

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.

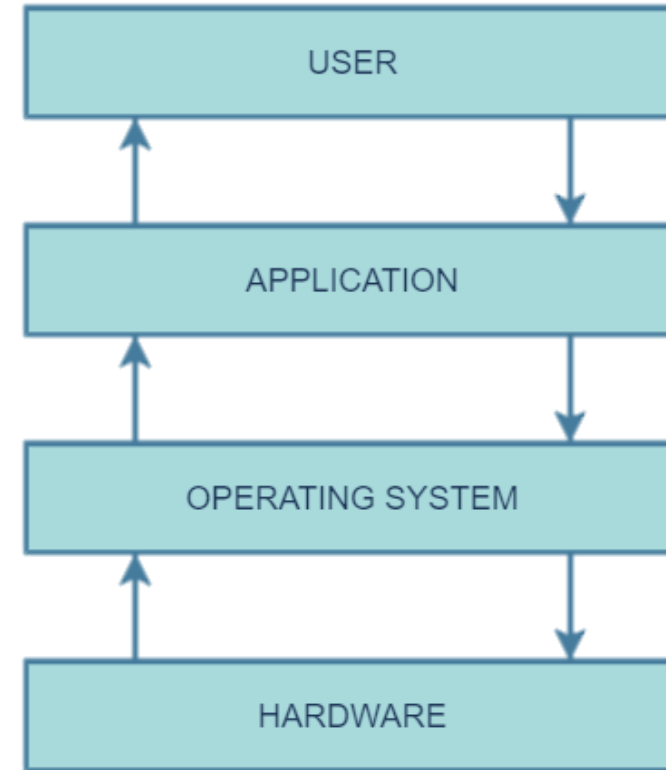
The logo consists of a large red rectangle with a smaller red rectangle on top of it. The word "RTOS" is written in white, sans-serif, uppercase letters in the center of the larger rectangle. The background features a light gray pattern of concentric circles and dashed lines.

RTOS

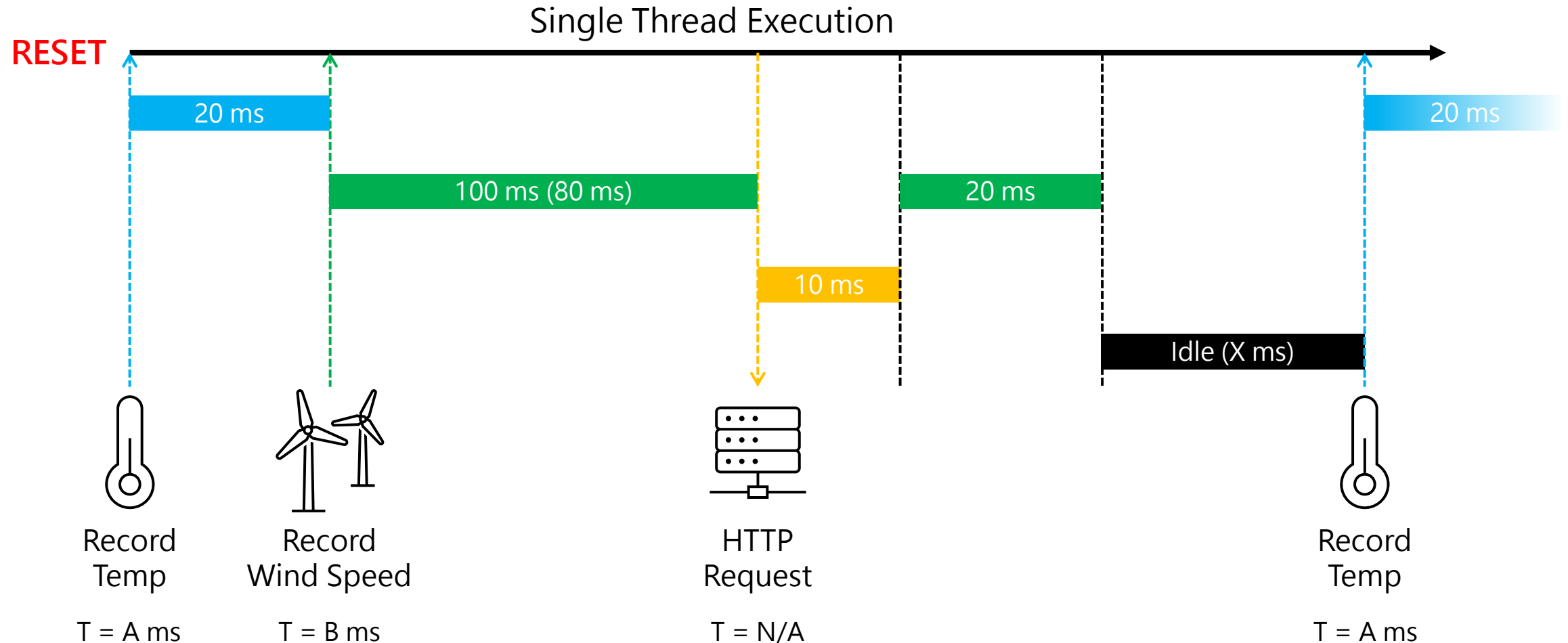
Operating System

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](https://en.wikipedia.org/wiki/Operating_system)



Scheduling – Weather Station



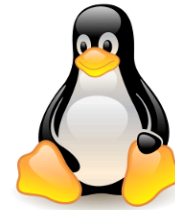
Types of OS



RTOS

Real Time Operation System

[RTOS real-life example](#)



Linux



GPOS

General Purpose Operation System

Hardware OSs Operate On



RTOS

Real Time Operation System

[RTOS vs. GPOS](#)



GPOS

General Purpose Operation System

The background features a series of concentric circles in light gray, some solid and some dashed, creating a ripple effect. A large red speech bubble is centered on the page, with the text 'NVIC Introduction' written inside in white.

