Timers

rev1.0 24/03/2020

Blink the on-board LEDs using timers interrupts

PREREQUISITES

Software needed:

- STM32IDE

Hardware used in this example:

- **NUCLEO-F446ZE**

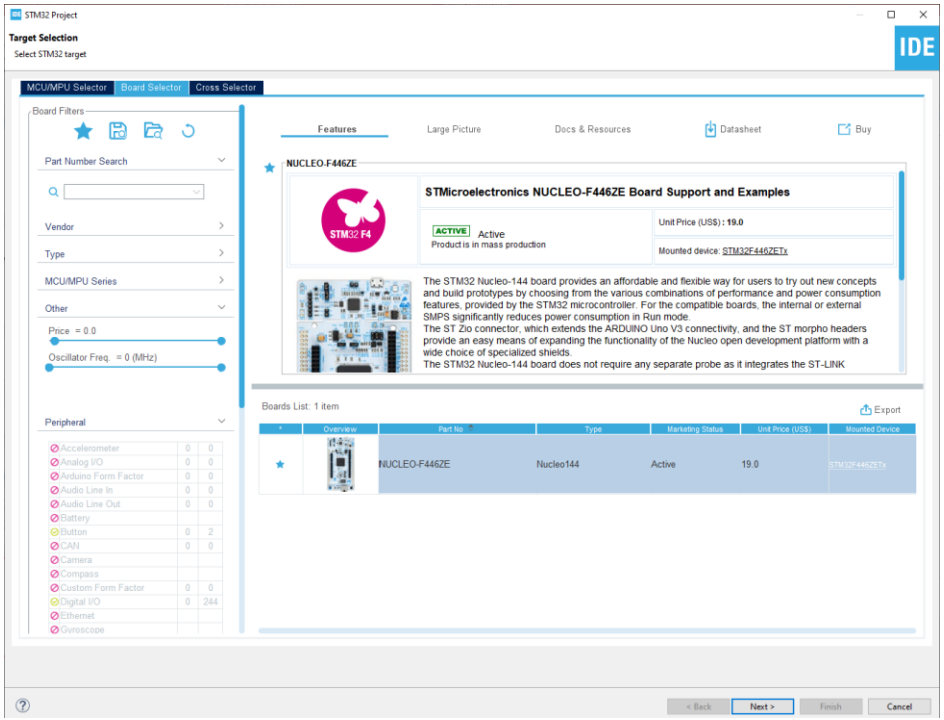
What is a Timer?

- The timer is a tool that allows the timing of some actions.
- They are very useful for having time references: it is important to note that they keep the date and time reference but only time intervals
- They can generate interrupts at regular time intervals
- Within the microcontroller there are several.

Start a new project

From the stm32IDE software click on File -> New -> STM32 Project.

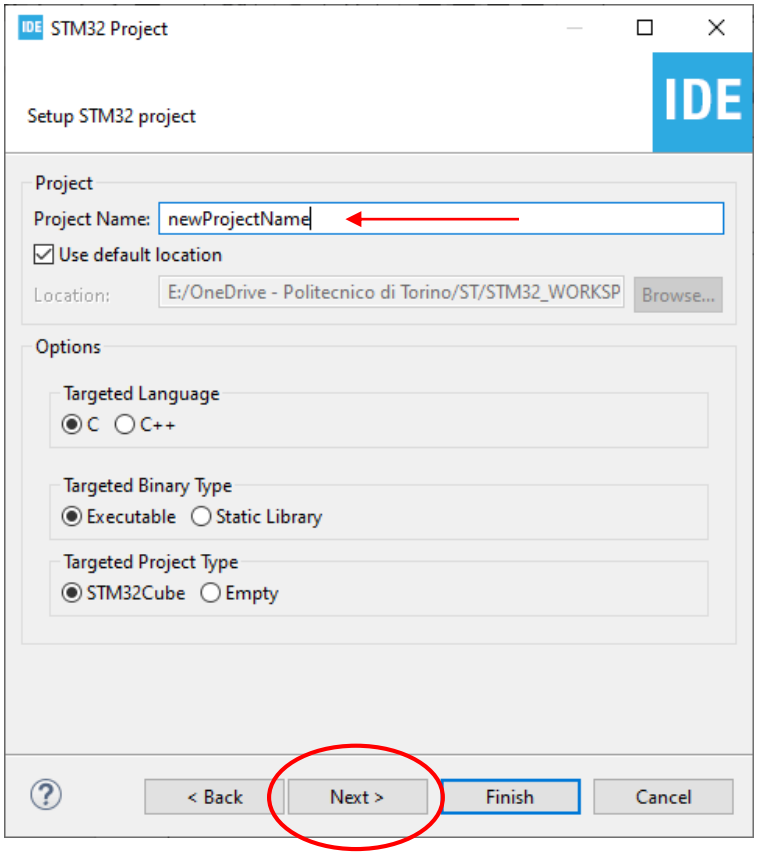
Select your board or your uC and click *next.*



