Micro-Controller Experiment

Week9

Teacher: 廖裕評 Yu-Ping Liao

TA: 陳大荃 Da-chuan Chen, 陳恩妮 En-ni Chen

Class Rules

1. No drink besides water.
2. Bring a laptop and breadboard if needed.
3. Ask us TAs to sign and borrow development boards. Do not sign or ask others to sign for you without TAs' permission.
4. Arriving 10 minutes after the bell rings will be regarded as absent.
5. If you damage any borrowed equipment, you have to pay for it.

Homework Rules

1. Includes: A. Class content, B. Class exercise, C. Homework (screenshot or video)
2. Editing software: MS PowerPoint
3. File format: PDF
4. Filename: "date_group_studentID_name.pdf", like "0916_第1組_11028XXX_陳OO.pdf"
5. The homework deadline is 23:59 of the day before the next class. If you are late, then your grade will be deducted.

Contact

If you encounter any problems with this class, please get in touch with us with the following

E-mails:

1. Teacher, Prof. Yu-Ping Liao 廖裕評： lyp@cycu.org.tw
2. TA, Da-chuan Chen 陳大荃： dachuan516@gmail.com
3. TA, En-ni Chen 陳恩妮： anna7125867@gmail.com

Or visit 篤信 Lab353 for further questions.

Outline of the Week

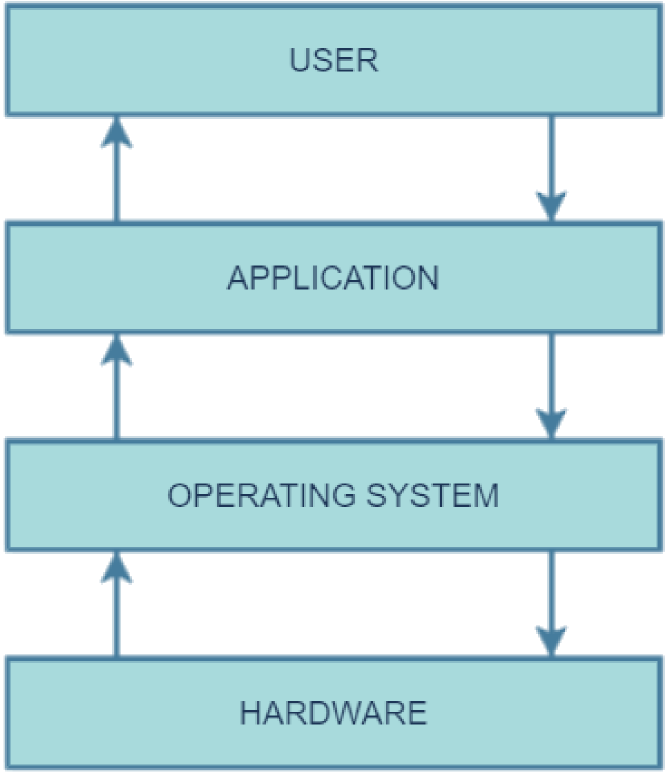
1. RTOS
2. NVIC introduction
3. EXTI introduction
4. EXTI Project.
5. Homework 9-1.
6. Homework 9-2.
7. Homework 9-3 Bonus.

RTOS

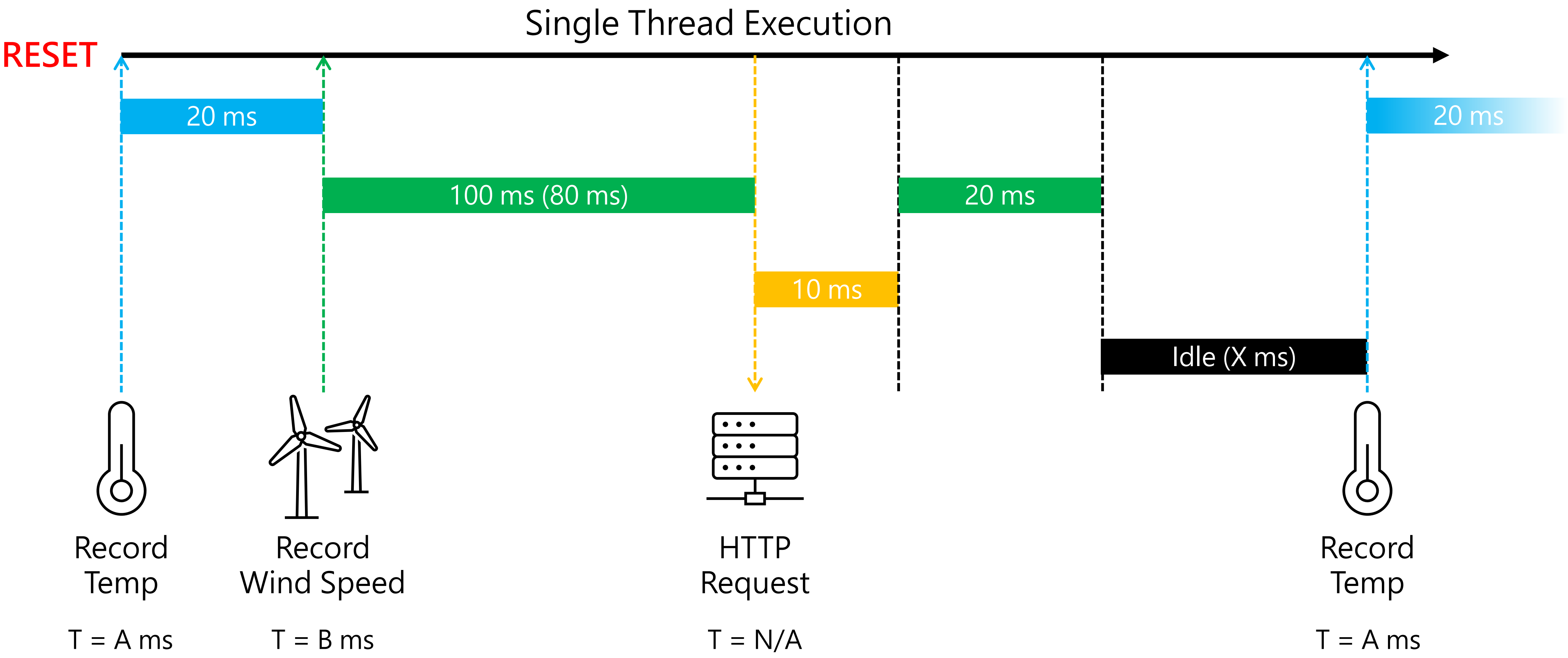
Operating System

An operating system (OS) is system software that manages computer hardware and software resources, and provides common services for computer programs.

Source: Wikipedia.org/wiki/Operating_system



Scheduling – Weather Station



Types of OS

CONTIKI
NEXT GENERATION

freeRTOS

RTOS
Real Timer Operation System

RTOS real-life example

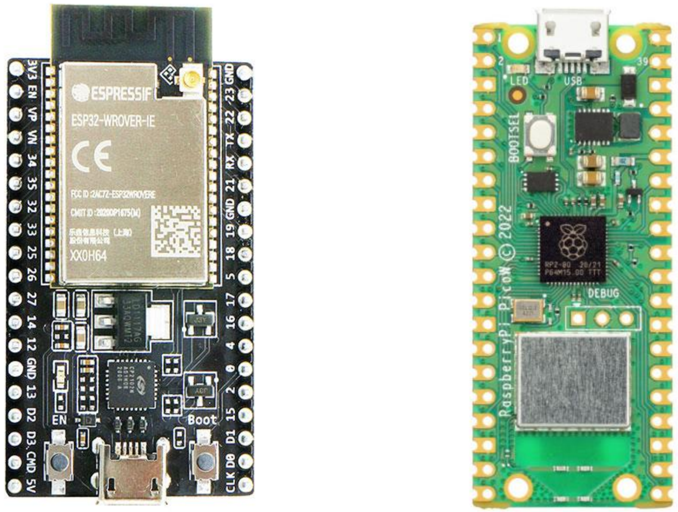
Windows

Linux

Mac OS

GPOS
General Purpose Operation System

Hardware OSs Operate On



RTOS

Real Timer Operation System

RTOS vs. GPOS

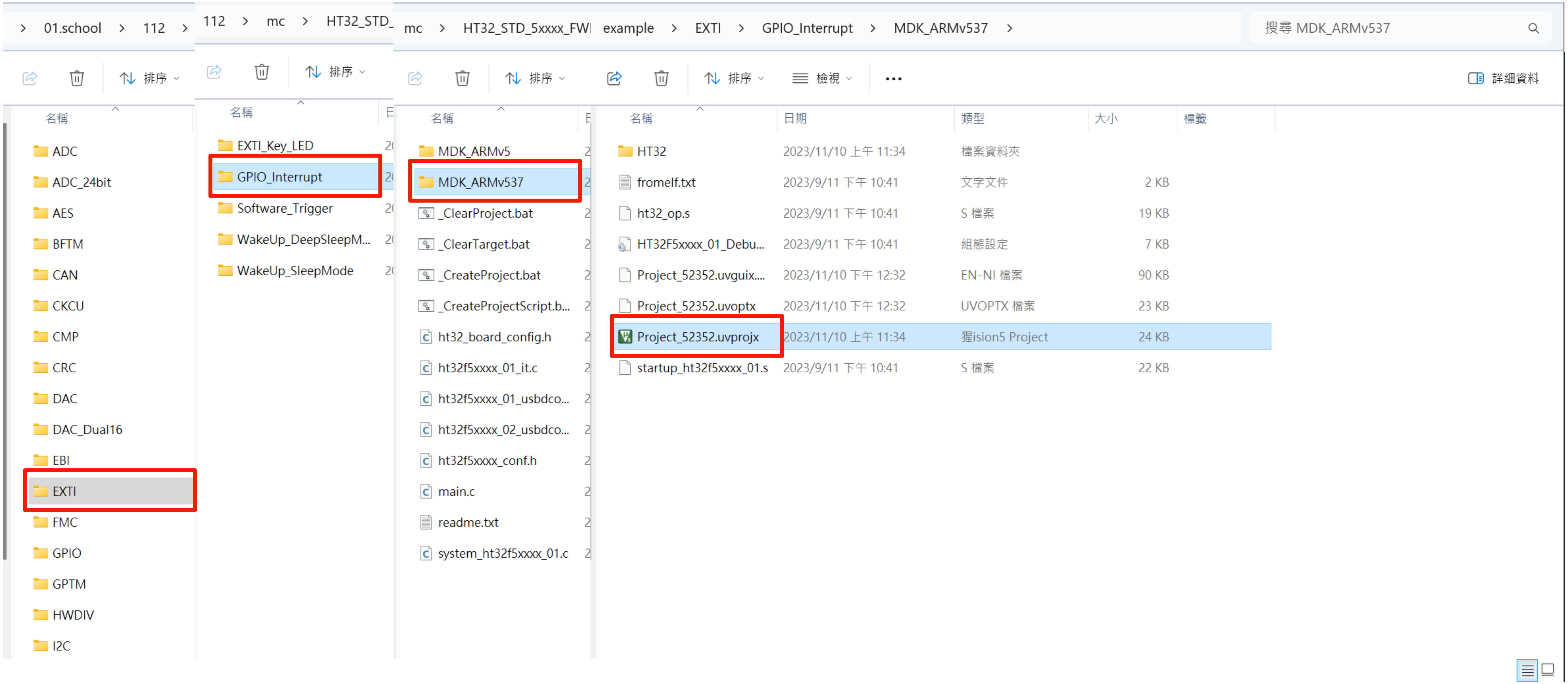
GPOS

General Purpose Operation System

NVIC Introduction

Open the example project

Go to “~/HT32_STD_5xxxx_FWLib_V1.5.1_7084/example/EXTI/GPIO_Interrupt”.



