PreLAB: External Interrupt

Name: Dongjun, Lee

ID: 22201042

I. Introduction

In this tutorial, we will learn how to use External Interrupt. We will create functions that capture the falling edge trigger by pushing a button using an external interrupt.

The objectives of this tutorial are how to

- Configure External input (EXTI) interrupt with NVIC
- Create your own functions for configuration of interrupts

Hardware

- NUCLEO -F411RE

Software

- VS code, CMSIS, EC_HAL

Documentation

- STM32 Reference Manual

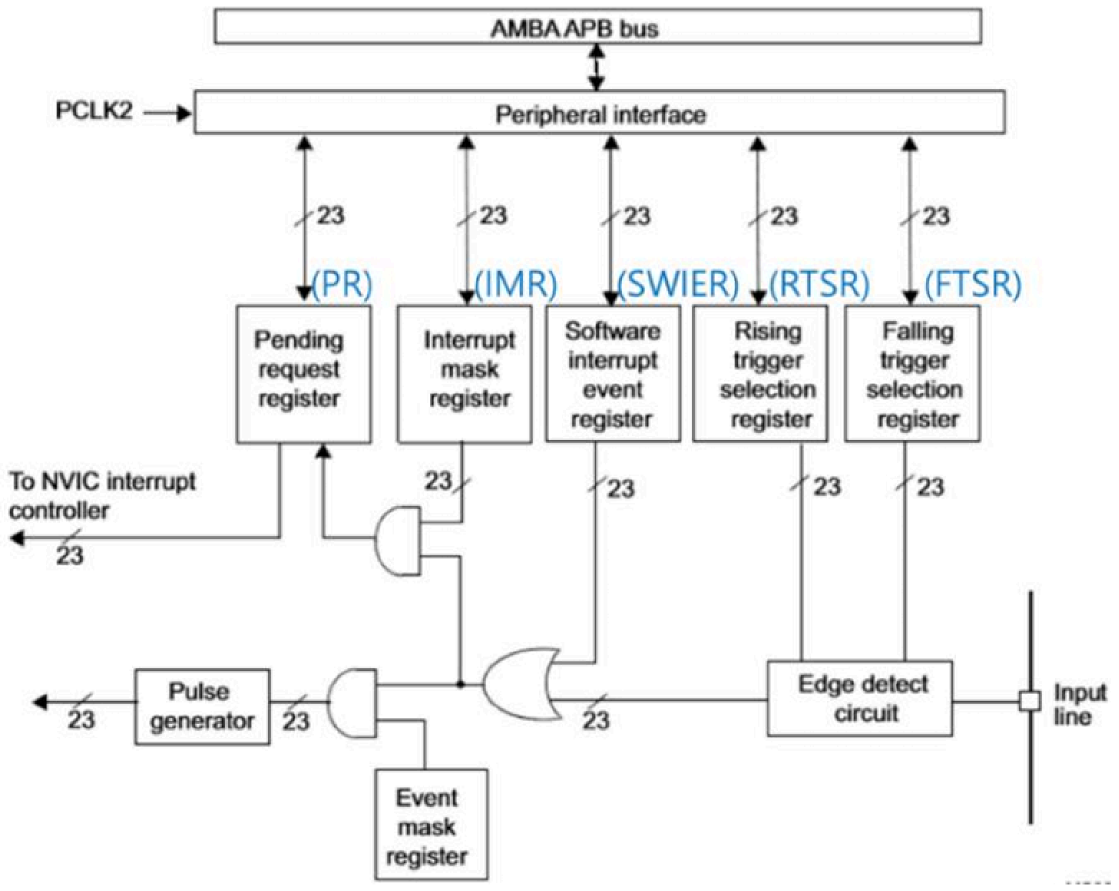
II.Basics of External Interrupt (EXTI)

A. Register List

List of external interrupt (EXTI) registers used in this tutorial [Reference Manual ch7, ch10.2]

Type	Register Name	Description
SYSCFG	SYSCFG_EXTICRx	External Interrupt Configuration, x=1 to 4 EXTICR1: for pin0~pin3 , EXTICR2: for pin4~pin7, etc
EXTI_	EXTI_IMR	Interrupt Mask
	EXTI_FTSR EXTI_RTSR	Falling/Rising Trigger Selection

