Use of GPIO on Microcontroller

Targeted competences: Use of GPIO in input and output mode.

Hardware: STM32F7 Nucleo board

Framework: Keil µvision and CubeMX from STMircoelectronics

The aim of this example is to study how use GPIOs as input or output. Here, we use one GPIO in input mode in order to detect if this GPIO has a high or low voltage level. If the level is high then we switch the GPIO in output mode at a low voltage level or at the high voltage if the input level is low.

Instead of realize a pooling onto the input GPIO to know its voltage level, we will use a timer to generate a sampling period. At each sampling period, we will read the voltage level on the input pin and we will realized the action onto the output pin.

The first step is to create the CubeMx project for the board that we will use. Here we use the NUCLEO STM32F767ZI board. We choose to use the PD7 pin in GPIO input mode and the PC8 pin in output mode (See Figure 1).

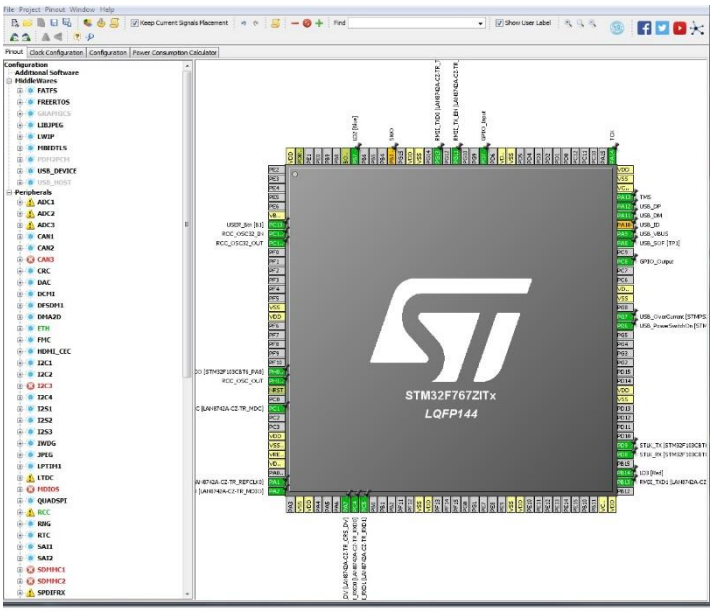

FIGURE 1: CONFIGURATION OF THE GPIO PIN

Then the second step is to select a timer in order to realize the sampling period. I chose to use the timer 1 and I configure it as shown in Figure 2 (just select the clock source).

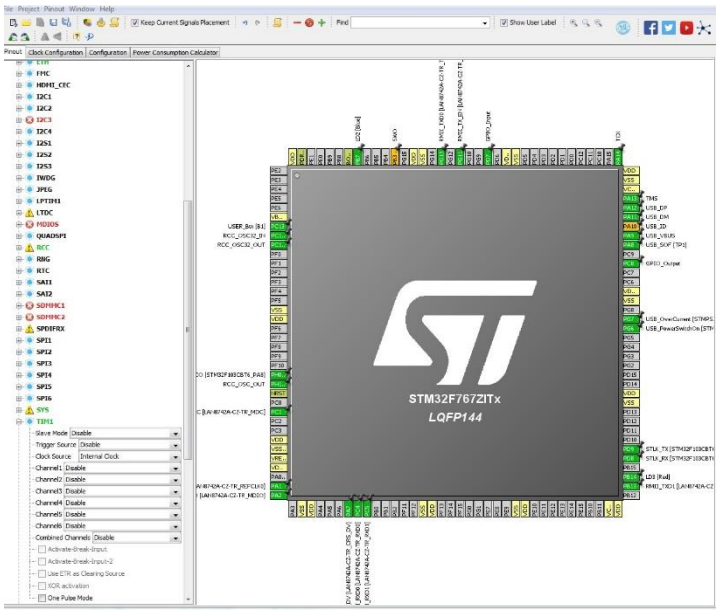

FIGURE 2: TIMER CONFIGURATION ON CUBEMX

Now, we have to configure the clock tree for the board, in this example I chose to use the maximum frequency of the CPU (216MHz); the configuration can be found in Figure 3.

