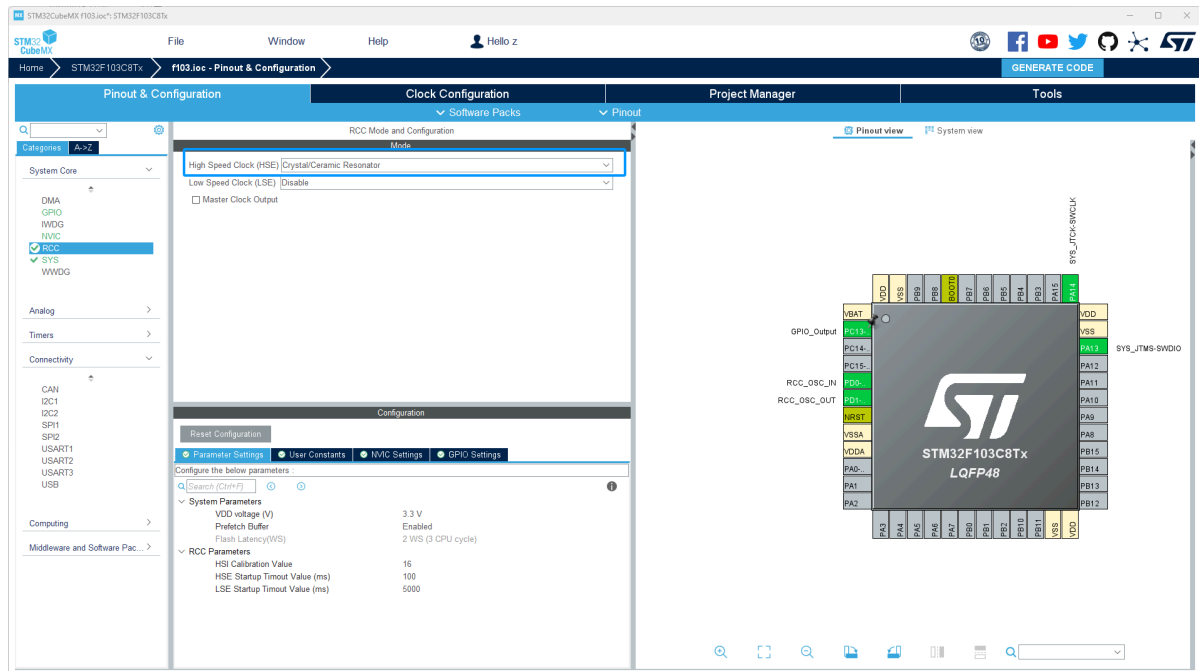


GPIO&EXTI

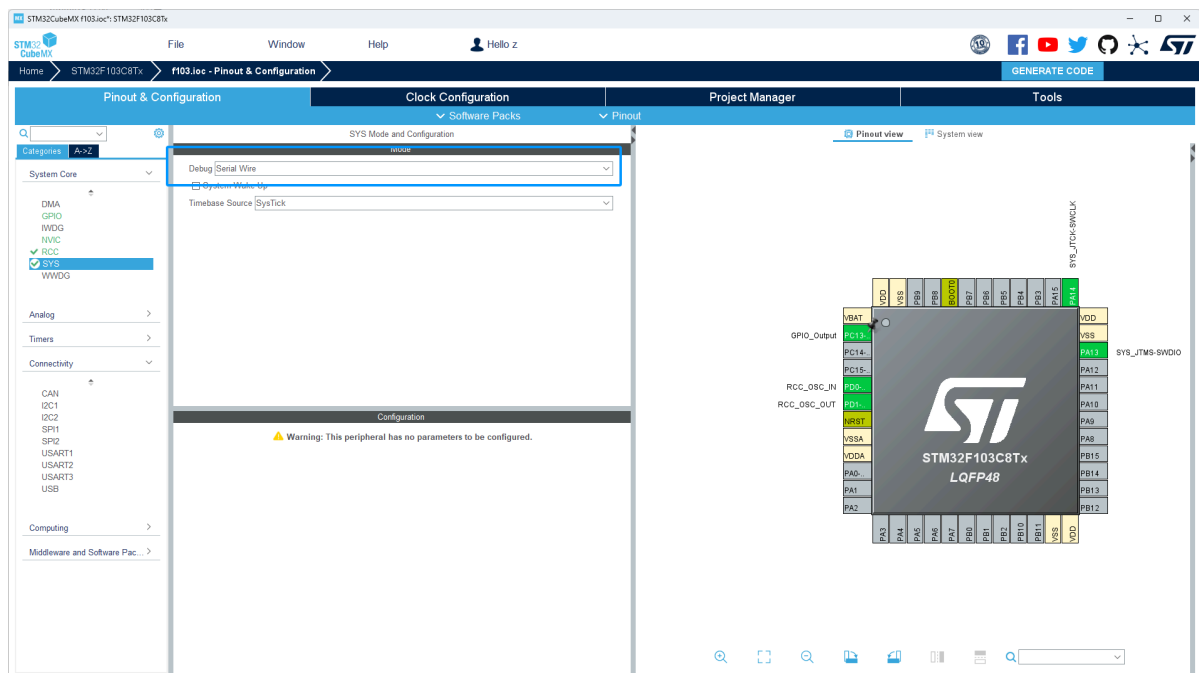
GPIO配置

CubeMX

