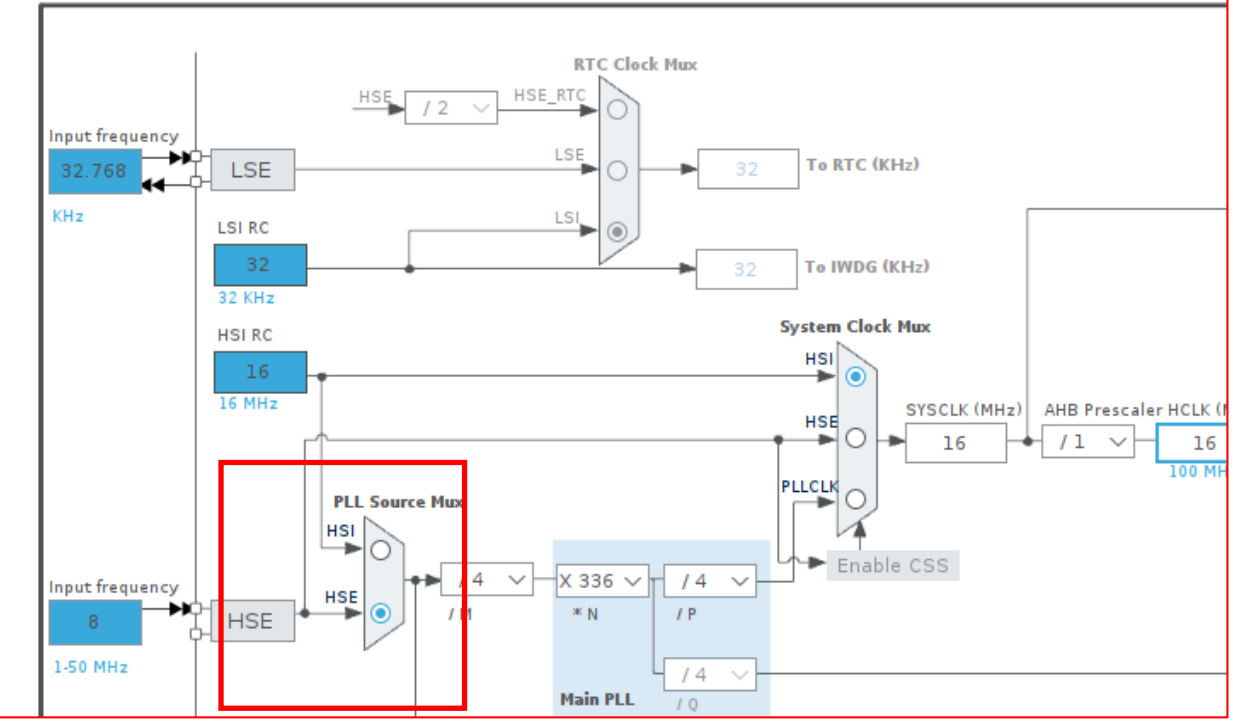
Lab 3-1 - Hint



Remark

Nucleo has two clocks inputs, HSI (internal clock) and HSE (external clock).

HSI is convenient for MCU because you can power board without external clock components. However, the internal clock relies on internal RLC circuit, which is not accurate. If you use it, you will see jittering in the oscilloscope.

