

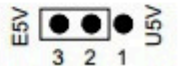
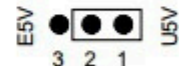
STM32CubeIDE를 이용한 STM32 따라하기

Chapter 5. NUCLEOEVB 보드를 이용한 실습

NUCLEOEVB 보드 사용 설정

Input power name	Connectors pins	Voltage range	Max current	Limitation
VIN	CN6 pin 8 CN7 pin 24	7 V to 12 V	800 mA	From 7 V to 12 V only and input current capability is linked to input voltage: 800 mA input current when $V_{in}=7\text{ V}$ 450 mA input current when $7\text{ V}<V_{in} (< \text{or } =) 9\text{ V}$ 250 mA input current when $9\text{ V}<V_{in} (< \text{or } =) 12\text{ V}$
E5V	CN7 pin 6	4.75 V to 5.25 V	500 mA	-

Jumper	Description
JP5	U5V (ST-LINK VBUS) is used as power source when JP5 is set as shown below (Default setting)
	VIN or E5V is used as power source when JP5 is set as shown below.



- 외부 장치를 위해 별도의 외부 전원(9V 800mA) 사용
- NUCLEO 보드에 외부 전원 설정 점퍼 핀(JP5)에 핀2와 핀3를 점퍼로 연결

5.1. GPIO

버튼 입력 핀 값을 읽어 LED 출력 핀으로 출력하여 GPIO 입출력을 실습하기

- 지금까지는 NUCLEO 보드만 사용하였기 때문에 프로젝트 생성 시 Default Mode로 Pinout 설정을 불러왔었음. 이번부터는 사용자 설계에 맞춘 MCU 선정 및 기능을 선택할 것임.

5.1.1. STM32CubeIDE 프로젝트 생성

- **File** → **New** → **STM32 Project**

Target Selection
Select STM32 target or STM32Cube example

MCU/MPU Selector | Board Selector | Example Selector | Cross Selector

Commercial Part Number: [Search]

PRODUCT INFO

- Segment: >
- Series: >
- Line: >
- Marketing Status: >
- Price: >
- Package: >
- Core: >
- Coprocessor: >

MEMORY

- Flash: From 64 to 1536 (kBytes) [Slider]
- EEPROM: = 0 (Bytes)
- RAM Total: From 32 to 320 (kBytes) [Slider]
- RAM: From 32 to 320 (kBytes) [Slider]
- CCM RAM: = 0 (kBytes)

Features | Block Diagram | Docs & Resources | CAD Resources | Datasheet | Buy

STM32F4 Series

STM32F411RE **High-performance access line, Arm Cortex-M4 core with DSP and FPU, 512 Kbytes of Flash memory, 100 MHz CPU, ART Accelerator**