名稱

- ADC
- ADC_24bit
- AES
- BFTM
- CAN
- CKCU
- CMP
- CRC
- DAC
- DAC_Dual16
- EBI
- EXTI
- FMC
- GPIO
- GPTM
- HWDIV
- I2C

名稱

- EXTI_Key_LED
- GPIO_Interrupt
- Software_Trigger
- WakeUp_DeepSleepM...
- WakeUp_SleepMode

名稱

- MDK_ARMv5
- MDK_ARMv537
- _ClearProject.bat
- _ClearTarget.bat
- _CreateProject.bat
- _CreateProjectScript.b...
- ht32_board_config.h
- ht32f5xxxx_01_it.c
- ht32f5xxxx_01_usbdco...
- ht32f5xxxx_02_usbdco...
- ht32f5xxxx_conf.h
- main.c
- readme.txt
- system_ht32f5xxxx_01.c

名稱	日期	類型	大小	標籤
HT32	2023/11/10 上午 11:34	檔案資料夾		
fromelf.txt	2023/9/11 下午 10:41	文字文件	2 KB	
ht32_op.s	2023/9/11 下午 10:41	S 檔案	19 KB	
HT32F5xxxx_01_Debu...	2023/9/11 下午 10:41	組態設定	7 KB	
Project_52352.uvguix....	2023/11/10 下午 12:32	EN-NI 檔案	90 KB	
Project_52352.uvoptx	2023/11/10 下午 12:32	UVOPTX 檔案	23 KB	
Project_52352.uvprojx	2023/11/10 上午 11:34	猩ision5 Project	24 KB	
startup_ht32f5xxxx_01.s	2023/9/11 下午 10:41	S 檔案	22 KB	

What is NVIC?

- NVIC(Nested vectored interrupt controller 內嵌向量中斷控制器) is a component of the processor responsible for handling exception and interrupt-related procedures.
- First, let's briefly discuss the purpose of interrupts. Let's assume you are listening to music on your phone, and suddenly a call comes in. When you answer the call, the phone automatically pauses the music and waits for the call to end before resuming playback. Listening to music represents the original task, the incoming call is the interrupt, and answering the call is the Interrupt Service Routine (ISR). To be more specific, the incoming call is referred to as the interrupt-generating event, and answering the call is the ISR.

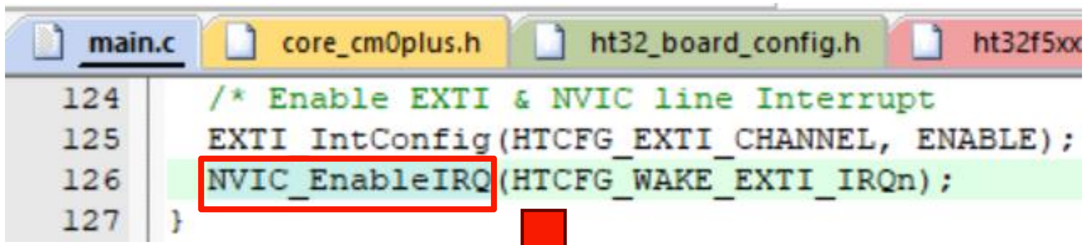
The processing flow of interrupts

1. Pause the currently executing program.
2. Save the execution status of this program.
3. The CPU searches the interrupt vector table based on the interrupt request.
4. Obtain the starting address of the ISR (Interrupt Service Routine).
5. Execute the ISR.
6. After the ISR execution is complete, return to the execution of the original program before the interrupt.

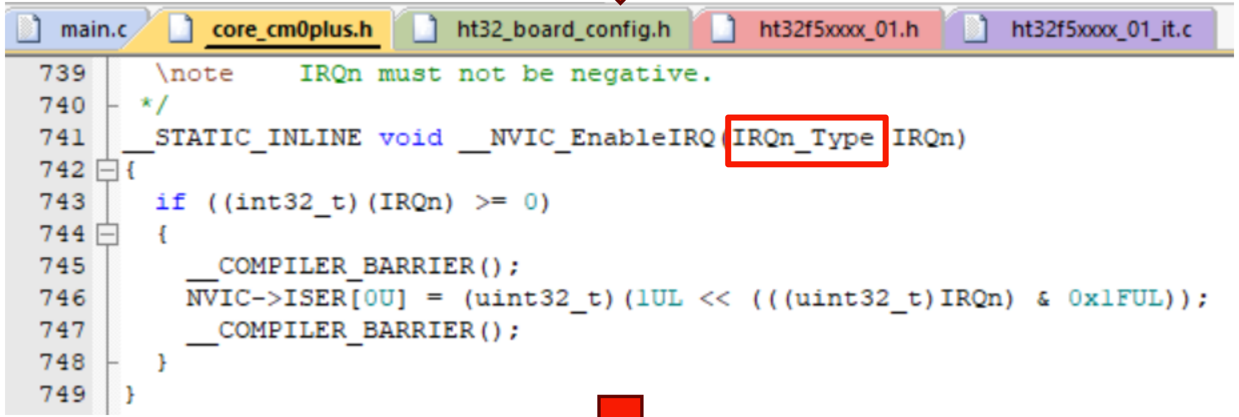
NVIC Key Points:

1. Interrupt Request (IRQ)
2. Interrupt Function Name
3. Contents of Interrupt Service Routine (ISR)
4. Interrupt Priority

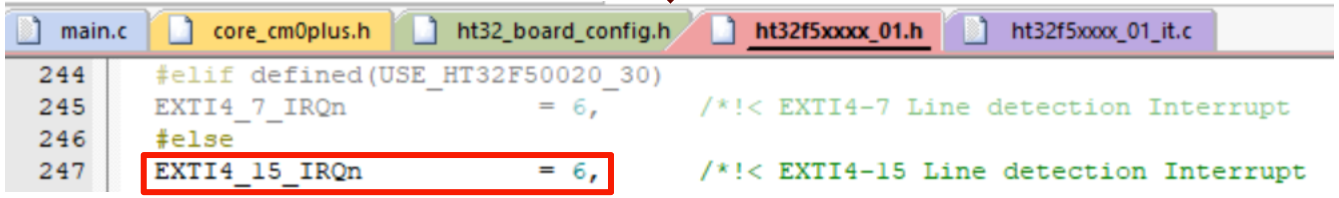
Interrupt Request (IRQ)



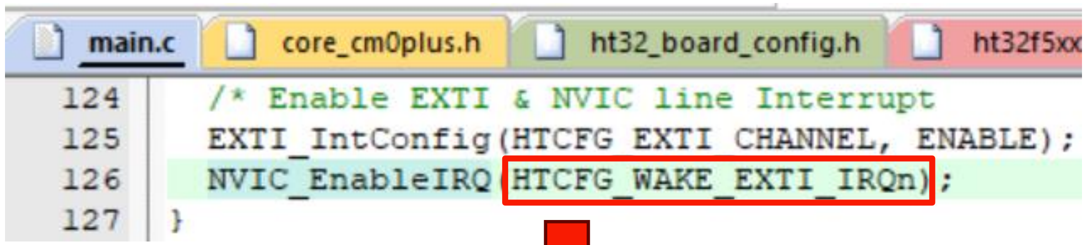
```
/* Enable EXTI & NVIC line Interrupt
EXTI_IntConfig(HTCFG_EXTI_CHANNEL, ENABLE);
NVIC_EnableIRQ(HTCFG_WAKE_EXTI_IRQn);
}
```



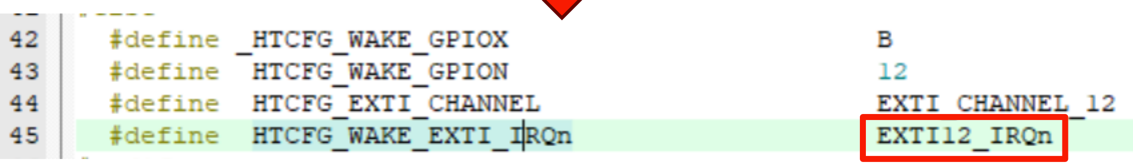
```
  \note    IRQn must not be negative.
 */
__STATIC_INLINE void __NVIC_EnableIRQ(IRQn_Type IRQn)
{
  if ((int32_t)(IRQn) >= 0)
  {
    __COMPILER_BARRIER();
    NVIC->ISER[0U] = (uint32_t)(1UL << (((uint32_t)IRQn) & 0x1FUL));
    __COMPILER_BARRIER();
  }
}
```



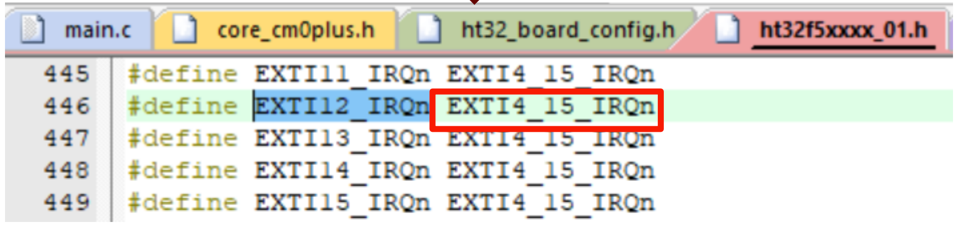
```
#elif defined(USE_HT32F50020_30)
EXTI4_7_IRQn                = 6,      /*!< EXTI4-7 Line detection Interrupt
#else
EXTI4_15_IRQn               = 6,      /*!< EXTI4-15 Line detection Interrupt
```



```
/* Enable EXTI & NVIC line Interrupt
EXTI_IntConfig(HTCFG_EXTI_CHANNEL, ENABLE);
NVIC_EnableIRQ(HTCFG_WAKE_EXTI_IRQn);
}
```



```
#define _HTCFG_WAKE_GPIOX                      B
#define  HTCFG_WAKE_GPION                      12
#define  HTCFG_EXTI_CHANNEL                    EXTI_CHANNEL_12
#define  HTCFG_WAKE_EXTI_IRQn                  EXTI12_IRQn
```



```
#define EXTI11_IRQn EXTI4_15_IRQn
#define EXTI12_IRQn EXTI4_15_IRQn
#define EXTI13_IRQn EXTI4_15_IRQn
#define EXTI14_IRQn EXTI4_15_IRQn
#define EXTI15_IRQn EXTI4_15_IRQn
```

- ➢ UserManual p.182

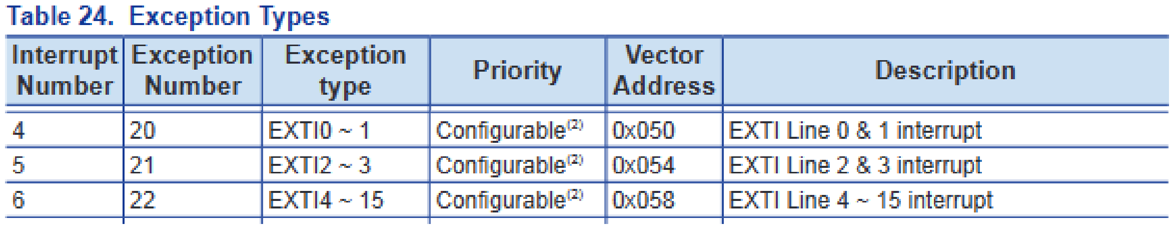
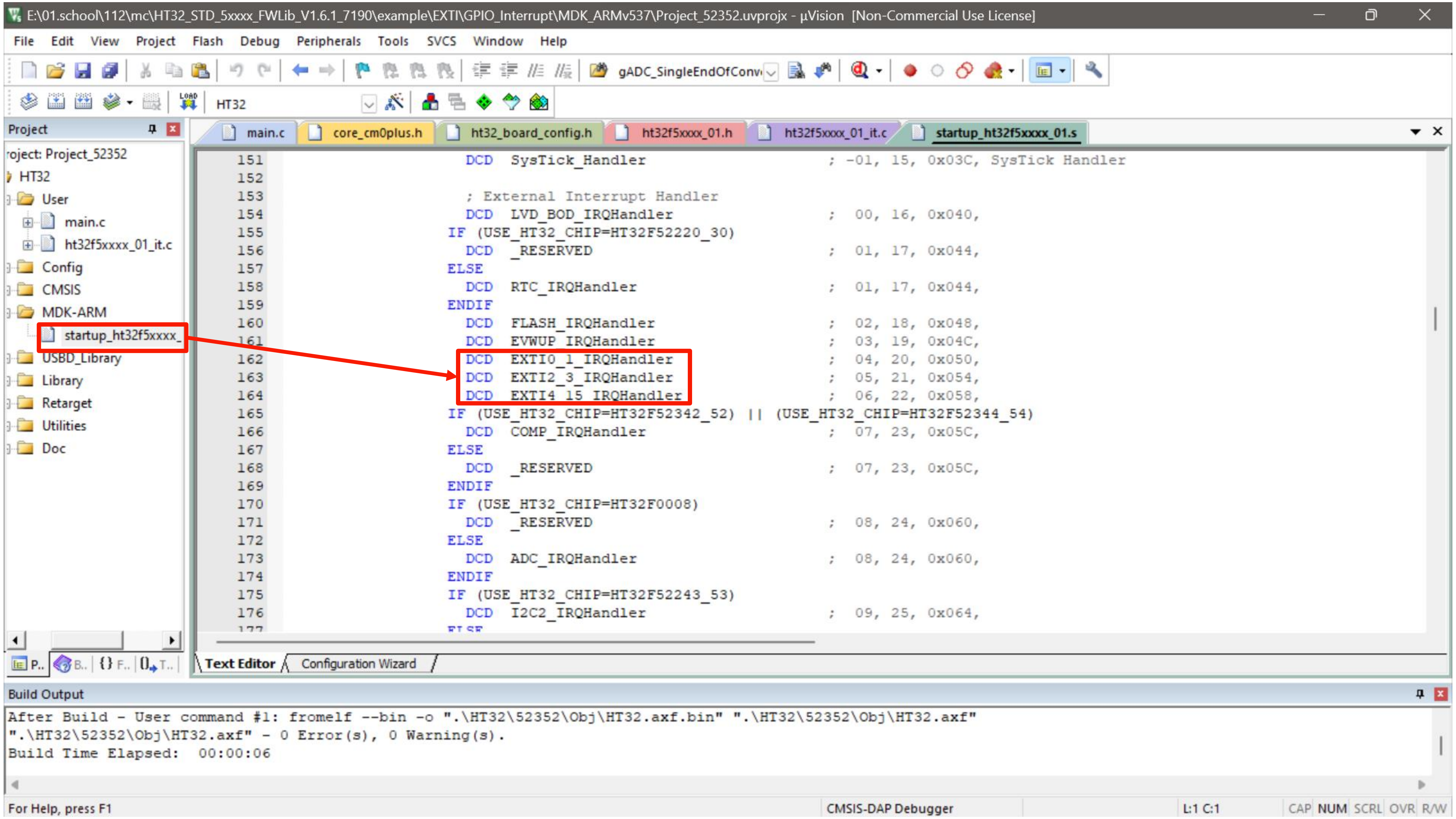


