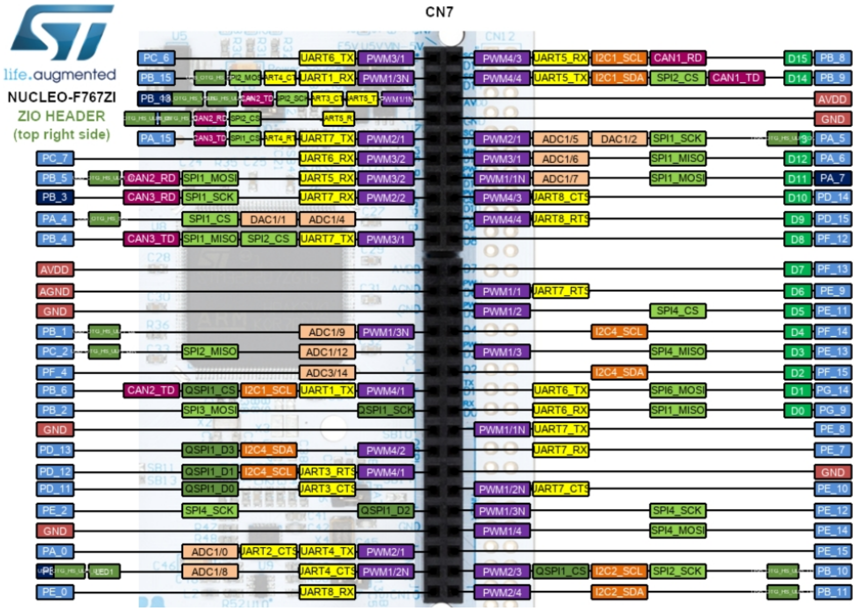
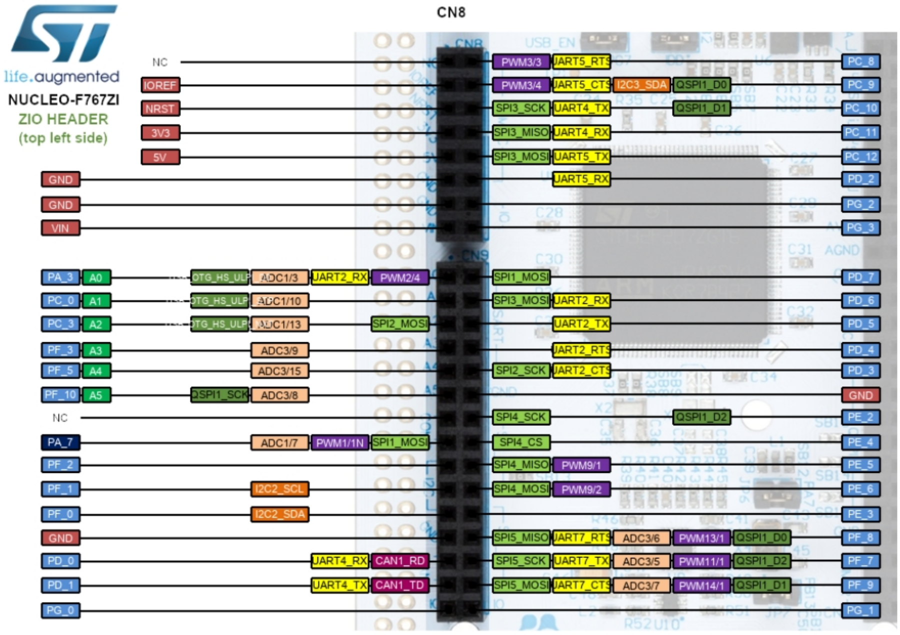
Overview Configuration of Microcontroller [Nucleo-F767ZI]

After create project in STM32CubeIDE let go to file *.ioc then go to Clock Configuration then set High-speed Clock(HCLK) to 216 MHz and go to Project Manager > Code Generator then enable `Generate peripheral initialization as a pair of '.c/.h' file per peripheral`

In the table below is **well-known port** of Nucleo-F767ZI

PORT NUMBER	DESCRIPTION
PB0	LED[GREEN]
PB7	LED[BLUE]
PB14	LED[RED]
PC13	Blue Switch [Buid-in]


