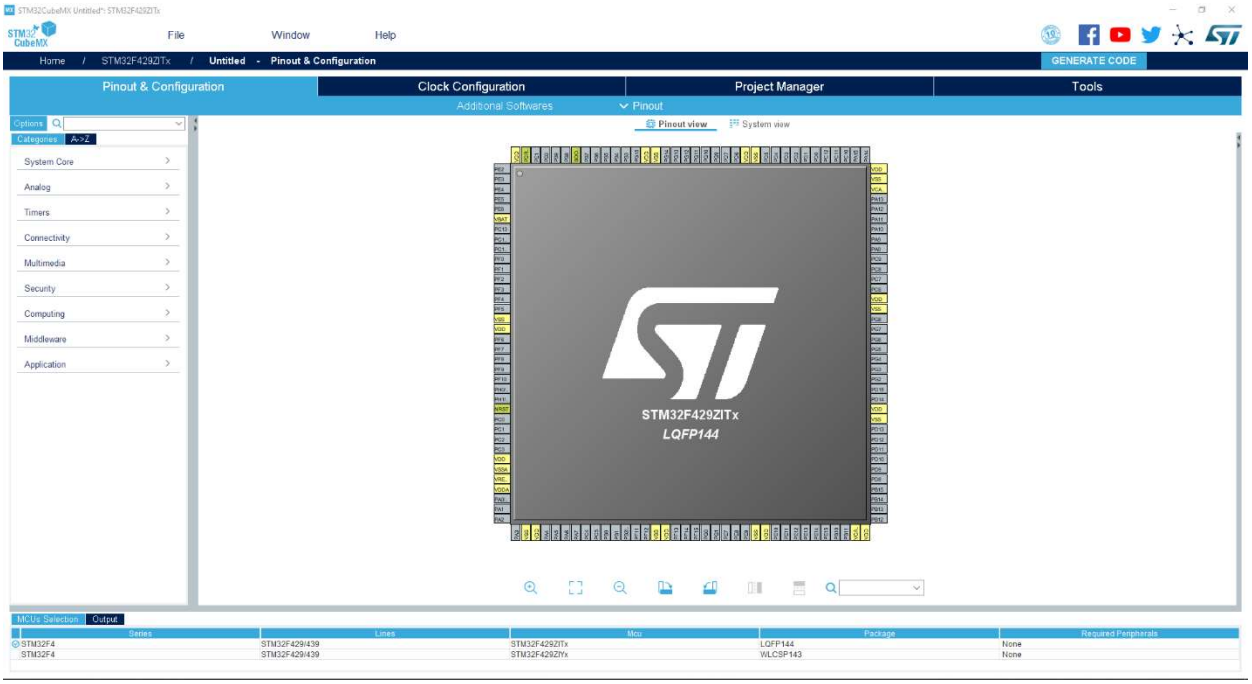
UNIVERSITY OF SAINT THOMAS – SAINT PAUL – MN
Electrical and Computer Engineering

ENGR 432: REAL TIME SYSTEMS

To help you get started, here are the steps to work with STMCubeMX

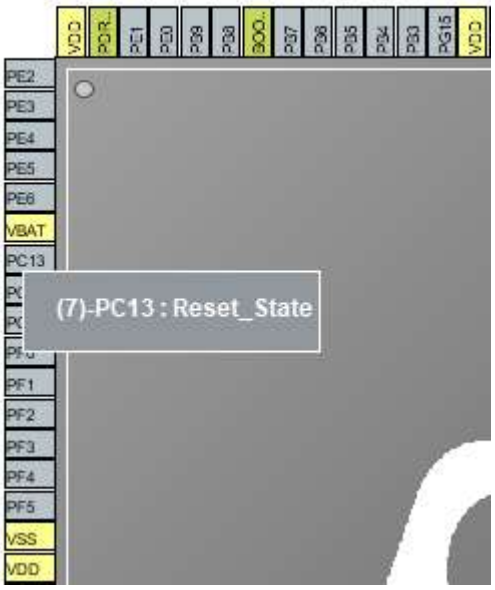
CubeMX (HAL) Guide

1. Select the "File" option at the top of the window.
    a. Select New Project and a new GUI will pop-up
    b. Type into the Search bar on the left "STM32F429ZI"
    c. Click on the board in the "Part No." column in the bottom right list
    d. Click "Start Project" in the top right.
    e. It may ask if you want to initialize the defaults – choose yes
2. Once complete, you should see the following screen:

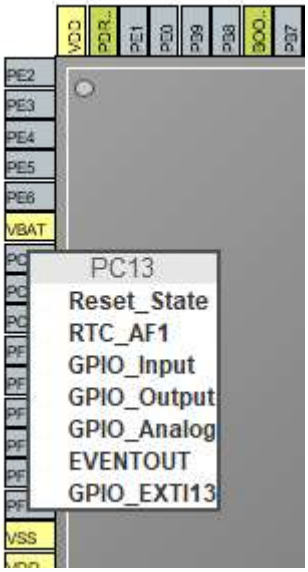


3. In particular we are looking to complete the following tasks:
    a. Set up PC13 (the user button) as an interrupt
    b. Toggle the LEDS (PB0, P7, and PB14)
    c. Set up Timer(s)

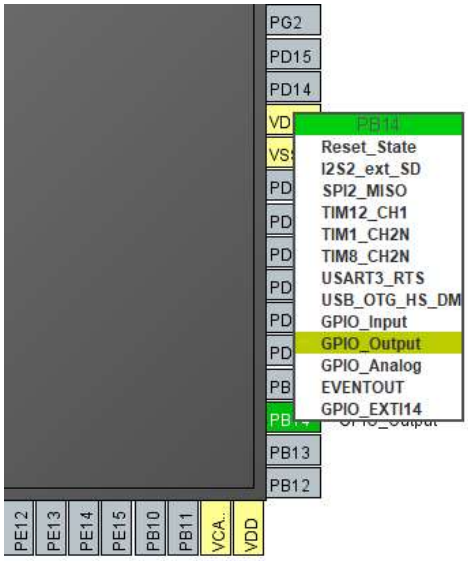
4. Let's start with the Pinouts
    a. If you zoom in on the STM32F429ZITx, you can see tiny little buttons, each

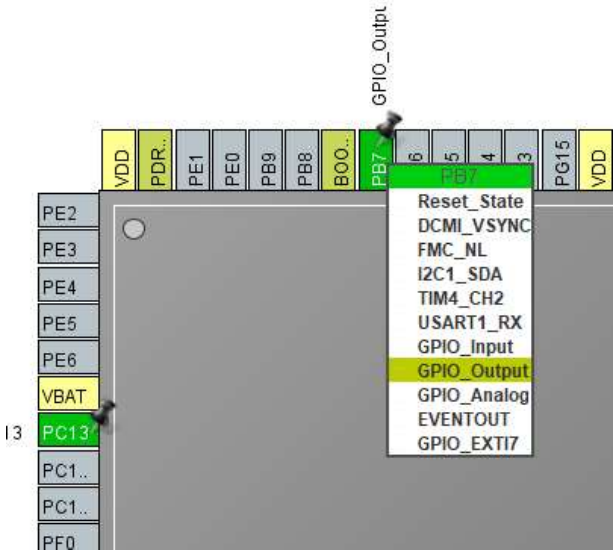


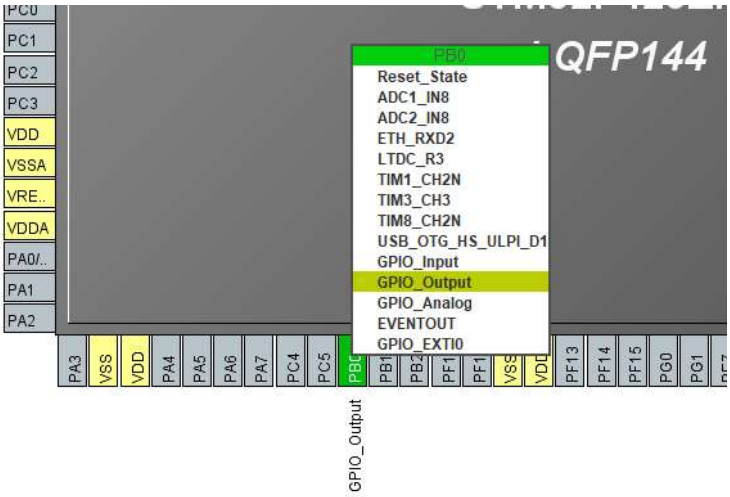
    b. In particular, the 8th one down on the left side is PC13, which by default is in its "reset_state". We want to change that to GPIO_EXTI13 (External Interrupt).