Table 24. Exception Types

Interrupt Number	Exception Number	Exception type	Priority	Vector Address	Description
4	20	EXTI0 ~ 1	Configurable[2]	0x050	EXTI Line 0 & 1 interrupt
5	21	EXTI2 ~ 3	Configurable[2]	0x054	EXTI Line 2 & 3 interrupt
6	22	EXTI4 ~ 15	Configurable[2]	0x058	EXTI Line 4 ~ 15 interrupt

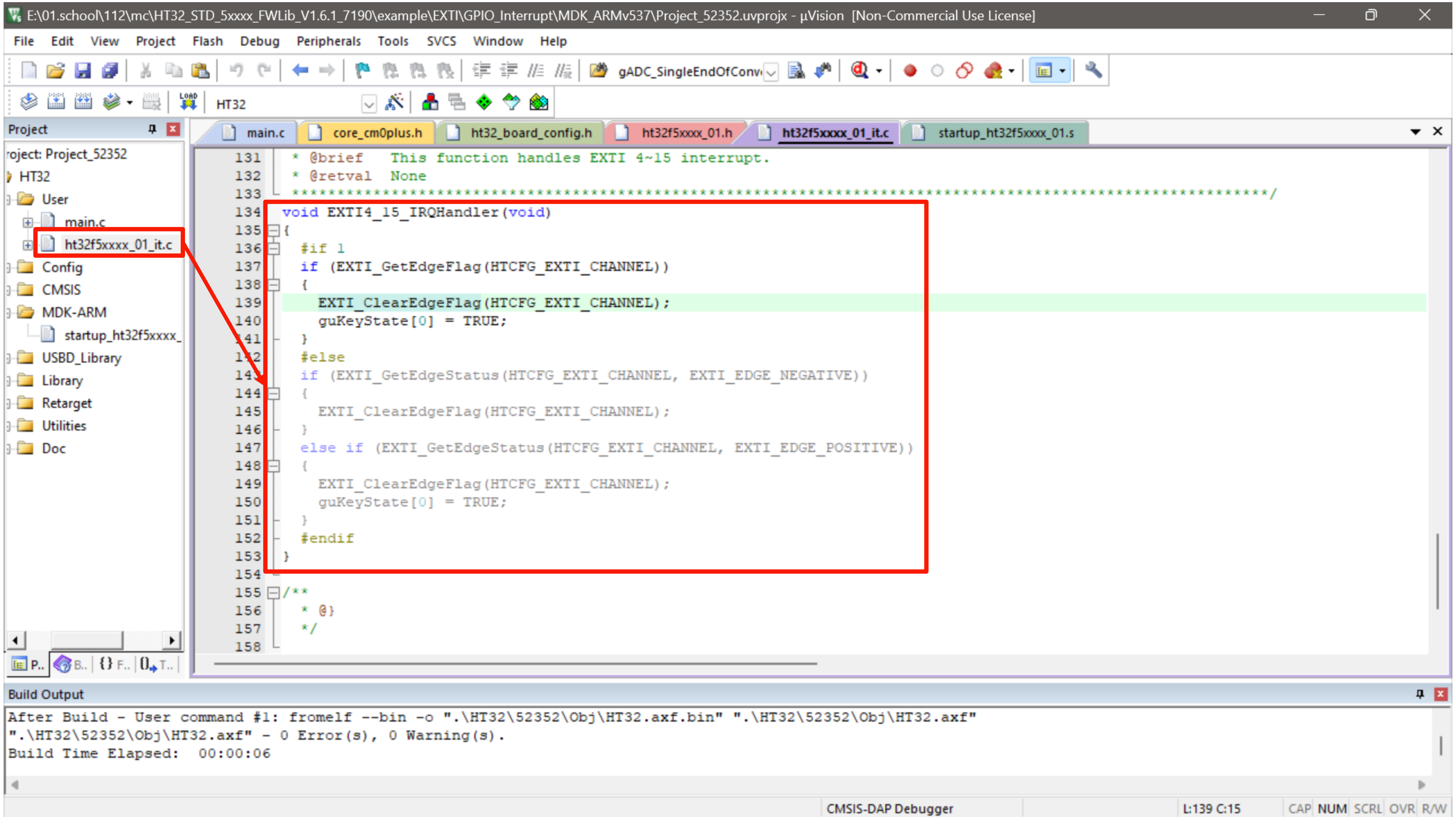
Interrupt Function Name



```
                DCD     SysTick_Handler                   ; -01, 15, 0x03C, SysTick Handler

                ; External Interrupt Handler
                DCD     LVD_BOD_IRQHandler                ;  00, 16, 0x040,
              IF (USE_HT32_CHIP=HT32F52220_30)
                DCD     _RESERVED                         ;  01, 17, 0x044,
              ELSE
                DCD     RTC_IRQHandler                    ;  01, 17, 0x044,
              ENDIF
                DCD     FLASH_IRQHandler                  ;  02, 18, 0x048,
                DCD     EVWUP_IRQHandler                  ;  03, 19, 0x04C,
                DCD     EXTI0_1_IRQHandler                ;  04, 20, 0x050,
                DCD     EXTI2_3_IRQHandler                ;  05, 21, 0x054,
                DCD     EXTI4_15_IRQHandler               ;  06, 22, 0x058,
              IF (USE_HT32_CHIP=HT32F52342_52) || (USE_HT32_CHIP=HT32F52344_54)
                DCD     COMP_IRQHandler                   ;  07, 23, 0x05C,
              ELSE
                DCD     _RESERVED                         ;  07, 23, 0x05C,
              ENDIF
              IF (USE_HT32_CHIP=HT32F0008)
                DCD     _RESERVED                         ;  08, 24, 0x060,
              ELSE
                DCD     ADC_IRQHandler                    ;  08, 24, 0x060,
              ENDIF
              IF (USE_HT32_CHIP=HT32F52243_53)
                DCD     I2C2_IRQHandler                   ;  09, 25, 0x064,
              ELSE
```

```
After Build - User command #1: fromelf --bin -o ".\HT32\52352\Obj\HT32.axf.bin" ".\HT32\52352\Obj\HT32.axf"
".\HT32\52352\Obj\HT32.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:06
```

Contents of Interrupt Service Routine (ISR)



```c
  * @brief  This function handles EXTI 4~15 interrupt.
  * @retval None
  ************************************************************************************************************/
void EXTI4_15_IRQHandler(void)
{
  #if 1
  if (EXTI_GetEdgeFlag(HTCFG_EXTI_CHANNEL))
  {
    EXTI_ClearEdgeFlag(HTCFG_EXTI_CHANNEL);
    guKeyState[0] = TRUE;
  }
  #else
  if (EXTI_GetEdgeStatus(HTCFG_EXTI_CHANNEL, EXTI_EDGE_NEGATIVE))
  {
    EXTI_ClearEdgeFlag(HTCFG_EXTI_CHANNEL);
  }
  else if (EXTI_GetEdgeStatus(HTCFG_EXTI_CHANNEL, EXTI_EDGE_POSITIVE))
  {
    EXTI_ClearEdgeFlag(HTCFG_EXTI_CHANNEL);
    guKeyState[0] = TRUE;
  }
  #endif
}

/**
  * @}
  */
```

```
After Build - User command #1: fromelf --bin -o ".\HT32\52352\Obj\HT32.axf.bin" ".\HT32\52352\Obj\HT32.axf"
".\HT32\52352\Obj\HT32.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:06
```

Interrupt Priority

- To ensure the system can handle all interrupts in real-time, interrupts are categorized into multiple levels based on the importance and urgency of the interrupt events. This categorization, known as interrupt priority, allows setting priorities for interrupts during programming, with lower numerical values indicating higher priority.

EXTI Introduction

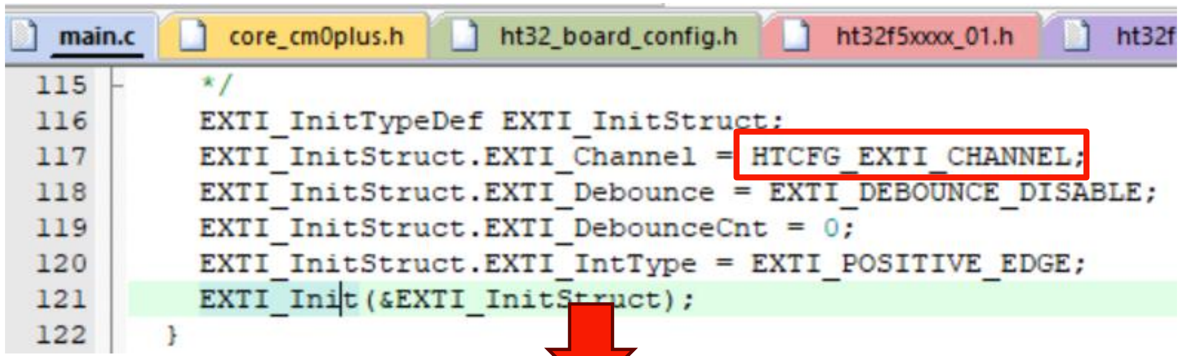
What is EXTI?

- EXTI (External Interrupt) refers to an external interrupt triggered by detecting input pulses through GPIO. It initiates an interrupt event, interrupting the execution flow of the original code and entering the Interrupt Service Routine (ISR) for processing. Once processing is complete, the system returns to the code that was running before the interrupt. All GPIO pins can serve as input sources for external interrupts. Exploiting this feature, we can replace polling detection for buttons with interrupts, significantly improving software efficiency.

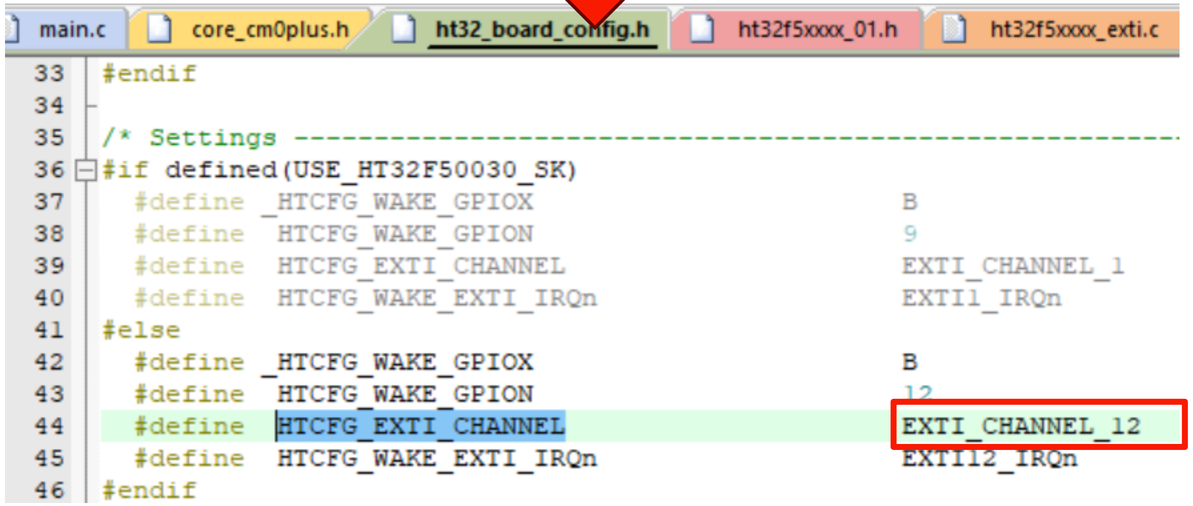
EXTI Key Points:

1. Configure external interrupt sources.
2. Configure debounce for switches.
3. Set debounce delay time.
4. Configure trigger source type.
5. Set up pull-up or pull-down resistors (based on switch hardware configuration).

