Embedded System and Microcomputer Principle

LAB5 External Interrupts

2024 Fall

wangq9@mail.sustech.edu.cn

CONTENTS

1 NVIC Function Description

2 EXTI Function Description

3 How to program

4 Practice

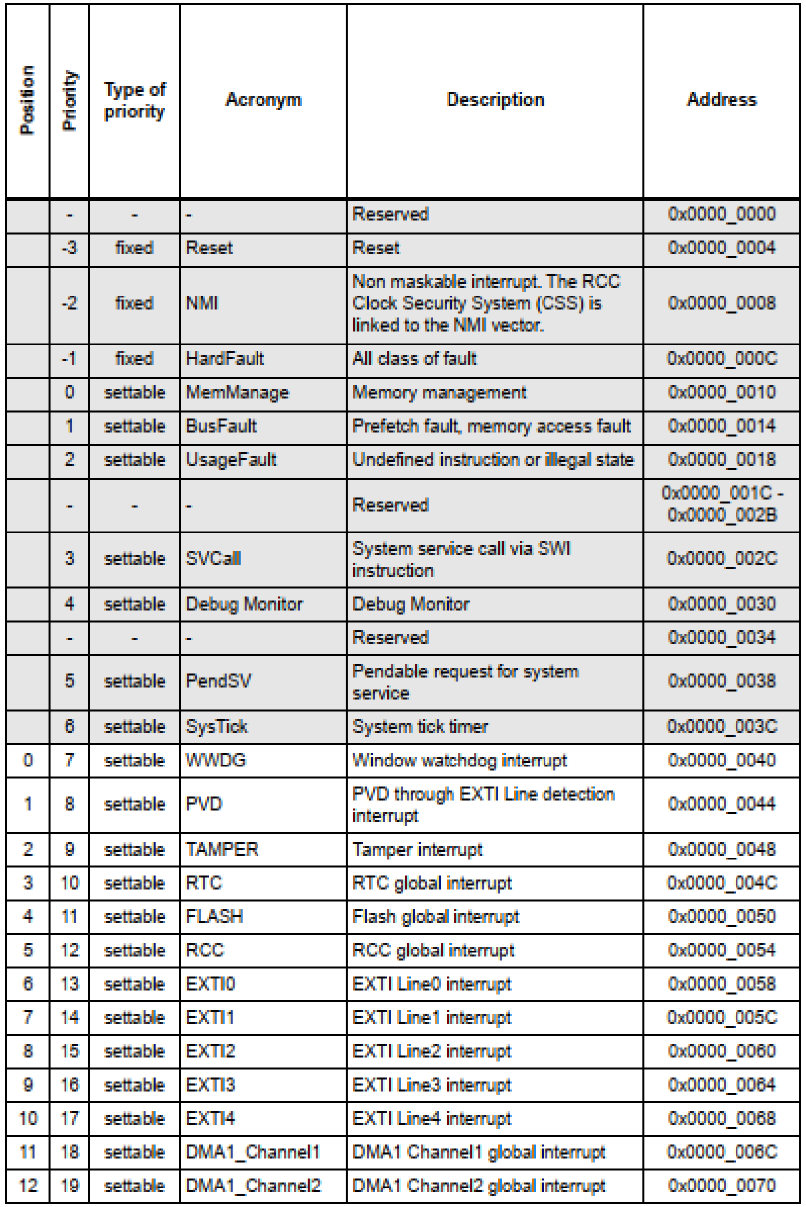
01

NVIC Function Description

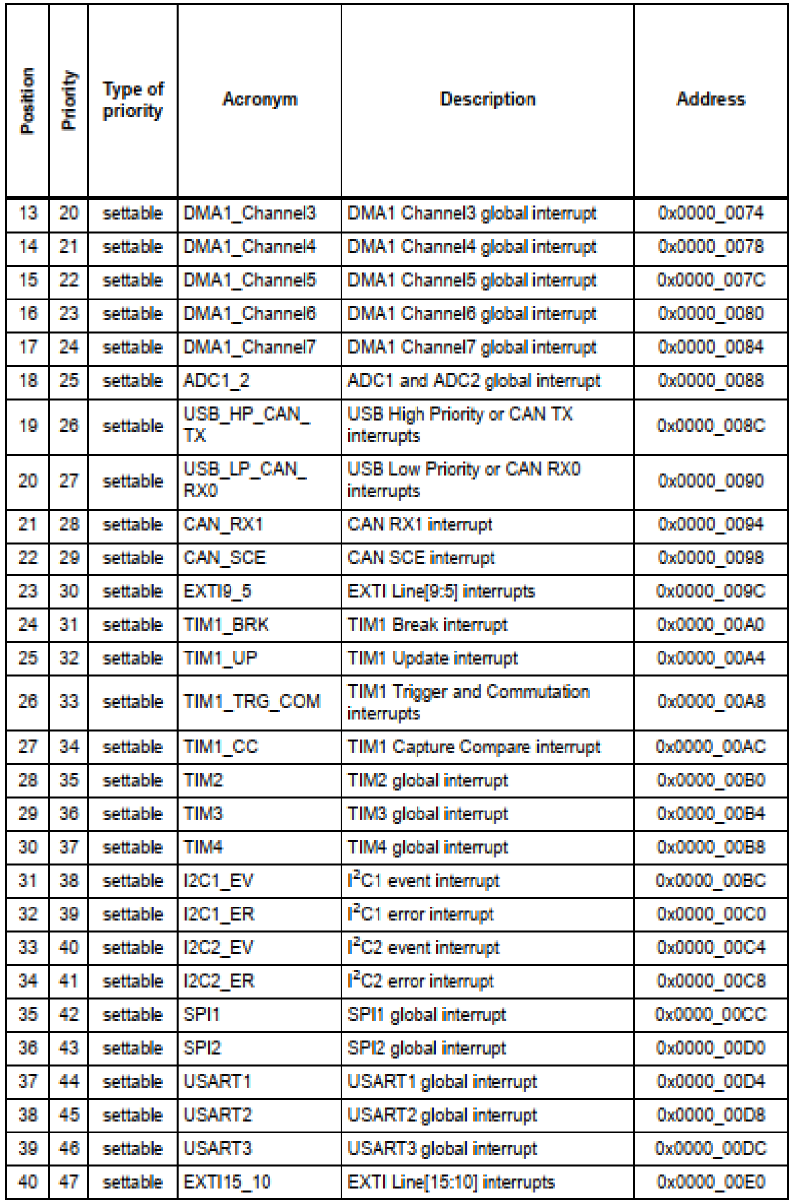
1. NVIC -- Sketch

- Nested Vectored Interrupt Controller
- Cortex-M3 core supports 256 programmable priority levels, including 16 Cortex-M3 interrupt lines and 240 external interrupt inputs
- STM32 doesn’t take use of all the Cortex-M3 interrupts, STM32 supports 84 interrupts, including 16 Cortex-M3 interrupt lines and 68 maskable interrupt inputs, and supports 16 programmable priority levels
- STM32F103 series support 60 maskable interrupt inputs

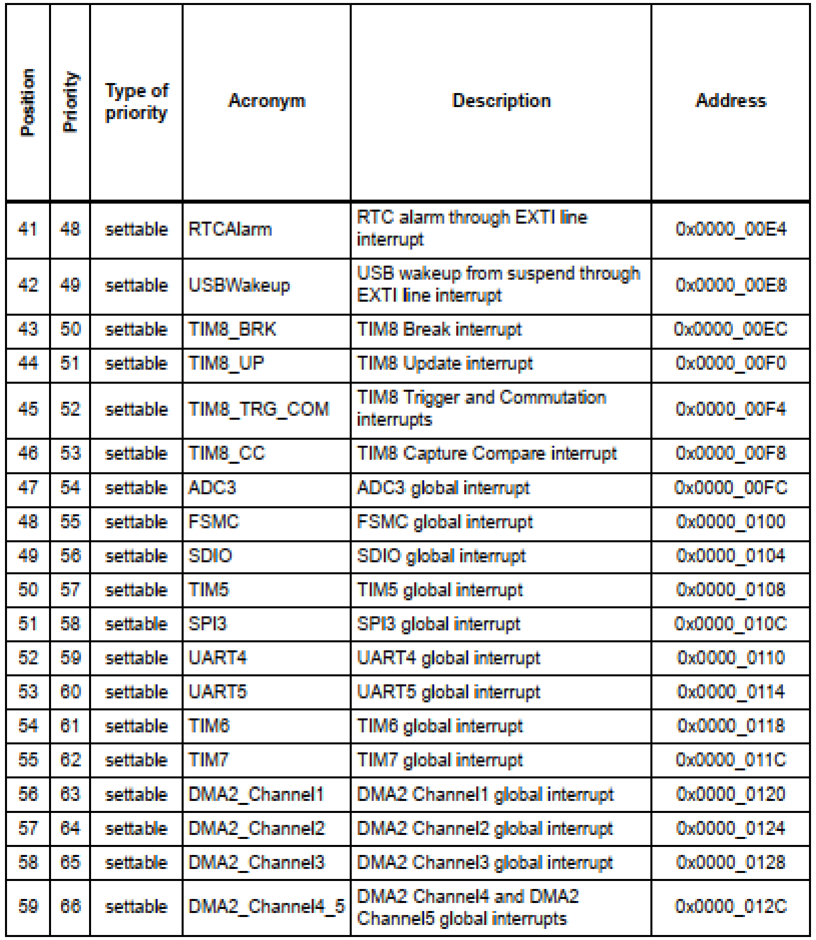
1. NVIC -- Sketch



Position	Priority	Type of priority	Acronym	Description	Address
	-	-	-	Reserved	0x0000_0000
	-3	fixed	Reset	Reset	0x0000_0004
	-2	fixed	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000_0008
	-1	fixed	HardFault	All class of fault	0x0000_000C
	0	settable	MemManage	Memory management	0x0000_0010
	1	settable	BusFault	Prefetch fault, memory access fault	0x0000_0014
	2	settable	UsageFault	Undefined instruction or illegal state	0x0000_0018
	-	-	-	Reserved	0x0000_001C - 0x0000_002B
	3	settable	SVCall	System service call via SWI instruction	0x0000_002C
	4	settable	Debug Monitor	Debug Monitor	0x0000_0030
	-	-	-	Reserved	0x0000_0034
	5	settable	PendSV	Pendable request for system service	0x0000_0038
	6	settable	SysTick	System tick timer	0x0000_003C
0	7	settable	WWDG	Window watchdog interrupt	0x0000_0040
1	8	settable	PVD	PVD through EXTI Line detection interrupt	0x0000_0044
2	9	settable	TAMPER	Tamper interrupt	0x0000_0048
3	10	settable	RTC	RTC global interrupt	0x0000_004C
4	11	settable	FLASH	Flash global interrupt	0x0000_0050
5	12	settable	RCC	RCC global interrupt	0x0000_0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000_0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000_005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000_0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000_0064
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000_0068
11	18	settable	DMA1_Channel1	DMA1 Channel1 global interrupt	0x0000_006C
12	19	settable	DMA1_Channel2	DMA1 Channel2 global interrupt	0x0000_0070



