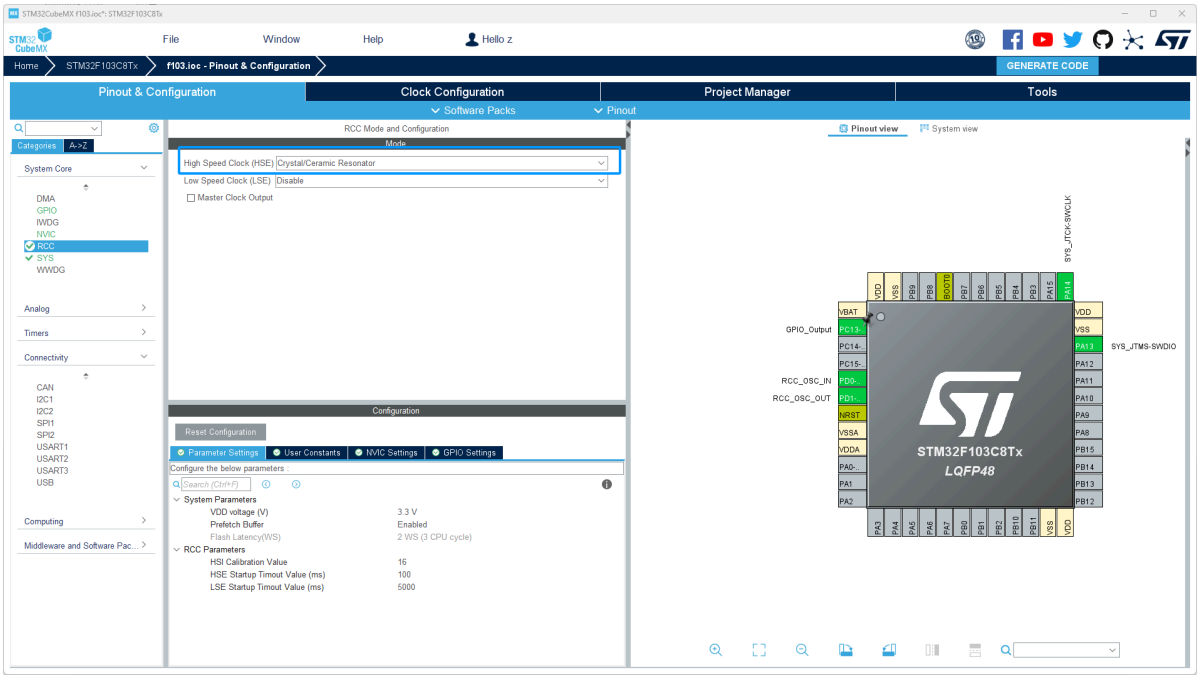
GPIO&EXTI

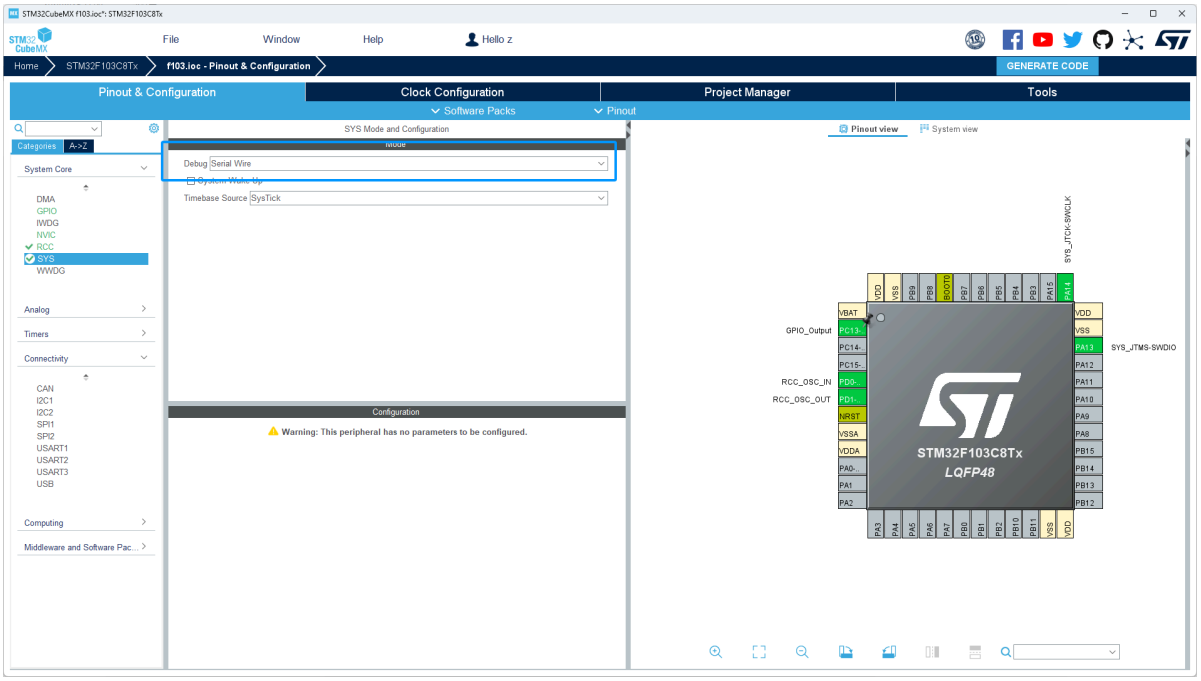
GPIO配置

CubeMX

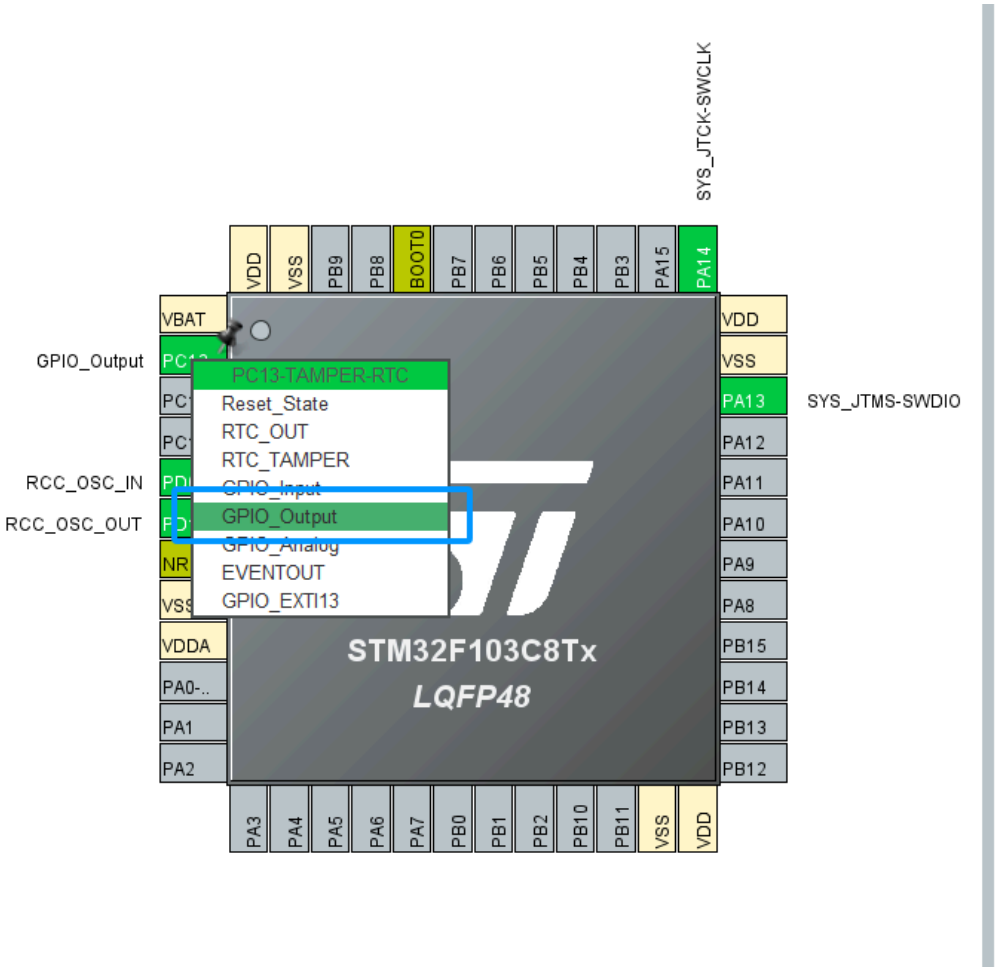
