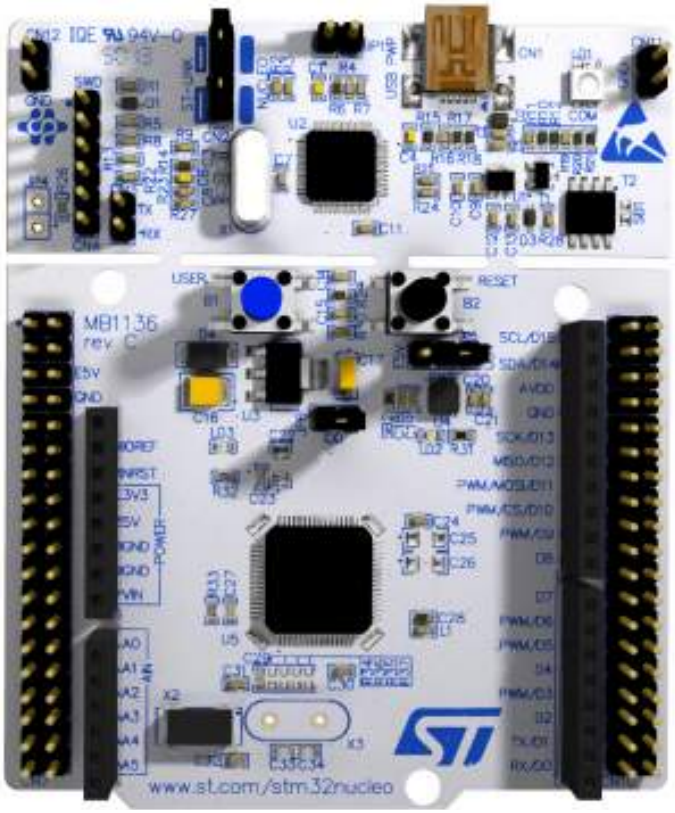
임베디드시스템설계
EMBEDDED SYSTEM DESIGN

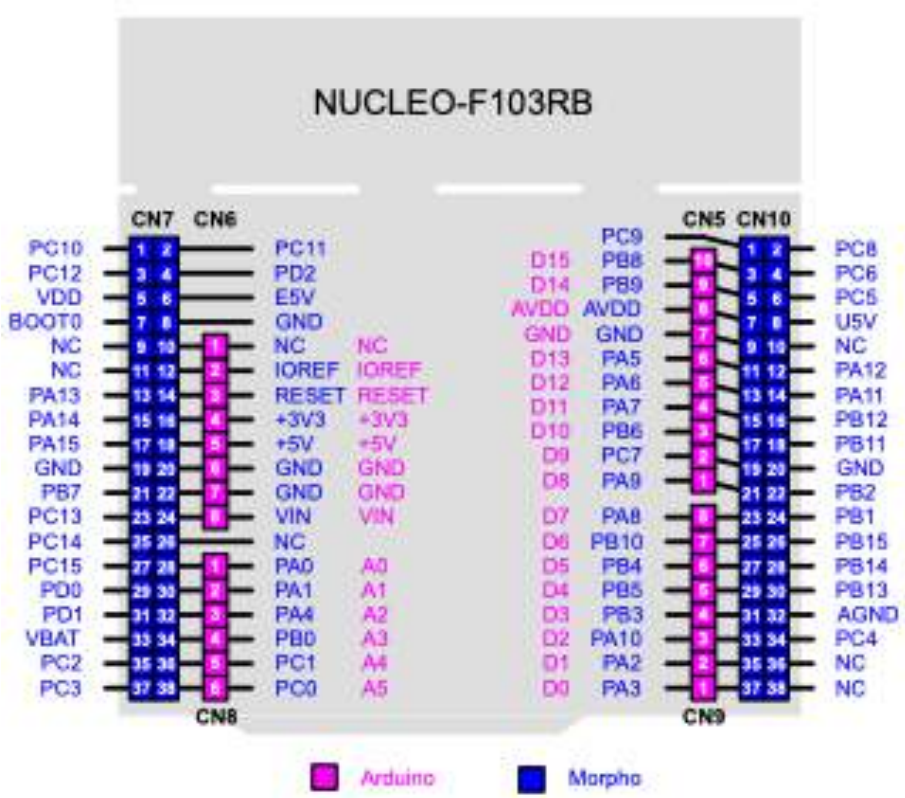
CHAPTER 06

GPIO를 이용한 입출력

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

Genaral Purpose I/O 의 구조

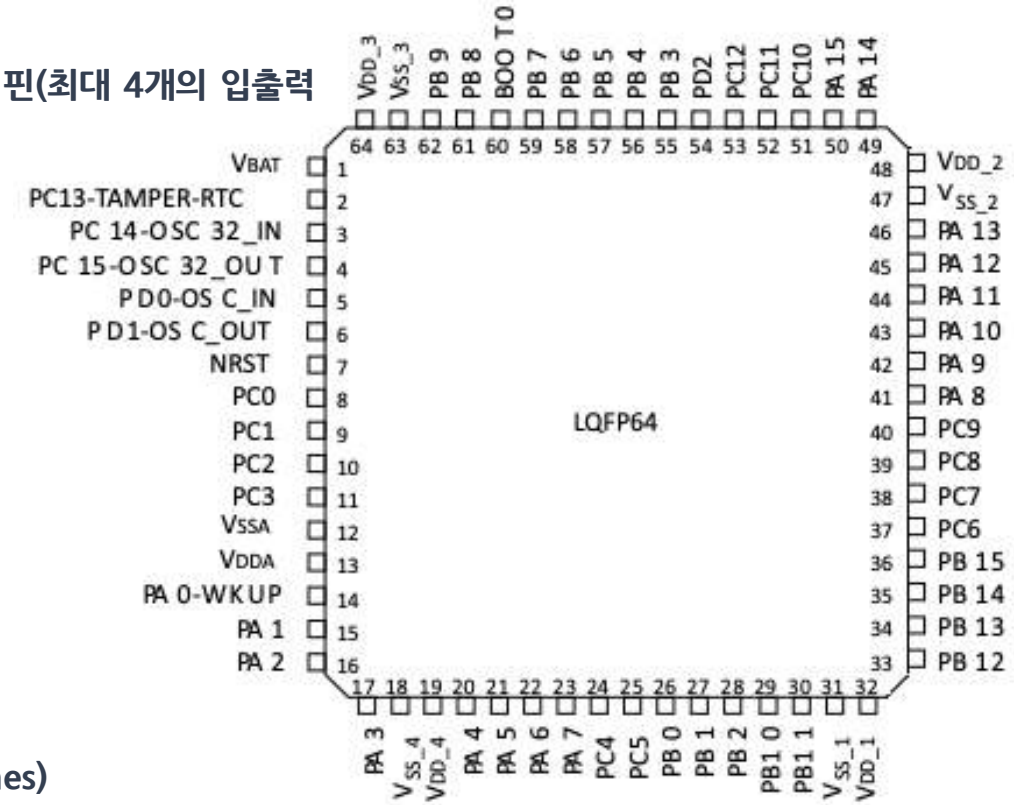




6.1 GPIO(Genaral Purpose I/O) 의 구조 및 기능

Genaral Purpose I/O의 구조

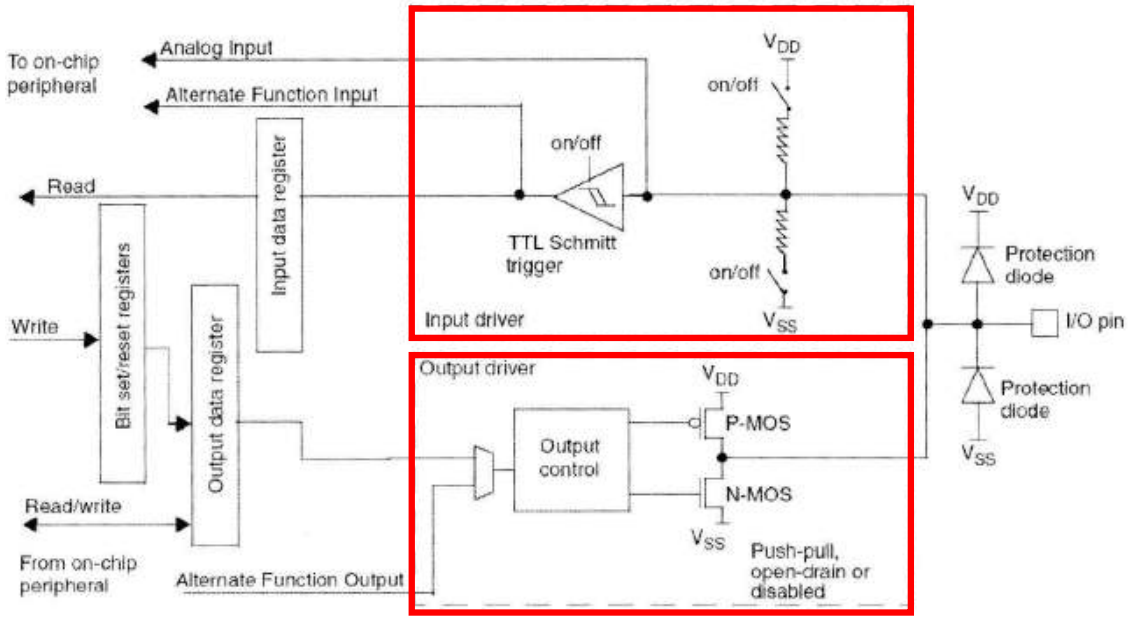
- STM32F103Rx는 오른쪽 그림과 같이 총 64핀 중에 51핀(최대 4개의 입출력 포트(PORT A ~ PORT C))이 GPIO임
- 입력 모드
  - 플로팅(Floating) 방식 입력
  - 풀업(Pull-up) 방식 입력
  - 풀다운(Pull-down) 방식 입력
  - 아날로그 입력
- 출력 모드
  - 오픈 드레인(Open drain) 방식 출력
  - 푸시-풀(Push-pull) 방식 출력
- 대체 기능(Open drain) 방식 출력
  - 오픈 드레인(Open drain) 방식 대체 기능
  - 푸시-풀(Push-pull) 방식 대체 기능
- 외부 인터럽트/이벤트 입력(External interrupt/event lines)



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

GPIO 핀의 구조

- **1개의 입출력 핀으로는 1비트의 입출력이 가능**



GPIO 핀의 기본 구조

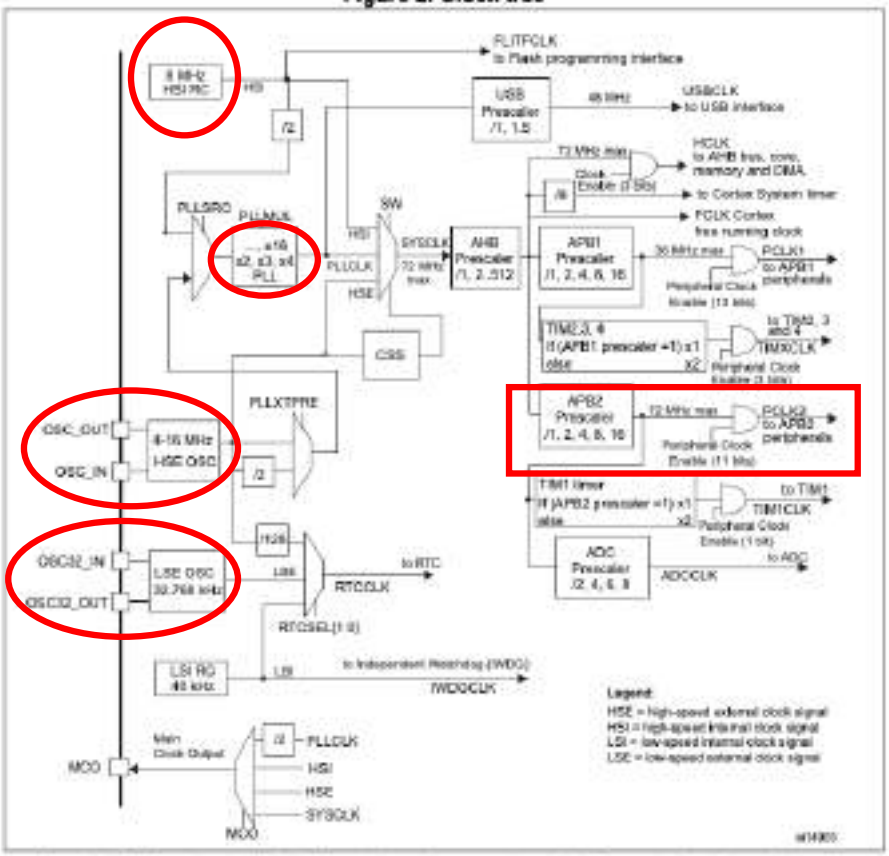
KYUNG HEE UNIVERSITY

6.1 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.

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

주요기능 2. 대체 기능(Alternate function)

- 대체 기능을 사용하기 위해서는 먼저 포트 비트 설정 레지스터(Port Bit Configuration Register)의 설정이 필요
- 대체 기능의 입력을 사용하려면 해당 포트가 입력 모드(floating, pull-up 또는 pull-down)로 설정되어 있어야 하고 입력 핀은 외부에서 구동되어야 함
- 대체 기능의 출력을 사용하려면 해당 포트의 비트가 대체기능 출력 모드(push-pull 또는 open-drain)로 설정되어 있어야 함
- 대체 기능 출력 모드인 경우 해당 핀은 출력 레지스터로는 연결되지 않으며 대응되는 주변장치의 출력 신호로 연결됨
- 양방향(입출력)의 대체 기능을 사용하려면 해당 포트의 비트가 대체가능 출력 모드(push-pull 또는 open-draing)로 설정되어 있어야 함
- 이 경우 입력 드라이버는 플로팅 입력 모드(input floating mode)로 설정됨

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

주요기능 3. 외부 인터럽트/이벤트 입력

- 모든 포트는 외부 인터럽트 또는 이벤트의 입력이 가능
- 외부 인터럽트 입력을 사용하기 위해서는 포트가 입력 모드로 설정되어 있어야 함
- 사용 가능한 모든 GPIO 핀은 EXTI 입력을 받을 수 있음

6.2 GPIO 구동용 HAL 드라이버

GPIO 설정용 구조체

- **GPIO의 동작조건을 설정하기 위한 구조체**

GPIO_InitTypeDef

GPIO의 초기 설정을 위한 구조체이며 stm32f1xx_hal_gpio.h에 정의되어 있다.

[데이터 형]

- uint32_t Pin
- uint32_t Mode
- uint32_t Pull
- uint32_t Speed

[데이터 형식 설명]

- Pin : 설정할 핀을 지정한다. 이 파라미터가 가질 수 있는 값은 다음과 같다.

GPIO_PIN_0	GPIO_PIN_1	GPIO_PIN_2
GPIO_PIN_3	GPIO_PIN_4	GPIO_PIN_5
GPIO_PIN_6	GPIO_PIN_7	GPIO_PIN_8
GPIO_PIN_9	GPIO_PIN_10	GPIO_PIN_11
GPIO_PIN_12	GPIO_PIN_13	GPIO_PIN_14
GPIO_PIN_15		
GPIO_PIN_All	GPIO_PIN_MASK	

- Mode : 핀의 동작 모드를 설정한다. 이 파라미터가 가질 수 있는 값은 다음과 같다.

GPIO_MODE_INPUT	입력 플로팅(Input Floating) 모드
GPIO_MODE_OUTPUT_PP	출력 푸시 풀(Output Push Pull) 모드
GPIO_MODE_OUTPUT_OD	출력 오픈 드레인(Output Open Drain) 모드
GPIO_MODE_AF_PP	대체 기능 푸시 풀(Alternate Function Push Pull) 모드
GPIO_MODE_AF_OD	대체 기능 오픈 드레인(Alternate Function Open Drain) 모드
GPIO_MODE_AF_INPUT	대체 기능 입력(Alternate Function Input) 모드
GPIO_MODE_ANALOG	아나로그(Analog) 모드
GPIO_MODE_IT_RISING	외부 인터럽트(External Interrupt)모드, 상승 에지(Rising edge)에 트리거(trigger)됨
GPIO_MODE_IT_FALLING	외부 인터럽트(External Interrupt)모드, 하강 에지(Falling edge)에 트리거(trigger)됨
GPIO_MODE_IT_RISING_FALLING	외부 인터럽트(External Interrupt)모드, 상승/하강 에지(Rising/Falling edge)에 트리거(trigger)됨
GPIO_MODE_EVT_RISING	이벤트(Event) 모드, 상승 에지(Rising edge)에 트리거(trigger)됨
GPIO_MODE_EVT_FALLING	이벤트(Event) 모드, 하강 에지(Falling edge)에 트리거(trigger)됨
GPIO_MODE_EVT_RISING_FALLING	이벤트(Event) 모드, 상승/하강 에지(Rising/Falling edge)에 트리거(trigger)됨

- Pull : 핀의 Pull-up, Pull-Down 모드를 설정한다. 이 파라미터가 가질 수 있는 값은 다음과 같다.

GPIO_NOPULL	No Pull-up or Pull-down activation
GPIO_PULLUP	Pull-up activation
GPIO_PULLDOWN	Pull-down activation

- Speed : 핀의 동작속도를 설정한다. 이 파라미터가 가질 수 있는 값은 다음과 같다.

GPIO_SPEED_FREQ_LOW	Low speed
GPIO_SPEED_FREQ_MEDIUM	Medium speed
GPIO_SPEED_FREQ_HIGH	High speed

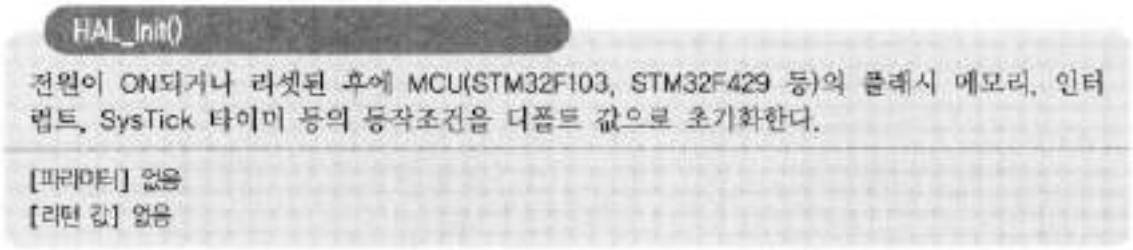
참고 위의 값은 STM32CubeF1 V1.3.0에서 변경된 것이며, V1.2.0까지는 핀의 동작속도 설정 파라미터를 다음과 같이 사용하였다.