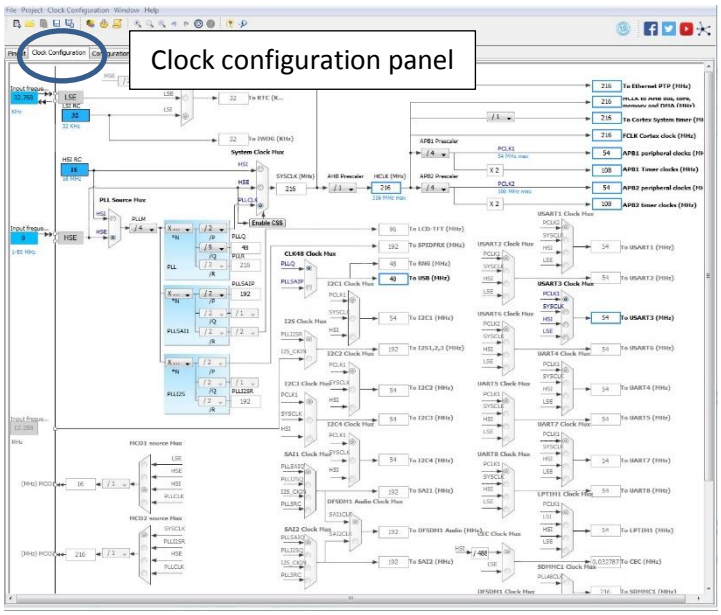

FIGURE 3: CONFIGURATION OF THE CLOCK TREE OF THE BOARD

Next step is to configure the GPIO pin; for that click and the GPIO panel onto the configuration panel of the CubeMx tool. Here, I give a name for each pin that I will use later. The PD7 pin is named Input_Pin and the PC8 is named Output_Pin (see Figure 4). By clicking on ok the modification will be taken into account.

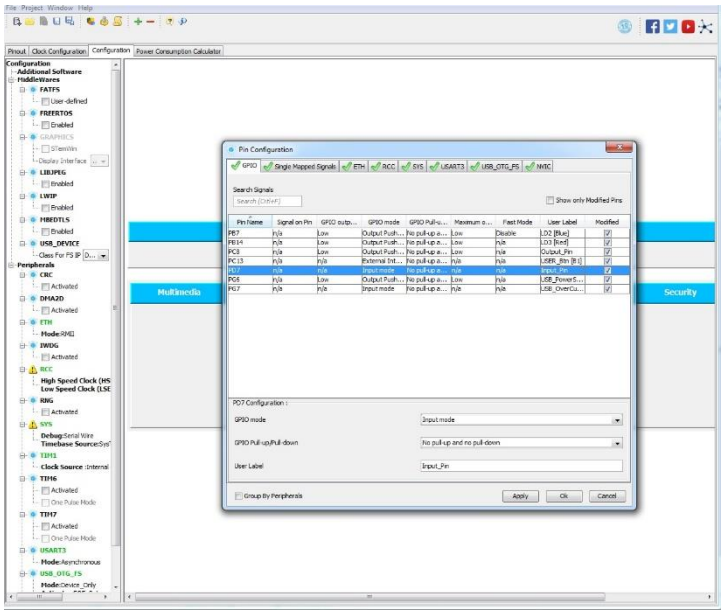

FIGURE 4: GPIO PIN CONFIGURATION IN CUBEMX

The last step of the configuration is to configure the timer 1; to do that click on the timer1 panel. Now we want to scan the input pin all the second so, we have to configure the right value in the Prescaler and in the Counter Period. The base time of the timer is compute with the frequency of the bus timer and the value of the prescaler. The timer 1 is connected to the APB2 bus so here the frequency of the bus is 216Mhz. So the counter period will be calculated by the following equation:

$$sampling\ period = \frac{1}{\frac{F_{timer}}{Prescaler\_value}} * Counter\ period$$

As F_{timer} is equal to 216Mhz and as the prescaler value is chosen to be equal to 65000 then the counter period will be equal to 3334 in order to obtain a sampling period of 1 second. The configuration of the timer is shown in Figure 5.