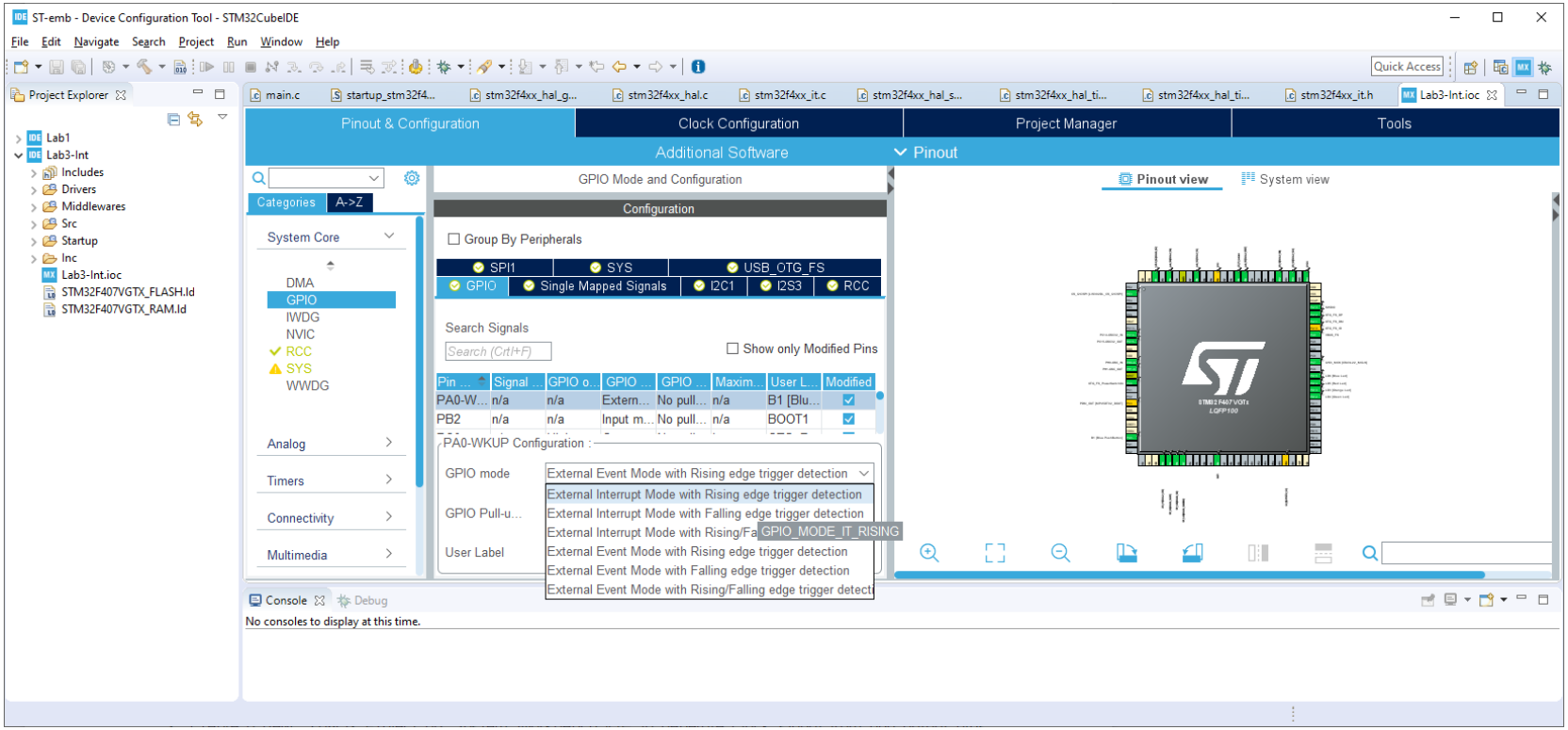
HSE (in Nucleo/Discovery) uses clocks from an external crystal. This can be much more accurate.

Use only HSE in this lab by going to “Clock Configuration” and choose HSE. If you change clocks speed (SYSCLK), make sure that it is still using HSE.