life.augmented
NUCLEO-F767ZI
ZIO HEADER
(top right side)
CN7
PC_6 UART6_TX PWM3/1
PB_15 UART1_RX PWM1/3N
PB_13
PA_15 UART7_TX PWM2/1
PC_7 UART6_RX PWM3/2
PB_5 CAN2_RD SPI1_MOSI UART5_RX PWM3/2
PB_3 CAN3_RD SPI1_SCK UART7_RX PWM2/2
PA_4 SPI1_CS DAC1/1 ADC1/4
PB_4 CAN3_TD SPI1_MISO SPI2_CS UART7_TX PWM3/1
AVDD
AGND
GND
PB_1 ADC1/9 PWM1/3N
PC_2 SPI2_MISO ADC1/12
PF_4 ADC3/14
PB_6 CAN2_TD QSPI1_CS I2C1_SCL UART1_TX PWM4/1
PB_2 SPI3_MOSI QSPI1_SCK
GND
PD_13 QSPI1_D3 I2C4_SDA PWM4/2
PD_12 QSPI1_D1 I2C4_SCL UART3_RTS PWM4/1
PD_11 QSPI1_D0 UART3_CTS
PE_2 SPI4_SCK QSPI1_D2
GND
PA_0 ADC1/0 UART2_CTS UART4_TX PWM2/1
PB_0 LED1 ADC1/8 UART4_CTS PWM1/2N
PE_0 UART8_RX
PWM4/3 UART5_RX I2C1_SCL CAN1_RD D15 PB_8
PWM4/4 UART5_TX I2C1_SDA SPI2_CS CAN1_TD D14 PB_9
AVDD
GND
PWM2/1 ADC1/5 DAC1/2 SPI1_SCK D13 PA_5
PWM3/1 ADC1/6 SPI1_MISO D12 PA_6
PWM1/1N ADC1/7 SPI1_MOSI D11 PA_7
PWM4/3 UART8_CTS D10 PD_14
PWM4/4 UART8_RTS D9 PD_15
D8 PF_12
D7 PF_13
PWM1/1 UART7_RTS D6 PE_9
PWM1/2 SPI4_CS D5 PE_11
I2C4_SCL D4 PF_14
PWM1/3 SPI4_MISO D3 PE_13
I2C4_SDA D2 PF_15
UART6_TX SPI6_MOSI D1 PG_14
UART6_RX SPI1_MISO D0 PG_9
PWM1/1N UART7_TX PE_8
UART7_RX PE_7
GND
PWM1/2N UART7_CTS PE_10
PWM1/3N SPI4_SCK PE_12
PWM1/4 SPI4_MOSI PE_14
PE_15
PWM2/3 QSPI1_CS I2C2_SCL SPI2_SCK PB_10
PWM2/4 I2C2_SDA PB_11



Part 1: GPIO

In the *.ioc file, you can configure pinMode to GPIO_OUTPUT or GPIO_INPUT. After saving the *.ioc file, GPIO configuration is generated in MX_GPIO_Init() function inside main.c

In the table below is **well-known GPIO function**

FUNCTION	DESCRIPTION
`HAL_GPIO_ReadPin(GPIOx, GPIO_PIN_x);`	Read Digital value frV, asom PIN
`HAL_GPIO_WritePin(GPIOx, GPIO_PIN_x, PinState);`	Set Digital value of PIN
`GPIO_PIN_SET`	Is PinState, Return logic 1
`GPIO_PIN_RESET`	Is PinState, Return logic 0

In STM32CubeIDE Board Nucleo-F767ZI **able to read output pin**

Part 2 : UART/USART

UART and USART in Nucleo-F767ZI is port communication, method of data tranfer/receive are **Asynchronous** for UART and **Synchronous** for USART.

In the table below is **well-known UART/USART port**

PORT NUMBER	FUNCTION
PF6	UART7_RX
PF7	UART7_TX
PA0	UART4_TX
PA1	UART4_RX
PD8	USART3_TX
PD9	USART3_RX

Step to configure UART/USART

I. In the *.ico file, choose port UART/USART RX and TX
II. go to configuration of UART/USART that you chosen (you can fill UART/USART in search bar at top-left)
III. select Mode **Asynchronous** (I don't know why)
IV. In Parameter Settings
V. set **Baud Rate** = 115200 Bits/s, **Word Length** = 8 Bit, **Parity** = None, **Stop Bits** = 1
VI. In Advance Parameters Set **Data Direction** = Receive and Transmit, **Over Sampling** = 16 Samples, **Single Sample** = Disable

In **GPIO Settings** select UART/USART Pin then set **GPIO Mode** = Alternate Function Push Pull, **GPIO Pull-up/Pull-down** = No pull-up and no pull-down, **Maximum output speed** = Very High

After saving the *.ioc file, Configuration about UART/USART is generated in usart.c , gpio.c and main.c

In the table **below is well-known function of UART/USART**

FUNCTION	Description
`HAL_UART_Transmit(&huart3, (uint8_t *)buf, strlen(buf), 100);`	This function will transmit data to UART/USART communication and timeout is 100ms

`if (HAL_UART_Receive(&huart3, (uint8_t*)&input , 1 , 100) == HAL_OK )`
`{`
`  …`
`}`
	This codition will wait until UART/USART retrun HAL_OK that mean have any input from communication

`while(__HAL_UART_GET_FLAG(&huart3,UART_FLAG_TC)==RESET ){}`	**Flag TC** (transmission control), This function will wait until before transmission complete
`while(__HAL_UART_GET_FLAG(&huart3,UART_FLAG_RXNE)== RESET ){}`	**Flag RXNE** (register not empty), This funtion will check that receive data before complete?

