임베디드시스템설계
EMBEDED SYSTEM DESIGN

CHAPTER 07

외부 인터럽트(EXTI)를 이용한 입출력 제어

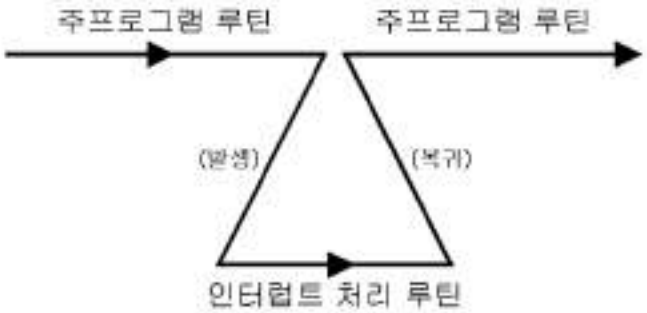
KYUNG HEE UNIVERSITY

Human-Robot Interaction Laboratory

중요. 인터럽트

기본개념

- CPU외부의 하드웨어적인 요구에 의해서 정상적인 프로그램의 실행 순서를 변경하여 보다 시급한 작업을 먼저 수행한 후에 다시 원래의 프로그램으로 복귀하는 것
- 비동기적으로 발생하는 주변장치의 서비스 요청에 CPU가 가장 빠르게 대응할 수 있는 방법
- 비동기적으로 동작하는 CPU(고속)와 주변장치(저속) 사이에서 효율적으로 일을 수행
- 인터럽트가 발생하면 나중에 돌아올 복귀주소(return address)가 자동적으로 스택에 저장되었다가, 인터럽트 서비스루틴의 마지막에서 복귀 명령을 만나면 다시 자동적으로 복귀주소로 돌아온다.



KYUNG HEE UNIVERSITY

중요. 인터럽트

인터럽트의 예

- 타이머에서 지정된 시간 경과
- 입력장치에서의 서비스 요구
- 출력장치의 작업종료
- A/D 변환의 완료
- DMA 동작의 종료
- 멀티프로세서간의 통신 요구 등 마이크로프로세서와 독립되어 있는 외부장치에 의해 발생되는 순수한 의미에서의 인터럽트

KYUNG HEE UNIVERSITY

7.1 외부 인터럽트(EXTI)의 구조 및 기능

외부 인터럽트의(EXTI)의 구조

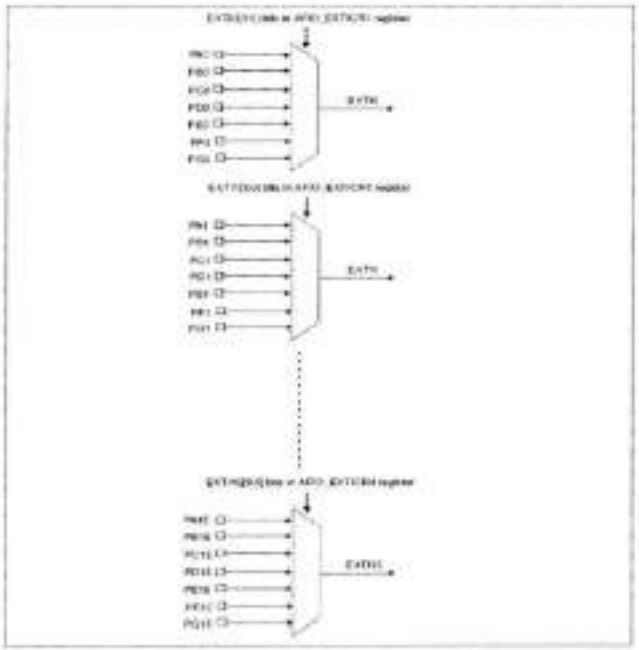
*EXTI : EXTernal Interrupt(외부 인터럽트)

STM32F10x은 총 19개의 외부 인터럽트(EXTI0~EXTI18)을 가지며 그 구성은 다음과 같다

- **EXTI0 ~ EXTI15 : 각각 PORT A ~ PORT G의 0번 핀 ~ 15번 핀에 연결 (F103RB는 PORT C까지)**
- **EXTI16 : PVD 출력에 연결**
- **EXTI17 : RTC(Real Time Clock) 알람에 연결**
- **EXTI18 : USB 웨이크업(Wakeup)에 연결**

▼표 7-1-1 EXTI와 대응되는 인터럽트명

외부인터럽트 번호	연결되는 단자	일생 인터럽트명
EXTI0	PA0 ~ PG0	EXTI0
EXTI1	PA1 ~ PG1	EXTI1
EXTI2	PA2 ~ PG2	EXTI2
EXTI3	PA3 ~ PG3	EXTI3
EXTI4	PA4 ~ PG4	EXTI4
EXTI5 ~ EXTI9	PA5/6/7/8/9 ~ PG5/6/7/8/9	EXTI9_5
EXTI10 ~ EXTI15	PA10/11/12/13/14/15 ~ PG10/11/12/13/14/15	EXTI15_10
EXTI16	PVD 출력	PVD
EXTI17	RTC 알람	RCT Alram
EXTI18	USB 웨이크업	USB Wakeup



△ 그림 7-1-1 GPIO와 EXTI의 연결 (STM32F10x)

7.2 외부 인터럽트 관련 HAL 드라이버

외부 인터럽트 구동용 함수

외부 인터럽트 구동을 위해 NVIC의 인터럽트 함수와 GPIO의 API함수 사용

- **인터럽트 함수는 5.5절 HAL CORTEX 드라이버 부분 참고**
- **API 함수는 6.2.2절 GPIO 구동용 함수 부분 참고**

다음의 두개의 함수가 외부 인터럽트의 처리를 위해 자주 사용

- **두 함수는 6.2.3절 함수의 상세설명 부분 참고**
- **HAL_GPIO_EXTI_IRQHandler()**
  - **EXTI 인터럽트의 핸들러 함수**
- **HAL_GPIO_EXTI_Callback()**
  - **EXTI 인터럽트를 처리하기 위한 콜백함수**
    이 함수는 EXTI 핸들러 함수(EXTIx_IRQHandler())내에서 사용

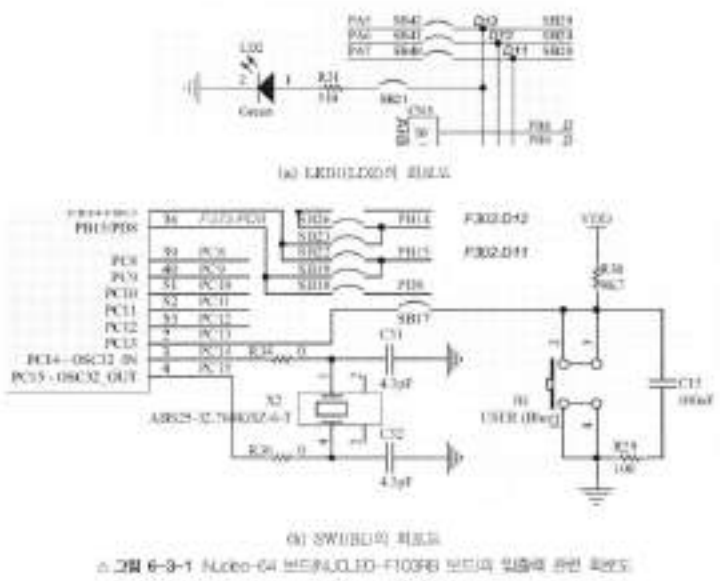
KYUNG HEE UNIVERSITY

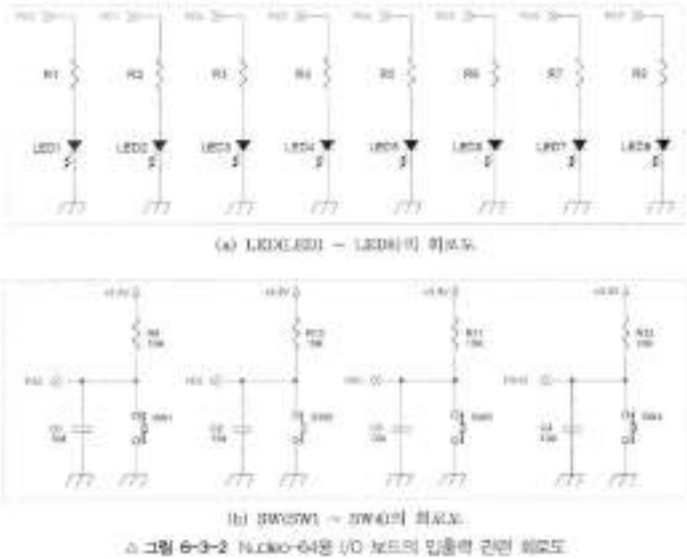
7.3 외부 인터럽트 응용 예제

외부 인터럽트 예제 관련 회로도

Nucleo-F103 확장보드

- **6.3.0절 GPIO 예제 관련 회로도와 동일**
- **스위치는 Pull-up 방식으로 연결**





7.3 외부 인터럽트 응용 예제

EXTI 예제 1 : 외부 인터럽트를 이용한 LED의 On/Off

예제 설명

- SW1을 이용해 외부 인터럽트를 발생
- 인터럽트가 발생하면 LED1~8을 일정시간 On한 뒤 Off
- SW1의 인터럽트는 Falling edge에서 발생하므로 눌리는 순간 LED가 ON됨을 유의

소스코드 작성

- 폴더명 : [Examples_Nucleo F103] - [EXTI] - [EXTI 1_F103] - [MDK_ARM]
- 위의 폴더에 있는 [Project.uvprojx] 파일을 더블클릭하여 실행
- 파일이 열리면 [Example/User] 폴더 내의 [main.c] 파일을 더블 클릭하여 연후 다음 장과 같이 소스코드 작성

7.3 외부 인터럽트 응용 예제

EXTI 예제 1 : 외부 인터럽트를 이용한 LED의 On/Off

```
// 프로그램의 수행에 필요한 헤더파일
#include "main.h"
#include "Nucleo_F103.h"            // Nucleo-F103 확인보드용 헤더 파일
// #include "Nucleo_F429.h"         // Nucleo-F429 확인보드용 헤더 파일

// -- <1> 프로그램 수행에 필요한 전역변수의 선언
int  flag_Sw1, flag_Sw2, flag_Sw3, flag_Sw4;

// ------------------------------------------------------------------------------ //

int main(void)
{
    // -- <2> LED를 켜기 위한 변수
    uint16_t  led = 0x01;

    HAL_Init();
    SystemClock_Config();
    LED_Config();
    // -- <3> 스위치가 눌러지면 EXTI를 발생시키도록 설정
    SwEXTI_Config();

    LED_OnOff(GPIO_PIN_LedAll, 500);

    // 무한 루프로 동작
    while (1) {
        led = 0x00;     // 변수 led를 초기화

        // -- <4> SW1이 눌러지면 -> led의 1, 2번째 bit를 1로 설정 (0x03 = 0000 0011)
        if ( flag_Sw1 == 1 ) {
            led = led | 0x03;
            LED_OnOff(led, 300);
            LED_OnOff(led, 300);
            flag_Sw1 = 0;
        }
        // -- <5> SW2이 눌러지면 -> led의 3, 4번째 bit 를 1로 설정 (0x0c = 0000 1100)
        if ( flag_Sw2 == 1 ) {
            led = led | 0x0c;
            LED_OnOff(led, 300);
            flag_Sw2 = 0;
        }
        // -- <6> SW3 이 눌러지면 -> led 의 5, 6번째 bit 를 1로 설정  (0x30 = 0011 0000)
        if ( flag_Sw3 == 1 ) {
            led = led | 0x30;
            LED_OnOff(led, 300);
            flag_Sw3 = 0;
        }
        // -- <7> SW4 이 눌러지면 -> led 의 7, 8번째 bit 를 1로 설정  (0xc0 = 1100 0000)
        if ( flag_Sw4 == 1 ) {
            led = led | 0xc0;
            LED_OnOff(led, 300);
            flag_Sw4 = 0;
        }
    }
}

// ------------------------------------------------------------------------------ //
// -- <8> EXTI 인터럽트 Callback 함수의 구현

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_PIN)
{
    // -- <9> SW1이 눌러지면 flag_Sw1 = 1 으로 설정
    if ( GPIO_PIN == GPIO_PIN_Sw1)  flag_Sw1 = 1;
    // -- <10> SW2이 눌러지면 flag_Sw2 = 1 으로 설정
    if ( GPIO_PIN == GPIO_PIN_Sw2)  flag_Sw2 = 1;
    // -- <11> SW3이 눌러지면 flag_Sw3 = 1 으로 설정
    if ( GPIO_PIN == GPIO_PIN_Sw3)  flag_Sw3 = 1;
    // -- <12> SW4이 눌러지면 flag_Sw4 = 1 으로 설정
    if ( GPIO_PIN == GPIO_PIN_Sw4)  flag_Sw4 = 1;
}
```

7.3 외부 인터럽트 응용 예제

EXTI 예제 1 : 외부 인터럽트를 이용한 LED의 On/Off

인터럽트 핸들러(handler)함수 작성

- **EXTI 인터럽트 사용을 위해서는 인터럽트 발생시 처리되어야 하는 동작이 정의 되어있는 핸들러 함수 작성 필요**
- **[Example/User] 폴더 내의 [stm32f1xx_it.c] 파일 실행 후 아래와 같이 코드 작성**