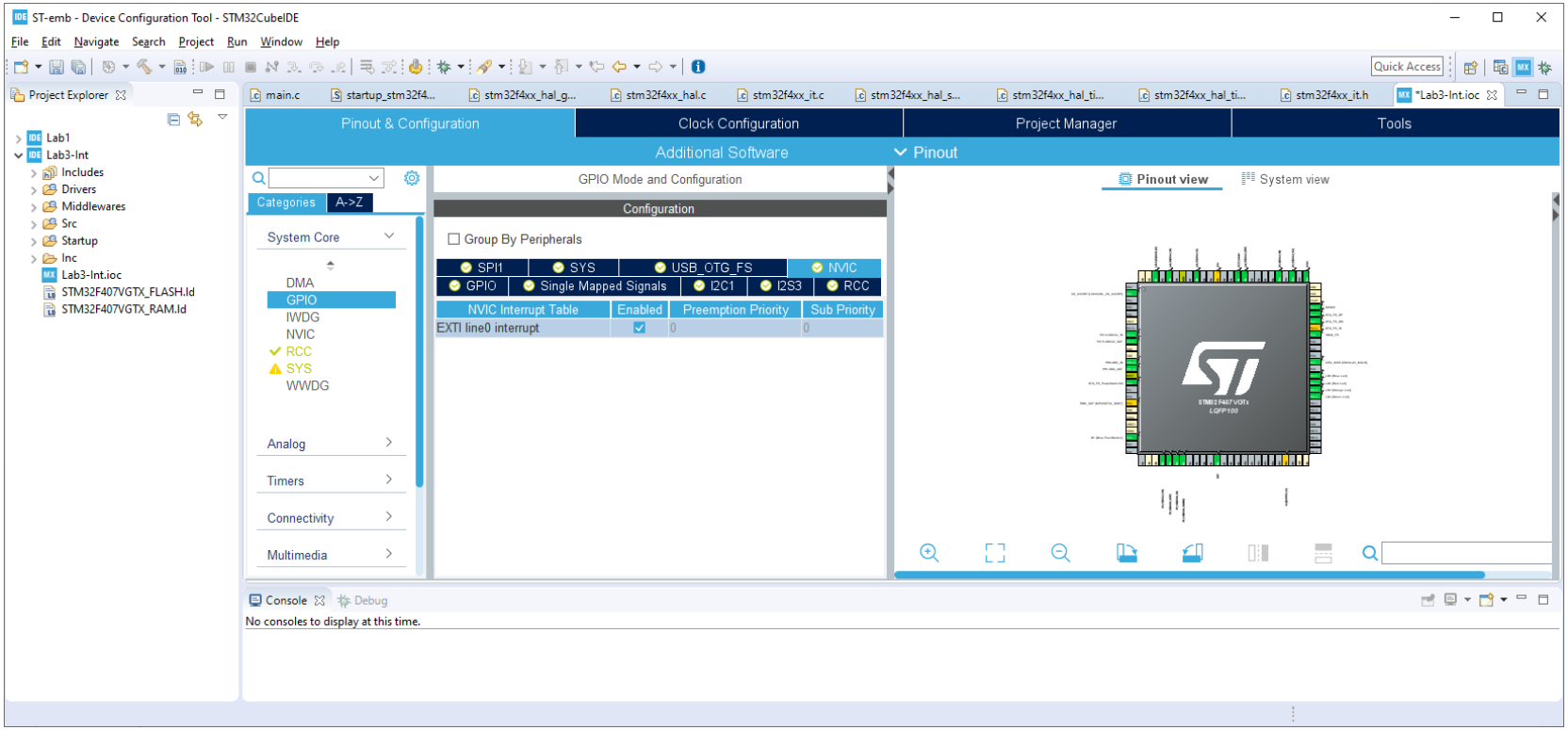
Lab 3 – 1 – To Do

- GPIO Generate Interrupt
- Processor received the Interrupt
- Change Priority of the Interrupt so that HAL_Delay works (this makes deboucing much easier)
- Using GPIO Interrupt
- Write code to change the LED “blinking”
  - Use Timer to Generate PWM Signal for LED
  - Setting through GPIO