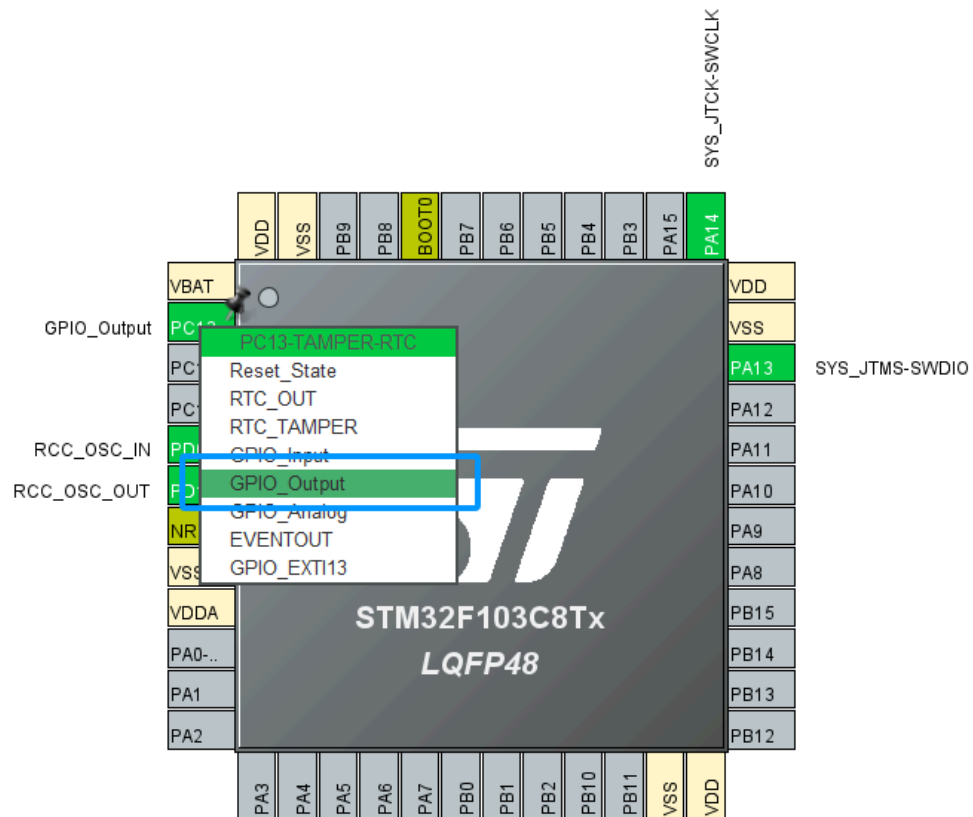
选好板子（stm32f103c8t6）之后，先把时钟源配置为如下模式



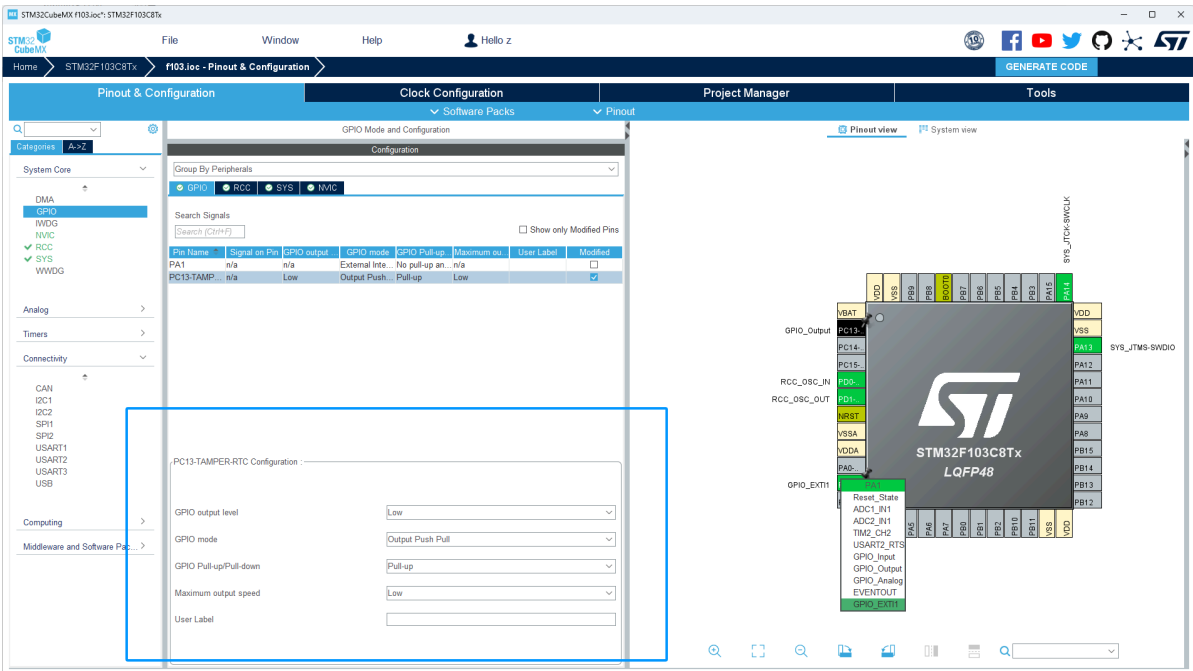
然后是SYS中的Debug:



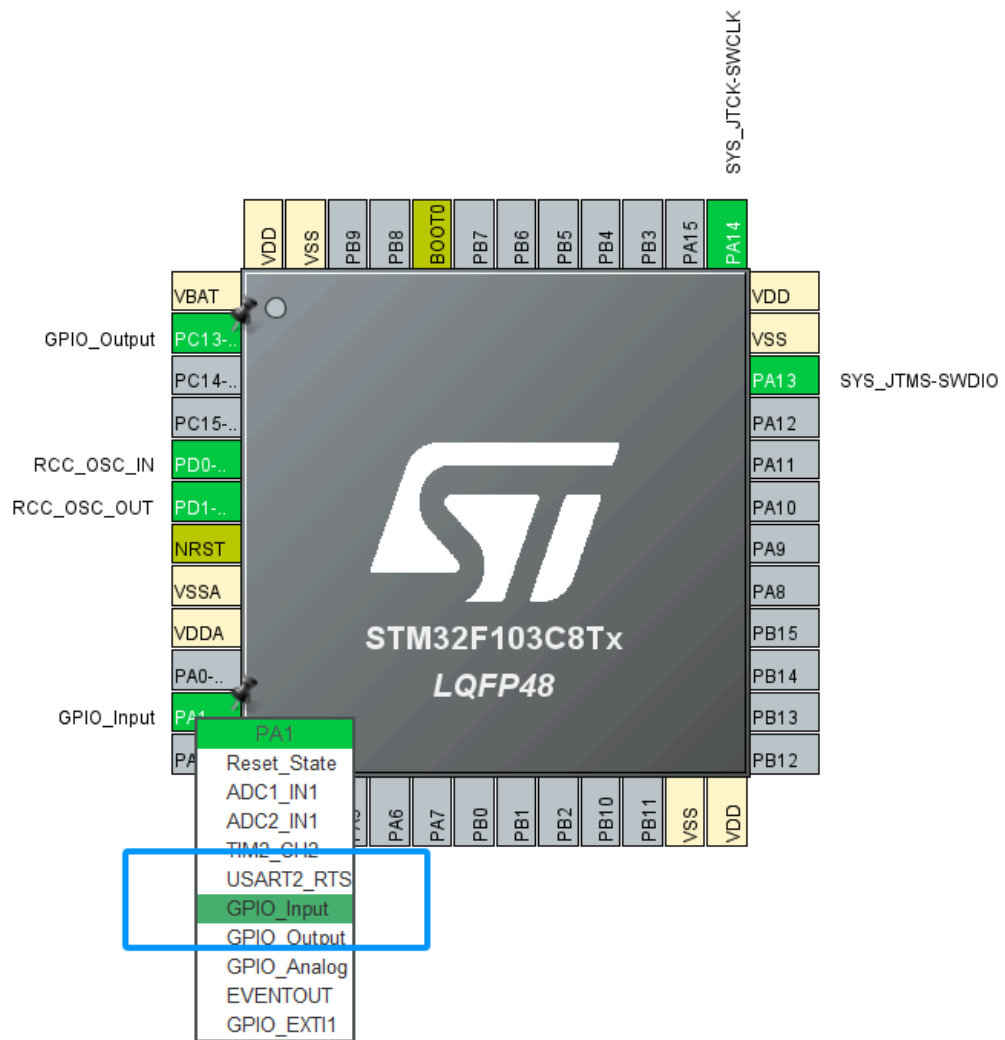
输出模式选这个：（输出选PC13是因为板载LED连在PC13上）



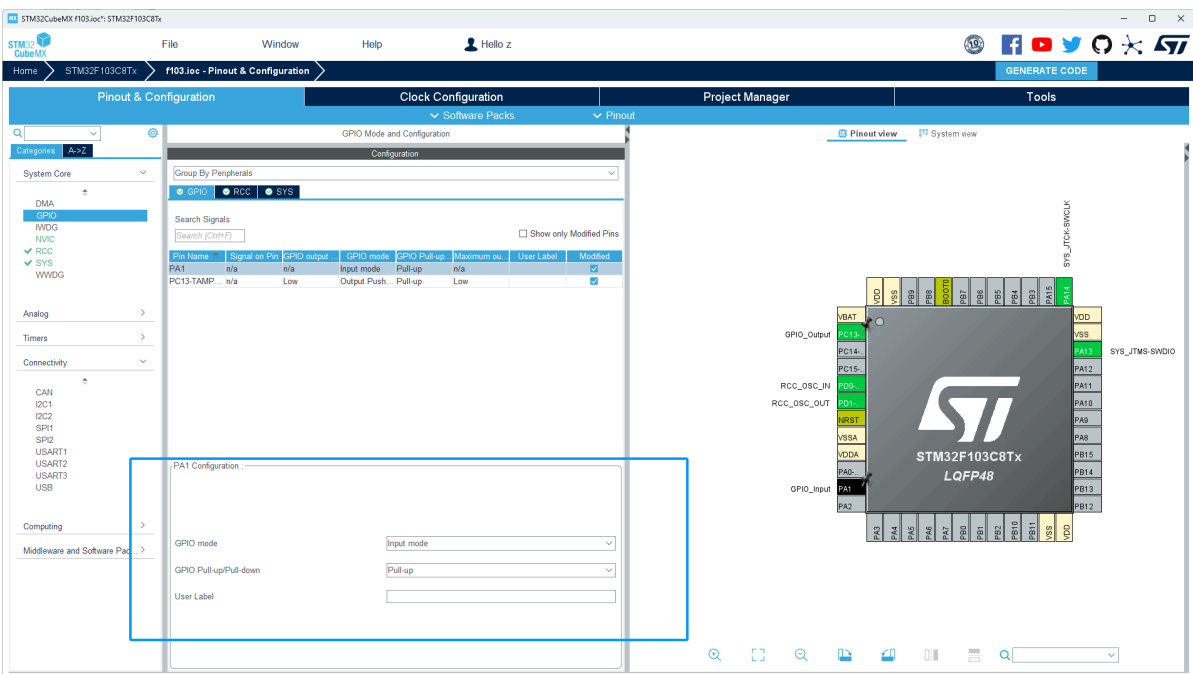
输出选项卡配置如下 注意一定要选择推挽模式



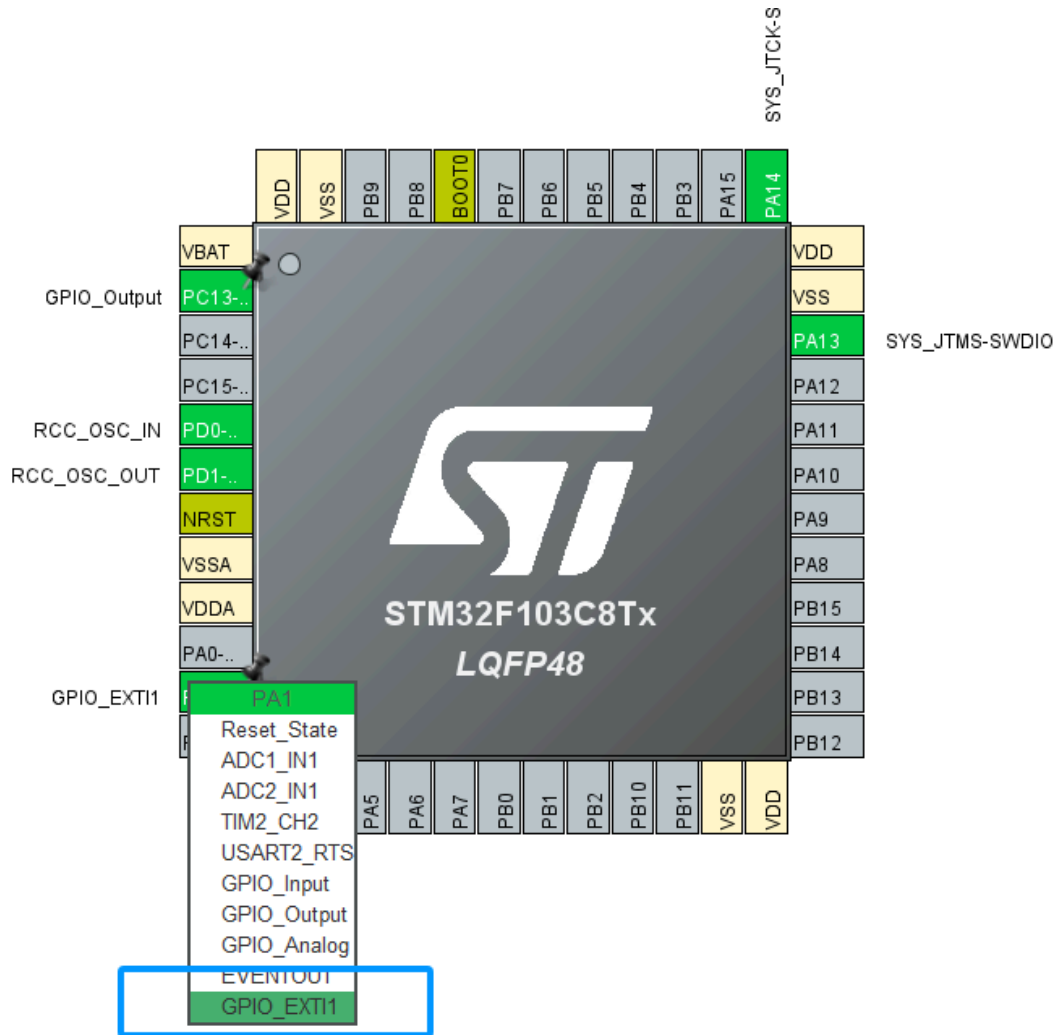