NVIC Introduction

Open the example project

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

File Explorer navigation path: > 01.school > 112 > 112 > mc > HT32_STD_5xxxx_FWLib_V1.5.1_7084 > mc > HT32_STD_5xxxx_FWLib_V1.5.1_7084 > example > EXTI > GPIO_Interrupt > MDK_ARMv537

Search: 搜尋 MDK_ARMv537

Actions: 排序, 檢視, 詳細資料

名稱	日期	類型	大小	標籤
ADC		檔案資料夾		
ADC_24bit		檔案資料夾		
AES		檔案資料夾		
BFTM		檔案資料夾		
CAN		檔案資料夾		
CKCU		檔案資料夾		
CMP		檔案資料夾		
CRC		檔案資料夾		
DAC		檔案資料夾		
DAC_Dual16		檔案資料夾		
EBI		檔案資料夾		
EXTI		檔案資料夾		
FMC		檔案資料夾		
GPIO		檔案資料夾		
GPTM		檔案資料夾		
HWDIV		檔案資料夾		
I2C		檔案資料夾		
EXTI_Key_LED		檔案資料夾		
GPIO_Interrupt		檔案資料夾		
Software_Trigger		檔案資料夾		
WakeUp_DeepSleepM...		檔案資料夾		
WakeUp_SleepMode		檔案資料夾		
MDK_ARMv5		檔案資料夾		
MDK_ARMv537		檔案資料夾		
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	狸ision5 Project	24 KB	
startup_ht32f5xxxx_01.s	2023/9/11 下午 10:41	S 檔案	22 KB	
_ClearProject.bat		檔案		
_ClearTarget.bat		檔案		
_CreateProject.bat		檔案		
_CreateProjectScript.b...		檔案		
ht32_board_config.h		檔案		
ht32f5xxxx_01_it.c		檔案		
ht32f5xxxx_01_usbdco...		檔案		
ht32f5xxxx_02_usbdco...		檔案		
ht32f5xxxx_conf.h		檔案		
main.c		檔案		
readme.txt		檔案		
system_ht32f5xxxx_01.c		檔案		

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)

```
main.c core_cm0plus.h ht32_board_config.h ht32f5xxx
124 /* Enable EXTI & NVIC line Interrupt
125 EXTI_IntConfig(HTCFG_EXTI_CHANNEL, ENABLE);
126 NVIC_EnableIRQ(HTCFG_WAKE_EXTI_IRQn);
127 }
```



```
main.c core_cm0plus.h ht32_board_config.h ht32f5xxxx_01.h ht32f5xxxx_01_it.c
739 \note IRQn must not be negative.
740 */
741 __STATIC_INLINE void __NVIC_EnableIRQ(IRQn_Type IRQn)
742 {
743     if ((int32_t)(IRQn) >= 0)
744     {
745         __COMPILER_BARRIER();
746         NVIC->ISER[0U] = (uint32_t)(1UL << (((uint32_t)IRQn) & 0x1FUL));
747         __COMPILER_BARRIER();
748     }
749 }
```



```
main.c core_cm0plus.h ht32_board_config.h ht32f5xxxx_01.h ht32f5xxxx_01_it.c
244 #elif defined(USE_HT32F50020_30)
245 EXTI4_7_IRQn = 6, /*!< EXTI4-7 Line detection Interrupt
246 #else
247 EXTI4_15_IRQn = 6, /*!< EXTI4-15 Line detection Interrupt
```

```
main.c core_cm0plus.h ht32_board_config.h ht32f5xxx
124 /* Enable EXTI & NVIC line Interrupt
125 EXTI_IntConfig(HTCFG_EXTI_CHANNEL, ENABLE);
126 NVIC_EnableIRQ(HTCFG_WAKE_EXTI_IRQn);
127 }
```



```
42 #define _HTCFG_WAKE_GPIOX B
43 #define _HTCFG_WAKE_GPION 12
44 #define _HTCFG_EXTI_CHANNEL EXTI_CHANNEL_12
45 #define _HTCFG_WAKE_EXTI_IRQn EXTI12_IRQn
```



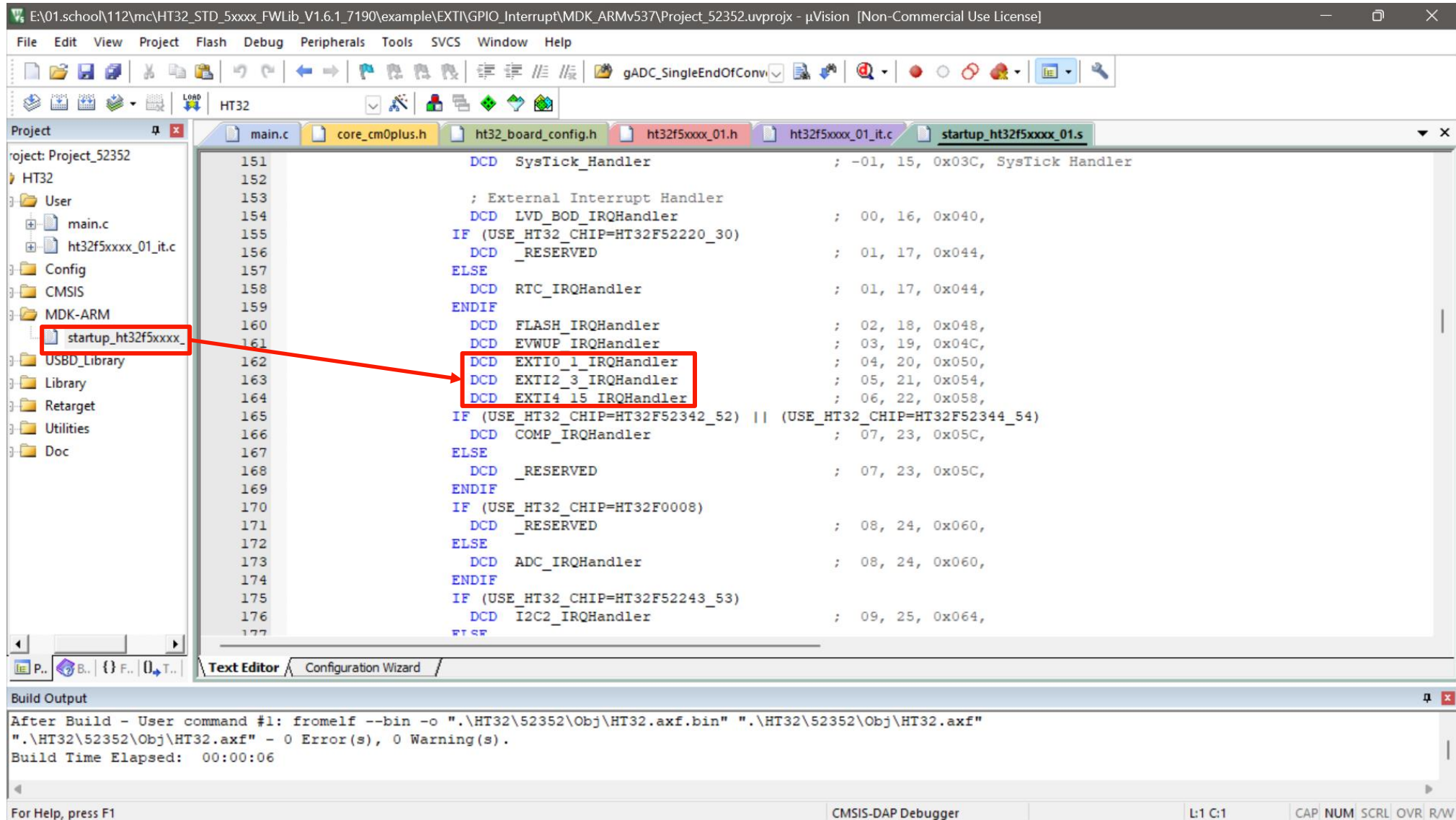
```
main.c core_cm0plus.h ht32_board_config.h ht32f5xxxx_01.h
445 #define EXTI11_IRQn EXTI4_15_IRQn
446 #define EXTI12_IRQn EXTI4_15_IRQn
447 #define EXTI13_IRQn EXTI4_15_IRQn
448 #define EXTI14_IRQn EXTI4_15_IRQn
449 #define EXTI15_IRQn EXTI4_15_IRQn
```

