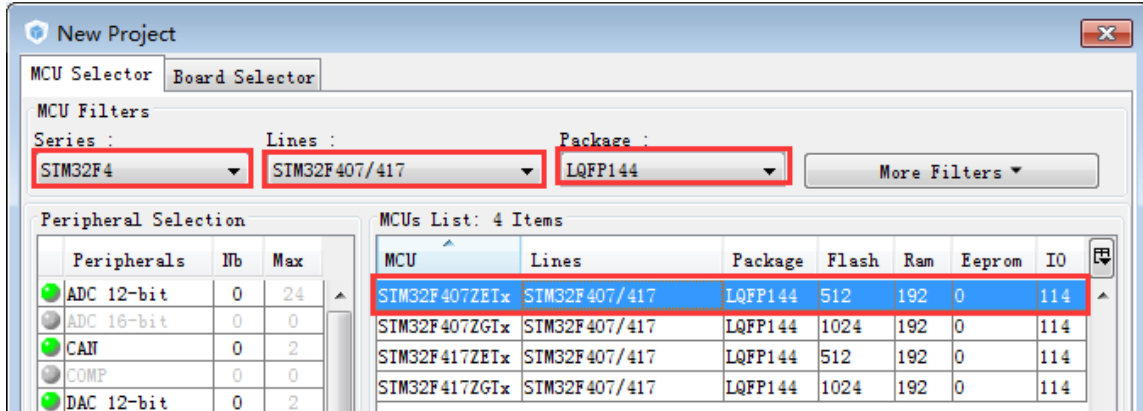


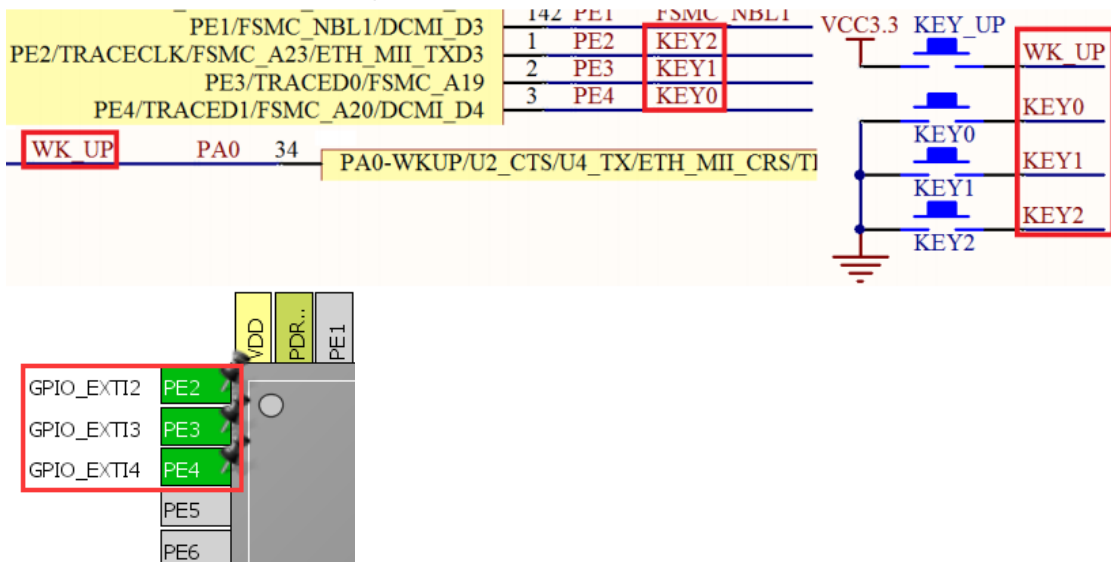
STM32Cube 学习之四：外部中断

假设已经安装好 STM32CubeMX 和 STM32CubeF4 支持包。

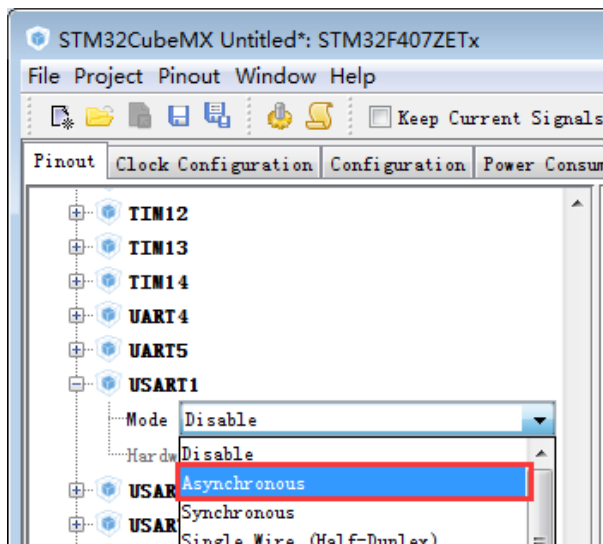
Step1.打开 STM32CubeMX，点击 “New Project”，选择芯片型号，STM32F407ZETx。



Step2.在 Pinout 界面下配置 KEY0,KEY1，KEY2 按键输入引脚为 EXTI 功能。



Step3.配置 USART，用于按键信息输出。



PA12	
PA11	
PA10	USART1_RX
PA9	USART1_TX
PA8	
PC9	

时钟树的配置不作任何修改，使用内部 16M 时钟源，内核时钟 16M。

Step4.配置串口参数和 GPIO 的参数。

在 configuration 界面中点击 USART1 按钮，可以进入参数配置界面，使用默认参数即可。

USART1 Configuration

Parameter Settings User Constants NVIC Settings DMA Settings GPIO Settings

Configure the below parameters :

Search : Search (Ctrl+F)

Basic Parameters

Baud Rate	115200 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1

Advanced Parameters

Data Direction	Receive and Transmit
Over Sampling	16 Samples

在 configuration 界面中点击 GPIO 按钮，配置 GPIO 的上拉电阻和中断触发沿。

Pin Configuration

GPIO

Search Signals

Search (Cr...

Show only Modified Pins

Pin Name	Signal on...	GPIO outp...	GPIO mode	GPIO Pull...	Maximum o...	User Label	Modified
PE2	n/a	n/a	External I...	Pull-up	n/a		✓
PE3	n/a	n/a	External I...	Pull-up	n/a		✓
PE4	n/a	n/a	External I...	Pull-up	n/a		✓

PE4 Configuration :