ACTIVE
Product is in mass production

Unit Price for 10kU (US\$): **2.9314**

Board: **NUCLEO-F411RE**

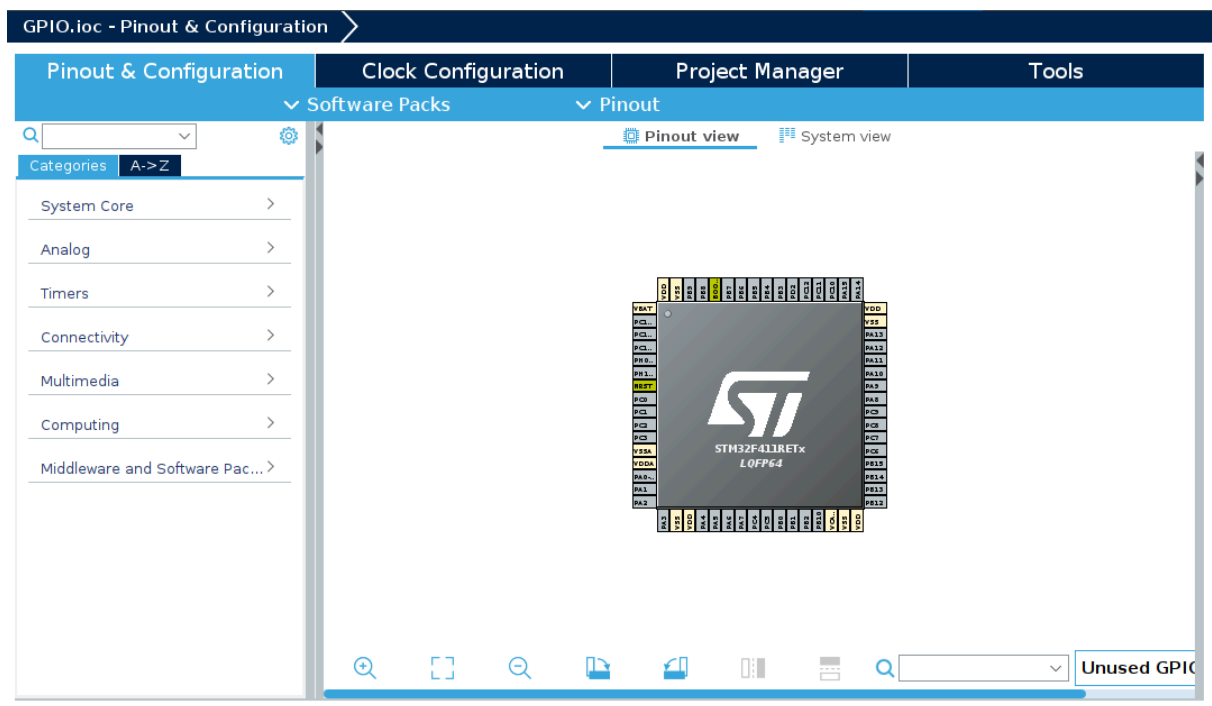
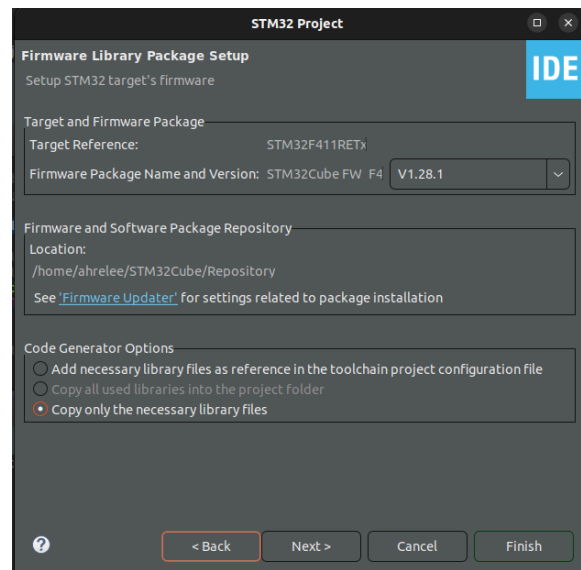
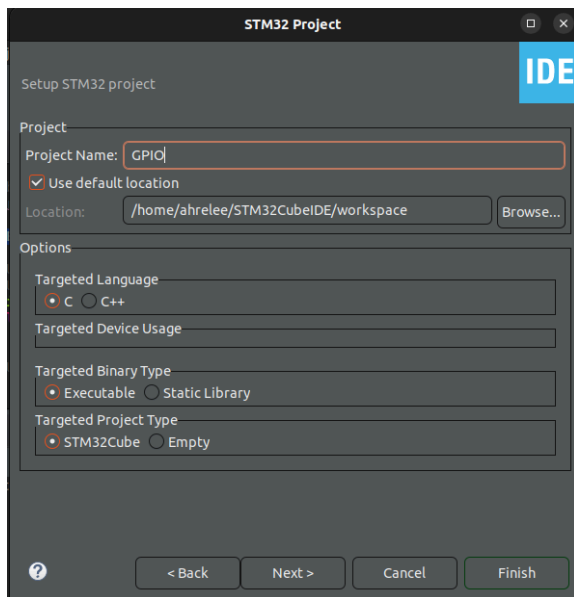
LQFP 64 10x10x1.4 mm