Position	Priority	Type of priority	Acronym	Description	Address
13	20	settable	DMA1_Channel3	DMA1 Channel3 global interrupt	0x0000_0074
14	21	settable	DMA1_Channel4	DMA1 Channel4 global interrupt	0x0000_0078
15	22	settable	DMA1_Channel5	DMA1 Channel5 global interrupt	0x0000_007C
16	23	settable	DMA1_Channel6	DMA1 Channel6 global interrupt	0x0000_0080
17	24	settable	DMA1_Channel7	DMA1 Channel7 global interrupt	0x0000_0084
18	25	settable	ADC1_2	ADC1 and ADC2 global interrupt	0x0000_0088
19	26	settable	USB_HP_CAN_ TX	USB High Priority or CAN TX interrupts	0x0000_008C
20	27	settable	USB_LP_CAN_ RX0	USB Low Priority or CAN RX0 interrupts	0x0000_0090
21	28	settable	CAN_RX1	CAN RX1 interrupt	0x0000_0094
22	29	settable	CAN_SCE	CAN SCE interrupt	0x0000_0098
23	30	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000_009C
24	31	settable	TIM1_BRK	TIM1 Break interrupt	0x0000_00A0
25	32	settable	TIM1_UP	TIM1 Update interrupt	0x0000_00A4
26	33	settable	TIM1_TRG_COM	TIM1 Trigger and Commutation interrupts	0x0000_00A8
27	34	settable	TIM1_CC	TIM1 Capture Compare interrupt	0x0000_00AC
28	35	settable	TIM2	TIM2 global interrupt	0x0000_00B0
29	36	settable	TIM3	TIM3 global interrupt	0x0000_00B4
30	37	settable	TIM4	TIM4 global interrupt	0x0000_00B8
31	38	settable	I2C1_EV	I^2C1 event interrupt	0x0000_00BC
32	39	settable	I2C1_ER	I^2C1 error interrupt	0x0000_00C0
33	40	settable	I2C2_EV	I^2C2 event interrupt	0x0000_00C4
34	41	settable	I2C2_ER	I^2C2 error interrupt	0x0000_00C8
35	42	settable	SPI1	SPI1 global interrupt	0x0000_00CC
36	43	settable	SPI2	SPI2 global interrupt	0x0000_00D0
37	44	settable	USART1	USART1 global interrupt	0x0000_00D4
38	45	settable	USART2	USART2 global interrupt	0x0000_00D8
39	46	settable	USART3	USART3 global interrupt	0x0000_00DC
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000_00E0



Position	Priority	Type of priority	Acronym	Description	Address
41	48	settable	RTCAlarm	RTC alarm through EXTI line interrupt	0x0000_00E4
42	49	settable	USBWakeup	USB wakeup from suspend through EXTI line interrupt	0x0000_00E8
43	50	settable	TIM8_BRK	TIM8 Break interrupt	0x0000_00EC
44	51	settable	TIM8_UP	TIM8 Update interrupt	0x0000_00F0
45	52	settable	TIM8_TRG_COM	TIM8 Trigger and Commutation interrupts	0x0000_00F4
46	53	settable	TIM8_CC	TIM8 Capture Compare interrupt	0x0000_00F8
47	54	settable	ADC3	ADC3 global interrupt	0x0000_00FC
48	55	settable	FSMC	FSMC global interrupt	0x0000_0100
49	56	settable	SDIO	SDIO global interrupt	0x0000_0104
50	57	settable	TIM5	TIM5 global interrupt	0x0000_0108
51	58	settable	SPI3	SPI3 global interrupt	0x0000_010C
52	59	settable	UART4	UART4 global interrupt	0x0000_0110
53	60	settable	UART5	UART5 global interrupt	0x0000_0114
54	61	settable	TIM6	TIM6 global interrupt	0x0000_0118
55	62	settable	TIM7	TIM7 global interrupt	0x0000_011C
56	63	settable	DMA2_Channel1	DMA2 Channel1 global interrupt	0x0000_0120
57	64	settable	DMA2_Channel2	DMA2 Channel2 global interrupt	0x0000_0124
58	65	settable	DMA2_Channel3	DMA2 Channel3 global interrupt	0x0000_0128
59	66	settable	DMA2_Channel4_5	DMA2 Channel4 and DMA2 Channel5 global interrupts	0x0000_012C

1. NVIC – Interrupt priority group

- How to manage so many interrupts?
  - (1) Group STM32 interrupts, group 0~4
  - (2) Set a preemption priority and a sub priority (response priority) for each interrupt. The smaller the value, the higher the priority

Group	AIRCR[10：8]	IP bit[7：4]	Description
0	111	0：4	0 bit for preemption interrupts，4 bits for sub interrupts
1	110	1：3	1 bit for preemption interrupts，3 bits for sub interrupts
2	101	2：2	2 bit for preemption interrupts，2 bits for sub interrupts
3	100	3：1	3 bit for preemption interrupts，1 bits for sub interrupts
4	011	4：0	4 bit for preemption interrupts，0 bits for sub interrupts

1. NVIC -- Interrupt priority

- Difference between preemption priority and sub priority
  - A high preemption priority can interrupt an ongoing low preemption priority interrupt
  - For interrupts with the same preemptive priority, interrupts with higher sub priority cannot interrupt interrupts with lower sub priority
  - For interrupts with the same preemptive priority, when two interrupts occur at the same time, which sub priority is higher and which is executed first
  - If the preemption priority and sub priority of two interrupts are the same, it depends on which interrupt occurs first

1. NVIC -- Interrupt priority

- Example
  - Suppose the interrupt priority group is set to 2.
  - Set the preemption priority of interrupt 3 to 2 and the sub priority to 1.
  - The preemptive priority of interrupt 6 is 3 and the sub priority is 0.
  - The preemption priority of interrupt 7 is 2 and the sub priority is 0.
  - Then the priority order of the three interrupts is: interrupt 7 > interrupt 3 > interrupt 6

1. NVIC -- Interrupt priority setting steps

- Set interrupt priority group after system operation. During the execution of the whole system, we only set the group one time.
- Set the corresponding preemption priority and sub priority for each interrupt.
- If you need to suspend / unhook, check the current activation status of the interrupt and call the relevant functions respectively.

02

EXTI Function Description

2. EXTI -- Sketch

- EXTernal Interrupt/event
- Each GPIO of STM32 can be used as an external interrupt input
- STM32 has19 edge detectors
  - EXTI line 0~15： input interrupt corresponding to external GPIO port
  - EXTI line 16 : connected to the PVD output
  - EXTI line 17 : connected to the RTC Alarm event
  - EXTI line 18 : connected to the USB Wakeup event
- How to use 16 wires to control 51 GPIO ports?

