LAB: EXTI & SysTick

Date: 2023.10.16

Author: Hanmin Kim

Github: https://github.com/ansterdaz/HKim

Demo video

EXTI: https://youtube.com/shorts/dJO2sDttWJM

Systick: https://youtube.com/shorts/8Z27eyVI_fk

Introduction

In this lab, you are required to create two simple programs using interrupt:

(1) displaying the number counting from 0 to 9 with Button Press

(2) counting at a rate of 1 second

Requirement

Hardware

MCU

- NUCLEO-F411RE

Actuator/Sensor/Others:

- 7-segment display(5101ASR)
- Array resistor (330 ohm)
- decoder chip(74LS47)
- breadboard

Software

Keil uVision, CMSIS, EC_HAL library

Problem 1: Counting numbers on 7-Segment using EXTI Button

Procedure

1-1. Creat HAL library

1. Download sample header files: **ecEXTI_student.h, ecEXTI_student.c**
2. Rename these files as **ecEXTI.h, ecEXTI.c**

   Save these files in directory `EC \lib\`.
3. Declare and define the following functions in library : **ecEXTI.h**

ecEXTI.h

1-2. Procedure

1. 1. Create a new project under the directory `\EC\LAB\LAB_EXTI`

- The project name is "**LAB_EXTI".**
- Create a new source file named as "**LAB_EXTI.c"**

2. Include your updated library in `\EC\lib\` to your project.

- **ecGPIO.h, ecGPIO.c**
- **ecRCC.h, ecRCC.c**
- **ecEXTI.h, ecEXTI.c**

3. Use the decoder chip (**74LS47**). Connect it to the bread board and 7-segment display.
4. First, check if every number, 0 to 9, can be displayed properly on the 7-segment.
5. Then, create a code to display the number counting from 0 to 9 and repeats

- by pressing the push button. (External Interrupt)

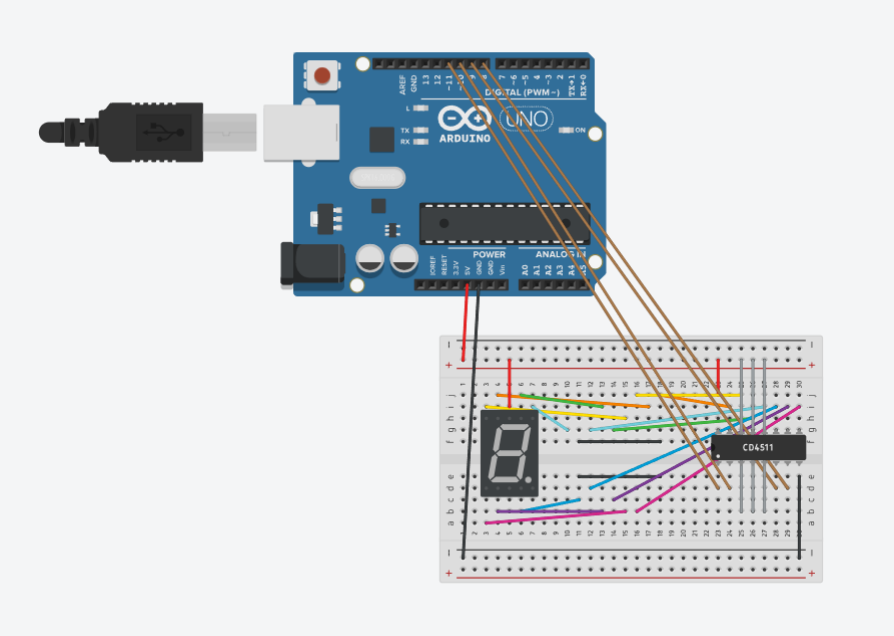
6. You must use your library function of EXTI.

Configuration

Digital In for Button (B1)	Digital Out for 7-Segment decoder
Digital In	Digital Out
PC13	PA7, PB6, PC7, PA9
PULL-UP	Push-Pull, No Pull-up-Pull-down, Medium Speed

Connection Diagram

circuit diagram



Discussion

1. We can use two different methods to detect an external signal: polling and interrupt. What are the advantages and disadvantages of each approach?

   Poling

Advantage:

Simple to implement and easy to understand. It is suitable for use in simple systems.

Disadvantage:

Continuous polling increases CPU usage. Processing time may be wasted because external signals must be continuously checked.

Interrupt

Advantage:

If used when external signals are irregular, it can consume less CPU power.

Disadvantage:

Implementing interrupts can be more complex than polling.

2. What would happen if the EXTI interrupt handler does not clear the interrupt pending flag? Check with your code

   If the pending flag is not cleared, the system will not operate as desired by the designer because the interrupt is not properly released after execution. System crashes or latency issues may also occur.