The STM32F411xC/E devices are based on the high-performance Arm® Cortex®-M4 32-bit RISC core operating at a frequency of up to 100 MHz. The Cortex®-M4 core features a floating-point unit (FPU) single precision, which supports all Arm single-precision data-processing instructions and data types. It also implements a full set of DSP instructions and a memory protection unit (MPU), which enhances application security. The STM32F411xC/E belongs to the STM32 Dynamic Efficiency product line (with products combining power efficiency, performance and integration) while adding a new innovative feature called Batch Acquisition Mode (BAM) allowing to save even more power consumption during data batching.

MCUs/MPUs List: 56 items

* Comm...	Part No	Reference	Marketi...	Unit Pri...	Board	Package	Flash	RAM	I/O	Frequ...
☆	STM32F4...	STM32F4...	Active	1.762		LQFP 64 ...	128 kBytes	32 kBytes	50	100 MHz
☆	STM32F4...	STM32F4...	Active	1.8853		LQFP 64 ...	128 kBytes	32 kBytes	50	100 MHz
☆	STM32F4...	STM32F4...	Active	1.8853		LQFP 64 ...	128 kBytes	32 kBytes	50	100 MHz
☆	STM32F4...	STM32F4...	Active	2.5363		LQFP 64 ...	256 kBytes	128 kBytes	50	100 MHz
☆	STM32F4...	STM32F4...	Active	2.5363		LQFP 64 ...	256 kBytes	128 kBytes	50	100 MHz
☆	STM32F4...	STM32F4...	Active	2.7138		LQFP 64 ...	256 kBytes	128 kBytes	50	100 MHz
☆	STM32F4...	STM32F4...	Active	2.7138		LQFP 64 ...	256 kBytes	128 kBytes	50	100 MHz
☆	STM32F4...	STM32F4...	Active	2.9314	NUCLEO-F...	LQFP 64 ...	512 kBytes	128 kBytes	50	100 MHz
☆	STM32F4...	STM32F4...	Active	2.9314		LQFP 64 ...	512 kBytes	128 kBytes	50	100 MHz
☆	STM32F4...	STM32F4...	Active	3.1366		LQFP 64 ...	512 kBytes	128 kBytes	50	100 MHz
☆	STM32F4...	STM32F4...	Active	3.3705		LQFP 64 ...	512 kBytes	256 kBytes	50	100 MHz
☆	STM32F4...	STM32F4...	Active	3.3705		LQFP 64 ...	512 kBytes	256 kBytes	50	100 MHz
☆	STM32F4...	STM32F4...	Active	3.6064		LQFP 64 ...	512 kBytes	256 kBytes	50	100 MHz
☆	STM32F4...	STM32F4...	Active	3.6064		LQFP 64 ...	512 kBytes	256 kBytes	50	100 MHz