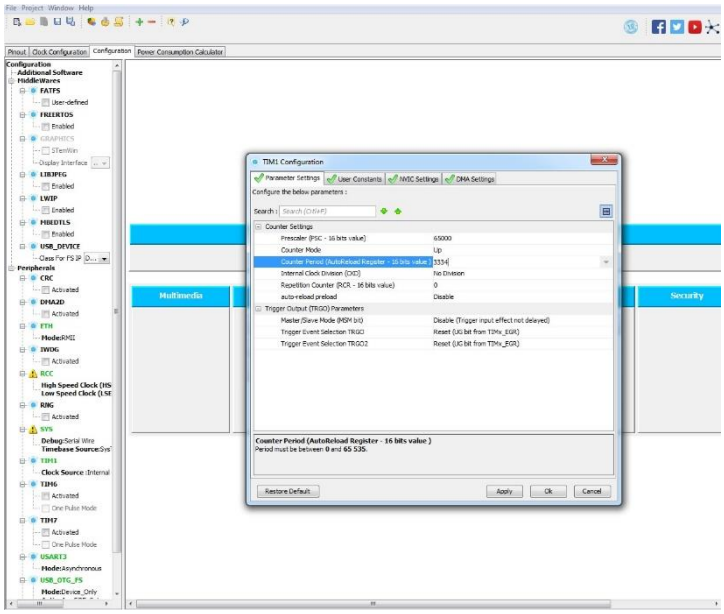

FIGURE 5: TIMER CONFIGURATION IN CUBEMX

Now we are going to configure the timer in order to obtain an interruption when the timer reaches a period of 1 second. To do that we have only to enable the TIM1 update interrupt as shown in Figure 6.

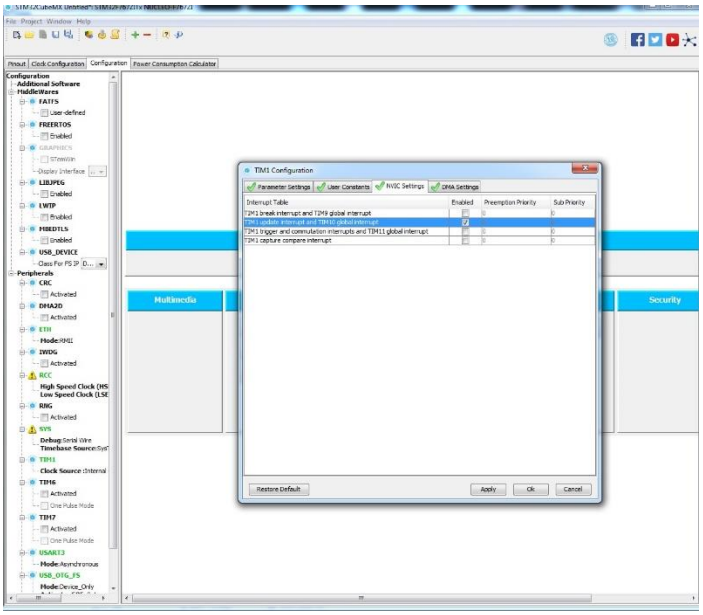

FIGURE 6: TIMER INTERRUPT CONFIGURATION

As we have only one interruption in this code we do not need to configure the NVIC (Nested Vector Interrupt Controller) but if in your code, you have more than one interruption it is necessary to do it in order to have priorities between the different interruptions.

Now, we are going to generate the project code; to do that click on generate source code and configure the generator as presented in Figure 7.