输入模式选这个：（不一定要选这个引脚，这里只是一个示例）



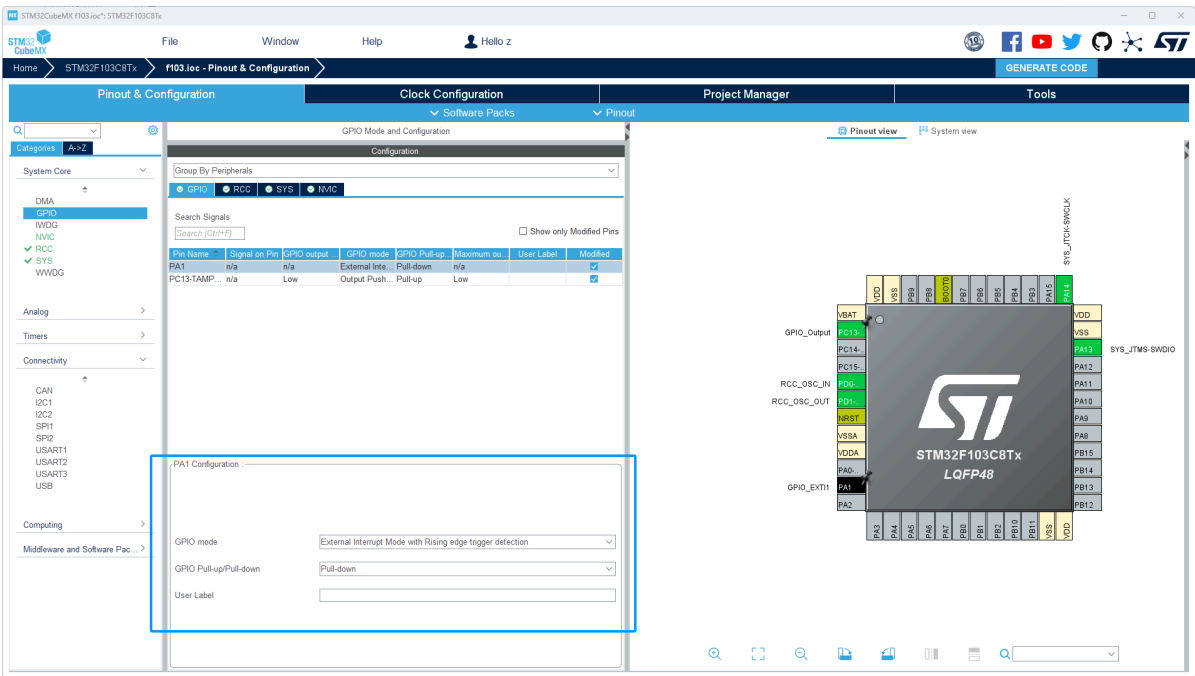
配置选项卡如下： pull-up 和 pull-down 根据实际情况选（尽量让上升沿或下降沿更明显），不要选 no pull-up and no pull-down