[stm32f1xx_it.c]

```
// --  <1> 프로그램의 수행에 필요한 헤더 파일
#include "main.h"
#include "stm32f1xx_it.h"            // 인터럽트 사용에 필요한 헤더 파일

// ------------------------------------------------------------------------- //
// --  <2> SysTick 인터럽트의 핸들러 함수

void SysTick_Handler(void)
{
    // -- <2-1> HAL_IncTick() 함수를 호출
    HAL_IncTick();
}
    // ------------------------------------------------------------------------- //
    // -- <3>  EXTI15_10 인터럽트의 핸들러 함수


void EXTI15_10_IRQHandler(void)
{
    // -- <3-1> GPIO_PIN_10에서 EXTI가 발생한 경우는
    //              함수 HAL_GPIO_EXTI_Callback(GPIO_PIN_10) 를 호출
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_10);
    // -- <3-2> GPIO_PIN_13에서 EXTI가 발생한 경우는
    //              함수 HAL_GPIO_EXTI_Callback(GPIO_PIN_13) 를 호출
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_13);
}
```

```
// ------------------------------------------------------------------------- //
// -- <4>  EXTI4 인터럽트의 핸들러 함수

void EXTI4_IRQHandler(void)
{
    // -- <4-1> GPIO_PIN_8에서 EXTI가 발생한 경우는
    //              함수 HAL_GPIO_EXTI_Callback(GPIO_PIN_4) 를 호출
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_4);
}

// ------------------------------------------------------------------------- //
// -- <5>  EXTI5 ~ 9 인터럽트의 핸들러 함수

void EXTI9_5_IRQHandler(void)
{
    // -- <5-1> GPIO_PIN_5에서 EXTI가 발생한 경우는
    //              함수 HAL_GPIO_EXTI_Callback(GPIO_PIN_5) 를 호출
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_5);
    // -- <5-2> GPIO_PIN_8에서 EXTI가 발생한 경우는
    //              함수 HAL_GPIO_EXTI_Callback(GPIO_PIN_8) 를 호출
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_8);
}
```

7.3 외부 인터럽트 응용 예제

EXTI 예제 1 : 외부 인터럽트를 이용한 LED의 On/Off

소스코드 설명

- **소스 코드의 설명에 붙어있는 번호는 (예 : <1>, <2> 등)은 앞 절의 소스 코드 내에 붙어있는 번호와 동일하다.**

[main. c]

<1> 전역변수 flag_Sw1을 선언하고 값을 0으로 초기화한다. 이 변수는 스위치가 눌러진 것을 식별하기 위한 용도로 사용된다.

<2> SwEXTI_Config() 함수 : 스위치가 눌러지면 EXTI를 발생시키도록 설정하는 함수. 이 함수는 각각 <Nucleo_F103.c> / <Nucleo_F429.c>에 정의되어 있다. [14.3절. 이제 소스코드 추가 설명 : Nucleo_F103.c 및 Nucleo_F429.c]을 참고하기 바란다.

* 이 함수에는 GPIO 입력핀을 EXTI 발생으로 설정하는 소스코드가 있다. 이 소스코드는 EXTI의 이해를 위해서 매우 중요한 코드이므로 반드시 찾아서 살펴보기 바란다.

<3> flag_Sw1=1 이면 (즉, SW1 이 눌러지면) LED를 모두 On 시킨 후에 Off 한다. 그리고 flag_Sw1 = 0 으로 다시 변경한다.

<4> HAL_GPIO_EXTI_Callback() 함수 : EXTI 인터럽트 Callback 함수를 구현한 부분이다. EXTI 인터럽트가 발생하면 MCU는 자동적으로 이 함수를 호출한다. 따라서 EXTI 인터럽트가 발생된 경우 실행하고 싶은 내용을 이 함수내에 코딩하면 된다.

<5> Nucleo-64 보드 S/W(즉, B1이 눌러지면)가 눌러지면 flag_Sw1 = 1 로 설정한다.

<5-1> SW1이 눌러지면 flag_Sw1 = 1 로 설정한다.

KYUNG HEE UNIVERSITY

7.3 외부 인터럽트 응용 예제

EXTI 예제 1 : 외부 인터럽트를 이용한 LED의 On/Off

소스코드 설명

[stm32f10x_it.c]

<1> 프로그램의 수행에 필요한 헤더 파일을 인클루드한다. 특히 인터럽트의 수행을 위해서는 "stm32f1xx_it.h" 헤더파일이 필요하다.

<2> SysTick_Handler() 함수 : SysTick 인터럽트의 핸들러 함수이다. SysTick 인터럽트는 Cortex-M3 프로세서가 1@msec 마다 자동으로 발생시켜주는 인터럽트이다.

<2-1> HAL_IncTick() 함수 : SysTick timer를 증가시켜주는 함수이다. 이 함수는 HAL_Delay( ) 함수를 사용하기 위해서는 필요한 함수이며, stm32f1xx_hal.c에 다음과 같이 정의되어 있다.

```
/**
  * @brief This function is called to increment  a global variable "uwTick"
  *        used as application time base.
  * @note In the default implementation, this variable is incremented each 1ms
  *       in Systick ISR.
  * @note This function is declared as __weak to be overwritten in case of other
  *      implementations in user file.
  * @retval None
  */
__weak void HAL_IncTick(void)
{
  uwTick++;
}
```

<3> EXTI15_10_IRQHandler( ) : EXTI15_10 인터럽트가 발생한 경우 이를 처리하는 핸들러 함수이다.

<3-1> HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_10) : GPIO_PIN_10에서 EXTI가 발생한 경우 HAL_GPIO_EXTI_Callback() 함수를 호출하는 함수이다. HAL_GPIO_EXTI_Callback() 함수는 <main.c>에 정의되어 있다.

<3-2> HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_13) : GPIO_PIN_13에서 EXTI가 발생한 경우 HAL_GPIO_EXTI_Callback() 함수를 호출하는 함수이다. HAL_GPIO_EXTI_Callback() 함수는 <main.c>에 정의되어 있다.

<4> EXTI4_IRQHandler( ) : EXTI4 인터럽트가 발생한 경우 이를 처리하는 핸들러 함수이다.

<4-1> HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_4) : GPIO_PIN_4에서 EXTI가 발생한 경우 HAL_GPIO_EXTI_Callback() 함수를 호출하는 함수이다. HAL_GPIO_EXTI_Callback() 함수는 <main.c>에 정의되어 있다.

<5> EXTI9_5_IRQHandler( ) : EXTI9_5 인터럽트가 발생한 경우 이를 처리하는 핸들러 함수이다.

<5-1> HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_5) : GPIO_PIN_5에서 EXTI가 발생한 경우 HAL_GPIO_EXTI_Callback() 함수를 호출하는 함수이다. HAL_GPIO_EXTI_Callback() 함수는 <main.c>에 정의되어 있다.

<5-2> HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_8) : GPIO_PIN_8에서 EXTI가 발생한 경우 HAL_GPIO_EXTI_Callback() 함수를 호출하는 함수이다. HAL_GPIO_EXTI_Callback() 함수는 <main.c>에 정의되어 있다.

7.3 외부 인터럽트 응용 예제

EXTI 예제 1 : 외부 인터럽트를 이용한 LED의 On/Off

참고자료

- Nucleo-F103 확장보드의 경우 스위치가 눌러지면 이에 대응하여 발생하는 EXTI는 다음과 같다.

스위치	GPIO 포트, 핀	발생되는 EXTI
B1(User Button)	GPIOC 13	EXTI15_10
SW1	GPIOA 8	EXTI9_5
SW2	GPIOB 4	EXTI4
SW3	GPIOB 5	EXTI9_5
SW4	GPIOB 10	EXTI5_10

- EXTI를 사용하지 않더라도 일반적으로 Systick_Handler() 함수는 사용된다.
- 따라서 별도의 프로그램을 하는 경우에도 항상 [stm32f10x_it.c] 파일을 작성하고 여기에 Systick_Handler()함수를 포함하여 두면된다.

7.3 외부 인터럽트 응용 예제

EXTI 예제 2 : 외부 인터럽트를 이용한 LED 이동 점멸

예제 설명

- 입력용 스위치 SW1, SW2를 이용해 외부 인터럽트 발생
- SW1을 눌러 인터럽트가 발생하면 LED를 1번->2번->···->8번 까지 점멸을 반복
- SW2를 눌러 인터럽트가 발생하면 LED를 8번->7번->···->1번 까지 점멸을 반복
- SW1,SW2의 인터럽트는 모두 Falling edge에서 발생

소스코드 작성

- 폴더명 : [Examples_Nucleo F103] - [EXTI] - [EXTI 2_F103] - [MDK_ARM]
- 위의 폴더에 있는 [Project.uvprojx] 파일을 더블클릭하여 실행
- 파일이 열리면 [Example/User] 폴더 내의 [main.c] 파일을 더블 클릭하여 연후 다음 장과 같이 소스코드 작성
- 예제 1번에서 사용한 handler 함수를 동일하게 사용하므로 handler 함수에 대한 추가 설명은 생략

KYUNG HEE UNIVERSITY

7.3 외부 인터럽트 응용 예제

EXTI 예제 2 : 외부 인터럽트를 이용한 LED 이동 점멸

[main.c]

```
// 프로그램에 수행에 필요한 헤더파일
#include "main.h"
#include "Nucleo_F103.h"              // Nucleo-F103 확장보드용 헤더 파일
// #include "Nucleo_F429.h"           // Nucleo-F429 확장보드용 헤더 파일

// -- <1> 프로그램 수행에 필요한 전역변수의 선언
int  flag_Sw1 = 0, flag_Sw2 = 0;

int main(void)
{
    // -- <2> LED를 제어하는 변수
    uint16_t  led = 0x01;
    HAL_Init();
    SystemClock_Config();
    LED_Config();
    // -- <3> 스위치가 눌러지면 EXTI를 발생시키도록 설정
    SwEXTI_Config();

    LED_OnOff(GPIO_PIN_LedAll, 500);

    // 무한 루프로 동작
    while (1) {
        // -- <4> SW1 이 눌러지면 -> LED를 1번 → 2번 → 3번 → ··· → 8번 순서로 점멸하는 동작을 반복
        if ( flag_Sw1 == 1 ) {
                led <<= 1;
                if ( led > 0x80 ) led = 0x01;
                LED_OnOff(led, 100);
        }
        // -- <5> SW2 이 눌러지면 -> LED를 8번 → 7번 → ··· → 1번 순서로 점멸하는 동작을 반복
        else if ( flag_Sw2 == 1 ) {
                led >>= 1;
                if ( led < 0x01 ) led = 0x80;
                LED_OnOff(led, 100);
        }
    }
}

// ------------------------------------------------------------------------ //
// -- <6> EXTI 인터럽트 Callback 함수의 구현

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_PIN)
{
    // -- <7> SW1이 눌러지면 flag_Sw1 = 1 , flag_Sw2 = 0 으로 설정
    if ( GPIO_PIN == GPIO_PIN_Sw1) {
        flag_Sw1 = 1;
        flag_Sw2 = 0;
    }
    // -- <8> SW2이 눌러지면 flag_Sw1 = 0 , flag_Sw2 = 1 로 설정
    else if ( GPIO_PIN == GPIO_PIN_Sw2) {
        flag_Sw1 = 0;

        flag_Sw2 = 1;
    }
}
```

7.3 외부 인터럽트 응용 예제

EXTI 예제 2 : 외부 인터럽트를 이용한 LED 이동 점멸

소스코드 설명

- **소스 코드의 설명에 붙어있는 번호는 (예 : <1>, <2> 등)은 앞 절의 소스 코드 내에 붙어있는 번호와 동일**
- **Handler 함수의 설명은 예제 1번과 동일**

[main.c]

<1> 전역변수 flag_Sw1, flag_Sw2를 선언하고 값을 0으로 초기화한다. 이 변수는 스위치가 눌러진 것을 식별하기 위한 용도로 사용된다.
<2> LED를 켜기 위한 변수 led를 선언하고 값을 0x01로 초기화 한다.
<3> SwEXTI_Config() 함수 : 스위치가 눌러지면 EXTI를 발생시키도록 설정하는 함수. 이 함수는 각각 <Nucleo_F103.c> / <Nucleo_F429.c>에 정의되어 있다. 이 함수에 대한 설명은 [부록1.4. 예제 소스코드 추가 설명 ; Nucleo_F103.c 및 Nucleo_F429.c]을 참고하기 바란다.
<4> flag_Sw1=1 이면 (즉, SW1 이 눌러지면) LED를 1번 → 2번 → 3번 → … → 8번 순서로 점멸하는 동작을 반복하는 소스코드이다.
<5> flag_Sw2=1 이면 (즉, SW2 가 눌러지면) LED를 8번 → 7번 → … → 1번 순서로 점멸하는 동작을 반복하는 소스코드이다.
<6> HAL_GPIO_EXTI_Callback() 함수 : EXTI 인터럽트 Callback 함수를 구현한 부분이다. 이 함수는 다음 동작을 실행한다.
<7> SW1이 눌러지면 flag_Sw1 = 1 , flag_Sw2 = 0 으로 설정한다.
<8> SW2이 눌러지면 flag_Sw1 = 0 , flag_Sw2 = 1 로 설정한다.