GPIO mode: External Interrupt Mode with Falling edge trigger detection

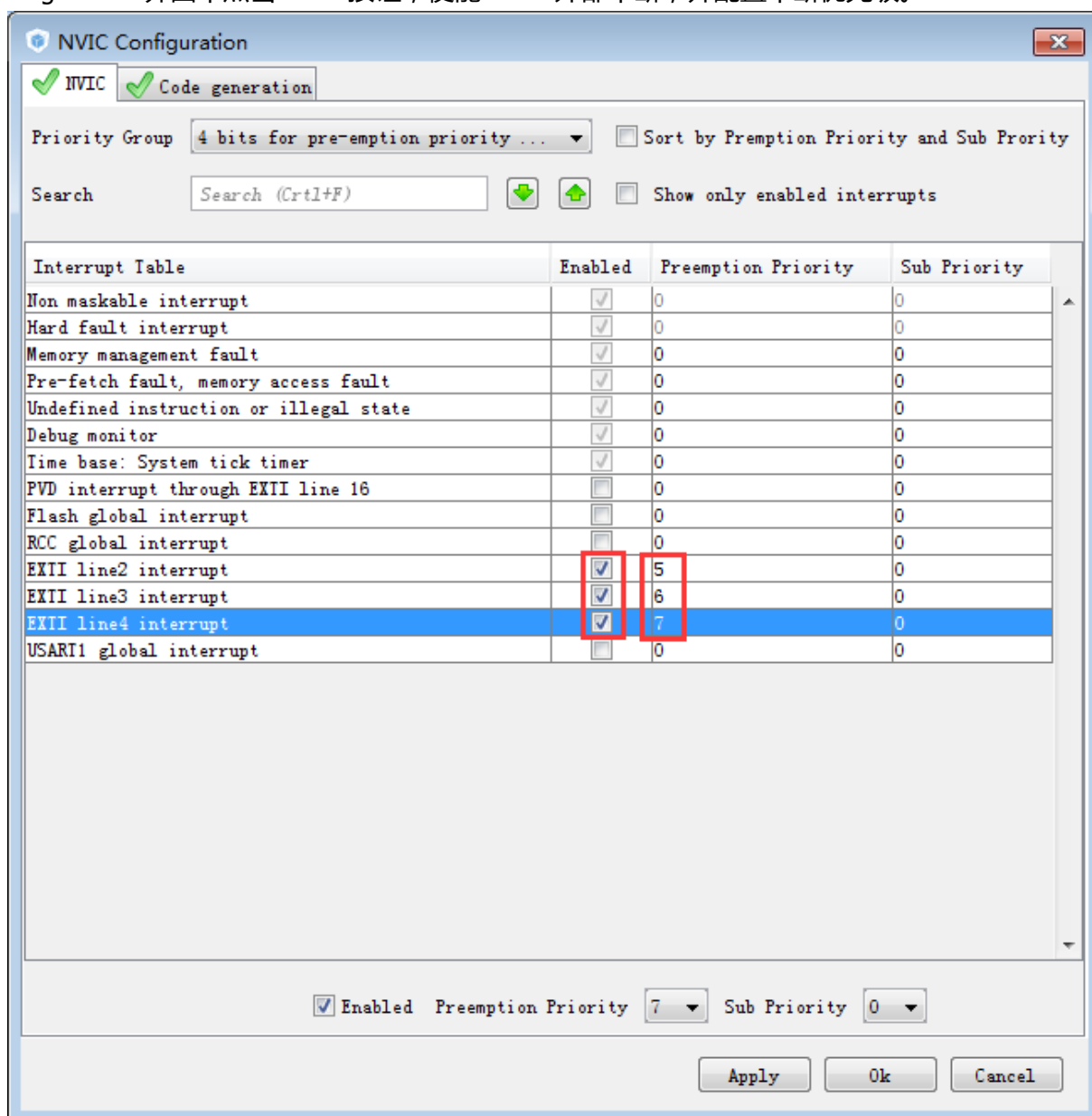
GPIO Pull-up/Pull-down: Pull-up

User Label:

Group By IP

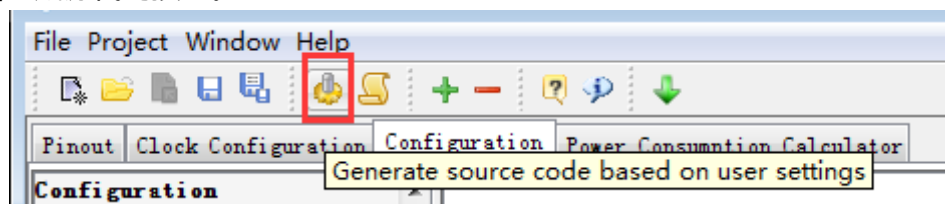
Apply Ok Cancel

在 configuration 界面中点击 NVIC 按钮，使能 GPIO 外部中断，并配置中断优先级。

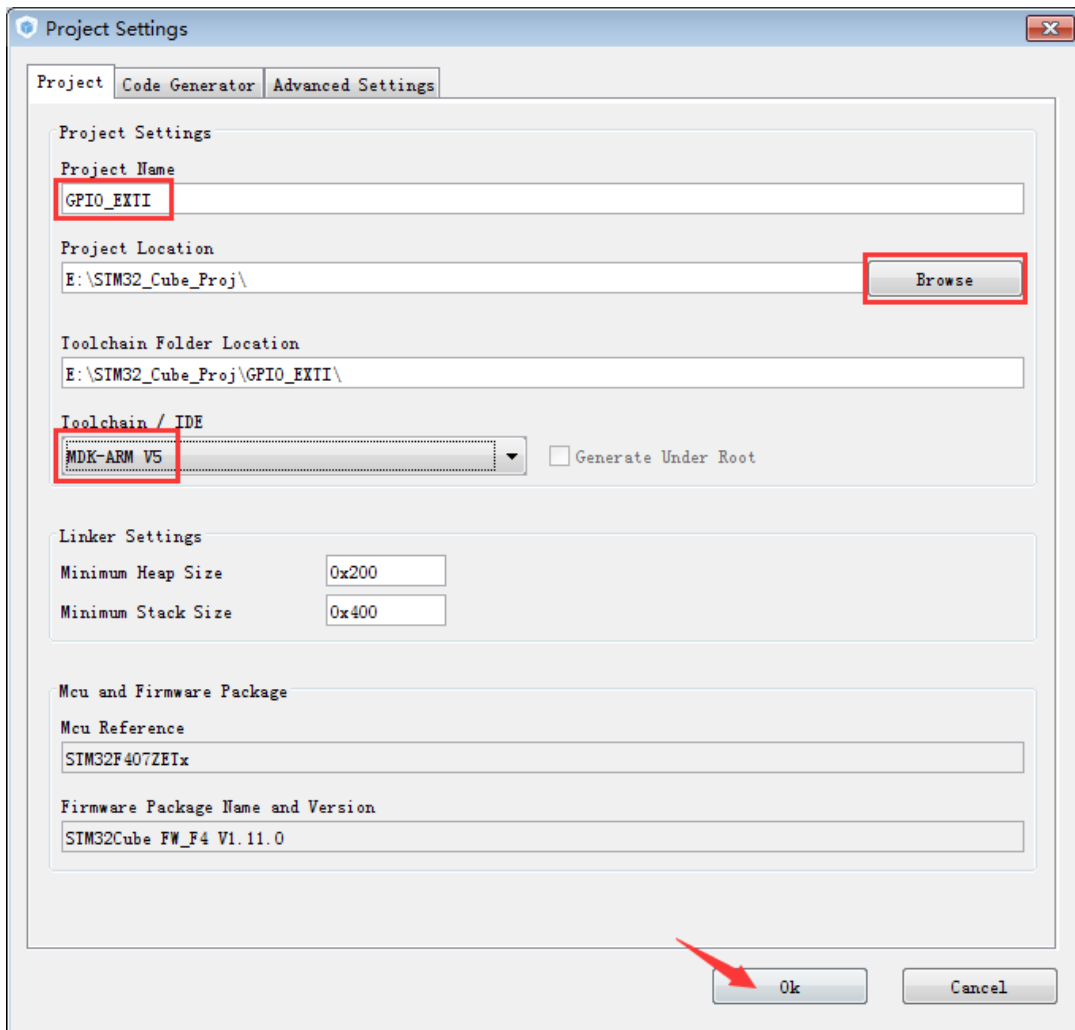


Step5.生成源代码。

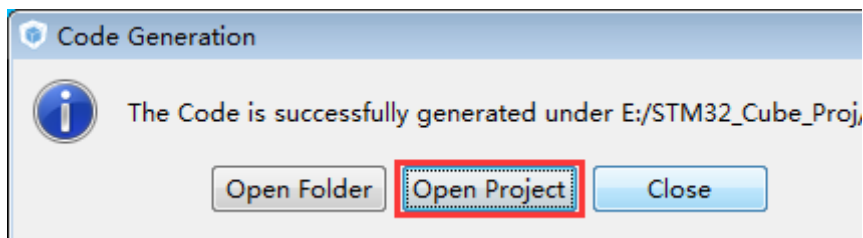