Start a new project

Type the name of your project and click next.

By default the project will be created in the workspace folder.



Start a new project

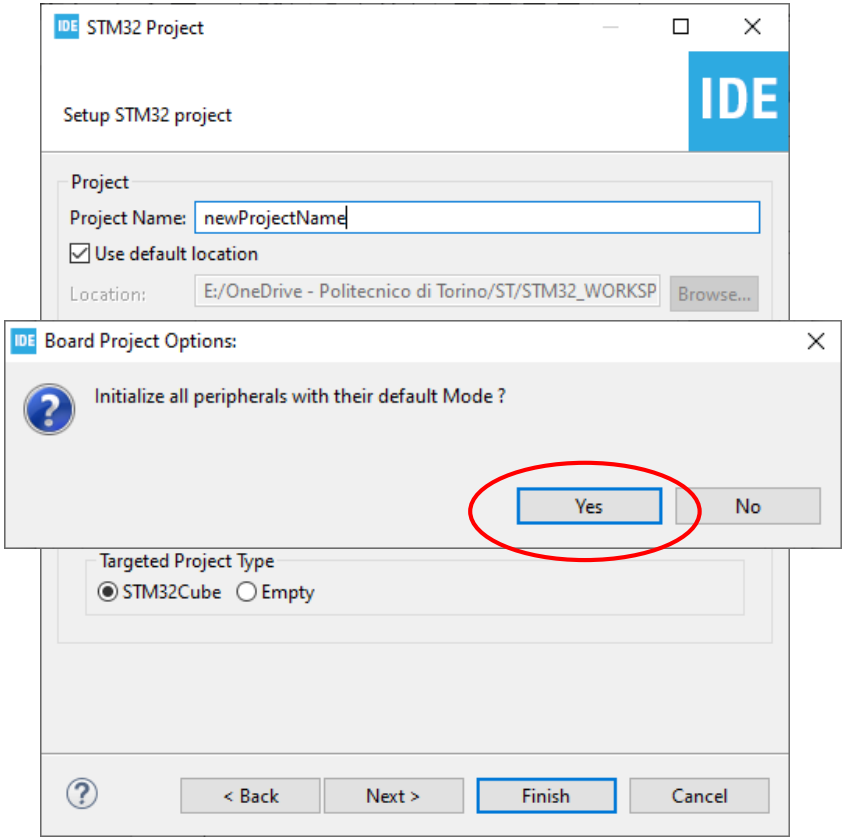
Type the name of your project and click next.

By default the project will be created in the *workspace* folder.