Configure external interrupt sources



```
    */
    EXTI_InitTypeDef EXTI_InitStruct;
    EXTI_InitStruct.EXTI_Channel = HTCFG_EXTI_CHANNEL;
    EXTI_InitStruct.EXTI_Debounce = EXTI_DEBOUNCE_DISABLE;
    EXTI_InitStruct.EXTI_DebounceCnt = 0;
    EXTI_InitStruct.EXTI_IntType = EXTI_POSITIVE_EDGE;
    EXTI_Init(&EXTI_InitStruct);
  }
```



```
#endif

/* Settings ------------------------------------------------------------
#if defined(USE_HT32F50030_SK)
  #define _HTCFG_WAKE_GPIOX                         B
  #define  HTCFG_WAKE_GPION                         9
  #define  HTCFG_EXTI_CHANNEL                       EXTI_CHANNEL_1
  #define  HTCFG_WAKE_EXTI_IRQn                     EXTI1_IRQn
#else
  #define _HTCFG_WAKE_GPIOX                         B
  #define  HTCFG_WAKE_GPION                         12
  #define  HTCFG_EXTI_CHANNEL                       EXTI_CHANNEL_12
  #define  HTCFG_WAKE_EXTI_IRQn                     EXTI12_IRQn
#endif
```

There are a total of 16 external interrupt sources:
PA0 ~ PD0 corresponds to EXTI_CHANNEL_0,
PA1 ~ PD1 corresponds to EXTI_CHANNEL_1,
PA2 ~ PD2 corresponds to EXTI_CHANNEL_2, and so on.

- UserManual p.176

External Interrupt Pin Selection

The GPIO pins are connected to the 16 EXTI lines as shown in the accompanying figure. For example, the user can set the EXTI0PIN [3:0] field in the ESSR0 register to b0000 to select the GPIO PA0 pin as EXTI line 0 input. Since not all the pins of the Port A ~ D pins are available in all package types, please refer to the pin assignment section for detailed pin information. The setting of the EXTInPIN [3:0] field is invalid when the corresponding pin is not available.

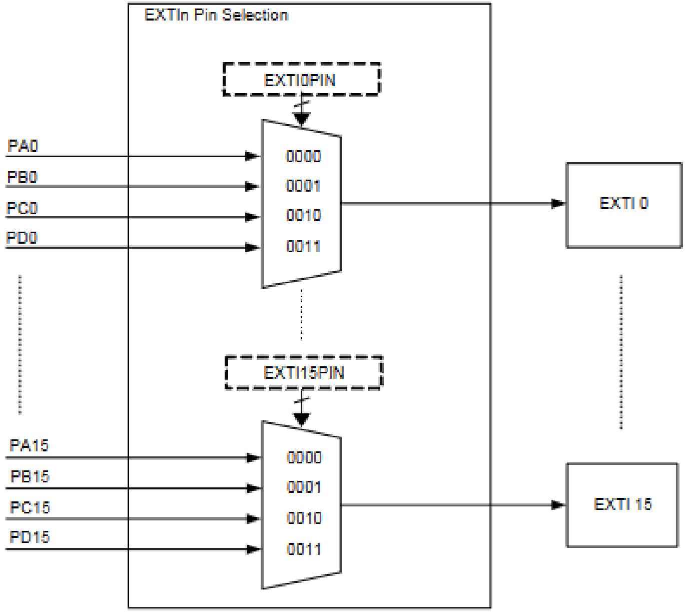
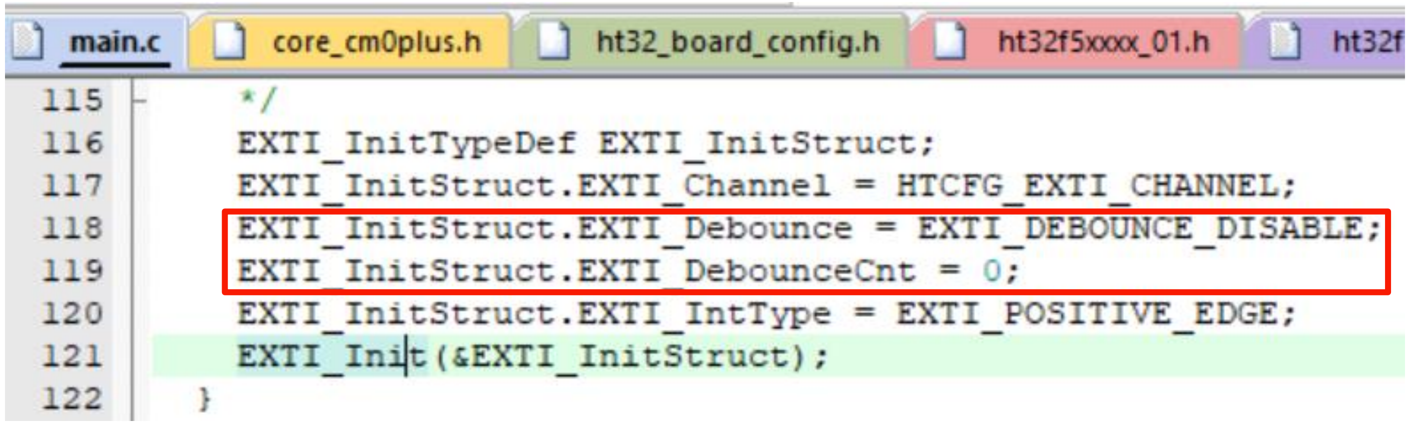


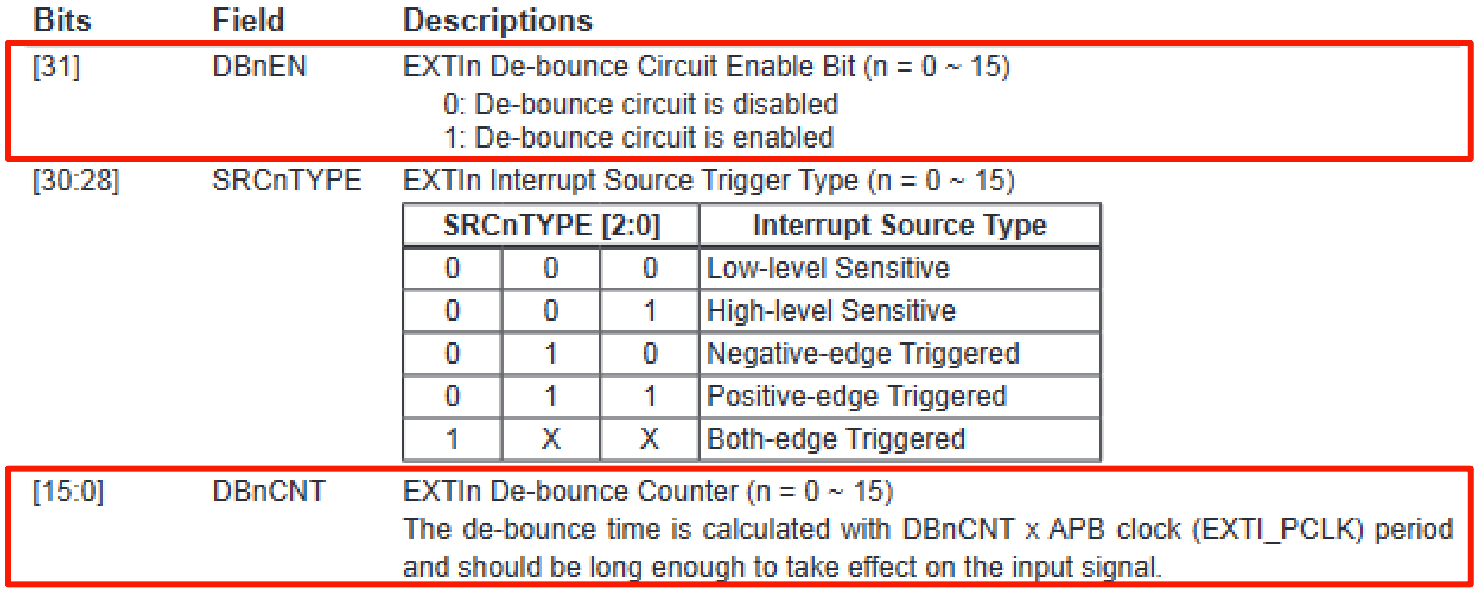
Figure 23. EXTI Channel Input Selection

Configure debounce for switches



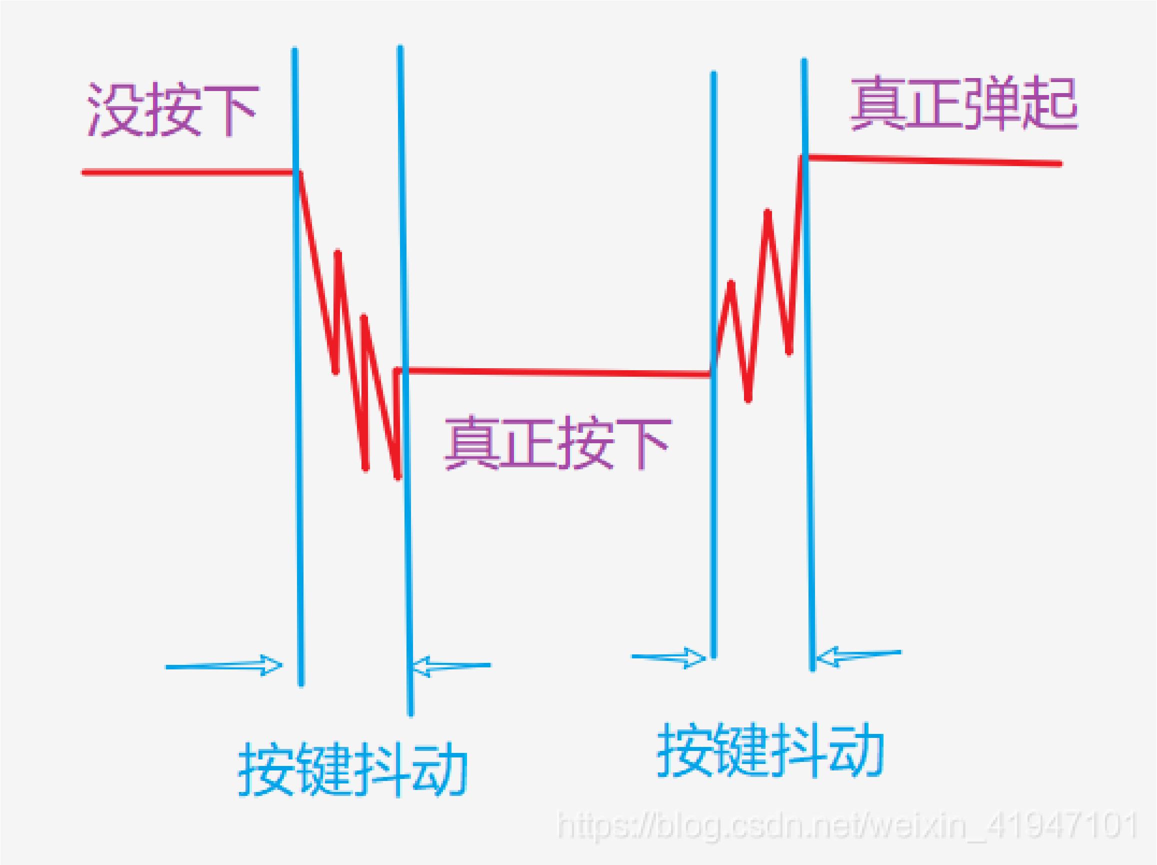
```
    */
    EXTI_InitTypeDef EXTI_InitStruct;
    EXTI_InitStruct.EXTI_Channel = HTCFG_EXTI_CHANNEL;
    EXTI_InitStruct.EXTI_Debounce = EXTI_DEBOUNCE_DISABLE;
    EXTI_InitStruct.EXTI_DebounceCnt = 0;
    EXTI_InitStruct.EXTI_IntType = EXTI_POSITIVE_EDGE;
    EXTI_Init(&EXTI_InitStruct);
  }
```

