Interrupt with LL Libraries

rev1.0 24/03/2020

Generate an interrupt when the blue push button has been pressed

PREREQUISITES

Software needed:

- STM32IDE

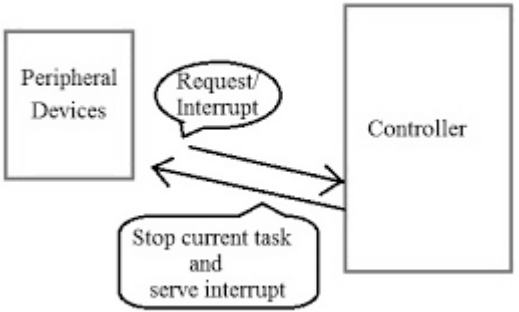
Hardware used in this example:

- **NUCLEO-F446ZE**

What is an Interrupt?

An ***interrupt*** is an asynchronous signal that allows you to manage situations in which particular attention is needed: there may be events such that it is necessary for the microcontroller to temporarily interrupt the normal execution of the program in order to execute a certain process.

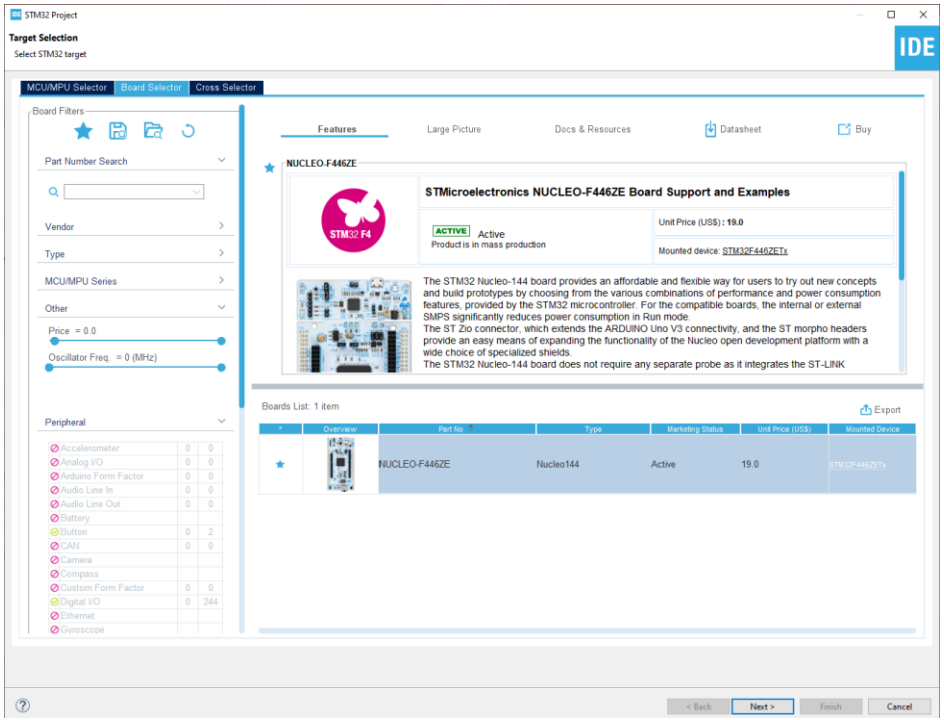
Examples that generate an interrupt may be the pressing of a button, rather than an event at regular time intervals (**timers**) or a certain input being exceeded by a certain threshold.