The *STM32IDE* has the option to initialize all the peripheral with their ***default*** mode:

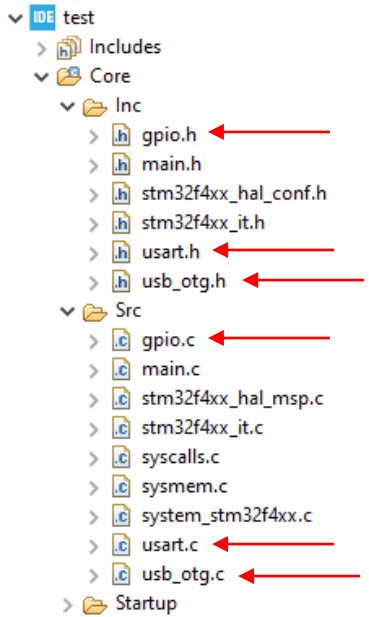
Clicking *Yes* the *USART3*, all the *LEDs* and the blue *UserButton* will be configured as default.

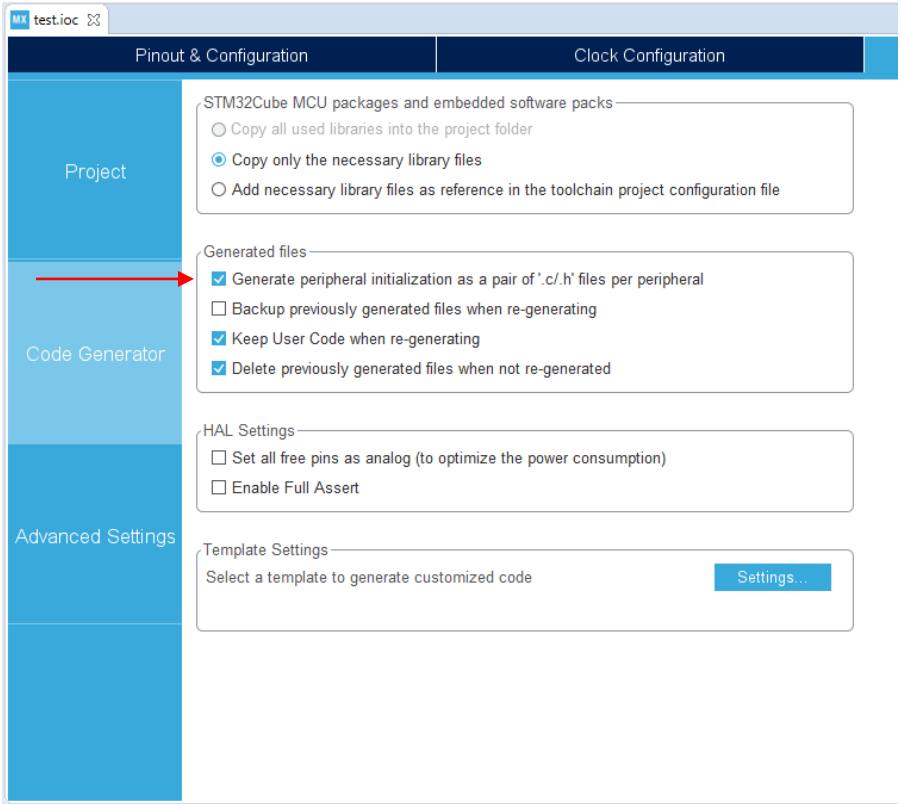
Click *Yes*.



Project Manager

In the Code Generator Tab check the ***Generate peripheral initialization […]*** box: each periperhal will have a disting *periph.c* and *periph.h* files.