Code

EXTI.h

```
void EXTI_init(GPIO_TypeDef *Port, int pin, int trig, int priority);
void EXTI_enable(uint32_t pin);
void EXTI_disable(uint32_t pin);
uint32_t is_pending_EXTI(uint32_t pin);
void clear_pending_EXTI(uint32_t pin);

void EXTI_init(GPIO_TypeDef *port, int pin, int trig_type, int priority);
void EXTI_enable(uint32_t pin);  // mask in IMR
void EXTI_disable(uint32_t pin);  // unmask in IMR
uint32_t  is_pending_EXTI(uint32_t pin);
void clear_pending_EXTI(uint32_t pin);
```

EXTI.c

```
void EXTI_init(GPIO_TypeDef *Port, int Pin, int trig_type,int priority){

  // SYSCFG peripheral clock enable
  RCC->APB2ENR |= RCC_APB2ENR_SYSCFGEN;
  // Connect External Line to the GPIO
  int EXTICR_port;
  if      (Port == GPIOA) EXTICR_port = 0;
  else if (Port == GPIOB) EXTICR_port = 1;
  else if (Port == GPIOC) EXTICR_port = 2;
  else if (Port == GPIOD) EXTICR_port = 3;
  else                    EXTICR_port = 4;

  SYSCFG->EXTICR[Pin/4] &= ~15<<4*(Pin & 0x03);             // clear
  SYSCFG->EXTICR[Pin/4] |= EXTICR_port<<4*(Pin &0x03);      // set port number

  // Configure Trigger edge
  if (trig_type == FALL) EXTI->FTSR |= 1<<Pin;   // Falling trigger enable
  else if (trig_type == RISE) EXTI->RTSR |= 1<<Pin;   // Rising trigger enable
  else if (trig_type == BOTH) {     // Both falling/rising trigger enable
    EXTI->RTSR |= 1<<Pin;
    EXTI->FTSR |= 1<<Pin;
  }

  // Configure Interrupt Mask (Interrupt enabled)
  EXTI->IMR  |= 1<<Pin;     // not masked


  // NVIC(IRQ) Setting
  int EXTI_IRQn = 0;

  if (Pin < 5)  EXTI_IRQn = EXTI0_IRQn+Pin;
  else if (Pin < 10)  EXTI_IRQn = EXTI9_5_IRQn;
  else      EXTI_IRQn = EXTI15_10_IRQn;

  NVIC_SetPriority(EXTI_IRQn, 0); // EXTI priority
  NVIC_EnableIRQ(EXTI_IRQn);  // EXTI IRQ enable
}
```

The EXTI init was designed as follows.

Main code

```
void setup(void)
{
  RCC_PLL_init();
  sevensegment_display_init();
  // Priority Highest(0) External Interrupt
  EXTI_init(GPIOC, BUTTON_PIN, FALL, 0);
}

int main(void) {
  setup();
  while (1) {}
}



//EXTI for Pin 13
void EXTI15_10_IRQHandler(void) {
  if (is_pending_EXTI(BUTTON_PIN)) {

    sevensegment_display(cnt % 10);
    cnt++;
    if (cnt > 9) cnt = 0;
    for(int i = 0; i < 500000;i++){}
    clear_pending_EXTI(BUTTON_PIN);
  }
}
```

When the button pin is pressed, an interrupt is activated by the IRQHandler to operate the sevensegment display.

Results

Each time the button is pressed, the number changes and outputs numbers from 0 to 9.

video link: https://youtube.com/shorts/dJO2sDttWJM

Problem 2: Counting numbers on 7- Segment using SysTick

Display the number 0 to 9 on the 7-segment LED at the rate of 1 sec. After displaying up to 9, then it should display '0' and continue counting.

When the button is pressed, the number should be reset '0' and start counting again.

Procedure

2-1. Creat HAL library

1. Download sample header files**ecSysTick_student.h, ecSysTick_student.c**
2. Rename these files as **ecSysTick.h, ecSysTick.c**
    - You MUST write your name and other information at the top of the library code files.
    - Save these files in directory `EC \lib\`.
3. Declare and define the following functions in library : **ecSysTick.h**

ecSysTick.h

2-2. Procedure

1. Create a new project under the directory

   `\EC\LAB\LAB_EXTI_SysTick`

- The project name is "**LAB_EXTI_SysTick".**
- Create a new source file named as "**LAB_EXTI_SysTick.c"**