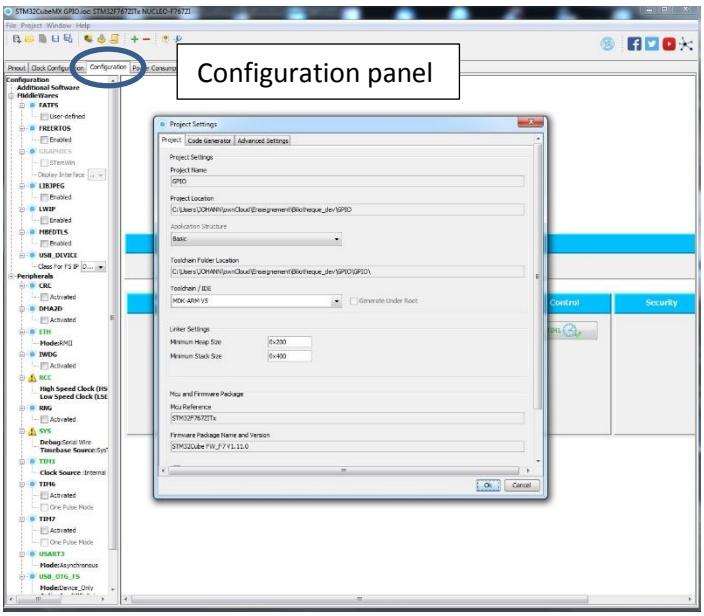

FIGURE 7: CONFIGURATION OF CODE SOURCE GENERATOR

When the code is generated, the MDK-ARM framework opens and we can see that the different source codes have been created (see Figure 8).

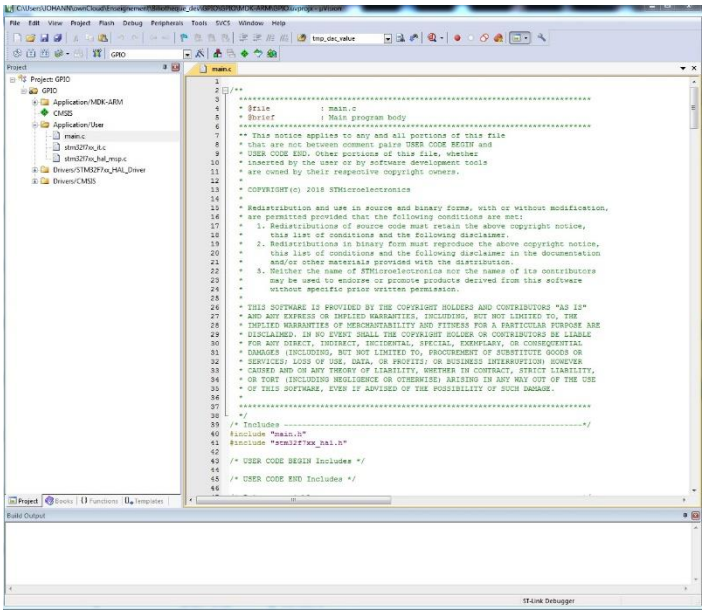

FIGURE 8: MDK-ARM FRAMEWORK

The second part of the development is to write the application code. Here, we want to check the value of the input GPIO (2 states: '1' or '0' level) and in function of this value the output GPIO will be set or reset. The first step is to start the timer 1 in order to obtain a sampling period of 1 second; this step is realized by starting the timer 1 in interrupt mode as shown in Figure 9. All the Timer functions API can be found in the HAL_tim.c file.