Part 3: NVIC and EXTI

NVIC (Nested Vectored Interrupt Controller) is module in microcontroller have responsibility to control, configure and response interrupt

EXTI (External Interrupt) is interrupt from GPIO signal

NVIC_PriorityGroup	PreemptionPriority		SubPriority	
	Bits	Possible value	Bits	Possible value
NVIC_PriorityGroup_0	0	0	4	0-15
NVIC_PriorityGroup_1	1	0-1	3	0-7
NVIC_PriorityGroup_2	2	0-3	2	0-3
NVIC_PriorityGroup_3	3	0-7	1	0-1
NVIC_PriorityGroup_4	4	0-15	0	0

In the table above, mention about **PreemptionPriority** and **SubPriority**, Each interrupt need a configure priority, When there is more than one interrupt, NVIC will check **PreemptionPriority**. If PreemptionPriority value is lower (0 is the highest priority), that interrupt will have higher permission to execute. If PreemtionPriority value is equal or lower then go to wait state, In wait state if there is more that one interrupt NVIC will check **SubPriority** If SubPriority value is lower (0 is the highest priority)

Step to configure NVIC/EXTI

I. In the *.ioc file, choose PIN GPIO_EXTI
II. go to NVIC Configuration (fill “NVIC” in search bar at top-left)
III. choose priority group (According to table above)
IV. enable EXTI.* NVIC interrupt
V. set Preemtion and Sub Priority

In GPIO Setting select EXTI Pin and set GPIO mode to External Interrupt Mode with **Rising edge trigger detection** or **Falling edge trigger detection** and set GPIO Pull-up/Pull-down to No pull-up and no pull-down

After saving *.ioc file configuration about NVIC will generated by HAL_MspInit() inside file **stm32f7xx_hal_msp.c**

ISR of each EXTI

EXTI Number	Interrupt Name	ISR Name	Remark
EXTI0	EXTI0_IRQn	EXTI0_IRQHandler	-
EXTI1	EXTI1_IRQn	EXTI1_IRQHandler	-
EXTI2	EXTI2_IRQn	EXTI2_IRQHandler	-
EXTI3	EXTI3_IRQn	EXTI3_IRQHandler	-
EXTI4	EXTI4_IRQn	EXTI4_IRQHandler	-
EXTI5 – EXTI9	EXTI9_5_IRQn	EXTI9_5_IRQHandler	EXIT 5 -9 use same ISR
EXTI10 – EXTI15	EXTI15_10_IRQn	EXTI15_10_IRQHandler	EXIT 10 -15 use same ISR

How to implement **ISR (Interrupt Service Routine)** of EXTI, In file **stm32f7xx_it.c** you can implement code ISR in function’s name is ISR Name (According to table above)

Callback Function, how to implement callback function, you have to create function below in file **main.c**

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
        if (GPIO_Pin == GPIO_PIN_13)
        {
                // implement callback function
        }
}
```

Part 4: ADC (Analog to Digital Converter)

In Nucleo-F767ZI contain 3 ADC Module ADC1-3 connect to APB2 (Advanced Peripheral Bus) each module contain 19 channel (external signal channel 0-15 , channel 17 is Vref)

Step to configure ADC

I. In *.ioc file, Select ADC port

II. In Parameter Settings, set **Continuous Conversion Mode** to Enabled and set **End of Conversion Selection** to EOC flag at the end of all conversions

Implement example code get ADC value in main.c function main.

```
volatile uint32_t adc_val = 0;

HAL_ADC_Start(&hadc1);