- STM32F411RE microcontroller with LQFP64 package
- **Project Name: GPIO**



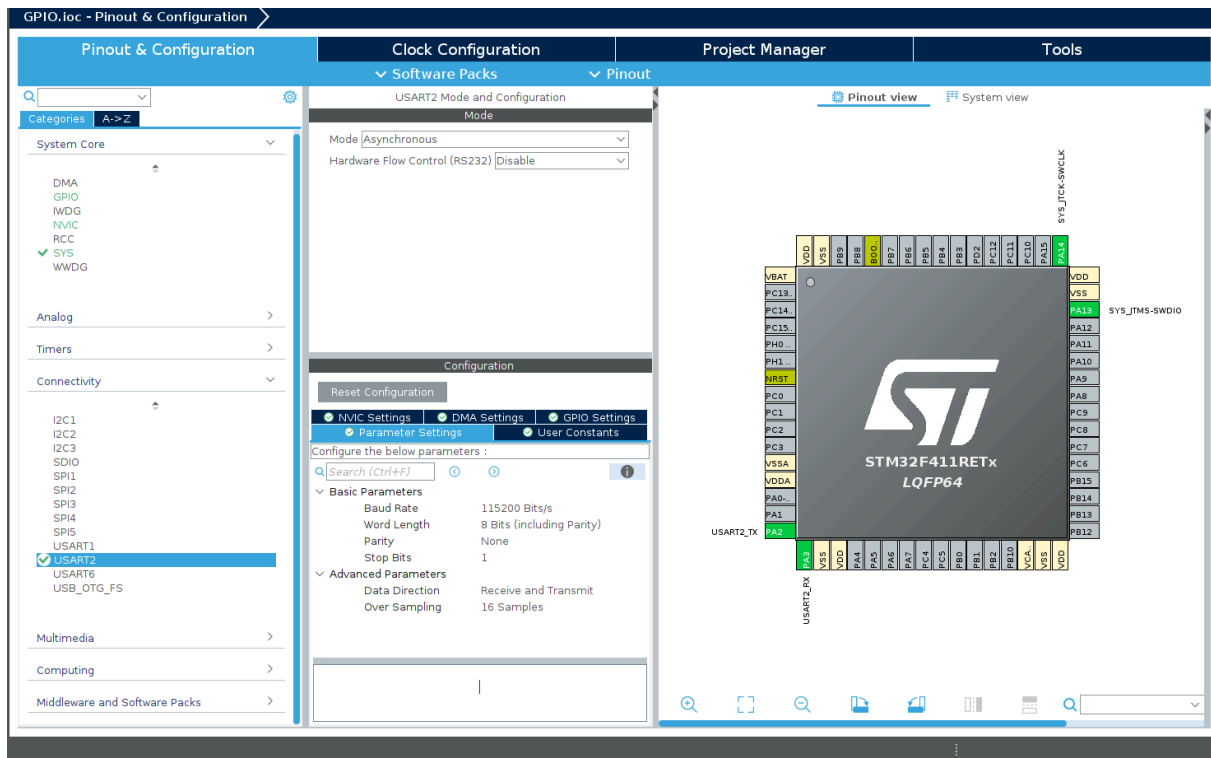
5.1.2. Pinout & Configuration 설정

1. SYS 설정

Categories → System Core → SYS 의 Debug 를 Serial Wire