GPIO Generate Interrupt



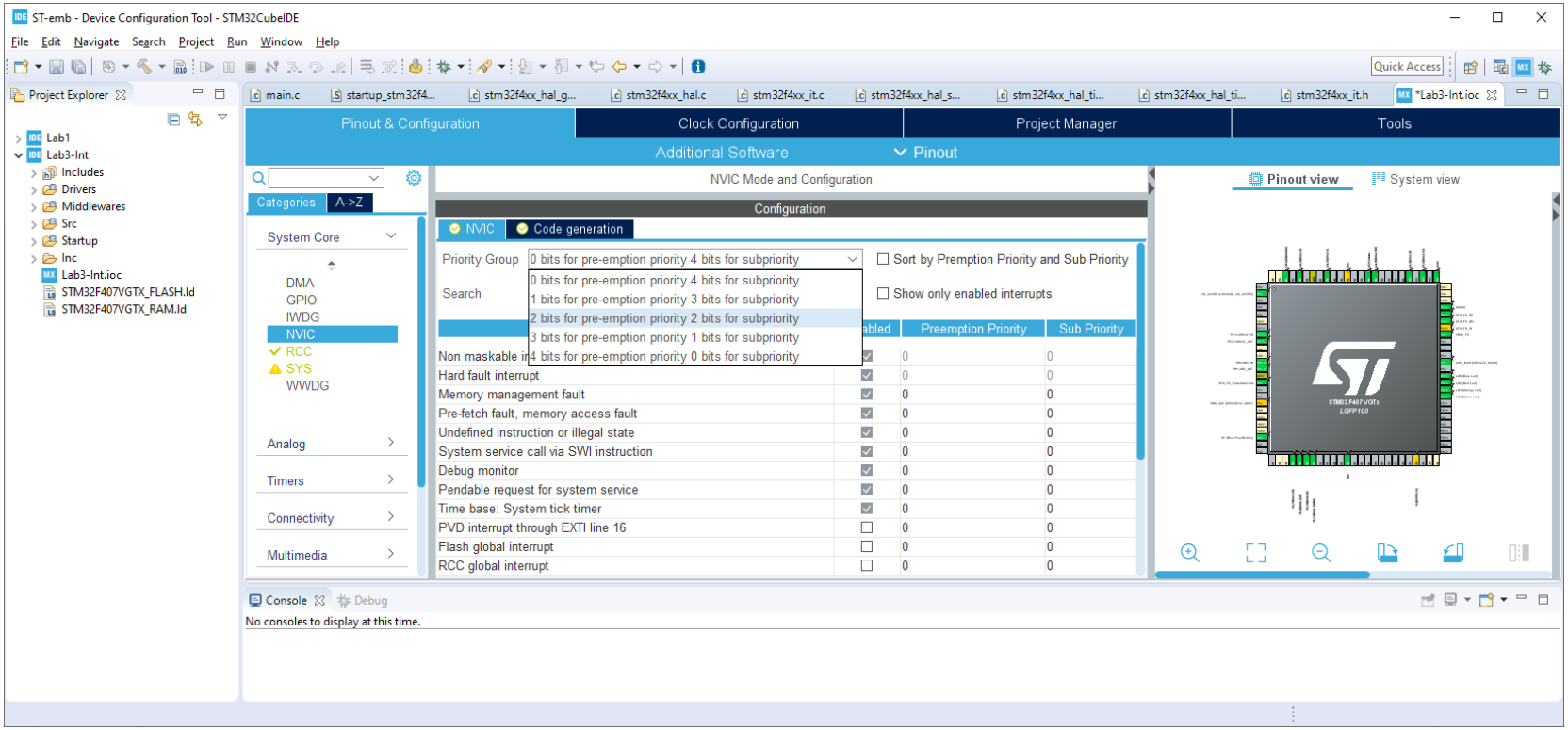
Processor received the Interrupt



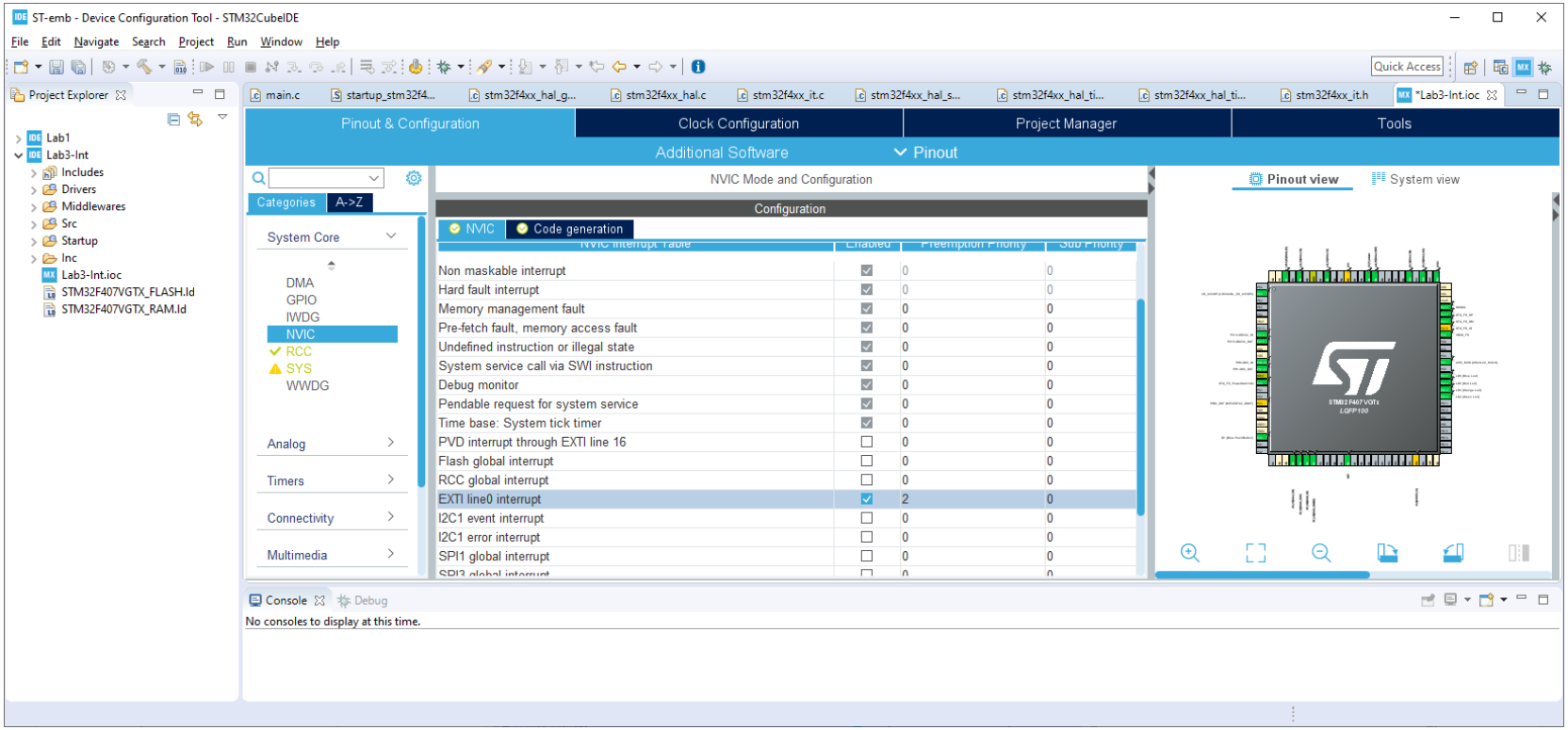
Change Priority of the Interrupt

- All interrupts, by default, has the same priority
- HAL_Delay uses timer from SysTick interrupt, which means that you will not be able to HAL_Delay in the interrupt unless SysTick has a higher priority

Change Priority of the Interrupt: Change preemptive



Change Priority of the Interrupt:
Make our GPIO interrupt with lower priority (higher number – lower the priority)



Using GPIO Interrupt

- In stm32f4xx_it.c, you will see a new interrupt

```
/**
  * @brief This function handles EXTI line0 interrupt.
  */
void EXTI0_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI0_IRQn 0 */

  /* USER CODE END EXTI0_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0);
  /* USER CODE BEGIN EXTI0_IRQn 1 */

  /* USER CODE END EXTI0_IRQn 1 */
}
```

This function called

```
HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0);
```

Using GPIO Interrupt

- Following **HAL_GPIO_EXTI_IRQHandler** (in stm32f4xx_hal_gpio.c), you will see that it calls HAL_GPIO_EXTI_Callback(GPIO_Pin)

```
void HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin)
{
  /* EXTI line interrupt detected */
  if(__HAL_GPIO_EXTI_GET_IT(GPIO_Pin) != RESET)
  {
    __HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);
    HAL_GPIO_EXTI_Callback(GPIO_Pin);
  }
}
```

HAL_GPIO_EXTI_Callback is declared in this file as weak, which means that you can redeclare this in other files