➤ UserManual p.182

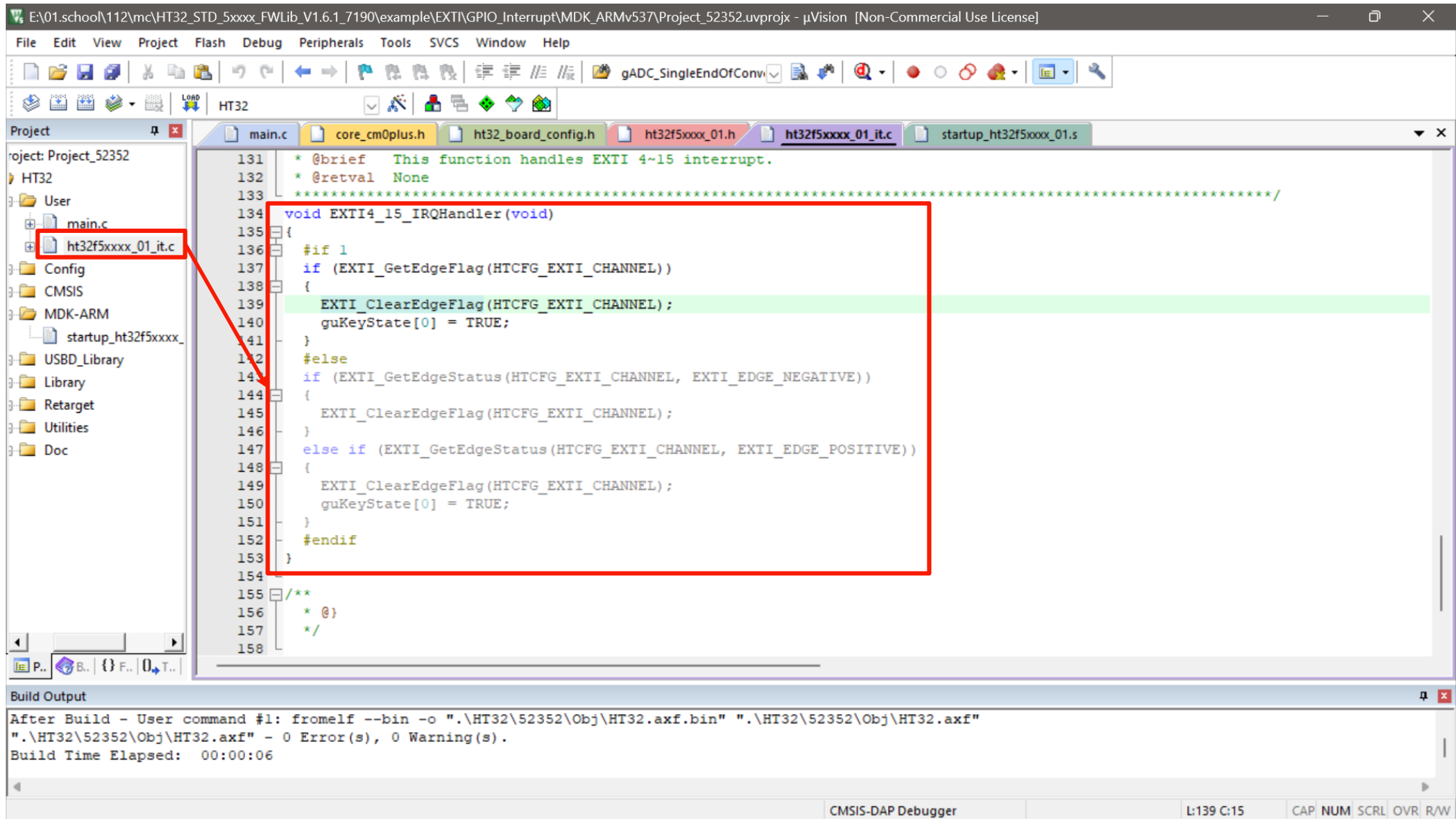
Table 24. Exception Types

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



Contents of Interrupt Service Routine (ISR)



```
131  * @brief This function handles EXTI 4~15 interrupt.
132  * @retval None
133  *****/
134  void EXTI4_15_IRQHandler(void)
135  {
136      #if 1
137      if (EXTI_GetEdgeFlag(HTCFG_EXTI_CHANNEL))
138      {
139          EXTI_ClearEdgeFlag(HTCFG_EXTI_CHANNEL);
140          guKeyState[0] = TRUE;
141      }
142      #else
143      if (EXTI_GetEdgeStatus(HTCFG_EXTI_CHANNEL, EXTI_EDGE_NEGATIVE))
144      {
145          EXTI_ClearEdgeFlag(HTCFG_EXTI_CHANNEL);
146      }
147      else if (EXTI_GetEdgeStatus(HTCFG_EXTI_CHANNEL, EXTI_EDGE_POSITIVE))
148      {
149          EXTI_ClearEdgeFlag(HTCFG_EXTI_CHANNEL);
150          guKeyState[0] = TRUE;
151      }
152      #endif
153  }
154
155  /**
156   * @}
157   */
158
```

Build Output

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

CMSIS-DAP Debugger L:139 C:15 CAP NUM SCRL OVR: R/W

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.

The background features a series of concentric circles in light gray, some solid and some dashed, creating a ripple effect. Overlaid on this is a large red speech bubble with a pointed bottom. Inside the bubble, the text "EXTI Introduction" is written in white.