Start a new project

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

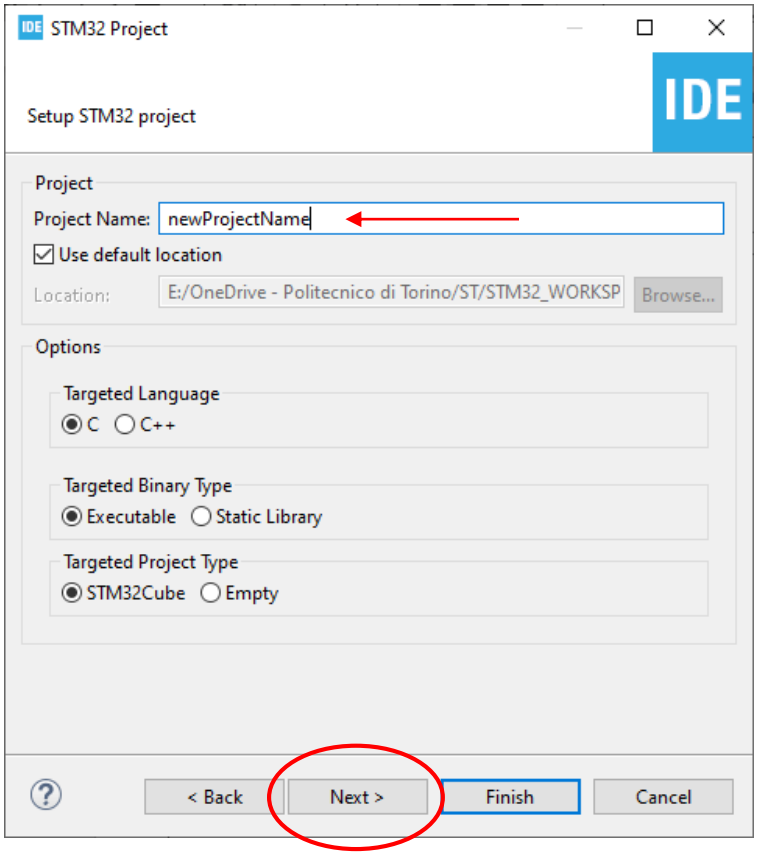
Select your board or your uC and click *next.*



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

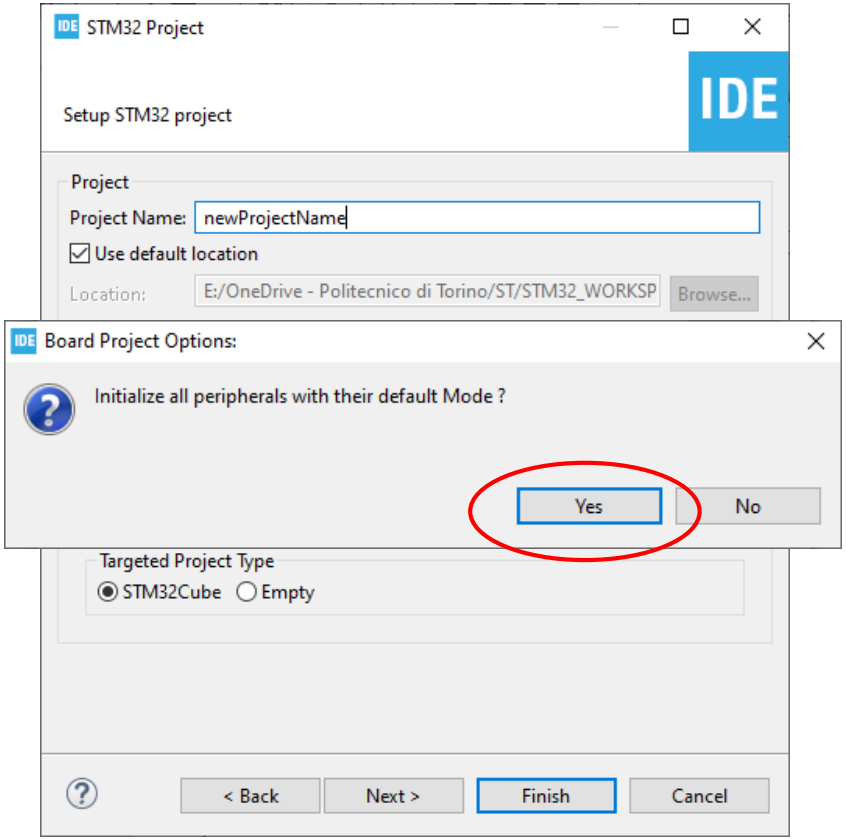
Type the name of your project and click next.