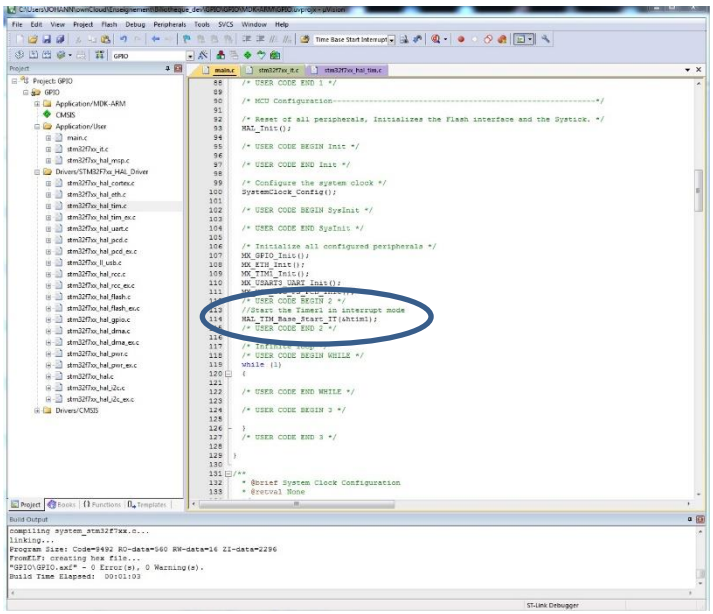

FIGURE 9: LAUCHING OF THE TIMER

Now, the timer will generate an interruption every second so we have to write the code in order to check the value of the input pin and write the right value on the output pin. These functionalities have to be coded in the Timer_1 interruption routine. I created a variable to store the state of the input pin thus, I can know, with this variable, if the output pin must be set or reset. The associated code shown in Figure 10 & 11.

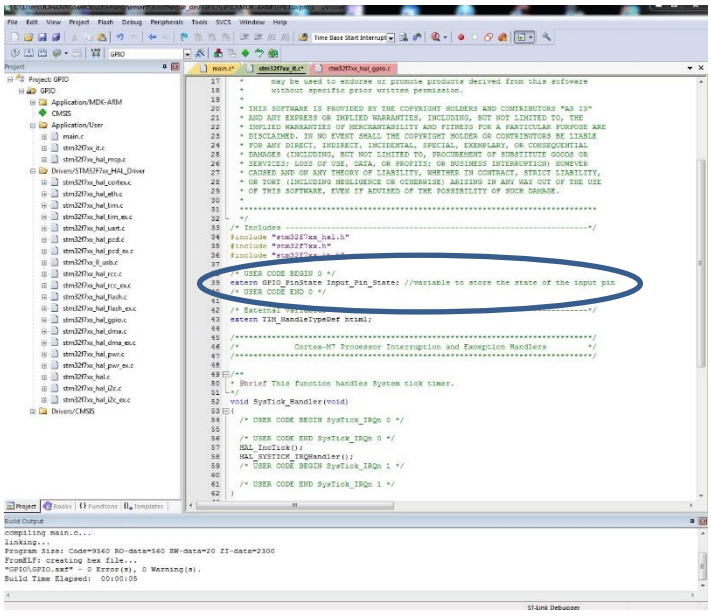

FIGURE 10: DECLARATION OF THE VARIABLE INPUT_PIN_STATE TO KNOWN THE STATE OF THE INPUT PIN

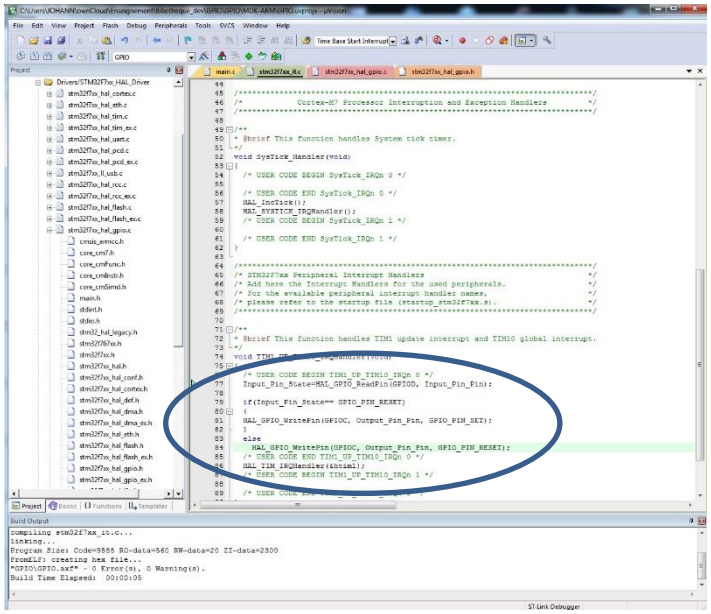

FIGURE 11: CODE TO SET OR RESET THE OUTPUT PIN IN FUNCTION OF THE STATE OF THE INPUT ONE

A video of the code behavior can be found here: https://youtu.be/hMhh4VqM4JU