Using GPIO Interrupt

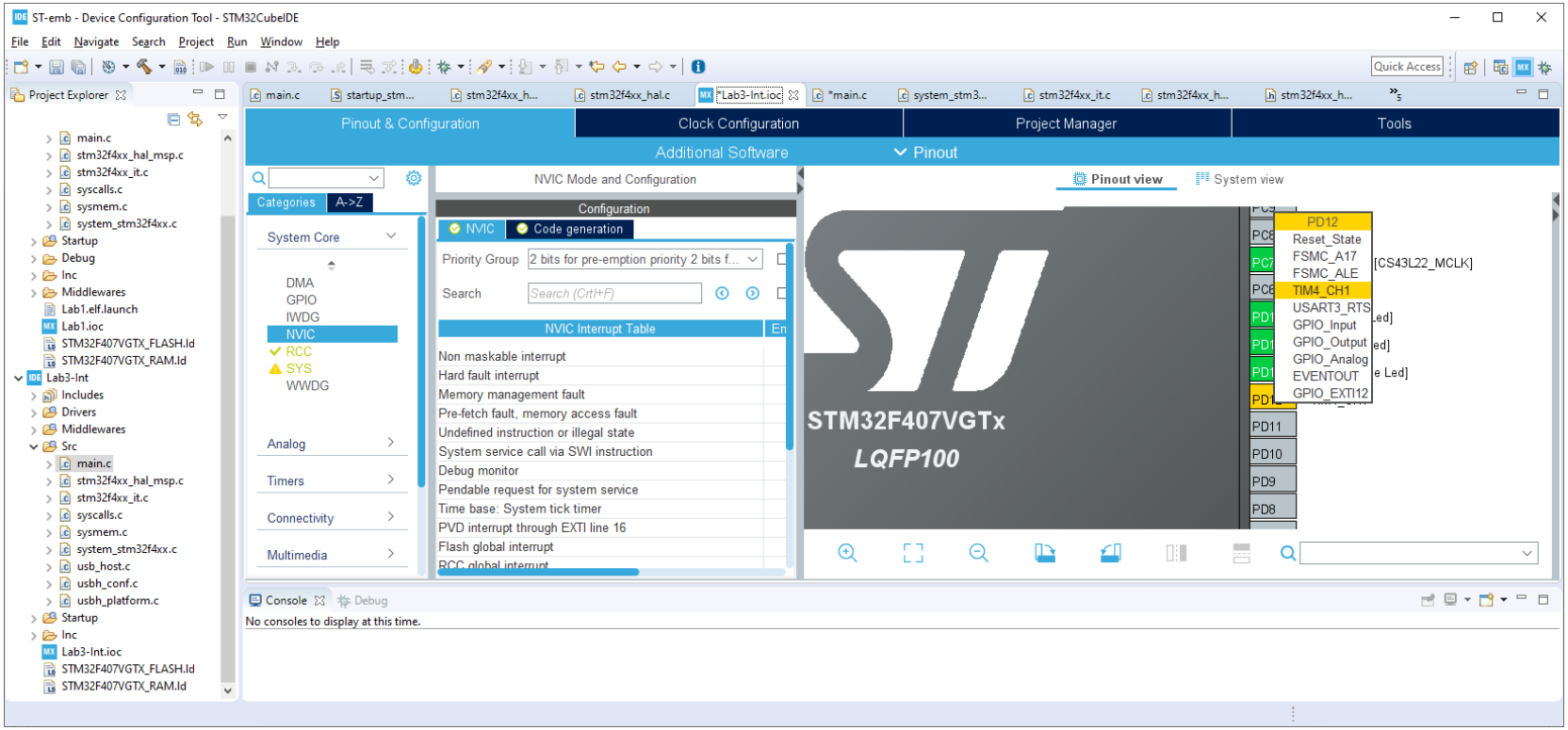
- In order to test your interrupt, you can add the following code in main.c