- ➢ UserManual p.189

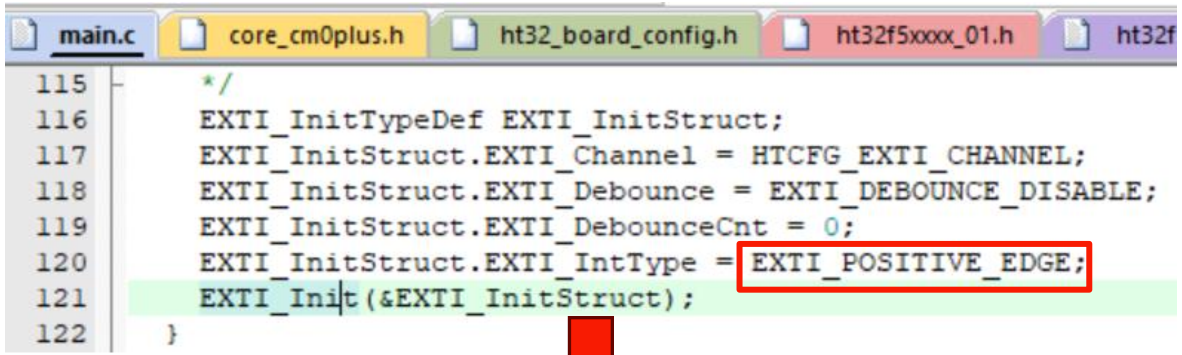


<table>
<tr><th>Bits</th><th>Field</th><th>Descriptions</th></tr>
<tr><td>[31]</td><td>DBnEN</td><td>EXTIn De-bounce Circuit Enable Bit (n = 0 ~ 15)
0: De-bounce circuit is disabled
1: De-bounce circuit is enabled</td></tr>
<tr><td>[30:28]</td><td>SRCnTYPE</td><td>EXTIn Interrupt Source Trigger Type (n = 0 ~ 15)
<table>
<tr><th colspan="3">SRCnTYPE [2:0]</th><th>Interrupt Source Type</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Low-level Sensitive</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>High-level Sensitive</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Negative-edge Triggered</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Positive-edge Triggered</td></tr>
<tr><td>1</td><td>X</td><td>X</td><td>Both-edge Triggered</td></tr>
</table></td></tr>
<tr><td>[15:0]</td><td>DBnCNT</td><td>EXTIn De-bounce Counter (n = 0 ~ 15)
The de-bounce time is calculated with DBnCNT x APB clock (EXTI_PCLK) period and should be long enough to take effect on the input signal.</td></tr>
</table>

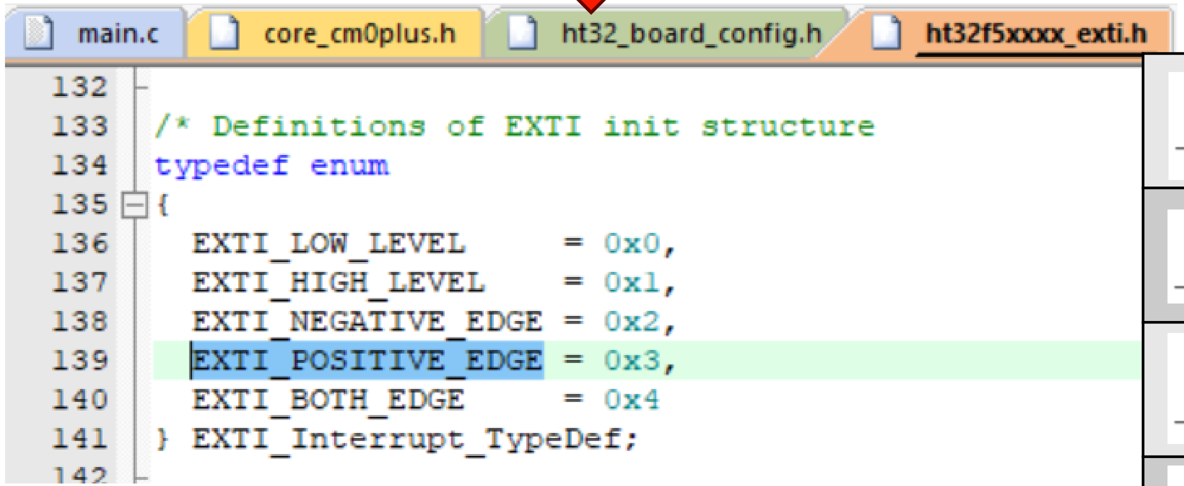
What is debounce?



Configure trigger source type

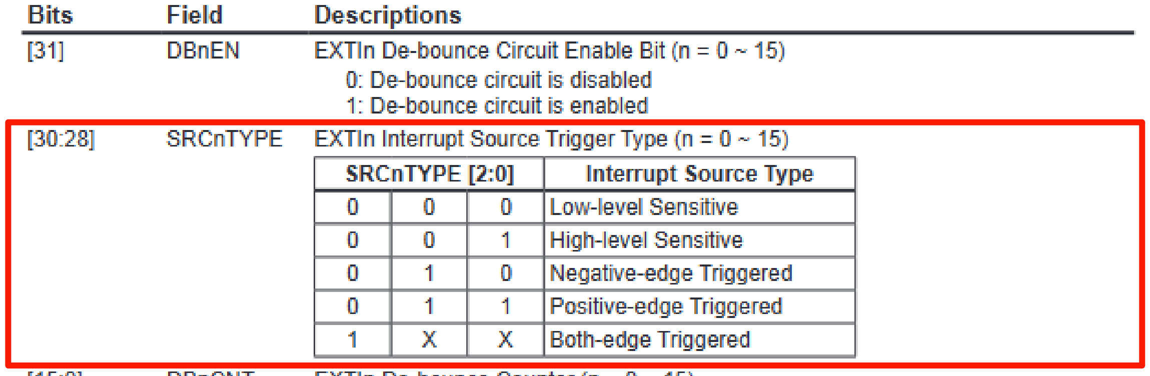


```
    */
    EXTI_InitTypeDef EXTI_InitStruct;
    EXTI_InitStruct.EXTI_Channel = HTCFG_EXTI_CHANNEL;
    EXTI_InitStruct.EXTI_Debounce = EXTI_DEBOUNCE_DISABLE;
    EXTI_InitStruct.EXTI_DebounceCnt = 0;
    EXTI_InitStruct.EXTI_IntType = EXTI_POSITIVE_EDGE;
    EXTI_Init(&EXTI_InitStruct);
  }
```



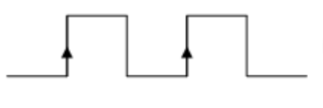
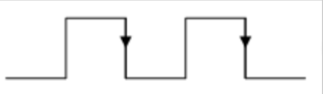
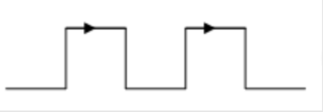
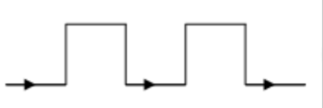
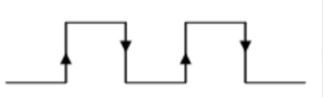
```

/* Definitions of EXTI init structure
typedef enum
{
  EXTI_LOW_LEVEL      = 0x0,
  EXTI_HIGH_LEVEL     = 0x1,
  EXTI_NEGATIVE_EDGE  = 0x2,
  EXTI_POSITIVE_EDGE  = 0x3,
  EXTI_BOTH_EDGE      = 0x4
} EXTI_Interrupt_TypeDef;
```

- UserManual p.189



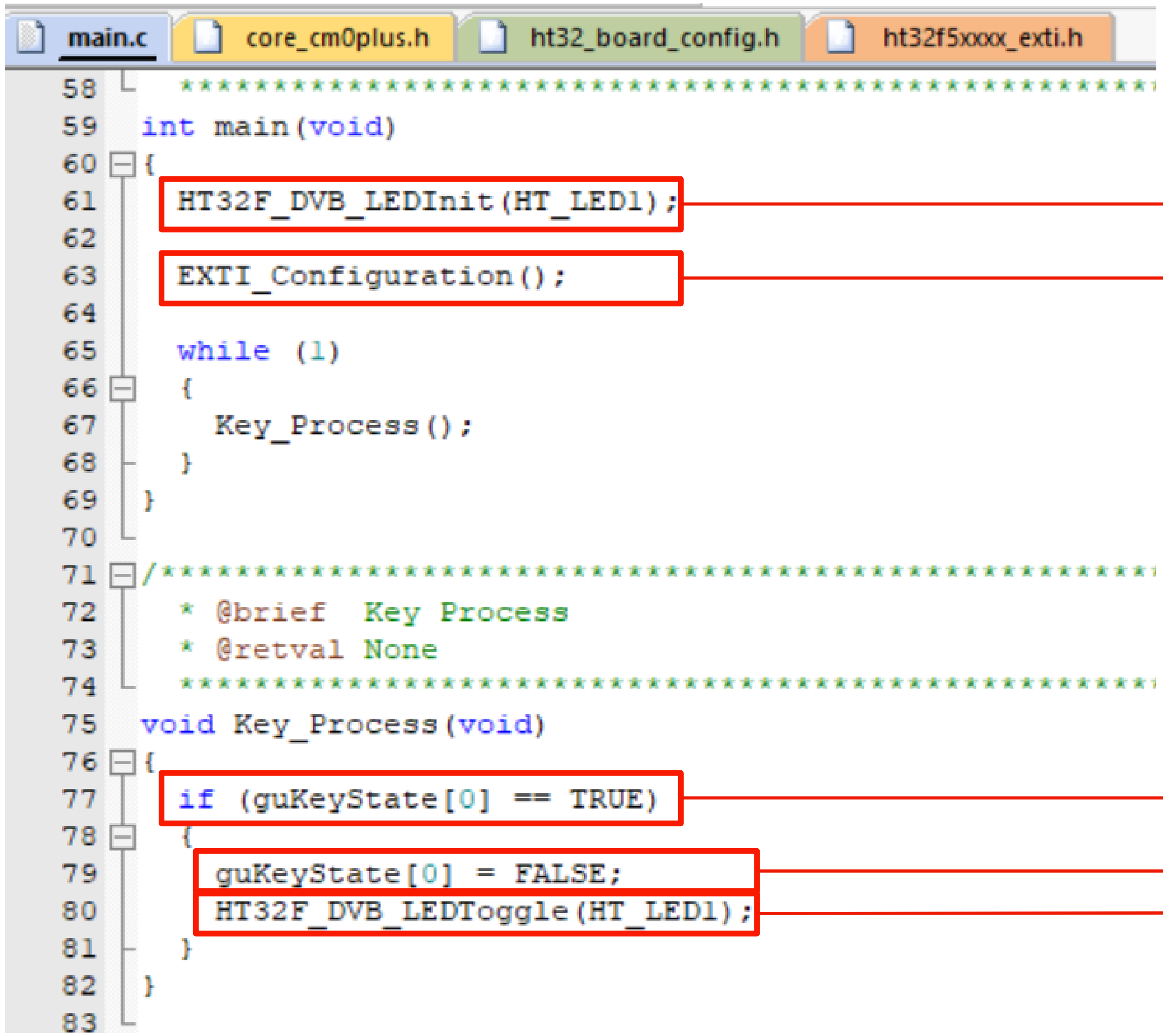
Bits	Field	Descriptions
[31]	DBnEN	EXTIn De-bounce Circuit Enable Bit (n = 0 ~ 15) 0: De-bounce circuit is disabled 1: De-bounce circuit is enabled
[30:28]	SRCnTYPE	EXTIn Interrupt Source Trigger Type (n = 0 ~ 15)

SRCnTYPE [2:0]			Interrupt Source Type
0	0	0	Low-level Sensitive
0	0	1	High-level Sensitive
0	1	0	Negative-edge Triggered
0	1	1	Positive-edge Triggered
1	X	X	Both-edge Triggered

	Rising edge trigger: Used to detect a clean rising edge without any bouncing.
	Falling edge trigger: Used to detect a clean falling edge without any bouncing.
	High-level trigger: Used to detect a high-level state.
	Low-level trigger: Used to detect a low-level state.
	Dual-edge trigger: Used to detect non-bouncing dual edges.