GPIO_SPEED_LOW	Low speed
GPIO_SPEED_MEDIUM	Medium speed
GPIO_SPEED_HIGH	High speed

6.2 GPIO 구동용 HAL 드라이버

MCU의 초기화를 위한 함수

- 전원이 ON되거나 리셋된 후에 MCU(STM32F103, STM32F429 등)를 초기화하기 위한 함수
- 모든 프로그램의 처음에는 **반드시** 이 함수가 호출되어야 함
  - **HAL_Init()**
    - MCU의 초기화를 위한 함수
    - 이 함수를 실행하면 MCU가 리셋 이후의 디폴트 값으로 초기화됨
    - 이 함수는 HAL 라이브러리에서 제공하는 함수



KYUNG HEE UNIVERSITY

6.2 GPIO 구동용 HAL 드라이버

GPIO 구동용 함수의 종류

(1) 초기화(Initialization) 및 초기화 해제(de-initialization)용 함수
- HAL_GPIO_Init()
  - GPIO를 설정조건에 맞추어 초기화함
- HAL_GPIO_DeInit()
  - GPIO를 리셋시의 디폴트 값으로 설정(초기화 해제)함

(2) 입출력용 함수
- HAL_GPIO_ReadPin()
  - GPIO의 지정된 핀의 값을 읽어옴
- HAL_GPIO_WritePin()
  - GPIO의 지정된 핀을 0 또는 1로 설정함
- HAL_GPIO_TogglePin()
  - GPIO의 지정된 핀의 값을 토글(toggle)시킴
- HAL_GPIO_LockPin()
  - GPIO의 지정된 핀의 설정 값을 변경 금지(lock)시킴

6.2 GPIO 구동용 HAL 드라이버

GPIO 구동용 함수의 종류

(3) 인터럽트 처리용 함수

- HAL_GPIO_EXTI_IRQHandler()
  - EXTI 인터럽트의 핸들러 함수
- HAL_GPIO_EXTI_Callback()
  - EXTI 인터럽트를 처리하기 위한 콜백 함수
  - 이 함수는 EXTI 핸들러 함수(EXTIx_IRQHandler()) 내에서 사용됨

6.2 GPIO 구동용 HAL 드라이버

구동용 함수의 사용 방법

1) _HAL_RCC_GPIOx_CLK_ENABLE() 함수를 이용하려 사용하려는 GPIO의 클럭을 활성화

2) HAL_GPIO_Init() 함수를 이용하여 GPIO 핀의 동작조건을 설정

3) 외부 인터럽트/이벤트 모드를 선택한 경우는 다음을 설정
   HAL_NVIC_SetPriority() 함수를 이용하여 사용할 인터럽트(EXTI)의 우선순위를 설정
   HAL_NVIC_EnableRQ() 함수를 이용하여 인터럽트를 활성화

4) 입력모드에서 핀의 상태값을 읽으려면 HAL_GPIO_ReadPin() 함수를 사용

5) 출력모드에서 핀의 레벨을 설정하거나 토글하려면 HAL_GPIO_WritePin(), HAL_GPIO_TogglePin() 함수를 사용

6) 다음 번 리셋까지 핀의 설정을 유지시키려면 HAL_GPIO_LockPin() 함수를 사용

6.2 GPIO 구동용 HAL 드라이버

GPIO 구동용 함수

HAL_GPIO_Init (GPIO_TypeDef * GPIOx, GPIO_InitTypeDef * GPIO_Init)

GPIOx를 GPIO_Init 구조체 변수의 설정 값에 맞추어 초기화한다.

[파라미터]
- GPIOx : GPIO의 이름. (x는 A..K 까지의 값을 가질 수 있다)
- GPIO_Init : GPIO의 설정 값을 가지고 있는 GPIO_InitTypeDef 구조체형의 변수

[리턴 값] 없음

HAL_GPIO_DeInit (GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

GPIOx를 리셋시의 디폴트 값으로 설정(초기화 해제)한다.

[파라미터]
- GPIOx : GPIO의 이름. (x는 A..K 까지의 값을 가질 수 있다)
- GPIO_Pin : GPIO pin을 지정하며, GPIO_PIN_0 – GPIO_PIN_15 사이의 값을 가질 수 있다.

[리턴 값] 없음

HAL_GPIO_ReadPin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin)

GPIOx의 지정된 핀(GPIO_Pin)에 입력된 값을 읽어온다.

[파라미터]
- GPIOx : GPIO의 이름. (x는 A..K 까지의 값을 가질 수 있다)
- GPIO_Pin : GPIO pin을 지정하며, GPIO_PIN_0 – GPIO_PIN_15 사이의 값을 가질 수 있다.

[리턴 값]
- 지정된 핀의 입력 값

HAL_GPIO_WritePin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)

GPIOx의 지정된 핀(GPIO_Pin)을 0, 또는 1로 설정한다.

[파라미터]
- GPIOx : GPIO의 이름. (x는 A..K 까지의 값을 가질 수 있다)
- GPIO_Pin : GPIO pin을 지정하며, GPIO_PIN_0 – GPIO_PIN_15 사이의 값을 가질 수 있다.
- PinState : 핀의 상태를 설정한다. 이 파라미터가 가질 수 있는 값은 다음과 같다.
  GPIO_PIN_RESET : 핀을 리셋(0) 한다.
  GPIO_PIN_SET : 핀을 셋(1) 한다.

[리턴 값] 없음

6.2 GPIO 구동용 HAL 드라이버

GPIO 구동용 함수

HAL_GPIO_TogglePin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin)

GPIOx의 지정된 핀(GPIO_Pin)의 값을 토글(toggle)시킨다.

[파라미터]
- GPIOx : GPIO의 이름. (x는 A..K 까지의 값을 가질 수 있다)
- GPIO_Pin : GPIO pin을 지정하며, GPIO_PIN_0 ~ GPIO_PIN_15 사이의 값을 가질 수 있다.

[리턴 값] 없음

HAL_GPIO_LockPin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin)

GPIOx의 지정된 핀(GPIO_Pin)의 값을 록(Lock) 시킨다.

[파라미터]
- GPIOx : GPIO의 이름. (x는 A..K 까지의 값을 가질 수 있다)
- GPIO_Pin : GPIO pin을 지정하며, GPIO_PIN_0 ~ GPIO_PIN_15 사이의 값을 가질 수 있다.

[리턴 값] 없음

참고 핀이 록(Lock)되면 리셋되기 전까지는 그 핀의 값을 변경할 수 없다.

HAL_GPIO_EXTI_IRQHandler (uint16_t GPIO_Pin)

GPIO에서 발생하는 EXTI 인터럽트 처리를 위한 공백함수이다. 본 교재의 예제에서 이 함수는 stm32l10x_it.c 파일내의 인터럽트 핸들러 함수(EXTIx_IRQHandler())에서 호출되어 사용된다.

[파라미터]
- GPIO_Pin : GPIO pin을 지정하며, GPIO_PIN_0 ~ GPIO_PIN_15 사이의 값을 가질 수 있다.

[리턴 값] 없음

HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin)

EXTI 인터럽트를 처리하기 위한 공백 함수이다. 본 교재의 예제에서 이 함수는 main.c 파일내에서 구현되어 사용된다.

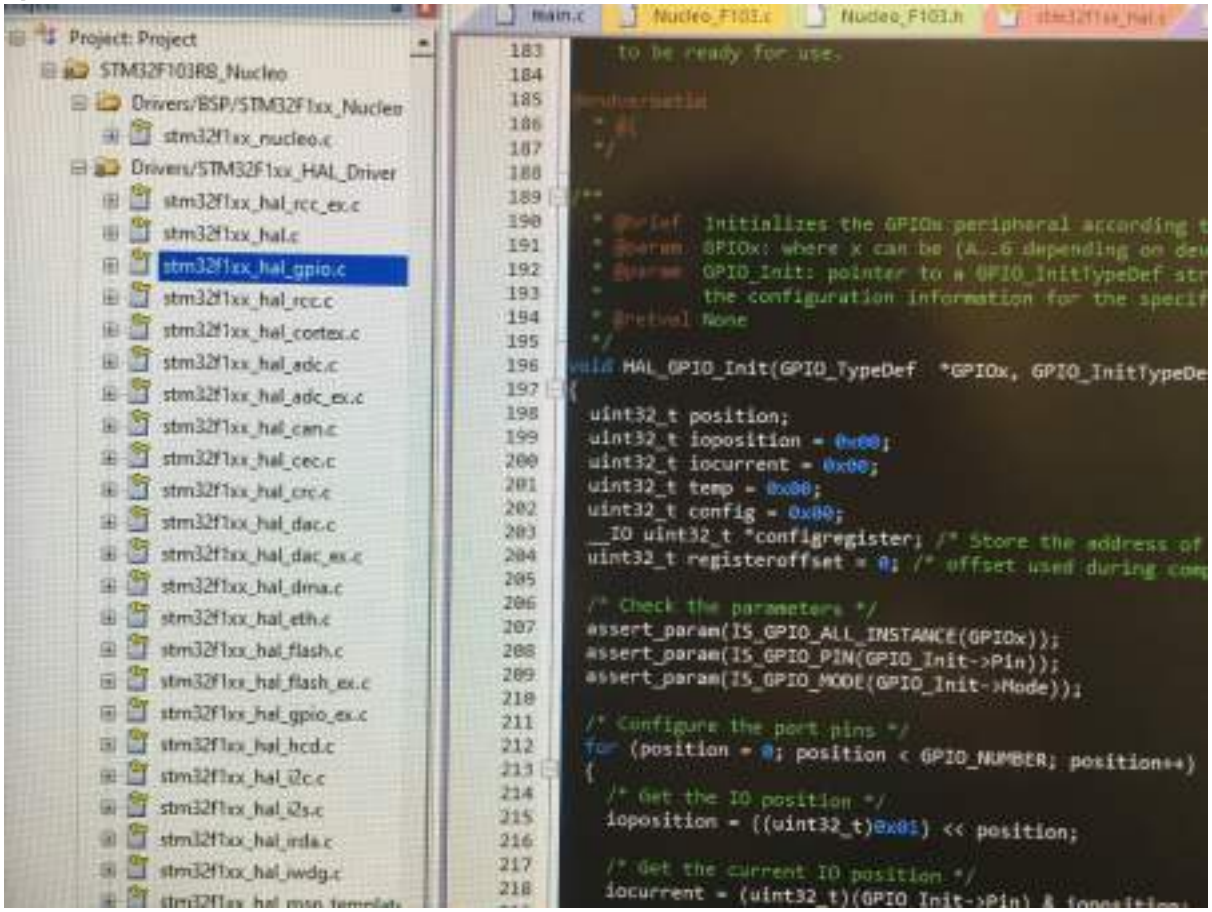
[파라미터]
- GPIO_Pin : GPIO pin을 지정하며, GPIO_PIN_0 ~ GPIO_PIN_15 사이의 값을 가질 수 있다.

[리턴 값] 없음

KYUNG HEE UNIVERSITY

6.2 GPIO 구동용 HAL 드라이버

실습보드 구동용 함수



KYUNG HEE UNIVERSITY

6.3 GPIO 응용 예제

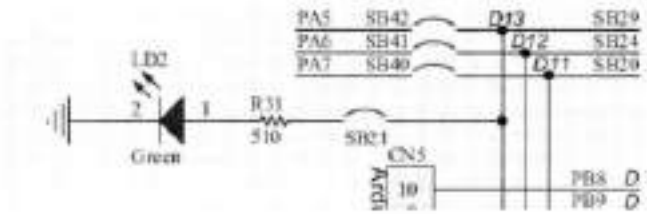
GPIO 예제 관련 회로도

- Nucleo-F103 확장보드에는 출력용 LED와 입력용 스위치가 다음과 같이 있음
  - Nucleo-64 보드(Nucleo-F103RB 보드) : 사용자 출력용 LED 1개, 입력용 스위치 1개
  - Nucleo-64용 I/O 보드 : 사용자 출력용 LED 8개, 입력용 스위치 4개

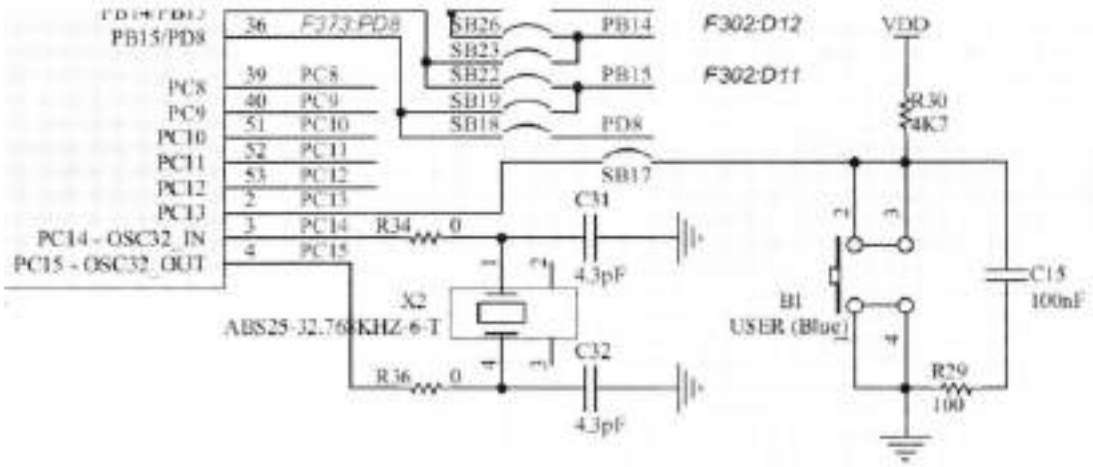
보드	구분	회로도 상의 기호	포트 및 핀 번호
Nucleo-64 보드 (NUCLEO-F103RB)	LED	LD2	PA5
	S/W	B1(User Button, 파랑색)	PC13
Nucleo-64용 I/O 보드	LED	LED1 ~ LED8	PC0 ~ PC7
	S/W	SW1 ~ SW4	PA8, PB4, PB5, PB10

6.3 GPIO 응용 예제

GPIO 예제 관련 회로도



(a) LED1(LD2)의 회로도



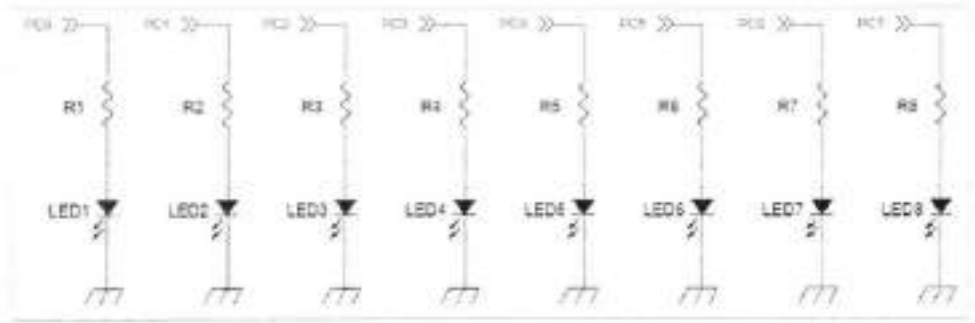
(b) SW1(B1)의 회로도

Nucleo-64 보드(Nucleo-F103RB 보드)의 입출력 관련 회로도

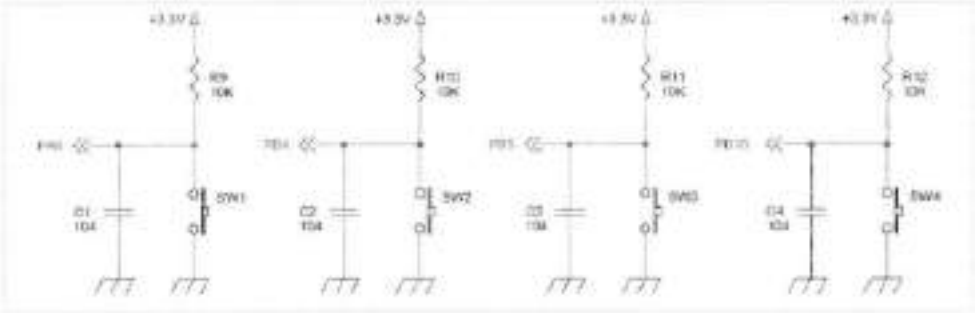
KYUNG HEE UNIVERSITY

6.3 GPIO 응용 예제

GPIO 예제 관련 회로도



(a) LED(LED1 ~ LED8)의 회로도



(b) SW(SW1 ~ SW4)의 회로도

Nucleo-64용 I/O 보드의 입출력 관련 회로도

- 2개 보드 모두 출력핀이 High이면 대응되는 LED가 ON이 되며 반대로 Low이면 OFF 됨을 알 수 있음
- 스위치는 풀업 방식이므로 스위치를 누르면 대응되는 입력 핀이 Low가 되며, 반대로 스위치를 떼면 High가 됨을 알 수 있음

6.3 GPIO 응용 예제

GPIO 예제 1 : LED를 깜빡이기

- 폴더 명 : [Examples_Nucleo F103] - [GPIO] - [GPIO 1_F103] - [MDK-ARM]
- 위의 폴더에 있는 [Project.uvprojx] 파일을 더블클릭하여 실행

6.3 GPIO 응용 예제

GPIO 예제 1 : LED를 깜빡이기

[main.c]

```
// -- <1> 프로그램의 수행에 필요한 헤더파일
#include "main.h"
#include "Nucleo_F103.h"              // Nucleo-F103 확장보드용 헤더파일


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

int main(void)
{
    // -- <2> MCU의 초기화 함수
    HAL_Init();
    // -- <3> system clock의 초기화 함수
    SystemClock_Config();
    // --<4>  실습보드의 출력용 LED의 초기화 함수
    LED_Config();

    // -- <5>  500msec 동안 LED를 모두 On한 후에 Off 하는 함수
    LED_OnOff(GPIO_PIN_LedAll, 500);
    // -- <6> 무한 루프로 동작
    while (1)  {
        // -- <7> I/O 보드의 LED(LED1~LED8)를 토글 모드로 동작
        HAL_GPIO_TogglePin(GPIOLedExt, GPIO_PIN_LedAll);

        // -- <8> Nucleo 보드의 LD2(Nucleo-64의 경우), 또는 LD1, LD2(Nucleo-144의 경우)를 토글
        모드로 동작
        HAL_GPIO_TogglePin(GPIOLedNucleo, GPIO_PIN_LedAll);

        // -- <9> 시간지연 함수
        HAL_Delay(200);
    }
}
```

LED 깜빡이기 예제 코드

6.3 GPIO 응용 예제

실습보드 구동용 함수의 종류 → 아래의 함수들은 책의 저자가 만든 함수임. 각 함수 내의 내용을 이해하는 것이 중요함.

- SystemClock_Config(void)
  - MCU의 system clock을 설정하기 위한 함수
- LED_Config(void)
  - 실습보드 내의 출력용 LED를 구동하기 위한 GPIO의 초기설정을 하는 함수
- LED_OnOff(int GPIO_PIN, int interval)
  - GPIO_PIN으로 정의된 LED를 모두 On하고 일정시간 후에 Off 시키는 함수
- Sw_Config(void)
  - 실습보드 내의 입력용 SW를 구동하기 위한 GPIO의 초기설정을 하는 함수

6.3 GPIO 응용 예제

실습보드 구동용 함수

SystemClock_Config(void);

MCU의 system clock을 설정하기 위한 함수이다. 이 함수는 Nucleo_F103.c (또는 Nucleo_F429.c)에 정의되어 있다

[파라미터] 없음
[리턴 값] 없음

LED_Config(void);

실습보드 내의 출력용 LED를 구동하기 위한 GPIO의 초기설정을 하는 함수이다. 이 함수는 Nucleo_F103.c (또는 Nucleo_F429.c)에 정의되어 있다.