选好板子（`stm32f103c8t6`）之后，先把时钟源配置为如下模式



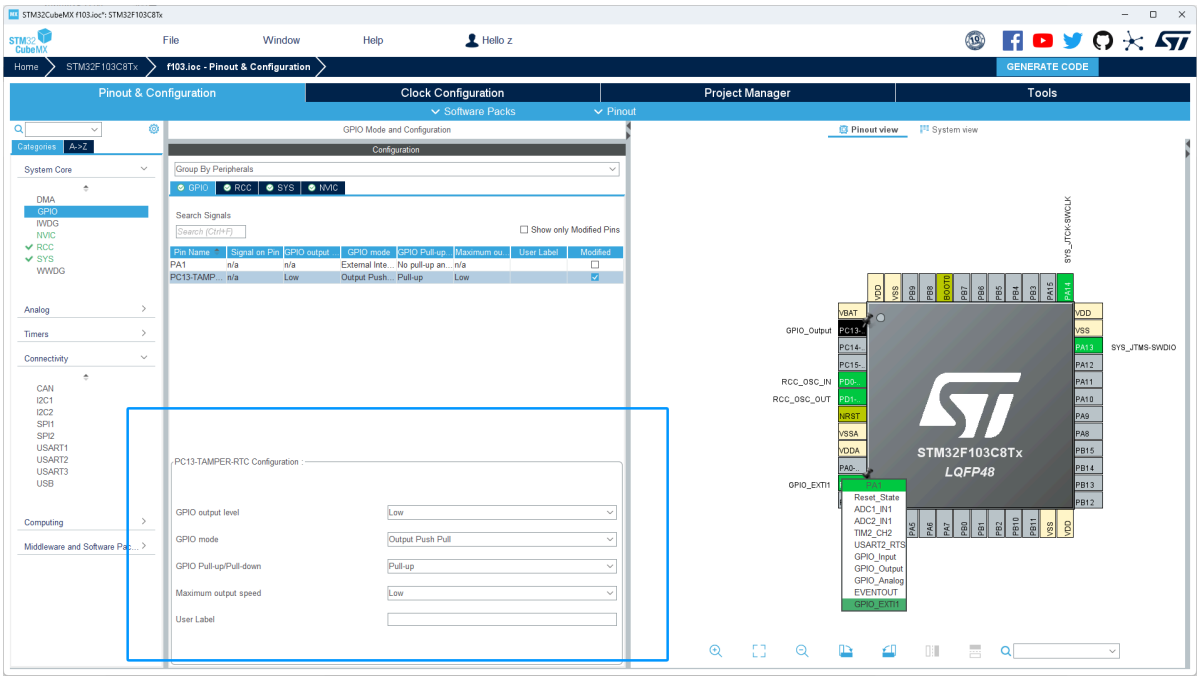
然后是SYS中的Debug：



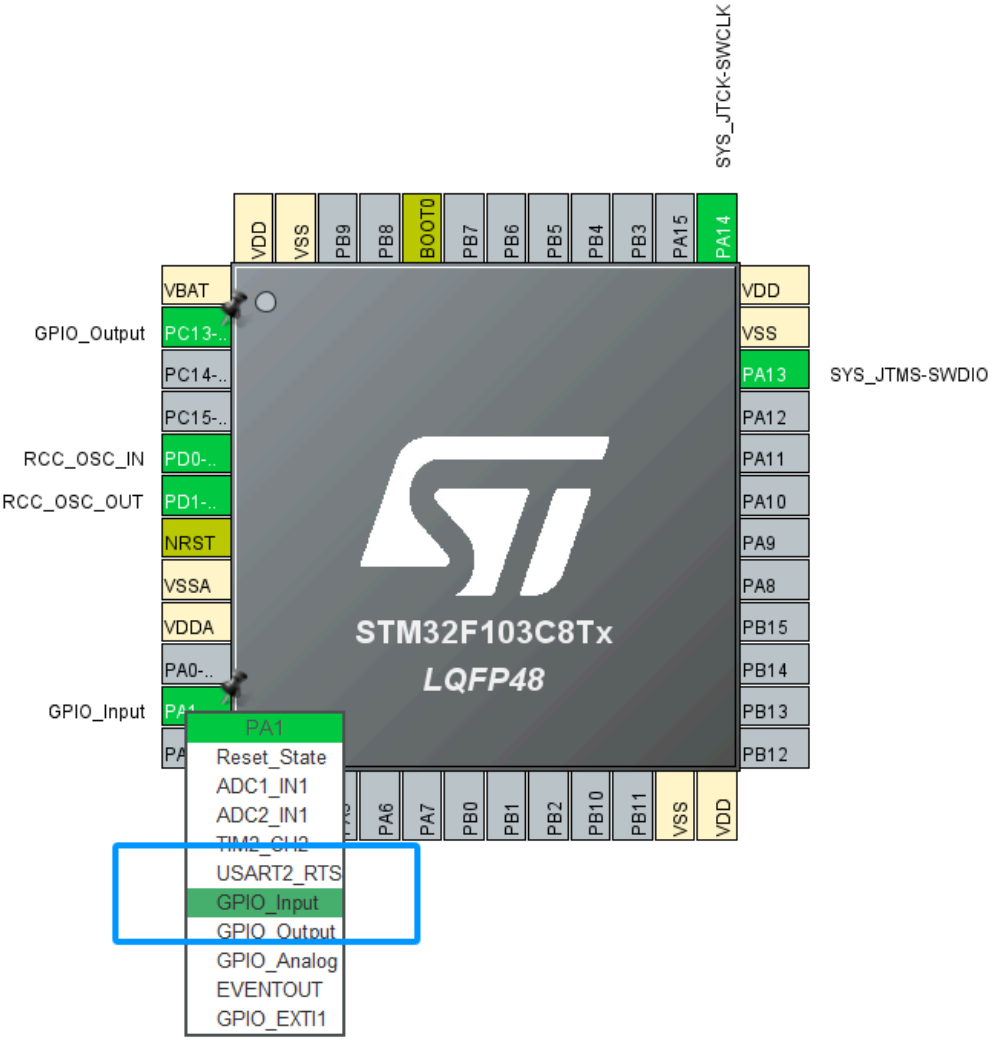
