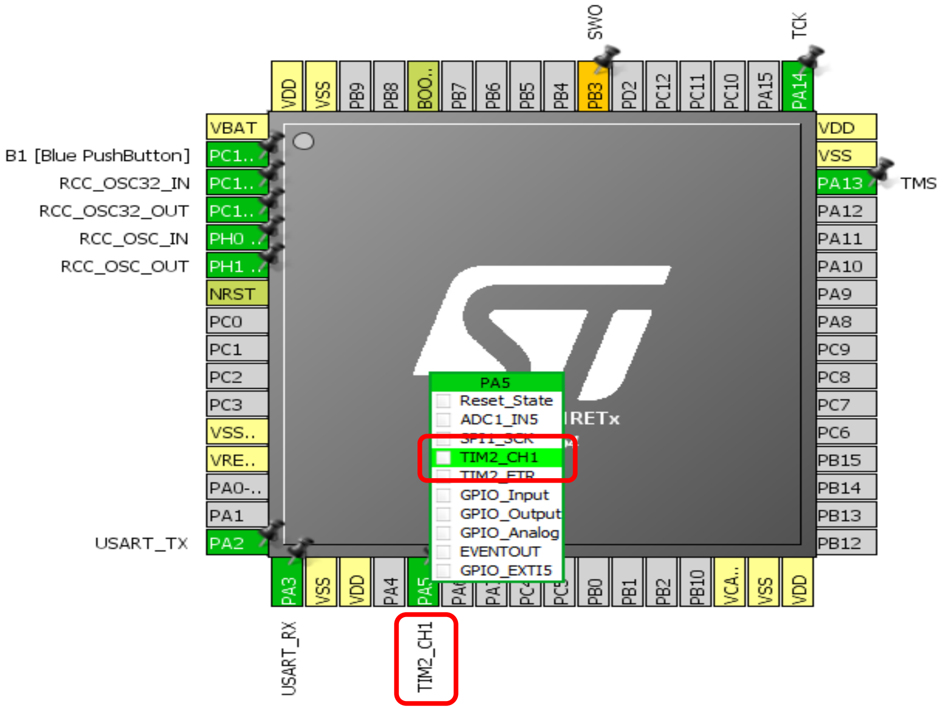
As previously said, the F4 MCUs have different timers, with a range of functions. Some of them are able to generate a PWM signal



We can use TIM2 to generate a PWM signal.

Moreover, we can directly connect the signal generated by the timer to a specific channel (i.e. GPIO)

STM32 TIMERS – PWM

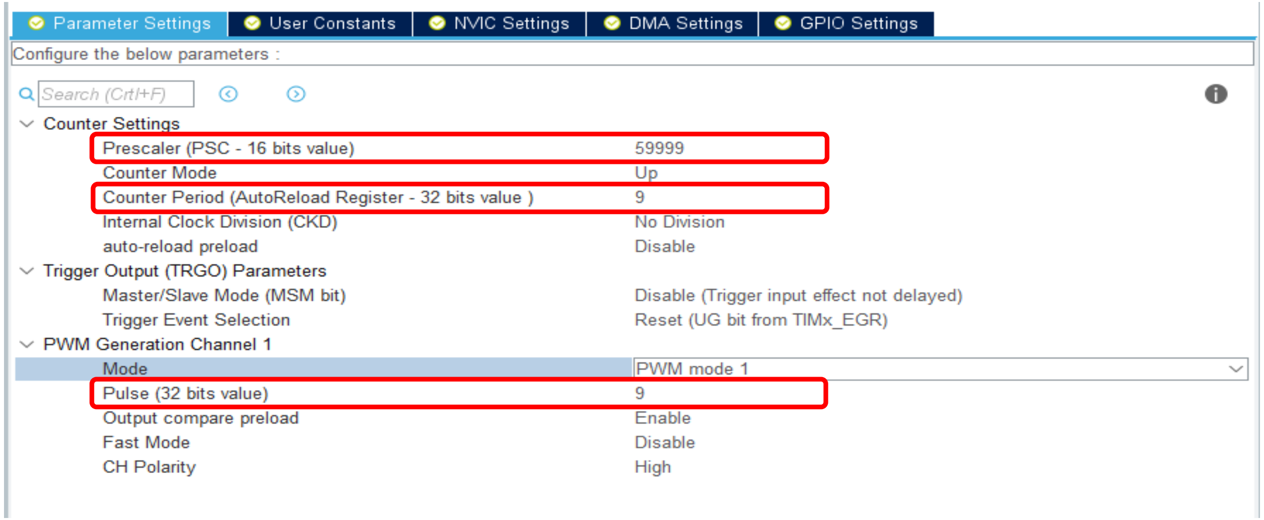


PA5, the GPIO associated to the green LED, can be connected to Channel 1 of TIM2.

In this way, we can drive the green LED using the PWM signal generated by the timer.

STM32 TIMERS – PWM

Click on TIM2 configuration. Set the Prescaler value equal to 59999 and a Counter Period value equal to 9 (to achieve ~10 mS).



Under PWM Gen. Channel 1, set Pulse equal to 9

STM32 TIMERS – PWM

After the configurations are done using CubeMX, we can regenerate the code and open it using System Workbench. Then, we just have to start the timer in PWM mode

```
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_TIM2_Init();
MX_TIM6_Init();
MX_USART2_UART_Init();
/* USER CODE BEGIN 2 */
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1); // start timer 2
```

CHANGE THE DUTY CYCLE DYNAMICALLY

CHANGE THE DUTY CYCLE DYNAMICALLY

Use the previously configured timer to generate a PWM signal to drive the LED.

1. Use CubeMX to **configure a timer which can generate a PWM**
2. Configure the timer **as previously shown**
3. **Enable an interrupt for the blue pushbutton**
4. Using proper callbacks, modify the duty cycle of the PWM when the button is pressed

   (To modify the duty cycle: **TIM2 -> CCR1** .

   Remember to restart the timer after this change each time.)

FADING LED THROUGH PWM

FADING LED THROUGH PWM

Use the previously configured timer to generate a PWM signal to fade the LED.

1. Use CubeMX to **configure a timer which can generate a PWM**
2. Configure the timer **as previously shown**
3. Using the proper callback, **cycle the duty cycle of the PWM to get a fading effect of the LED**, i.e. increase gradually the PWM duty cycle up to a maximum and then decrease it down to a minimum and keep repeating this loop.

   (To modify the duty cycle: **TIM2 -> CCR1** .

   Remember to restart the timer after this change each time.)

FADING LED THROUGH PWM WITH INTERRUPTS

LED FADING WITH PWM USING INTERRUPTS

Use a second timer to drive the fading generated by the previously configured timer.

1. Use CubeMX to **configure a second timer**
2. Using the callback of the second timer, **cycle the duty cycle of the PWM to get a fading effect of the LED**, i.e. increase gradually the PWM duty cycle up to a maximum and then decrease it down to a minimum and keep repeating this loop.

   (To modify the duty cycle: **TIM2 -> CCR1** .

   Remember to restart the timer after this change each time.)