while (1)
{
  /* USER CODE END WHILE */
  /* USER CODE BEGIN 3 */
        while( HAL_ADC_PollForConversion(&hadc1, 100) != HAL_OK){}
        uint32_t adc_value = HAL_ADC_GetValue(&hadc1);
        displayHEX(adc_value);
        HAL_Delay(400);
}
```

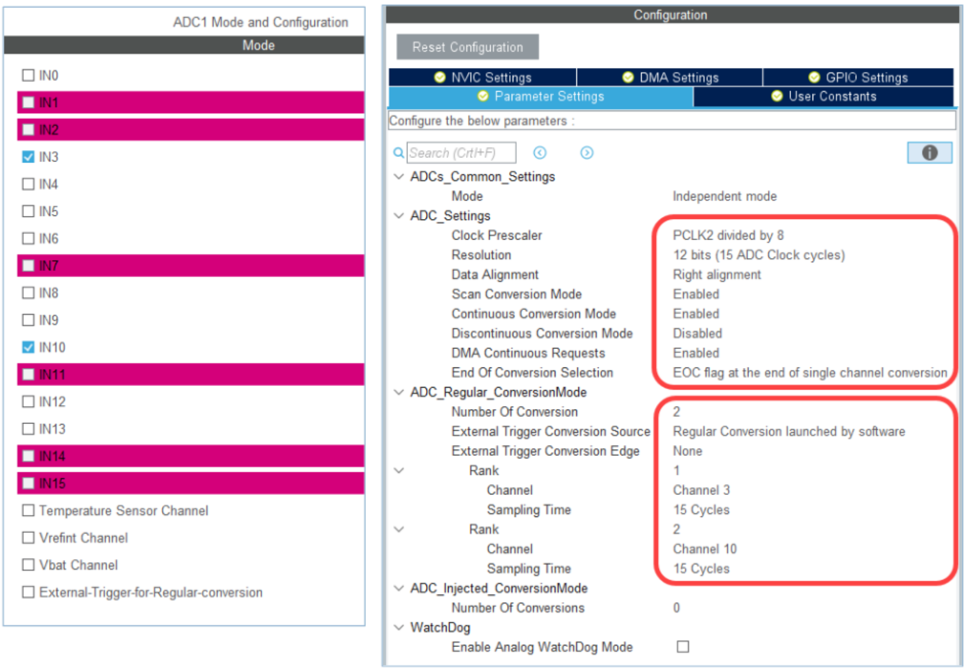
DMA (Direct Memory Access Controller) is a module that transfers data without using the CPU, so CPU able to execute other instructions. I use DMA when there are multiple channels to read ADC values.

Step to configure DMA

I. In the *.ioc file go to ADC Configuration

II. Select channel that used

III. config DMA (According to below picture)



□ IN0
□ IN1
□ IN2
☑ IN3
□ IN4
□ IN5
□ IN6
□ IN7
□ IN8
□ IN9
☑ IN10
□ IN11
□ IN12
□ IN13
□ IN14
□ IN15
□ Temperature Sensor Channel
□ Vrefint Channel
□ Vbat Channel
□ External-Trigger-for-Regular-conversion

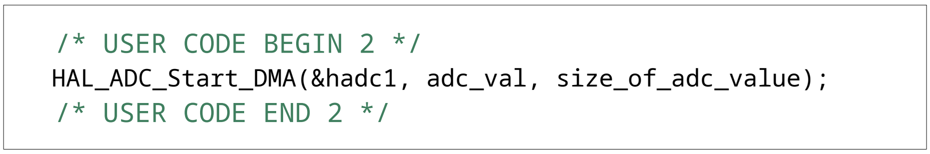
ADCs_Common_Settings	
Mode	Independent mode
ADC_Settings	
Clock Prescaler	PCLK2 divided by 8
Resolution	12 bits (15 ADC Clock cycles)
Data Alignment	Right alignment
Scan Conversion Mode	Enabled
Continuous Conversion Mode	Enabled
Discontinuous Conversion Mode	Disabled
DMA Continuous Requests	Enabled
End Of Conversion Selection	EOC flag at the end of single channel conversion
ADC_Regular_ConversionMode	
Number Of Conversion	2
External Trigger Conversion Source	Regular Conversion launched by software
External Trigger Conversion Edge	None
Rank	1
Channel	Channel 3
Sampling Time	15 Cycles
Rank	2
Channel	Channel 10
Sampling Time	15 Cycles
ADC_Injected_ConversionMode	
Number Of Conversions	0
WatchDog	
Enable Analog WatchDog Mode	□

DMA Configuration (ref. lecture 2568 Lec07 ADC.pdf)

IV. Rank in Configuration is order of reading ADC
V. Go to DMA Configuration, then select the ADC module that is used
VI. Click **Add**, Set **Mode** to **Circular** and set **Data Width** to **Word**

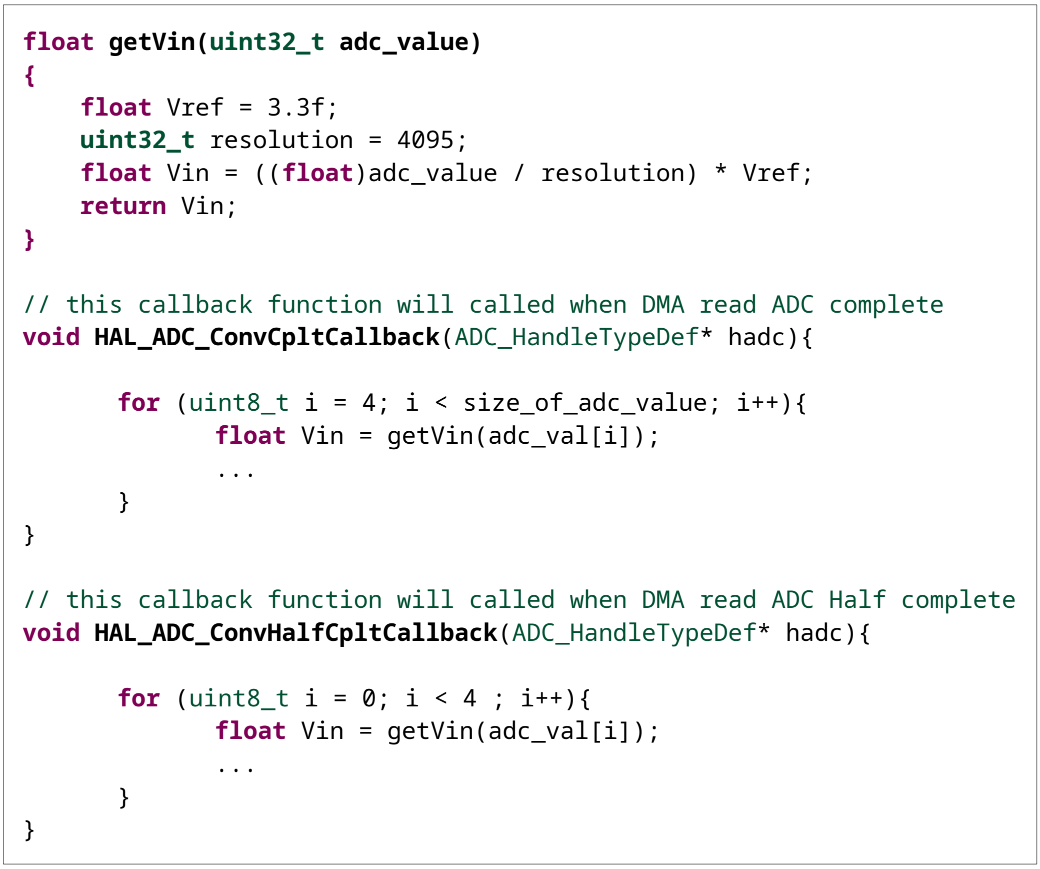
then go to **NVIC** configuration choose Priority group and enable DMA , set Preemtion&Sub Priority. ISR DMA is `DMA2_Stream0_IRQHandler`

Implement example code read multichannel ADC