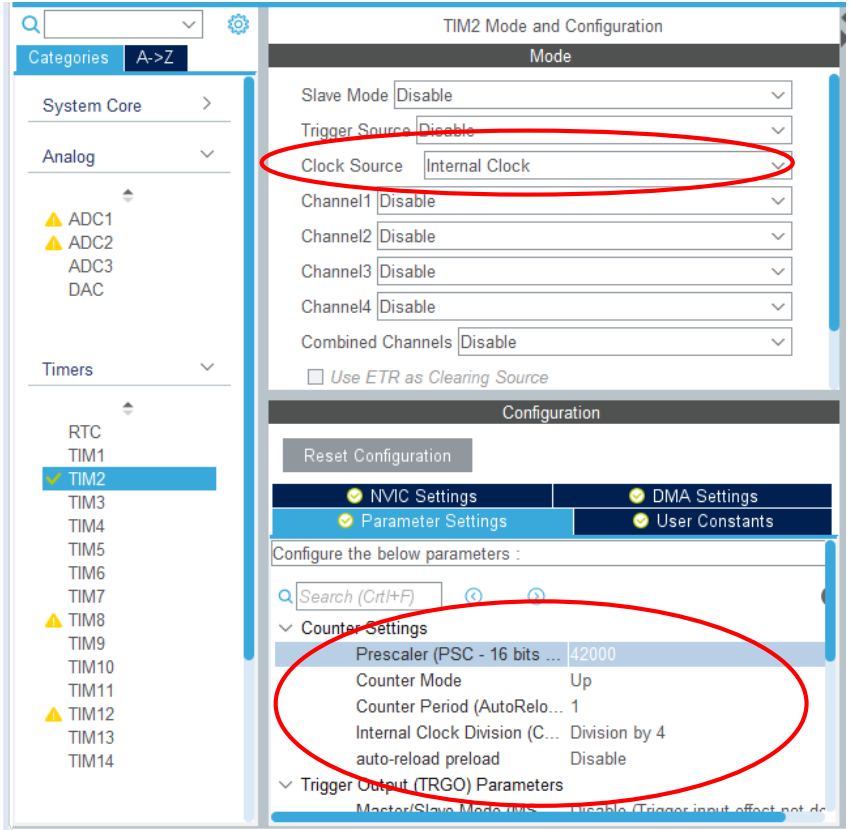




Timer Parameters

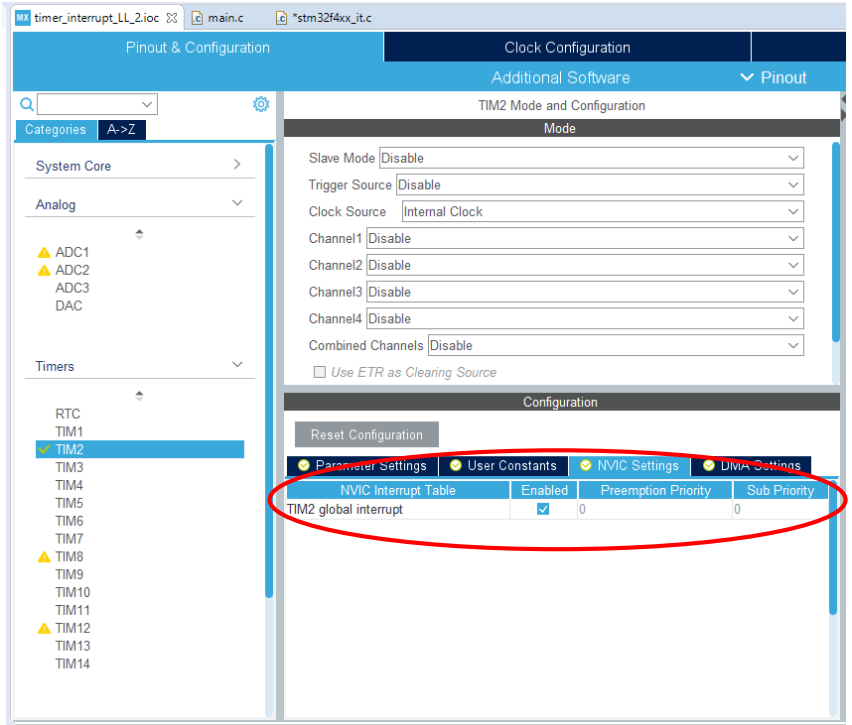
In this example we will use TIM2, the parameters to be set are as follows:

- *Clock Source*: is the clock reference of the timer
- *Prescaler (PSC):* is a divider of the timer activation frequency
- *Counter Period:* timer activation period