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:

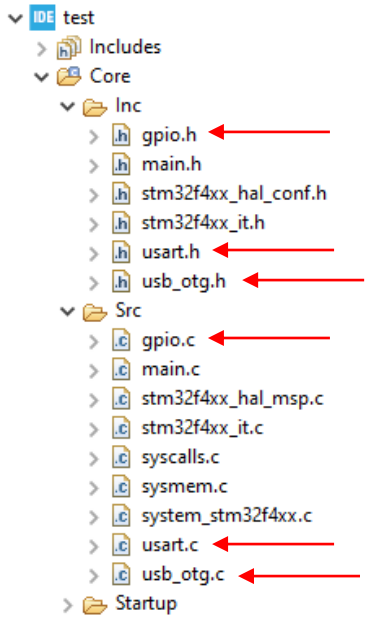
Clicking *Yes* the *USART3*, all the *LEDs* and the blue *UserButton* will be configured as default.

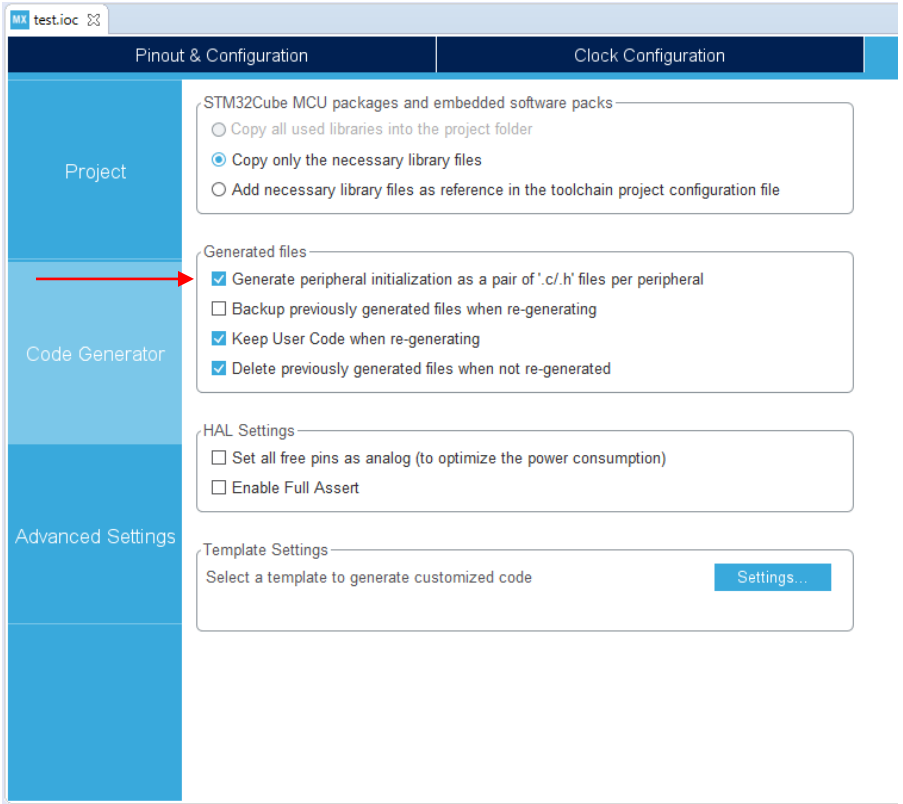
Click *Yes*.



Project Manager

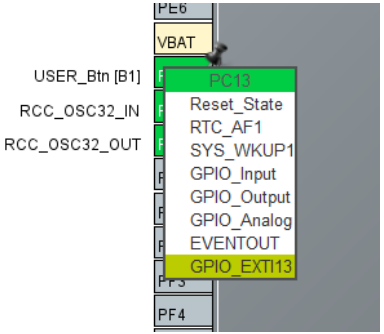
In the Code Generator Tab check the ***Generate peripheral initialization […]*** box: each periperhal will have a disting *periph.c* and *periph.h* files.