Schematic



B. Register Setting

(Digital Input Setting)

- Enable GPIO peripheral clock **RCC->AHB1ENR**
- Configure DigitalIn pin

(EXTI Setting)

- Enable SYSCFG peripheral clock. **RCC->APB2ENR**
- Connect the corresponding external line to GPIO **SYSCFG->EXTICR**
- Configure the trigger edge. **EXTI->FTSR/RTSR**
- Configure Interrupt mask **EXTI->IMR**
- Enable EXTI. **EXTI->IMR**

(NVIC Setting)

- Configure the priority of EXTI interrupt request. **NVIC_SetPriority(**)
- Enable EXTI interrupt request. **NVIC_EnableIRQ(**)

(EXTI Use)

- Create user codes in handler **EXTIx_IRQHandler()**
- Clear pending bit after interrupt call

III. Tutorial

A. Register Configuration

Fill in the blanks below

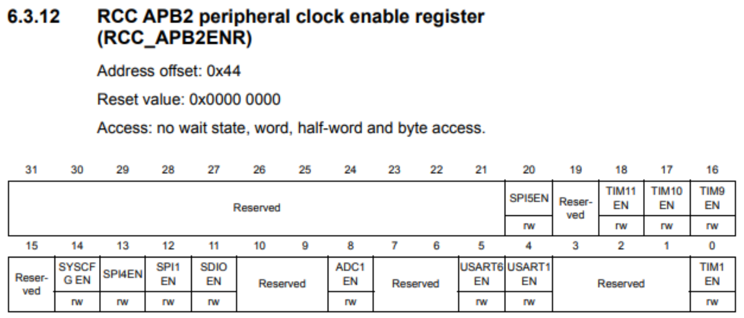
1. **Pin Initialization & Set LED and Push-button**

- LED Pin : Port B Pin 12 / Output / Push-Pull / No Pull-Up & No Pull-Down
- Push-Button: Port A Pin 4 / Input / No Pull-Up & No Pull-Down

```
    GPIO_init(LED_PIN, OUTPUT);
    GPIO_init(BUTTON_PIN, INPUT);
    GPIO_pupd(BUTTON_PIN, nopupd);
    GPIO_pupd(LED_PIN, nopupd);
    GPIO_otype(LED_PIN, pushpull);
```

2. **Enable Peripheral Clock:** SYSCFGEN

- **RCC_APB2ENR:** Enable SYSCFG
- **RCC -> APB2ENR |= 1<<14



6.3.12 RCC APB2 peripheral clock enable register (RCC_APB2ENR)

Address offset: 0x44

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access.

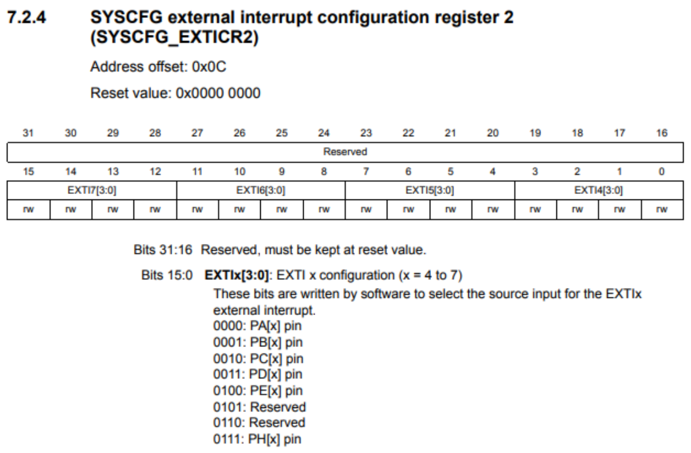
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved											SPI5EN	Reser-ved	TIM11 EN	TIM10 EN	TIM9 EN
											rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reser-ved	SYSCFG EN	SPI4EN	SPI1 EN	SDIO EN	Reserved		ADC1 EN	Reserved		USART6 EN	USART1 EN	Reserved			TIM1 EN
	rw	rw	rw	rw			rw			rw	rw				rw