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

```
main.c | core_cm0plus.h | ht32_board_config.h | ht32f5xxxx_01.h | ht32f5xxxx_exti.c
115  /*
116  EXTI_InitTypeDef EXTI_InitStructure;
117  EXTI_InitStructure.EXTI_Channel = HTCFG_EXTI_CHANNEL;
118  EXTI_InitStructure.EXTI_Debounce = EXTI_DEBOUNCE_DISABLE;
119  EXTI_InitStructure.EXTI_DebounceCnt = 0;
120  EXTI_InitStructure.EXTI_IntType = EXTI_POSITIVE_EDGE;
121  EXTI_Init(&EXTI_InitStructure);
122  }

main.c | core_cm0plus.h | ht32_board_config.h | ht32f5xxxx_01.h | ht32f5xxxx_exti.c
33  #endif
34
35  /* Settings -----
36  #if defined(USE_HT32F50030_SK)
37  #define HTCFG_WAKE_GPIOX          B
38  #define HTCFG_WAKE_GPION          9
39  #define HTCFG_EXTI_CHANNEL        EXTI_CHANNEL_1
40  #define HTCFG_WAKE_EXTI_IRQn     EXTI1_IRQn
41  #else
42  #define HTCFG_WAKE_GPIOX          B
43  #define HTCFG_WAKE_GPION          12
44  #define HTCFG_EXTI_CHANNEL        EXTI_CHANNEL_12
45  #define HTCFG_WAKE_EXTI_IRQn     EXTI12_IRQn
46  #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 EXTI0PIN [3:0] field is invalid when the corresponding pin is not available.

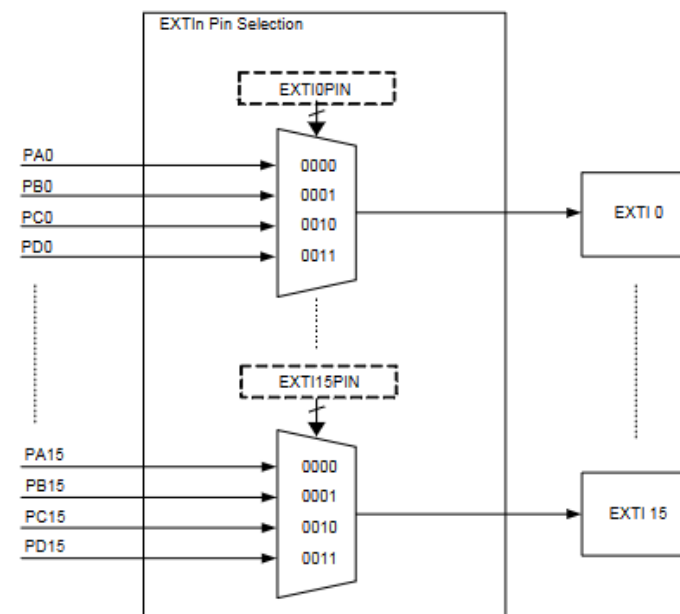


Figure 23. EXTI Channel Input Selection

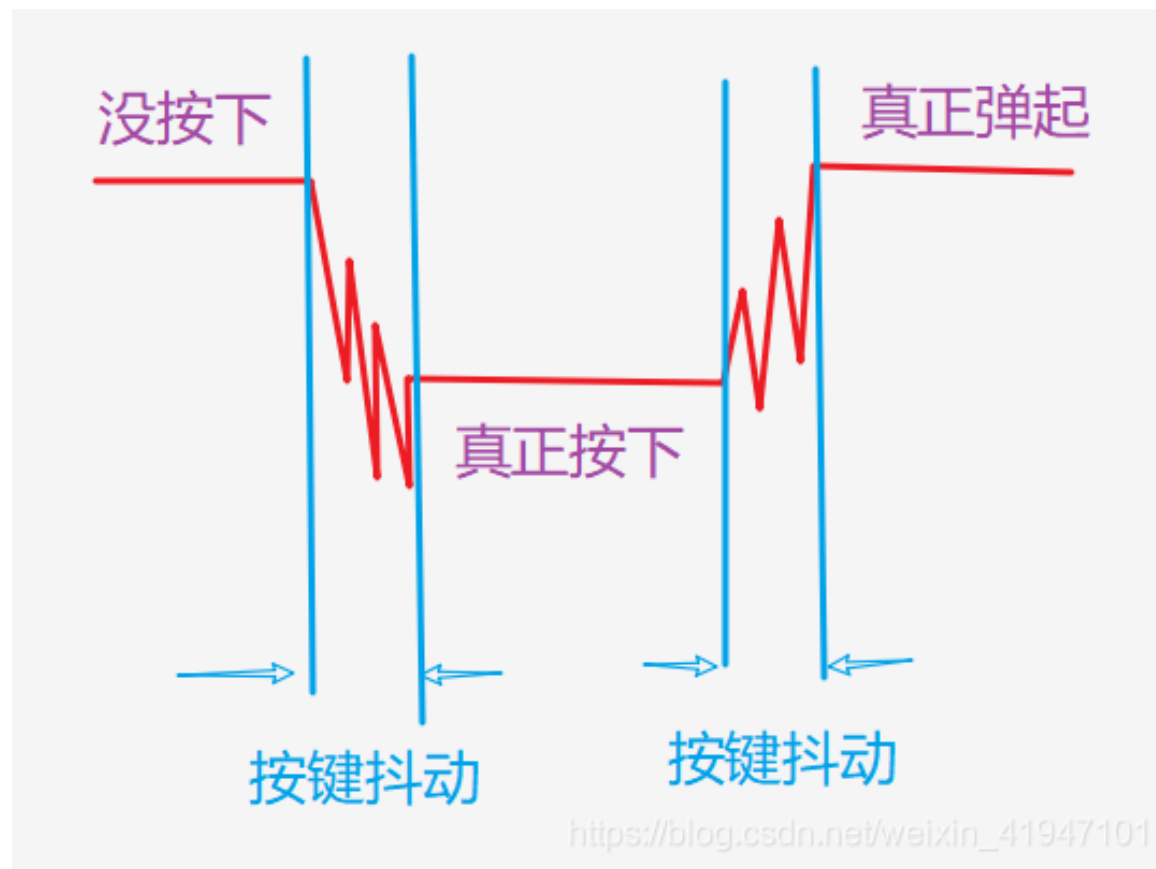
Configure debounce for switches