输出模式选这个：（输出选PC13是因为板载LED连在PC13上）



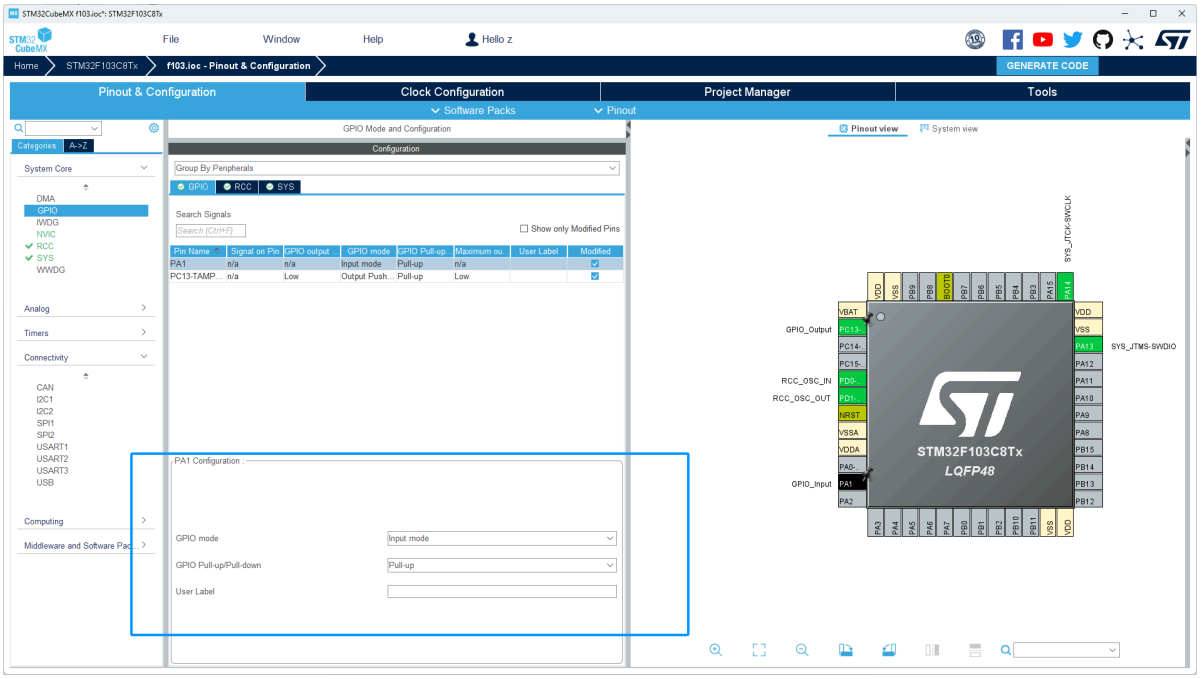
输出选项卡配置如下 注意一定要选择推挽模式



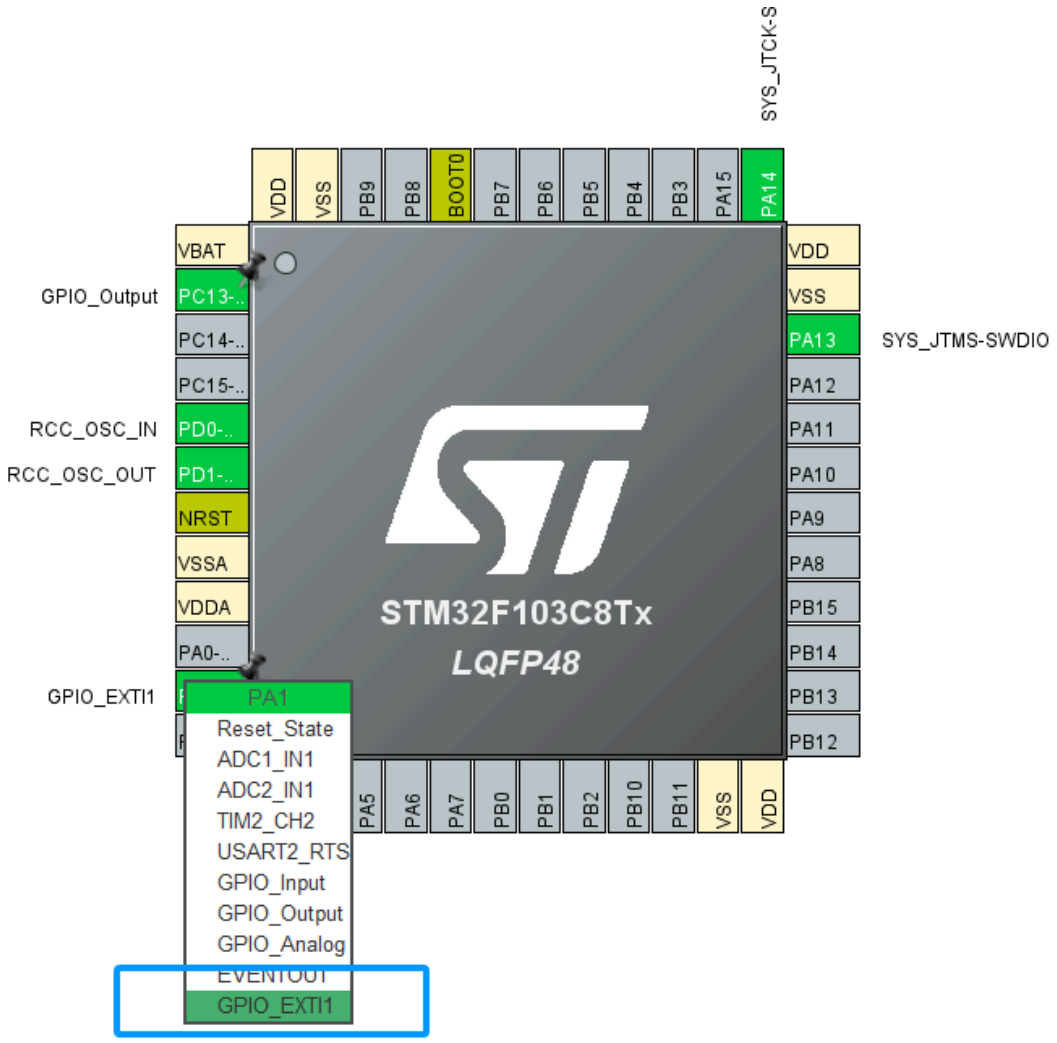