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) {
        HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_15);
        HAL_Delay(100);
        __HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);
}
```

Setting up PWM output Timer

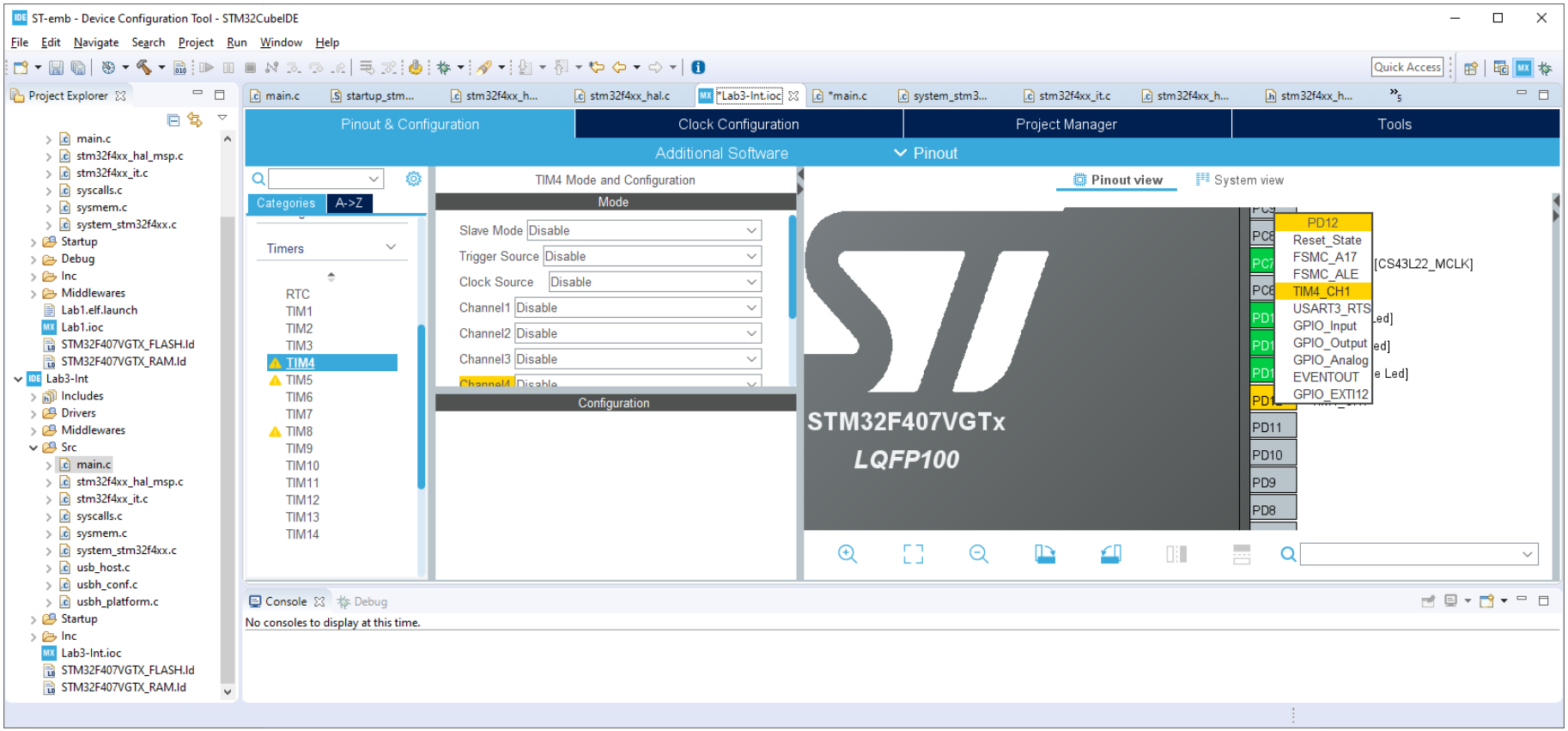
- See document in timer.pptx (in MCV) to see how timer work
- Choose a PIN that has timer output. You can see this in the STM32F4 Datasheet. In your board, all the LEDs are connected to a Timer

Setting up PWM output Timer: Set pin as alternative function for timer



Setting up PWM output Timer:

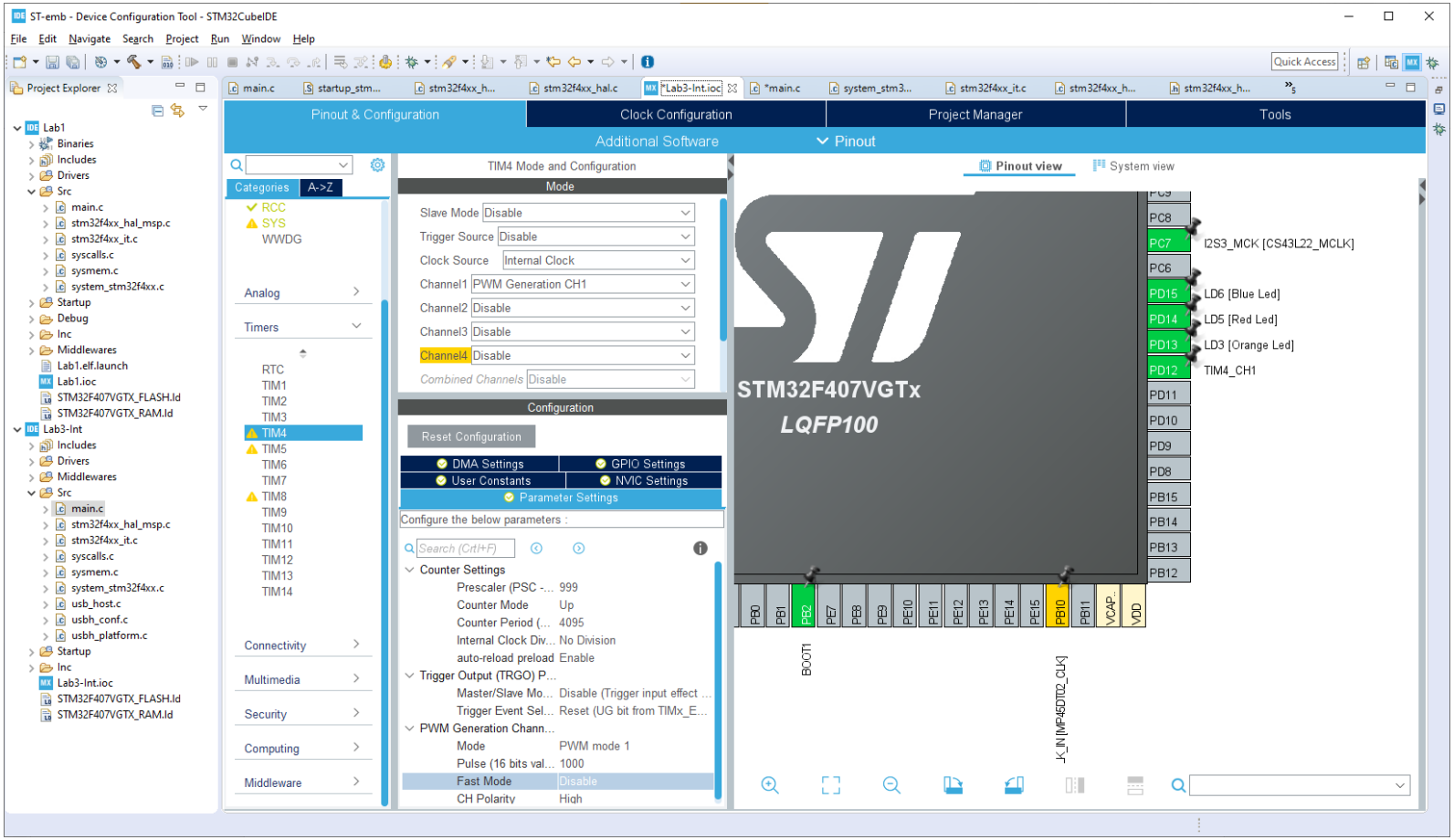
- Since we are using TIM4 (in the pin), go to Timer-> TIM4



Setting up PWM output Timer:

- Set the following
  - Clock Source : Internal Clock
  - Channel 1 (because we are using): PWM Generation CH1
  - Prescaler : Change to appropriated values (see the other documents)
  - Counter Period : Change to appropriated values (see the other documents)
  - Auto-reload preload : Enable
  - PWM Generation Channel 1 -> Pulse : Change to appropriated values (see the other documents)

Setting up PWM output Timer: Example



Starting Clock/PWM generation

- Add the following after MX_TIM4_Init() in main.c

```
  /* USER CODE BEGIN 2 */
    HAL_TIM_PWM_Start(&htim4,
TIM_CHANNEL_1);
    /* USER CODE END 2 */
```

Clock/ PWM registers

- TIMx->CNT - Clock counter (if you monitor this, it should always change)
- TIMx->PSC - Prescaler values
- TIMx->ARR - Period values
- TIMx->CCR1 – PWM for channel 1 (replace 1 with other value for other channels)

Your task

- Make debouching changed the values of PSC, ARR, and CCR1 to appropriated values