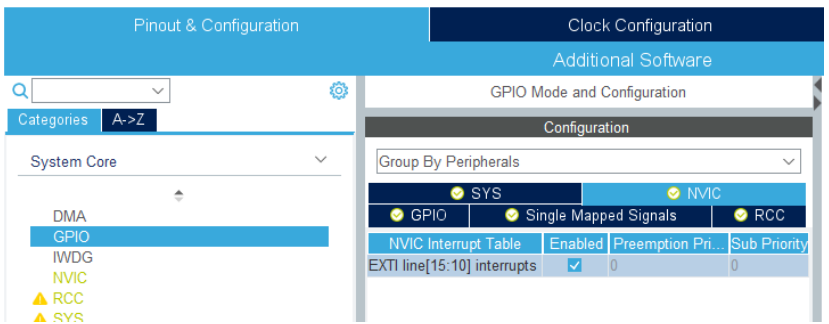




Impostare un GPIO come interrupt

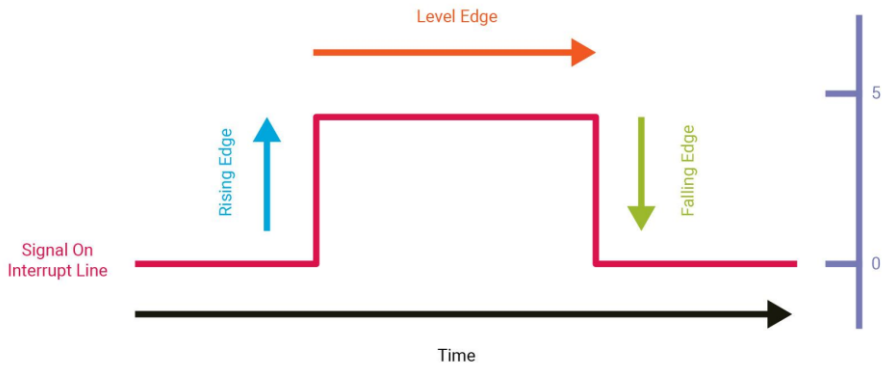
- From *CubeMX* it is necessary to set the *GPIO* pin as **GPIO_EXTIxx.**
- From the ***NVIC*** (Nested vector interrupt control) Tab we enable the interrupt: this causes *CubeMX* to worry about enabling the interrupt and setting its priority.