输入模式选这个：（不一定要选这个引脚，这里只是一个示例）



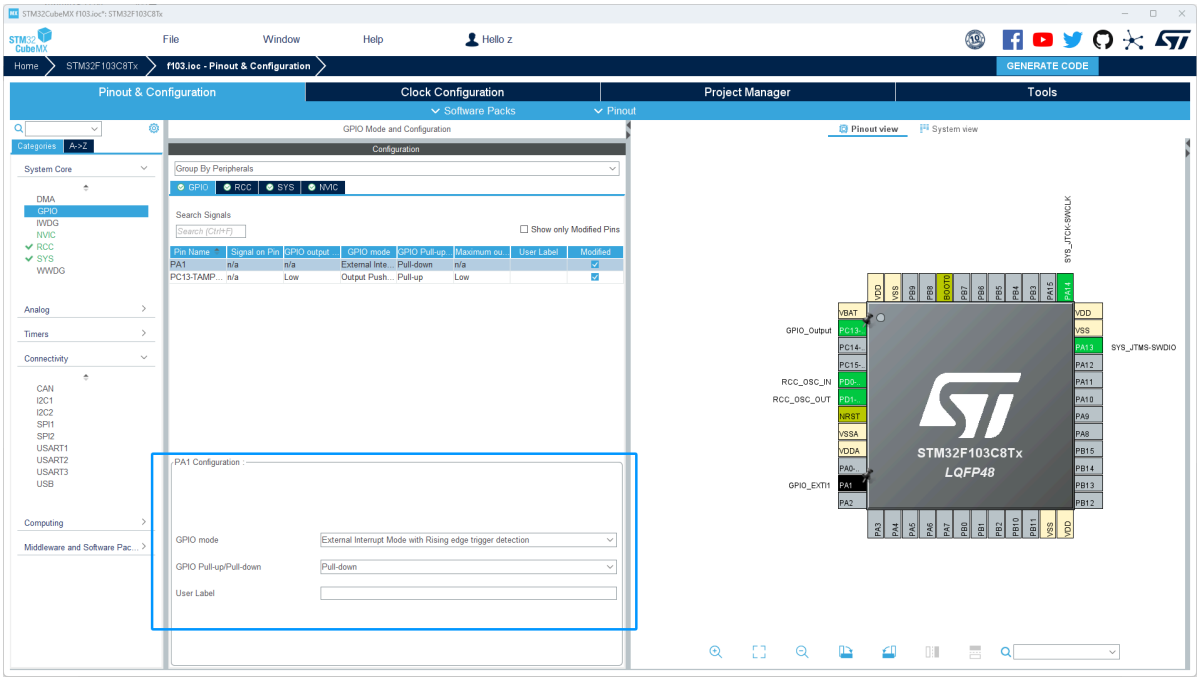
配置选项卡如下：`pull-up`和`pull-down`根据实际情况选（尽量让上升沿或下降沿更明显），不要选`no pull-up and no pull-down`