3. **EXTI Initialization & Connect Push-button to EXTI line**

- **SYSCFG_EXTICR2:** Connect PA_4(push-button) to EXTI4 line

**SYSFG -> EXTICR[1] & = ~15 <<0
SYSFG -> EXTICR[1] | = 0000



7.2.4 SYSCFG external interrupt configuration register 2 (SYSCFG_EXTICR2)

Address offset: 0x0C

Reset value: 0x0000 0000

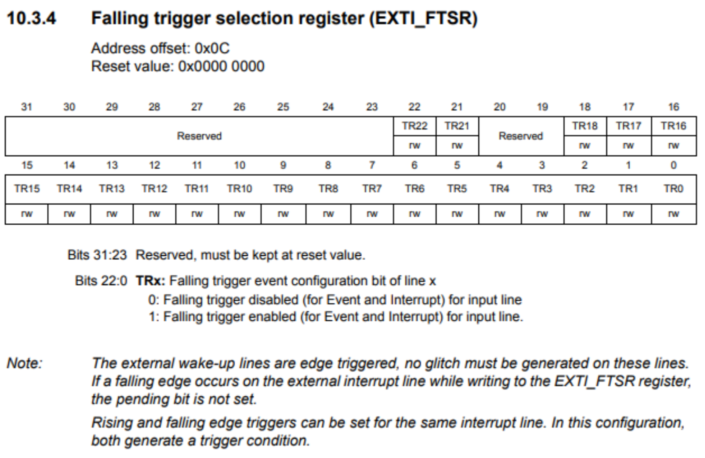
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 4 to 7)
These bits are written by software to select the source input for the EXTIx external interrupt.
0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: Reserved
0110: Reserved
0111: PH[x] pin

- **EXTI_FTSR:** Enable Falling Trigger
  EXTI -> FTSR |= 1<<13// TR4=1



10.3.4 Falling trigger selection register (EXTI_FTSR)

Address offset: 0x0C
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									TR22	TR21	Reserved		TR18	TR17	TR16
									rw	rw			rw	rw	rw

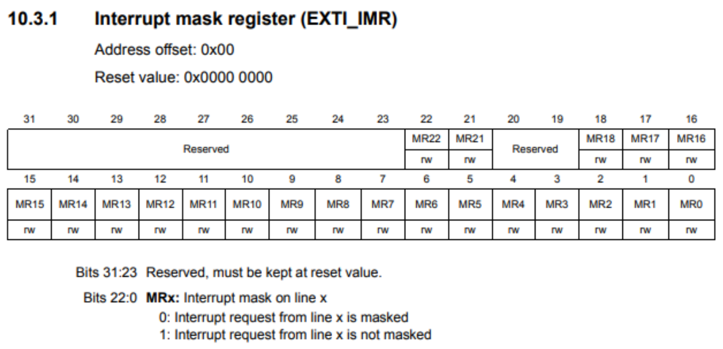
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **TRx:** Falling trigger event configuration bit of line x
0: Falling trigger disabled (for Event and Interrupt) for input line
1: Falling trigger enabled (for Event and Interrupt) for input line.

Note: *The external wake-up lines are edge triggered, no glitch must be generated on these lines. If a falling edge occurs on the external interrupt line while writing to the EXTI_FTSR register, the pending bit is not set.*
Rising and falling edge triggers can be set for the same interrupt line. In this configuration, both generate a trigger condition.

- **EXTI_IMR:** Interrupt NOT masked (Enable)
  EXTI -> IMR |= 1<<4// MR4 = 1



10.3.1 Interrupt mask register (EXTI_IMR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									MR22	MR21	Reserved		MR18	MR17	MR16
									rw	rw			rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **MRx:** Interrupt mask on line x
0: Interrupt request from line x is masked
1: Interrupt request from line x is not masked

B. Programming

This is an example code for toggling LED on/off with the button input trigger (EXTI)

Fill in the empty spaces in the code.