2. Include your updated library in `\EC\lib\` to your project.

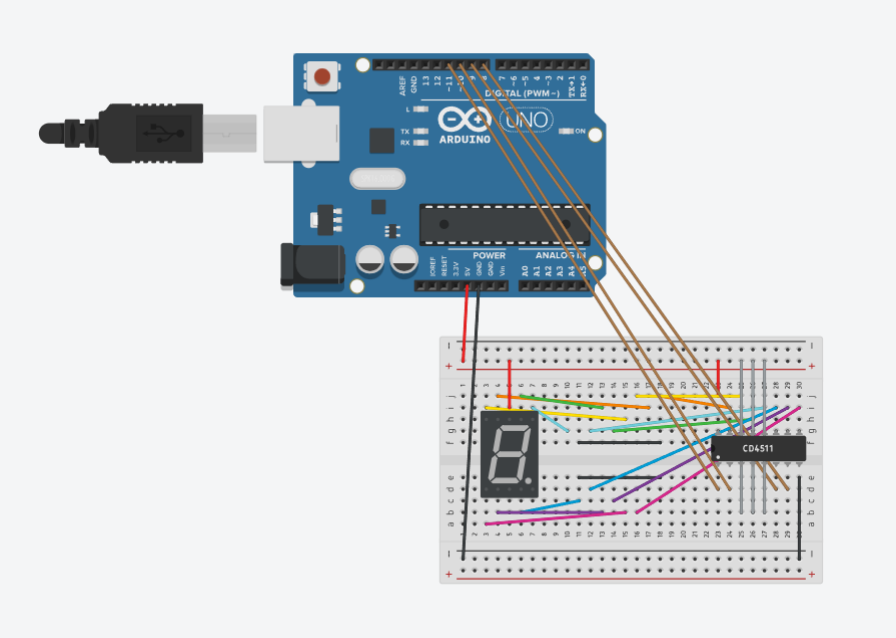
- **ecGPIO.h, ecGPIO.c**
- **ecRCC.h, ecRCC.c**
- **ecEXTI.h, ecEXTI.c**
- **ecSysTick.h, ecSysTick.c**

3. Use the decoder chip (**74LS47**). Connect it to the bread board and 7-segment display.
4. First, check if every number, 0 to 9, can be displayed properly on the 7-segment.
5. Then, create a code to display the number counting from 0 to 9 and repeats at the rate of 1 second.
6. When the button is pressed, it should start from '0' again.

Configuration

Digital In for Button (B1)	Digital Out for 7-Segment decoder
Digital In	Digital Out
PC13	PA7, PB6, PC7, PA9
PULL-UP	Push-Pull, No Pull-up-Pull-down, Medium Speed

Circuit Diagram



Code

SysTick.h

```
extern volatile uint32_t msTicks;
void SysTick_init(void);
void SysTick_Handler(void);
void SysTick_counter();
void delay_ms(uint32_t msec);
void SysTick_reset(void);
uint32_t SysTick_val(void);
```

SysTick.c

```
void SysTick_init(void){
  //  SysTick Control and Status Register
  SysTick->CTRL = 0;                        // Disable SysTick IRQ and SysTick Counter

  // Select processor clock
  // 1 = processor clock;  0 = external clock
  SysTick->CTRL |= SysTick_CTRL_CLKSOURCE_Msk;

  // uint32_t MCU_CLK=EC_SYSTEM_CLK
  // SysTick Reload Value Register
  SysTick->LOAD = MCU_CLK_PLL / 1000 - 1;           // 1ms, for HSI PLL = 84MHz.

  // SysTick Current Value Register
  SysTick->VAL = 0;

  // Enables SysTick exception request
  // 1 = counting down to zero asserts the SysTick exception request
  SysTick->CTRL |= SysTick_CTRL_TICKINT_Msk;

  // Enable SysTick IRQ and SysTick Timer
  SysTick->CTRL |= SysTick_CTRL_ENABLE_Msk;

  NVIC_SetPriority(SysTick_IRQn, 16);    // Set Priority to 1
  NVIC_EnableIRQ(SysTick_IRQn);     // Enable interrupt in NVIC
}
```

The Systick init was designed as follows.

Main code

```
int count = 0;
// Initialiization
void setup(void)
{
  EXTI_init(GPIOC, BUTTON_PIN, FALL, 0);
  RCC_PLL_init();
  SysTick_init();
  sevensegment_init();
  sevensegment_display_init();

}
// interrupt
void EXTI15_10_IRQHandler(void) {
  if (is_pending_EXTI(BUTTON_PIN)) {
     sevensegment_display(0);   // push button to print number 0
     count = 0;
    clear_pending_EXTI(BUTTON_PIN);
  }
}


int main(void) {
  // Initialiization --------------------------------------------------------
    setup();

  // Inifinite Loop ----------------------------------------------------------
  while(1){
    sevensegment_display(count);
    delay_ms(1000);
    count++;
    if (count >9) count =0;
    SysTick_reset();
  }
}
```

It automatically outputs numbers from 1 to 9, and when an interrupt is activated by the IRQHandler by pressing the button, the display is initialized with count = 0.

Results

When starting by pressing the reset button, the 7 segment display automatically outputs numbers from 0 to 9. If pressing the button at this time, the numbers start again from 0.

video link: https://youtube.com/shorts/8Z27eyVI_fk

Reference

https://ykkim.gitbook.io/ec/ec-course/lab/lab-gpio-digital-inout-7segment

https://ykkim.gitbook.io/ec/ec-course/lab/lab-exti-and-systick

74LS47 Data sheet, NUCLEO-F411RE Data sheet