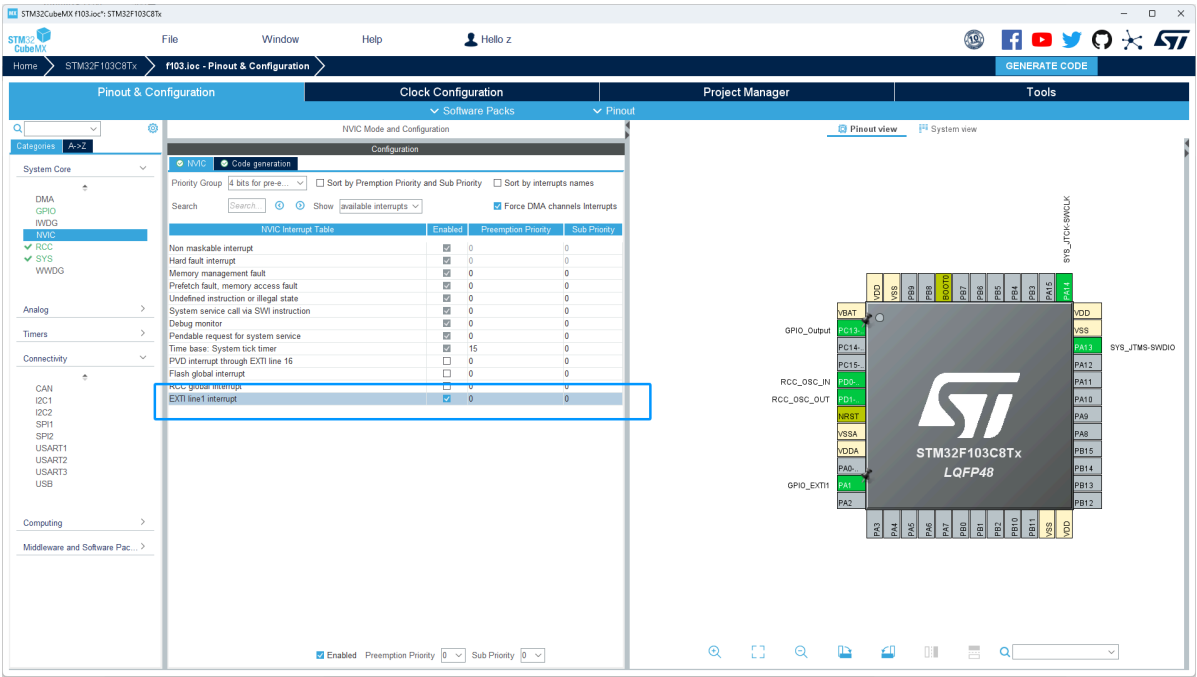
外部中断模式选这个：



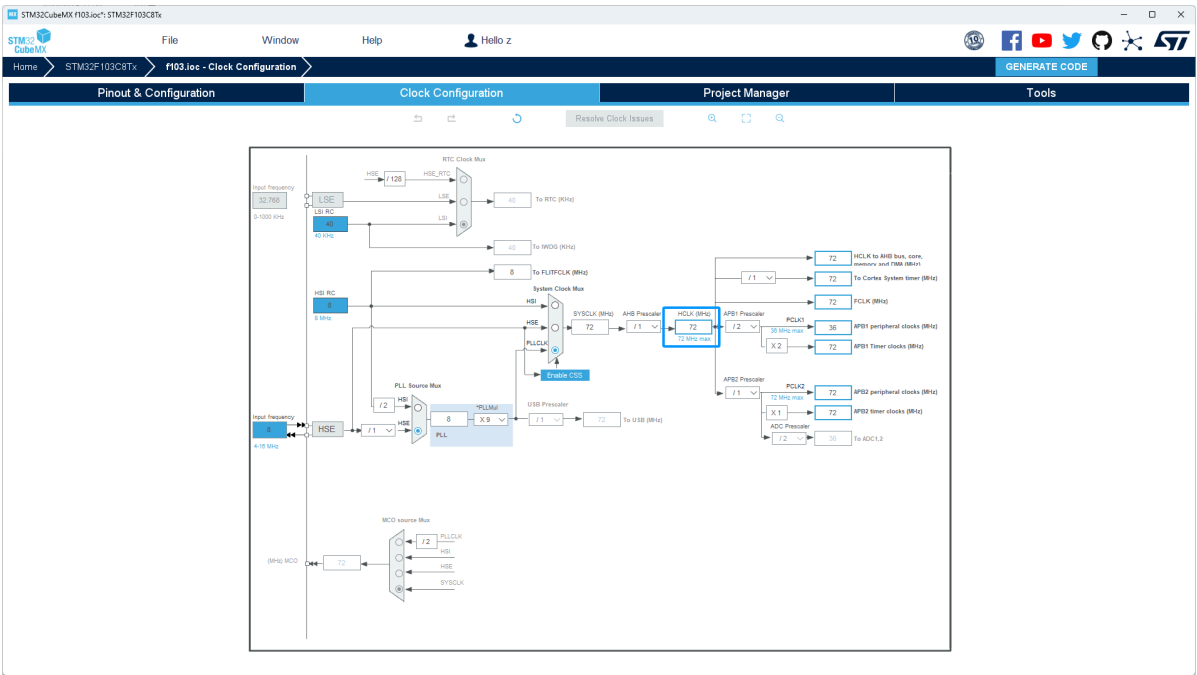
选项卡如下，上拉下拉同理