Procedure

- Name the project as `TU_EXTI` by creating a new folder as `tutorial\TU_EXTI`
- Download the template code
  - `TU_EXTI_student.c` Click here to download
- Fill in the empty spaces in the code.
- Run the program and check your result.
- Your tutorial report must be submitted to LMS

{% hint style="info" %}
DO NOT use `ecEXTI2_student.h` for this tutorial.
{% endhint %}

> You MUST write your name on the source file inside the comment section

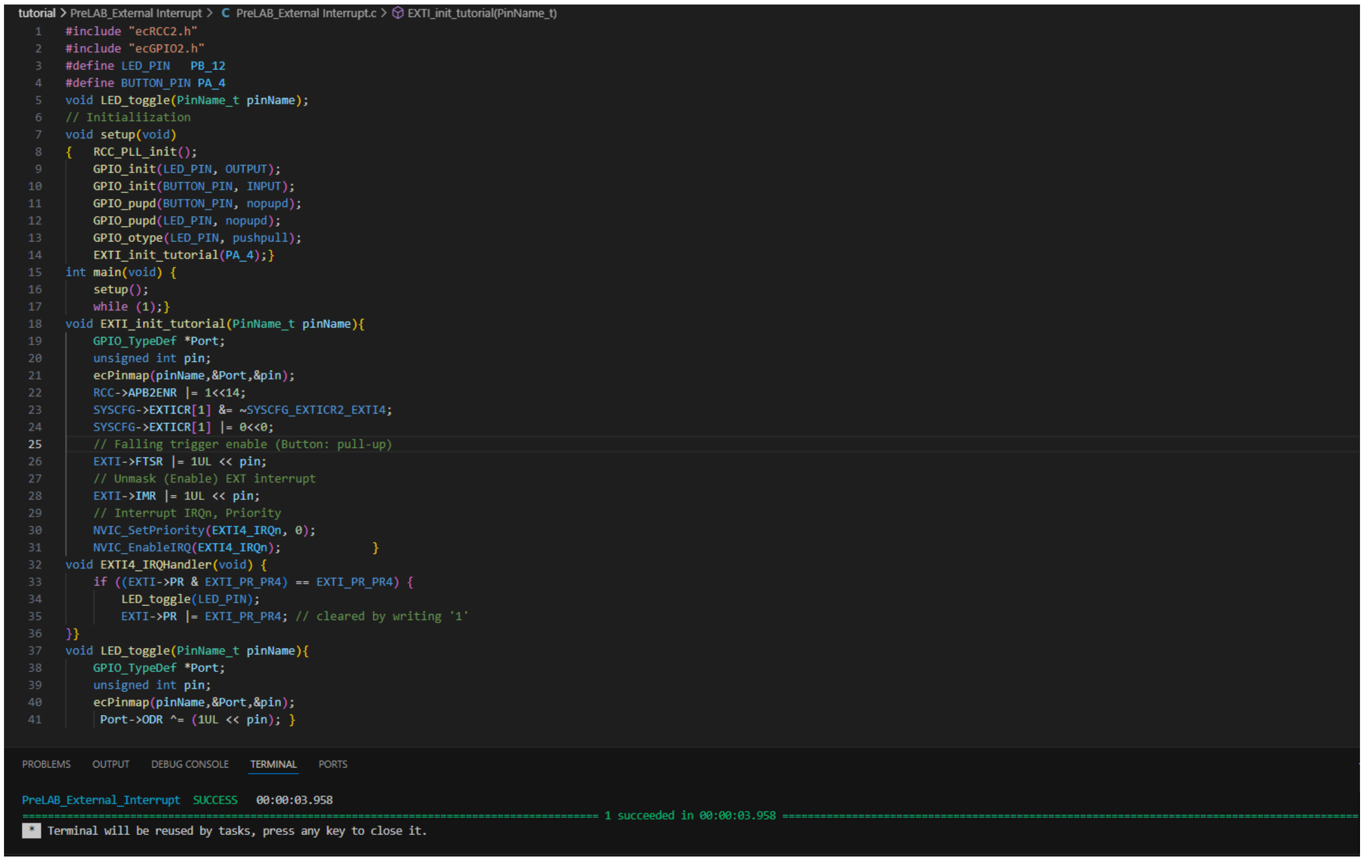
```
//#include "ecSTM32F4v2.h"
#include "ecRCC2.h"
#include "ecGPIO2.h"
#define LED_PIN   PB_12         //EVAL board JKIT
#define BUTTON_PIN PA_4         //EVAL board JKIT
void LED_toggle(PinName_t pinName);
// Initialiization
void setup(void)
{

    RCC_PLL_init();                         // System Clock = 84MHz
    // Initialize GPIOB_12 for Output
    GPIO_init(LED_PIN, OUTPUT);     // LED for EVAL board
    // Initialize GPIOA_4 for Input Button
```

```c
    GPIO_init(BUTTON_PIN, INPUT);  // OUTPUT for EVAL borad
    EXTI_init_tutorial(PA_4);
}
// MAIN  ----------------------------------------
int main(void) {
    setup();
    while (1);
}
// EXTI Initialiization ------------------------------------------------------
// YOUR CODE GOES HERE
void EXTI_init_tutorial(PinName_t pinName){
    GPIO_TypeDef *Port;
    unsigned int pin;
    ecPinmap(pinName,&Port,&pin);
    // SYSCFG peripheral clock enable
    RCC->APB2ENR |= 1<<14;
    // Connect External Line to the GPIO
    // Button: PA_4 -> EXTICR2(EXTI4)
    SYSCFG->EXTICR[1] &= ~SYSCFG_EXTICR2_EXTI4;
    SYSCFG->EXTICR[1] |= 0<<0;
    // Falling trigger enable (Button: pull-up)
    EXTI->FTSR |= 1UL << pin;
    // Unmask (Enable) EXT interrupt
    EXTI->IMR |= 1UL << pin;
    // Interrupt IRQn, Priority
    NVIC_SetPriority(EXTI4_IRQn, 0);        // Set EXTI priority as 0
    NVIC_EnableIRQ(EXTI4_IRQn);             // Enable EXTI
}
// YOUR CODE GOES HERE
void EXTI4_IRQHandler(void) {
    if ((EXTI->PR & EXTI_PR_PR4) == EXTI_PR_PR4) {
        LED_toggle(LED_PIN);
        EXTI->PR |= EXTI_PR_PR4; // cleared by writing '1'
    }
}




void LED_toggle(PinName_t pinName){
    GPIO_TypeDef *Port;
    unsigned int pin;
    ecPinmap(pinName,&Port,&pin);
     Port->ODR ^= (1UL << pin);
}
}
```