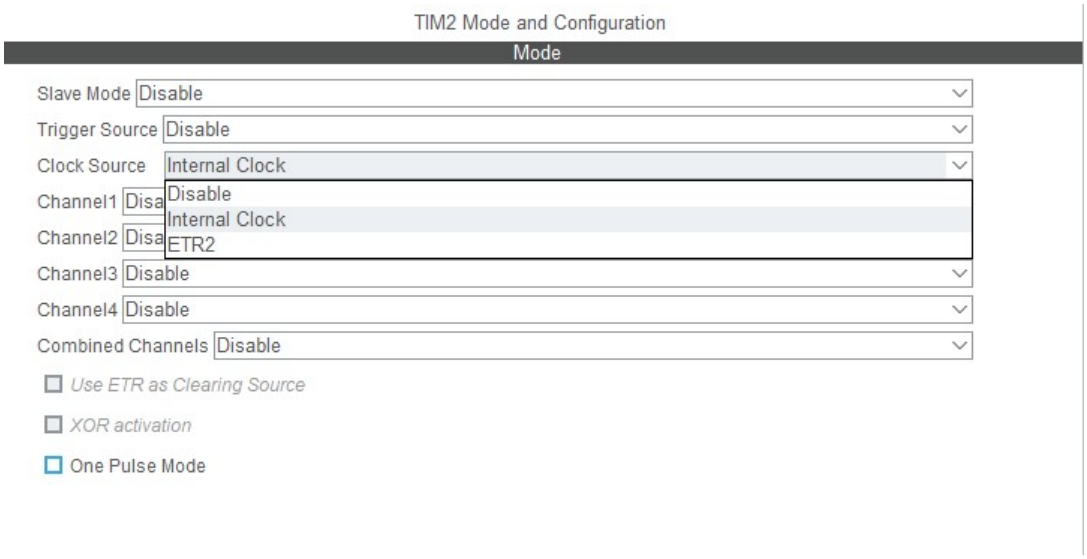
    c. Similarly, we want to locate the PB0, PB7, and PB14 ones, and set them up as well. These ones should be setup as GPIO_Output.


PG2
PD15
PD14
PB14
Reset_State
I2S2_ext_SD
SPI2_MISO
TIM12_CH1
TIM1_CH2N
TIM8_CH2N
USART3_RTS
USB_OTG_HS_DM
GPIO_Input
GPIO_Output
GPIO_Analog
EVENTOUT
GPIO_EXTI14
PB13
PB12
PE12
PE13
PE14
PE15
PB10
PB11
VDD


GPIO_Output
VDD
PDR..
PE1
PE0
PB9
PB8
BOO..
PB7
PG15
VDD
PB7
Reset_State
DCMI_VSYNC
FMC_NL
I2C1_SDA
TIM4_CH2
USART1_RX
GPIO_Input
GPIO_Output
GPIO_Analog
EVENTOUT
GPIO_EXTI7
PE2
PE3
PE4
PE5
PE6
VBAT
PC13
PC1..
PC1..
PF0