KYUNG HEE UNIVERSITY

7.3 외부 인터럽트 응용 예제

EXTI 예제 3 : 4개의 외부 인터럽트를 이용한 LED On/Off(1)

예제 설명

- SW1을 누르면 LED1~2가 SW2를 누르면 LED3~4, SW3은 LED5~6, SW4는 LED7~8이 각각 On/Off
- 스위치가 눌러지면 이에 대응되는 EXTI가 발생
- 이와 연결된 콜백함수에서 해당 인터럽트 플래그를 발생
- SW1,SW2,SW3,SW4의 인터럽트는 모두 Falling edge에서 발생

소스코드 작성

- 폴더명 : [Examples_Nucleo F103] - [EXTI] - [EXTI 3_F103] - [MDK_ARM]
- 위의 폴더에 있는 [Project.uvprojx] 파일을 더블클릭하여 실행
- 파일이 열리면 [Example/User] 폴더 내의 [main.c] 파일을 더블 클릭하여 연후 다음 장과 같이 소스코드 작성
- 예제 1번에서 사용한 handler 함수를 동일하게 사용하므로 handler 함수에 대한 추가 설명은 생략

7.3 외부 인터럽트 응용 예제

EXTI 예제 3 : 4개의 외부 인터럽트를 이용한 LED On/Off(1)

[main.c]

```
// 프로그램에 수정해 필요한 헤더파일
#include "main.h"
#include "Nucleo_F103.h"            // Nuclec-F103 타겟보드용 헤더 파일
// #include "Nucleo_F429.h"         // Nuclec-F429 타겟보드용 헤더 파일

// -- <1> 프로그램 수행에 필요한 전역변수의 선언
int  flag_Sw1, flag_Sw2, flag_Sw3, flag_Sw4;

// ------------------------------------------------------------------------------ //

int main(void)
{
    // -- <2> LED를 켜기 위한 변수
    uint16_t  led = 0x00;

    HAL_Init();
    SystemClock_Config();
    LED_Config();
    // -- <3> 스위치가 눌러지면 EXTI를 발생시키도록 설정
    SwEXTI_Config();

    LED_OnOff(GPIO_PIN_LedAll, 500);

    // 무한 루프로 동작
    while (1) {
        led = 0x00;     // 변수 led를 초기화

        // -- <4> Sw1이 눌러지면 -> led의 1, 2번째 bit을 1로 설정 (0x03 = 0000 0011)
        if ( flag_Sw1 == 1 ) {
            led = led | 0x03;
            LED_OnOff(led, 300);
            LED_OnOff(led, 300);
            flag_Sw1 = 0;
        }
        // -- <5> Sw2이 눌러지면 -> led의 3, 4번째 bit 을 1로 설정 (0x0c = 0000 1100)
        if ( flag_Sw2 == 1 ) {
            led = led | 0x0c;
            LED_OnOff(led, 300);
            flag_Sw2 = 0;
        }
        // -- <6> Sw3 이 눌러지면 -> led 의 5, 6번째 bit 을 1로 설정  (0x30 = 0011 0000)
        if ( flag_Sw3 == 1 ) {
            led = led | 0x30;
            LED_OnOff(led, 300);
            flag_Sw3 = 0;
        }
        // -- <7> Sw4 이 눌러지면 -> led 의 7, 8번째 bit 을 1로 설정  (0xc0 = 1100 0000)
        if ( flag_Sw4 == 1 ) {
            led = led | 0xc0;
            LED_OnOff(led, 300);
            flag_Sw4 = 0;
        }
    }
}

// ------------------------------------------------------------------------------ //
// -- <8> EXTI 인터럽트 Callback 함수의 구현

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_PIN)
{
    // -- <9> Sw1이 눌러지면 flag_Sw1 = 1 으로 설정
    if ( GPIO_PIN == GPIO_PIN_Sw1)  flag_Sw1 = 1;
    // -- <10> Sw2이 눌러지면 flag_Sw2 = 1 으로 설정
    if ( GPIO_PIN == GPIO_PIN_Sw2)  flag_Sw2 = 1;
    // -- <11> Sw3이 눌러지면 flag_Sw3 = 1 으로 설정
    if ( GPIO_PIN == GPIO_PIN_Sw3)  flag_Sw3 = 1;
    // -- <12> Sw4이 눌러지면 flag_Sw4 = 1 으로 설정
    if ( GPIO_PIN == GPIO_PIN_Sw4)  flag_Sw4 = 1;
}
```

KYUNG HEE UNIVERSITY

7.3 외부 인터럽트 응용 예제

EXTI 예제 3 : 4개의 외부 인터럽트를 이용한 LED On/Off(1)

소스코드 설명

- **소스 코드의 설명에 붙어있는 번호는 (예 : <1>, <2> 등)은 앞 절의 소스 코드 내에 붙어있는 번호와 동일**
- **Handler 함수의 설명은 예제 1번과 동일**

[main.c]

<1> 전역변수 flag_Sw1, flag_Sw2, flag_Sw3, flag_Sw4를 선언하고 값을 0으로 초기화 한다. 이 변수는 스위치가 눌러진 것을 식별하기 위한 용도로 사용된다.
<2> LED를 켜기위한 변수 led를 선언하고 값을 0x01로 초기화 한다.
<3> F103_SwEXTI_Config()/F107_SwEXTI_Config() 함수 : 스위치가 눌러지면 EXTI를 발생시키도록 설정하는 함수. 이 함수는 각각 <Nucleo_F103.c>/<F107_Board.c>에 정의되어 있다. 이 함수에 대한 설명은 6장 GPIO의 [6.3.1.4 소스코드 설명]을 참고하기 바란다.
<4> SW1이 눌러지면 -> led의 1, 2번째 bit를 1로 설정 (0x03 = 0000 0011) 한다.
<5> SW2이 눌러지면 -> led의 3, 4번째 bit를 1로 설정 (0x0c = 0000 1100) 한다.
<6> SW3이 눌러지면 -> led의 5, 6번째 bit를 1로 설정 (0x30 = 0011 0000) 한다.
<7> SW4이 눌러지면 -> led의 7, 8번째 bit를 1로 설정 (0xc0 = 1100 0000) 한다.

<8> HAL_GPIO_EXTI_Callback() 함수 : EXTI 인터럽트 Callback 함수를 구현한 부분이다. 이 함수는 다음 동작을 실행한다.
<9> SW1이 눌러지면 flag_Sw1 = 1로 설정한다.
<10> SW2이 눌러지면 flag_Sw2 = 1로 설정한다.
<11> SW3이 눌러지면 flag_Sw3 = 1로 설정한다.
<12> SW4이 눌러지면 flag_Sw4 = 1로 설정한다.

KYUNG HEE UNIVERSITY

7.3 외부 인터럽트 응용 예제

EXTI 예제 4 : 4개의 외부 인터럽트를 이용한 LED On/Off(2)

예제 설명

- SW1을 누르면 LED1~2가 SW2를 누르면 LED3~4, SW3은 LED5~6, SW4는 LED7~8이 각각 토글 되어 On/Off
- [EXTI 예제3]과 동작은 유사하지만 코드를 살펴보면 EXTI가 발생하면 이와 연결된 콜백 함수에서 직접적으로 토글하여 On/Off를 하는 차이점이 존재
- 본 예제는 스위치를 누를 때 LED의 동작이 원활하게 On/Off되지 않는 것으로 보일 수도 있으나 이것은 스위치를 누를 때 나타나는 chattering 현상때문에 토글이 여러 번 발생하여 일어나는 현상

소스코드 작성

- 폴더명 : [Examples_Nucleo F103] - [EXTI] - [EXTI 4_F103] - [MDK_ARM]
- 위의 폴더에 있는 [Project.uvprojx] 파일을 더블클릭하여 실행
- 파일이 열리면 [Example/User] 폴더 내의 [main.c] 파일을 더블 클릭하여 연후 다음 장과 같이 소스코드 작성
- 예제 1번에서 사용한 handler 함수를 동일하게 사용하므로 handler 함수에 대한 추가 설명은 생략

7.3 외부 인터럽트 응용 예제

EXTI 예제 4 : 4개의 외부 인터럽트를 이용한 LED On/Off(2)

소스코드 작성

[main.c]

```
// 프로그램의 수행에 필요한 헤더파일
#include "main.h"
#include "Nucleo_F103.h"              // Nucleo-F103 확장보드용 헤더 파일
// #include "Nucleo_F429.h"           // Nucleo-F429 확장보드용 헤더 파일

// ------------------------------------------------------------------------- //

int main(void)
{
    HAL_Init();
    SystemClock_Config();
    LED_Config();
    SwEXTI_Config();

    LED_OnOff(GPIO_PIN_LedAll, S00);

    // -- <1> 무한 루프로 동작하지만 여기서는 아무런 일도 하지 않음
    while (1) { }
}
```

```
// ------------------------------------------------------------------------- //
// -- <2> EXTI 인터럽트 Callback 함수의 구현
/*      이 함수는 HAL_GPIO_TogglePin() 함수를 이용하여 LED를 On/Off 한다.
        따라서 동작시에 Switch의 Chattering 현상으로 LED On/Off가 여러번 발생할 수도 있다.*/
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_PIN)
{
  // -- <3> SW1 이 눌러지면 -> LED 1, 2를 Toggle
  if ( GPIO_PIN == GPIO_PIN_Sw1) {
      HAL_GPIO_TogglePin(GPIOExt, GPIO_PIN_Led1 | GPIO_PIN_Led2 );
  }
  // -- <4> SW2 이 눌러지면 -> LED 3, 4를 Toggle
  if ( GPIO_PIN == GPIO_PIN_Sw2) {
      HAL_GPIO_TogglePin(GPIOExt, GPIO_PIN_Led3 | GPIO_PIN_Led4);
  }
  // -- <5> SW3 이 눌러지면 -> LED 5, 6를 Toggle
  if ( GPIO_PIN == GPIO_PIN_Sw3) {
      HAL_GPIO_TogglePin(GPIOExt, GPIO_PIN_Led5 | GPIO_PIN_Led6);
  }
  // -- <6> SW4이 눌러지면 -> LED 7, 8를 Toggle
  if ( GPIO_PIN == GPIO_PIN_Sw4) {
      HAL_GPIO_TogglePin(GPIOExt, GPIO_PIN_Led7 | GPIO_PIN_Led8);
  }
}
```

7.3 외부 인터럽트 응용 예제

EXTI 예제 4 : 4개의 외부 인터럽트를 이용한 LED On/Off(2)

소스코드 설명

- **소스 코드의 설명에 붙어있는 번호는 (예 : <1>, <2> 등)은 앞 절의 소스 코드 내에 붙어있는 번호와 동일**
- **Handler 함수의 설명은 예제 1번과 동일**

[main.c]

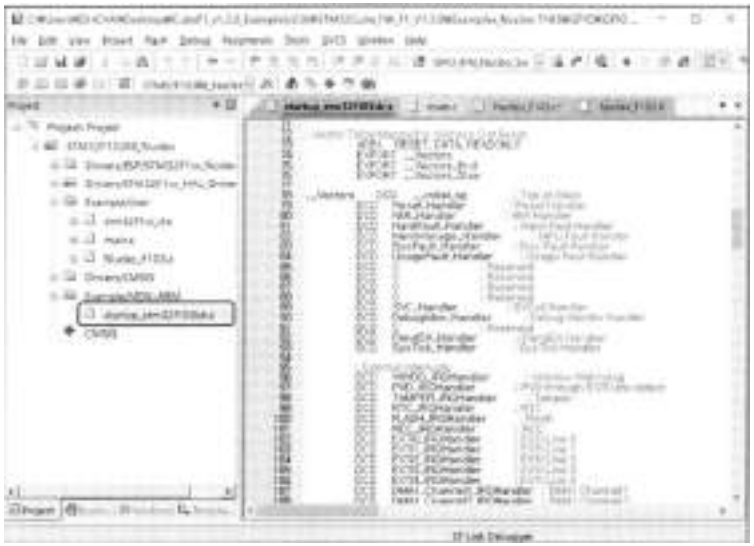
<1> main() 함수의 무한 루프내에서는 아무런 작업도 하지 않는다.
<2> HAL_GPIO_EXTI_Callback() 함수 : EXTI 인터럽트 Callback 함수를 구현한 부분이다. 이 함수는 다음 동작을 실행한다.
<3> SW1 이 눌러지면 -> LED1, 2를 Toggle
<4> SW2 이 눌러지면 -> LED3, 4를 Toggle
<5> SW4 이 눌러지면 -> LED5, 6을 Toggle
<6> SW4 이 눌러지면 -> LED7, 8을 Toggle

7.3 외부 인터럽트 응용 예제

[심화학습] 인터럽트 관련 HAL 드라이버 자세히 알아보기

STM32F1xx의 인터럽트 핸들러의 이름은 어디에 정의되어 있나?