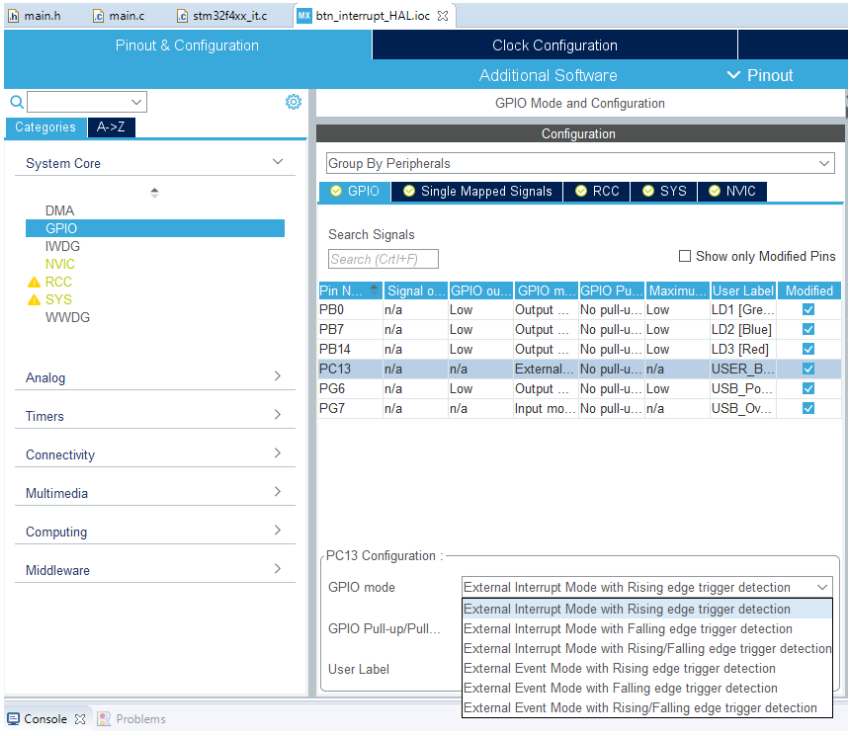




Falling & Rising edge detection

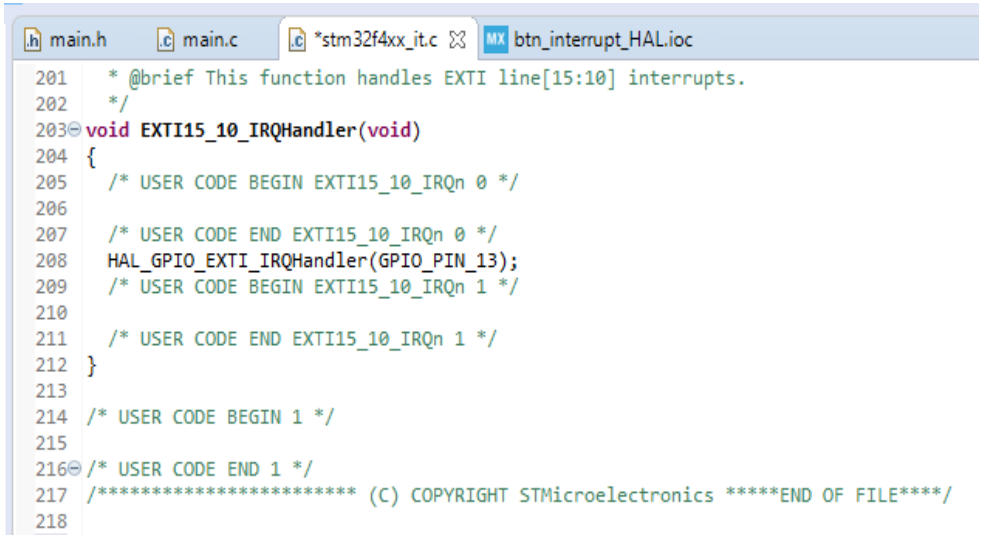
- From the GPIO card, on the other hand, it is possible to manage the interrupt mode, in particular choosing whether the interrupt should be generated when detecting the rising edge rather than the falling edge, or both.





- At this point we just have to generate the code and start writing our program. In this regard, we want to recreate the previous example but with the difference that this time when the button is pressed an ***interrupt*** will be generated.

- It is useful to note how *CubeMX* has already generated Handler for our interrupt. All Handlers are managed within the ***stm32f4xx_it.c*** file.
- The ***HAL_GPIO_EXTI_IRQHandler()*** will check that the interrupt was generated by the correct line.
- In that case, it will call the *Callback* function that we define in ***gpio.c*** file



Each time the button is pressed, our counter variable will then be increased following the generation of the interrupt.

```
242 /* USER CODE BEGIN 4 */
243 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
244     if (GPIO_Pin==USER_Btn_Pin) {
245         //HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin);
246         counter++;
247     }
248 }
249
```

- Now we just have to slightly modify the code that we wrote in the previous example: the function that took care of reading the button and increasing the counter has been eliminated, this is because now everything is managed through the interrupt.

main.h | main.c | *stm32f4xx_it.c | btn_interrupt_HAL.ioc

```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
      counter=0;
      int timer=USR_TIME; //max time for make the decision : press the button one or two times

      while(timer>0 && counter<2){
            timer--;
            HAL_Delay(1);//wait 1ms
        }

      switch (counter) {
        case 1: //blink led
            blink_once(BLINK_TIME,LD1_GPIO_Port, LD1_Pin);
            break;
        case 2: //LEDs on
            HAL_GPIO_WritePin(LD1_GPIO_Port, LD1_Pin, SET);
            break;
         default: //do nothing
            break;
      }
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

Debug

- To view the correct management of the interrupt, it may be useful to start the program in **debug** mode  , put a breakpoint inside the Callback function and start debugging  **.**
- When the button is pressed, it will take you to the function that manages the interrupt, as in the next figure.
- At this point it is possible to press resume again or move within the program using the *step arrows.*