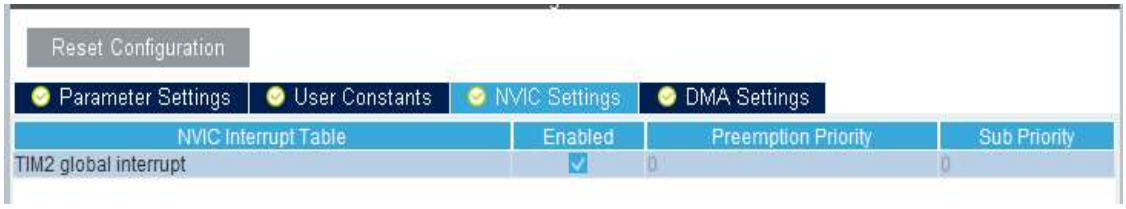

QFP144
PC1
PC2
PC3
VDD
VSSA
VRE..
VDDA
PA0/..
PA1
PA2
PB0
Reset_State
ADC1_IN8
ADC2_IN8
ETH_RXD2
LTDC_R3
TIM1_CH2N
TIM3_CH3
TIM8_CH2N
USB_OTG_HS_ULPI_D1
GPIO_Input
GPIO_Output
GPIO_Analog
EVENTOUT
GPIO_EXTI0
PA3
VSS
VDD
PA4
PA5
PA6
PA7
PC4
PC5
PB1
PF13
PF14
PF15
PG0
PG1
GPIO_Output

5. Now that we've told the board which pins we want to use (and for what), we can configure the clocks/timers.
    a. On the left hand side, there is a Timers tab. Open it up and hover over a clock, it will tell you the uses. In this case, we want to setup one as a "general purpose timer". So, I selected Timer2, and changed the clock source to Internal Clock.

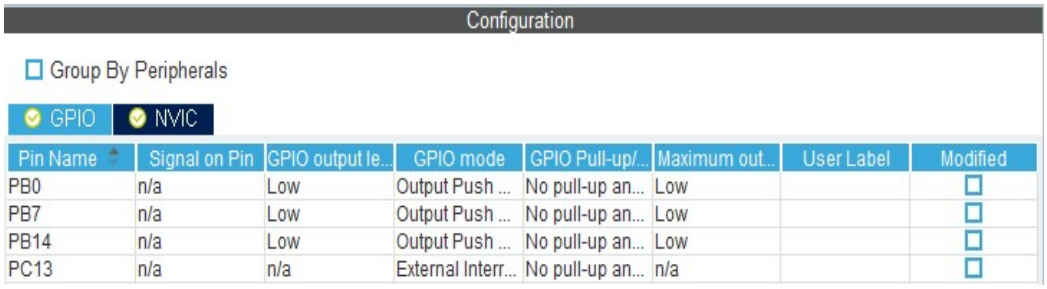


    b. A new section became available, where we are able to select ARR and PSC values for the clock, to ensure it is a 1 second clock. After testing, I determined that an ARR of 21000 and PSC of 999 worked to get a near perfect 1 second timer, but your results may vary.

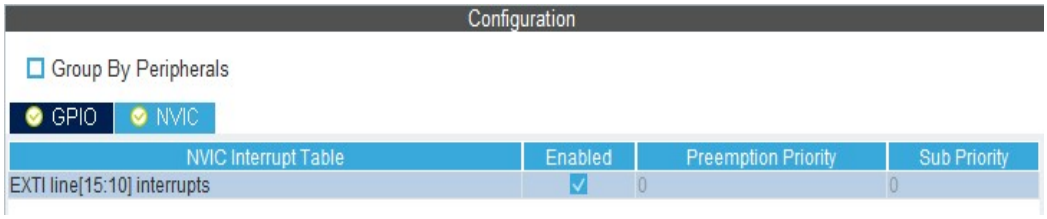
    c. Under the NVIC settings, we want to enable the TIM2 Global Interrupt checkbox.