- **STM32F1xx의 모든 인터럽트 핸들러의 이름 정의는 스타트 업 파일(startup_stm32f1xx.s)에 정의**
- **폴더명 : [Projects] - [STM32F103RB-Nucleo] - [Templates] - [EWARM]**
- **다음장은 스타트 업 파일에서 인터럽트 핸들러의 이름이 정의된 소스코드의 예시**



KYUNG HEE UNIVERSITY

7.3 외부 인터럽트 응용 예제

[심화학습] 인터럽트 관련 HAL 드라이버 자세히 알아보기

< startup_stm32f103xb.s >의 일부 소스 코드

```
...
(중 략 생략)
...

__Vectors
  DCD    __initial_sp               ; Top of Stack
  DCD    Reset_Handler              ; Reset Handler
  DCD    NMI_Handler                ; NMI Handler
  DCD    HardFault_Handler          ; Hard Fault Handler
  DCD    MemManage_Handler          ; MPU Fault Handler
  DCD    BusFault_Handler           ; Bus Fault Handler
  DCD    UsageFault_Handler         ; Usage Fault Handler
  DCD    0                          ; Reserved
  DCD    0                          ; Reserved
  DCD    0                          ; Reserved
  DCD    0                          ; Reserved
  DCD    SVC_Handler                ; SVCall Handler
  DCD    DebugMon_Handler           ; Debug Monitor Handler
  DCD    0                          ; Reserved
  DCD    PendSV_Handler             ; PendSV Handler
  DCD    SysTick_Handler            ; SysTick Handler

  ; External Interrupts
  DCD    WWDG_IRQHandler            ; Window Watchdog
  DCD    PVD_IRQHandler             ; PVD through EXTI Line detect
  DCD    TAMPER_IRQHandler          ; Tamper
  DCD    RTC_IRQHandler             ; RTC
  DCD    FLASH_IRQHandler           ; Flash
  DCD    RCC_IRQHandler             ; RCC
  DCD    EXTI0_IRQHandler           ; EXTI Line 0
  DCD    EXTI1_IRQHandler           ; EXTI Line 1
  DCD    EXTI2_IRQHandler           ; EXTI Line 2
  DCD    EXTI3_IRQHandler           ; EXTI Line 3
  DCD    EXTI4_IRQHandler           ; EXTI Line 4
  DCD    DMA1_Channel1_IRQHandler   ; DMA1 Channel 1
  DCD    DMA1_Channel2_IRQHandler   ; DMA1 Channel 2
  DCD    DMA1_Channel3_IRQHandler   ; DMA1 Channel 3
  DCD    DMA1_Channel4_IRQHandler   ; DMA1 Channel 4
  DCD    DMA1_Channel5_IRQHandler   ; DMA1 Channel 5
  DCD    DMA1_Channel6_IRQHandler   ; DMA1 channel 6
  DCD    DMA1_Channel7_IRQHandler   ; DMA1 channel 7
  DCD    DMA2_Channel1_IRQHandler   ; DMA2 Channel 1
```

```
  DCD    ADC1_2_IRQHandler          ; ADC1 and ADC2
  DCD    CAN1_TX_IRQHandler         ; CAN1 TX
  DCD    CAN1_RX0_IRQHandler        ; CAN1 RX0
  DCD    CAN1_RX1_IRQHandler        ; CAN1 RX1
  DCD    CAN1_SCE_IRQHandler        ; CAN1 SCE
  DCD    EXTI9_5_IRQHandler         ; EXTI Line 9..5
  DCD    TIM1_BRK_IRQHandler        ; TIM1 Break
  DCD    TIM1_UP_IRQHandler         ; TIM1 Update
  DCD    TIM1_TRG_COM_IRQHandler    ; TIM1 Trigger and Commutation
  DCD    TIM1_CC_IRQHandler         ; TIM1 Capture Compare
  DCD    TIM2_IRQHandler            ; TIM2
  DCD    TIM3_IRQHandler            ; TIM3
  DCD    TIM4_IRQHandler            ; TIM4
  DCD    I2C1_EV_IRQHandler         ; I2C1 Event
  DCD    I2C1_ER_IRQHandler         ; I2C1 Error
  DCD    I2C2_EV_IRQHandler         ; I2C2 Event
  DCD    I2C2_ER_IRQHandler         ; I2C1 Error
  DCD    SPI1_IRQHandler            ; SPI1
  DCD    SPI2_IRQHandler            ; SPI2
  DCD    USART1_IRQHandler          ; USART1
  DCD    USART2_IRQHandler          ; USART2
  DCD    USART3_IRQHandler          ; USART3
  DCD    EXTI15_10_IRQHandler       ; EXTI Line 15..10
  DCD    RTC_Alarm_IRQHandler       ; RTC alarm through EXTI line
  DCD    OTG_FS_WKUP_IRQHandler     ; USB OTG FS Wakeup through EXTI line
  DCD    0                          ; Reserved
  DCD    0                          ; Reserved
  DCD    0                          ; Reserved
  DCD    0                          ; Reserved
  DCD    0                          ; Reserved
  DCD    0                          ; Reserved
  DCD    0                          ; Reserved
  DCD    TIM5_IRQHandler            ; TIM5
  DCD    SPI3_IRQHandler            ; SPI3
  DCD    UART4_IRQHandler           ; UART4
  DCD    UART5_IRQHandler           ; UART5
  DCD    TIM6_IRQHandler            ; TIM6
  DCD    TIM7_IRQHandler            ; TIM7
  DCD    DMA2_Channel1_IRQHandler   ; DMA2 Channel1
  DCD    DMA2_Channel2_IRQHandler   ; DMA2 Channel2
  DCD    DMA2_Channel3_IRQHandler   ; DMA2 Channel3
  DCD    DMA2_Channel4_IRQHandler   ; DMA2 Channel4
  DCD    DMA2_Channel5_IRQHandler   ; DMA2 Channel5
  DCD    ETH_IRQHandler             ; Ethernet
  DCD    ETH_WKUP_IRQHandler        ; Ethernet Wakeup through EXTI line
  DCD    CAN2_TX_IRQHandler         ; CAN2 TX
  DCD    CAN2_RX0_IRQHandler        ; CAN2 RX0
  DCD    CAN2_RX1_IRQHandler        ; CAN2 RX1
  DCD    CAN2_SCE_IRQHandler        ; CAN2 SCE
  DCD    OTG_FS_IRQHandler          ; USB OTG FS
__Vectors_End

...
(이하 생략)
...
```

KYUNG HEE
UNIVERSITY

7.3 외부 인터럽트 응용 예제

[심화학습] 인터럽트 관련 HAL 드라이버 자세히 알아보기

HAL_GPIO_EXTI_IRQHandler()와 HAL_GPIO_EXTI_Callback() 함수는 어떻게 동작하나?

- **STM32F1xx의 GPIO관련 HAL 함수들은 [stm32f1xx_hal_gpio.c]파일 내에 정의**
- **폴더명 : [Drivers] - [STM32F1xx_HAL_Driver] - [Src]**
- **아래는 [stm32f1xx_hal_gpio.c] 파일에서 EXTI 관련 부분의 소스코드의 예시**

[stm32f1xx_hal_gpio.c]의 EXTI 관련 소스코드

```
...
(윗 부분 생략)
...

/**
  * @brief This function handles EXTI interrupt request.
  * @param GPIO_Pin : Specifies the pins connected EXTI line
  * @retval None
  */
void HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin)
{
  /* EXTI line interrupt detected */
  if(__HAL_GPIO_EXTI_GET_IT(GPIO_Pin) != RESET)
  {
    __HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);
    HAL_GPIO_EXTI_Callback(GPIO_Pin);
  }
}
```

```
/**
  * @brief  EXTI line detection callback
  * @param GPIO_Pin : Specifies the pins connected EXTI line
  * @retval None
  */
__weak void HAL_GPIO_EXTI_Callback(uint16_t 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
   */
}

...
(아래 생략)
...
```

7.3 외부 인터럽트 응용 예제

[심화학습] 인터럽트 관련 HAL 드라이버 자세히 알아보기

HAL_GPIO_EXTI_IRQHandler()와 HAL_GPIO_EXTI_Callback() 함수는 어떻게 동작하나?

- **EXTI의 동작의 수행을 알기 위해 [EXTI 예제3]의 인터럽트 처리 루틴인 <stm32f1xx_it.c>를 다시 살펴본다**

```
...
(앞 부분 생략)
...
// ------------------------------------------------------------------------------ //
// -- <3>  EXTI15_10 인터럽트의 핸들러 함수

void EXTI15_10_IRQHandler(void)
{
    // -- <3-1> GPIO_PIN_10에서 EXTI가 발생한 경우는
    //              함수 HAL_GPIO_EXTI_Callback(GPIO_PIN_10) 를 호출
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_10);
    // -- <3-2> GPIO_PIN_13에서 EXTI가 발생한 경우는
    //              함수 HAL_GPIO_EXTI_Callback(GPIO_PIN_13) 를 호출
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_13);
}
// ------------------------------------------------------------------------------ //
...
(아래 부분 생략)
...
```

- **앞에서 EXTI15_10_IRQHandler() 핸들러 함수의 소스코드는 다음과 같다**

```
HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_10);
HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_13);
```

KYUNG HEE UNIVERSITY

7.3 외부 인터럽트 응용 예제

[심화학습] 인터럽트 관련 HAL 드라이버 자세히 알아보기

HAL_GPIO_EXTI_IRQHandler()와 HAL_GPIO_EXTI_Callback() 함수는 어떻게 동작하나?

- **GPIO_PIN_13에서 EXTI가 발생한 경우**

  1)EXTI15_10_IRQHandler() 함수에서 → HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_13) 함수를 호출한다.
  이 함수는 <stm32f1xx_hal_gpio.c>파일 내에 있다.

  2)HAL_GPIO_EXTI_IRQHandler() 함수에서 → HAL_GPIO_EXTI_Callback(GPIO_PIN_13) 함수를 호출한다.
  이 함수는 <main.c> 파일 내에 있다.

  3)HAL_GPIO_EXTI_Callback() 함수에서 → 대응되는 LED를 ON시켜준다.

- **GPIO_PIN_10이나 GPIO_PIN_13이 아닌 다른 핀에서 EXTI가 발생한 경우**

  - 예를 들어 GPIO_PIN_14에서 EXTI가 발생하였다면 다음과 같이 동작하게 된다.

  1)EXTI15_10_IRQHandler() 함수에서 → 해당되는 부분이 없으므로 HAL_GPIO_EXTI_IRQHandler() 함수가 호출되지 않는다.

  2)따라서 이후의 동작은 없다.