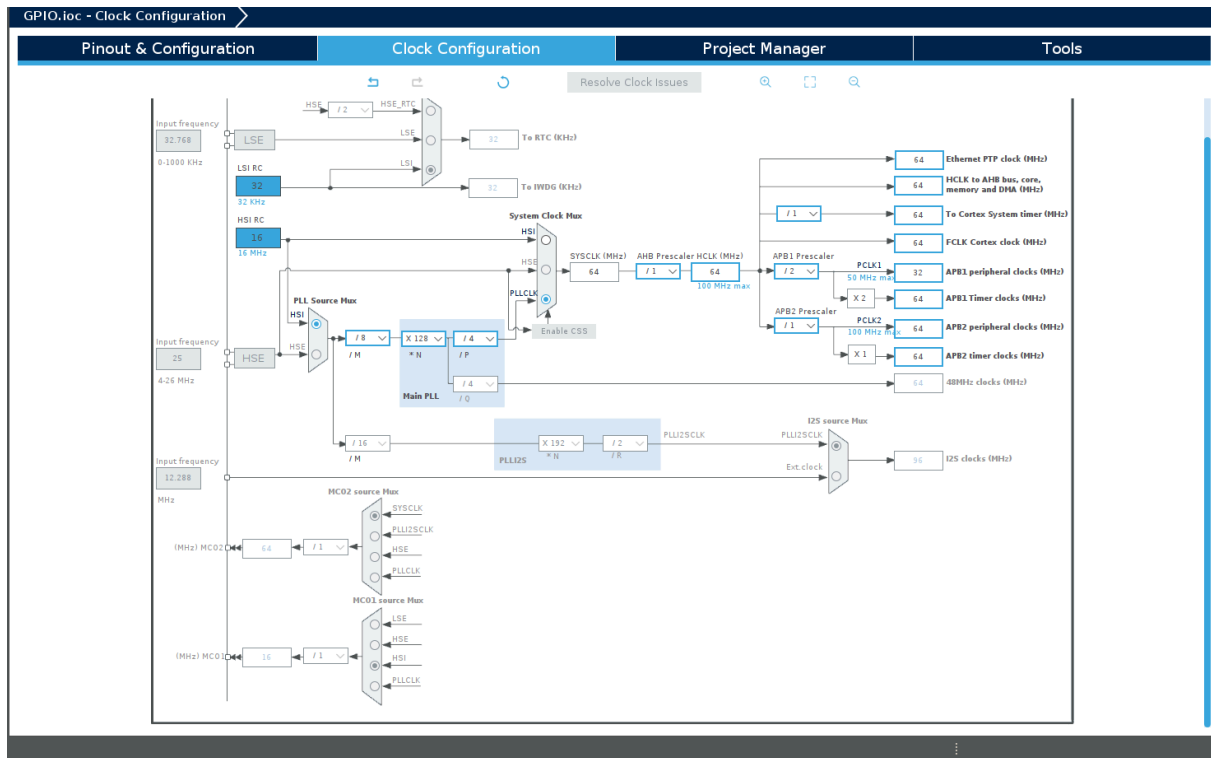
2. USART 설정

Categories → Connectivity → USART2 → Mode → Asynchronous
Configuration 확인



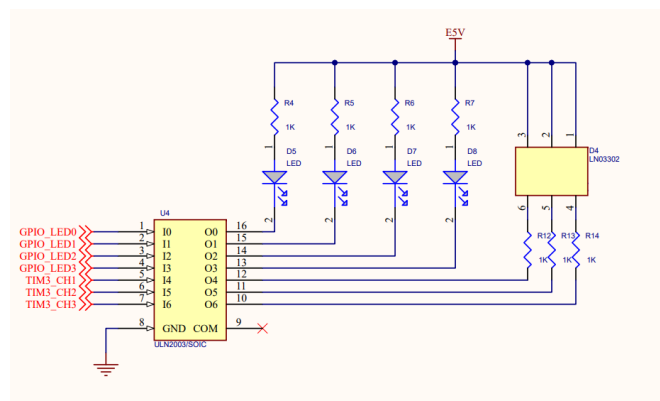
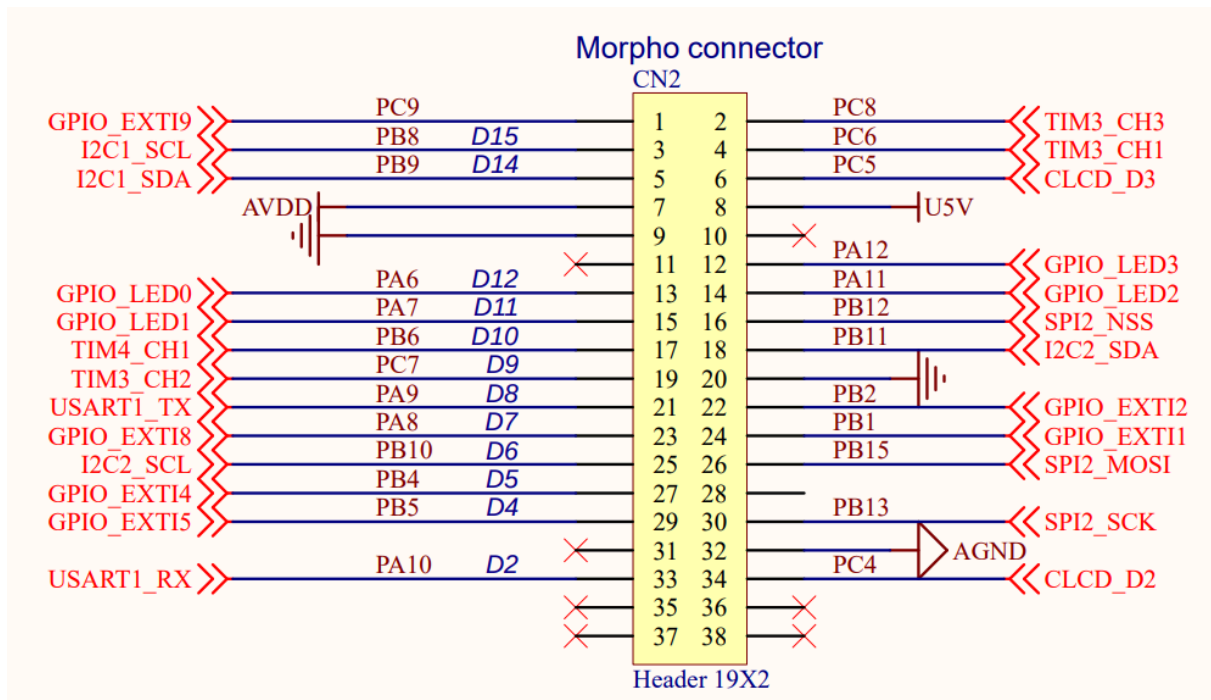
5.1.3. Clock Configuration 설정