[파라미터] 없음
[리턴 값] 없음

LED_OnOff(int GPIO_PIN, int interval);

GPIO_PIN으로 정의된 LED를 모두 On하고 일정시간(interval) 후에 Off 시키는 함수이다. 이 함수는 Nucleo_F103.c (또는 Nucleo_F429.c)에 정의되어 있다.

[파라미터]
- GPIO_Pin : GPIO pin을 지정하며, GPIO_PIN_0 ~ GPIO_PIN_15 사이의 값을 가질 수 있다.
- interval : LED가 On되어 있는 시간값(msec). LED는 On되고 interval 시간 후에 Off 된다.

[리턴 값] 없음

Sw_Config(void);

실습보드 내의 입력용 SW를 구동하기 위한 GPIO의 초기설정을 하는 함수이다. 이 함수는 Nucleo_F103.c (또는 Nucleo_F429.c)에 정의되어 있다.

[파라미터] 없음
[리턴 값] 없음

6.3 GPIO 응용 예제

GPIO 예제 1 : LED를 깜빡이기

- **소스 코드 주석 설명**

[main.c]

<1> 프로그램의 수행에 필요한 헤더파일을 인클루드한다. 필요한 헤더 파일은 다음과 같다.
- Nucleo-F103 확장보드의 경우 : "main.h"와 "Nucleo_F103.h"
- Nucleo-F429 확장보드의 경우 : "main.h"와 "Nucleo_F429.h"

<2> HAL_Init() 함수 : MCU의 초기화를 위한 함수이다. 이 함수를 실행하면 MCU가 리셋 이후의 디폴트 값으로 초기화된다. 이 함수는 HAL 라이브러리에서 제공하는 함수이다.

<3> SystemClock_Config() 함수 : MCU의 system clock을 설정하기 위한 함수이다. 이 함수는 Nucleo_F103.c (또는 Nucleo_F429.c)에 정의되어 있다. 이 함수에 대한 자세한 설명은 [부록1.4. 예제 소스코드 추가 설명 ; Nucleo_F103.c 및 Nucleo_F429.c]을 참고하기 바란다.

<4> LED_Config() 함수 : 실습보드 내의 출력용 LED를 구동하기 위한 GPIO의 초기설정을 하는 함수이다. 이 함수는 Nucleo_F103.c (또는 Nucleo_F429.c)에 정의되어 있다. 이 함수에 대한 자세한 설명은 [14.3절. 예제 소스코드 추가 설명 : Nucleo_F103.c 및 Nucleo_F429.c]을 참고하기 바란다.
** 이 함수에는 GPIO 출력핀을 설정하는 소스코드가 있으므로 반드시 찾아서 보기 바란다.

<5> LED_OnOff(GPIO_PIN_LedAll, 500) : LED를 모두 On하고 500mse 후에 Off 시키기 위해 이 함수를 호출한다. 이것은 프로그램의 실행 전에 LED의 정상동작 여부를 체크하기 위한 것이다. 이 함수는 Nucleo_F103.c (또는 Nucleo_F429.c)에 정의되어 있다. 이 함수에 대한 자세한 설명은 [14.3절. 예제 소스코드 추가 설명 : Nucleo_F103.c 및 Nucleo_F429.c]을 참고하기 바란다.

<6> while 문은 조건이 항상 참(1)이므로 무한 루프로 동작하게 된다.

<7> HAL_GPIO_TogglePin(GPIOLedExt, GPIO_PIN_LedAll) : Nucleo-64용 (또는, Nucleo-144용) I/O 보드의 모든 LED(LED1~LED8)를 토글 모드로 동작시키는 함수이다. "GPIOLedExt"는 Nucleo_F103.h (또는 Nucleo_F429.h)내에 정의되어 있으며, I/O 보드내의 LED의 GPIO 포트를 나타낸다.

이 함수가 실행되면 LED의 점등 상태가 바뀌게 된다. 즉 LED가 ON이면 OFF로 변하고, OFF이면 ON으로 변한다. 이 함수는 HAL 라이브러리에서 제공하는 함수이다.

<8> HAL_GPIO_TogglePin(GPIOLedNucleo, GPIO_PIN_LedAll) : Nucleo-64 보드 (또는, Nucleo-144 보드)의 모든 LED를 토글 모드로 동작시키는 함수이다. "GPIOLedNucleo"는 Nucleo_F103.h (또는 Nucleo_F429.h)내에 정의되어 있으며, Nucleo 보드의 출력용 LED의 GPIO 포트를 나타낸다. 이 함수는 HAL 라이브러리에서 제공하는 함수이다.

<9> HAL_Delay(200) : 이 함수는 시간지연 함수이다. 단위는 msec이므로 여기서는 200msec 만큼 시간지연을 한다. 이 함수는 HAL 라이브러리에서 제공하는 함수이다.

- **소스코드 <7>은 Nucleo-64용 I/O 보드의 LED를 On/Off 함**
- **소스코드 <8>은 Nucleo 보드의 LD2를 On/Off 함**
- **따라서 이 프로그램을 실행하면 Nucleo-64 보드와 I/O 보드에 있는 LED가 모두 동작**

6.3 GPIO 응용 예제

GPIO 예제 2 : LED를 순차적으로 점멸

- Nucleo-64용 I/O 보드의 8개의 LED(LED1~LED8)가 순차적으로 점멸하는 동작을 수행하는 프로그램

- 폴더 명 : [Examples_Nucleo F103] - [GPIO] - [GPIO 2_F103] - [MDK-ARM]
- 위의 폴더에 있는 [Project.uvprojx] 파일을 더블클릭하여 실행

6.3 GPIO 응용 예제

GPIO 예제 2 : LED를 순차적으로 점멸

[main.c]

```
// -- <1> 프로그램의 수행에 필요한 헤더파일
#include "main.h"
#include "Nucleo_F103.h"              // Nucleo-F103 확장보드용 헤더파일


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

int main(void)
{
    // -- <2> LED를 On/Off 하기 위한 변수 (0x01 = 0000 0001)
    unsigned char led = 0x01;

    // -- <3> MCU의 초기화 함수
    HAL_Init();
    // -- <4> system clock의 초기화 함수
    SystemClock_Config();
    // --<5>  실습보드의 출력용 LED의 초기화 함수
    LED_Config();

    // -- <6>  500msec 동안 LED를 모두 On한 후에 Off 하는 함수
    LED_OnOff(GPIO_PIN_ledAll, 500);

    // -- <7> 무한 루프로 동작
    while (1) {

        // -- <8> LED1 -> LED2  -> .. -> LED8 의 순서로 쉬프트하며 LED를  On/Off 함
        do {
            // -- <8-1> 변수 led에 해당하는 LED를 ON한 후에 다시 OFF 함
            LED_OnOff(led, 200);
            // -- <8-2> 변수 led를 왼쪽으로 쉬프트시키는 비트연산
            led = led << 1;
            led = led & 0xfe;
        } while (led != 0x80);        // -- <8-3> led = 0x80 (1000 0000) 될 때까지 동작

        // -- <9> LED8 -> LED7  -> .. -> LED1 의 순서로 쉬프트 시키며 On/Off 함
        do {
            // -- <9-1> 변수 led에 해당하는 LED를 ON한 후에 다시 OFF 함
            LED_OnOff(led, 200);
            // -- <9-2>  변수 led를 오른쪽으로 쉬프트시키는 비트연산
            led = led >> 1;
            led = led & 0x7f;
        } while (led != 0x01);        // --  <9-3> led = 0x01 (0000 0001) 될 때까지 동작
    }
}
```

LED 순차적으로 점멸 예제 코드

6.3 GPIO 응용 예제

GPIO 예제 2 : LED를 순차적으로 점멸

- **소스 코드 주석 설명**

[main.c]

<1> 프로그램의 수행에 필요한 헤더 파일은 앞의 [GPIO 예제 1]과 동일하다.
<2> LED를 On/Off 하기 위한 변수 unsigned char led를 선언하고 0x01로 초기화 한다.
<3>~<6> 이 부분은 [GPIO 예제 1]의 <2>~<5>과 동일한 과정이다.
<7> while 문은 조건이 항상 참(1)이므로 무한 루프로 동작하게 된다.
<8> 이 부분은 LED1 -> LED2 -> .. -> LED8 의 순서로 쉬프트하며 LED를 On/Off 하는 소스코드이다.
<8-1> 변수 led에 해당하는 LED를 ON하고 200msec 후에 OFF 시킨다. 이 do-while 문이 수행될 때는 led=0x01의 초기값을 가지므로 LED1이 On/Off 된다.
<8-2> 이 부분은 변수 led를 왼쪽으로 1비트만큼 쉬프트시키는 비트연산을 하는 부분이다.
<8-3> do-while 문은 led=0x80 (1000 0000)이 아닐때 까지 동작을 한다. 즉, led=0x40 (0100 0000)까지 동작을 하여 LED7까지 On/Off 되게 된다.
<9> 이 부분은 LED8 -> LED7 -> .. -> LED1 의 순서로 쉬프트하며 LED를 On/Off 하는 소스코드이다.
<8-1> 변수 led에 해당하는 LED를 ON하고 200msec 후에 OFF 시킨다. 이 do-while 문이 수행될 때는 led=0x80의 초기값을 가지므로 LED8이 On/Off 된다.
<8-2> 이 부분은 변수 led를 오른쪽으로 1비트만큼 쉬프트시키는 비트연산을 하는 부분이다.
<8-3> do-while 문은 led = 0x01이 아닐때 까지 동작을 한다. 즉, led=0x02 (0000 0010)까지 동작을 하여 LED2까지 On/Off 되게 된다.

6.3 GPIO 응용 예제

GPIO 예제 3 : 스위치 입력을 받아 LED를 On/Off하기 (1)

- **입력용 스위치 SW1을 누르면 확장보드의 LED1~8이 On되고, 일정 시간 후에 Off 됨**

- **폴더 명 : [Examples_Nucleo F103] - [GPIO] - [GPIO 3_F103] - [MDK-ARM]**
- **위의 폴더에 있는 [Project.uvprojx] 파일을 더블클릭하여 실행**

KYUNG HEE UNIVERSITY

6.3 GPIO 응용 예제

GPIO 예제 3 : 스위치 입력을 받아 LED를 On/Off하기 (1)

[main.c]

```
// -- <1> 프로그램의 수행에 필요한 헤더파일
#include "main.h"
#include "Nucleo_F103.h"              // Nucleo-F103 확장보드용 헤더파일

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

int main(void)
{
    // -- <2> LED를 On/Off 하기 위한 변수  (0x01 = 0000 0001)
    uint16_t led = 0x01;

    // -- <3> MCU의 초기화 함수
    HAL_Init();
    // -- <4> system clock의 초기화 함수
    SystemClock_Config();
    // -- <5>  실습보드의 출력용 LED의 초기화 함수
    LED_Config();
    // --<6>  실습보드의 입력용 Switch의 초기화 함수
    Sw_Config();
    LED_OnOff(GPIO_PIN_LedAll, 500);
    // -- <7> 무한 루프로 동작
    while (1) {
        // -- <8>  변수 led 를 0으로 초기화
        led = 0x00;

        // -- <9> I/O 보드의 SW1 이 눌러지면 -> led 전체 bit 을 1로 설정 (0xFF = 1111 1111)
        if( HAL_GPIO_ReadPin(GPIOSw1, GPIO_PIN_Sw1 ) == RESET )  led = led | 0xFF;

        // -- <10> Nucleo 보드의 SW (B1)가 눌러지면 -> led 전체 bit 을 1로 설정 (0xFF = 1111 1111)
        // -- <10-1> Nucleo-64 보드의 경우는 이 코드를 실행한다.
        if( HAL_GPIO_ReadPin(GPIOSwNucleo, GPIO_PIN_Nucleo_Sw ) == RESET )  led = led | 0xFF;
        // -- <10-2> Nucleo-144 보드의 경우는 이 코드를 실행한다.
        // if( HAL_GPIO_ReadPin(GPIOSwNucleo, GPIO_PIN_Nucleo_Sw ) == SET )  led = led | 0xFF;

        // -- <11> 변수 led에 해당하는 LED를 ON한 후에 다시 OFF 함
        LED_OnOff(led, 300);
    }
}
```

스위치 입력을 받아 LED On/Off 예제 (1) 코드

6.3 GPIO 응용 예제

GPIO 예제 3 : 스위치 입력을 받아 LED를 On/Off하기 (1)

- **소스 코드 주석 설명**

[main.c]

<1> 프로그램의 수행에 필요한 헤더파일은 앞의 [GPIO 예제 1]과 동일하다.
<2> LED를 On/Off 하기 위한 변수 uint16_t led를 선언하고 0x01로 초기화 한다.
참고 [GPIO 예제 2]에서는 8비트의 unsigned char형으로 변수 led를 선언하였으나, 이 예제에서는 16비트의 uint16_t형으로 선언하였다. 어느 형으로 선언하여도 예제의 수행에는 문제가 없다.

<3>~<5> 이 부분은 [GPIO 예제 1]의 <2>~<4>과 동일한 과정이다.
<6> Sw_Config() : 실습보드 내의 입력용 SW를 구동하기 위한 GPIO의 초기설정을 하는 함수이다. 이 함수는 Nucleo_F103.c (또는 Nucleo_F429.c)에 정의되어 있다. [14.3절. 예제 소스코드 추가 설명 ; Nucleo_F103.c 및 Nucleo_F429.c]를 참고하기 바란다.
** 이 함수에는 GPIO 입력핀을 설정하는 소스코드가 있으므로 반드시 찾아서 보기 바란다.

<7> while 문은 조건이 항상 참(1)이므로 무한 루프로 동작하게 된다.
<8> 변수 led 를 0으로 초기화해준다.
<9> I/O 보드의 SW1이 눌러지면 led 전체 bit를 1로 설정한다. (0xFF = 1111 1111)
<10> Nucleo-64 (또는 Nucleo-144) 보드의 SW (B1)가 눌러지면 led 전체 bit를 1로 설정한다. 그런데 SW (B1)가 눌러지면 Nucleo-64 보드의 경우 RESET이 되고, Nucleo-144 보드의 경우는 SET이 된다. 따라서
<11> LED_OnOff(led, 300) : 변수 led에 해당하는 LED를 일정시간 ON한 후에 OFF 한다.

- **소스코드 〈10-1〉, 〈10-2〉는 Nucleo-64 보드의 SW 입력을 받기 위한 소스코드**
- **이 프로그램을 실행하면 Nucleo-64 보드의 SW를 눌러도 LED가 On/Off 됨**

KYUNG HEE UNIVERSITY

6.3 GPIO 응용 예제

GPIO 예제 4 : 스위치 입력을 받아 LED를 On/Off하기 (2)

- 입력용 스위치 SW1~SW4를 누르면 이에 대응하는 LED가 On/Off 되는 예제
- 스위치를 누르면 아래에 표시한 LED가 On되고, 일정 시간 후에 Off 됨

```
SW 1    : LED1, 2가 On/Off
SW 2    : LED3, 4가 On/Off
SW 3    : LED5, 6이 On/Off
SW 4    : LED7, 8이 On/Off
```

- 폴더 명 : [Examples_Nucleo F103] - [GPIO] - [GPIO 4_F103] - [MDK-ARM]
- 위의 폴더에 있는 [Project.uvprojx] 파일을 더블클릭하여 실행

KYUNG HEE UNIVERSITY

6.3 GPIO 응용 예제

GPIO 예제 4 : 스위치 입력을 받아 LED를 On/Off하기 (2)

[main.c]

```
// -- <1> 프로그램의 수행에 필요한 헤더파일
#include "main.h"
#include "Nucleo_F103.h"            // Nucleo-F103 확장보드용 헤더파일
//#include "Nucleo_F429.h"          // Nucleo-F429 확장보드용 헤더 파일

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

int main(void)
{
    // -- <2> LED를 On/Off 하기 위한 변수  (0x01 = 0000 0001)
    uint16_t led = 0x01;

    // -- <3> MCU의 초기화 함수
    HAL_Init();
    // -- <4> system clock의 초기화 함수
    SystemClock_Config();
    // --<5>  실습보드의 출력용 LED의 초기화 함수
    LED_Config();
    // --<6>  실습보드의 입력용 Switch의 초기화 함수
    Sw_Config();

    LED_OnOff(GPIO_PIN_LedAll, 500);

    // -- <7> 무한 루프로 동작
    while (1) {
        // -- <8>  변수 led 를 0으로 초기화
        led = 0x00;

        // -- <9-1> SW1 이 눌러지면 -> led 의 1, 2번째 bit 을 1로 설정 (0x03 = 0000 0011)
        if( HAL_GPIO_ReadPin(GPIOSw1, GPIO_PIN_Sw1 ) ==  RESET )  led = led | 0x03;

        // -- <9-2> SW2 이 눌러지면 -> led 의 3, 4번째 bit 을 1로 설정 (0x0c = 0000 1100)
        if( HAL_GPIO_ReadPin(GPIOSw2, GPIO_PIN_Sw2 ) ==  RESET )  led = led | 0x0c;

        // -- <9-3> SW3 이 눌러지면 -> led 의 5, 6번째 bit 을 1로 설정  (0x30 = 0011 0000)
        if( HAL_GPIO_ReadPin(GPIOSw3, GPIO_PIN_Sw3 ) ==  RESET )  led = led | 0x30;

        // -- <9-4> SW4 이 눌러지면 -> led 의 7, 8번째 bit 을 1로 설정  (0xc0 = 1100 0000)
        if( HAL_GPIO_ReadPin(GPIOSw4, GPIO_PIN_Sw4 ) ==  RESET )  led = led | 0xc0;

         // -- <10> 변수 led에 해당하는 LED를 ON한 후에 다시 OFF 함
        LED_OnOff(led, 100);
    }
}
```

스위치 입력을 받아 LED On/Off 예제 (2) 코드

6.3 GPIO 응용 예제

GPIO 예제 4 : 스위치 입력을 받아 LED를 On/Off하기 (2)

- **소스 코드 주석 설명**

[main.c]

```
<1> 프로그램의 수행에 필요한 헤더파일은 앞의 [GPIO 예제 1]과 동일하다.
<2> LED를 On/Off 하기 위한 변수 uint16_t led를 선언하고 0x01로 초기화 한다.
<3>~<5> 이 부분은 [GPIO 예제 1]의 <2>~<4>과 동일한 과정이다.
<6> Sw_Config() : 실습보드 내의 입력용 SW를 구동하기 위한 GPIO의 초기설정을 하는 함수이다. 이 함수는
    Nucleo_F103.c (또는 Nucleo_F429.c)에 정의되어 있다.
<7> while 문은 조건이 항상 참(1)이므로 무한 루프로 동작하게 된다.
<8> 변수 led 를 0으로 초기화해준다.
<9-1> SW1이 눌러지면 -> led 의 1, 2번째 bit 를 1로 설정 (0x03 = 0000 0011)
<9-2> SW2가 눌러지면 -> led 의 3, 4번째 bit 를 1로 설정 (0x0c = 0000 1100)
<9-3> SW3이 눌러지면 -> led 의 5, 6번째 bit 를 1로 설정 (0x30 = 0011 0000)
<9-4> SW4 이 눌러지면 -> led 의 7, 8번째 bit 를 1로 설정 (0xc0 = 1100 0000)
<10> 변수 led에 해당하는 LED를 ON한 후에 다시 OFF 한다.
```