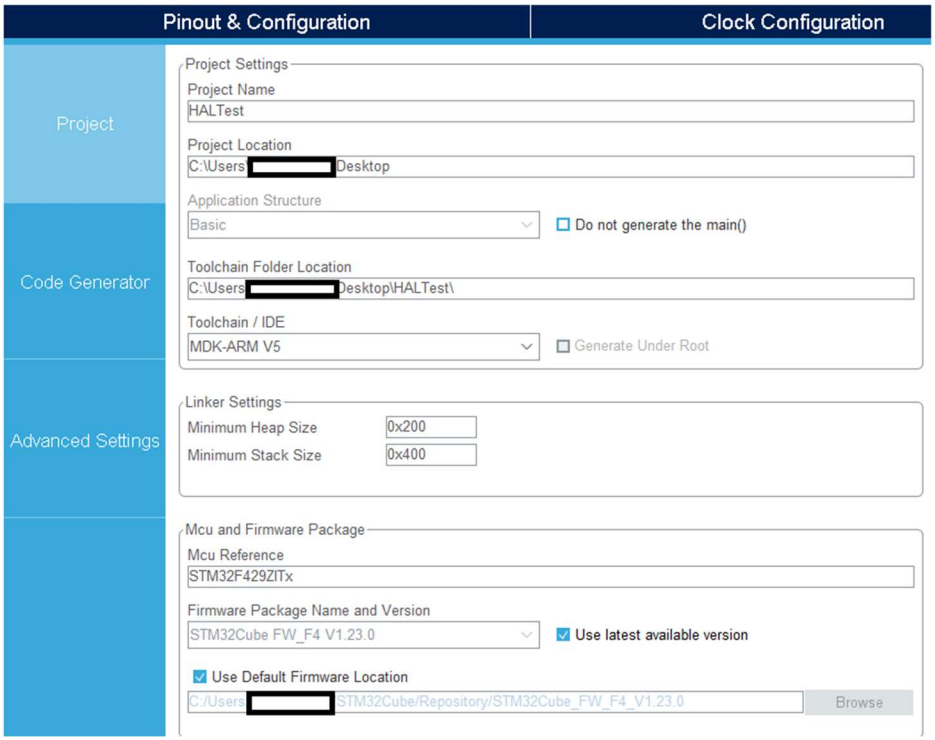
d. Now go back to the left panel, and close up Timers, and select System Core.
e. Click on the GPIO options, and you should see our 4 items listed



f. Select the NVIC tab, and check the option to enable EXTI line[15:10] interrupts

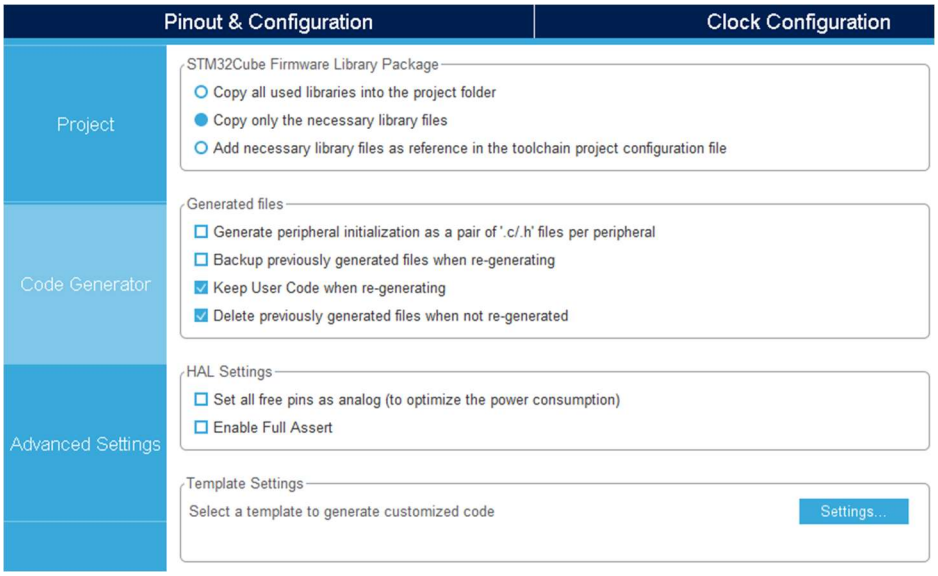


6. Go over to the Project Manager tab
   a. In the Project tab on the left
      i. Give the file a name, and select the location
      ii. Change the Toolchain / IDE to MDK-ARM V5



   b. Click the Code Generator option on the left

i. Select Copy only the necessary library files



7. Hit GENERATE CODE in the top right

8. Choose to Open Project to open directly in Keil

9. From there, the code should be all set up and initialized for use. Obviously, there are some additional functions so accomplish, such as the logic regarding the LEDs, and configuring what actually occurs inside the interrupts.
    a. A quick way to confirm that the initialization worked is to the following code in main.c file to toggle the LEDs every second, using the built in HAL_delay() function:

```
while(1) {
HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_0);
HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_7);
HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_14);
HAL_Delay(1000);
}
```