4a laboratory work

TIMER and PWM

1. Aim

- Learn how to use STM32CubeIDE for the programming of STM32 microcontrollers.
- Use basic in Embedded C language.
- Learn how to use timer interrupts to do precise time delays and generate PWM.

2. Theory

During previous labs, we learned how to use HAL_DELAY to turn on/off LEDs with different durations. In this lab, we will also toggle the LED, but we'll do it through a timer interrupt handler.

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:

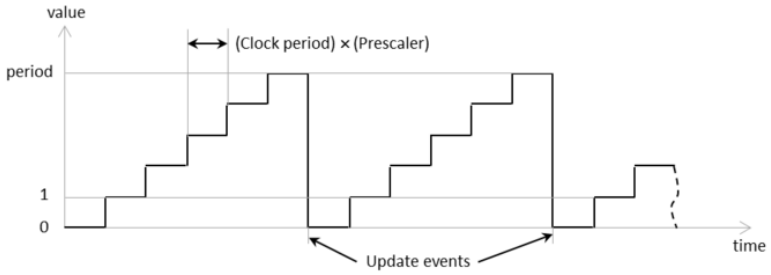

Fig 1. Timer counting

Let's create a new project and set up PA5 as **GPIO Output with Name LED**.

Select TIM1 timer and choose **Internal Clock** for **Clock source**.

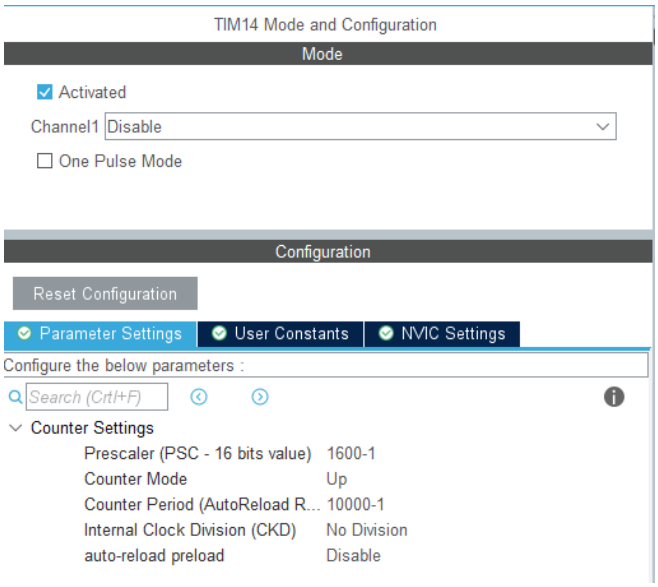

Fig 2. Timer1 settings

TIM1 is connected to APB1 bus which default frequency is 16 MHz (in one of the future posts I'll describe how to set different frequencies of all busses). Let's set the prescaler value to 16000 (in counter settings we should enter (PSC – 1) value). Thus, to calculate timer frequency we should divide APB1 frequency by 16000 prescaler:

10kHz pulse is 1/10000=0,0001 sec

Furthermore, we should set the proper value of counter period. If we set it to 10000 (as shown at the screenshot 10000-1), we'll get the following value of timer period:

*T = 0,0001 sec * 10000 = 1 sec*

And the final step of TIM1 configuration is enabling its interrupt. This can be done at the "Nvic Settings" tab. Finally, let's start code generation!