tutorial > PreLAB_External Interrupt > C PreLAB_External Interrupt.c > EXTI_init_tutorial(PinName_t)
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS

```c
#include "ecRCC2.h"
#include "ecGPIO2.h"
#define LED_PIN    PB_12
#define BUTTON_PIN PA_4
void LED_toggle(PinName_t pinName);
// Initialiization
void setup(void)
{   RCC_PLL_init();
    GPIO_init(LED_PIN, OUTPUT);
    GPIO_init(BUTTON_PIN, INPUT);
    GPIO_pupd(BUTTON_PIN, nopupd);
    GPIO_pupd(LED_PIN, nopupd);
    GPIO_otype(LED_PIN, pushpull);
    EXTI_init_tutorial(PA_4);}
int main(void) {
    setup();
    while (1);}
void EXTI_init_tutorial(PinName_t pinName){
    GPIO_TypeDef *Port;
    unsigned int pin;
    ecPinmap(pinName,&Port,&pin);
    RCC->APB2ENR |= 1<<14;
    SYSCFG->EXTICR[1] &= ~SYSCFG_EXTICR2_EXTI4;
    SYSCFG->EXTICR[1] |= 0<<0;
    // Falling trigger enable (Button: pull-up)
    EXTI->FTSR |= 1UL << pin;
    // Unmask (Enable) EXT interrupt
    EXTI->IMR |= 1UL << pin;
    // Interrupt IRQn, Priority
    NVIC_SetPriority(EXTI4_IRQn, 0);
    NVIC_EnableIRQ(EXTI4_IRQn);             }
void EXTI4_IRQHandler(void) {
    if ((EXTI->PR & EXTI_PR_PR4) == EXTI_PR_PR4) {
        LED_toggle(LED_PIN);
        EXTI->PR |= EXTI_PR_PR4; // cleared by writing '1'
}}
void LED_toggle(PinName_t pinName){
    GPIO_TypeDef *Port;
    unsigned int pin;
    ecPinmap(pinName,&Port,&pin);
     Port->ODR ^= (1UL << pin); }
```

```
PreLAB_External_Interrupt  SUCCESS   00:00:03.958
=========================================================================================== 1 succeeded in 00:00:03.958 ===========================================================================================
 *  Terminal will be reused by tasks, press any key to close it.
```