- 위 예제는 main() 함수내의 무한 반복되는 while 루프 안에서 HAL_GPIO_ReadPin() 함수를 이용하여 스위치의 입력여부를 계속 폴링(polling)하며 체크함
- 따라서 스위치를 계속 누르고 있으면 해당되는 LED가 계속 On 되어있으며 스위치를 떼면 LED가 Off 됨
- 동시에 여러 개의 키를 눌러도 해당되는 LED가 모두 On됨

6.4 Nucleo_F103.c와 .h 파일은 누가 만들었나?

- Keil uVision에서 ARM 보드를 제어하는 프로그램을 만들기 위해서는
  1. 직접 프로젝트 생성하고 칩 선택, 환경 선택, 드라이버, 라이브러리 등을 설치
     1. 레지스터를 직접 제어 - 칩마다 다르므로 datasheet 를 보고 해당 레지스터를 직접 제어 : 가장 어렵지만 제일 많이 배움
     2. HAL 라이브러리 활용
  2. 다른 사람이 만든 라이브러리와 틀을 사용 (현재 교과서 방법) : 가장 쉽지만 배우는 것이 없음
  3. CubeMX와 HAL 라이브러리 활용 : 활용도가 가장 좋음
- 이번 학기는 2번과 3번을 병행해서 진행할 것임
- 교과서에 있는 예제(2번 방법)는 소스코드가 공개, 배포되어 있음
- CubeMX와 HAL 라이브러리 활용 방법은 교과서에 없으며, klas에 강의자료와 소스코드 배포함
- 실무에서는 CubeMX와 HAL 라이브러리가 많이 활용됨 → 연습 필수

6.4 Nucleo_F103.c와 .h 파일은 누가 만들었나?

- 예제 1을 다시 보면,

```
#include "main.h"
#include "Nucleo_F103.h"

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

     LED_OnOff(GPIO_PIN_LedAll, 500);
          while (1) {
               HAL_GPIO_TogglePin(GPIONucleo, GPIO_PIN_LedAll);
               HAL_GPIO_TogglePin(GPIOExt, GPIO_PIN_LedAll);
               HAL_Delay(100);
               }
}
```

다른 사람이 만든 함수

HAL 라이브러리에 있는 함수

6.4 Nucleo_F103.c와 .h 파일은 누가 만들었나?

- 예제 1에서 다른 사람이 만든 함수를 main.c에 다 넣으면,

```
#include "main.h"

#define GPIONucleo GPIOA
#define GPIOExt GPIOC
#define GPIO_PIN_Led1 GPIO_PIN_0
#define GPIO_PIN_Led2 GPIO_PIN_1
#define GPIO_PIN_Led3 GPIO_PIN_2
#define GPIO_PIN_Led4 GPIO_PIN_3
#define GPIO_PIN_Led5 GPIO_PIN_4
#define GPIO_PIN_Led6 GPIO_PIN_5
#define GPIO_PIN_Led7 GPIO_PIN_6
#define GPIO_PIN_Led8 GPIO_PIN_7
#define GPIO_PIN_LedAll GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN
_4 | GPIO_PIN_5 | GPIO_PIN_6 | GPIO_PIN_7

#define GPIOSwNucleo GPIOC
#define GPIOSw1 GPIOA
#define GPIOSw2 GPIOB
#define GPIOSw3 GPIOB
#define GPIOSw4 GPIOB
#define GPIO_PIN_Nucleo_Sw GPIO_PIN_13
#define GPIO_PIN_Sw1 GPIO_PIN_8
#define GPIO_PIN_Sw2 GPIO_PIN_4
#define GPIO_PIN_Sw3 GPIO_PIN_5
#define GPIO_PIN_Sw4 GPIO_PIN_10
#define TIMER2 TIM2


void LED_Config(void);
void LED_OnOff(int led, int interval);
void SystemClock_Config(void);

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

LED_Config();

LED_OnOff(GPIO_PIN_LedAll, 500);
while (1) {
HAL_GPIO_TogglePin(GPIONucleo, GPIO_PIN_LedAll);
HAL_GPIO_TogglePin(GPIOExt, GPIO_PIN_LedAll);
HAL_Delay(2000);
}
}
```

```
void LED_Config(void)
{
GPIO_InitTypeDef GPIO_Init_Struct;
__HAL_RCC_GPIOA_CLK_ENABLE();

GPIO_Init_Struct.Pin = GPIO_PIN_5;
GPIO_Init_Struct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_Init_Struct.Pull = GPIO_PULLUP;
GPIO_Init_Struct.Speed = GPIO_SPEED_FREQ_HIGH;
HAL_GPIO_Init(GPIOA, &GPIO_Init_Struct);
__HAL_RCC_GPIOC_CLK_ENABLE();

GPIO_Init_Struct.Pin = GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3 |
GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_6 | GPIO_PIN_7;
GPIO_Init_Struct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_Init_Struct.Pull = GPIO_PULLUP;
GPIO_Init_Struct.Speed = GPIO_SPEED_FREQ_HIGH;
HAL_GPIO_Init(GPIOC, &GPIO_Init_Struct);
}


void Sw_Config(void)
{
GPIO_InitTypeDef GPIO_Init_Struct;
__HAL_RCC_GPIOC_CLK_ENABLE();
GPIO_Init_Struct.Pin = GPIO_PIN_13;
GPIO_Init_Struct.Mode = GPIO_MODE_INPUT;
GPIO_Init_Struct.Pull = GPIO_PULLUP;
GPIO_Init_Struct.Speed = GPIO_SPEED_FREQ_HIGH;
HAL_GPIO_Init(GPIOC, &GPIO_Init_Struct);

__HAL_RCC_GPIOA_CLK_ENABLE();
GPIO_Init_Struct.Pin = GPIO_PIN_8;
GPIO_Init_Struct.Mode = GPIO_MODE_INPUT;
GPIO_Init_Struct.Pull = GPIO_PULLUP;
GPIO_Init_Struct.Speed = GPIO_SPEED_FREQ_HIGH;
HAL_GPIO_Init(GPIOA, &GPIO_Init_Struct);

__HAL_RCC_GPIOB_CLK_ENABLE();
GPIO_Init_Struct.Pin = GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_10;
GPIO_Init_Struct.Mode = GPIO_MODE_INPUT;
GPIO_Init_Struct.Pull = GPIO_NOPULL;
GPIO_Init_Struct.Speed = GPIO_SPEED_FREQ_HIGH
HAL_GPIO_Init(GPIOB, &GPIO_Init_Struct);
}


void LED_OnOff(int led, int interval)
{
HAL_GPIO_WritePin(GPIONucleo, led, GPIO_PIN_SET );
HAL_GPIO_WritePin(GPIOExt, led, GPIO_PIN_SET );
HAL_Delay(interval);
HAL_GPIO_WritePin(GPIONucleo, led, GPIO_PIN_RESET );
HAL_GPIO_WritePin(GPIOExt, led, GPIO_PIN_RESET );
}
```

```
void SystemClock_Config(void)
{
RCC_ClkInitTypeDef clkinitstruct = {0};
RCC_OscInitTypeDef oscinitstruct = {0};

oscinitstruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
oscinitstruct.HSEState = RCC_HSE_OFF;
oscinitstruct.LSEState = RCC_LSE_OFF;
oscinitstruct.HSIState = RCC_HSI_ON;
oscinitstruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
oscinitstruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
oscinitstruct.PLL.PLLState = RCC_PLL_ON;
oscinitstruct.PLL.PLLSource = RCC_PLLSOURCE_HSI_DIV2;
oscinitstruct.PLL.PLLMUL = RCC_PLL_MUL16;
if (HAL_RCC_OscConfig(&oscinitstruct)!= HAL_OK) {
while(1);
}

clkinitstruct.ClockType = (RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_HCLK | RCC_C
LOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2);
clkinitstruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
clkinitstruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
clkinitstruct.APB2CLKDivider = RCC_HCLK_DIV1;
clkinitstruct.APB1CLKDivider = RCC_HCLK_DIV2;
if (HAL_RCC_ClockConfig(&clkinitstruct, FLASH_LATENCY_2)!= HAL_OK) {
while(1);
}
}
```

6.5 CubeMX 설치 및 설정 방법

1) CubeMX설치방법

- https://www.st.com/en/development-tools/stm32cubemx.html **링크에서 설치 파일 다운로드 후 설치**

