



南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Embedded System and Microcomputer Principle

LAB3 External Interrupts

2021 Fall
wangq9@mail.sustech.edu.cn



CONTENTS

- 1 NVIC Function Description
- 2 EXTI Function Description
- 3 How to program
- 4 Practice



01

NVIC Function Description



1. NVIC -- 概述

- Cortex-M3内核支持256个中断，其中包含了16个内核中断和240个外部中断，具有256级的可编程中断设置
- STM32并没有使用Cortex-M3内核的全部中断，STM32有84个中断，包括16个内核中断和68个可屏蔽中断，具有16级可编程的中断优先级
- STM32F103系列上面，只有60个可屏蔽中断

1. NVIC -- 概述



0	7	可设置	WWDG	窗口定时器中断	0x0000_0040
1	8	可设置	PVD	连到EXTI的电源电压检测(PVD)中断	0x0000_0044
2	9	可设置	TAMPER	侵入检测中断	0x0000_0048
3	10	可设置	RTC	实时时钟(RTC)全局中断	0x0000_004C
4	11	可设置	FLASH	闪存全局中断	0x0000_0050
5	12	可设置	RCC	复位和时钟控制(RCC)中断	0x0000_0054
6	13	可设置	EXTI0	EXTI线0中断	0x0000_0058
7	14	可设置	EXTI1	EXTI线1中断	0x0000_005C
8	15	可设置	EXTI2	EXTI线2中断	0x0000_0060
9	16	可设置	EXTI3	EXTI线3中断	0x0000_0064
10	17	可设置	EXTI4	EXTI线4中断	0x0000_0068
11	18	可设置	DMA1通道1	DMA1通道1全局中断	0x0000_006C
12	19	可设置	DMA1通道2	DMA1通道2全局中断	0x0000_0070
13	20	可设置	DMA1通道3	DMA1通道3全局中断	0x0000_0074
14	21	可设置	DMA1通道4	DMA1通道4全局中断	0x0000_0078
15	22	可设置	DMA1通道5	DMA1通道5全局中断	0x0000_007C
16	23	可设置	DMA1通道6	DMA1通道6全局中断	0x0000_0080
17	24	可设置	DMA1通道7	DMA1通道7全局中断	0x0000_0084
18	25	可设置	ADC1_2	ADC1和ADC2的全局中断	0x0000_0088
19	26	可设置	USB_HP_CAN_TX	USB高优先级或CAN发送中断	0x0000_008C
20	27	可设置	USB_LP_CAN_RX0	USB低优先级或CAN接收0中断	0x0000_0090
21	28	可设置	CAN_RX1	CAN接收1中断	0x0000_0094
21	28	可设置	CAN_RX1	CAN接收1中断	0x0000_0094
22	29	可设置	CAN_SCE	CAN SCE中断	0x0000_0098
23	30	可设置	EXTI9_5	EXTI线[9:5]中断	0x0000_009C
24	31	可设置	TIM1_BRK	TIM1刹车中断	0x0000_00A0
25	32	可设置	TIM1_UP	TIM1更新中断	0x0000_00A4
26	33	可设置	TIM1_TRG_COM	TIM1触发和通信中断	0x0000_00A8
27	34	可设置	TIM1_CC	TIM1捕获比较中断	0x0000_00AC
28	35	可设置	TIM2	TIM2全局中断	0x0000_00B0
29	36	可设置	TIM3	TIM3全局中断	0x0000_00B4