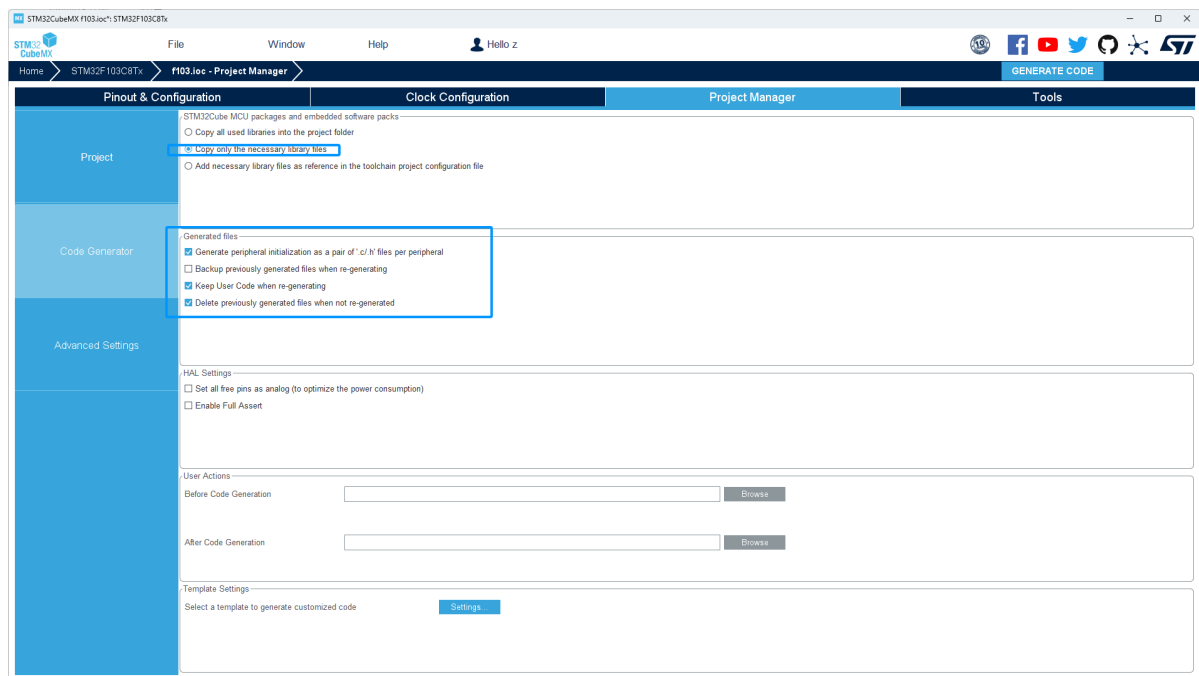
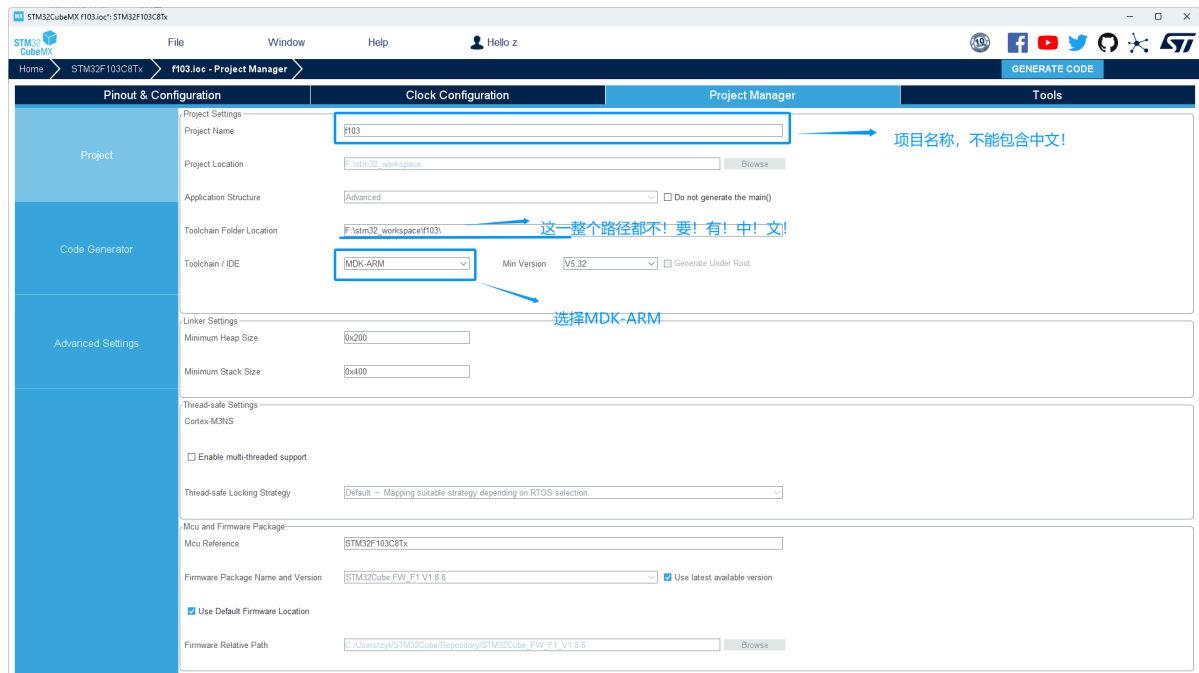
然后千万千万要把使能勾上