```
main.c  core_cm0plus.h  ht32_board_config.h  ht32f5xxxx_01.h  ht32f5xxxx_01.h
115      */
116      EXTI_InitTypeDef EXTI_InitStructure;
117      EXTI_InitStructure.EXTI_Channel = HTCFG_EXTI_CHANNEL;
118      EXTI_InitStructure.EXTI_Debounce = EXTI_DEBOUNCE_DISABLE;
119      EXTI_InitStructure.EXTI_DebounceCnt = 0;
120      EXTI_InitStructure.EXTI_IntType = EXTI_POSITIVE_EDGE;
121      EXTI_Init(&EXTI_InitStructure);
122  }
```

➤ 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) <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>	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
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																							
[15:0]	DBnCNT	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.																								

What is debounce?



Configure trigger source type

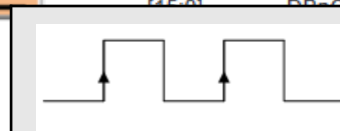
```
115  /*
116  EXTI_InitTypeDef EXTI_InitStructure;
117  EXTI_InitStructure.EXTI_Channel = HTCFG_EXTI_CHANNEL;
118  EXTI_InitStructure.EXTI_Debounce = EXTI_DEBOUNCE_DISABLE;
119  EXTI_InitStructure.EXTI_DebounceCnt = 0;
120  EXTI_InitStructure.EXTI_IntType = EXTI_POSITIVE_EDGE;
121  EXTI_Init(&EXTI_InitStructure);
122  }
```



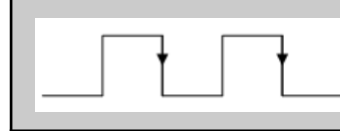
```
132  /* Definitions of EXTI init structure
133  typedef enum
134  {
135  136  EXTI_LOW_LEVEL      = 0x0,
137  EXTI_HIGH_LEVEL     = 0x1,
138  EXTI_NEGATIVE_EDGE  = 0x2,
139  EXTI_POSITIVE_EDGE  = 0x3,
140  EXTI_BOTH_EDGE      = 0x4
141  } EXTI_Interrupt_TypeDef;
142  }
```

➤ UserManual p.189

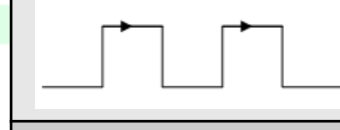
Bits	Field	Descriptions																								
[31]	DBnEN	EXTIn De-bounce Circuit Enable Bit (n = 0 ~ 15) 0: De-bounce circuit is disabled 1: De-bounce circuit is enabled																								
[30:28]	SRCnTYPE	EXTIn Interrupt Source Trigger Type (n = 0 ~ 15) <table><tr><th colspan="3">SRCnTYPE [2:0]</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>	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
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																							



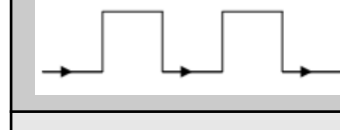
Rising edge trigger: Used to detect a clean rising edge without any bouncing.



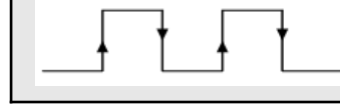
Falling edge trigger: Used to detect a clean falling edge without any bouncing.



High-level trigger: Used to detect a high-level state.



Low-level trigger: Used to detect a low-level state.

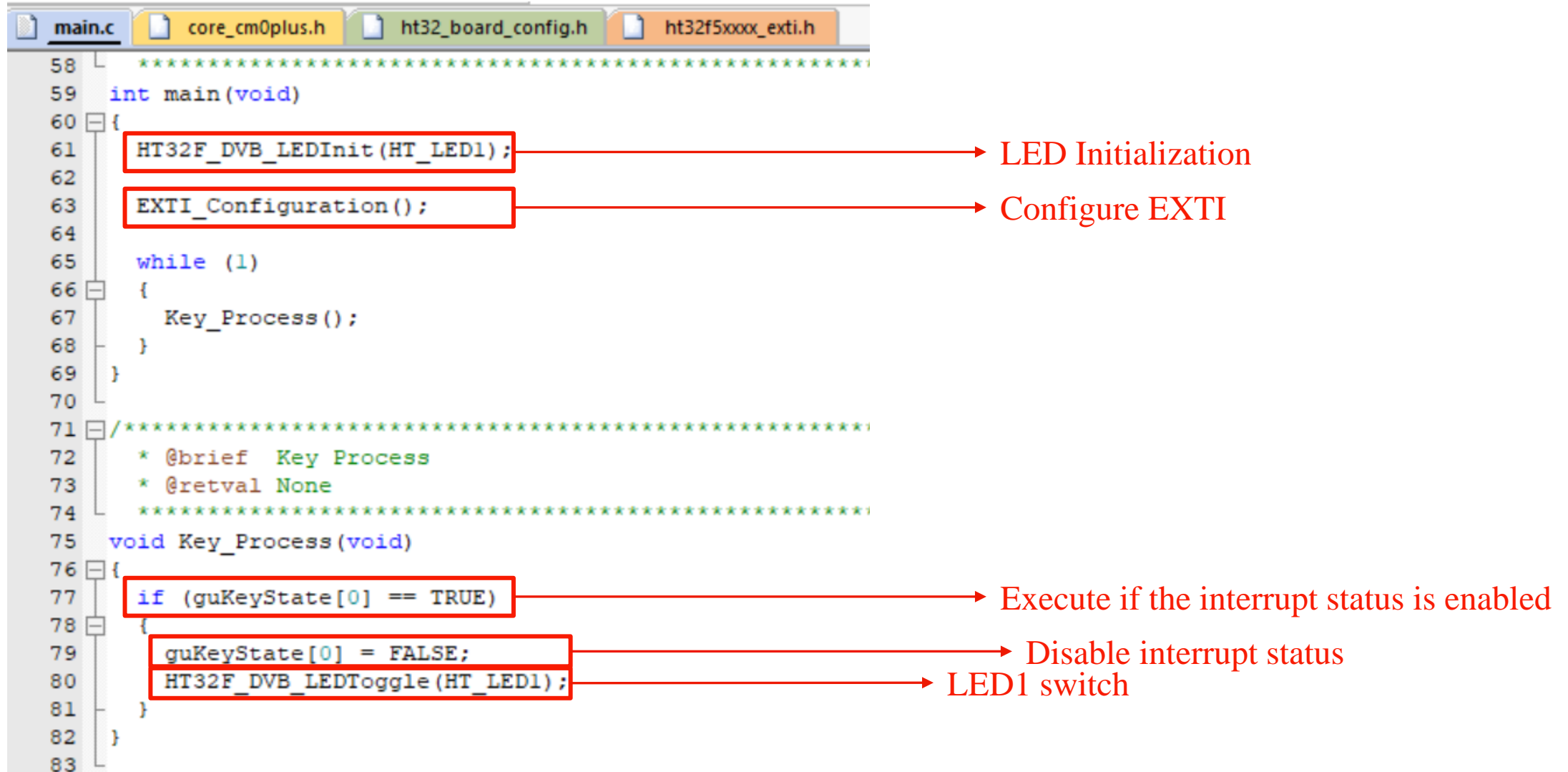


Dual-edge trigger: Used to detect non-bouncing dual edges.

The background features a series of concentric circles in light gray, some solid and some dashed, creating a ripple effect. A large red speech bubble is centered on the page, with a small tail pointing downwards.

EXTI Project

main.c



The image shows a code editor window with four tabs: `main.c`, `core_cm0plus.h`, `ht32_board_config.h`, and `ht32f5xxxx_exti.h`. The `main.c` file is open, showing lines 58 to 83. The code is as follows:

```
58  //*****
59  int main(void)
60  {
61      HT32F_DVB_LEDInit (HT_LED1);
62
63      EXTI_Configuration();
64
65      while (1)
66      {
67          Key_Process();
68      }
69  }
70
71  //*****
72  * @brief  Key Process
73  * @retval None
74  //*****
75  void Key_Process(void)
76  {
77      if (guKeyState[0] == TRUE)
78      {
79          guKeyState[0] = FALSE;
80          HT32F_DVB_LEDToggle (HT_LED1);
81      }
82  }
83
```

Annotations with red arrows point to specific lines of code:

- Line 61: `HT32F_DVB_LEDInit (HT_LED1);` → LED Initialization
- Line 63: `EXTI_Configuration();` → Configure EXTI
- Line 77: `if (guKeyState[0] == TRUE)` → Execute if the interrupt status is enabled
- Line 79: `guKeyState[0] = FALSE;` → Disable interrupt status
- Line 80: `HT32F_DVB_LEDToggle (HT_LED1);` → LED1 switch

```

88 void EXTI_Configuration(void)
89 {
90     /* Enable peripheral clock */
91     CKCU_PeripClockConfig_TypeDef CKCUClock = {{ 0 }};
92     CKCUClock.Bit.AFIO = 1;
93     CKCUClock.Bit.EXTI = 1;
94     CKCUClock.Bit.PB   = 1;
95     CKCU_PeripClockConfig(CKCUClock, ENABLE);
96 }
97
98 /* Configure AFIO mode of input pins */
99 AFIO_GPxConfig(HTCFG_WAKE_GPIO_ID, HTCFG_WAKE_AFIO_PIN, AFIO_FUN_GPIO);
100
101 /* Enable GPIO Input Function */
102 GPIO_InputConfig(HTCFG_WAKE_GPIO_PORT, HTCFG_WAKE_GPIO_PIN, ENABLE);
103
104 /* Configure GPIO pull resistor of input pins */
105 GPIO_PullResistorConfig(HTCFG_WAKE_GPIO_PORT, HTCFG_WAKE_GPIO_PIN, GPIO_PR_DISABLE);
106
107 /* Select Port as EXTI Trigger Source */
108 AFIO_EXTISourceConfig(HTCFG_WAKE_GPION, HTCFG_WAKE_GPIO_ID);
109
110 /* Configure EXTI Channel n as rising edge trigger */
111
112 /* !!! NOTICE !!!
113    Notice that the local variable (structure) did not have an initial value.
114    Please confirm that there are no missing members in the parameter settings below in this function.
115 */
116 EXTI_InitTypeDef EXTI_InitStruct;
117 EXTI_InitStruct.EXTI_Channel = HTCFG_EXTI_CHANNEL;
118 EXTI_InitStruct.EXTI_Debounce = EXTI_DEBOUNCE_DISABLE;
119 EXTI_InitStruct.EXTI_DebounceCnt = 0;
120 EXTI_InitStruct.EXTI_IntType = EXTI_POSITIVE_EDGE;
121 EXTI_Init(&EXTI_InitStruct);
122 }
123
124 /* Enable EXTI & NVIC line Interrupt */
125 EXTI_IntConfig(HTCFG_EXTI_CHANNEL, ENABLE);
126 NVIC_EnableIRQ(HTCFG_WAKE_EXTI_IRQn);
127 }
128

```

Configure system CLOCK

Configure AFIO

Input configuration

Configure resistor

Configure the pin for EXTI source

EXTI register

Enable interrupt

ht32f5xxx_01_it.c

```
134 void EXTI4_15_IRQHandler(void)
135 {
136     #if 1
137     if (EXTI_GetEdgeFlag(HTCFG_EXTI_CHANNEL))
138     {
139         EXTI_ClearEdgeFlag(HTCFG_EXTI_CHANNEL);
140         guKeyState[0] = TRUE;
141     }
142     #else
143     if (EXTI_GetEdgeStatus(HTCFG_EXTI_CHANNEL, EXTI_EDGE_NEGATIVE))
144     {
145         EXTI_ClearEdgeFlag(HTCFG_EXTI_CHANNEL);
146     }
147     else if (EXTI_GetEdgeStatus(HTCFG_EXTI_CHANNEL, EXTI_EDGE_POSITIVE))
148     {
149         EXTI_ClearEdgeFlag(HTCFG_EXTI_CHANNEL);
150         guKeyState[0] = TRUE;
151     }
152     #endif
153 }
```

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](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_InitStructure;  
EXTI_InitStructure.EXTI_Channel = HTCFG_EXTI_CHANNEL;  
EXTI_InitStructure.EXTI_Debounce = EXTI_DEBOUNCE_ENABLE;  
EXTI_InitStructure.EXTI_DebounceCnt = 60000;  
EXTI_InitStructure.EXTI_IntType = EXTI_POSITIVE_EDGE;  
EXTI_Init(&EXTI_InitStructure);
```

```
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.



The background features a series of concentric circles in light gray, some solid and some dashed, creating a ripple effect. A large, solid red speech bubble is centered on the page, pointing downwards.

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, **A11** to JP24-5.
 2. EXTI channel 10, **A10** to JP24-6

☆ PS. Please record.