点击生成源代码按钮。



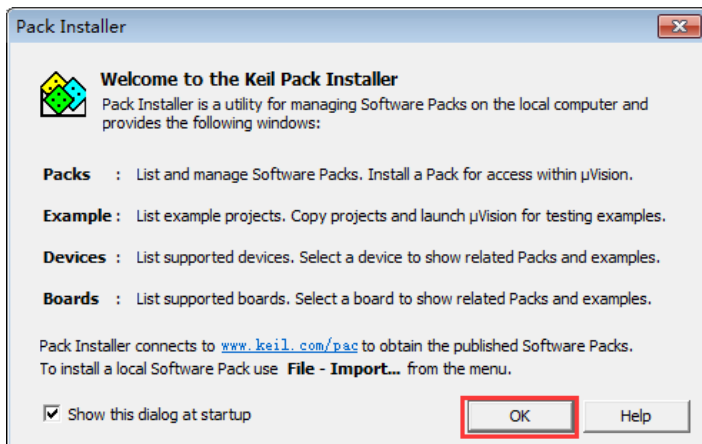
在设置界面中输入工程名，保存路径，工程 IDE 类型，点 OK 即可。



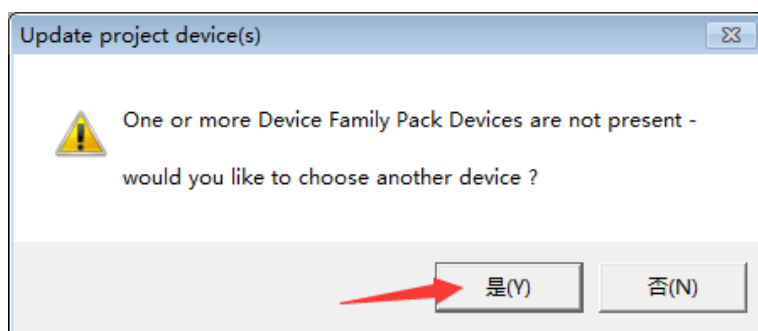
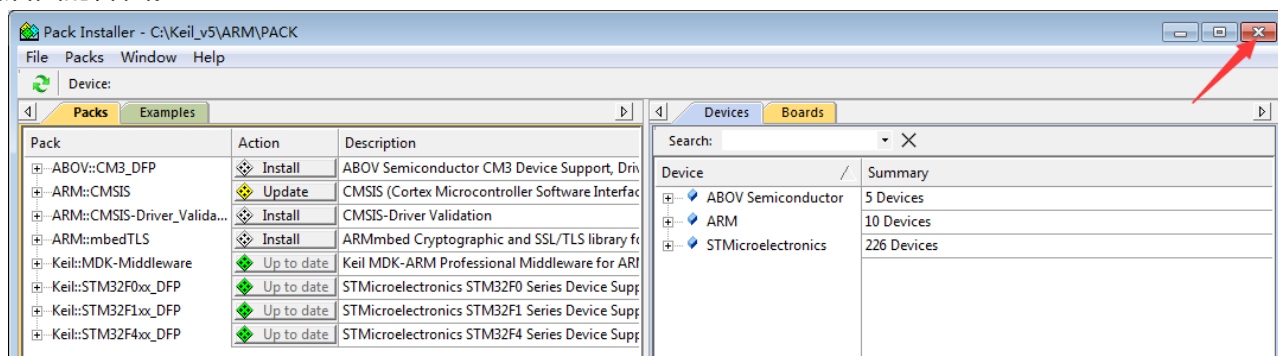
生成代码完成后可直接打开工程。