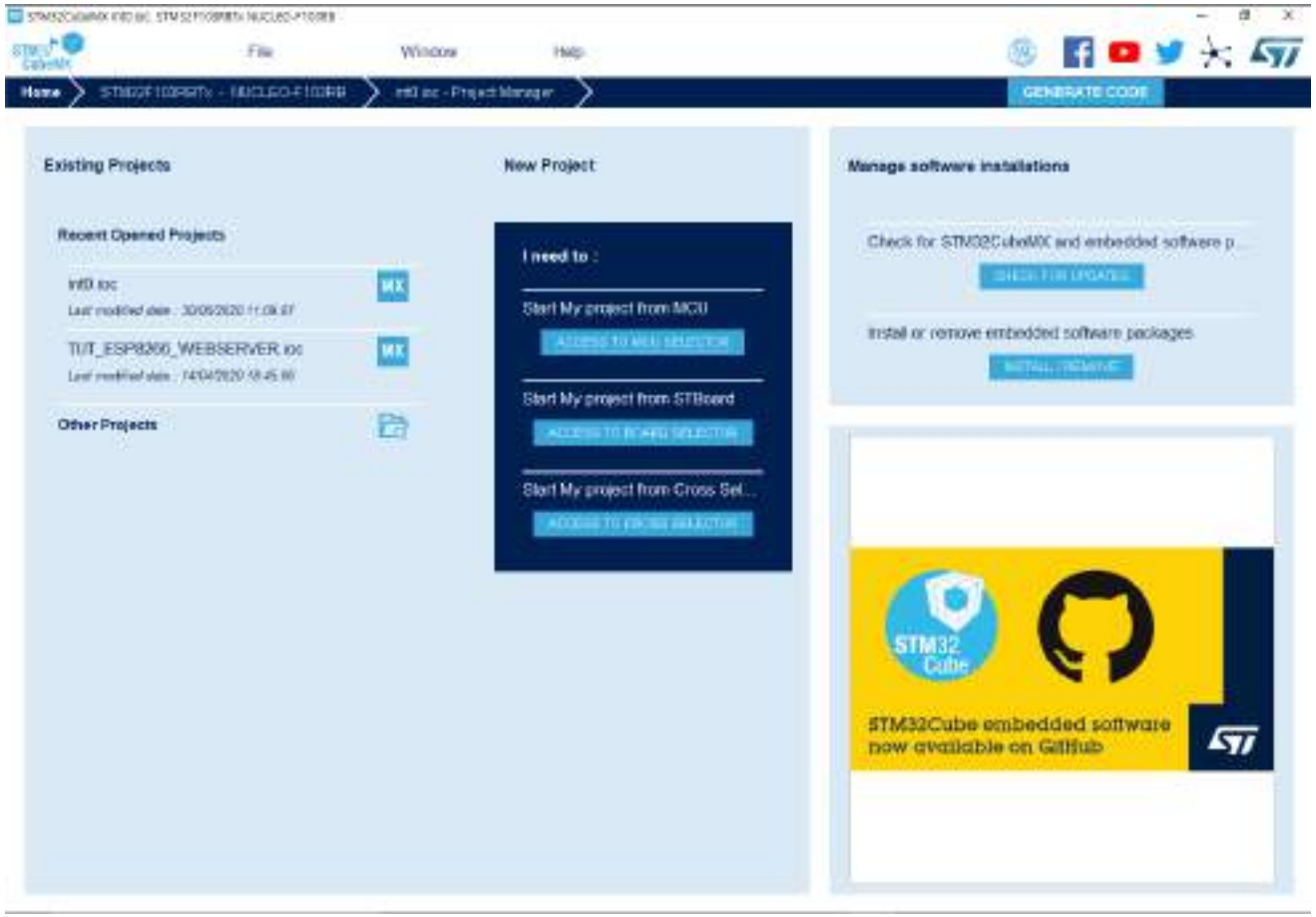
KYUNG HEE UNIVERSITY

7.4 외부 인터럽트 CubeMX과제

CubeMX로 예제 7.1 구현하기

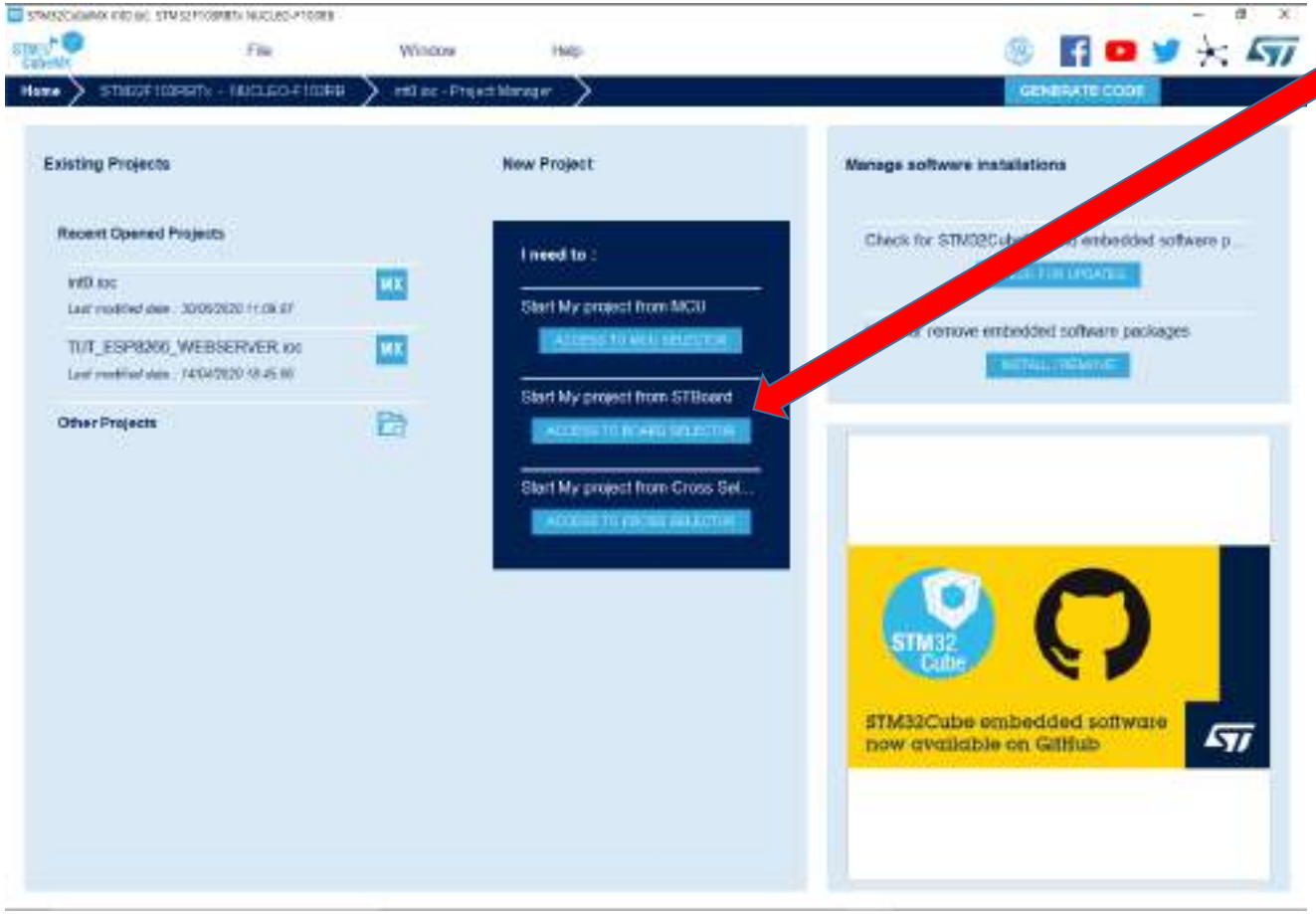
CubeMX 실행

초기화면



7.4 외부 인터럽트 CubeMX과제

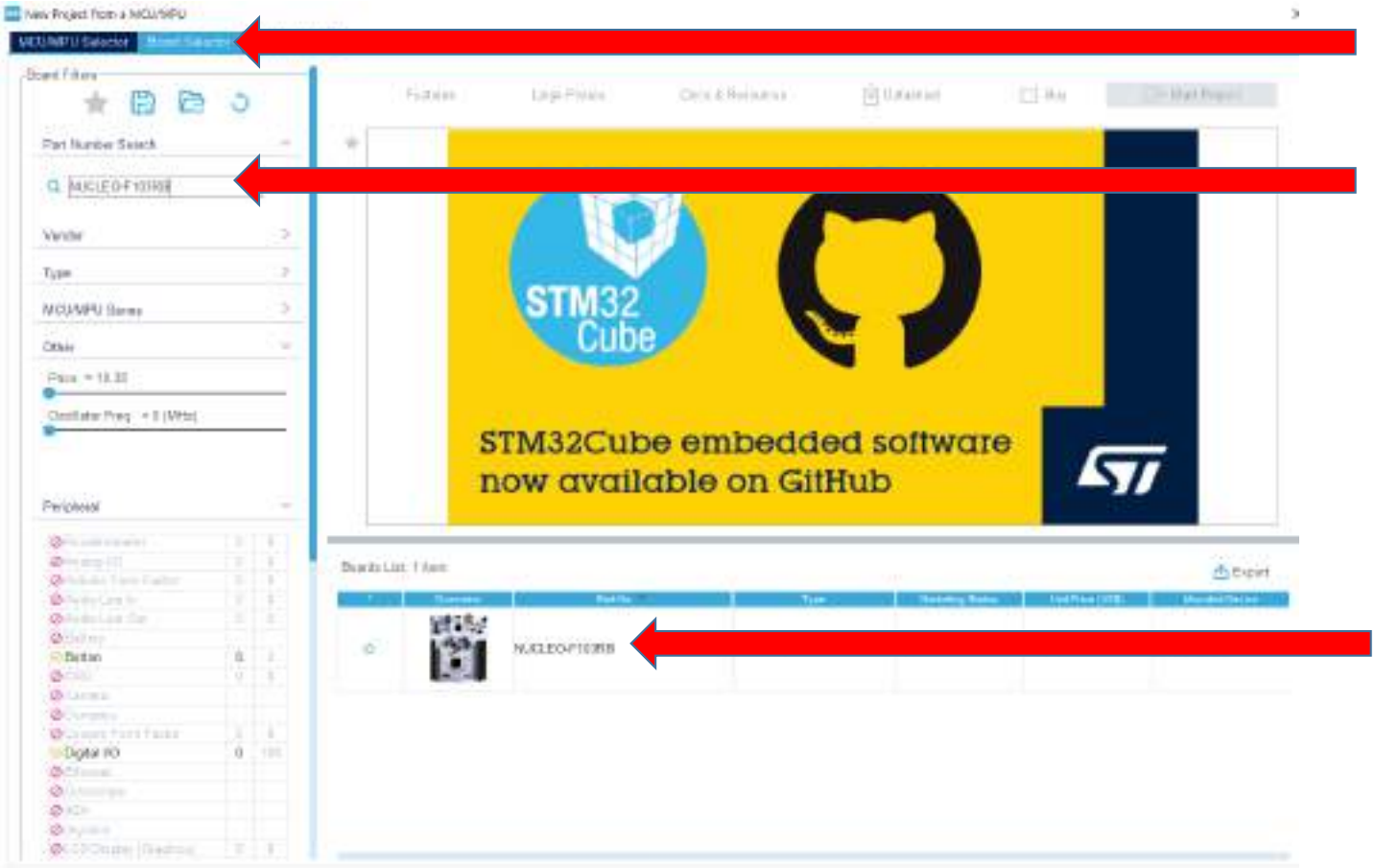
CubeMX로 예제 7.1 구현하기



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

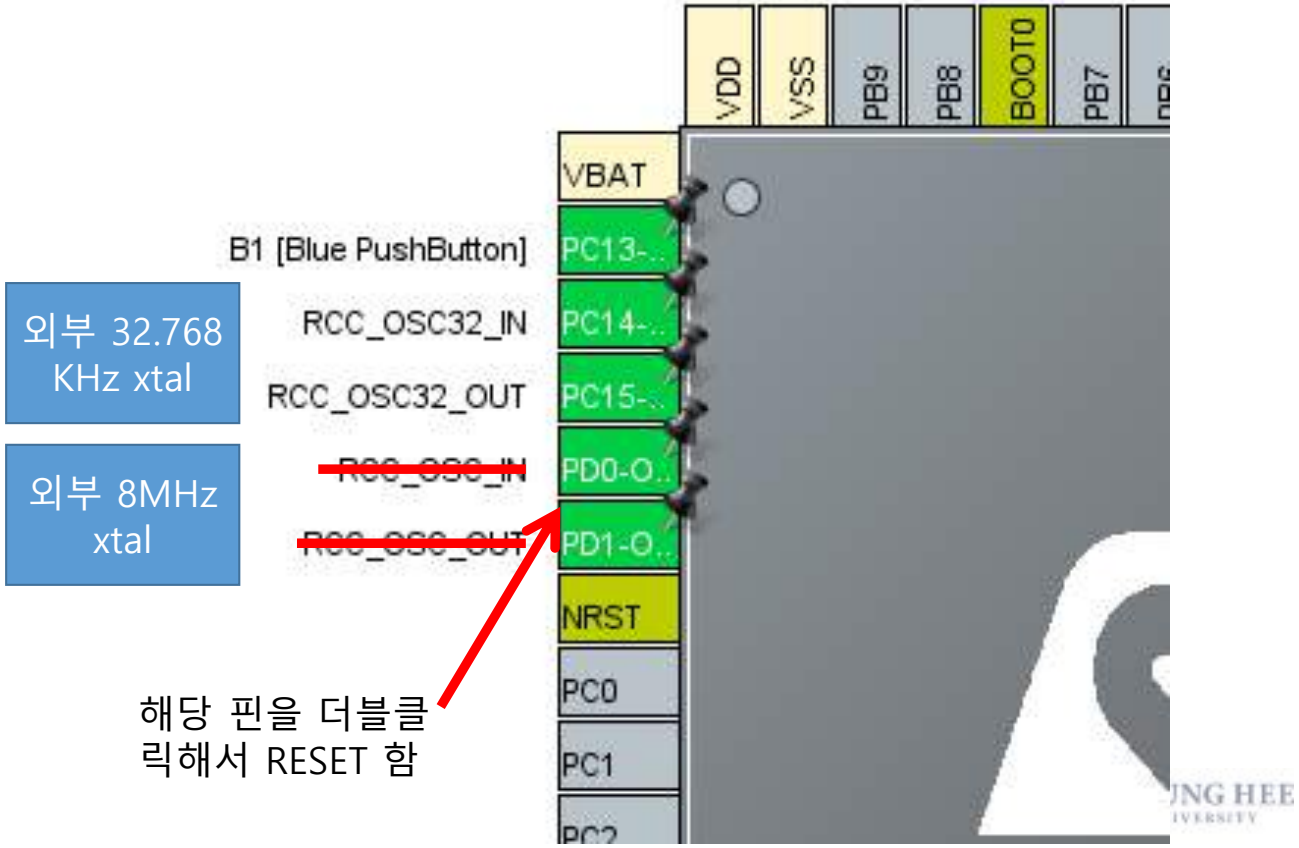
녹색으로 표시된 핀은 할당된 것
회색으로 표시된 핀은 할당안 된 것
연한 노란색 핀은 전원핀

7.4 외부 인터럽트 CubeMX과제

CubeMX로 예제 7.1 구현하기

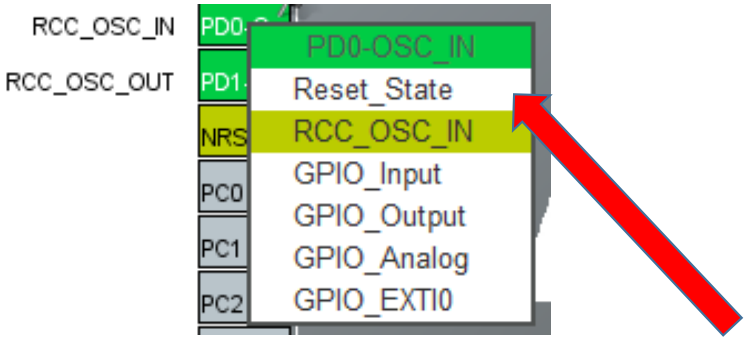
우리가 실습하는 nucleo 보드에는 외부에 추가할 수 있는 xtal 2개 중 RTC 용 32.768KHz 만 장착됨. 빠르고 정확한 동작을 위해서는 8MHz xtal을 달면 됨 (https://www.devicemart.co.kr/goods/view?no=7265)

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



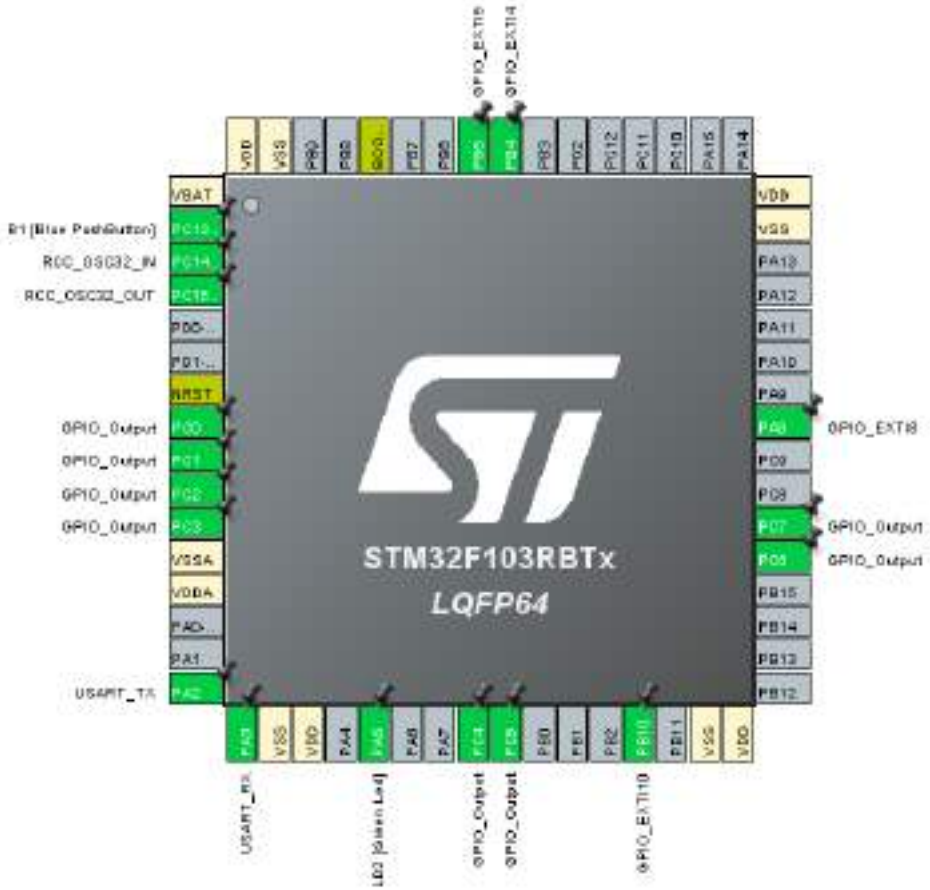
각 핀을 더블클릭하면 선택할 수 있는 옵션이 나옴. 핀 마다 모두 다르므로 주의

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

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

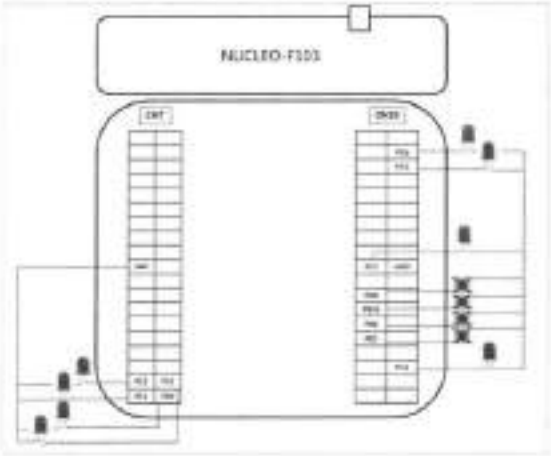
[연결용 회로도]

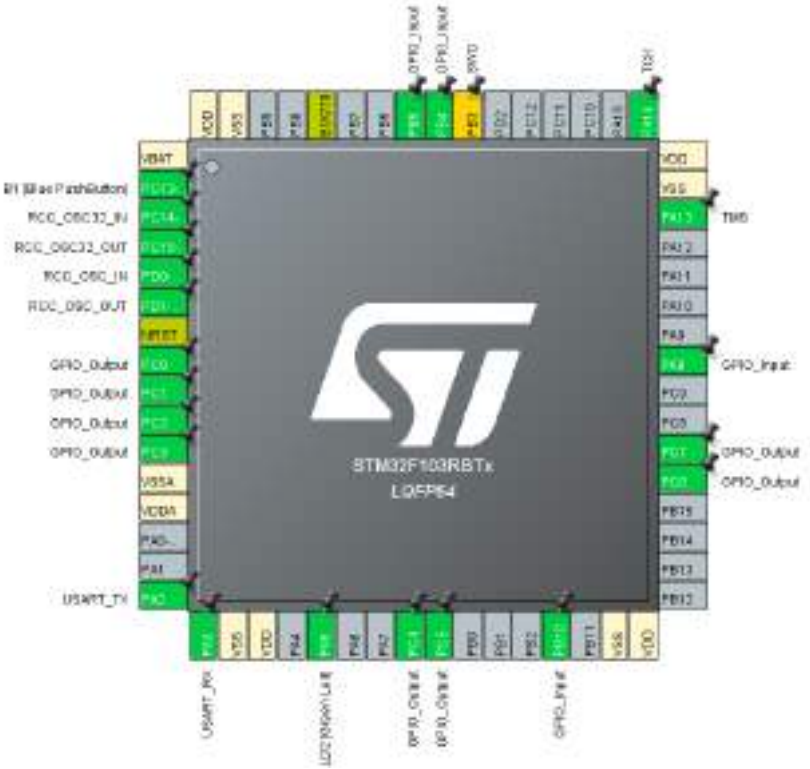
NUCLEO-F103RB 보드의 외부 연결용 커넥터(CN7, CN10)를 이용하여 LED 8개와 스위치 4개를 NUCLEO-F103RB 보드에 연결한다. 연결할 보드와 핀 번호는 다음과 같다.

- 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

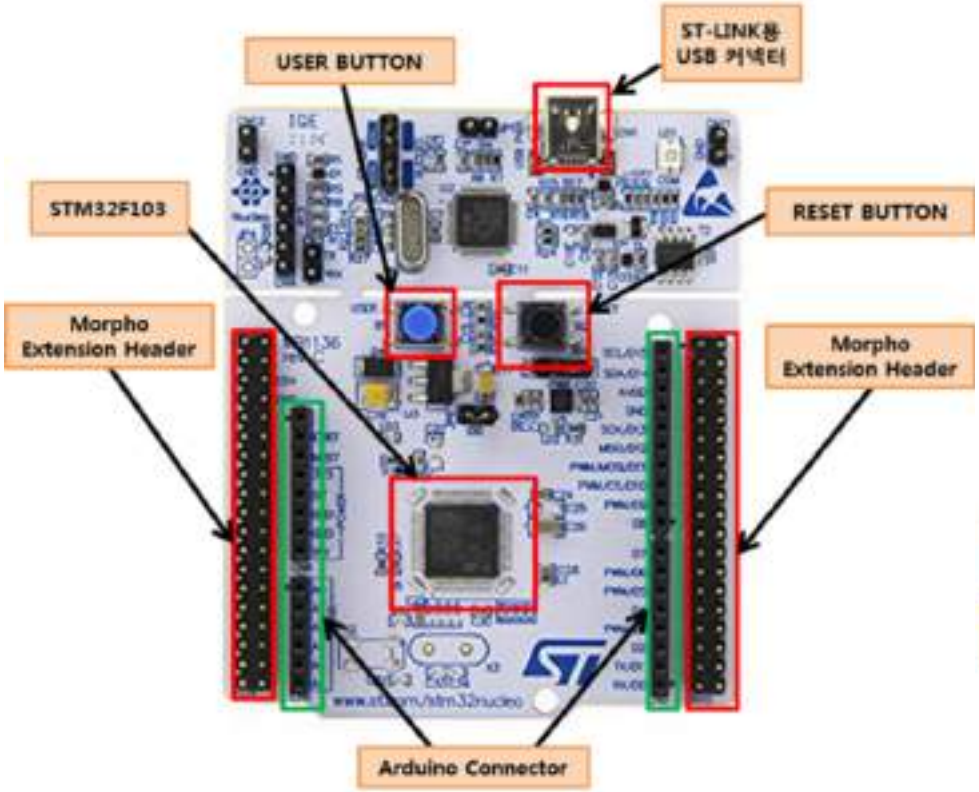
이 연결은 Nucleo-64용 I/O 보드의 회로도(그림 3-3-3(a))에 나타난 LED 1 ~ LED 8과 SW 1 ~ SW 4의 연결과 보드 및 핀 번호가 동일하다.



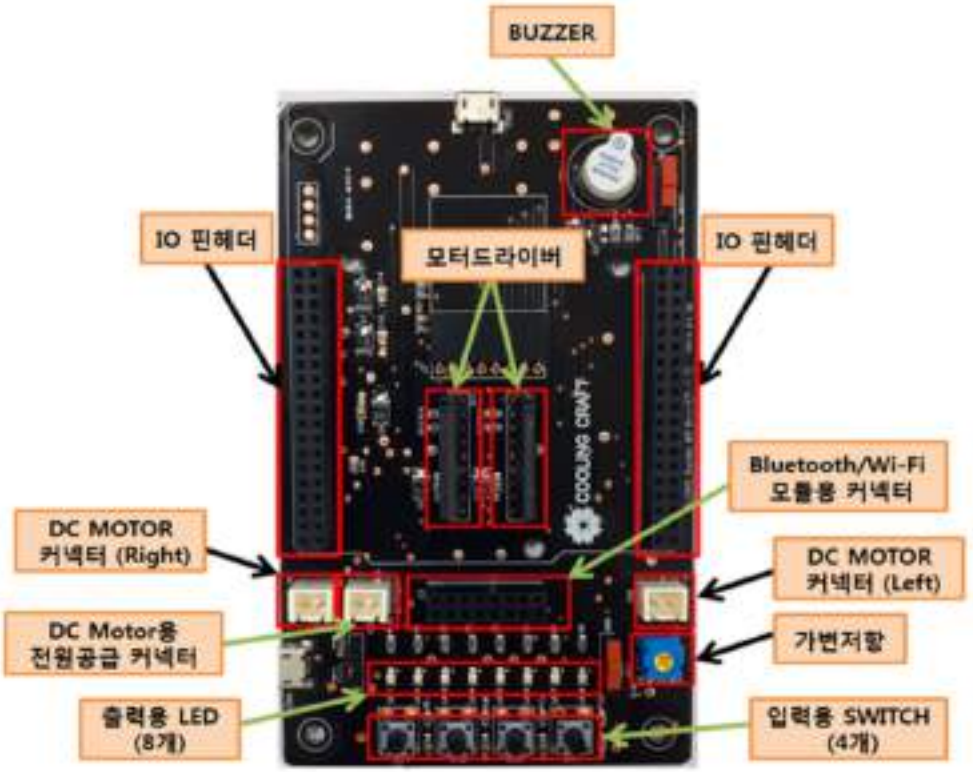


KYUNG HEE UNIVERSITY

참고. Nucleo-F103 보드와 I/O 확장보드



STM Nucleo-64 보드

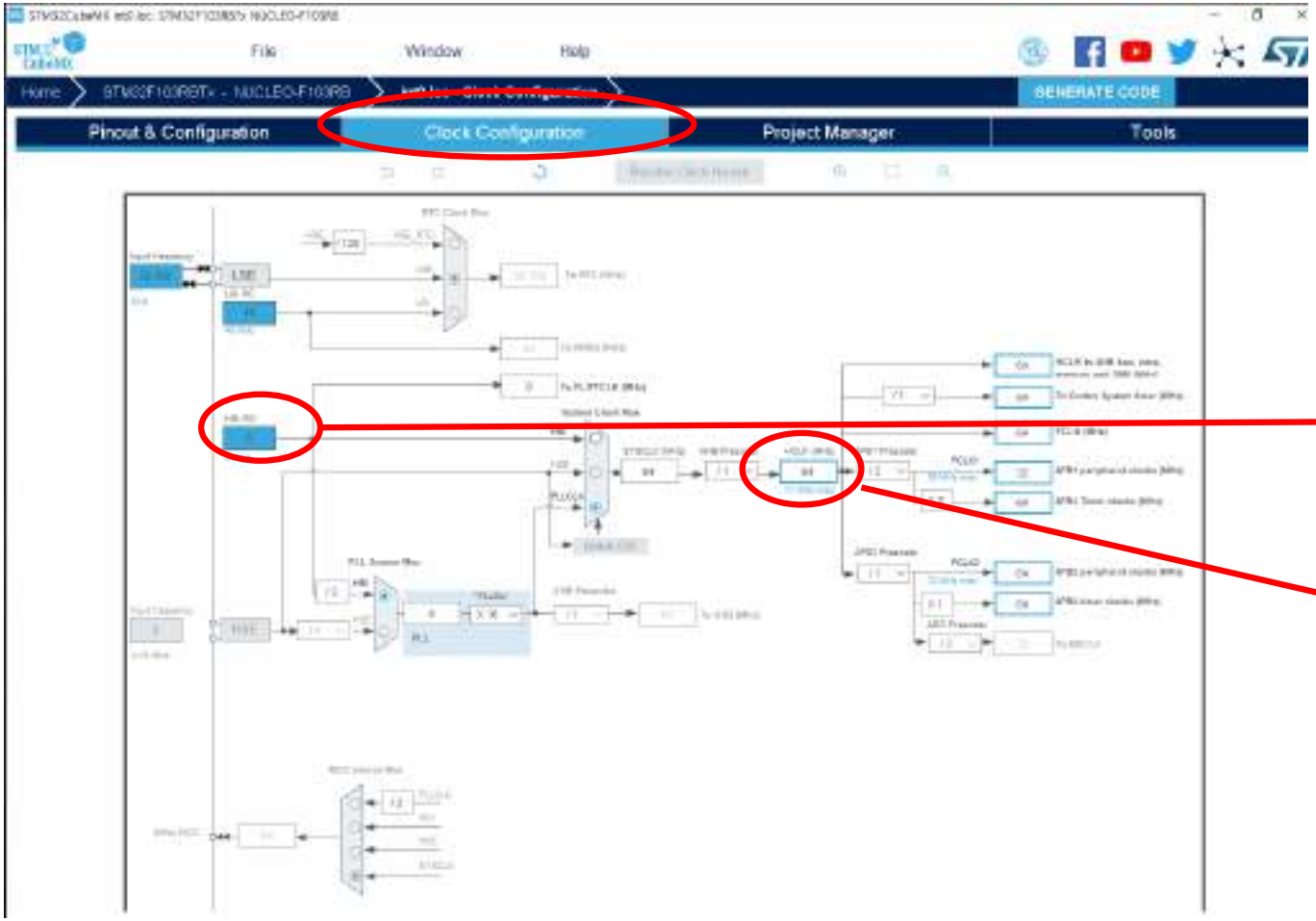


Nucleo-64용 I/O 보드

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, 8MHz HIS RC클럭을 사용함 (짙은 파란색 표시)
- 8MHz를 PLL로 빠르게 해서 최대 64MHz로 속도를 올림