```
59 vu32 guKeyState[3] = {FALSE, FALSE, FALSE};
```

```
63 int main(void)
```

```
64 {
65     RETARGET_Configuration();
66     HT32F_DVB_LEDInit(HT_LED1);
67     HT32F_DVB_LEDInit(HT_LED2);
68     HT32F_DVB_LEDInit(HT_LED3);
69     printf("Configured on-board LED\r\n");
70
71     EXTI_Configuration1();
72     EXTI_Configuration2();
73     EXTI_Configuration3();
74     printf("Configured EXTI\r\n");
75
76     while (1)
77     {
78         Key_Process();
79     }
80 }
```

1. main.c
(EXTI_Configuration+Key_Process)

```
115 extern vu32 guKeyState[3]; 2. ht32f5xxxx_01_it.c
```

```
134 void EXTI4_15_IRQHandler(void)
```

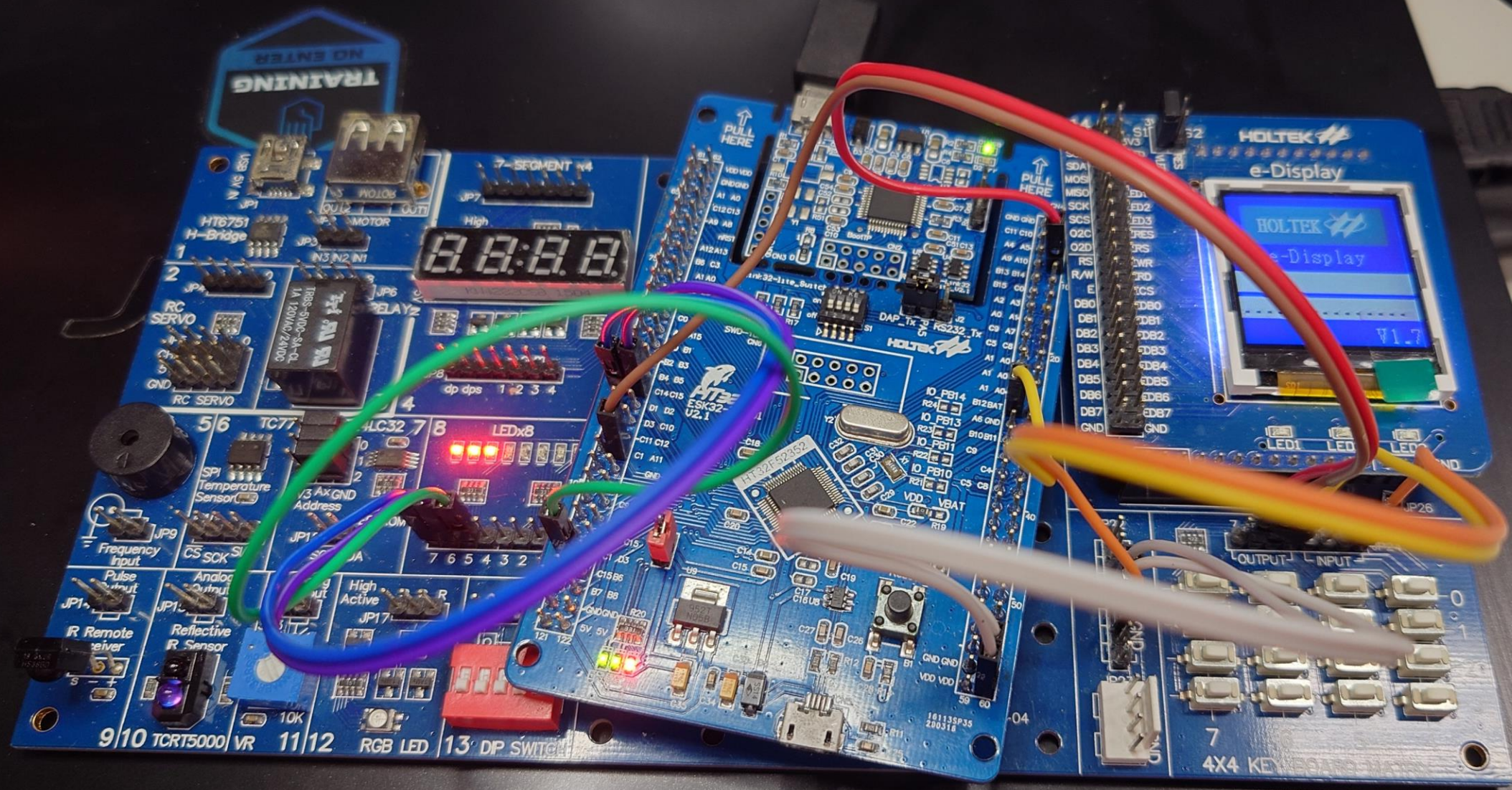
```
135 {
136     #if 1
137     if (EXTI_GetEdgeFlag(EXTI_CONF1_EXTI_CHANNEL))
138     {
139         EXTI_ClearEdgeFlag(EXTI_CONF1_EXTI_CHANNEL);
140         guKeyState[0] = TRUE;
141     }
142     if (EXTI_GetEdgeFlag(EXTI_CONF2_EXTI_CHANNEL))
143     {
144         EXTI_ClearEdgeFlag(EXTI_CONF2_EXTI_CHANNEL);
145         guKeyState[1] = TRUE;
146     }
147     if (EXTI_GetEdgeFlag(EXTI_CONF3_EXTI_CHANNEL))
148     {
149         EXTI_ClearEdgeFlag(EXTI_CONF3_EXTI_CHANNEL);
150         guKeyState[2] = TRUE;
151     }
152 }
```

```
57 #define EXTI_CONF1_PORT
58 #define EXTI_CONF1_GPIO_N
59
60 #define EXTI_CONF2_PORT
61 // #define EXTI_CONF2_PORT
62 #define EXTI_CONF2_GPIO_N
63
64 #define EXTI_CONF3_PORT
65 // #define EXTI_CONF3_PORT
66 #define EXTI_CONF3_GPIO_N
67
68 #define EXTI_CONF1_GPIO_CK
69 #define EXTI_CONF1_GPIO_ID
70 #define EXTI_CONF1_GPIO_PORT
71 #define EXTI_CONF1_AFIO_PIN
72 #define EXTI_CONF1_GPIO_PIN
73 #define EXTI_CONF1_EXTI_CHANNEL
74 #define EXTI_CONF1_EXTI_IRQn
75
76 #define EXTI_CONF2_GPIO_CK
77 #define EXTI_CONF2_GPIO_ID
78 #define EXTI_CONF2_GPIO_PORT
79 #define EXTI_CONF2_AFIO_PIN
80 #define EXTI_CONF2_GPIO_PIN
81 #define EXTI_CONF2_EXTI_CHANNEL
82 #define EXTI_CONF2_EXTI_IRQn
83
84 #define EXTI_CONF3_GPIO_CK
85 #define EXTI_CONF3_GPIO_ID
86 #define EXTI_CONF3_GPIO_PORT
87 #define EXTI_CONF3_AFIO_PIN
88 #define EXTI_CONF3_GPIO_PIN
89 #define EXTI_CONF3_EXTI_CHANNEL
90 #define EXTI_CONF3_EXTI_IRQn
```