EXTI Project

main.c



```c
  ***************************************************************
int main(void)
{
  HT32F_DVB_LEDInit(HT_LED1);

  EXTI_Configuration();

  while (1)
  {
    Key_Process();
  }
}

/***************************************************************
  * @brief  Key Process
  * @retval None
  ***************************************************************
void Key_Process(void)
{
  if (guKeyState[0] == TRUE)
  {
    guKeyState[0] = FALSE;
    HT32F_DVB_LEDToggle(HT_LED1);
  }
}
```

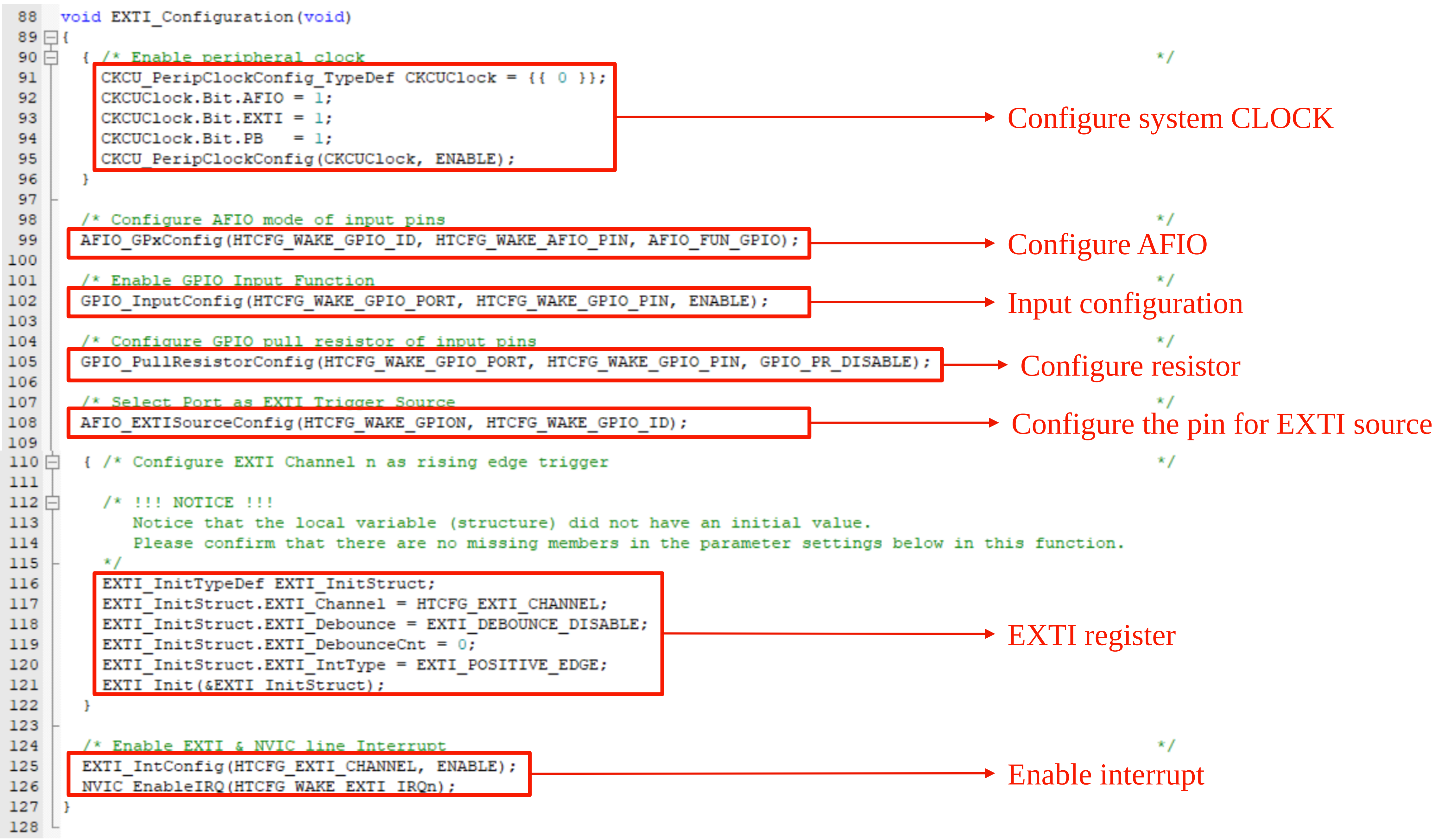
LED Initialization

Configure EXTI

Execute if the interrupt status is enabled

Disable interrupt status

LED1 switch



```c
void EXTI_Configuration(void)
{
  { /* Enable peripheral clock                                                                    */
    CKCU_PeripClockConfig_TypeDef CKCUClock = {{ 0 }};
    CKCUClock.Bit.AFIO = 1;
    CKCUClock.Bit.EXTI = 1;
    CKCUClock.Bit.PB   = 1;
    CKCU_PeripClockConfig(CKCUClock, ENABLE);
  }

  /* Configure AFIO mode of input pins                                                            */
  AFIO_GPxConfig(HTCFG_WAKE_GPIO_ID, HTCFG_WAKE_AFIO_PIN, AFIO_FUN_GPIO);

  /* Enable GPIO Input Function                                                                   */
  GPIO_InputConfig(HTCFG_WAKE_GPIO_PORT, HTCFG_WAKE_GPIO_PIN, ENABLE);

  /* Configure GPIO pull resistor of input pins                                                   */
  GPIO_PullResistorConfig(HTCFG_WAKE_GPIO_PORT, HTCFG_WAKE_GPIO_PIN, GPIO_PR_DISABLE);

  /* Select Port as EXTI Trigger Source                                                           */
  AFIO_EXTISourceConfig(HTCFG_WAKE_GPION, HTCFG_WAKE_GPIO_ID);

  { /* Configure EXTI Channel n as rising edge trigger                                            */

    /* !!! NOTICE !!!
       Notice that the local variable (structure) did not have an initial value.
       Please confirm that there are no missing members in the parameter settings below in this function.
    */
    EXTI_InitTypeDef EXTI_InitStruct;
    EXTI_InitStruct.EXTI_Channel = HTCFG_EXTI_CHANNEL;
    EXTI_InitStruct.EXTI_Debounce = EXTI_DEBOUNCE_DISABLE;
    EXTI_InitStruct.EXTI_DebounceCnt = 0;
    EXTI_InitStruct.EXTI_IntType = EXTI_POSITIVE_EDGE;
    EXTI_Init(&EXTI_InitStruct);
  }

  /* Enable EXTI & NVIC line Interrupt                                                            */
  EXTI_IntConfig(HTCFG_EXTI_CHANNEL, ENABLE);
  NVIC_EnableIRQ(HTCFG_WAKE_EXTI_IRQn);
}
```

Configure system CLOCK

Configure AFIO

Input configuration

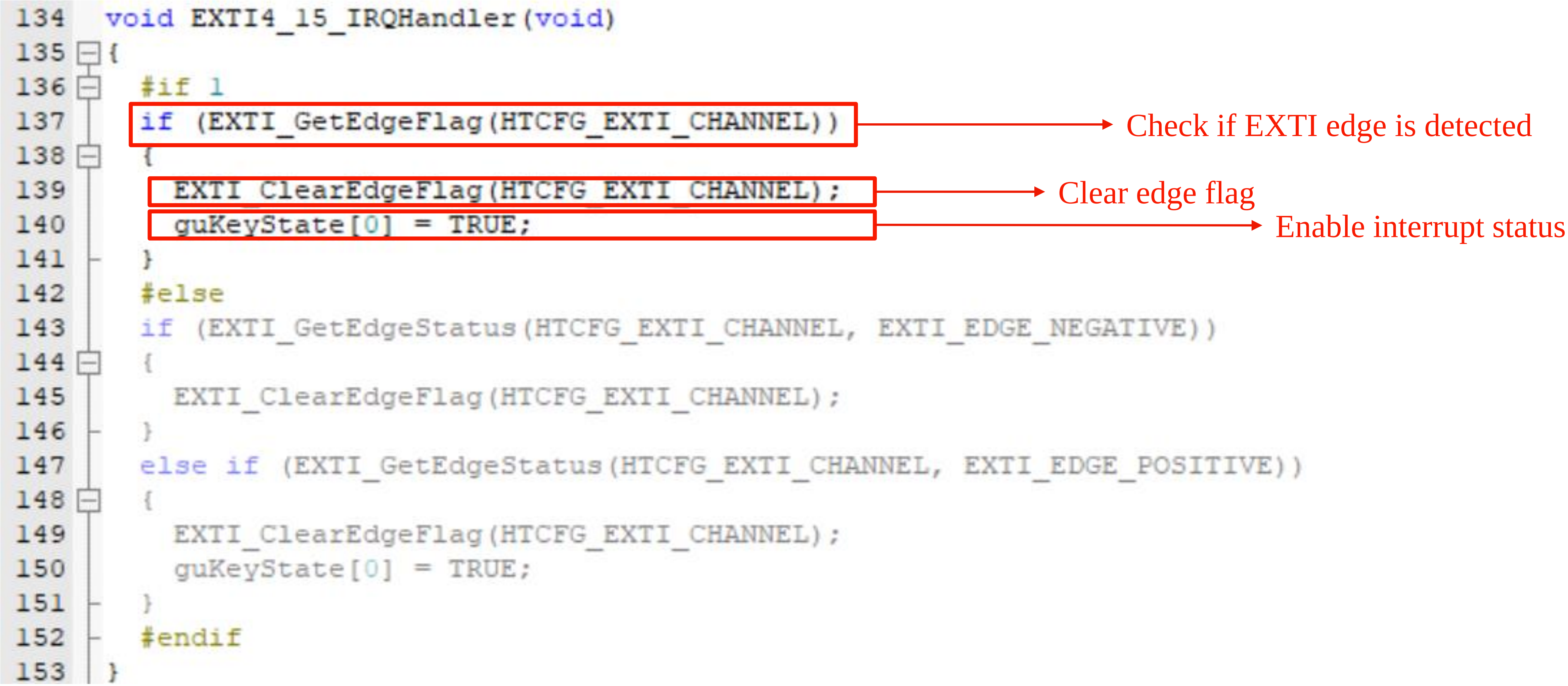
Configure resistor

Configure the pin for EXTI source

EXTI register

Enable interrupt

ht32f5xxxx_01_it.c



```c
void EXTI4_15_IRQHandler(void)
{
  #if 1
  if (EXTI_GetEdgeFlag(HTCFG_EXTI_CHANNEL))
  {
    EXTI_ClearEdgeFlag(HTCFG_EXTI_CHANNEL);
    guKeyState[0] = TRUE;
  }
  #else
  if (EXTI_GetEdgeStatus(HTCFG_EXTI_CHANNEL, EXTI_EDGE_NEGATIVE))
  {
    EXTI_ClearEdgeFlag(HTCFG_EXTI_CHANNEL);
  }
  else if (EXTI_GetEdgeStatus(HTCFG_EXTI_CHANNEL, EXTI_EDGE_POSITIVE))
  {
    EXTI_ClearEdgeFlag(HTCFG_EXTI_CHANNEL);
    guKeyState[0] = TRUE;
  }
  #endif
}
```

Check if EXTI edge is detected

Clear edge flag

Enable interrupt status

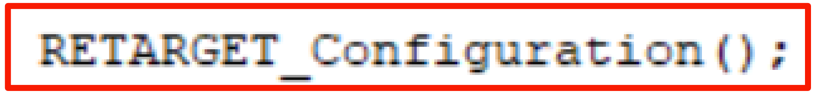
Homework W9-1.

https://github.com/CYCU-AIoT-System-Lab/Microcontroller-Experiment/blob/main/w9/EXTI-GPIO_Interrupt-Experiment_Steps.md

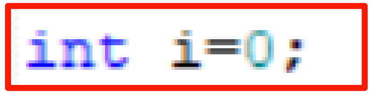
Execute example and display on Tera Term

- Objective: Trigger interrupts, which display on Tera Term and toggles LED, by pressing the button.
- Hint:

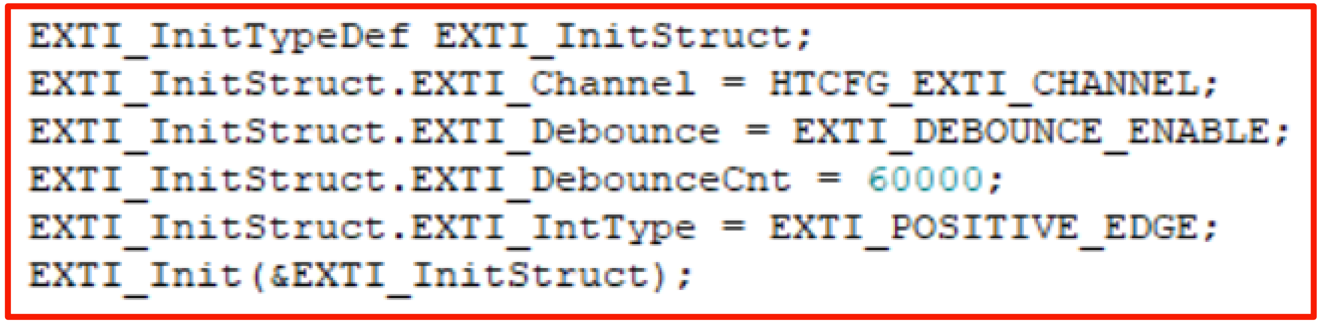
1. Use “F12” to find out the pin of interrupt and LED1.
2. Add the required functions.