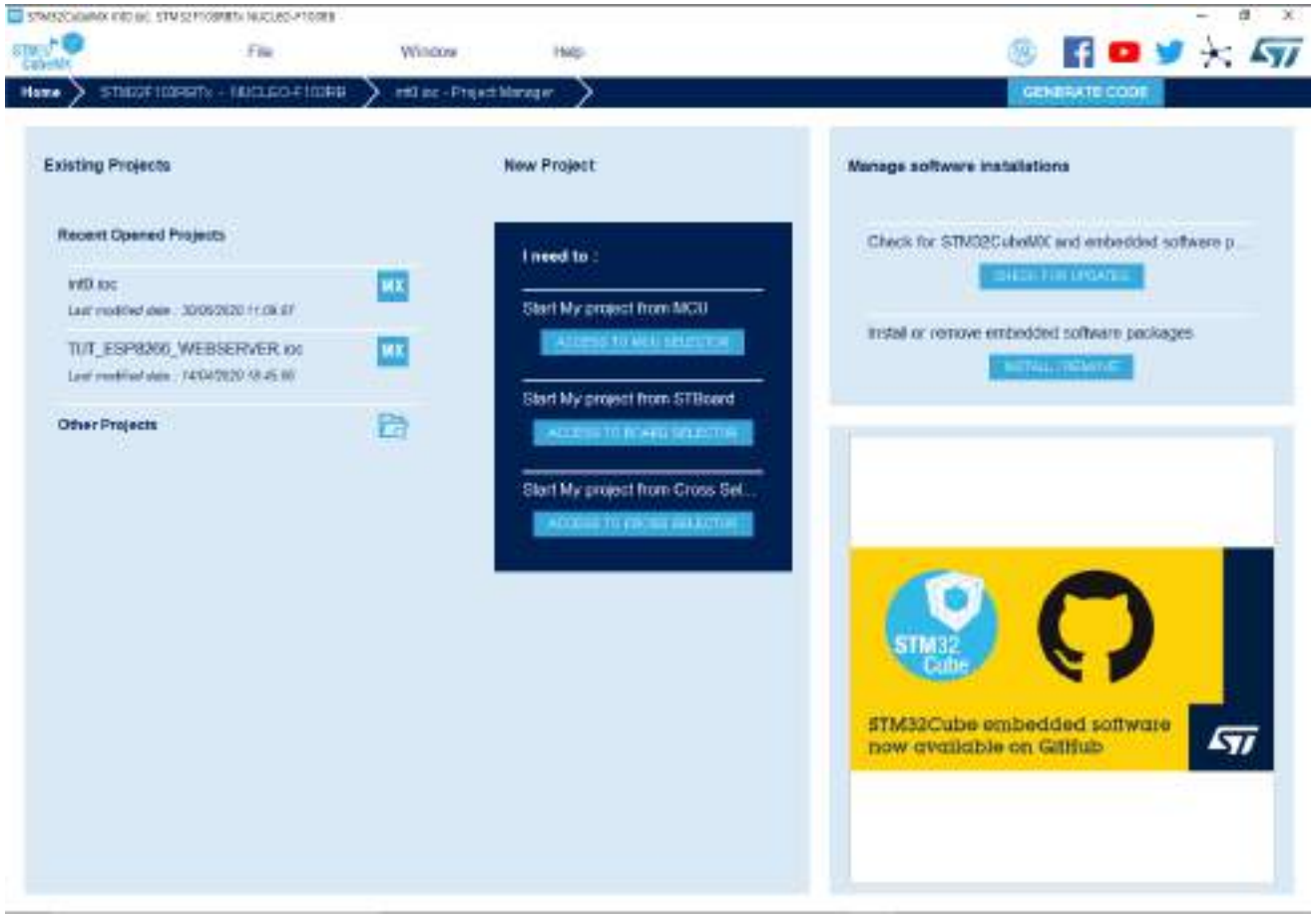
KYUNG HEE UNIVERSITY

7.4 외부 인터럽트 CubeMX과제

CubeMX로 예제 7.1 구현하기

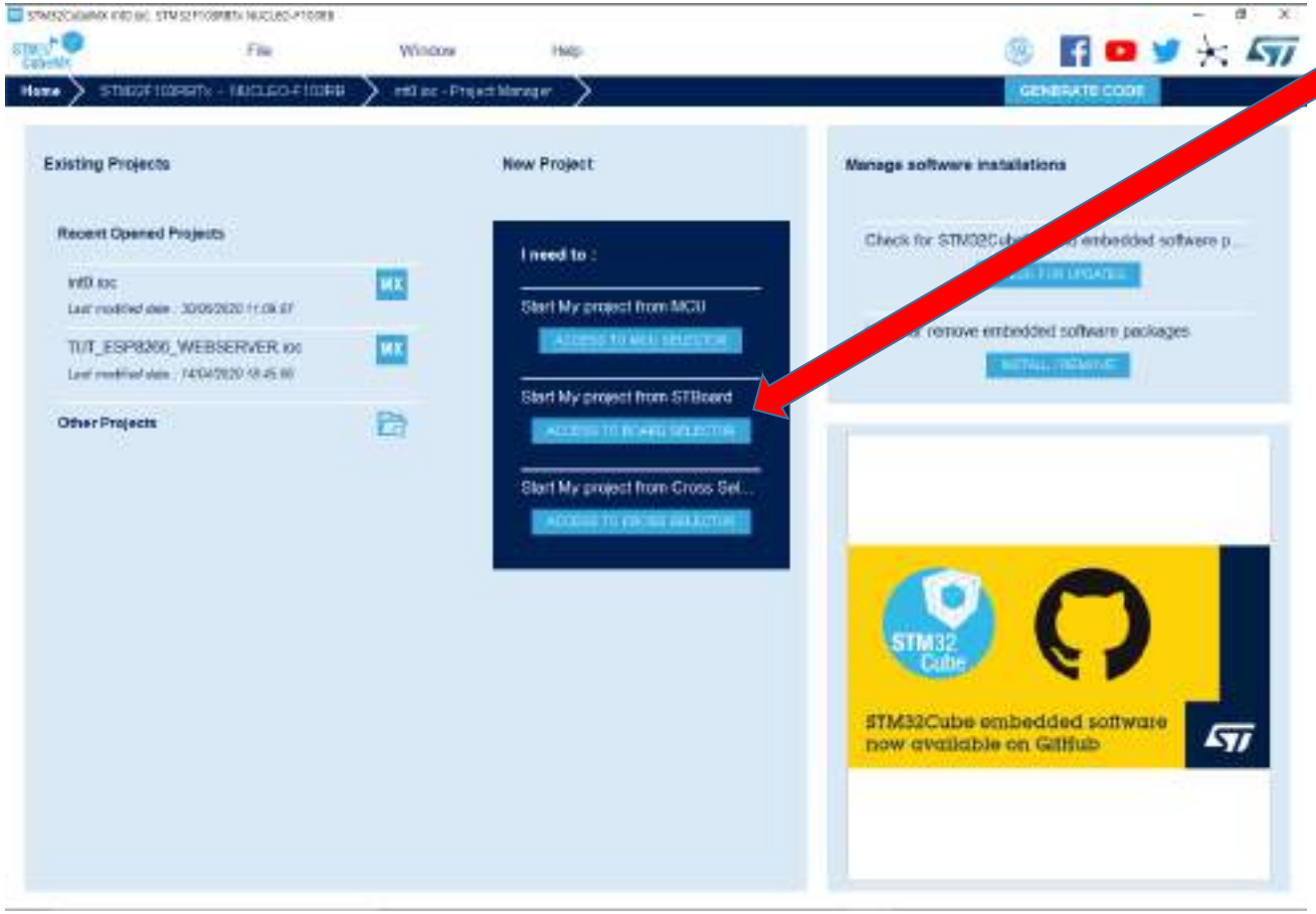
CubeMX 실행

초기화면



7.4 외부 인터럽트 CubeMX과제

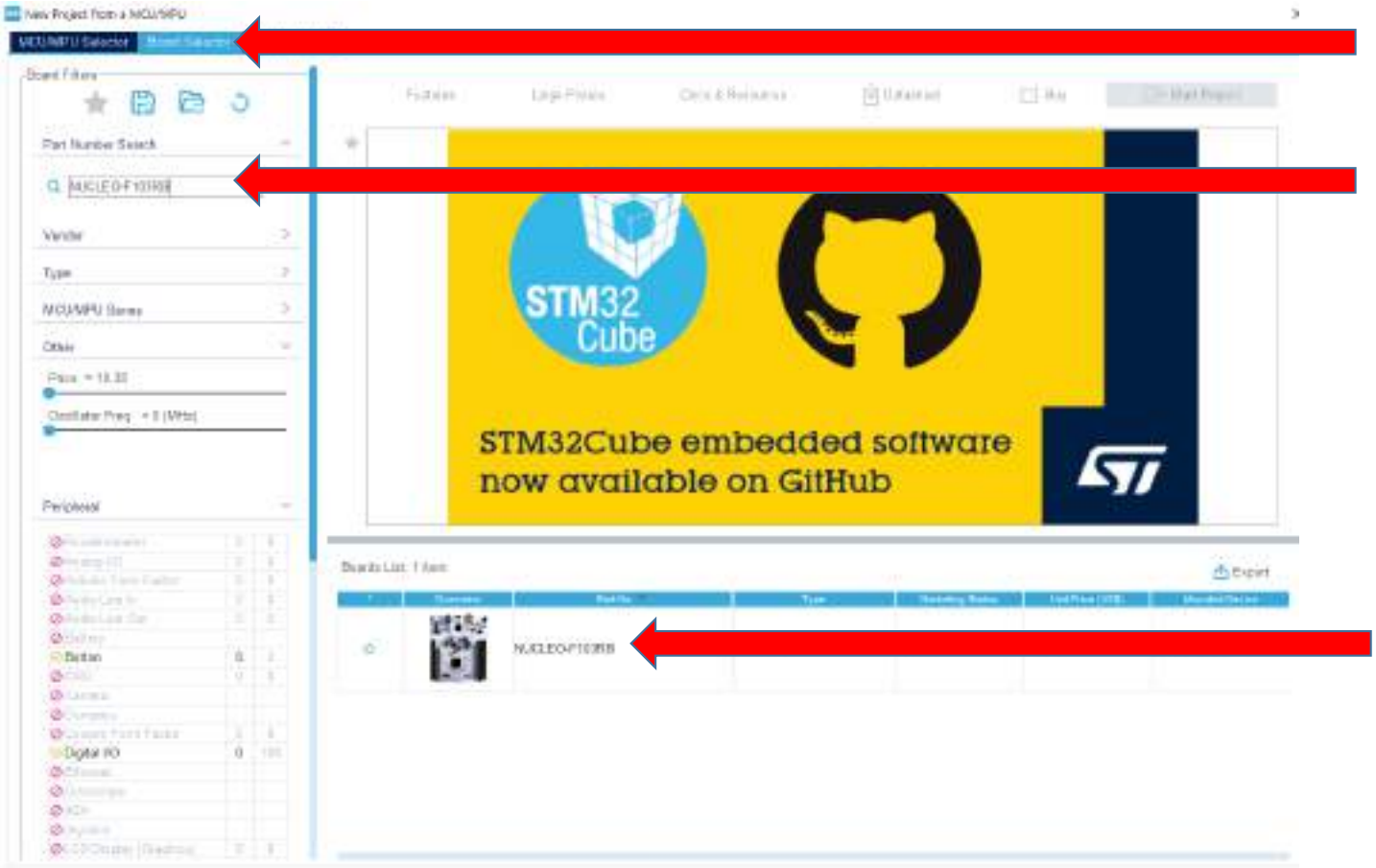
CubeMX로 예제 7.1 구현하기



보드 선택

7.4 외부 인터럽트 CubeMX과제

CubeMX로 예제 7.1 구현하기



보드 선택

NUCLEO-F103RB 선택

NUCLEO-F103RB 더블클릭

KYUNG HEE
UNIVERSITY

7.4 외부 인터럽트 CubeMX과제

CubeMX로 예제 7.1 구현하기

Yes 선택하면 오른쪽과 같이 칩이 보임

7.4 외부 인터럽트 CubeMX과제

CubeMX로 예제 7.1 구현하기

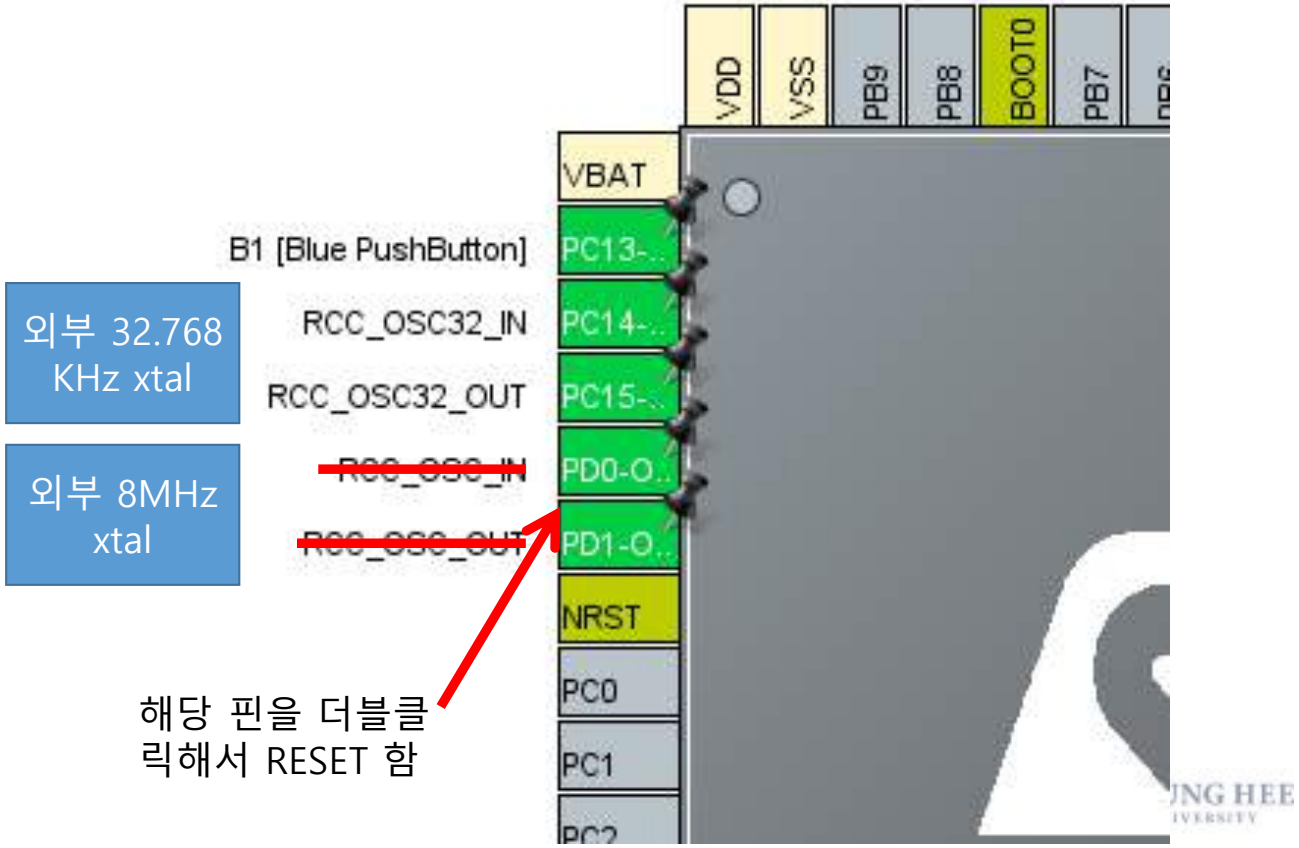
녹색으로 표시된 핀은 할당된 것
회색으로 표시된 핀은 할당안 된 것
연한 노란색 핀은 전원핀