2. EXTI

-- External interrupt/event GPIO mapping

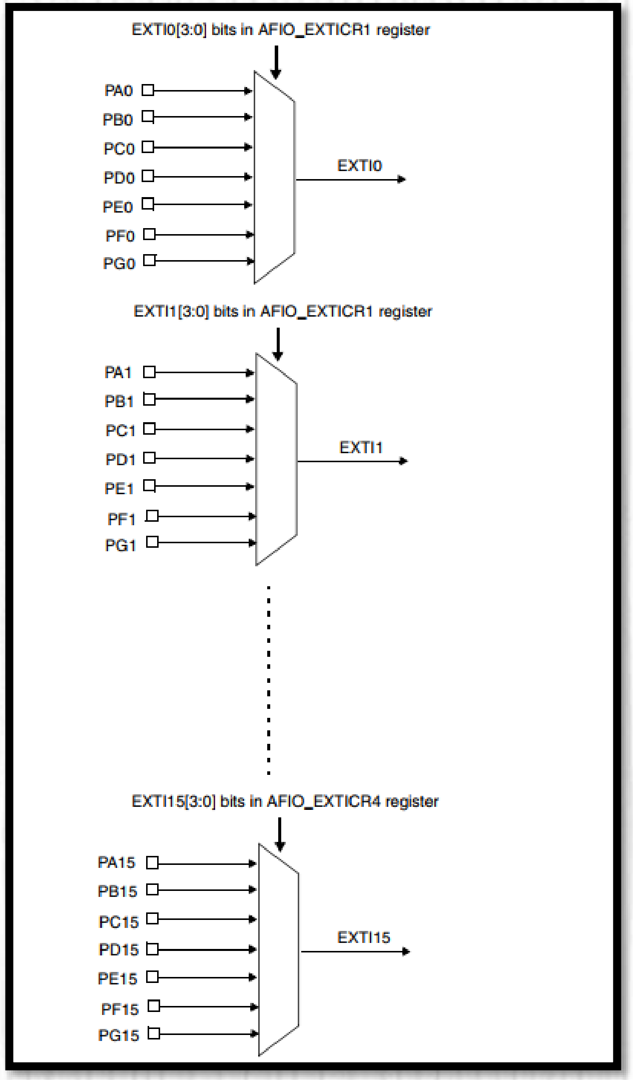
GPIOx.0 mapping to EXTI0

GPIOx.1 mapping to EXTI1

…

GPIOx.15 mapping to EXTI15

For each interrupt line, we can set the corresponding trigger mode (rising edge trigger, falling edge trigger, edge trigger) and enable or disable status.


EXTI0[3:0] bits in AFIO_EXTICR1 register
PA0
PB0
PC0
PD0
PE0
PF0
PG0
EXTI0
EXTI1[3:0] bits in AFIO_EXTICR1 register
PA1
PB1
PC1
PD1
PE1
PF1
PG1
EXTI1
EXTI15[3:0] bits in AFIO_EXTICR4 register
PA15
PB15
PC15
PD15
PE15
PF15
PG15
EXTI15

2. EXTI – Interrupt function

- Though there are 16 EXTI lines, only 7 interrupt vectors are allocated in the interrupt vector table
- EXTI 0-4 has its own interrupt function, while EXTI 5-9 share EXTI9_5_IRQHandler and EXTI 10~15 share EXTI15_10_IRQHandler

Position	Priority	Type	Acronmy	Description	Address	Interrupt function
6	13	Settable	EXTI0	EXTI line 0 interrupt	0x0000_0058	EXTI0_IRQHandler
7	14	Settable	EXTI1	EXTI line 1 interrupt	0x0000_005C	EXTI1_IRQHandler
8	15	Settable	EXTI2	EXTI line 2 interrupt	0x0000_0060	EXTI2_IRQHandler
9	16	Settable	EXTI3	EXTI line 3 interrupt	0x0000_0064	EXTI3_IRQHandler
10	17	Settable	EXTI4	EXTI line 4 interrupt	0x0000_0068	EXTI4_IRQHandler
23	30	Settable	EXTI9_5	EXTI line [9:5] interrupt	0x0000_009C	EXTI9_5_IRQHandler
40	47	Settable	EXTI15_10	EXTI line [15:10] interrupt	0x0000_00E0	EXTI15_10_IRQHandler

2. EXTI – Configuration steps

- Initialize GPIO port as input
- Enable GPIO port multiplexing clock
- Set the mapping relationship between GPIO port and interrupt line
- Initialize online interrupt, set trigger conditions, etc
- Configure NVIC and enable interrupts
- Write interrupt service function
- Clear interrupt flag bit