时钟树：直接把下框那个位置的数改成72，点击确认就好



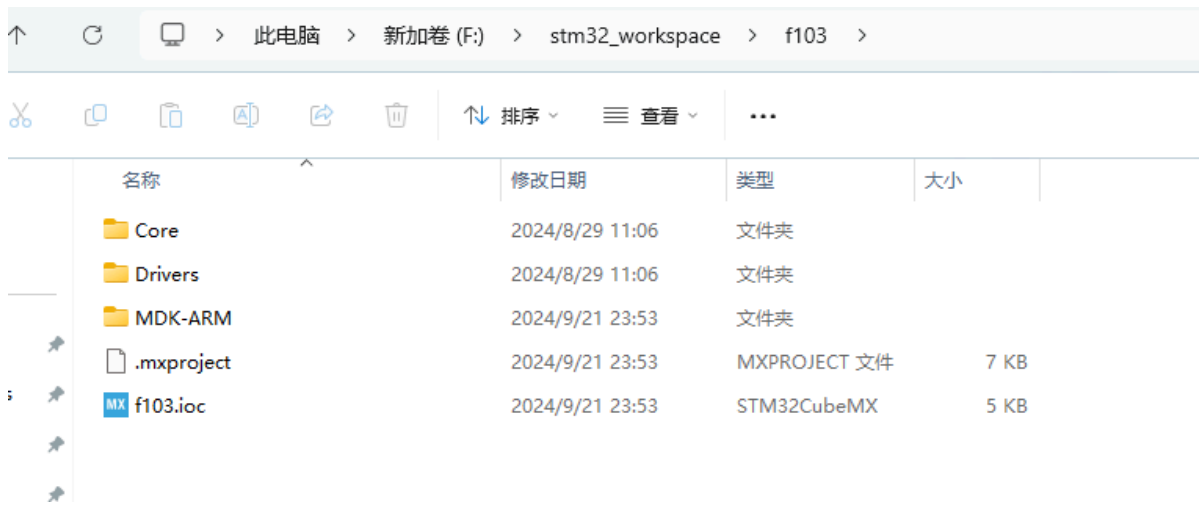
项目管理：路径里面千万不要有中文字符（包括空格和中文标点）！！



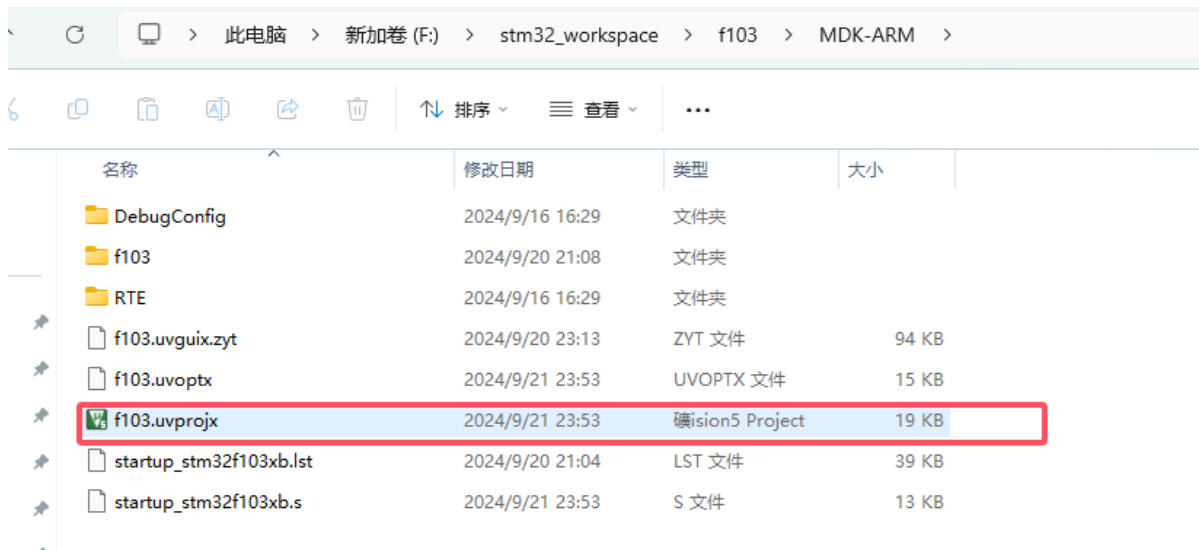
然后点击 generate code

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

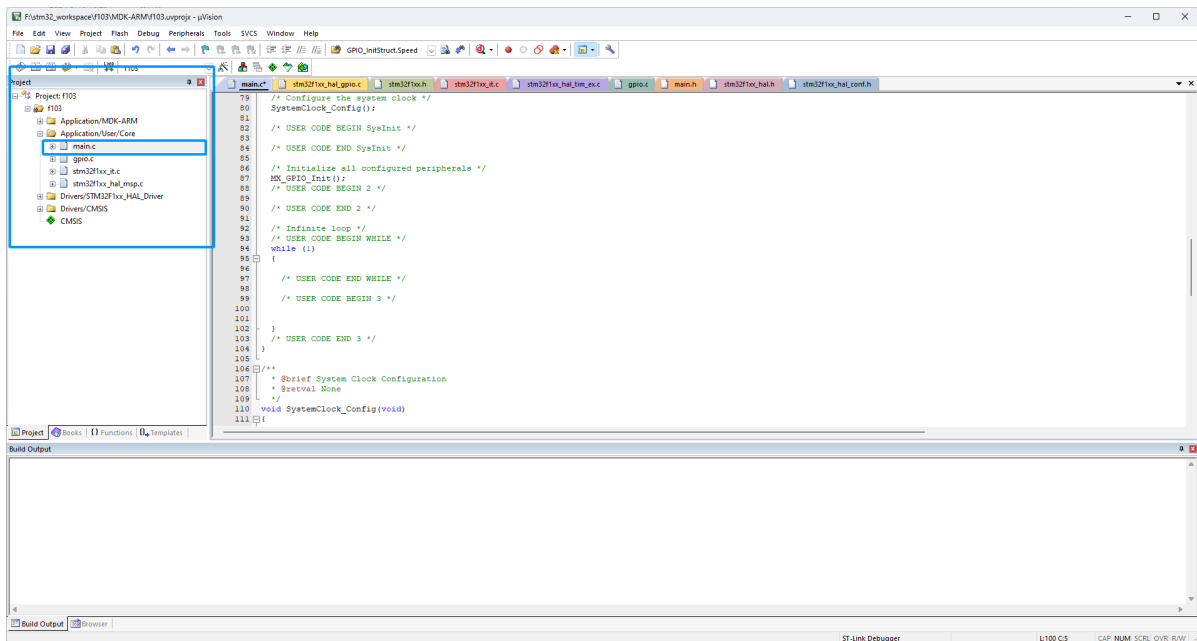


这个绿绿的就是keil文件，点开它可以直接打开keil（当然也可以直接在刚才的 open project 直达 keil）



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




```
main.c | stm32f1xx_hal_gpio.c | stm32f1xx.h | stm32f1xx_it.c | stm32f1xx_itm_ex.c | gpio.c | main.h | stm32f1xx_hal.h | stm32f1xx_hal_conf.h
64 int main(void)
65 {
66     /* USER CODE BEGIN 1 */
67     /* USER CODE END 1 */
68     /* MCU Configuration-----*/
69     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
70     HAL_Init();
71     /* USER CODE BEGIN Init */
72     /* USER CODE END Init */
73     /* Configure the system clock */
74     SystemClock_Config();
75     /* USER CODE BEGIN SysInit */
76     /* USER CODE END SysInit */
77     /* Initialize all configured peripherals */
78     MX_GPIO_Init();
79     /* USER CODE BEGIN 2 */
80     /* USER CODE END 2 */
81     /* Infinite loop */
82     /* USER CODE BEGIN WHILE */
83     while (1)
84     {
85         /* USER CODE END WHILE */
86         /* USER CODE BEGIN 3 */
87         /* USER CODE END 3 */
88     }
89     /* USER CODE END 3 */
90 }
91 /**
92  * @brief System Clock Configuration
93  * @retval None
94  */
95 void SystemClock_Config(void)
96 {
97     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
98     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
99     /** Initializes the RCC Oscillators according to the specified parameters
100      * in the RCC_OscInitTypeDef structure.
101      */
102     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
103     RCC_OscInitStruct.HSEState = RCC_HSE_ON;
104     RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
105     RCC_OscInitStruct.HSIState = RCC_HSI_ON;
106     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
107     RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
108     RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
109     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
110     {
111         Error_Handler();
112     }
113     /** Initializes the CPU, AHB and APB buses clocks
114      */
115     RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
116         |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
117     RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
118     RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
119     RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
120     RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
121     if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
122     {
123         Error_Handler();
124     }
125 }
```