```
 /* USER CODE BEGIN 2 */
HAL_ADC_Start_DMA(&hadc1, adc_val, size_of_adc_value);
 /* USER CODE END 2 */
```

Before infinite loop, Start DMA (adc_val is array), The value that is read will be stored in adc_val



```
float getVin(uint32_t adc_value)
{
    float Vref = 3.3f;
    uint32_t resolution = 4095;
    float Vin = ((float)adc_value / resolution) * Vref;
    return Vin;
}

// this callback function will called when DMA read ADC complete
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc){

       for (uint8_t i = 4; i < size_of_adc_value; i++){
              float Vin = getVin(adc_val[i]);
              ...
       }
}

// this callback function will called when DMA read ADC Half complete
void HAL_ADC_ConvHalfCpltCallback(ADC_HandleTypeDef* hadc){

       for (uint8_t i = 0; i < 4 ; i++){
              float Vin = getVin(adc_val[i]);
              ...
       }
}
```

Implement callback function of DMA (Complete, Half Complete)

Part 5: Timer

In Nucleo-F767ZI contain 18 Timer module (**TIM1 and TIM8** is advanced-control timers and **TIM2-5** is general-purpose timers). In the table below is connection between Timer and APB

Bus	MAX Bus Frequency (MHz)	MAX Timer Frequency (MHz)	Module
APB1	54	108	TIM2 TIM3 TIM4 TIM5 TIM12 TIM13 TIM14
APB2	108	216	TIM1 TIM8 TIM9 TIM10 TIM11

Step to configure Timer

I. Choose Port Timer in the *.ioc file
II. Go to TIM Configuration
III. In **Clock Source** select **Internal Clock**

In **Parameter Settings** you can configure module timer count the time that you want by set Prescaler, Counter Mode and Counter Period by according to formula below

```
Time Interval = (Clock Division x Prescaler x Period) / APBx Bus Speed
```

For example if want TIM1 count 1 ms. You can set Prescaler to 216-1, Counter Period to 1000-1 and No division (APB2 for TIM1 Bus Speed is 216 MHz)