7.4 외부 인터럽트 CubeMX과제

CubeMX로 예제 7.1 구현하기

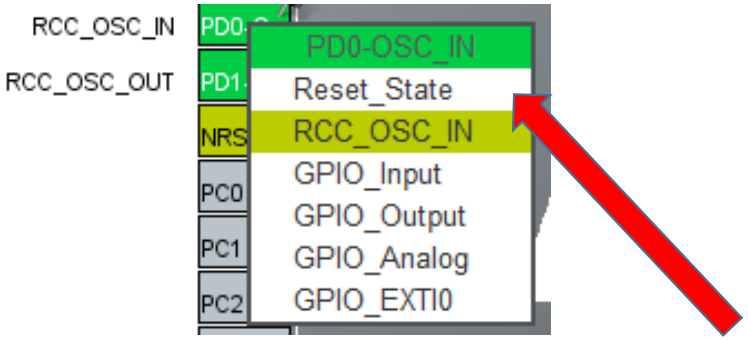
우리가 실습하는 nucleo 보드에는 외부에 추가할 수 있는 xtal 2개 중 RTC 용 32.768KHz 만 장착됨. 빠르고 정확한 동작을 위해서는 8MHz xtal을 달면 됨 (https://www.devicemart.co.kr/goods/view?no=7265)

따라서 8MHz xtal 연결 핀은 해제하고, 32.768KHz xtal 연결 핀은 남겨 놓음(이것도 선택사항임)



7.4 외부 인터럽트 CubeMX과제

CubeMX로 예제 7.1 구현하기



각 핀을 더블클릭하면 선택할 수 있는 옵션이 나옴. 핀 마다 모두 다르므로 주의

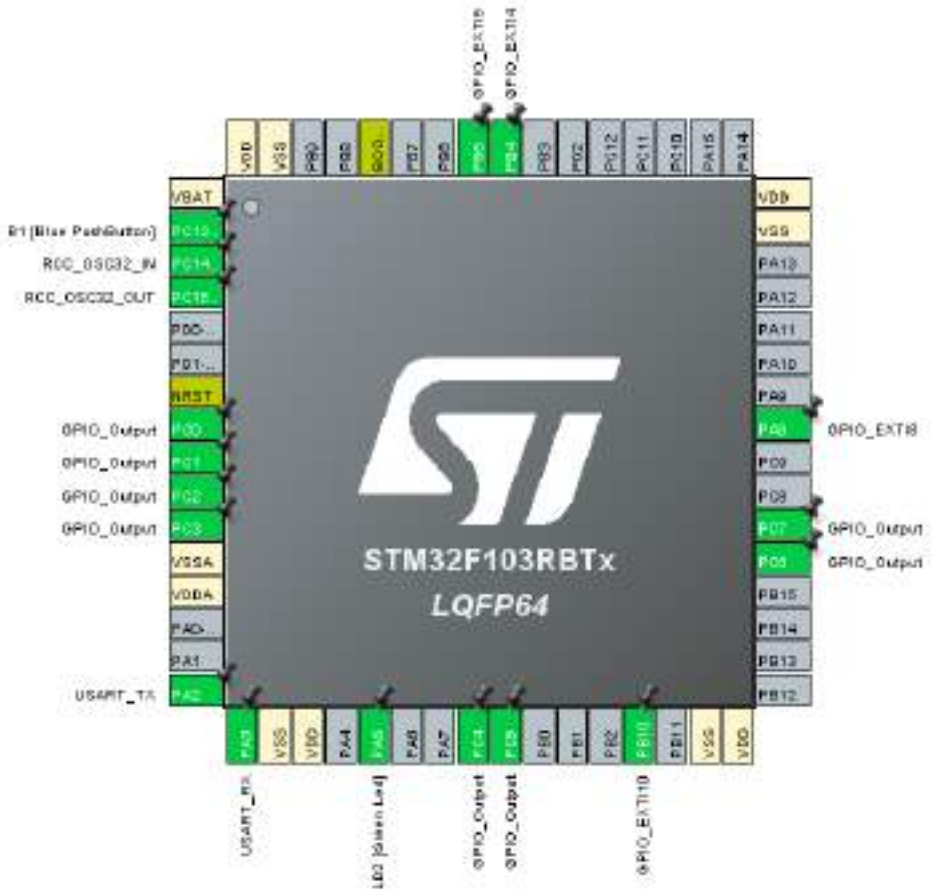
출력 : 6장 과제와 동일(PC 0~7)
입력 : PA8,PB4,PB5,PB10에 대해서 EXTI입력으로 변경

스위치	GPIO 포트, 핀	발생되는 EXTI
B1(User Button)	GPIOC 13	EXTI15_10
SW1	GPIOA 8	EXTI9_5
SW2	GPIOB 4	EXTI4
SW3	GPIOB 5	EXTI9_5
SW4	GPIOB 10	EXTI5_10

KYUNG HEE UNIVERSITY

7.4 외부 인터럽트 CubeMX과제

CubeMX로 예제 7.1 구현하기



출력 : 6장 과제와 동일(PC 0~7)
입력 : PA8,PB4,PB5,PB10에 대해서 EXTI입력으로 변경

스위치	GPIO 포트, 핀	발생되는 EXTI
B1(User Button)	GPIOC 13	EXTI15_10
SW1	GPIOA 8	EXTI9_5
SW2	GPIOB 4	EXTI4
SW3	GPIOB 5	EXTI9_5
SW4	GPIOB 10	EXTI5_10

- 추가적으로 I/O 확장보드의 LED 8개를 PC0~PC7로 연결함
- 추가적으로 nucleo 보드에 있는 녹색 LED를 PA5에 연결, 파란색 USER 버튼을 PC13에 연결시킴
- 다음 슬라이드의 그림 참조해서 확인

KYUNG HEE UNIVERSITY

참고. Nucleo-F103 보드와 I/O 확장보드

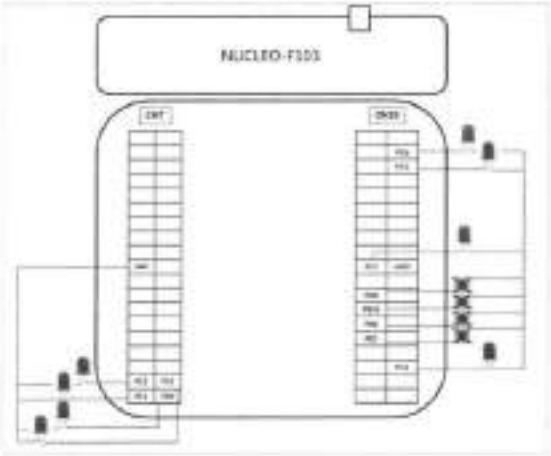
[연결용 회로도]

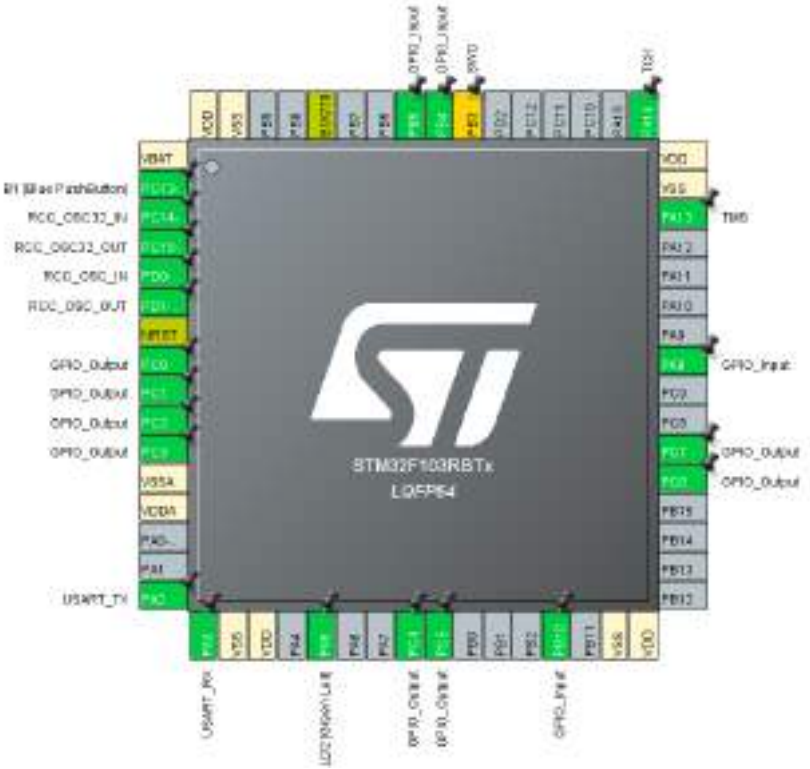
NUCLEO-F103RB 보드의 외부 연결용 커넥터(CN7, CN10)를 이용하여 LED 8개와 스위치 4개를 NUCLEO-F103RB 보드에 연결한다. 연결할 보드와 핀 번호는 다음과 같다.

- LED 1 ~ LED 8 : 각각 Port C의 0번 핀 ~ 7번 핀(PC0 ~ PC7)에 연결
- SW 1 ~ SW 4 : 각각 Port A의 8번 핀, Port B의 4번, 5번, 10번 핀(PA8, PB4, PB5, PB10)에 연결

참고

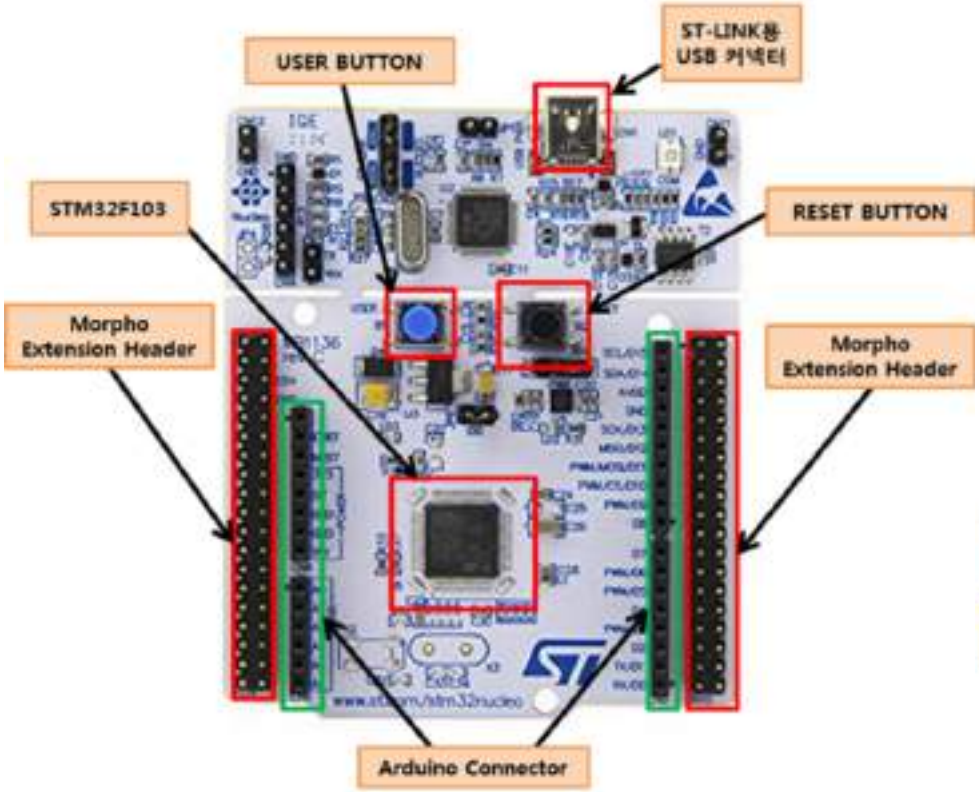
이 연결은 Nucleo-64용 I/O 보드의 회로도(그림 3-3-3(a))에 나타난 LED 1 ~ LED 8과 SW 1 ~ SW 4의 연결과 보드 및 핀 번호가 동일하다.