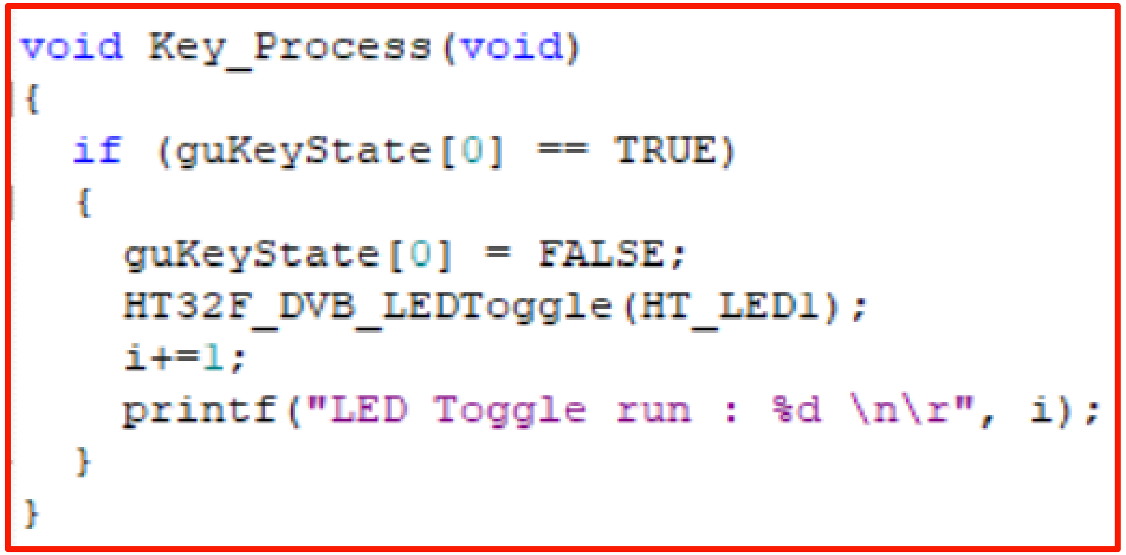
```
RETARGET_Configuration();
```



```
int i=0;
```

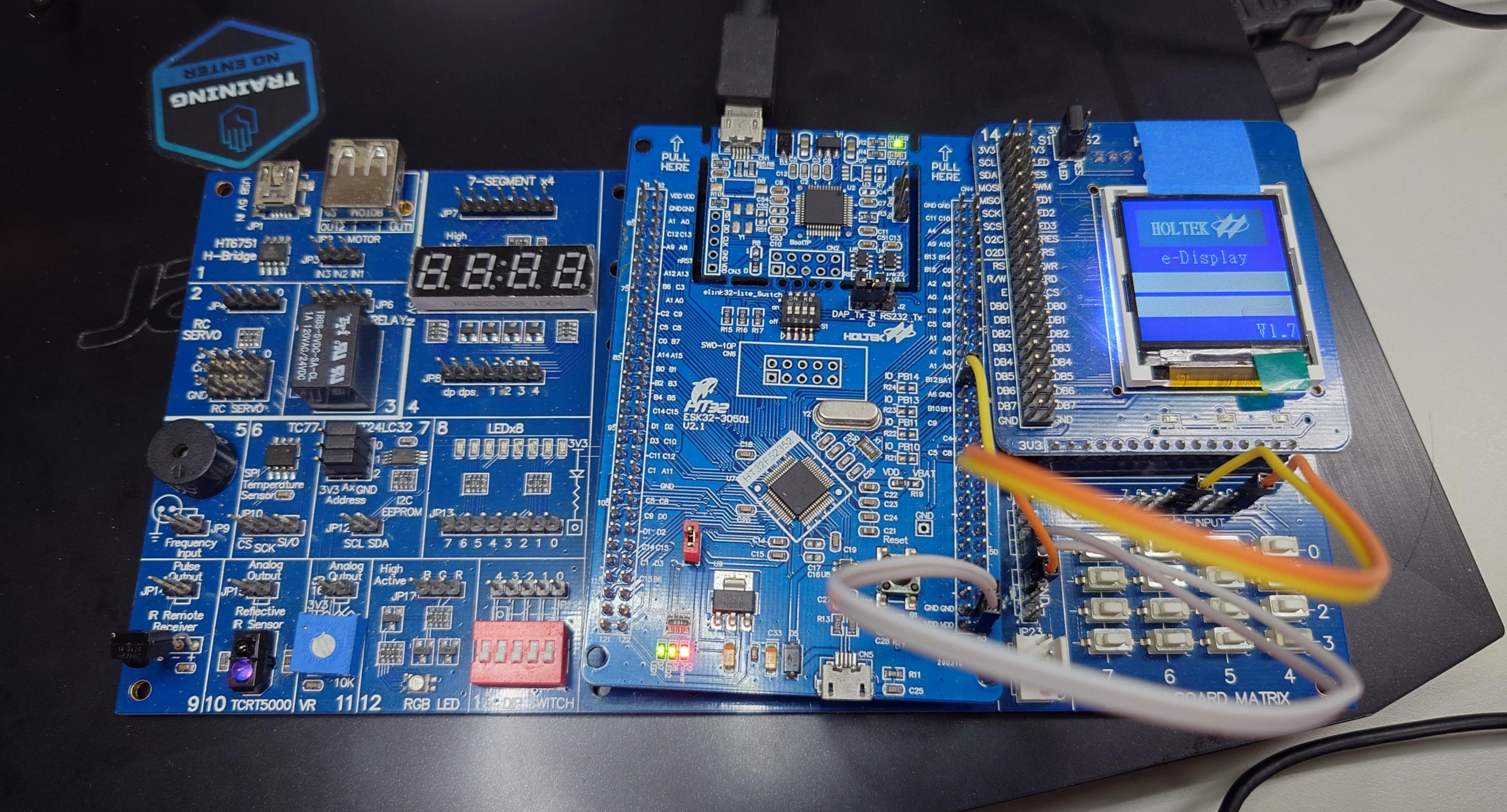


```
EXTI_InitTypeDef EXTI_InitStruct;
EXTI_InitStruct.EXTI_Channel = HTCFG_EXTI_CHANNEL;
EXTI_InitStruct.EXTI_Debounce = EXTI_DEBOUNCE_ENABLE;
EXTI_InitStruct.EXTI_DebounceCnt = 60000;
EXTI_InitStruct.EXTI_IntType = EXTI_POSITIVE_EDGE;
EXTI_Init(&EXTI_InitStruct);
```



```
void Key_Process(void)
{
  if (guKeyState[0] == TRUE)
  {
    guKeyState[0] = FALSE;
    HT32F_DVB_LEDToggle(HT_LED1);
    i+=1;
    printf("LED Toggle run : %d \n\r", i);
  }
}
```

☆ PS. Please record.



https://youtu.be/IBAPzCThSmw

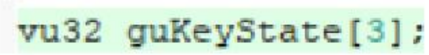
Homework W9-2.

Add two interrupts & LEDs

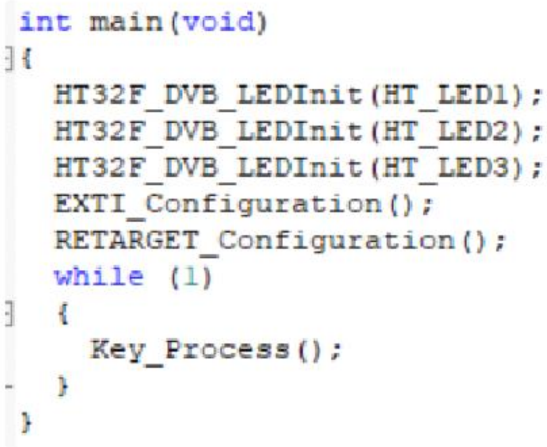
- Objective: Toggle LED1, LED2, and LED3 accordingly when the corresponding buttons are pressed.
- Hint:

1. Use “F12” to find out the pin of interrupts and LED1、LED2、LED3.
2. Add the required functions.
3. Connect wires with following requirement.
    1. EXTI channel 11, C11 to JP24-5.
    2. EXTI channel 10, C10 to JP24-6
    3. LED1~3 to JP13-7~5.

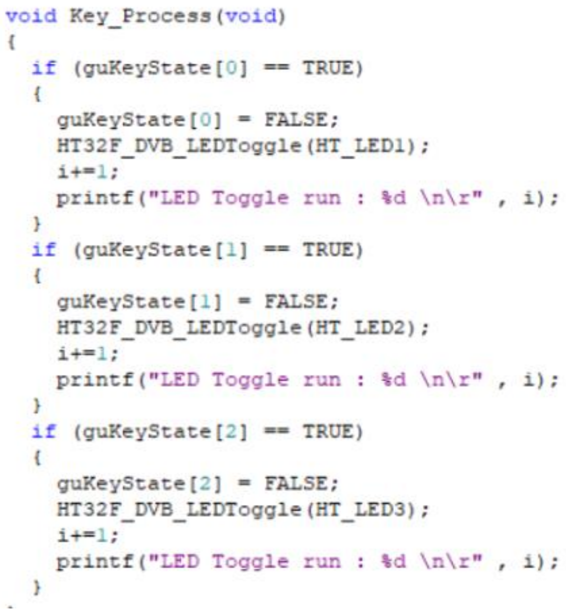
☆ PS. Please record.



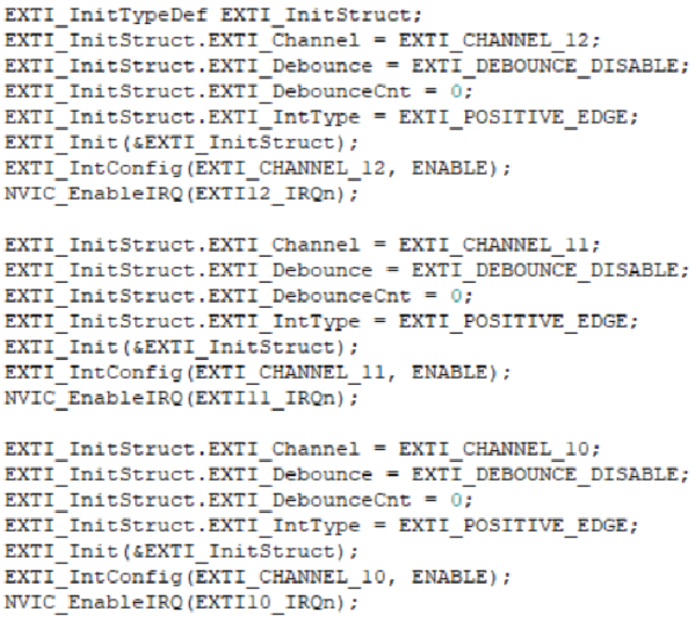
```c
vu32 guKeyState[3];
```



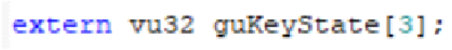
```c
int main(void)
{
  HT32F_DVB_LEDInit(HT_LED1);
  HT32F_DVB_LEDInit(HT_LED2);
  HT32F_DVB_LEDInit(HT_LED3);
  EXTI_Configuration();
  RETARGET_Configuration();
  while (1)
  {
    Key_Process();
  }
}
```