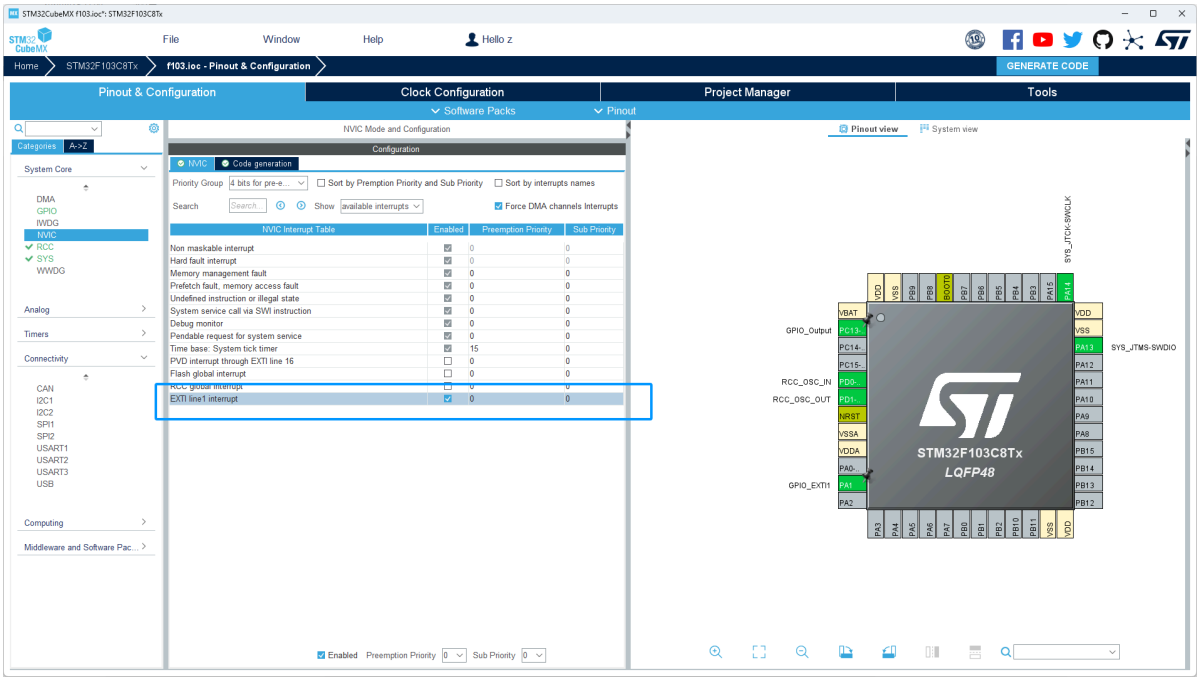
外部中断模式选这个：



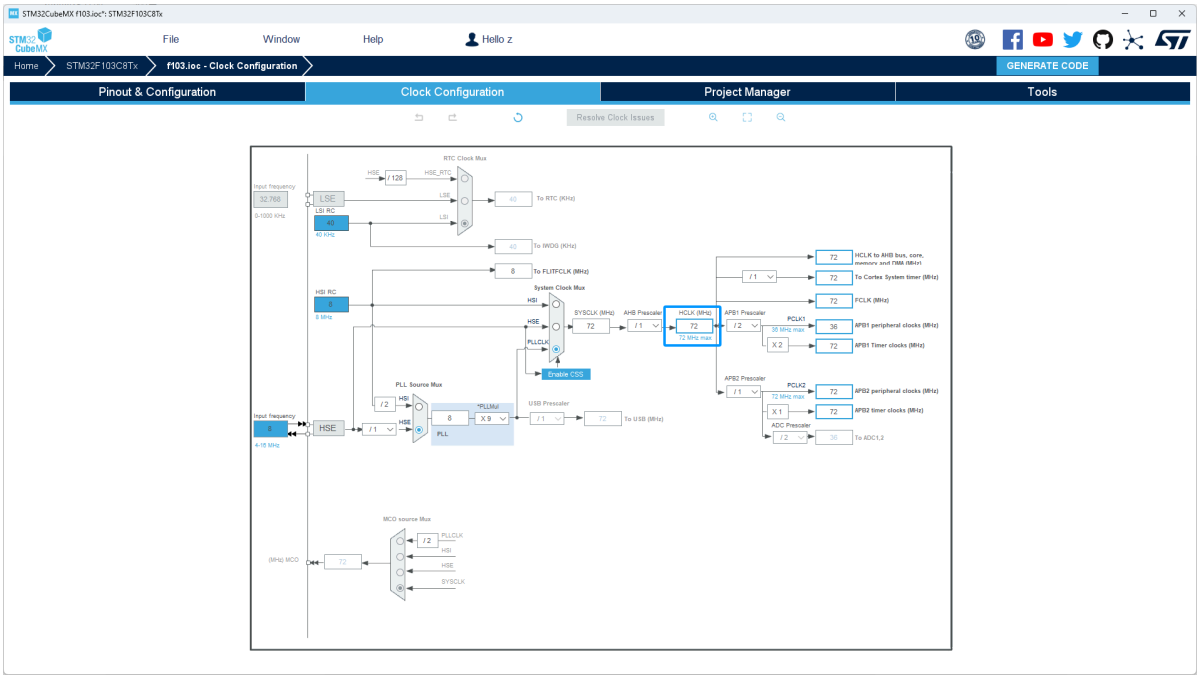
选项卡如下，上拉下拉同理



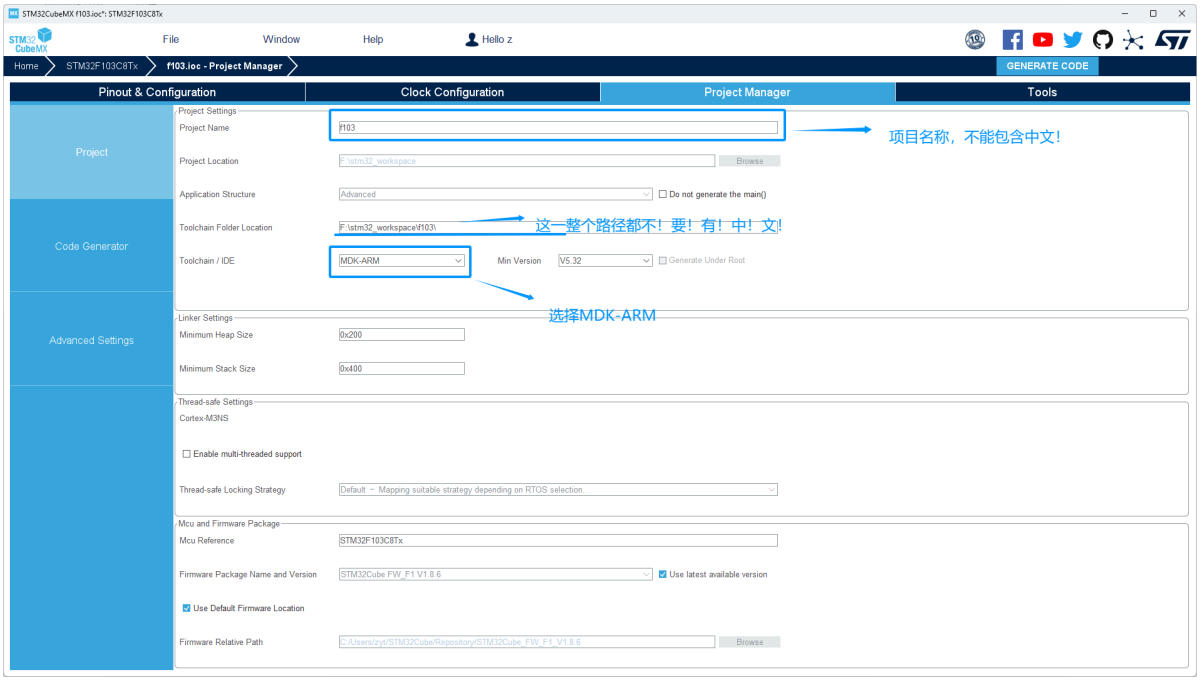
然后千万千万要把使能勾上

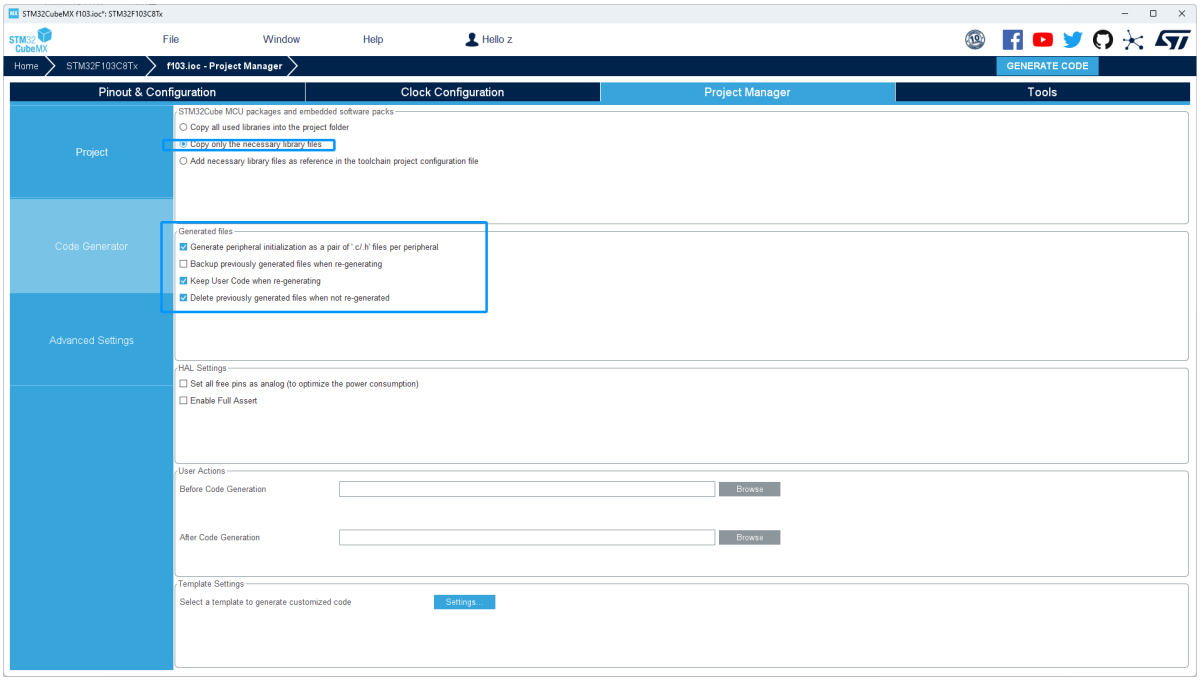


时钟树：直接把下框那个位置的数改成72，点击确认就好



项目管理：路径里面千万不要有中文字符（包括空格和中文标点）！！

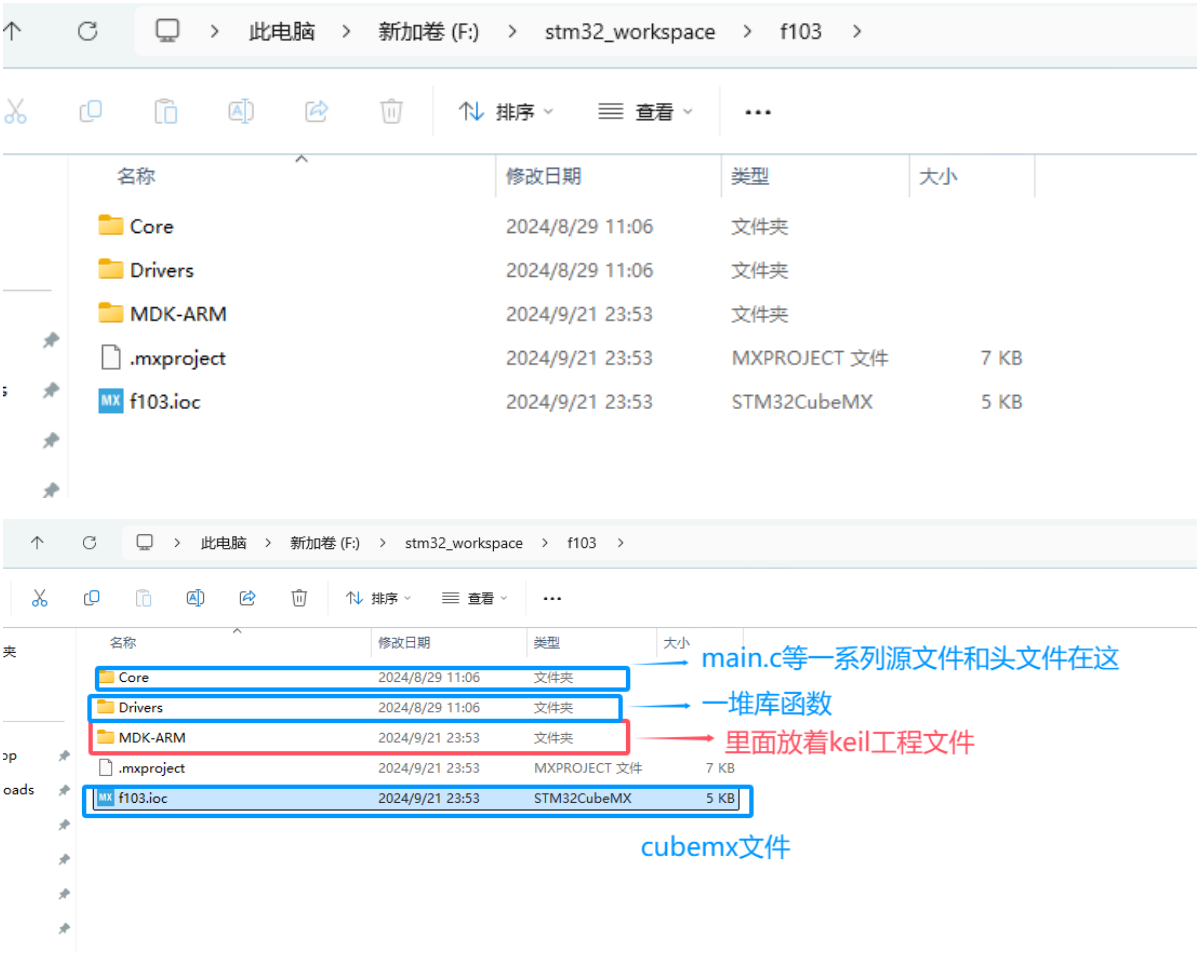