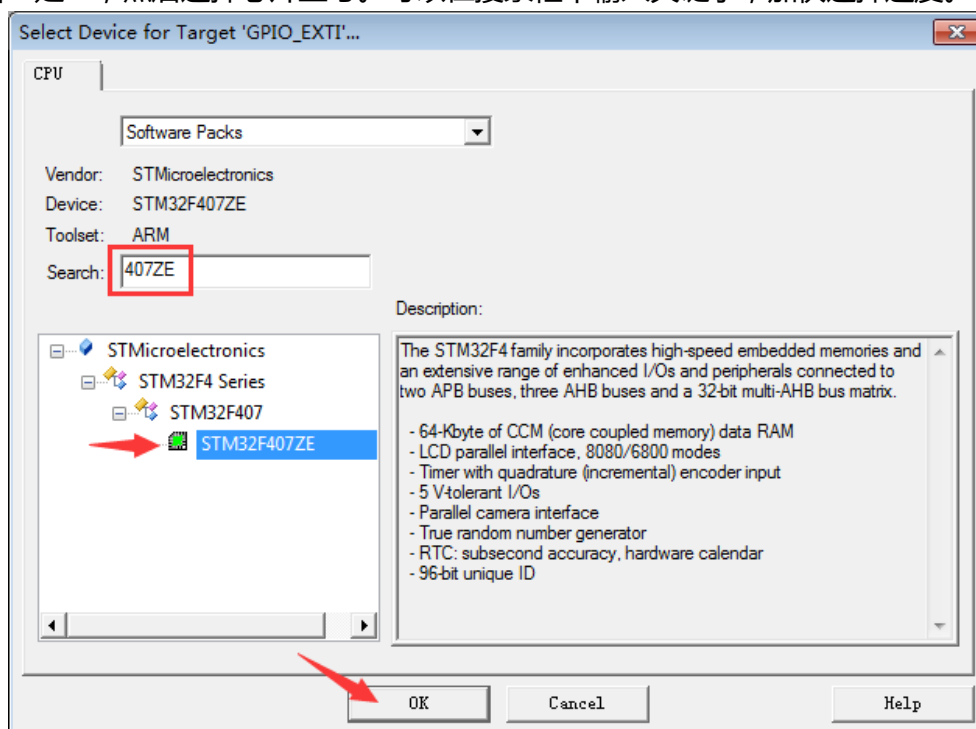
弹出如下对话框时，如果已经安装了 F4 的支持包，则点击 OK 关闭。如果没有安装，则点击界面中的 www.keil.com/... 链接，找到芯片的支持包，然后安装。



关闭后面的界面。



点击“是”，然后选择芯片型号。可以在搜索框中输入关键字，加快选择速度。



Step6.添加功能代码。

在/* USER CODE BEGIN 4 */和/* USER CODE END 4 */注释之间加入下述代码。实现的功能是，按键触发中断后，通过串口发送相应信息。要注意的是，按一次按键，可能会发送多条信息，原因是按键存在的抖动，这和事实相符。

```

182  /* USER CODE BEGIN 4 */
183  void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
184  {
185      if (GPIO_Pin & GPIO_PIN_2) {
186          HAL_UART_Transmit(&huart1, "PIN_2 EXTI\r\n", 12, 10);
187      }
188      if (GPIO_Pin & GPIO_PIN_3) {
189          HAL_UART_Transmit(&huart1, "PIN_3 EXTI\r\n", 12, 10);
190      }
191      if (GPIO_Pin & GPIO_PIN_4) {
192          HAL_UART_Transmit(&huart1, "PIN_4 EXTI\r\n", 12, 10);
193      }
194  }
195  /* USER CODE END 4 */