```
B
12
A
C
11
A
C
10
STRCAT2(P, EXTI_CONF1_PORT)
STRCAT2(GPIO_P, EXTI_CONF1_PORT)
STRCAT2(HT_GPIO, EXTI_CONF1_PORT)
STRCAT2(AFIO_PIN_, EXTI_CONF1_GPIO_N)
STRCAT2(GPIO_PIN_, EXTI_CONF1_GPIO_N)
STRCAT2(EXTI_CHANNEL_, EXTI_CONF1_GPIO_N)
STRCAT3(EXTI, EXTI_CONF1_GPIO_N, _IRQn)

STRCAT2(P, EXTI_CONF2_PORT)
STRCAT2(GPIO_P, EXTI_CONF2_PORT)
STRCAT2(HT_GPIO, EXTI_CONF2_PORT)
STRCAT2(AFIO_PIN_, EXTI_CONF2_GPIO_N)
STRCAT2(GPIO_PIN_, EXTI_CONF2_GPIO_N)
STRCAT2(EXTI_CHANNEL_, EXTI_CONF2_GPIO_N)
STRCAT3(EXTI, EXTI_CONF2_GPIO_N, _IRQn)

STRCAT2(P, EXTI_CONF3_PORT)
STRCAT2(GPIO_P, EXTI_CONF3_PORT)
STRCAT2(HT_GPIO, EXTI_CONF3_PORT)
STRCAT2(AFIO_PIN_, EXTI_CONF3_GPIO_N)
STRCAT2(GPIO_PIN_, EXTI_CONF3_GPIO_N)
STRCAT2(EXTI_CHANNEL_, EXTI_CONF3_GPIO_N)
STRCAT3(EXTI, EXTI_CONF3_GPIO_N, _IRQn)
```

The background features a series of concentric circles in light gray, some solid and some dashed, creating a ripple effect. A large red speech bubble is centered on the page, containing the text.

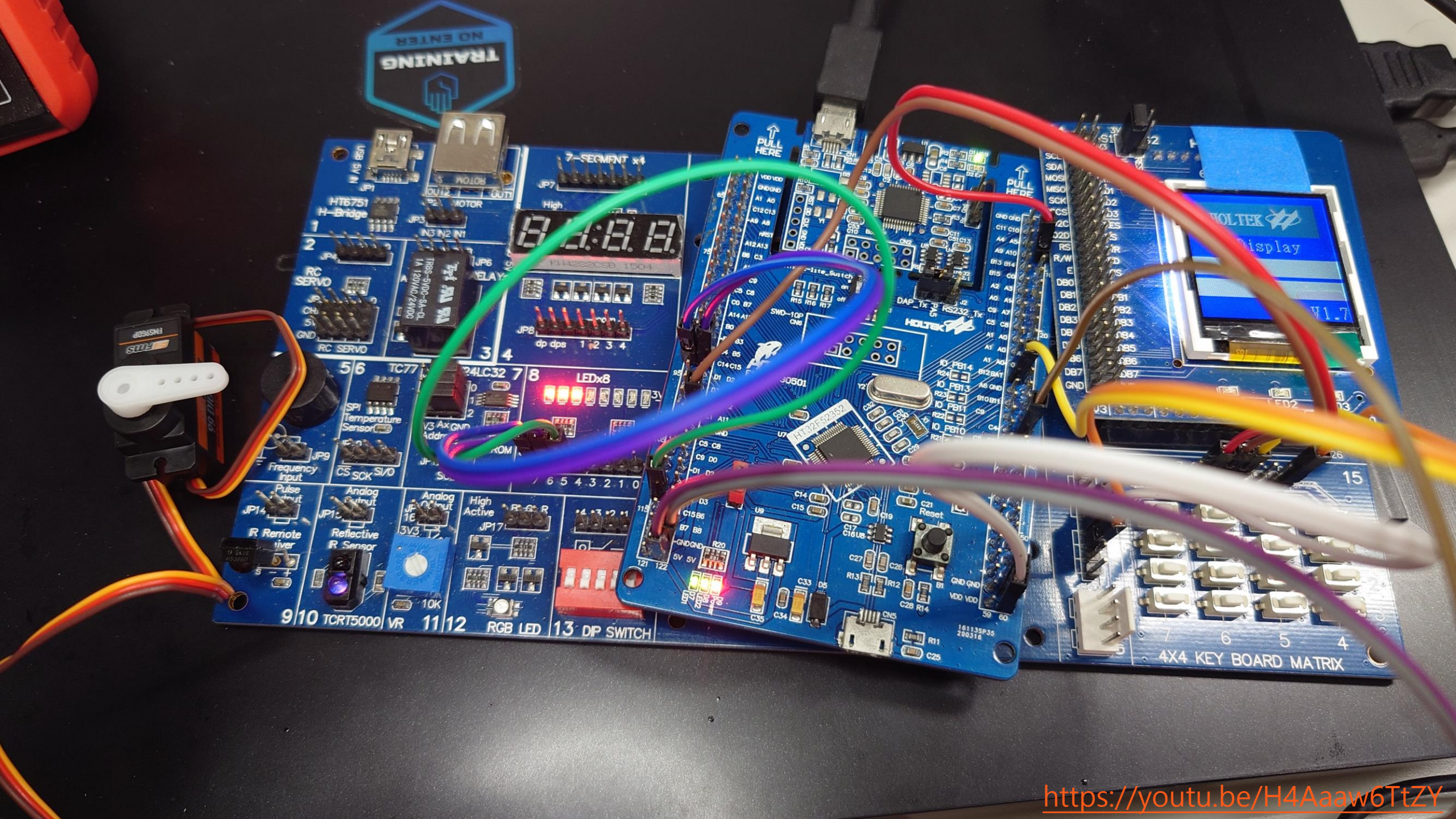
Homework W9-3.

(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_CH0.**

☆ PS. Please record and explain the code.



A red speech bubble with a white outline and a small tail pointing downwards. Inside the bubble, the text "Class Dismissed" is written in white, sans-serif font. The background of the slide is white with faint, curved, grey lines in the top-left and bottom-right corners.

Class
Dismissed