30	37	可设置	TIM4	TIM4全局中断	0x0000_00B8
31	38	可设置	I2C1_EV	I ² C1事件中断	0x0000_00BC
32	39	可设置	I2C1_ER	I ² C1错误中断	0x0000_00C0
33	40	可设置	I2C2_EV	I ² C2事件中断	0x0000_00C4
34	41	可设置	I2C2_ER	I ² C2错误中断	0x0000_00C8
35	42	可设置	SPI1	SPI1全局中断	0x0000_00CC
36	43	可设置	SPI2	SPI2全局中断	0x0000_00D0
37	44	可设置	USART1	USART1全局中断	0x0000_00D4
38	45	可设置	USART2	USART2全局中断	0x0000_00D8
39	46	可设置	USART3	USART3全局中断	0x0000_00DC
40	47	可设置	EXTI15_10	EXTI线[15:10]中断	0x0000_00E0
41	48	可设置	RTCAlarm	连到EXTI的RTC闹钟中断	0x0000_00E4
42	49	可设置	USB唤醒	连到EXTI的从USB待机唤醒中断	0x0000_00E8
43	50	可设置	TIM8_BRK	TIM8刹车中断	0x0000_00EC
44	51	可设置	TIM8_UP	TIM8更新中断	0x0000_00F0
45	52	可设置	TIM8_TRG_COM	TIM8触发和通信中断	0x0000_00F4
46	53	可设置	TIM8_CC	TIM8捕获比较中断	0x0000_00F8
47	54	可设置	ADC3	ADC3全局中断	0x0000_00FC
48	55	可设置	FSMC	FSMC全局中断	0x0000_0100
49	56	可设置	SDIO	SDIO全局中断	0x0000_0104
50	57	可设置	TIM5	TIM5全局中断	0x0000_0108
51	58	可设置	SPI3	SPI3全局中断	0x0000_010C
52	59	可设置	UART4	UART4全局中断	0x0000_0110
53	60	可设置	UART5	UART5全局中断	0x0000_0114
54	61	可设置	TIM6	TIM6全局中断	0x0000_0118
55	62	可设置	TIM7	TIM7全局中断	0x0000_011C
56	63	可设置	DMA2通道1	DMA2通道1全局中断	0x0000_0120
57	64	可设置	DMA2通道2	DMA2通道2全局中断	0x0000_0124
58	65	可设置	DMA2通道3	DMA2通道3全局中断	0x0000_0128
59	66	可设置	DMA2通道4_5	DMA2通道4和DMA2通道5全局中断	0x0000_012C

1. NVIC -- 中断优先级分组

- 中断管理方法
 - (1) 对STM32中断进行分组, 组0~4
 - (2) 对每个中断设置一个抢占优先级和一个子优先级 (响应优先级) 的值, 数值越小优先级越高

组	AIRCR[10: 8]	IP bit[7: 4]分配情况	分配结果
0	111	0: 4	0位抢占优先级, 4位子优先级
1	110	1: 3	1位抢占优先级, 3位子优先级
2	101	2: 2	2位抢占优先级, 2位子优先级
3	100	3: 1	3位抢占优先级, 1位子优先级
4	011	4: 0	4位抢占优先级, 0位子优先级



1. NVIC -- 中断优先级分组

- 分组配置: Group0~4
- 一般情况下, 系统代码执行过程中, 只设置一次中断优先级分组, 比如分组2, 设置好分组之后一般不会再改变分组。随意改变分组会导致中断管理混乱, 程序出现意想不到的执行结果。



1. NVIC -- 中断优先级分组

- 抢占优先级和子优先级的区别
 - 高优先级的抢占优先级可以打断正在进行的低抢占优先级中断
 - 抢占优先级相同的中断，子优先级较高的中断不可以打断子优先级较低的中断
 - 抢占优先级相同的中断，当两个中断同时发生的情况下，哪个子优先级高，哪个先执行
 - 如果两个中断的抢占优先级和子优先级都是一样的话，则看哪个中断先发生就先执行



1. NVIC -- 中断优先级分组

- 举例
 - 假定设置中断优先级组为2。
 - 设置中断3的抢占优先级为2，子优先级为1。
 - 中断6的抢占优先级为3，子优先级为0。
 - 中断7的抢占优先级为2，子优先级为0。
 - 那么这3个中断的优先级顺序为：中断7 > 中断3 > 中断6



1. NVIC -- 中断优先级设置步骤

- 系统运行后先设置中断优先级分组。整个系统执行过程中，只设置一次中断分组。
- 针对每个中断，设置对应的抢占优先级和响应优先级。
- 如果需要挂起/解挂，查看中断当前激活状态，分别调用相关函数即可。



02

EXTI Function Description



2. EXTI -- 概述

- STM32的每个IO都可以作为外部中断输入。
- STM32的中断控制器支持19个外部中断/事件请求：
 - 线0~15：对应外部IO口的输入中断。
 - 线16：连接到PVD输出。
 - 线17：连接到RTC闹钟事件。
 - 线18：连接到USB唤醒事件。
- 每个外部中断线可以独立的配置触发方式（上升沿，下降沿或者双边沿触发），触发/屏蔽，专用的状态位。
- How to use 16 wires to control 51 GPIO ports?