```
Time Interval = (Clock Division x Prescaler x Period) / APBx Bus Speed
              = (    1         x   (216-1) x 1000-1) / 216 MHz
              = 1 x 10^{-3}
              = 1 ms
```

After setting timer you have to go to **NVIC configuration** then set Priority Group, enable TIM module and set Preemion&Sub Priority

Interrupt Service Routine of Timer when you want to use timer you have call start timer function and implement code in ISR of timer in function **TIM1_UP_TIM10_IRQHandler** inside **stm32f7xx_it.c**

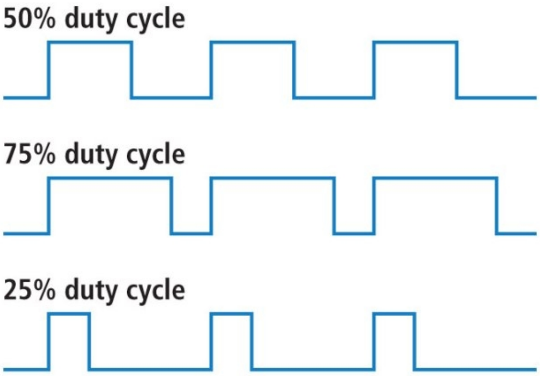
```
HAL_TIM_Base_Start_IT(&htim1);
while (1)
{
    ...
}
HAL_TIM_Base_Stop_IT(&htim1);
```

```c
void TIM1_UP_TIM10_IRQHandler(void)
{
  /* USER CODE BEGIN TIM1_UP_TIM10_IRQn 0 */

  /* USER CODE END TIM1_UP_TIM10_IRQn 0 */
  HAL_TIM_IRQHandler(&htim1);
  /* USER CODE BEGIN TIM1_UP_TIM10_IRQn 1 */
      ...
  /* USER CODE END TIM1_UP_TIM10_IRQn 1 */
}

/**
  * @brief This function handles TIM2 global interrupt.
  */
void TIM2_IRQHandler(void)
{
  /* USER CODE BEGIN TIM2_IRQn 0 */

  /* USER CODE END TIM2_IRQn 0 */
  HAL_TIM_IRQHandler(&htim2);
  /* USER CODE BEGIN TIM2_IRQn 1 */
      ...
  /* USER CODE END TIM2_IRQn 1 */
}
```

ISR of each timer

Part 6: PWM (Pulse-Width Modulation)

pwm signal is generated by **Timer module,** The PWM value is controlled by the **Duty cycle.** The **voltage received** depends on the pulse width of waveform.



What is Duty Cycle ?

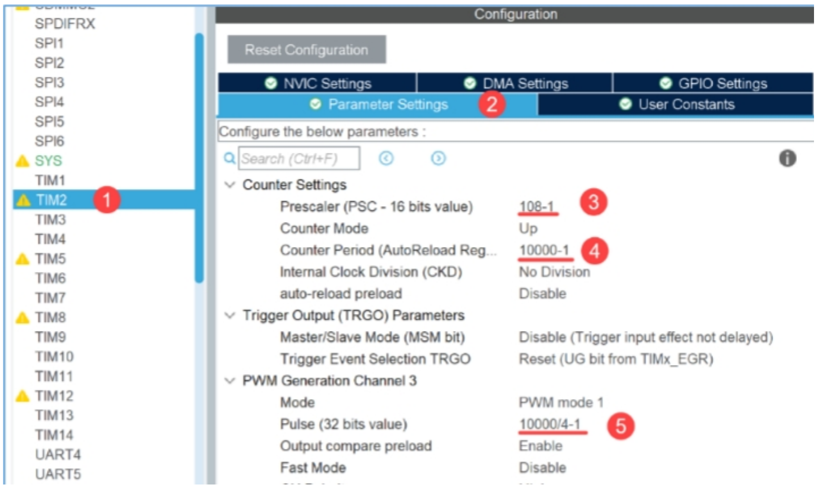
Step to configure PWM

I. In the *.ioc file, choose timer module
II. go to timer configuration
III. In Clock Source, select **Internal Clock**
IV. In Channel that you choose to generate PWM, select **PWM Generation CHx**
V. In **GPIO settings**, select pin that generate PWM then set Maximum output speed to **Very High**

Step to configure Duty Cycle

I. go to timer configuration
II. In Parameter Settings
III. In Counter Settings, set Prescaler according to APBx and set Counter Period (To generate a PWM signal with a period *x* seconds)
IV. In PWM Generation channel *x* , set Pulse according to configuration of Counter Settings

For example, if I want to generate a PWM signal with 10 ms period and 25% duty cycle. In the picture below In PWM Generation Channel 3, you will se **Mode** is PWM mode 1. PWM mode 1 generates PWM signal with **active high**, while PWM mode 2 generates PWM signal with **active low**.



Counter Settings
Prescaler (PSC - 16 bits value) 108-1
Counter Mode Up
Counter Period (AutoReload Reg... 10000-1
Internal Clock Division (CKD) No Division
auto-reload preload Disable
Trigger Output (TRGO) Parameters
Master/Slave Mode (MSM bit) Disable (Trigger input effect not delayed)
Trigger Event Selection TRGO Reset (UG bit from TIMx_EGR)
PWM Generation Channel 3
Mode PWM mode 1
Pulse (32 bits value) 10000/4-1
Output compare preload Enable
Fast Mode Disable