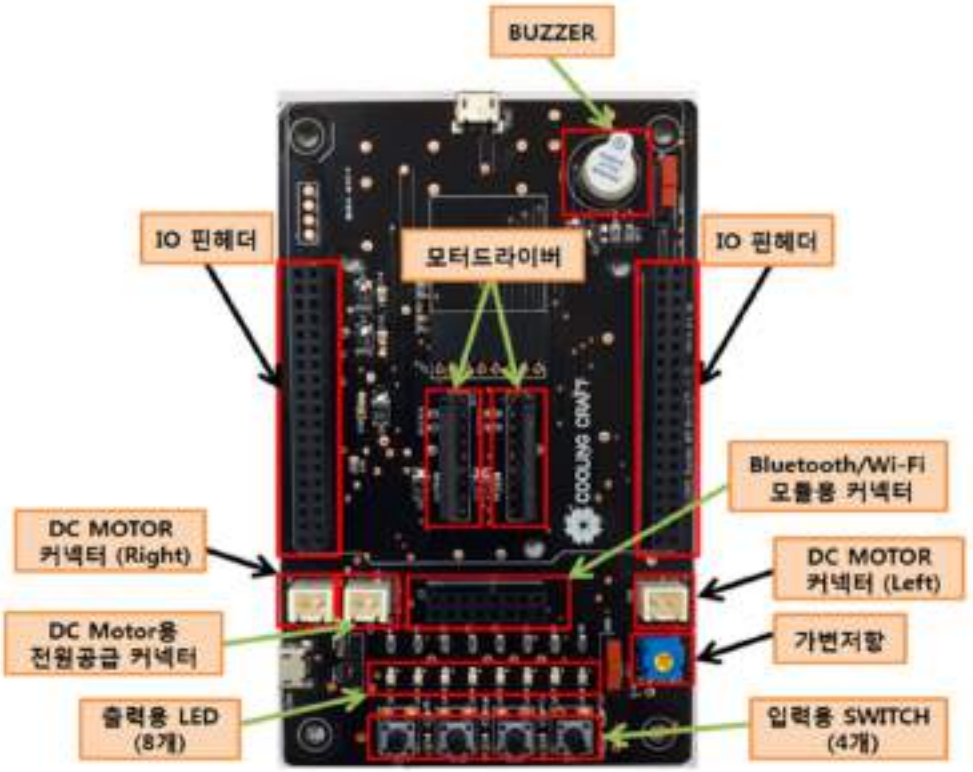


KYUNG HEE UNIVERSITY

참고. Nucleo-F103 보드와 I/O 확장보드



STM Nucleo-64 보드

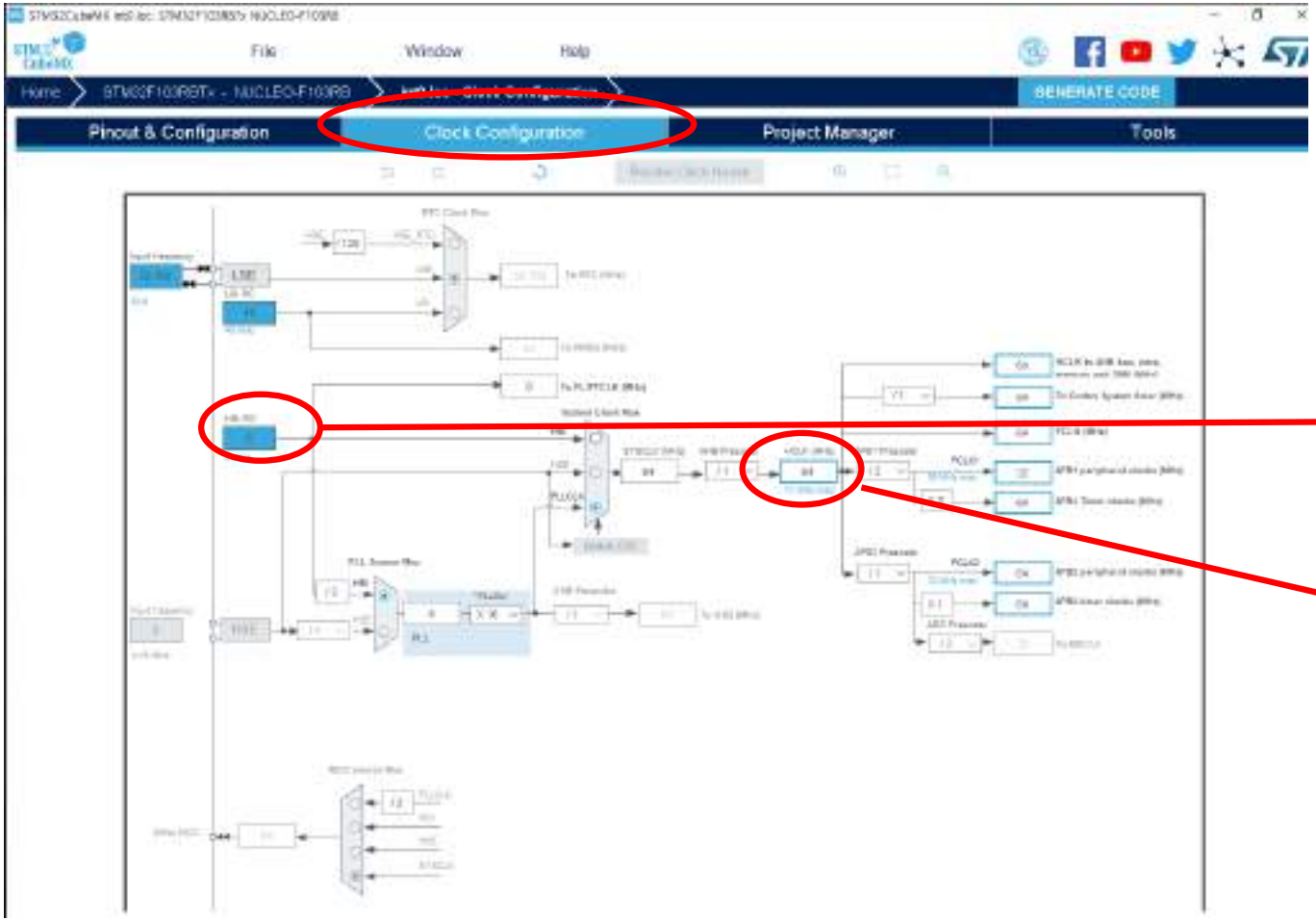


Nucleo-64용 I/O 보드

KYUNG HEE UNIVERSITY

7.4 외부 인터럽트 CubeMX과제

CubeMX로 예제 7.1 구현하기



- 앞 슬라이드의 핀 할당 후 제일 먼저 해야 할 일은 clock 설정임
- clock 소스는 HSE인 8MHz xtal, LSE인 32.768KHz xtal, HIS인 8MHz RC 클럭이 있음
- 이 중에서 32.768KHz xtal은 현재 연결되어 있지만, RTC용이므로 사용하지 않고, 8MHz HIS RC 클럭을 사용함(짙은 파란색 표시)
- 8MHz를 PLL로 빠르게 해서 최대 64MHz로 속도를 올림

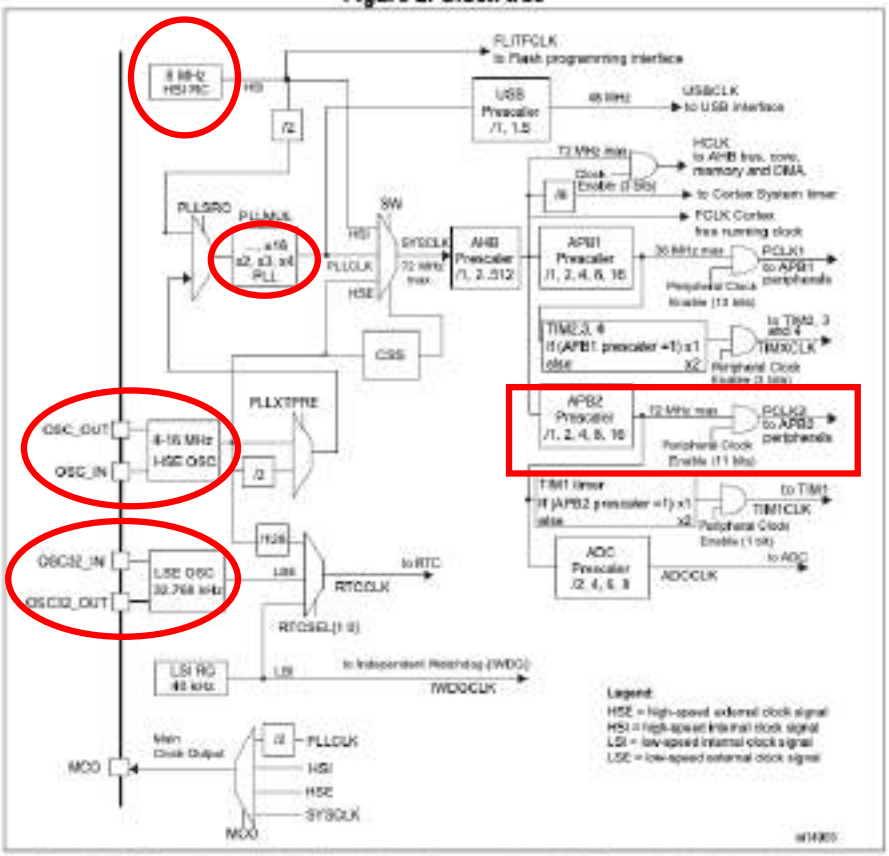
KYUNG HEE UNIVERSITY

참조. GPIO(Genaral Purpose I/O) 의 구조 및 기능

주요기능 1. GPIO(General Purpose I/O : 범용 입출력)

- GPIO 핀이 입력으로 설정된 경우에 입력 데이터 레지스터(Input data register)는 매 APB2 클럭(72MHz)마다 I/O 핀에서 데이터를 입력 받음
- 핀이 출력으로 설정된 경우에 출력 데이터 레지스터(Output data register)에 저장된 값이 I/O 핀에 출력됨
- 출력 모드에서만 출력 드라이버(Output driver)의 사용이 가능

Figure 2. Clock tree



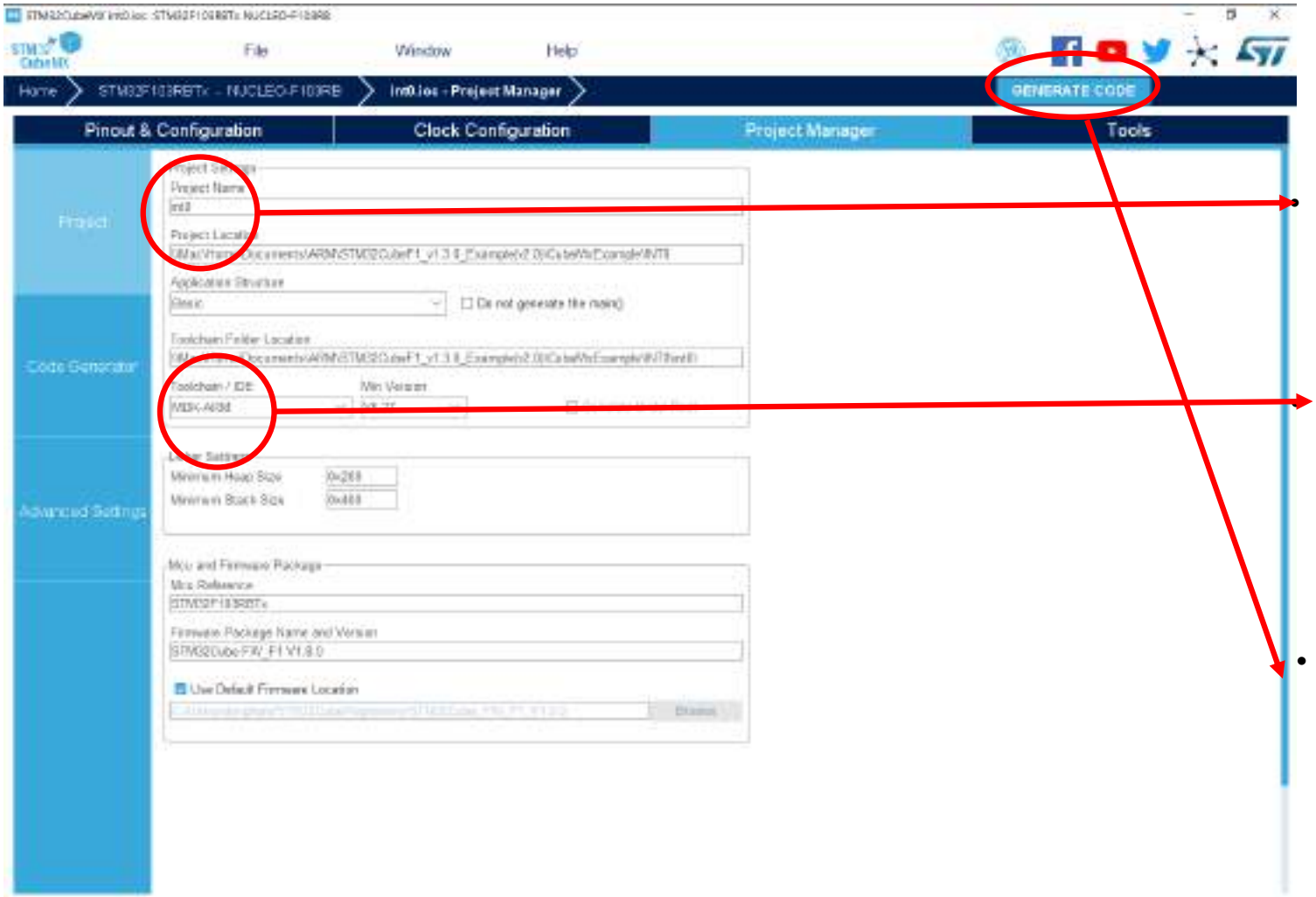
1. When the HSI is used as a PLL clock input, the maximum system clock frequency that can be achieved is 64 MHz.
2. For the USB function to be available, both HSE and PLL must be enabled, with USBCLK running at 48 MHz.
3. To have an ADC conversion time of 1 µs, APB2 must be at 14 MHz, 28 MHz or 56 MHz.