2. EXTI -- 外部中断线映射

GPIOx.0映射到EXTI0

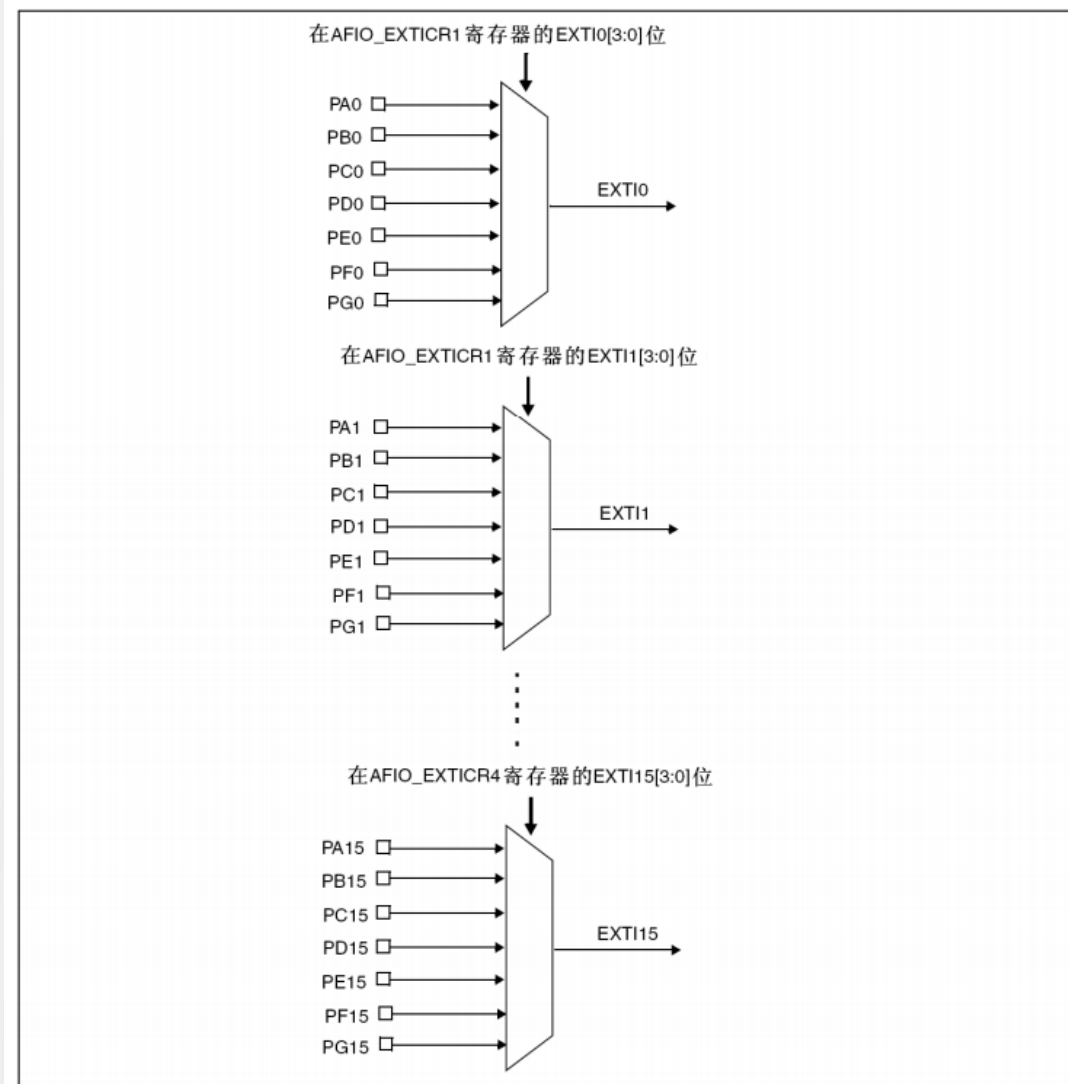
GPIOx.1映射到EXTI1

...

GPIOx.15映射到EXTI15

对于每个中断线，我们可以设置相应的触发方式（上升沿触发，下降沿触发，边沿触发）以及使能。

图20 外部中断通用I/O映像



2. EXTI -- 外部中断服务函数

- IO口外部中断在中断向量表中只分配了7个中断向量，也就是只能使用7个中断服务函数
- 外部中断线5~9分配一个中断向量，共用一个服务函数
- 外部中断线10~15分配一个中断向量，共用一个中断服务函数。