```
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_3);
HAL_Delay(100);
pwm = (GPIOB->IDR & GPIO_PIN_10) >> 10 ;
HAL_TIM_PWM_Stop(&htim2, TIM_CHANNEL_3);
```

Example code to read pwm value

Part 7: LCD and Touch Sensor

So, let’s start to set enviroment to use LCD and Touch Sensor, First you have to install lib ***STM32-ILI9341***

Step to set enviroment

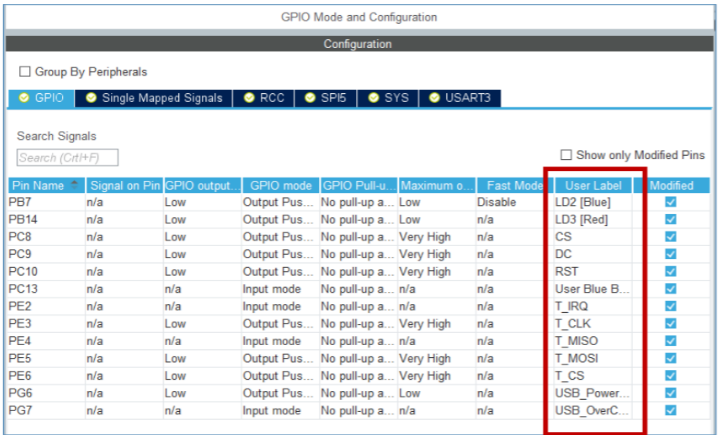
I. Install lib from [*https://github.com/martnak/STM32-ILI9341/tree/master/Src/ILI9341*](https://github.com/martnak/STM32-ILI9341/tree/master/Src/ILI9341)
II. download only this directory
III. you will see file *.c and *.h
IV. move file *.c to inside of directory Src/ on your project
V. move file *.h to inside of directory Inc/ on your project
VI. try to build and run project (you will see that error)

Now you can’t run this project. When you run will see error, you have to change variable name to correct name and In file **ILI9341_Touchscreen.h** don’t forget to `#include "main.h"`

If you set enviroment complete, let’s connect hardware to Nucleo-F767ZI according to the table bellow.

Nucleo-F767ZI	Function	LCD	Note
PE2	GPIO_Input	T_IRQ	
PE4	GPIO_Input	T_DO	MISO
PE5	GPIO_Output	T_DIN	MOSI
PE6	GPIO_Output	T_CS	
PE3	GPIO_Output	T_CLK	
PF8	SPI5_MISO	SDO	MISO
3.3V		LED	
PF7	SPI_SCK	SCK	
PF9	SPI_MOSI	SDI	MOSI
PC9	GPIO_Output	DC	
PC10	GPIO_Output	RESET	
PC8		GND	
3.3V		Vcc	

Configuration each moudle

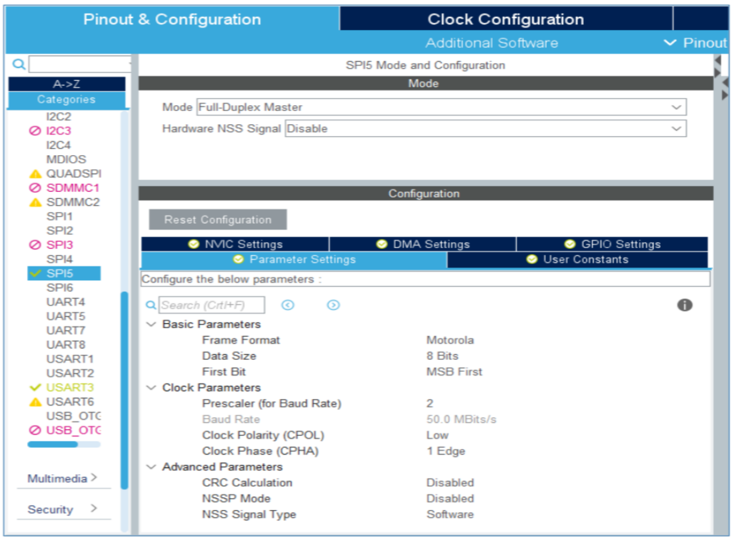


□ Group By Peripherals

□ Show only Modified Pins

Pin Name	Signal on Pin	GPIO output...	GPIO mode	GPIO Pull-u...	Maximum o...	Fast Mode	User Label	Modified
PB7	n/a	Low	Output Pus...	No pull-up a...	Low	Disable	LD2 [Blue]	☑
PB14	n/a	Low	Output Pus...	No pull-up a...	Low	n/a	LD3 [Red]	☑
PC8	n/a	Low	Output Pus...	No pull-up a...	Very High	n/a	CS	☑
PC9	n/a	Low	Output Pus...	No pull-up a...	Very High	n/a	DC	☑
PC10	n/a	Low	Output Pus...	No pull-up a...	Very High	n/a	RST	☑
PC13	n/a	n/a	Input mode	No pull-up a...	n/a	n/a	User Blue B...	☑
PE2	n/a	n/a	Input mode	No pull-up a...	n/a	n/a	T_IRQ	☑