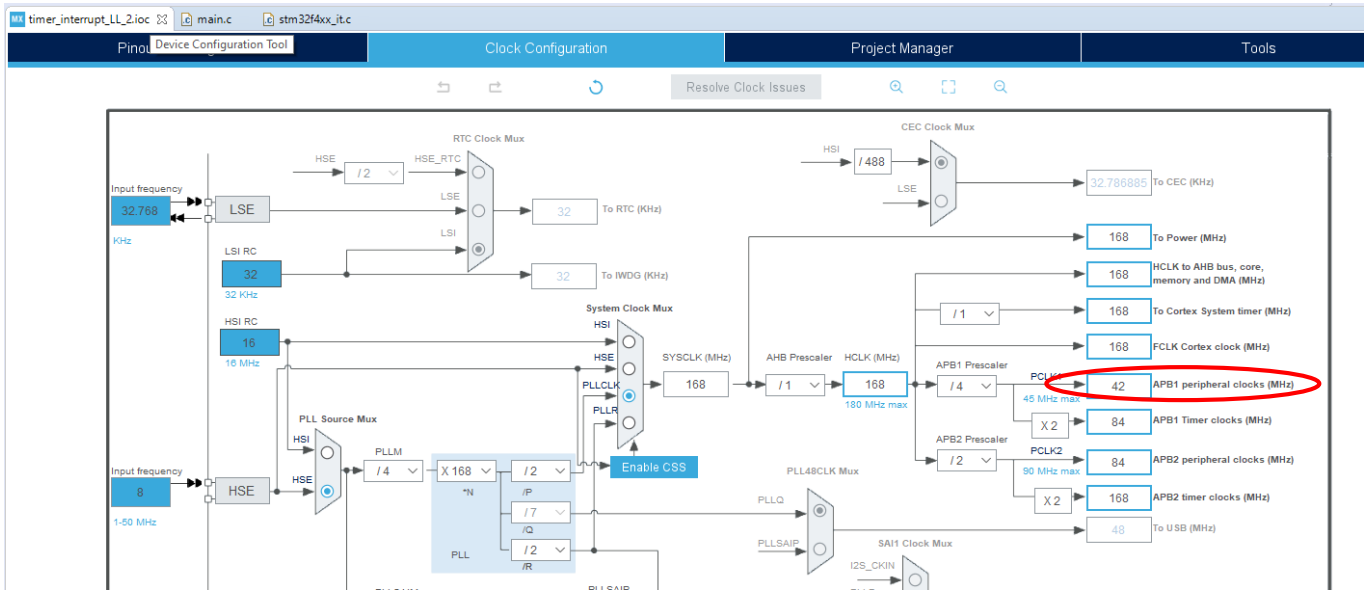
NVIC

- **NVIC** is fundamental for the generation of the interrupt to enable it and set its priority.
- Under the *NVIC tab* it is therefore necessary to tick the box for interrupt management



Prescaler

In our example the **Prescaler** has been set to *42000* as the **APB1** peripheral clock, to which **TIM2** also refers, is *42MHz*: in this way, having the Counter Period set to 1, the timer will be called every millisecond (42MHz / 42000 = 1000). At this point we just have to generate the code



Handler

- The first step is to check what happens as soon as the interrupt is generated: we then go to the Handler management within the **stm32f4xx_it.c** file.
- Just as in the previous cases, you check whether the interrupt was actually generated correctly, then reset the Flag and call the Callback function present in the main.c.