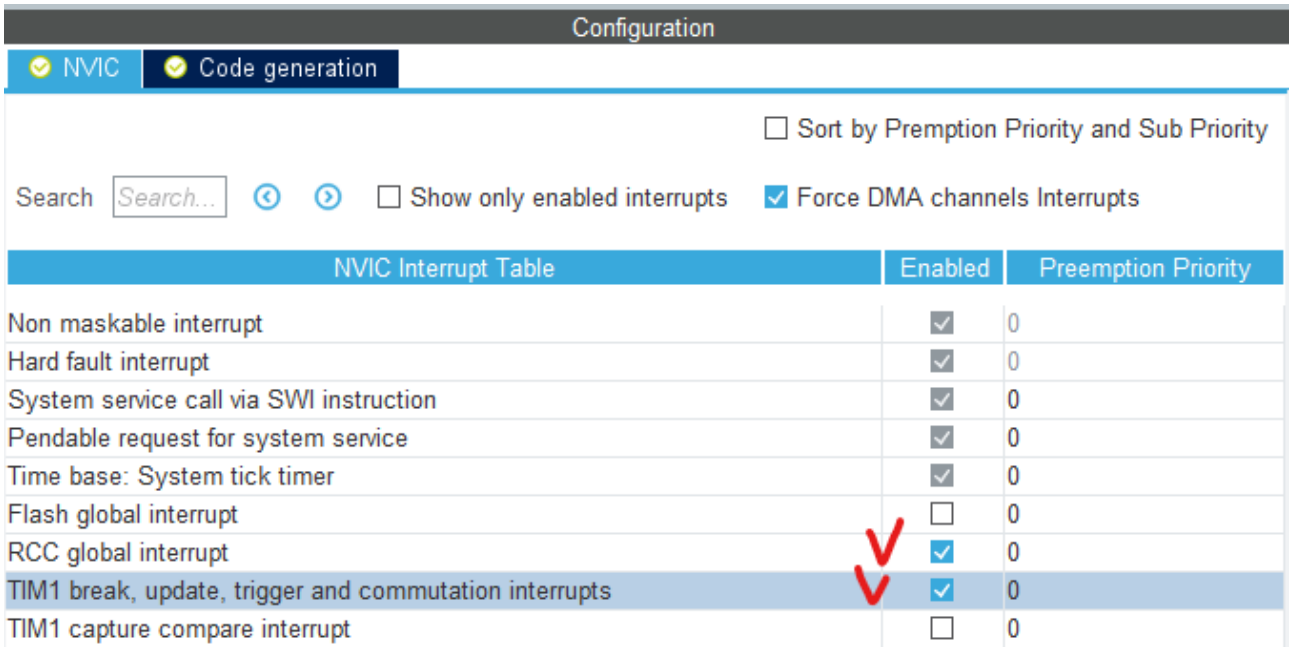

Fig 3. Timer interrupt enable

Add code bellow to the main.c. This is so called call back or interrupt routine function. During each interrupt, inside code will be executable. In our case it is TogglePin.

```
/* USER CODE BEGIN 0 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim1)
{
  HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);

}
/* USER CODE END 0 */
```

Timer callback function

Timer is not running. To run it you need to add code bellow to main.c

```
  /* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start_IT(&htim1);
  /* USER CODE END 2 */
```

Timer interrupt start function

PWM generation

Some timers of STM32 has many features, that gives us possibility to generate different type of signals, like PWM on microcontroller outputs.

Let's setup Timer1 for PWM generation on CH1.

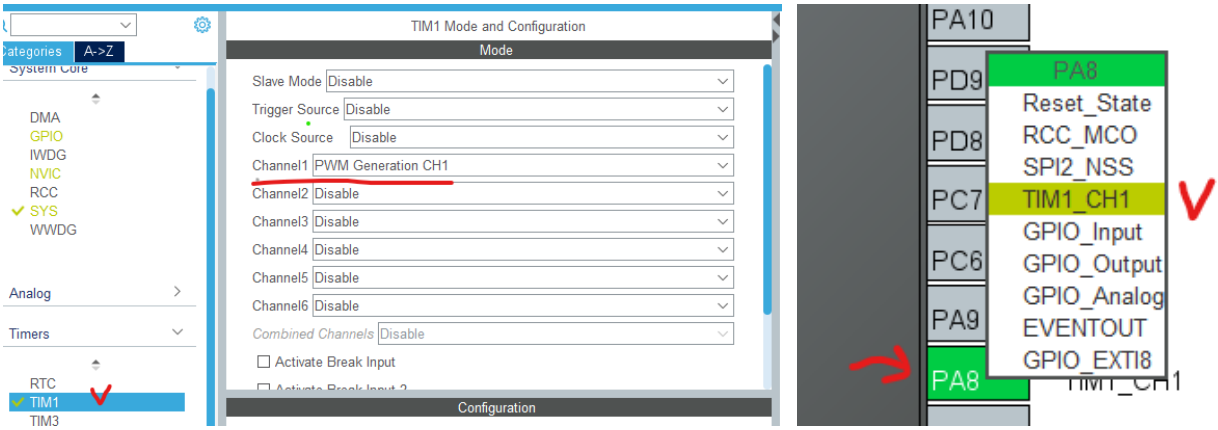

Fig 4. Timer Channel for PWM output selection

This will generate PWM signal on CH1 pin output (PA8).