位置	优先级	优先级类型	名称	说明	地址	中断服务函数
6	13	可设置	EXTI0	EXTI线0中断	0x0000_0058	EXTI0_IRQHandler
7	14	可设置	EXTI1	EXTI线1中断	0x0000_005C	EXTI1_IRQHandler
8	15	可设置	EXTI2	EXTI线2中断	0x0000_0060	EXTI2_IRQHandler
9	16	可设置	EXTI3	EXTI线3中断	0x0000_0064	EXTI3_IRQHandler
10	17	可设置	EXTI4	EXTI线4中断	0x0000_0068	EXTI4_IRQHandler
23	30	可设置	EXTI9_5	EXTI线[9:5]中断	0x0000_009C	EXTI9_5_IRQHandler
40	47	可设置	EXTI15_10	EXTI线[15:10]中断	0x0000_00E0	EXTI15_10_IRQHandler



2. EXTI -- 外部中断配置步骤

- 初始化IO口为输入
- 开启IO口复用时钟
- 设置IO口与中断线的映射关系
- 初始化线上中断，设置触发条件等
- 配置中断分组（NVIC），并使能中断
- 编写中断服务函数
- 清除中断标志位



03

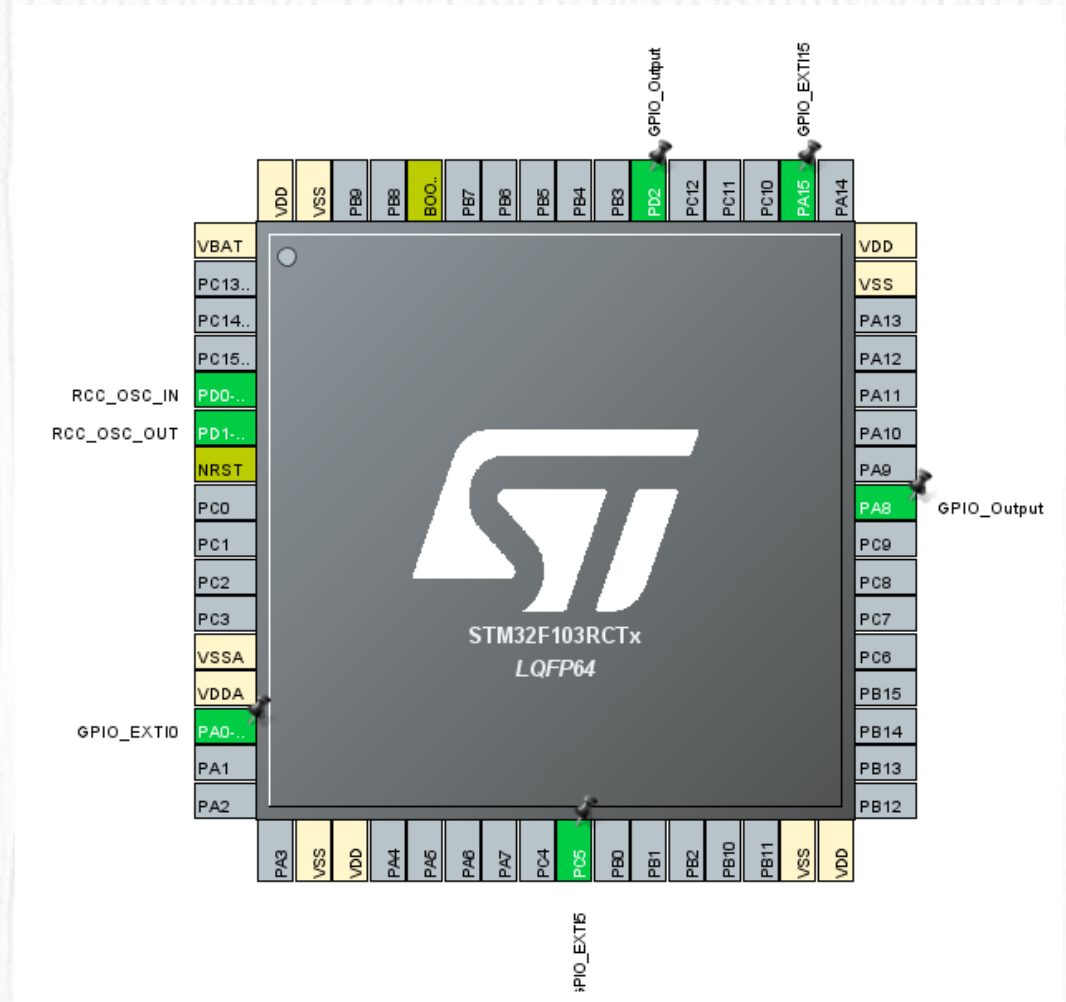
How to program

3. How to program

- Our goal
 - Use the three buttons, KEY0, KEY1 and WK_UP as EXTI input to control the LEDs, rather than check the value of these three GPIO pins in the main routine.