PE3	n/a	Low	Output Pus...	No pull-up a...	Very High	n/a	T_CLK	☑
PE4	n/a	n/a	Input mode	No pull-up a...	n/a	n/a	T_MISO	☑
PE5	n/a	Low	Output Pus...	No pull-up a...	Very High	n/a	T_MOSI	☑
PE6	n/a	Low	Output Pus...	No pull-up a...	Very High	n/a	T_CS	☑
PG6	n/a	Low	Output Pus...	No pull-up a...	Low	n/a	USB_Power...	☑
PG7	n/a	n/a	Input mode	No pull-up a...	n/a	n/a	USB_OverC...	☑

GPIO Settings (don't forget to change output speed to Very High)



Mode: Full-Duplex Master

Hardware NSS Signal: Disable

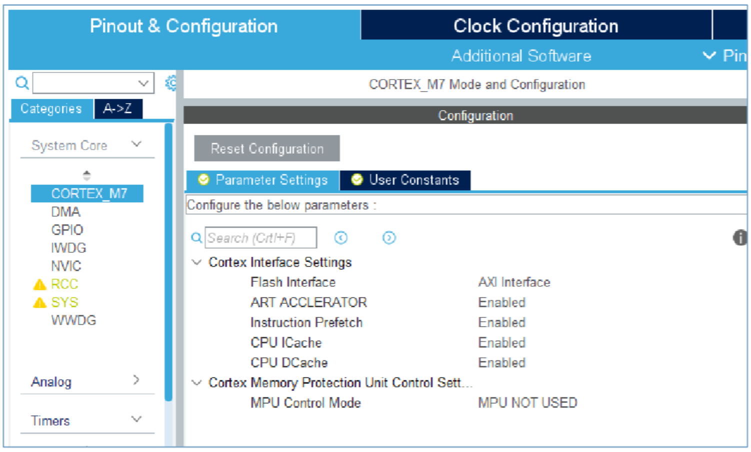
Configure the below parameters :

Basic Parameters
- Frame Format: Motorola
- Data Size: 8 Bits
- First Bit: MSB First

Clock Parameters
- Prescaler (for Baud Rate): 2
- Baud Rate: 50.0 MBits/s
- Clock Polarity (CPOL): Low
- Clock Phase (CPHA): 1 Edge

Advanced Parameters
- CRC Calculation: Disabled
- NSSP Mode: Disabled
- NSS Signal Type: Software

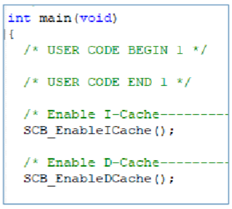
SPI5 Configuration



Configure the below parameters :

Cortex Interface Settings
- Flash Interface: AXI Interface
- ART ACCLERATOR: Enabled
- Instruction Prefetch: Enabled
- CPU ICache: Enabled
- CPU DCache: Enabled

Cortex Memory Protection Unit Control Sett...
- MPU Control Mode: MPU NOT USED

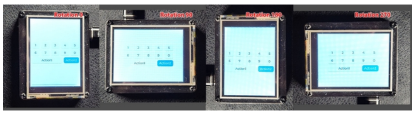
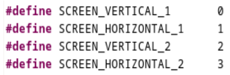


```
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* Enable I-Cache---------
  SCB_EnableICache();

  /* Enable D-Cache---------
  SCB_EnableDCache();
```

Cotex M7 Configuration

Table Function of ILI9341

FUNCTION	**RESUlT**	**NOTE**
`ILI9341_Draw_Filled_Circle(X, Y, Radius, Colour);`	●	
`ILI9341_Draw_Hollow_Circle(X, Y, Radius, Colour);`	○	
`ILI9341_Draw_Text(Text, X, Y, Colour, Size, Background_Colour);`	Display Typeface	

`ILI9341_Set_Rotation(Rotation);`		 `#define SCREEN_VERTICAL_1      0`
		`#define SCREEN_HORIZONTAL_1    1`
		`#define SCREEN_VERTICAL_2      2`
		`#define SCREEN_HORIZONTAL_2    3`

`ILI9341_Draw_Filled_Rectangle_Coord(X0, Y0, X1, Y1, Colour);`	■	
`ILI9341_Fill_Screen(Blue);`		