03

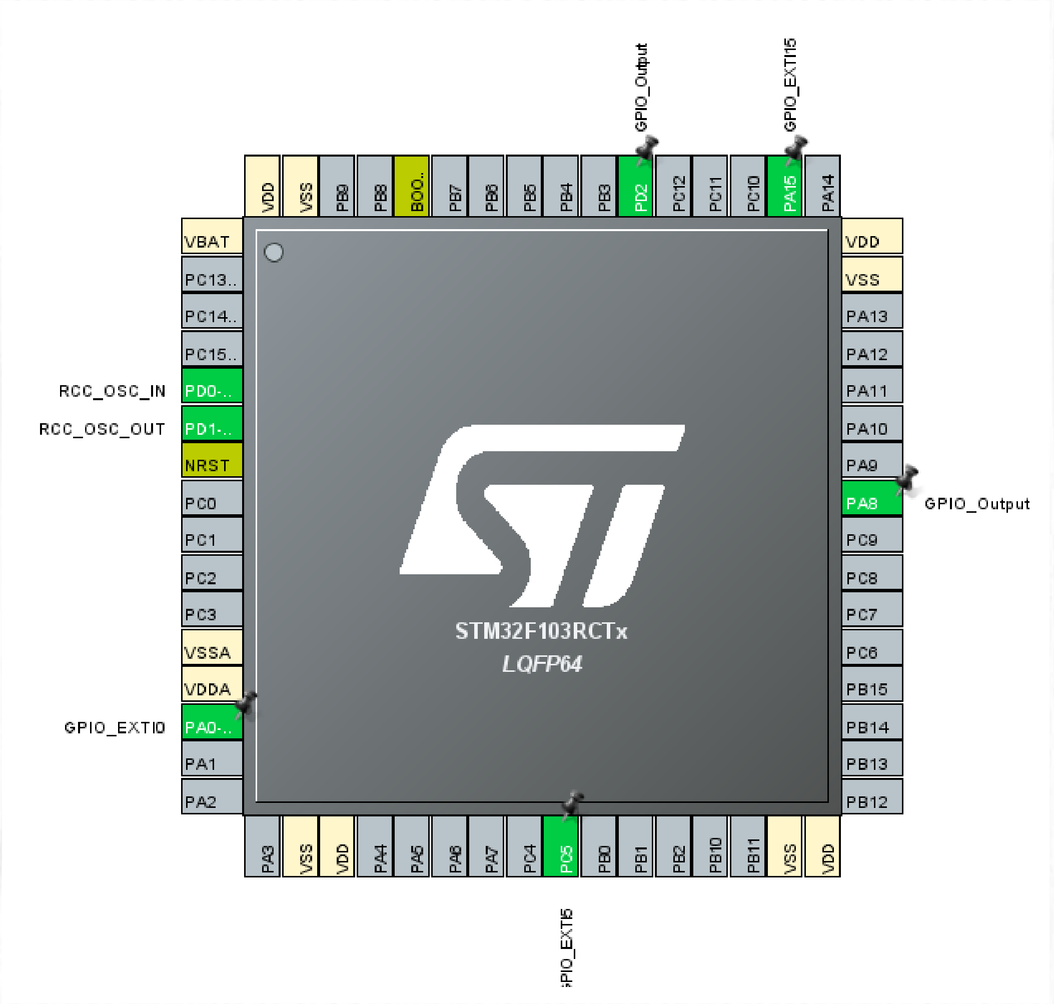
How to program

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.

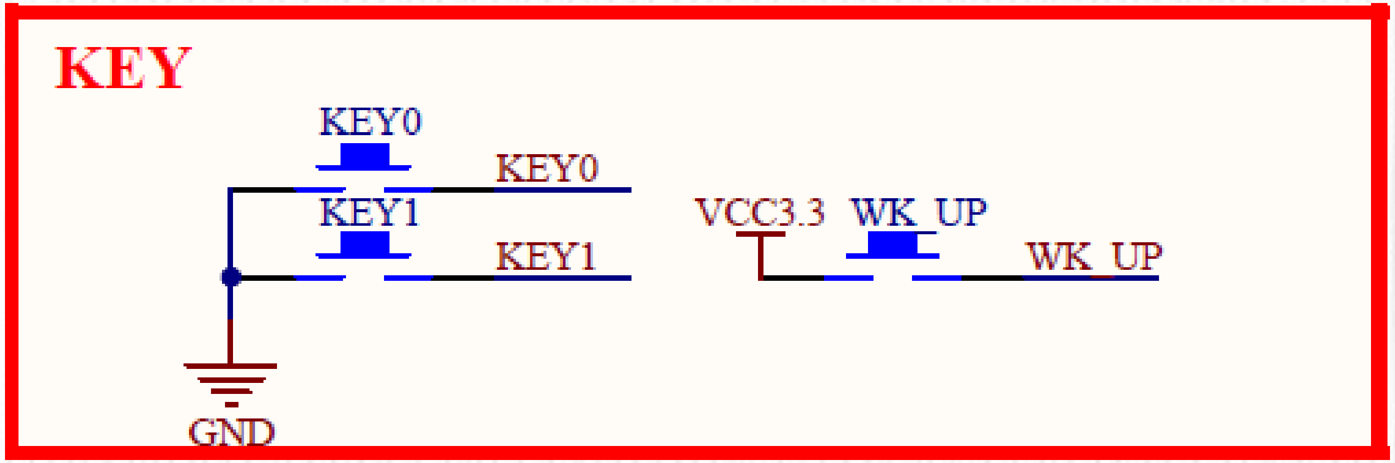
3. How to program

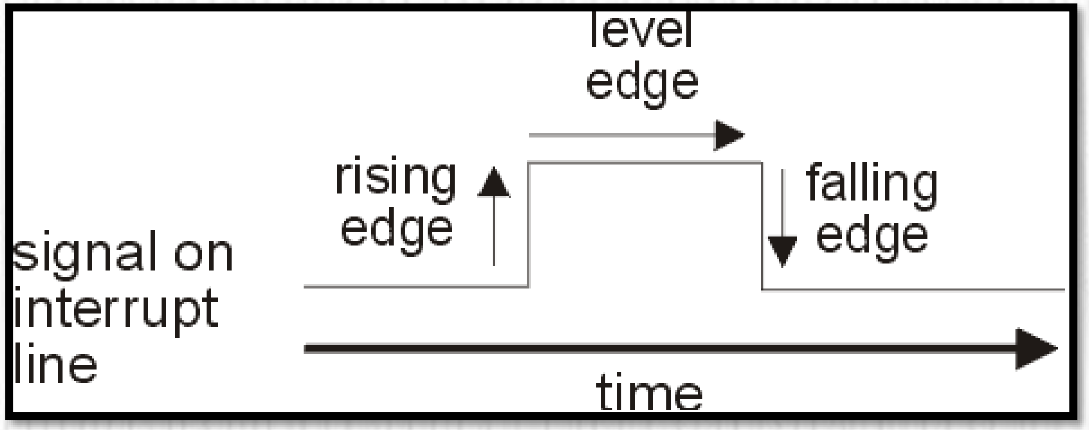
- GPIO configuration
  - Find the pins connected to KEY0, KEY1, WK_UP, LED0 and LED1, which is **PC5, PA15, PA0, PA8** and **PD2**
  - Configure the pins connected to the buttons as **GPIO_EXTI**, and the pins connected to LEDs as **GPIO_Output**