3. How to program

- GPIO configuration
 - Find the pins connected to KEY0, KEY1, WK_UP, LED0 and LED1, which is **PC5, PA15, PA0, PA8** and **PD2**
 - Config the pins connected to the buttons as **GPIO_EXTI**, and the pins connected to LEDs as **GPIO_Output**

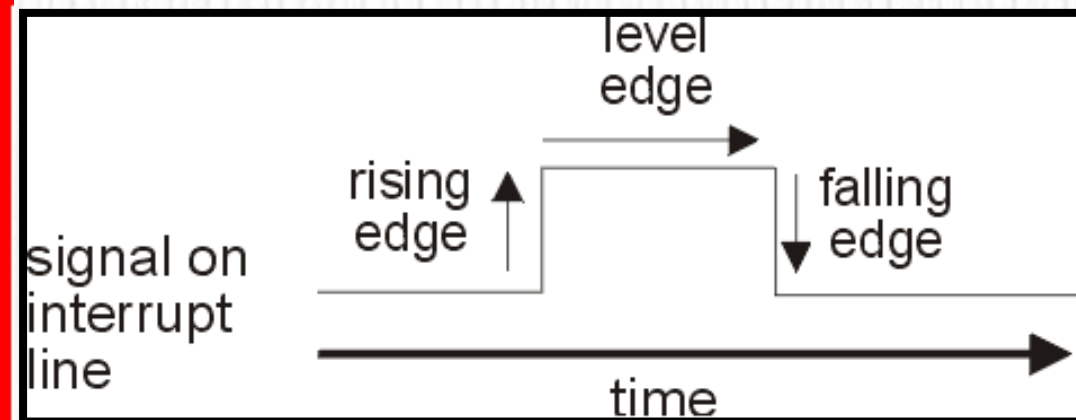
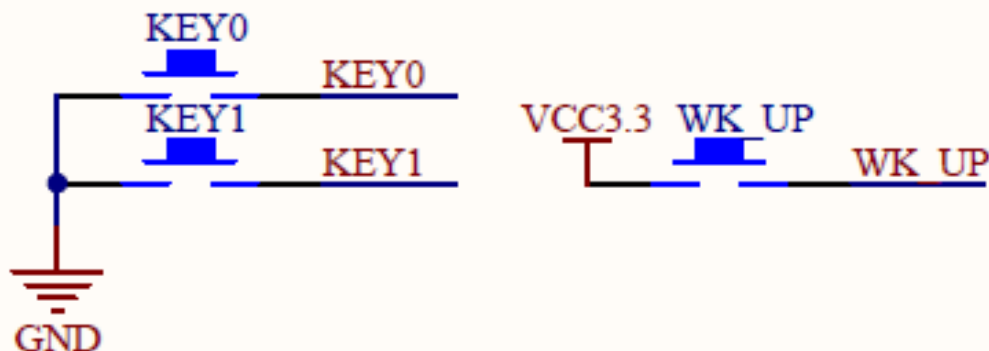


3. How to program

- Schematic

- we can configure the **GPIO Mode** as rising edge, falling edge or rising/falling edge to decide when to trigger interrupt
- The voltage should be 0v when KEY0 and KEY1 are pressed down, while the voltage should be 3.3v when WK_UP is pressed down. So the GPIO Mode of PA15 and PC5 should be falling edge, while the GPIO Mode of PA0 should be rising edge

KEY





3. How to program

- GPIO configuration
 - KEY_WK: EXTI with rising edge, GPIO pull-down
 - KEY0 and KEY1: EXTI with falling edge, GPIO pull-up

Pin Name	Signal on Pin	GPIO o...	GPIO mode	GPIO Pull-up/P...	Maximum out...	User Label
PA0-WKUP	n/a	n/a	External Interrupt Mode with Rising ...	Pull-down	n/a	KEY_WK
PA8	n/a	Low	Output Push Pull	No pull-up and ...	Low	LED0
PA15	n/a	n/a	External Interrupt Mode with Falling...	Pull-up	n/a	KEY1
PC5	n/a	n/a	External Interrupt Mode with Falling...	Pull-up	n/a	KEY0
PD2	n/a	Low	Output Push Pull	No pull-up and ...	Low	LED1



3. How to program

- Priority configuration
 - Two kinds of priority in STM32: preemption priority and sub priority.