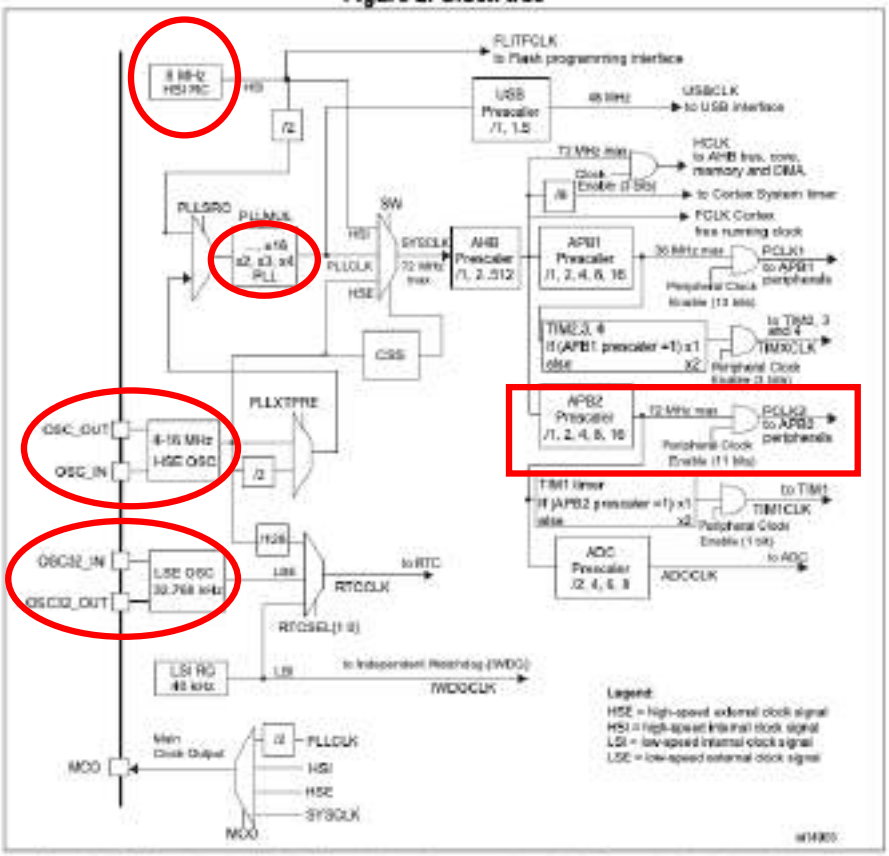
KYUNG HEE UNIVERSITY

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

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

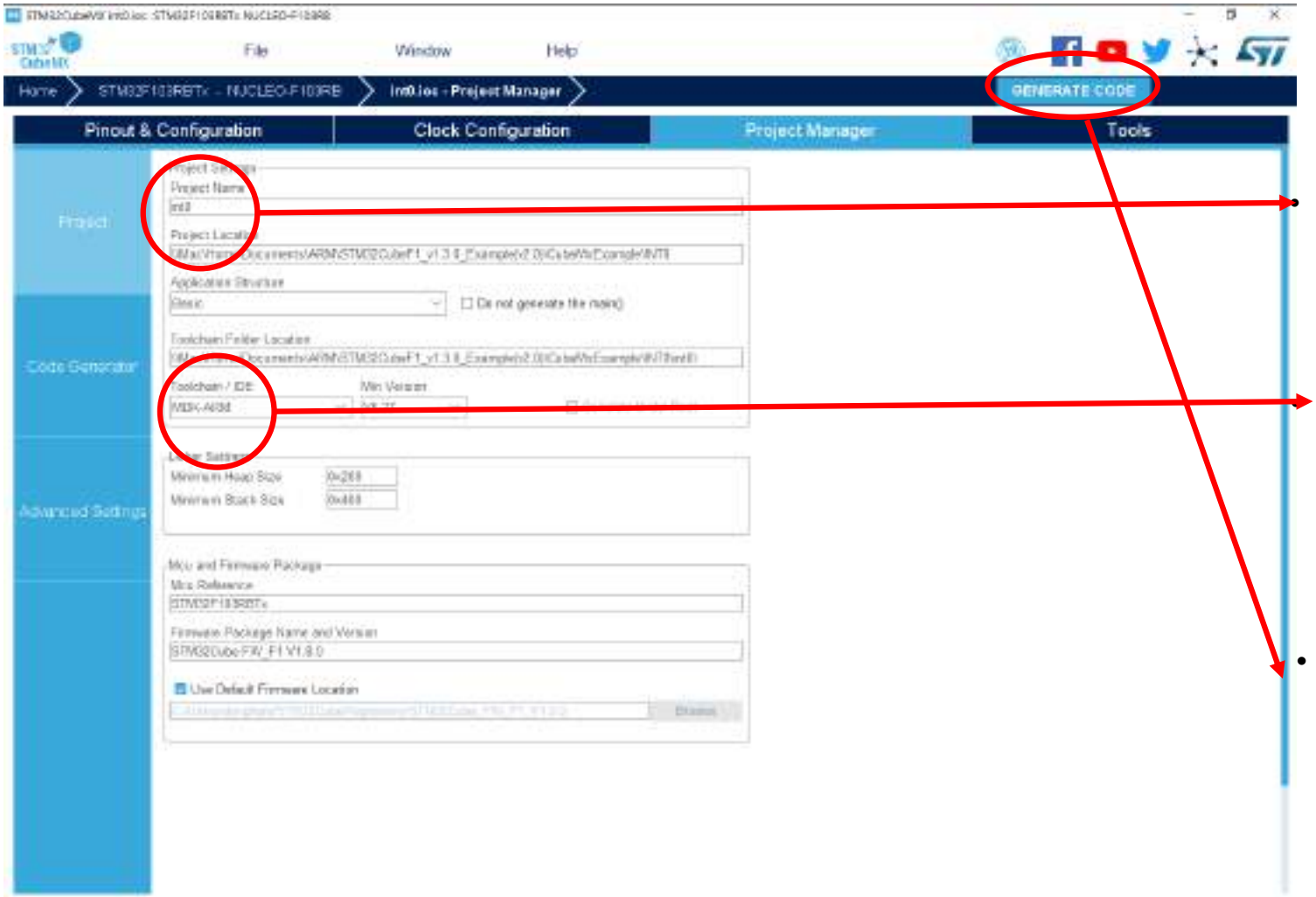
- GPIO 핀이 입력으로 설정된 경우에 입력 데이터 레지스터(Input data register)는 매 APB2 클럭(72MHz)마다 I/O 핀에서 데이터를 입력 받음
- 핀이 출력으로 설정된 경우에 출력 데이터 레지스터(Output data register)에 저장된 값이 I/O 핀에 출력됨
- 출력 모드에서만 출력 드라이버(Output driver)의 사용이 가능

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

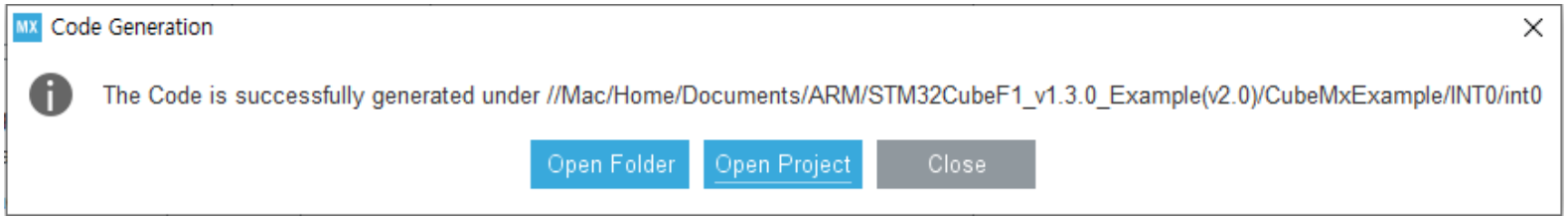


- Project Name을 입력하고 경로 지정(한글이름경로x)
- Toolchain/IDE를 MDK-ARM로 꼭 선택해야 함
- 마지막으로 GENERATE CODE 누름