3. How to program

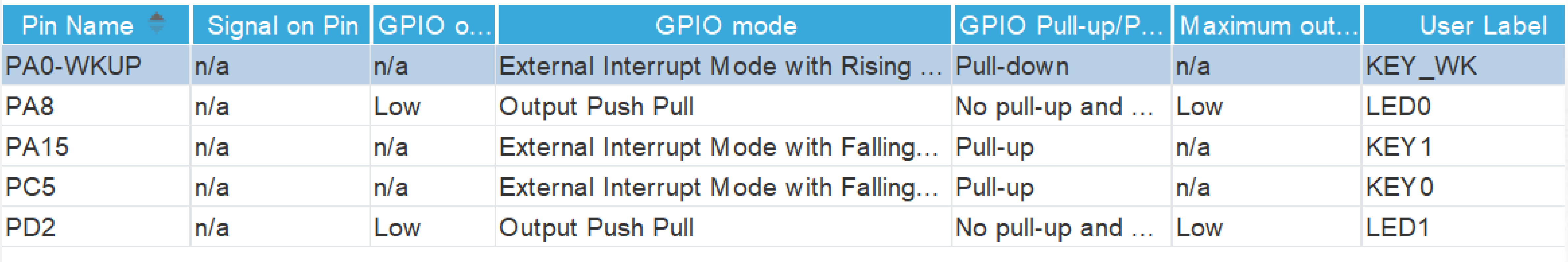
- Schematic
  - we can configure the **GPIO Mode** as rising edge, falling edge or rising/falling edge to decide when to trigger interrupt.
  - The voltage should be 0v when KEY0 and KEY1 are pressed down, while the voltage should be 3.3v when WK_UP is pressed down. So the GPIO Mode of PA15 and PC5 should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





3. How to program

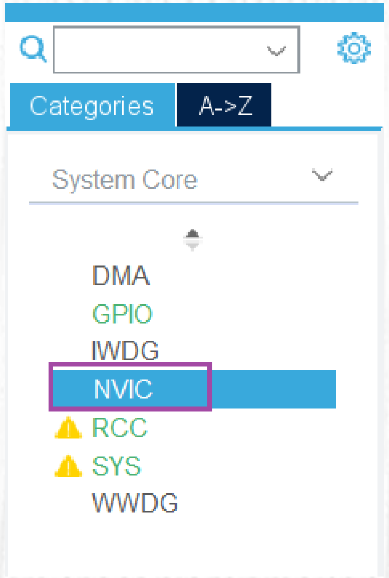
- GPIO configuration
  - KEY_WK: EXTI with rising edge, GPIO pull-down
  - KEY0 and KEY1: EXTI with falling edge, GPIO pull-up

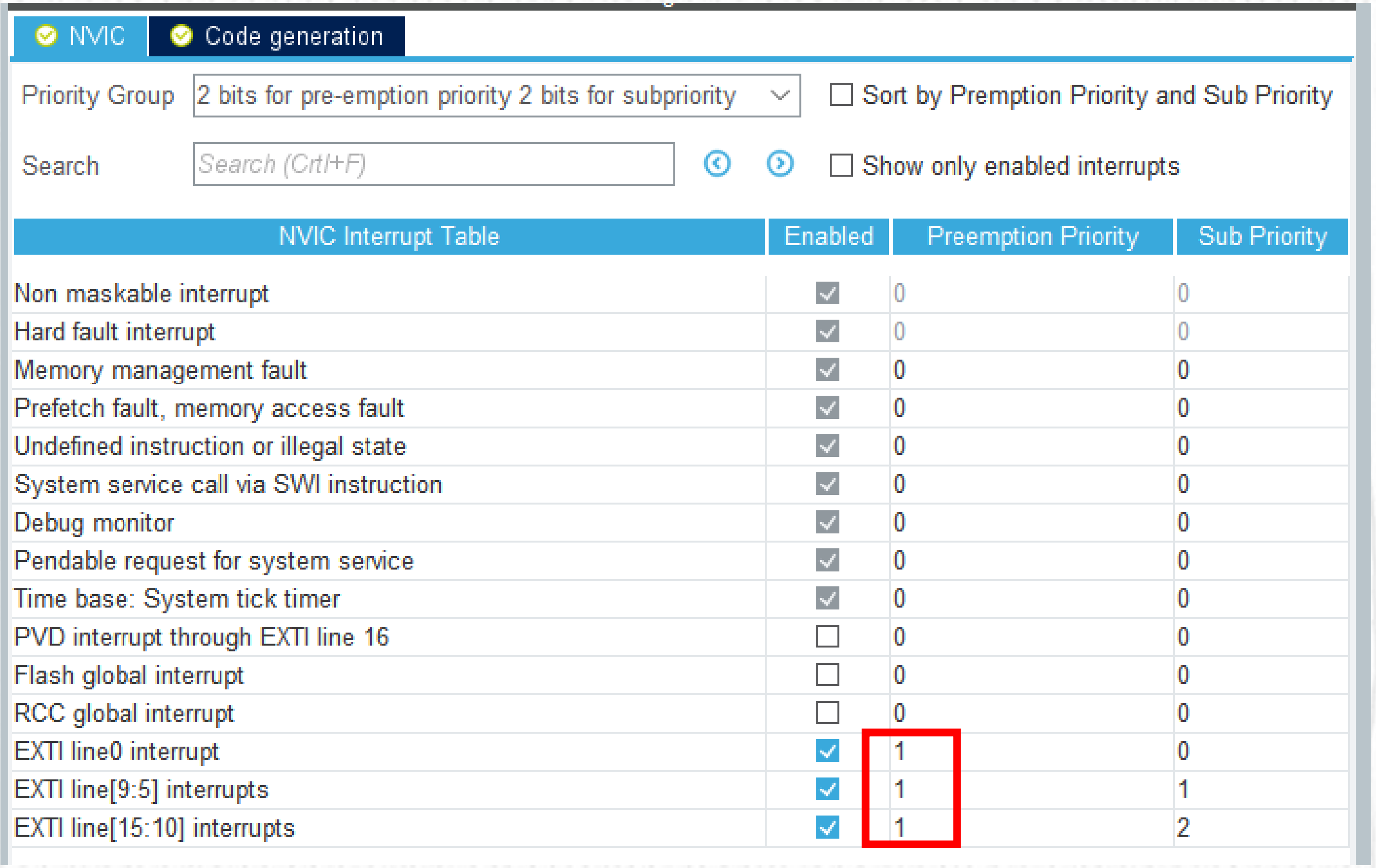


Pin Name	Signal on Pin	GPIO o...	GPIO mode	GPIO Pull-up/P...	Maximum out...	User Label
PA0-WKUP	n/a	n/a	External Interrupt Mode with Rising ...	Pull-down	n/a	KEY_WK
PA8	n/a	Low	Output Push Pull	No pull-up and ...	Low	LED0
PA15	n/a	n/a	External Interrupt Mode with Falling...	Pull-up	n/a	KEY1
PC5	n/a	n/a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