7.4 외부 인터럽트 CubeMX과제

CubeMX로 예제 7.1 구현하기

- Pinout & Configuration 탭으로 가면 왼쪽 윈도우에 카테리고가 나옴
- 이 중에서 System Core를 선택하면 GPIO가 보임
- 이때 오른쪽 윈도우에서 NVIC를 선택해서 외부인터럽트(EXTI)를 enabled 시킴

7.4 외부 인터럽트 CubeMX과제

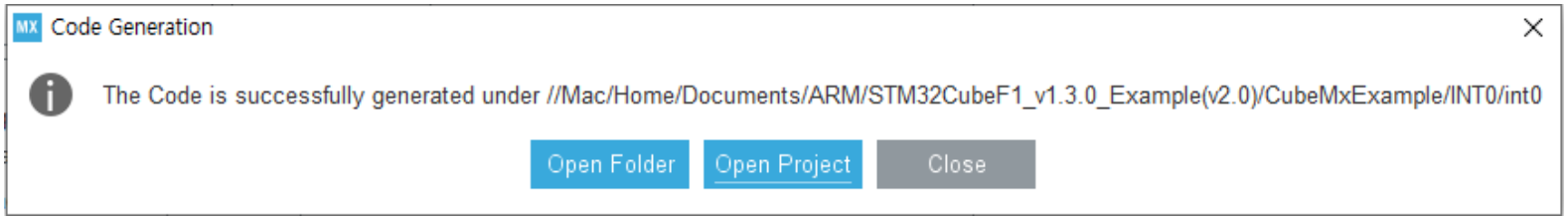


- Project Manager 선택 후, 프로젝트 이름과 경로 선택
- MDK-ARM을 꼭 선택해야 함
- 마지막으로 GENERATE CODE 누름

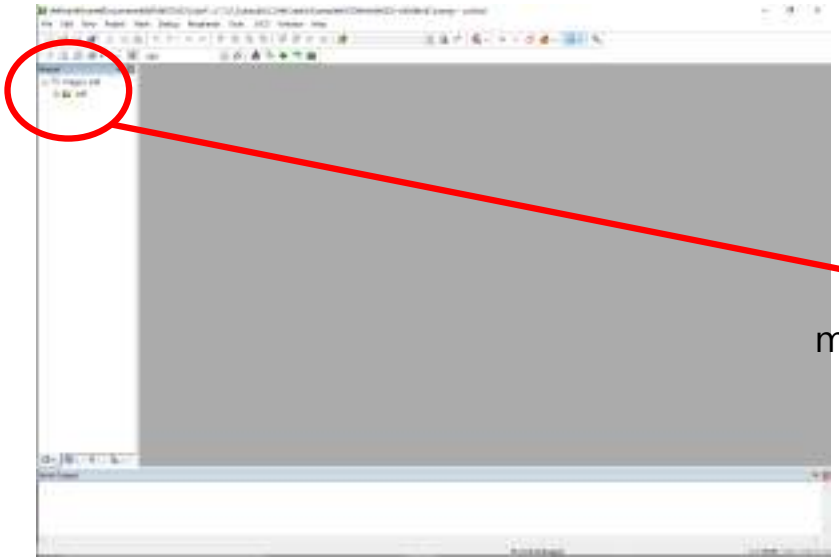
KYUNG HEE UNIVERSITY

7.4 외부 인터럽트 CubeMX과제

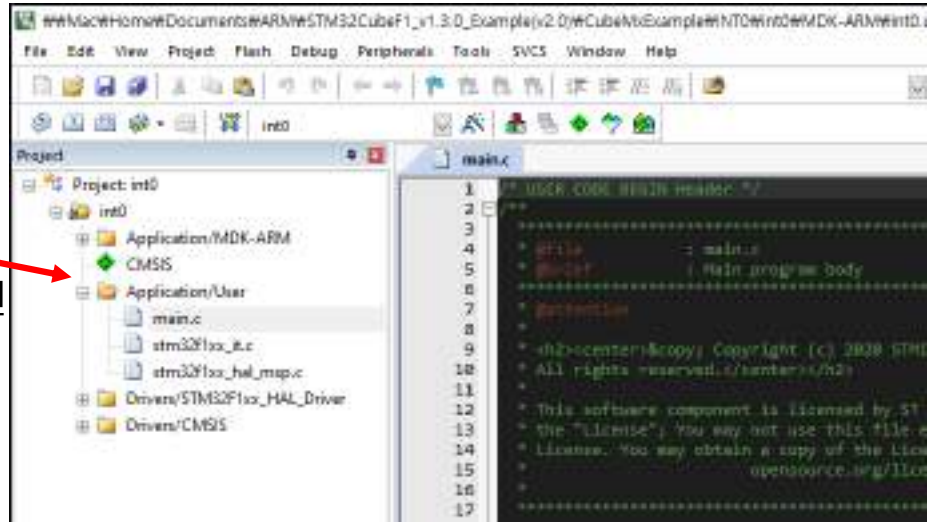
CubeMX로 예제 7.1 구현하기



Open Project 누르면, Keil uVision 실행됨



main.c 확인



7.4 외부 인터럽트 CubeMX과제

CubeMX와 Keil uVision 연동시 주의사항

라이브러리 include, 변수 선언, define, 함수 선언 등은 정해진 위치에만 해야함

```
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */

/* USER CODE END PV */
```

```
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
```

KYUNG HEE UNIVERSITY

7.4 외부 인터럽트 CubeMX과제

```
/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */
int flag_sw1, flag_sw2, flag_sw3, flag_sw4 = 0;
int SW1,SW2,SW3,SW4 = 0;

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{

  /* Prevent unused argument(s) compilation warning */
  if (GPIO_Pin == GPIO_PIN_8) flag_sw1 =1;
  if (GPIO_Pin == GPIO_PIN_4) flag_sw2 =1;
  if (GPIO_Pin == GPIO_PIN_5) flag_sw3 =1;
  if (GPIO_Pin == GPIO_PIN_10) flag_sw4 =1;

  /* NOTE: This function Should not be modified, when the callback is needed,
           the HAL_GPIO_EXTI_Callback could be implemented in the user file
   */
}
/* USER CODE END PFP */
```

→ 전역변수 선언

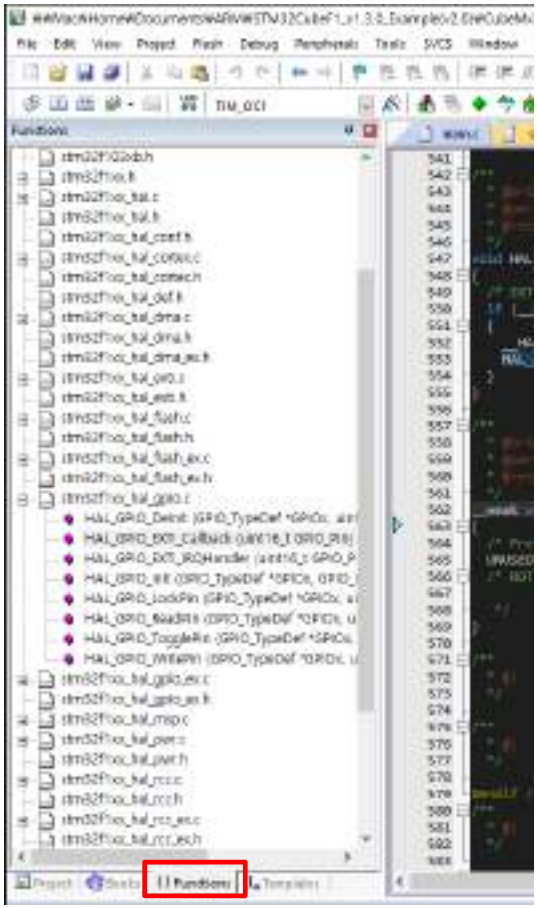
→ CubeMX 자동생성 부분

→ 외부 인터럽트 콜백함수

KYUNG HEE UNIVERSITY

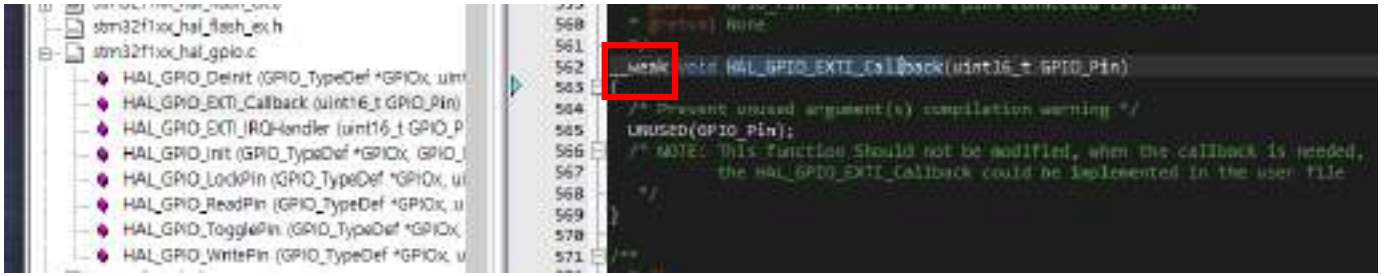
7.4 외부 인터럽트 CubeMX과제

CubeMX로 예제 7.1 구현하기



→ HAL 함수들은 MDK uVision의 왼쪽 윈도우에서 {} Function 탭을 열고 해당하는 c 라이브러리를 열면 찾을 수 있음

→ 예를 들어 앞 슬라이드의 HAL_GPIO_EXTI_Callback() 함수는 아래와 같이 찾을 수 있고 ___weak 부분을 제외하고 copy하면 됨

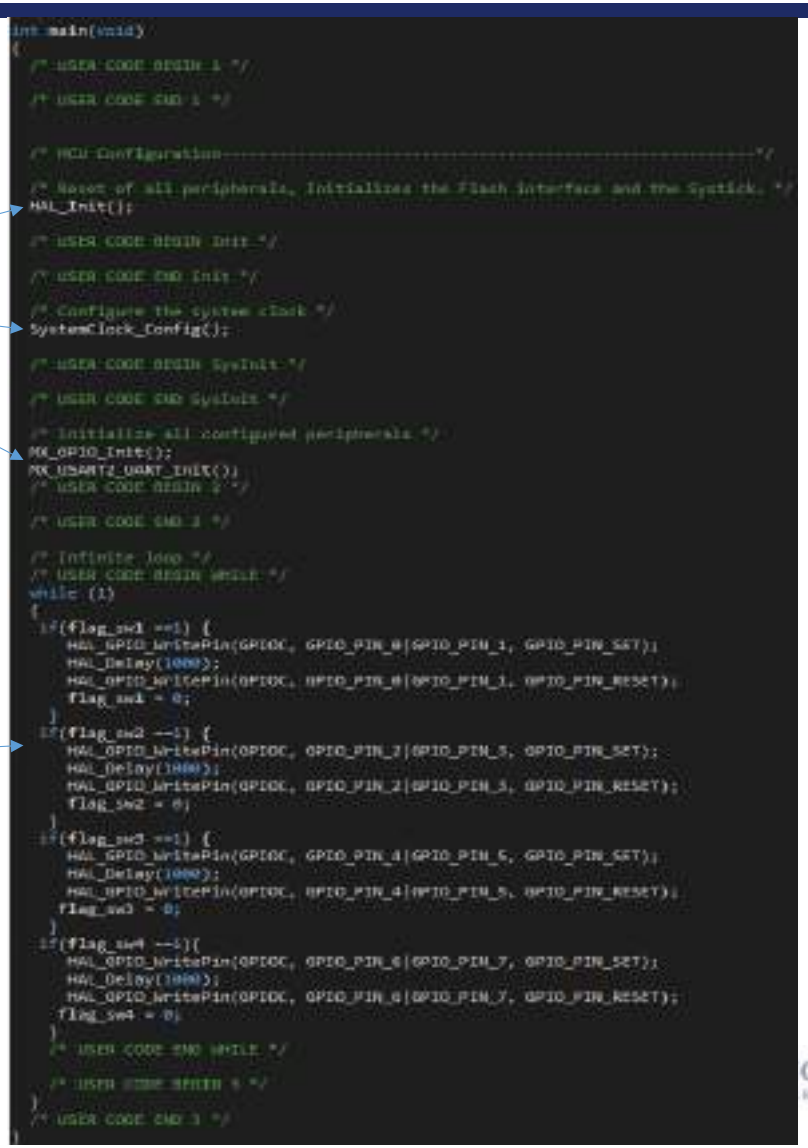


KYUNG HEE UNIVERSITY

7.4 외부 인터럽트 CubeMX과제

CubeMX가 자동생성

직접 코딩



7.4 외부 인터럽트 CubeMX과제

[과제 양식 안내]

1) Cubemx의 Pinout 그림 캡쳐 하여 첨부하고, 기능(ex.타이머,인터럽트)을 사용하였다면 기능에 대한 설정 창도 첨부
2) main문 코드와 기능에 대한 함수 코드 첨부하고 간단한 설명

→ 위 사항을 첨부하여 수업시간에 프린트하여 제출
→ 해당 과제 작동을 수업시간에 확인

7장 과제 → 예제 7.2, 7.3, 7.4를 Cubemx를 사용하여 프로그래밍 하기.

KYUNG HEE UNIVERSITY