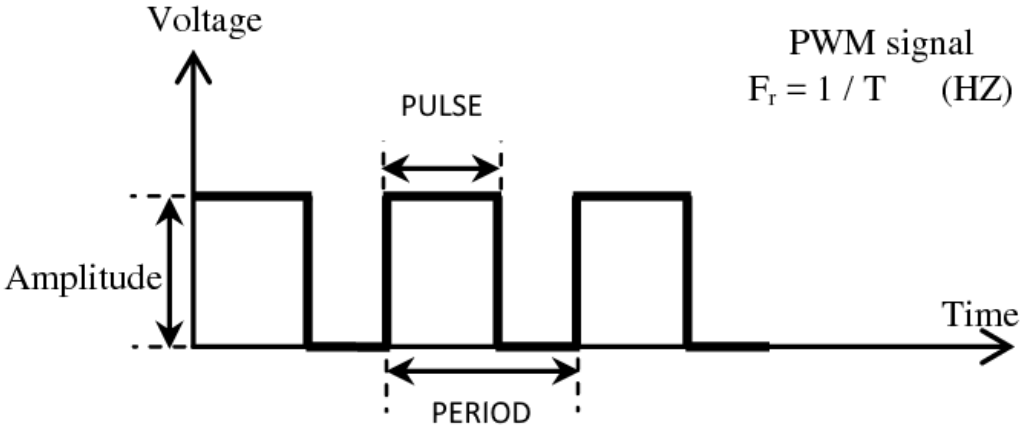

Fig 4. PWM output parameters

Equation for calculate period of PWM is this:

T = (1/APB_TIM_CLK in MHz) * (PRESCALER_Value + 1) * (PERIOD_Value + 1)

MCU Clock - APB_TIM_CLK = 16 MHz
PRESCALER_Value = 999
PERIOD_Value = 15999

the formula is:

T= (1/16*10^6) * (999+1) * (15999+1) = 1s

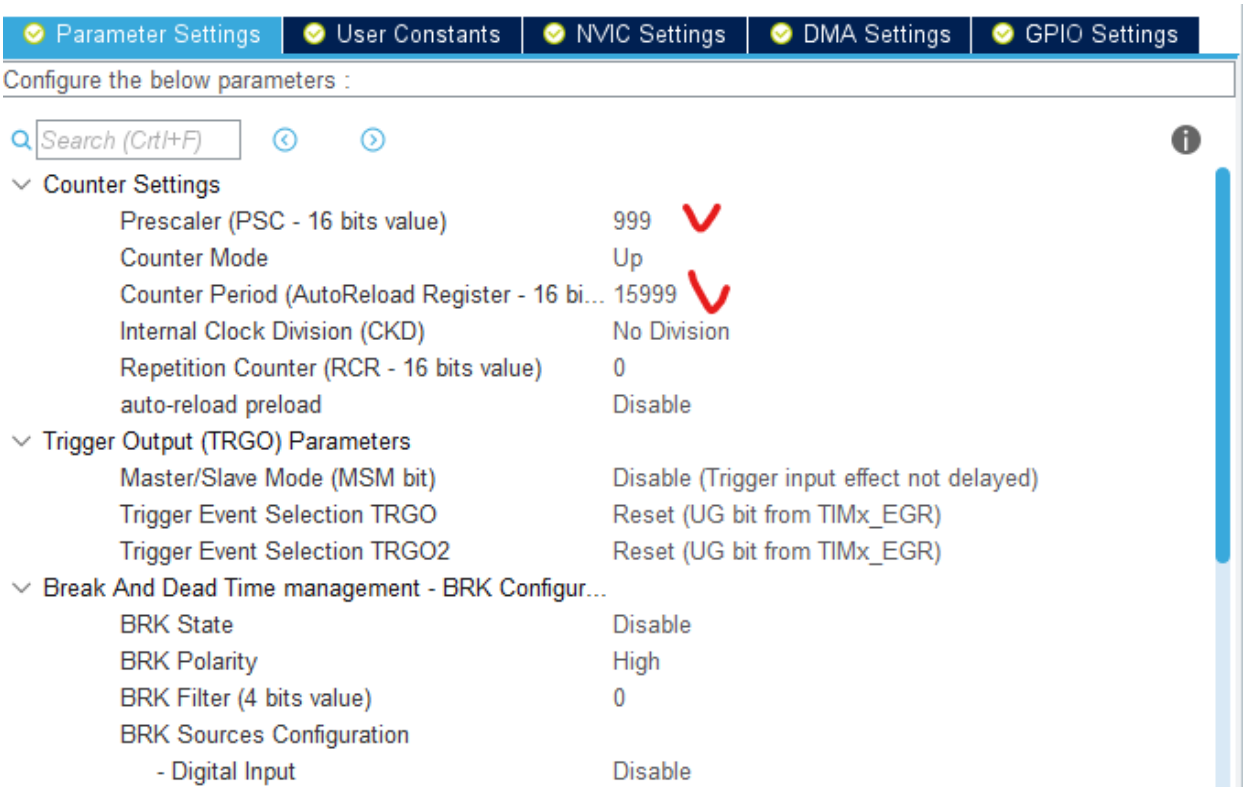

Fig 5. Timer Prescaler and Period settings

Pulse value is 7999 – half of period (50% PWM).

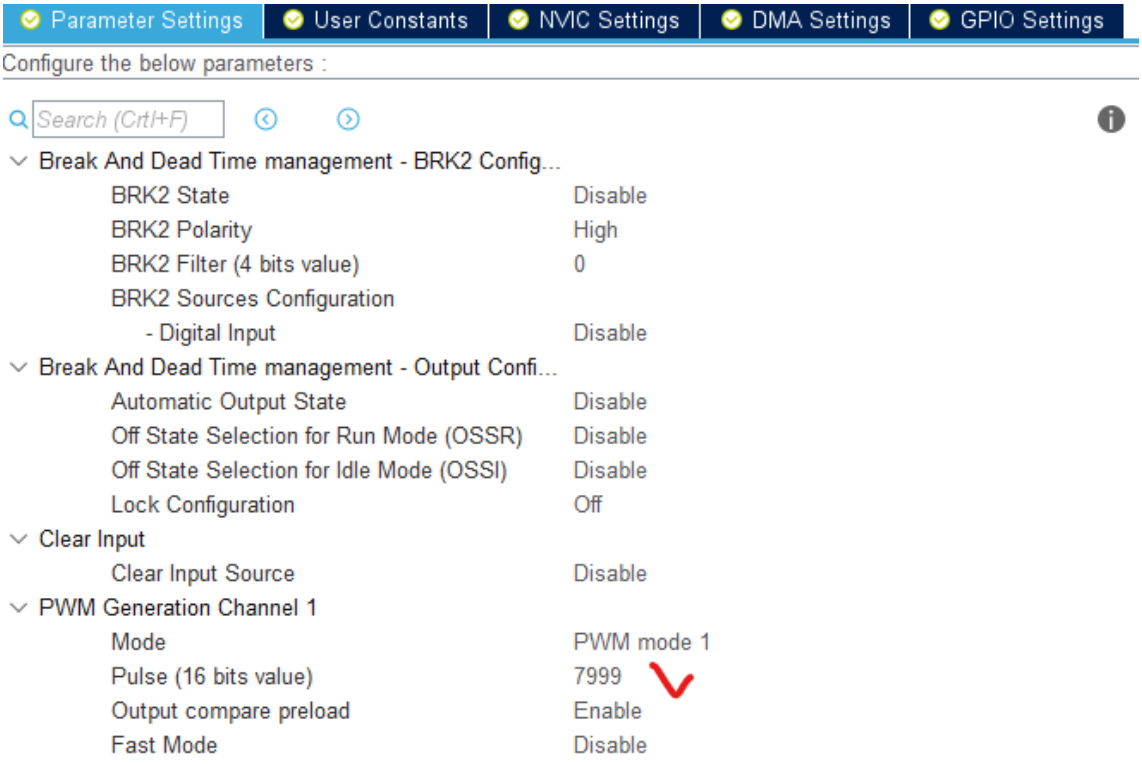

Fig 6. PWM pulse width settings

Take into account the GPIO Output speed settings.