Priority Group: 2 bits for pre-emption priority 2 bits for subpriority

☐ Sort by Premption Priority and Sub Priority

Search: Search (Crtl+F)

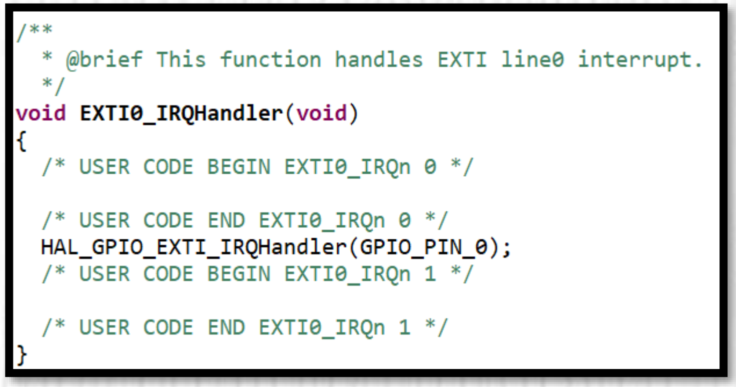
☐ Show only enabled interrupts

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	☑	0	0
Hard fault interrupt	☑	0	0
Memory management fault	☑	0	0
Prefetch fault, memory access fault	☑	0	0
Undefined instruction or illegal state	☑	0	0
System service call via SWI instruction	☑	0	0
Debug monitor	☑	0	0
Pendable request for system service	☑	0	0
Time base: System tick timer	☑	0	0
PVD interrupt through EXTI line 16	☐	0	0
Flash global interrupt	☐	0	0
RCC global interrupt	☐	0	0
EXTI line0 interrupt	☑	1	0
EXTI line[9:5] interrupts	☑	1	1
EXTI line[15:10] interrupts	☑	1	2

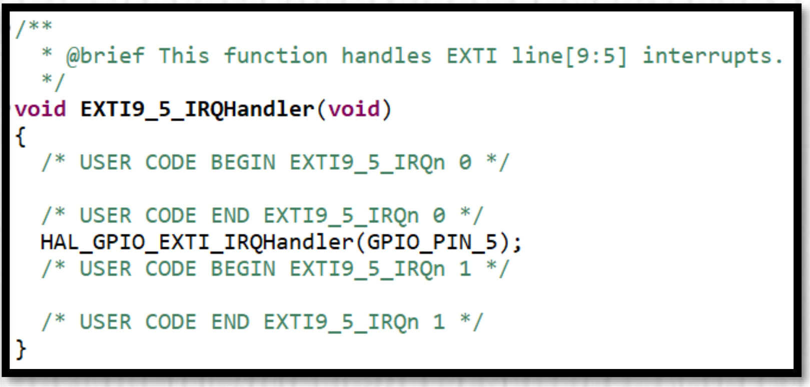
Do not use 0 as Preemption Priority

3. How to program

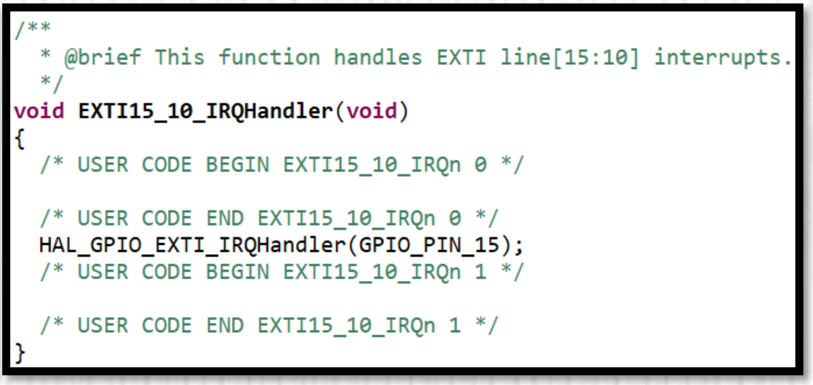
- EXTI interrupt function
  - Following is the code generated by STM32CubeIDE related to EXTI



```
/**
  * @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 */
}
```

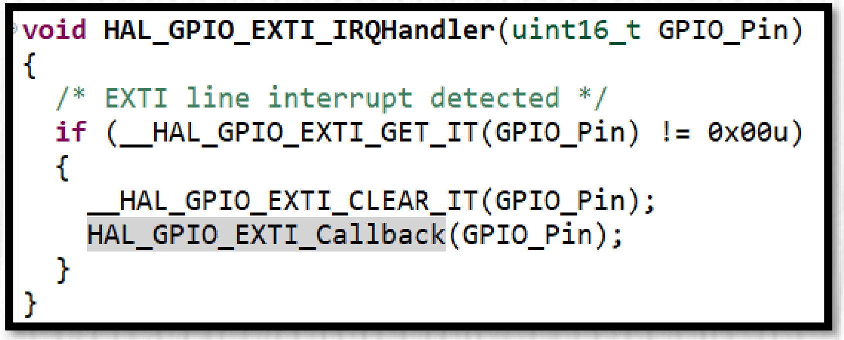


```
/**
  * @brief This function handles EXTI line[15:10] interrupts.
  */
void EXTI15_10_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI15_10_IRQn 0 */

  /* 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 */
}
```