只用跑一次的函数放在这里

需要反复执行的函数放在这里（注意放在循环里，begin while和end while之间或者是begin 3和end 3之间

```
102 }
103 /* USER CODE END 3 */
104 }
105 /**
106  * @brief System Clock Configuration
107  * @retval None
108  */
109 void SystemClock_Config(void)
110 {
111     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
112     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
113     /** Initializes the RCC Oscillators according to the specified parameters
114      * in the RCC_OscInitTypeDef structure.
115      */
116     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
117     RCC_OscInitStruct.HSEState = RCC_HSE_ON;
118     RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
119     RCC_OscInitStruct.HSIState = RCC_HSI_ON;
120     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
121     RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
122     RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
123     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
124     {
125         Error_Handler();
126     }
127     /** Initializes the CPU, AHB and APB buses clocks
128      */
129     RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
130         |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
131     RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
132     RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
133     RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
134     RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
135     if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
136     {
137         Error_Handler();
138     }
139     /* USER CODE BEGIN 4 */
140     /* USER CODE END 4 */
141     /**
142      * @brief This function is executed in case of error occurrence.
143      * @retval None
144      */
145 void Error_Handler(void)
146 {
147     /* USER CODE BEGIN 5 */
148     /* USER CODE END 5 */
149 }
```

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!!

```

63  L  */
64  int main(void)
65  {
66      /* USER CODE BEGIN 1 */
67
68      /* USER CODE END 1 */
69
70      /* MCU Configuration-----*/
71
72      /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
73      HAL_Init();
74
75      /* USER CODE BEGIN Init */
76
77      /* USER CODE END Init */
78
79      /* Configure the system clock */
80      SystemClock_Config();
81
82      /* USER CODE BEGIN SysInit */
83
84      /* USER CODE END SysInit */
85
86      /* Initialize all configured peripherals */
87      MX_GPIO_Init();
88      /* USER CODE BEGIN 2 */
89
90      /* USER CODE END 2 */
91
92      /* Infinite loop */
93      /* USER CODE BEGIN WHILE */
94      while (1)
95      {
96          /* USER CODE END WHILE */
97
98          /* USER CODE BEGIN 3 */
99
100         HAL_GPIO_WritePin(GPIOC,GPIO_PIN_1,0);
101     }
102     /* USER CODE END 3 */
103 }
104
105
106 /**
107  * @brief System Clock Configuration
108  * @retval None
109  */
110 void SystemClock_Config(void)
111 {
112     RCC_OscInitTypeDef RCC_OscInitStruct = {0};

```

```

L  */
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 */
    }
}

```

```

63  */
64  int main(void)
65  {
66      /* USER CODE BEGIN 1 */
67
68      /* USER CODE END 1 */
69
70      /* MCU Configuration-----*/
71
72      /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
73      HAL_Init();
74
75      /* USER CODE BEGIN Init */
76
77      /* USER CODE END Init */
78
79      /* Configure the system clock */
80      SystemClock_Config();
81
82      /* USER CODE BEGIN SysInit */
83
84      /* USER CODE END SysInit */
85
86      /* Initialize all configured peripherals */
87      MX_GPIO_Init();
88      /* USER CODE BEGIN 2 */
89
90      /* USER CODE END 2 */
91
92      /* Infinite loop */
93      /* USER CODE BEGIN WHILE */
94      while (1)
95      {
96          /* USER CODE END WHILE */
97
98          /* USER CODE BEGIN 3 */
99
100      HAL_GPIO_TogglePin(GPIOC,GPIO_PIN_13);
101      HAL_Delay(500);|
102
103      }
104      /* USER CODE END 3 */
105  }
106

```

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

```

//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() !!!!

```

142 |     Error_Handler();
143 | }
144 | }
145 |
146 | /* USER CODE BEGIN 4 */
147 | void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
148 | {
149 |     if(GPIO_Pin==GPIO_PIN_1)
150 |     {
151 |
152 |
153 |     }
154 |     else if(GPIO_Pin==GPIO_PIN_2)
155 |     {
156 |
157 |
158 |     }
159 | }
160 | /* USER CODE END 4 */
161 |
162 | /**

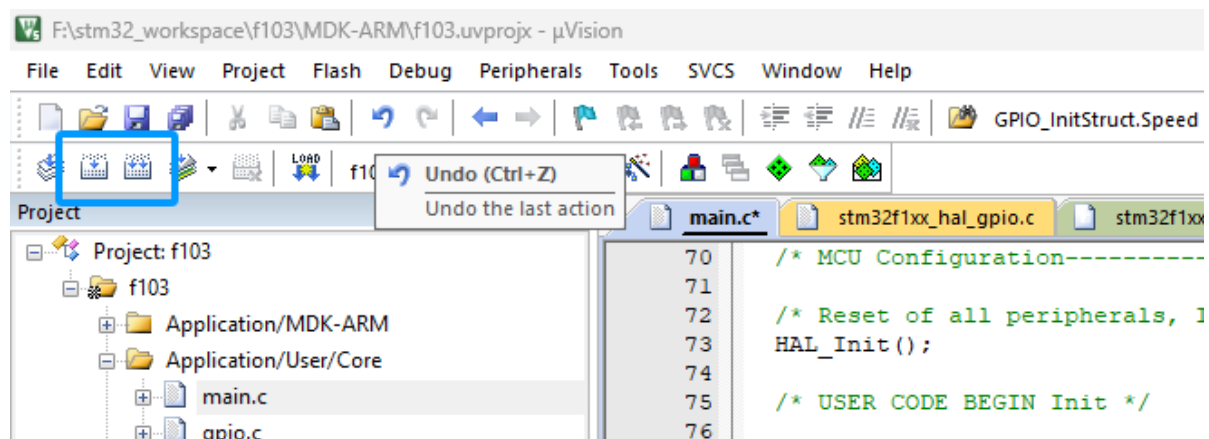
```

写下中断的时候你想让它干什么

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()-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！