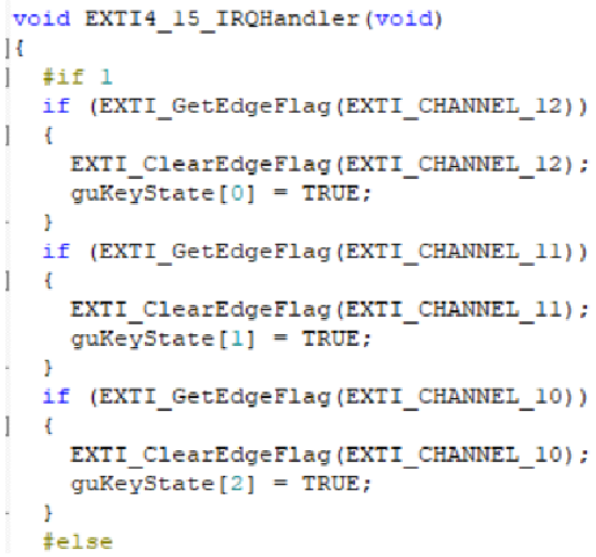
```c
void Key_Process(void)
{
  if (guKeyState[0] == TRUE)
  {
    guKeyState[0] = FALSE;
    HT32F_DVB_LEDToggle(HT_LED1);
    i+=1;
    printf("LED Toggle run : %d \n\r" , i);
  }
  if (guKeyState[1] == TRUE)
  {
    guKeyState[1] = FALSE;
    HT32F_DVB_LEDToggle(HT_LED2);
    i+=1;
    printf("LED Toggle run : %d \n\r" , i);
  }
  if (guKeyState[2] == TRUE)
  {
    guKeyState[2] = FALSE;
    HT32F_DVB_LEDToggle(HT_LED3);
    i+=1;
    printf("LED Toggle run : %d \n\r" , i);
  }
```



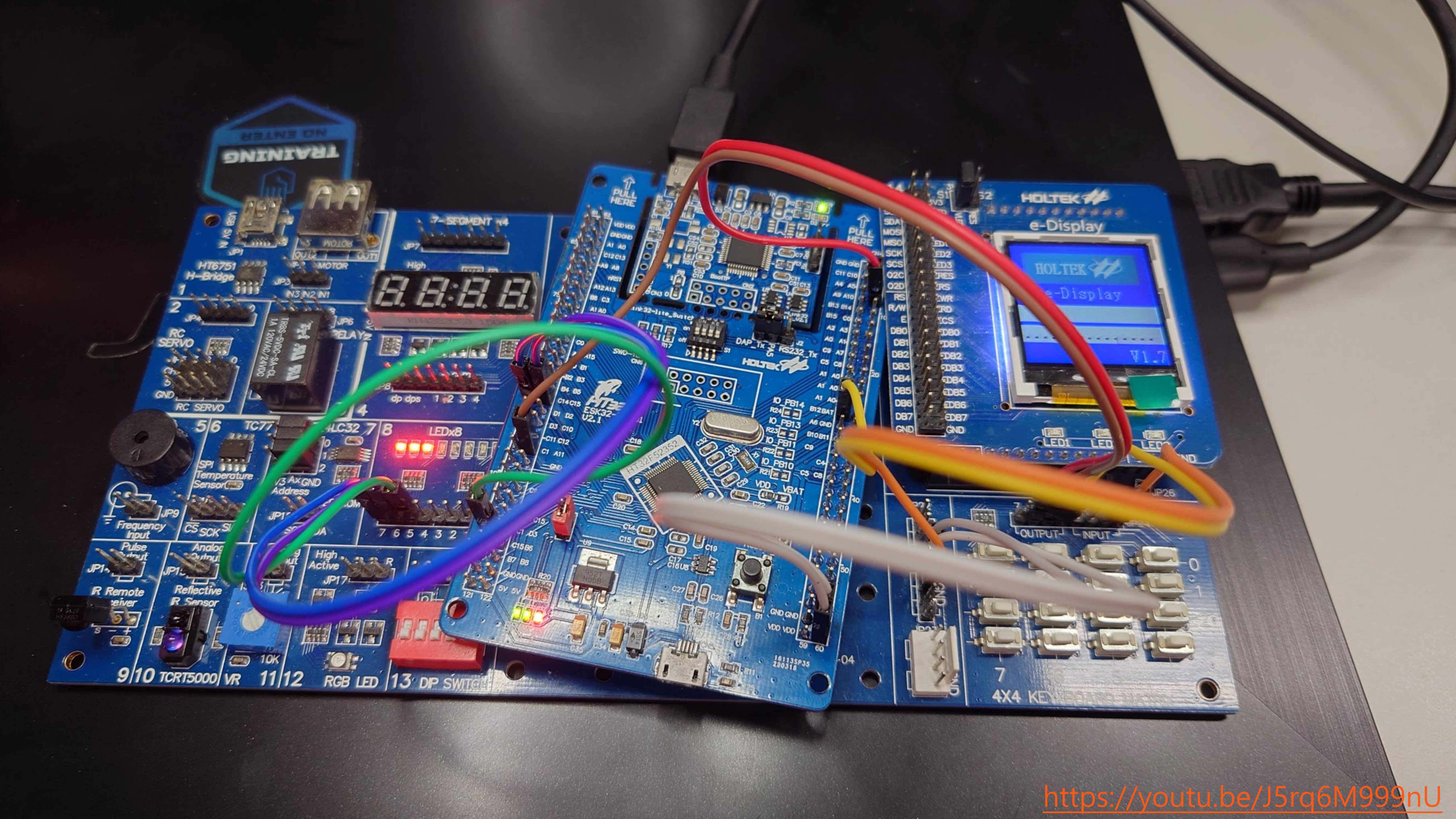
```c
EXTI_InitTypeDef EXTI_InitStruct;
EXTI_InitStruct.EXTI_Channel = EXTI_CHANNEL_12;
EXTI_InitStruct.EXTI_Debounce = EXTI_DEBOUNCE_DISABLE;
EXTI_InitStruct.EXTI_DebounceCnt = 0;
EXTI_InitStruct.EXTI_IntType = EXTI_POSITIVE_EDGE;
EXTI_Init(&EXTI_InitStruct);
EXTI_IntConfig(EXTI_CHANNEL_12, ENABLE);
NVIC_EnableIRQ(EXTI12_IRQn);

EXTI_InitStruct.EXTI_Channel = EXTI_CHANNEL_11;
EXTI_InitStruct.EXTI_Debounce = EXTI_DEBOUNCE_DISABLE;
EXTI_InitStruct.EXTI_DebounceCnt = 0;
EXTI_InitStruct.EXTI_IntType = EXTI_POSITIVE_EDGE;
EXTI_Init(&EXTI_InitStruct);
EXTI_IntConfig(EXTI_CHANNEL_11, ENABLE);
NVIC_EnableIRQ(EXTI11_IRQn);

EXTI_InitStruct.EXTI_Channel = EXTI_CHANNEL_10;
EXTI_InitStruct.EXTI_Debounce = EXTI_DEBOUNCE_DISABLE;
EXTI_InitStruct.EXTI_DebounceCnt = 0;
EXTI_InitStruct.EXTI_IntType = EXTI_POSITIVE_EDGE;
EXTI_Init(&EXTI_InitStruct);
EXTI_IntConfig(EXTI_CHANNEL_10, ENABLE);
NVIC_EnableIRQ(EXTI10_IRQn);
```



```c
extern vu32 guKeyState[3];
```



```c
void EXTI4_15_IRQHandler(void)
{
  #if 1
  if (EXTI_GetEdgeFlag(EXTI_CHANNEL_12))
  {
    EXTI_ClearEdgeFlag(EXTI_CHANNEL_12);
    guKeyState[0] = TRUE;
  }
  if (EXTI_GetEdgeFlag(EXTI_CHANNEL_11))
  {
    EXTI_ClearEdgeFlag(EXTI_CHANNEL_11);
    guKeyState[1] = TRUE;
  }
  if (EXTI_GetEdgeFlag(EXTI_CHANNEL_10))
  {
    EXTI_ClearEdgeFlag(EXTI_CHANNEL_10);
    guKeyState[2] = TRUE;
  }
  #else
```



https://youtu.be/J5rq6M999nU

Homework W9-3.

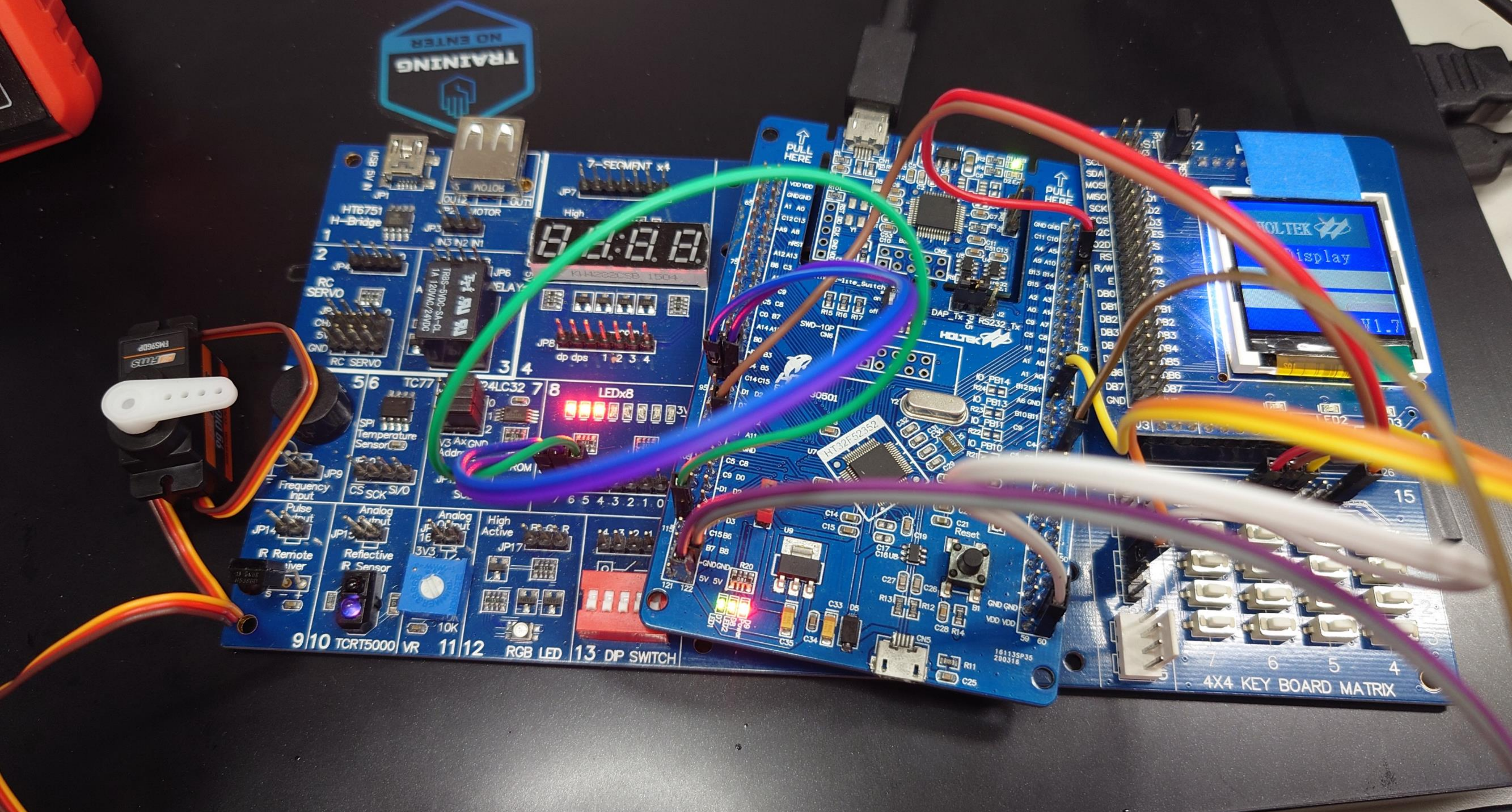
(Bonus)

Use EXTI to control three motors

- Objective: Use EXTI to control SG90 based on HW7-4.
- Hint:

1. Keep all of the wire connection in HW9-2 and add new ones to control SG90.
2. Implement button interrupt to rotate the servo motor to 0, 90, and 180 degrees angle.
3. PWM_CH2.

☆ PS. Please record and explain the code.



https://youtu.be/H4Aaaw6TtZY

Class Dismissed