- 기본 설정은 HCLK 16MHz
- PLL 클럭을 분주시켜 내부 클럭의 최대 클럭인 64MHz 사용하기 위해
- (근데 저희는 32MHz가 최대인가 봅니다)
- PLLM = 16, PLLN = 256, PLLP = 8 선택
- System Clock Mux에서 PLLCLK 선택
- HCLK 64MHz 설정 완료
- APB1 Prescaler = 2



5.1.4. LED Output 설정

- GPIO_LED0 > PA6
- GPIO_LED1 > PA7
- GPIO_LED2 > PA11
- GPIO_LED3 > PA12

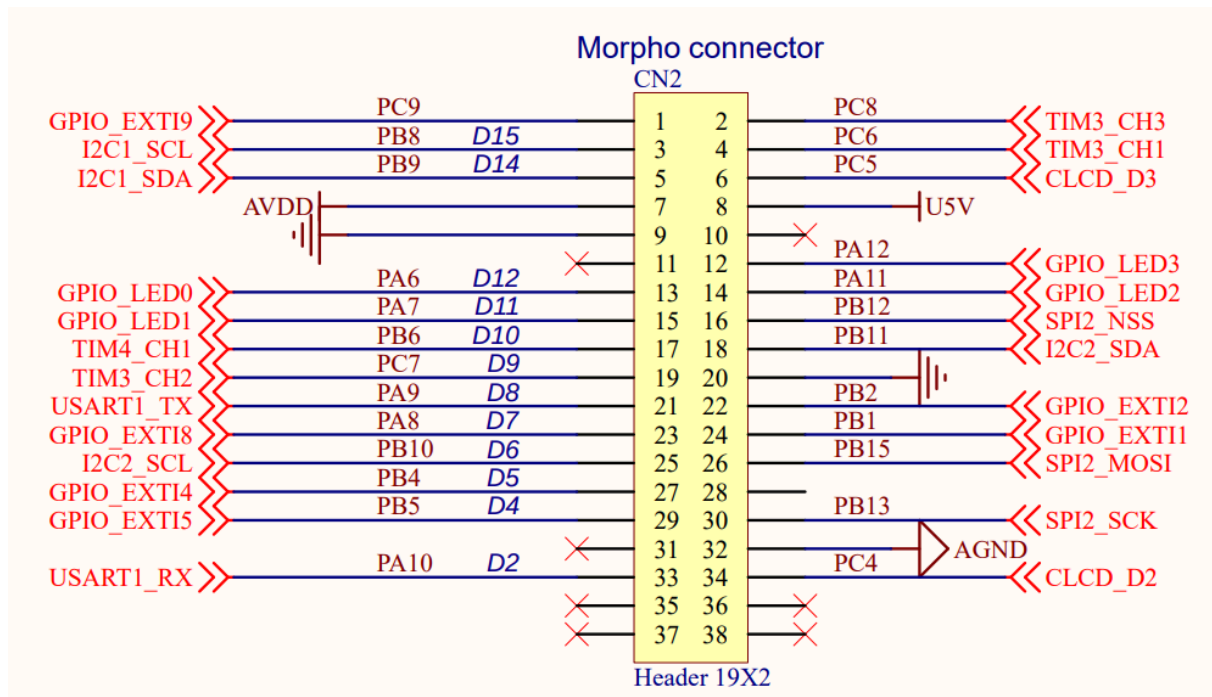