NVIC Code generation			
Priority Group	2 bits for pre-emption priority 2 bits for subpriority	<input type="checkbox"/> Sort by Preemption Priority and Sub Priority	
Search	Search (Ctrl+F)	<input type="checkbox"/> Show only enabled interrupts	
NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0
Memory management fault	<input checked="" type="checkbox"/>	0	0
Prefetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0
Debug monitor	<input checked="" type="checkbox"/>	0	0
Pendable request for system service	<input checked="" type="checkbox"/>	0	0
Time base: System tick timer	<input checked="" type="checkbox"/>	0	0
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0	0
Flash global interrupt	<input type="checkbox"/>	0	0
RCC global interrupt	<input type="checkbox"/>	0	0
EXTI line0 interrupt	<input checked="" type="checkbox"/>	1	0
EXTI line[9:5] interrupts	<input checked="" type="checkbox"/>	1	1
EXTI line[15:10] interrupts	<input checked="" type="checkbox"/>	1	2



3. How to program

- EXTI interrupt function
 - Following is the code generated by STM32CubeIDE related to EXTI

```
/**
 * @brief This function handles EXTI line0 interrupt.
 */
void EXTI0_IRQHandler(void)
{
    /* USER CODE BEGIN EXTI0_IRQn 0 */

    /* USER CODE END EXTI0_IRQn 0 */
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0);
    /* USER CODE BEGIN EXTI0_IRQn 1 */

    /* USER CODE END EXTI0_IRQn 1 */
}
```

```
/**
 * @brief This function handles EXTI line[9:5] interrupts.
 */
void EXTI9_5_IRQHandler(void)
{
    /* USER CODE BEGIN EXTI9_5_IRQn 0 */

    /* USER CODE END EXTI9_5_IRQn 0 */
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_5);
    /* USER CODE BEGIN EXTI9_5_IRQn 1 */

    /* USER CODE END EXTI9_5_IRQn 1 */
}
```

```
/**
 * @brief This function handles EXTI line[15:10] interrupts.
 */
void EXTI15_10_IRQHandler(void)
{
    /* USER CODE BEGIN EXTI15_10_IRQn 0 */

    /* USER CODE END EXTI15_10_IRQn 0 */
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_15);
    /* USER CODE BEGIN EXTI15_10_IRQn 1 */

    /* USER CODE END EXTI15_10_IRQn 1 */
}
```

- It is clear that all the handler call the public EXTI handler
- HAL_GPIO_EXTI_IRQHandler();



3. How to program

- HAL_GPIO_EXTI_IRQHandler() function
 - __HAL_GPIO_EXTI_CLEAR_IT() clear the EXTI's line pending bits, otherwise, the EXTI handler will be executed all the time
 - HAL_GPIO_EXTI_Callback() is a weak function

```
void HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin)
{
    /* EXTI line interrupt detected */
    if (__HAL_GPIO_EXTI_GET_IT(GPIO_Pin) != 0x00u)
    {
        __HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);
        HAL_GPIO_EXTI_Callback(GPIO_Pin);
    }
}
```

```
__weak void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    /* Prevent unused argument(s) compilation warning */
    UNUSED(GPIO_Pin);
    /* NOTE: This function Should not be modified, when the callback is needed,
       the HAL_GPIO_EXTI_Callback could be implemented in the user file
    */
}
```



3. How to program

- HAL_GPIO_EXTI_Callback() re-implement
 - we should re-implement HAL_GPIO_EXTI_Callback() function in stm32f1xx_it.c

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    HAL_Delay(100);
    switch (GPIO_Pin) {
        case KEY0_Pin:
            if (HAL_GPIO_ReadPin(KEY0_GPIO_Port, KEY0_Pin) == GPIO_PIN_RESET) {
                HAL_GPIO_TogglePin(LED0_GPIO_Port, LED0_Pin);
            }
            break;
        case KEY1_Pin:
            if (HAL_GPIO_ReadPin(KEY1_GPIO_Port, KEY1_Pin) == GPIO_PIN_RESET) {
                HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);
            }
            break;
        case KEY_WK_Pin:
            if (HAL_GPIO_ReadPin(KEY_WK_GPIO_Port, KEY_WK_Pin) == GPIO_PIN_SET) {
                HAL_GPIO_TogglePin(LED0_GPIO_Port, LED0_Pin);
                HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);
            }
            break;
        default:
            break;
    }
}
```




04

Practice

4. Practice



- Run the demo on MiniSTM32 board
- Use EXTI to control the LED: Press KEY0 to blink LED0 three times, press KEY1 to blink LED1 three times.