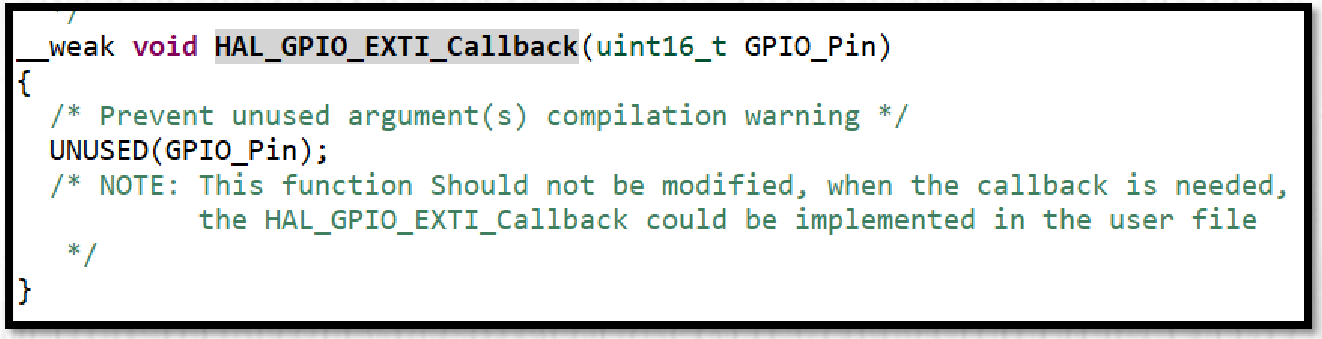
  - It is clear that all the handler call the public EXTI handler **HAL_GPIO_EXTI_IRQHandler()**;

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



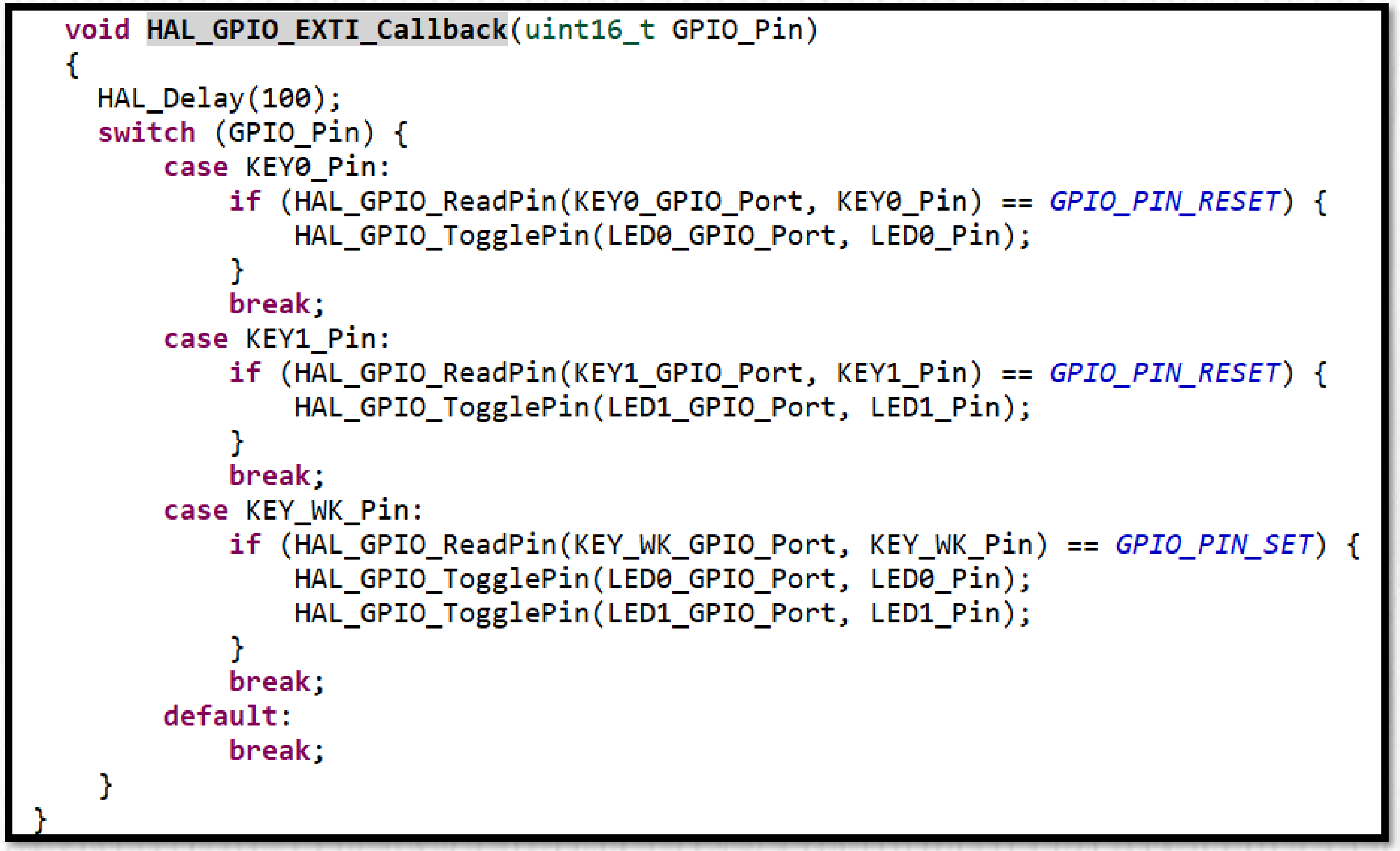
```
void HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin)
{
  /* EXTI line interrupt detected */
  if (__HAL_GPIO_EXTI_GET_IT(GPIO_Pin) != 0x00u)
  {
    __HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);
    HAL_GPIO_EXTI_Callback(GPIO_Pin);
  }
}
```



```
__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 show different messages on LCD screen
  - When KEY_WK is pressed, show the string “KEY_WAKEUP is pressed” on LCD screen.
  - When KEY0 is pressed, show the string “KEY0 is pressed” on LCD screen.
  - When KEY1 is pressed, show the string “KEY1 is pressed” on LCD screen.
  - Note: clear the previous string before displaying the new one.