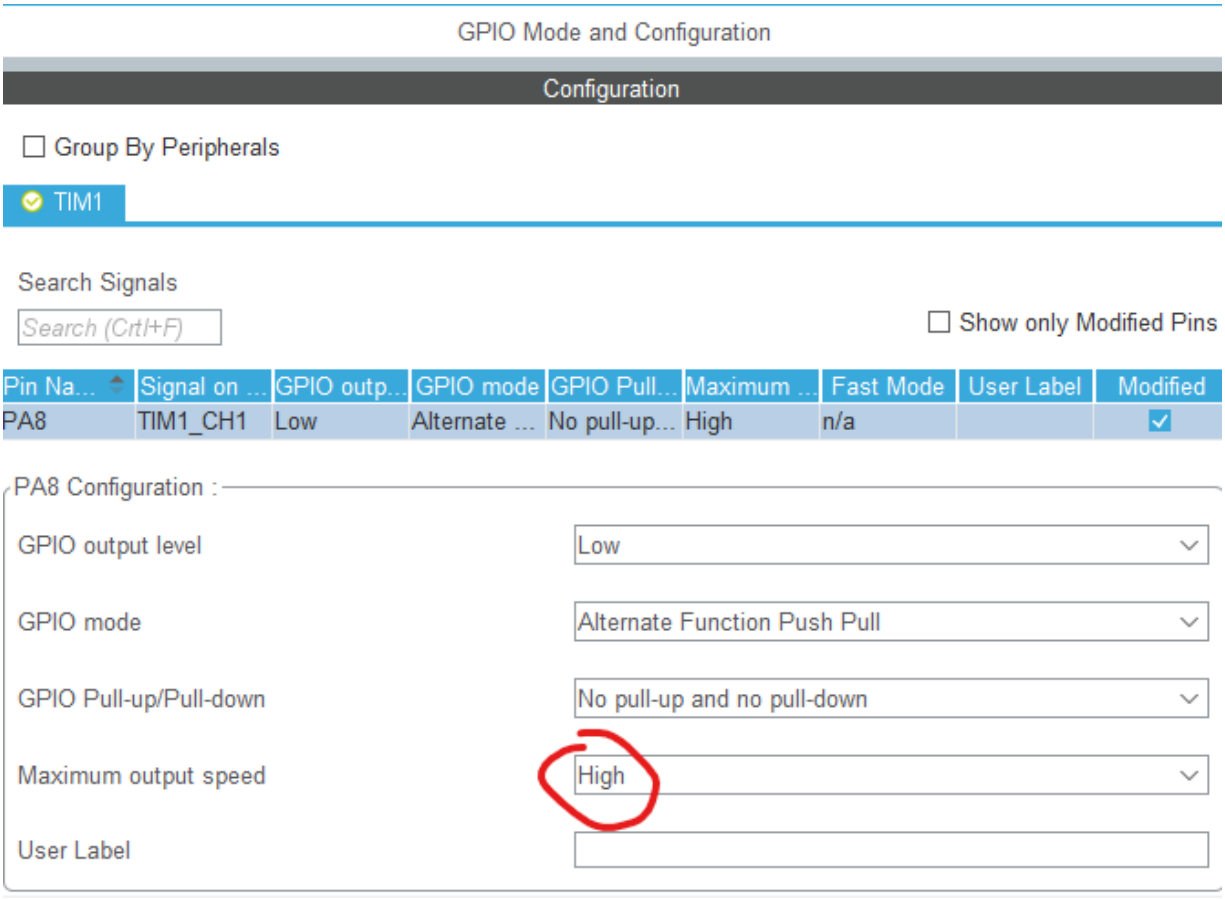

Fig 7. GPIO settings for PWM

To run PWM generation use code below.

```
/* USER CODE BEGIN 2 */

HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
```

PWM generate start function

3. Tasks

3.1. Create and set up STM32G070RB project.

3.2. Analyse Nucleo board schematic from DM00452640 pdf file.

3.3. Connect 3 LED's with a different color to the Nucleo board.

3.4. Write code for toggle 3 LEDs. Toggle one LED with 1Hz frequency inside of the while(1) loop and two other LEDs with 10 Hz and 20Hz using **two different** Timer's (TIM14 and TIM16) interrupts.

3.5. Write code for the check User button. If the button is pressed one-time LED is ON for about 10 sek and then OFF. If the button is pressed like "double click" LED must blink 5

seconds and after must be OFF. For time counting – please use Timer interrupt. You should use `HAL_TIM_Base_Start_IT` function.

3.6. Write code for changing two LED brightness using PWM (frequency 1 kHz and duty change every 20 mS, up from 20% to 100% and down).

4. Report content

1) Title.
2) Main blocks of source code for tasks with comments.
3) Conclusions.

5. References

1. https://www.st.com/content/st_com/en/products/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus/stm32-mainstream-mcus/stm32g0-series/stm32g0x0-value-line/stm32g070rb.html