```

在 CubeMX 生成的程序框架中，GPIO 外部中断共用一个回调函数接口。要想知道是哪个中断源，就判断其输入参数 GPIO_Pin。

问题：怎么知道外设对应的回调函数名称呢？

答：这其实不需要记忆，可以通过两种途径找到。

方法 1：打开外设对应的头文件，如 GPIO 就打开 stm32f4xx_hal_gpio.h，

UART 就打开 stm32f4xx_hal_uart.h。找到以“HAL_”开头，以“Callback”结尾的函数列表，如下图：

```

252  /* IO operation functions *****/
253  GPIO_PinState HAL_GPIO_ReadPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
254  void HAL_GPIO_WritePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState);
255  void HAL_GPIO_TogglePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
256  HAL_StatusTypeDef HAL_GPIO_LockPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
257  void HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin);
258  void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin);
259
654  HAL_StatusTypeDef HAL_UART_DMAResume(UART_HandleTypeDef *huart);
655  HAL_StatusTypeDef HAL_UART_DMAStop(UART_HandleTypeDef *huart);
656  void HAL_UART_IRQHandler(UART_HandleTypeDef *huart);
657  void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart);
658  void HAL_UART_TxHalfCpltCallback(UART_HandleTypeDef *huart);
659  void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart);
660  void HAL_UART_RxHalfCpltCallback(UART_HandleTypeDef *huart);
661  void HAL_UART_ErrorCallback(UART_HandleTypeDef *huart);

```

通过点击右键，然后点击“Go To Definition...”可以定位到函数实现。

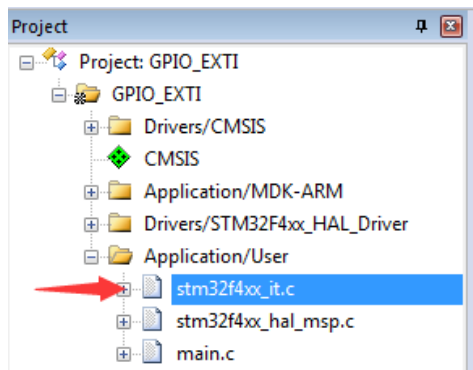
```

520  __weak void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
521  {
522      /* Prevent unused argument(s) compilation warning */
523      UNUSED(GPIO_Pin);
524      /* NOTE: This function should not be modified, when the callback is needed,
525         the HAL_GPIO_EXTI_Callback could be implemented in the user file */
526      /*
527  }

```

观察会发现，该函数使用__weak 修饰。里面的注释“NOTE:...”描述是，“该函数不应被修改，如果需要回调，则可以在用户文件中实现。”也就是说，要在用户文件中重写该函数。

方法 2：如果在 CubeMX 中已经配置使能了某个中断，则可在生成的工程中直接打开 User 文件组中的 stm32f4xx_it.c 文件。



找到相应的中断函数

```
165  * @brief This function handles EXTI line2 interrupt.
166  */
167  void EXTI2_IRQHandler(void)
168  {
169      /* USER CODE BEGIN EXTI2_IRQn 0 */
170
171      /* USER CODE END EXTI2_IRQn 0 */
172  → HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_2);
173      /* USER CODE BEGIN EXTI2_IRQn 1 */
174
175      /* USER CODE END EXTI2_IRQn 1 */
176  }
177
178  /**
179  * @brief This function handles EXTI line3 interrupt.
180  */
181  void EXTI3_IRQHandler(void)
182  {
183      /* USER CODE BEGIN EXTI3_IRQn 0 */
184
185      /* USER CODE END EXTI3_IRQn 0 */
186  → HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_3);
187      /* USER CODE BEGIN EXTI3_IRQn 1 */
188
189      /* USER CODE END EXTI3_IRQn 1 */
190  }
```

然后通过点击右键，选择“Go To Definition...”找到里面调用的函数的实现。然后就找到回调函数的名称了，如下图：

```
504  */
505  void HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin)
506  {
507      /* EXTI line interrupt detected */
508      if(__HAL_GPIO_EXTI_GET_IT(GPIO_Pin) != RESET)
509      {
510          __HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);
511          → HAL_GPIO_EXTI_Callback(GPIO_Pin);
512      }
513  }
```

官方例程请参考 stm32cubef4.zip 解压后

STM32Cube_FW_F4_V1.11.0\Projects\STM324xG_EVAL\Examples\GPIO\GPIO_EXTI 目录下的工程。