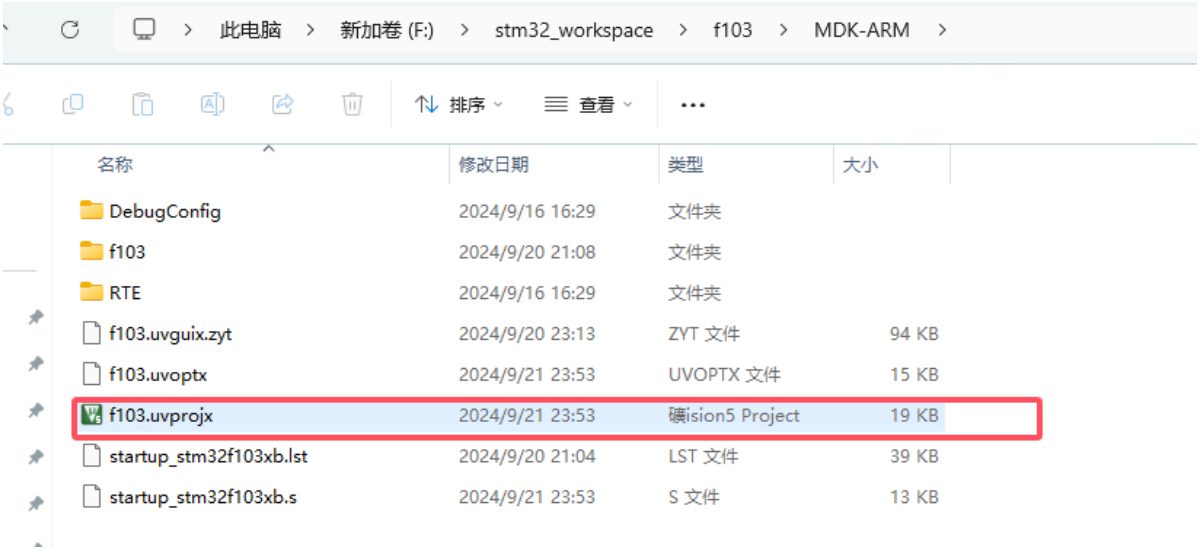
然后点击`generate code`

注意每一次修改cube都要重新generate！！！

此时工程文件就已经生成在上面填写的路径里面了，可以爬过去找找看

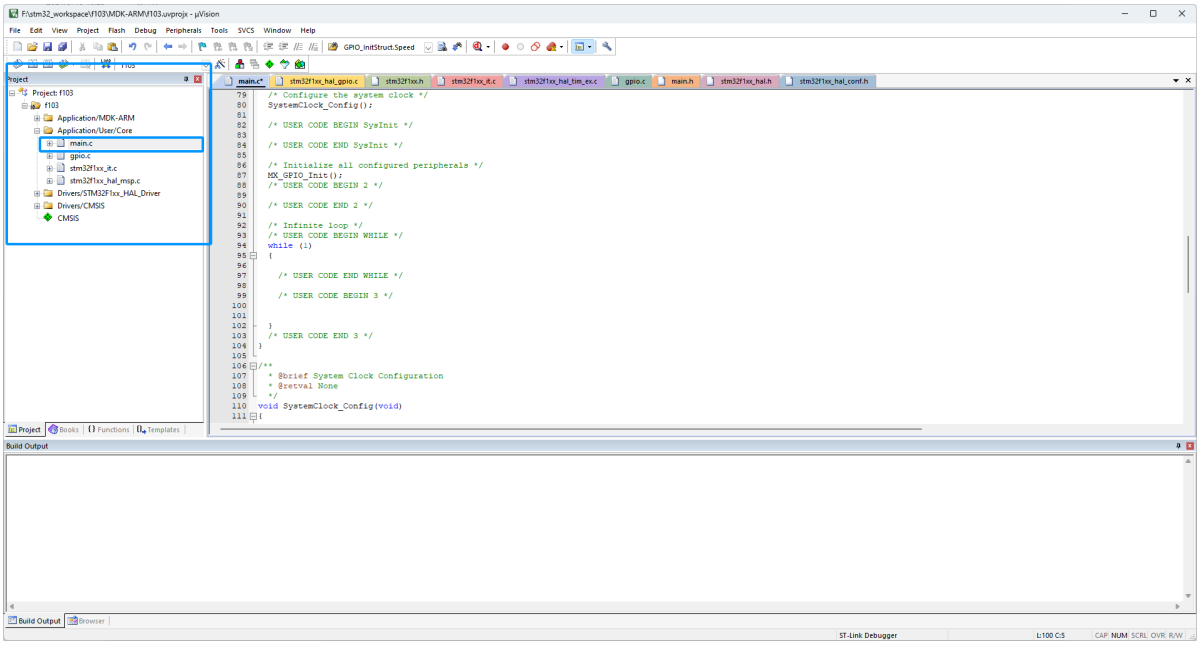


这个绿绿的就是keil文件，点开它可以直接打开keil （当然也可以直接在刚才的`open project`直达keil）

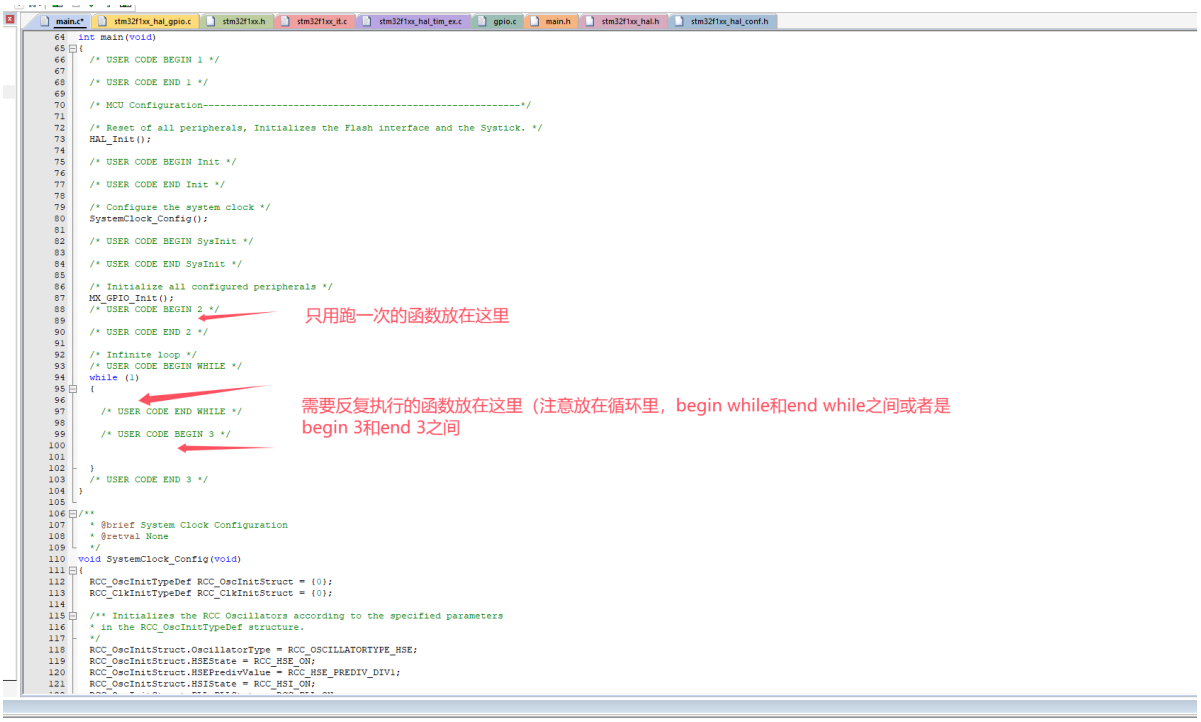


keil

在下面的文件里找到`main.c`（点击`Application\User\Core`左边的那个小加号可以下拉这个文件夹）



首先介绍一下函数都应该写在哪里




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

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */

  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
  {
    Error_Handler();
  }
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
```

中断回调函数的复写丢在这里

函数

常用函数