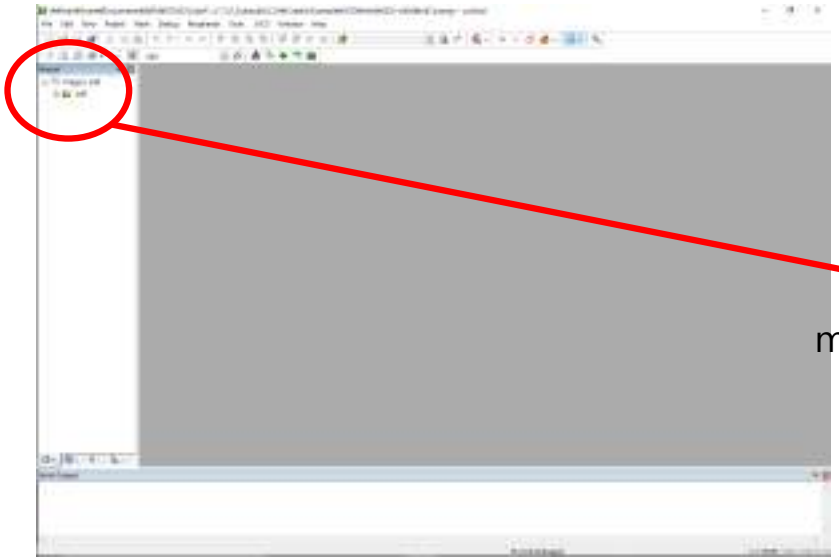
KYUNG HEE UNIVERSITY

7.4 외부 인터럽트 CubeMX과제

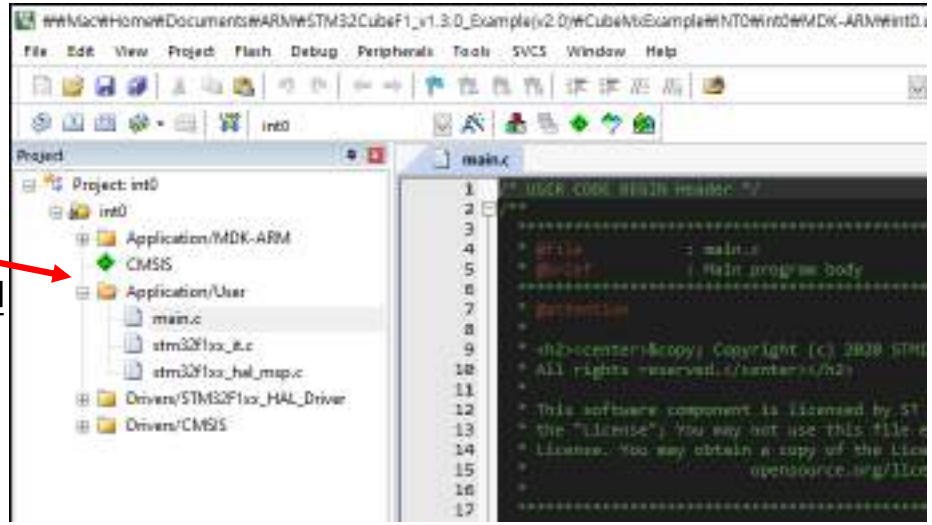
CubeMX로 예제 7.1 구현하기



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

6.5 CubeMX 설치 및 설정 방법

7) 코드 작성 (예제1을 Cube로 동작시켜 보자.)

```
int main(void)
{
  /* USER CODE BEGIN 1 */
  uint8_t PINSTATE;
  /* USER CODE END 1 */


  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* 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)
  {
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_6 | GPIO_PIN_7 ,GPIO_PIN_SET);
    HAL_Delay(500);
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_6 | GPIO_PIN_7 ,GPIO_PIN_RESET);
    HAL_Delay(500);
    /* USER CODE END WHILE */

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

Main.c 파일을 열어 while문 안에 앞에서 배운
HAL GPIO 구동용 함수를 사용하여 프로그래밍 하면 된다.

KYUNG HEE UNIVERSITY

6.6 과제

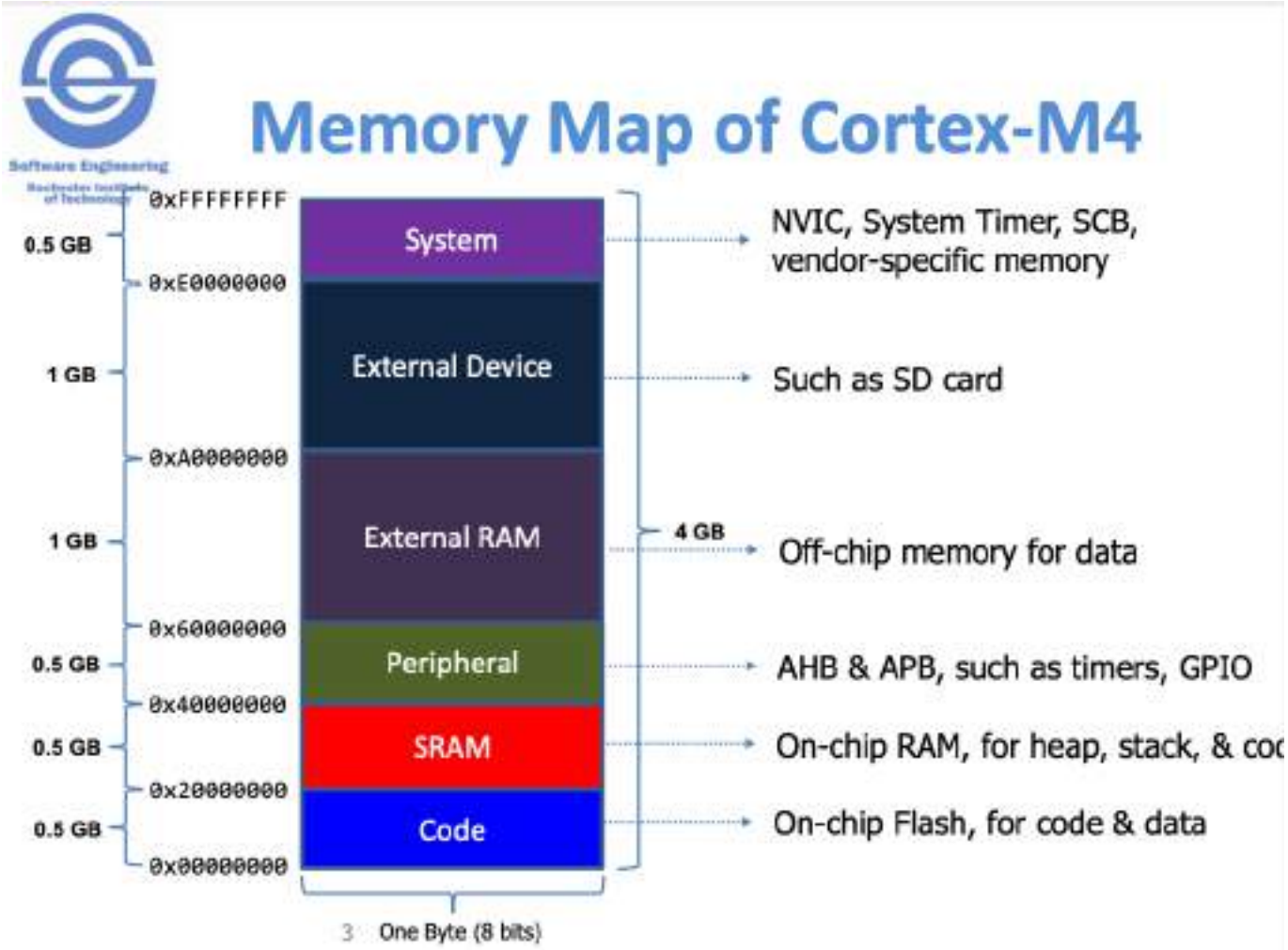
[과제 양식 안내]

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

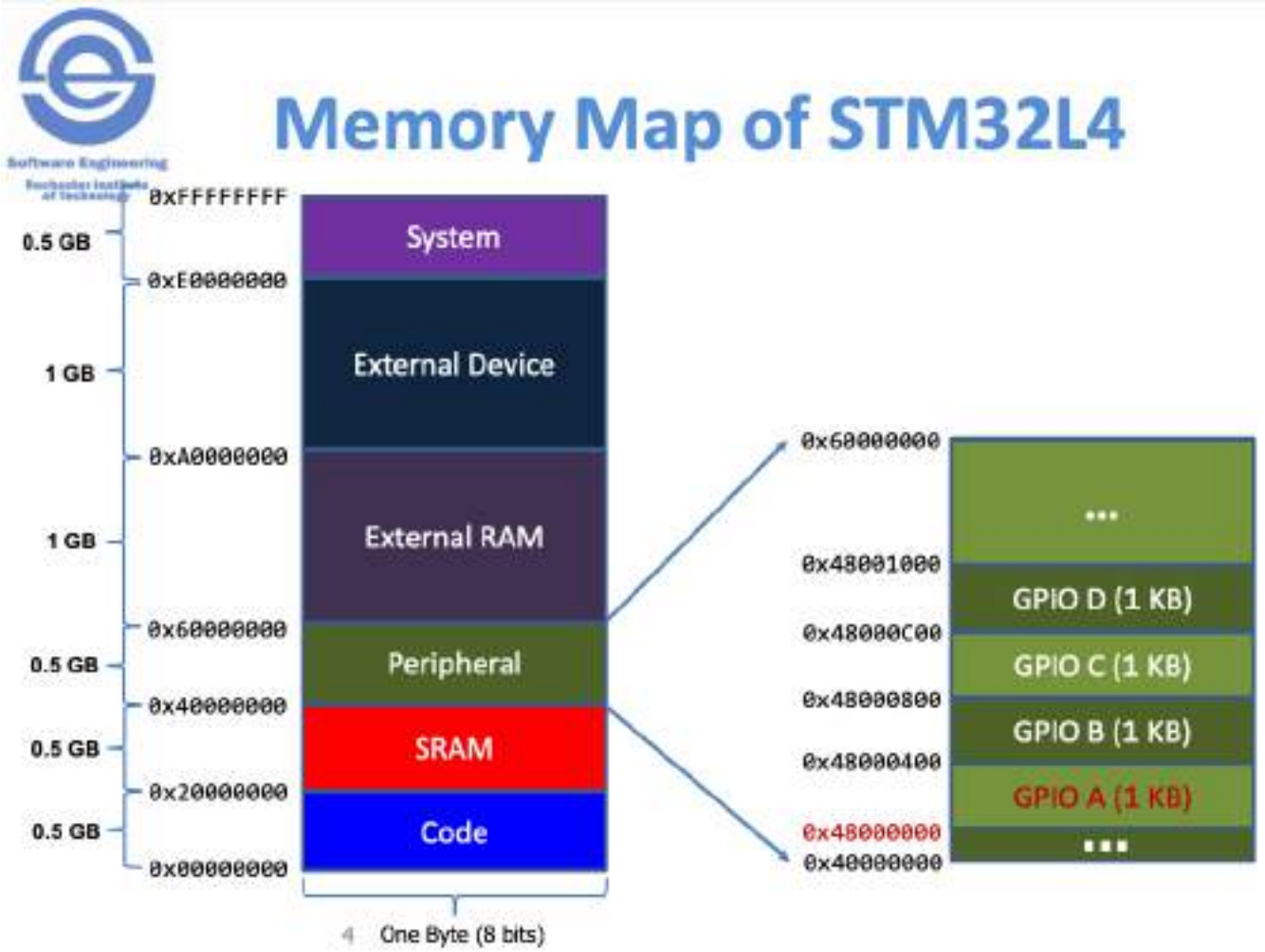
6장 과제 → 예제 6.2, 6.3, 6.4를 Cubemx를 사용하여 프로그래밍 하기.

부록. Memory Map (발췌: 로체스터 공과대학 강의자료)



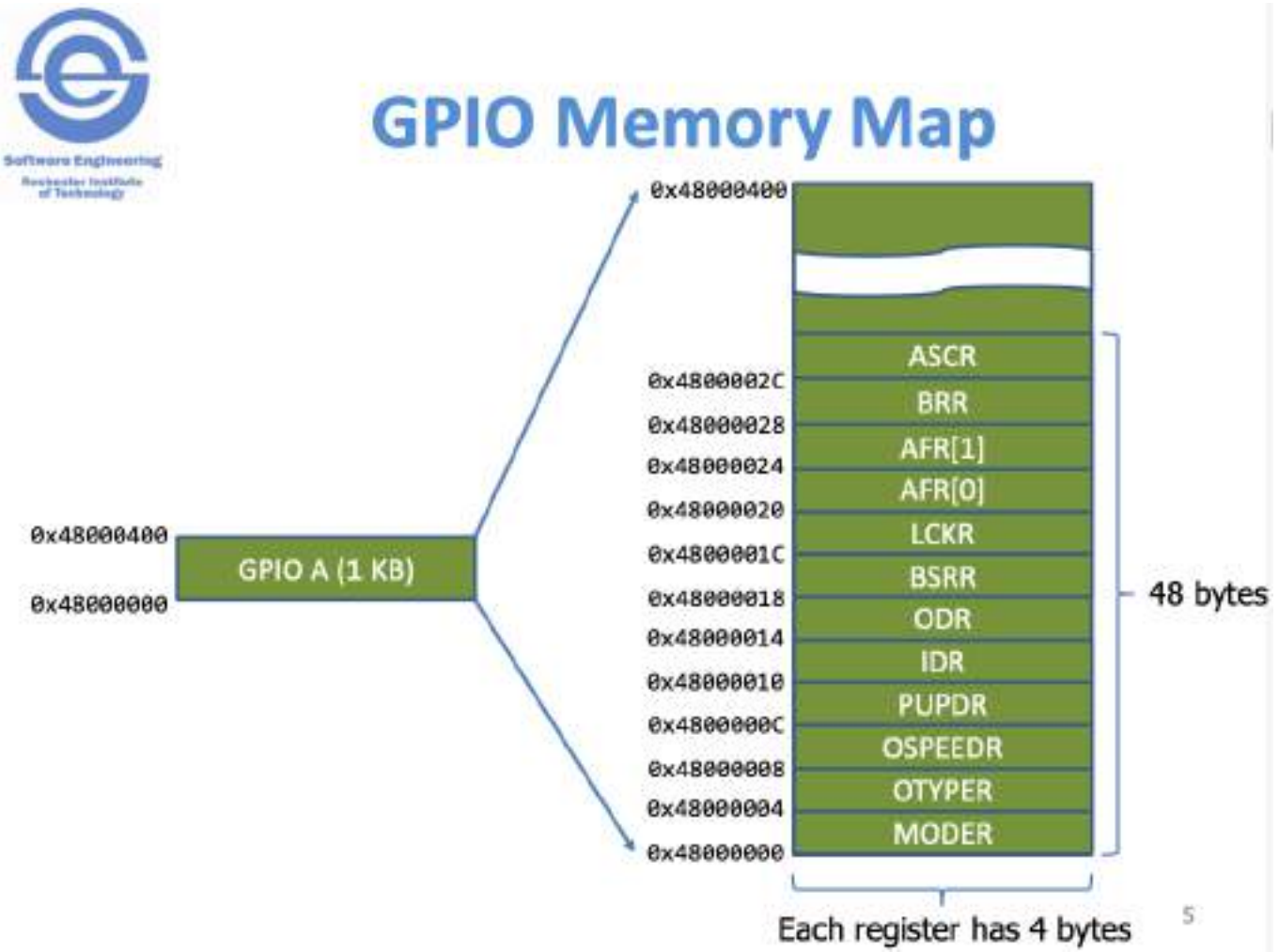
KYUNG HEE UNIVERSITY

부록. Memory Map (발췌: 로체스터 공과대학 강의자료)

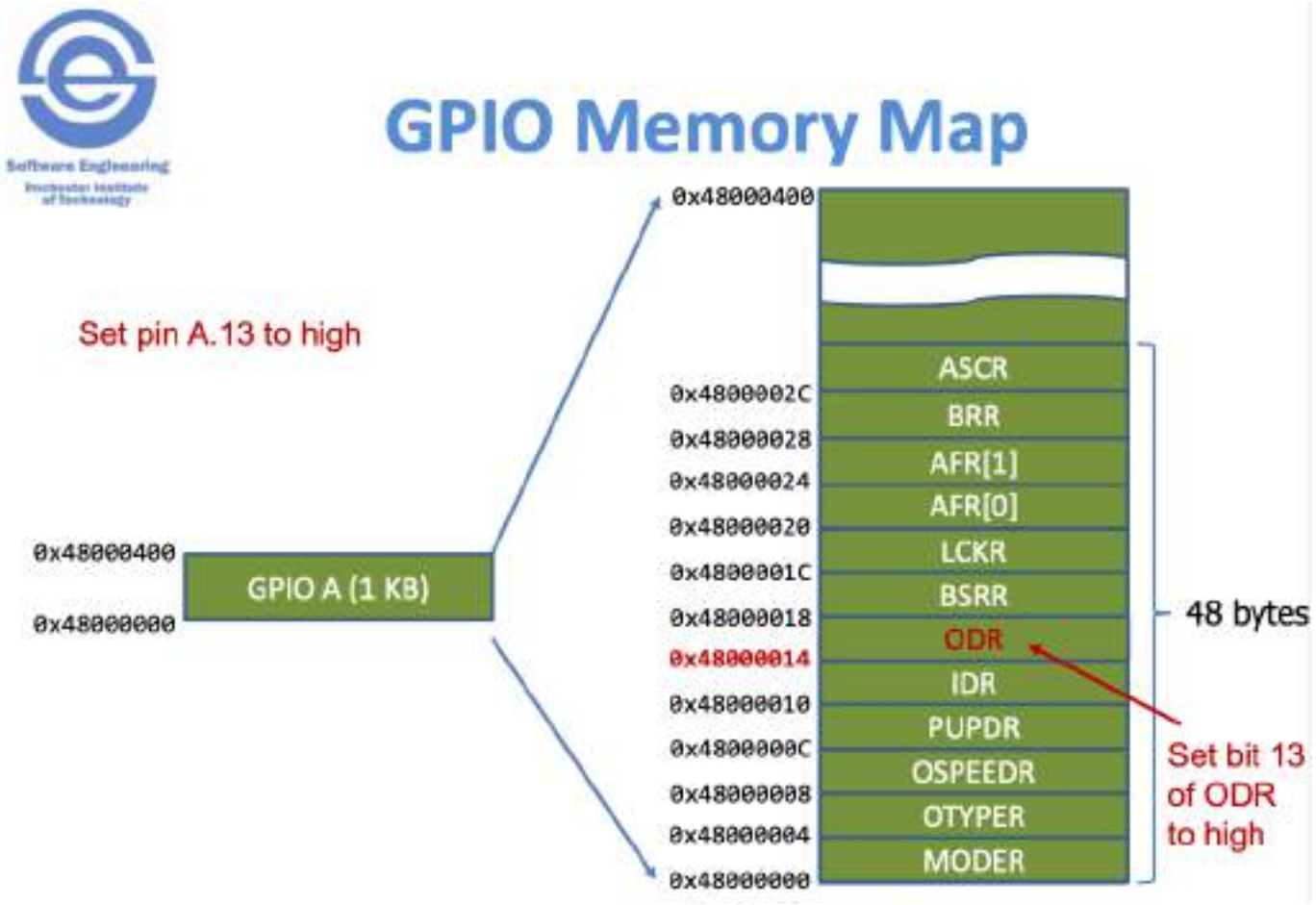


KYUNG HEE UNIVERSITY

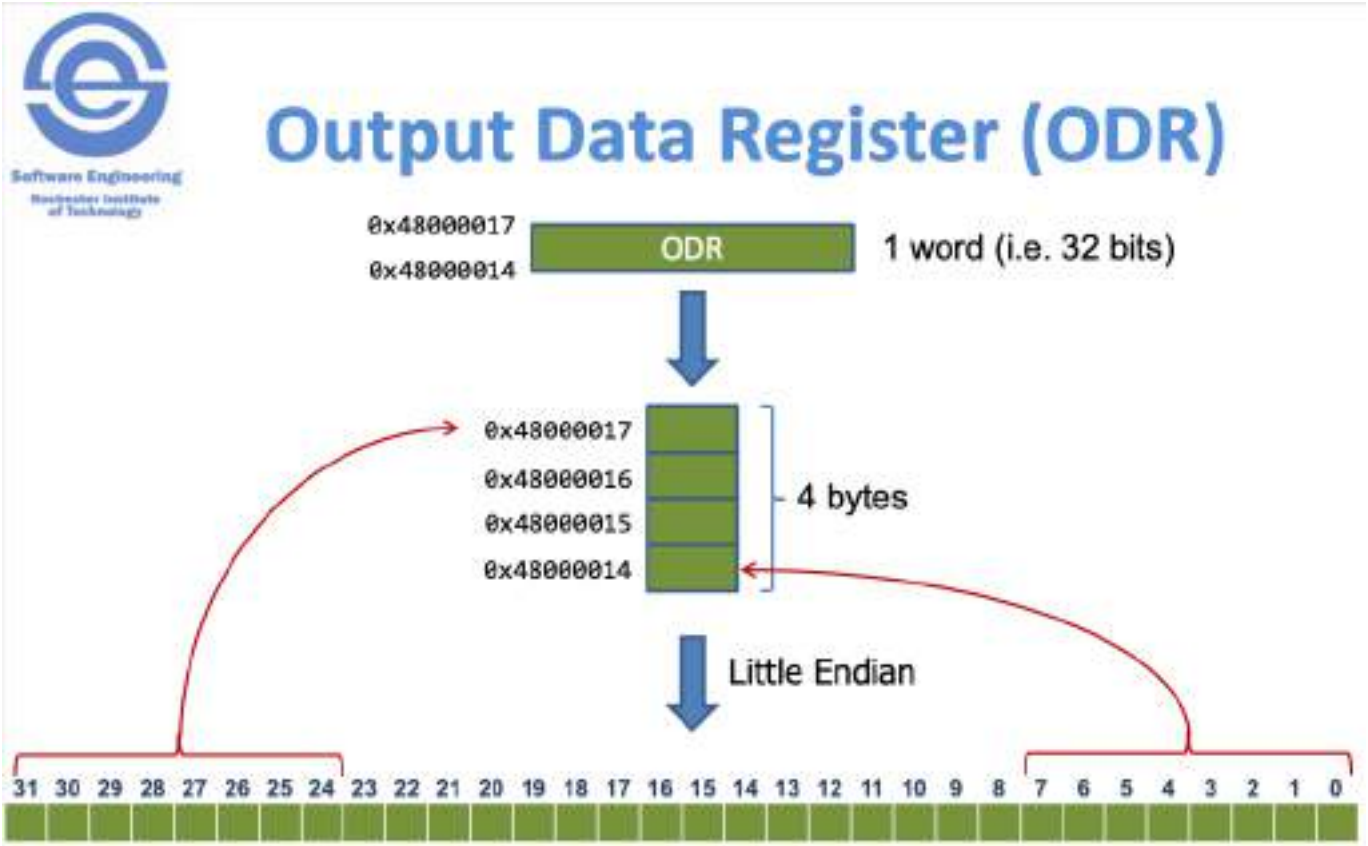
부록. Memory Map (발췌: 로체스터 공과대학 강의자료)



부록. Memory Map (발췌: 로체스터 공과대학 강의자료)

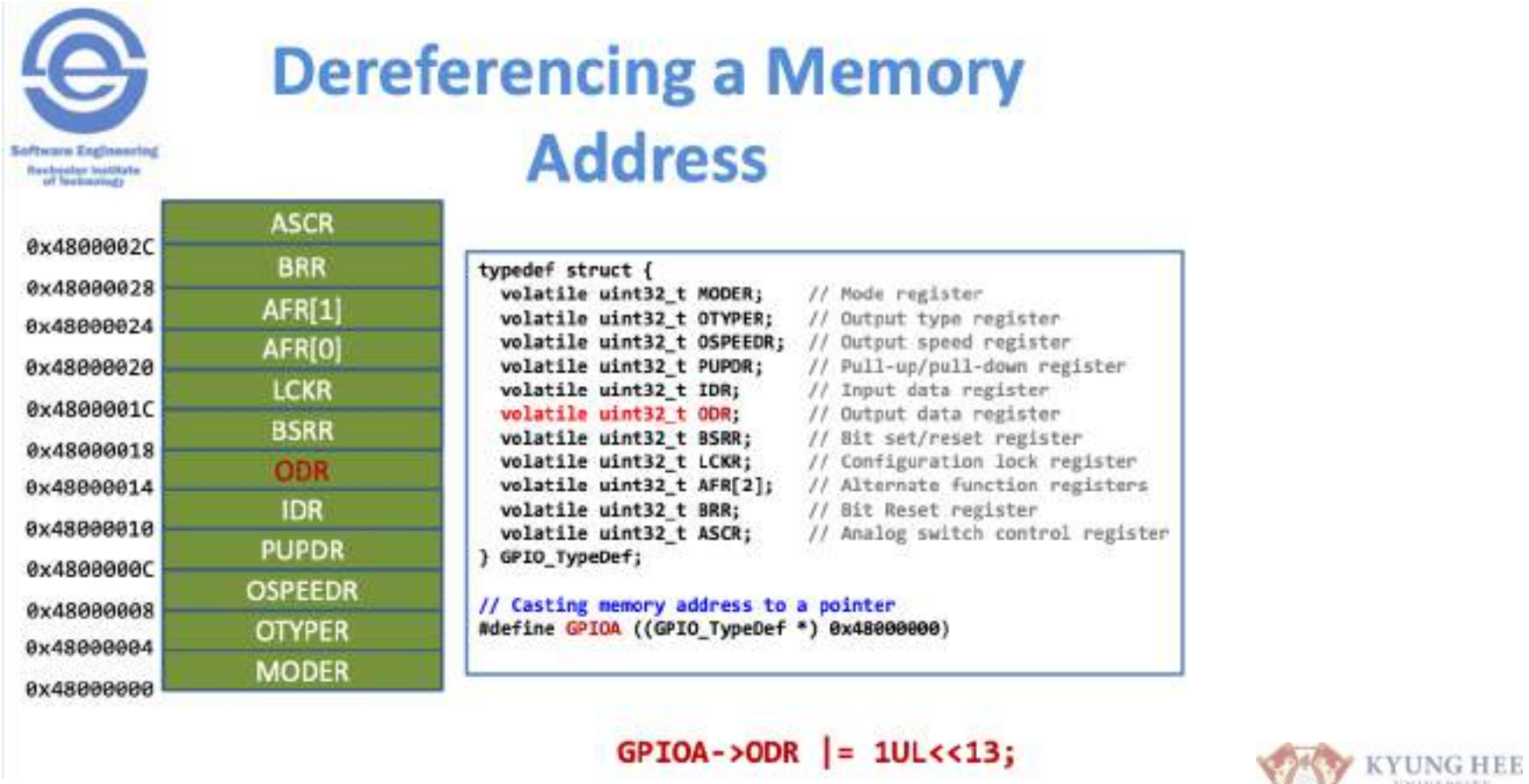