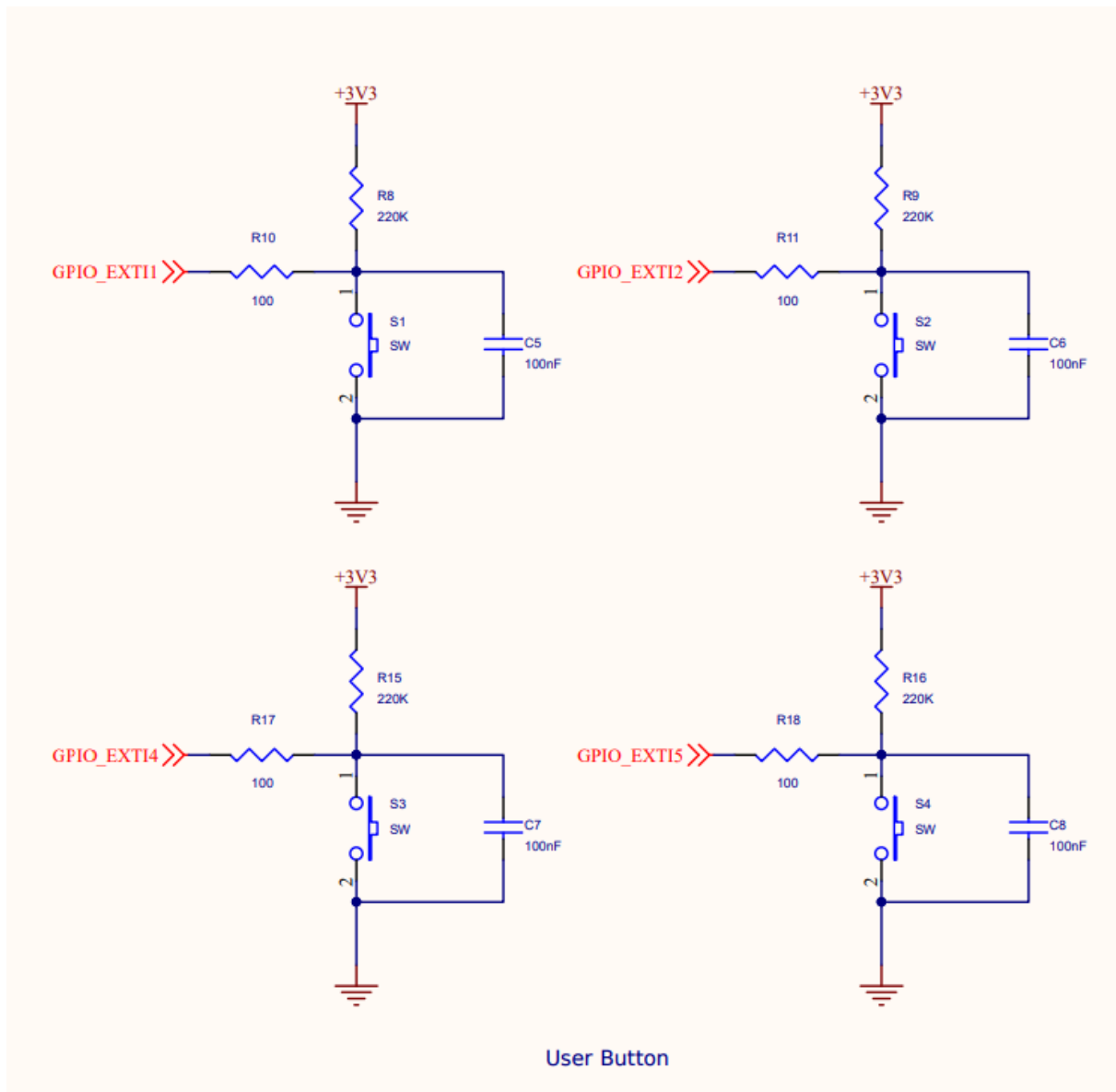
- ULN2003 다윈턴 트랜지스터 배열(Darlington Transistor Array), 작은 전류로 큰 전류를 제어할 수 있는 IC
- ULN2003이 GPIO 신호를 받아서 출력 단자를 GND로 연결하는 역할
- GPIO가 HIGH → LED ON
- **Pinout & Configuration** - **Pinout view** - 각 핀을 해당하는 LED로 설정
- **마우스 왼쪽 버튼** - **GPIO output**, **마우스 오른쪽 버튼** - **Label** 입력