```
void HAL_GPIO_WritePin(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)
/**GPIO写函数
  *参数1：GPIO_TypeDef *GPIOx，GPIO的端口，如GPIOA
  *参数2：GPIO_Pin，GPIO的引脚号，如GPIO_PIN_13
  *参数3：写入电平  如GPIO_PIN_RESET
  */
```

```
GPIO_PinState HAL_GPIO_ReadPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
/**GPIO读函数
  *参数1：GPIO端口
  *参数2：GPIO引脚号
  *返回值：引脚电平
  */
```

```
void HAL_GPIO_TogglePin(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin)
/**GPIO电平翻转函数
  *参数1：GPIO端口
  *参数2：GPIO引脚号
  */
```

```
void HAL_Delay(uint16_t period)
/**延迟函数
  *参数：延迟时间，1000==1秒
  */
```

一些使用示例

（注意main函数中调用函数不要直接把上面的函数原型给粘贴下来！！！！具体用法如下）

提醒：代码该写到while里面的别写到while的中括号后面去了！然后也不要写到begin和end外面！！

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

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */

    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_1,0);

  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
```

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

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    if(HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_1)==1)
    {

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

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

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */

    HAL_GPIO_TogglePin(GPIOC,GPIO_PIN_13);
    HAL_Delay(500);

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

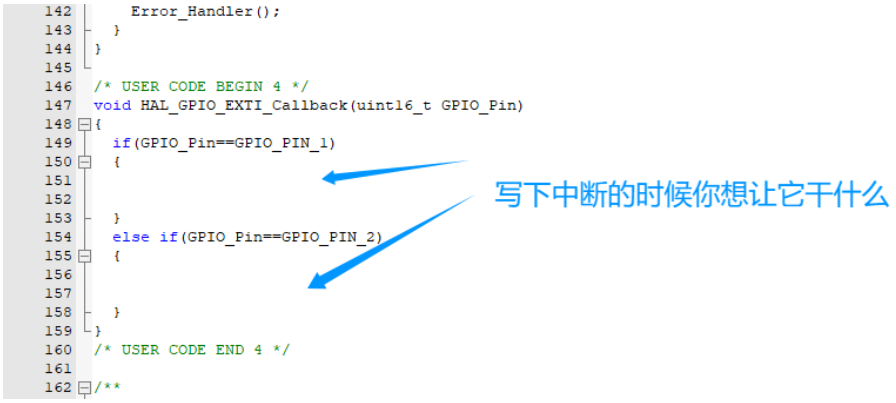
注意我们一般的主体任务都是放在while（1）死循环中的，但是在我们实现作业这些以练习为目的的小现象是，我们应该思考什么东西应该放主循环里面，什么东西应该在主循环之前（什么东西执行一遍就够了，什么东西需要反复执行）放错位置很可能导致现象不符合预期

```c
//中断回调函数的框架
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  if(GPIO_Pin==GPIO_PIN_1)  //因为所有外部中断最后都会调用这个函数，所以要判断一下是哪个引脚的
  {
  /*中断回调执行内容*/
```

```
  }
  else if(GPIO_Pin==GPIO_PIN_2)
  {
  /*中断回调执行内容*/

  }
}
```

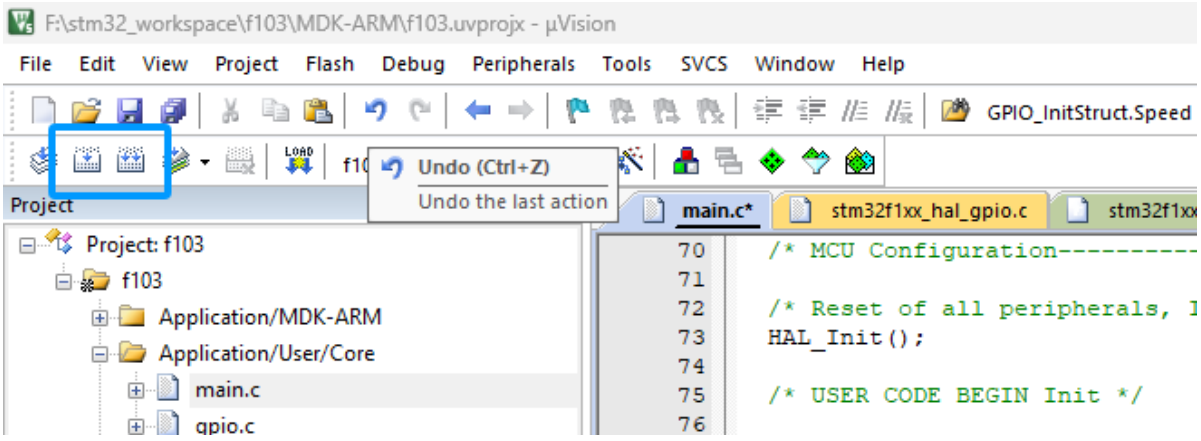
千万注意！！中断里面不要出现死循环和`HAL_Delay()`!!!!



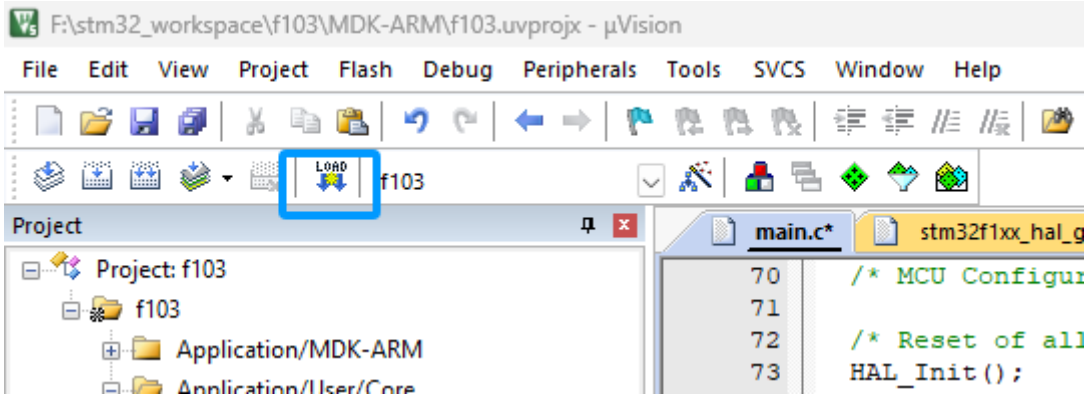
提醒：中断其实只是程序执行过程中的一个小插曲，程序执行完中断回调函数之后会回到主函数的while()之中!

编译和烧录

编译：这两个选项都可以，左边的是编译修改过的文件，速度较快，右边是全部重新编译，速度较慢



烧录：



注意每一次修改程序都要重新编译并且烧录！否则代码不会更新到板子上！