부록. Memory Map (발췌: 로체스터 공과대학 강의자료)



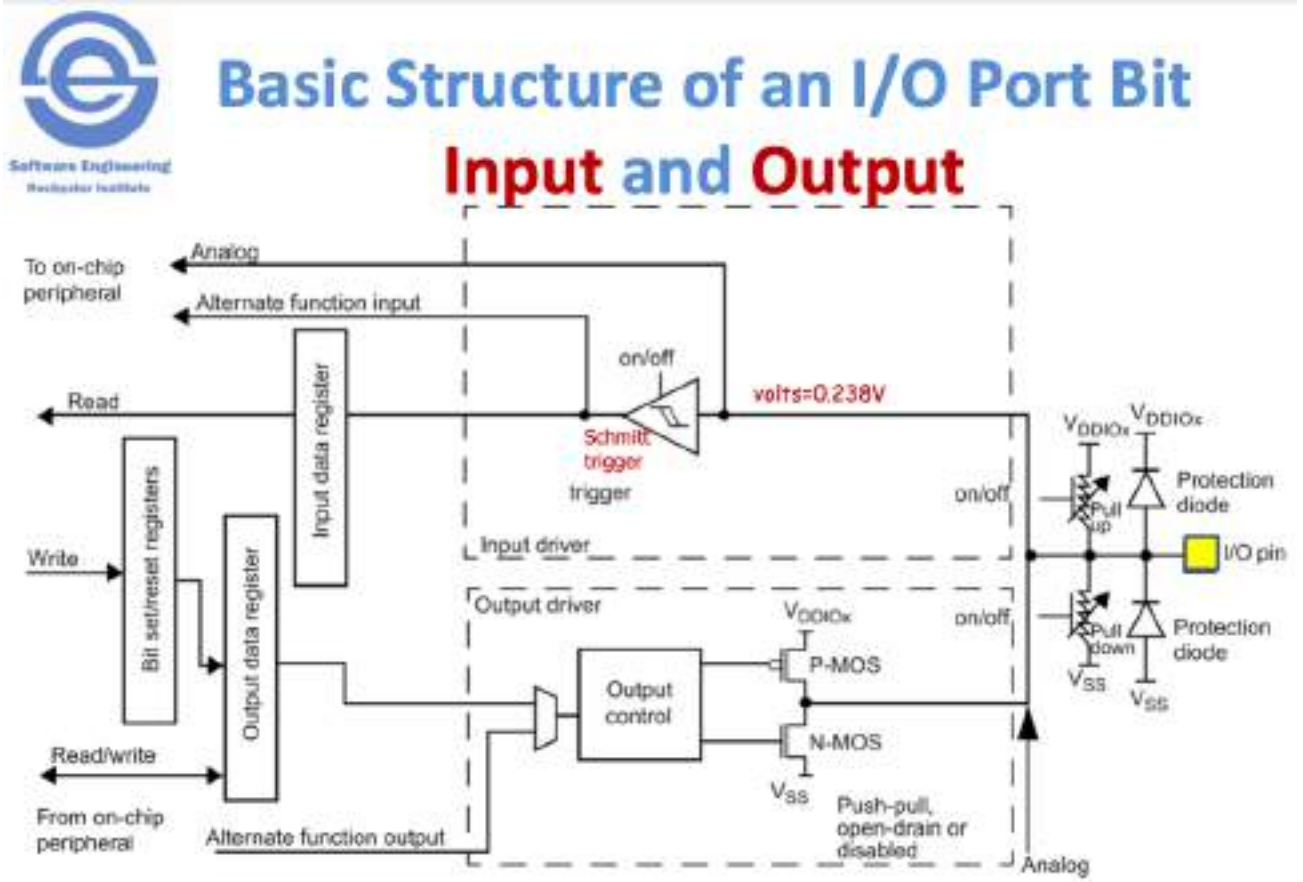
KYUNG HEE UNIVERSITY

부록. Memory Map (발췌: 로체스터 공과대학 강의자료)



KYUNG HEE UNIVERSITY

부록. GPIO 내부구조 (발췌: 로체스터 공과대학 강의자료)

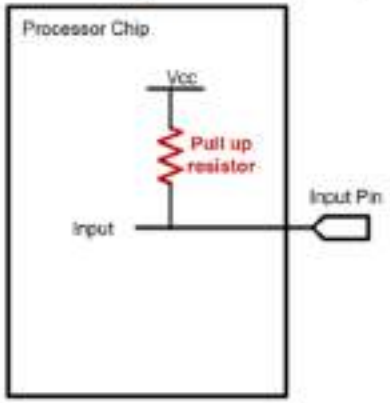


KYUNG HEE UNIVERSITY

부록. GPIO 내부구조 (발췌: 로체스터 공과대학 강의자료)

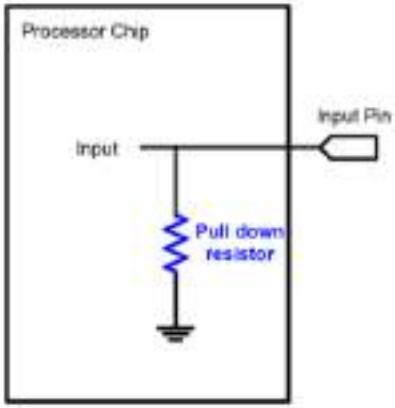
GPIO Input:
Pull Up and Pull Down

- A digital input can have three states: High, Low, and High-Impedance (also called floating, tri-stated, HiZ)



Pull-Up

If external input is HiZ, the input is read as a valid HIGH.

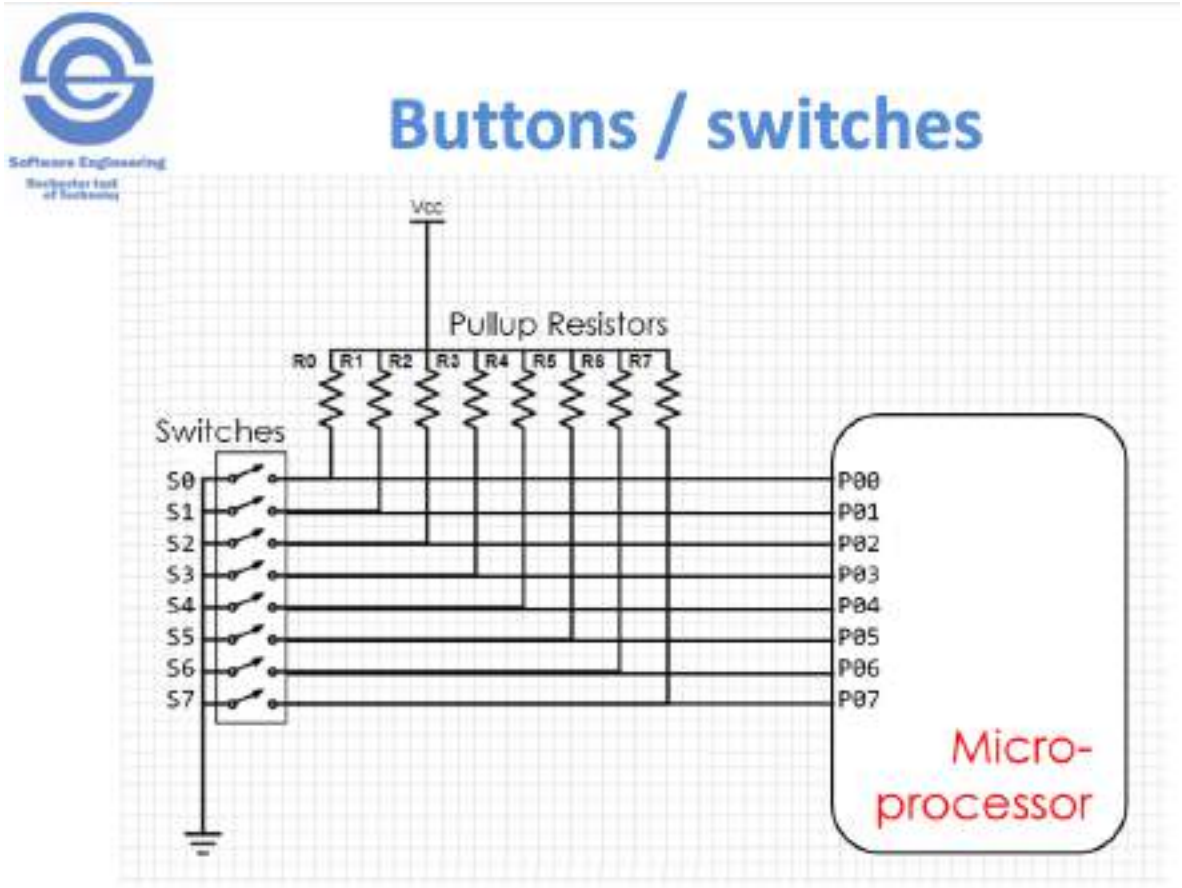


Pull-Down

If external input is HiZ, the input is read as a valid LOW.

KYUNG HEE UNIVERSITY

부록. GPIO 내부구조 (발췌: 로체스터 공과대학 강의자료)

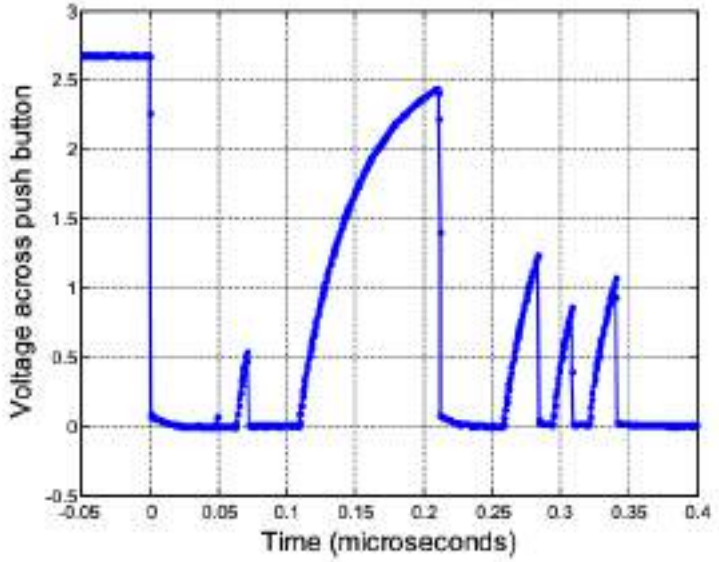


KYUNG HEE UNIVERSITY

부록. GPIO 내부구조 (발췌: 로체스터 공과대학 강의자료)

I/O Debouncing

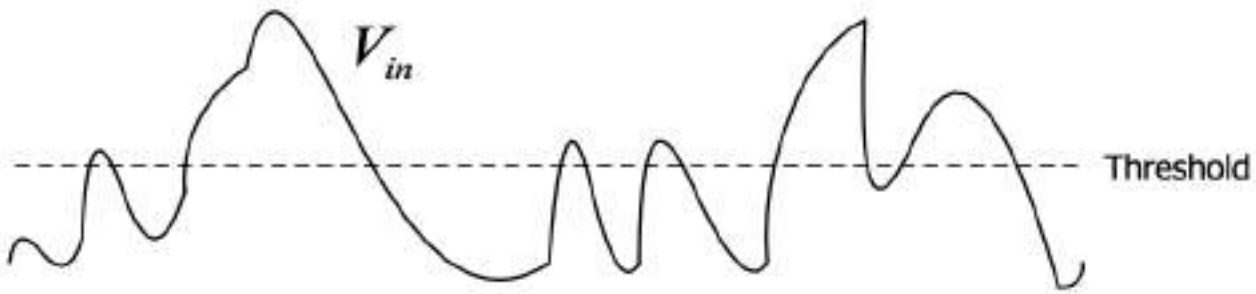
- Example signal when a button is pressed



KYUNG HEE UNIVERSITY

부록. GPIO 내부구조 (발췌: 로체스터 공과대학 강의자료)

Noisy analog signals ...

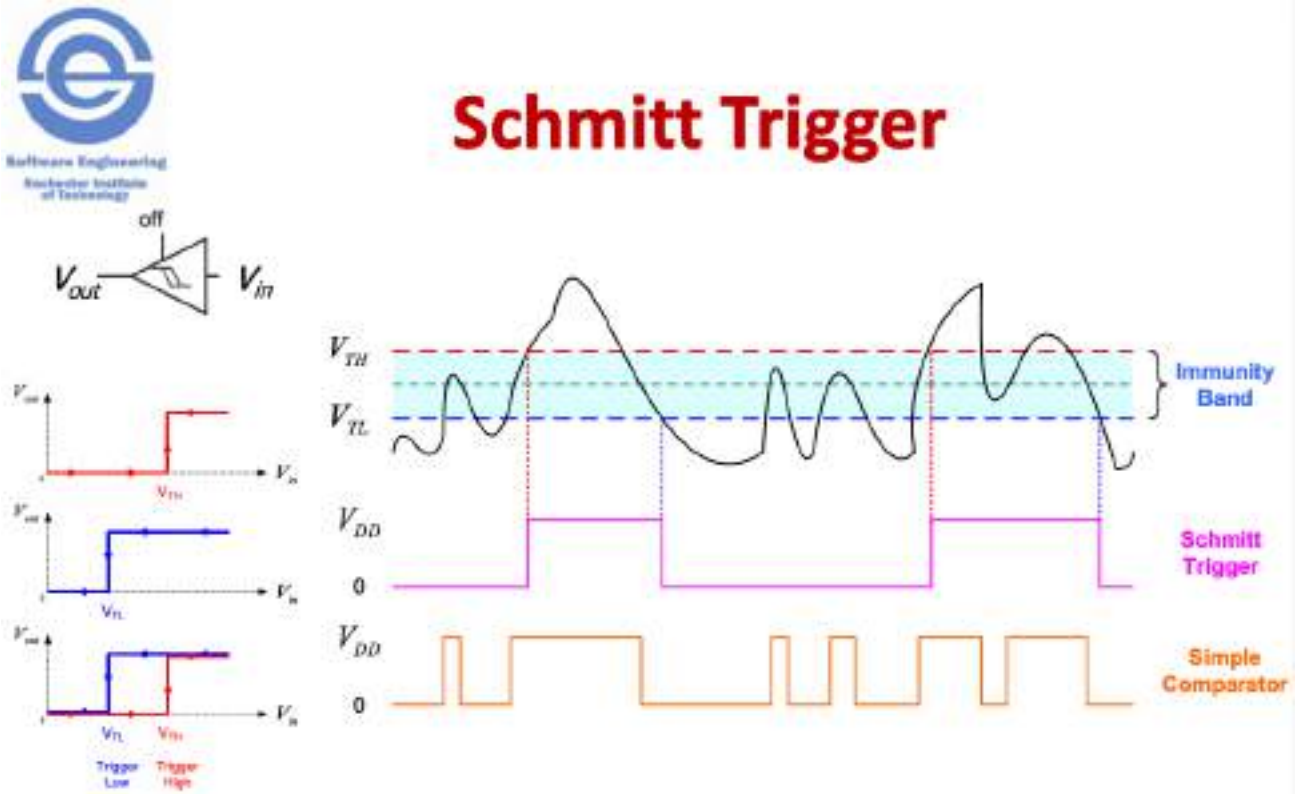


Analog signals

- Noisy
- Rise and fall slowly (small slew rate)
- A button press can generate a noisy transition for up to 20 msec if not debounced in hardware

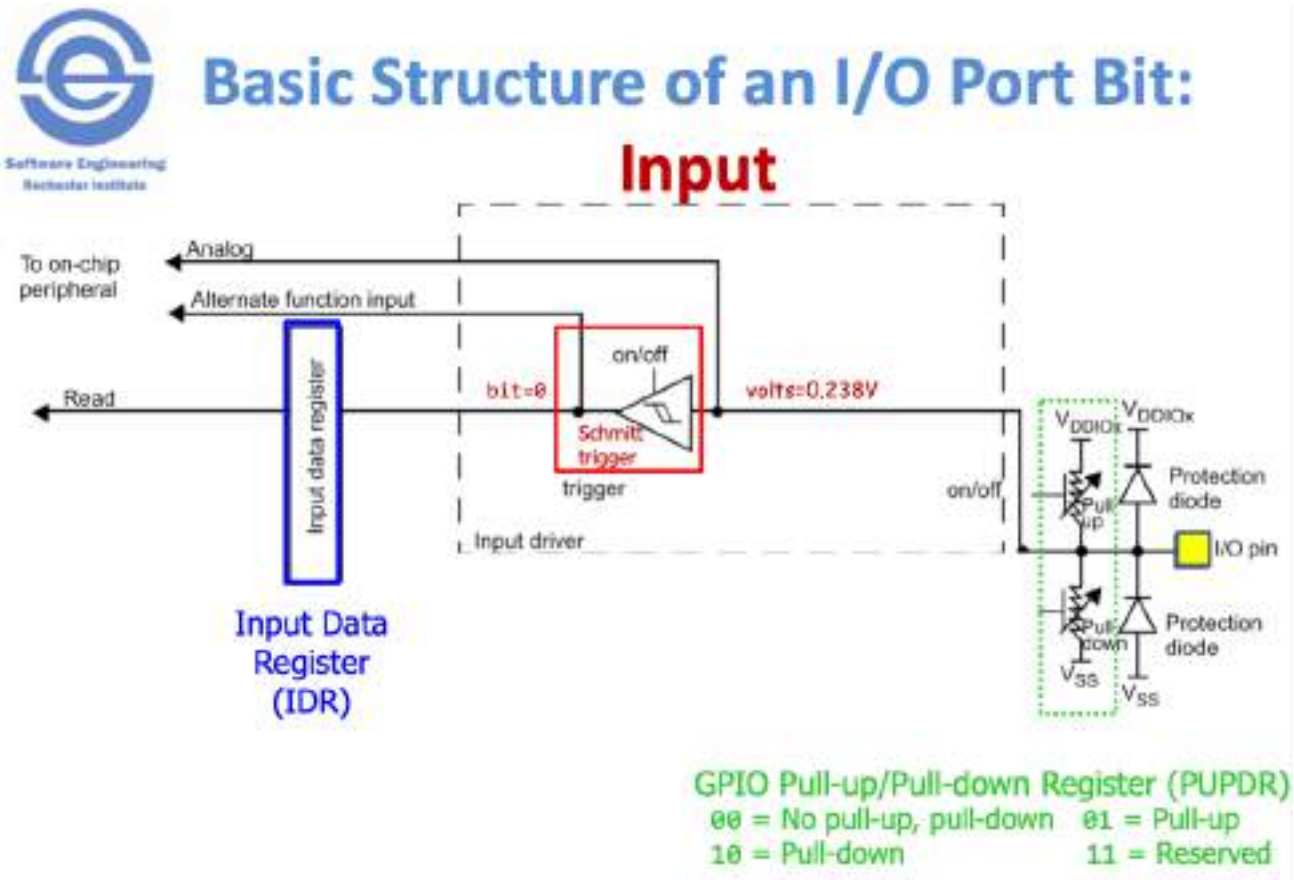
KYUNG HEE UNIVERSITY

부록. GPIO 내부구조 (발췌: 로체스터 공과대학 강의자료)



KYUNG HEE UNIVERSITY

부록. GPIO 내부구조 (발췌: 로체스터 공과대학 강의자료)



KYUNG HEE UNIVERSITY