timer_interrupt_LL_2.ioc | main.c | *stm32f4xx_it.c

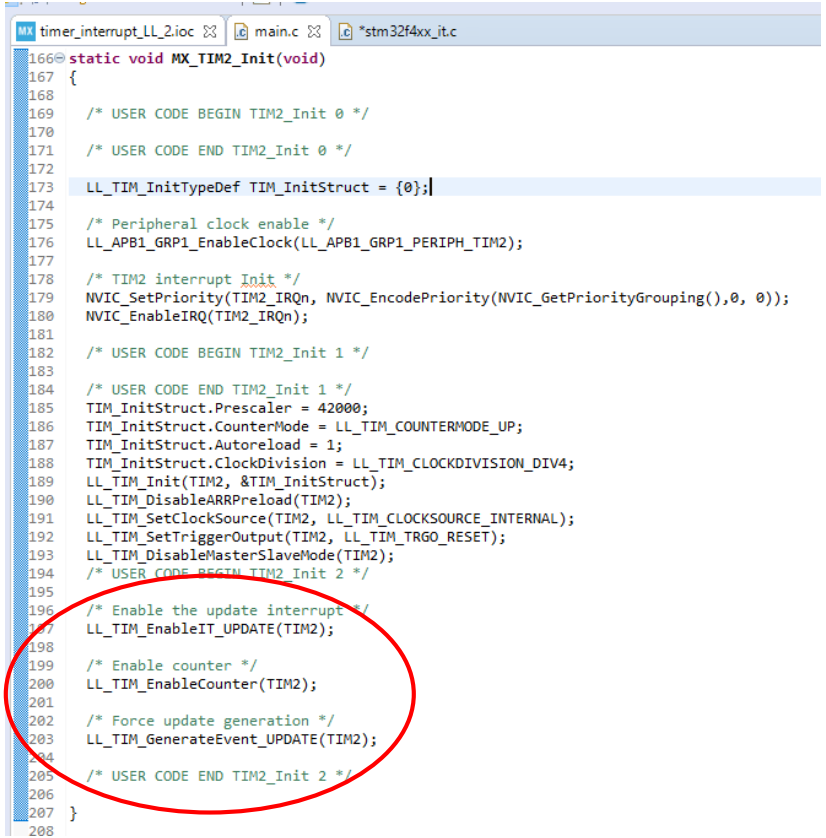
```
200 /**
201   * @brief This function handles TIM2 global interrupt.
202   */
203 void TIM2_IRQHandler(void)
204 {
205   /* USER CODE BEGIN TIM2_IRQn 0 */
206
207     /* Check whether update interrupt is pending */
208     if(LL_TIM_IsActiveFlag_UPDATE(TIM2) == 1)
209     {
210         /* Clear the update interrupt flag*/
211         LL_TIM_ClearFlag_UPDATE(TIM2);
212         /* TIM2 update interrupt processing */
213         TimerUpdate_Callback();
214     }
215
216     /* USER CODE END TIM2_IRQn 0 */
217   /* USER CODE BEGIN TIM2_IRQn 1 */
218
219   /* USER CODE END TIM2_IRQn 1 */
220 }
221
222 /* USER CODE BEGIN 1 */
223
224 /* USER CODE END 1 */
225 /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

Timer initialization

Before managing the interrupt, it must be enabled: for this reason, when the timer is initialized, it is necessary to:

- Enable the *interrupt*
- Enable the *counter*

Otherwise no interrupt will be generated.



Callback

Further down, in section 4, we can define the timer callback function:

Using the counter variable, it is possible to "count" how many interrupts have been generated: remember that we have set the timer parameters in order to generate an interrupt every millisecond.

Therefore only 1 second will pass after 1000 interrupts.

timer_interrupt_LL_2.ioc | *main.c | *stm32f4xx_it.c

```
297 /* USER CODE BEGIN 4 */
298 void TimerUpdate_Callback(void)
299 {
300   //LL_GPIO_TogglePin(LD1_GPIO_Port, LD1_Pin);
301   counter++;
302 }
303 /* USER CODE END 4 */
304
```

timer_interrupt_LL_2.ioc | *main.c | *stm32f4xx_it.c

```
102   /* Infinite loop */
103   /* USER CODE BEGIN WHILE */
104   while (1)
105   {
106     /* USER CODE END WHILE */
107      if (counter==1000) {
108           counter=0;
109           LL_GPIO_TogglePin(LD1_GPIO_Port, LD1_Pin);
110      }
111     /* USER CODE BEGIN 3 */
112   }
113   /* USER CODE END 3 */
114 }
```

Results

We load the program on the board and check that everything is working properly: Through the use of the oscilloscope we can note that exactly every second the LED changes its state, so we have a period of 2 seconds as we expected.