5.1.5. 버튼 Input 설정

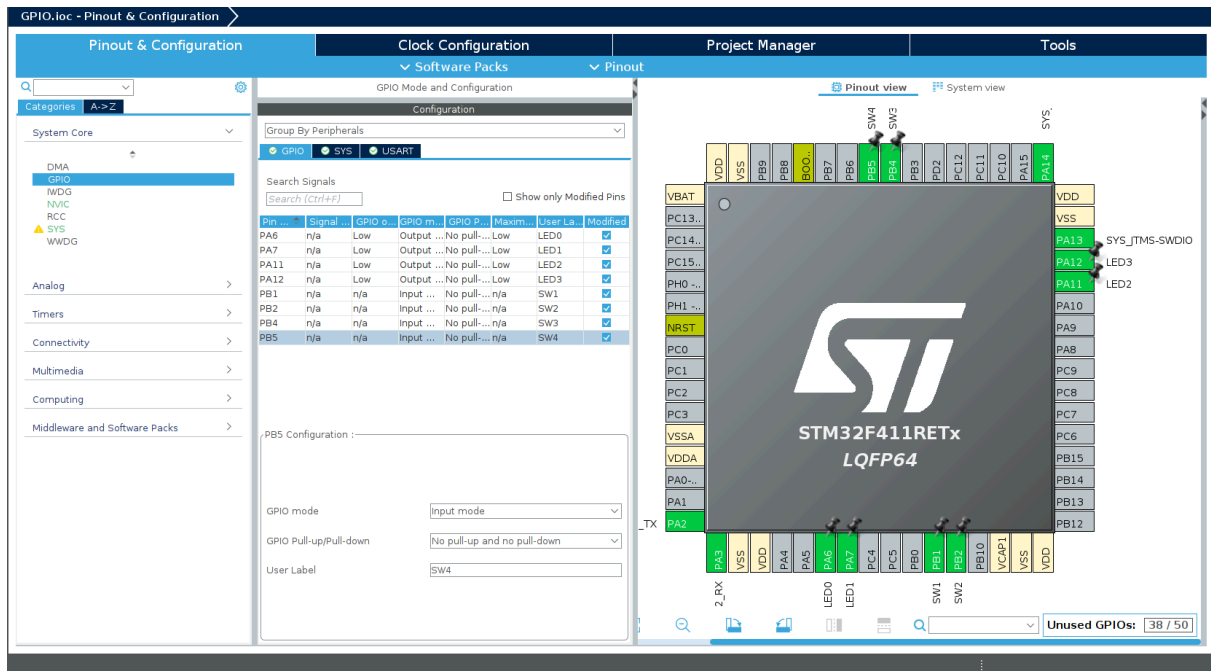
- 버튼 회로 확인
- GPIO_EXTI1 > PB1 - SW0
- GPIO_EXTI2 > PB2 - SW1

- GPIO_EXTI4 > PB4 - SW2
- GPIO_EXTI5 > PB5 - SW3

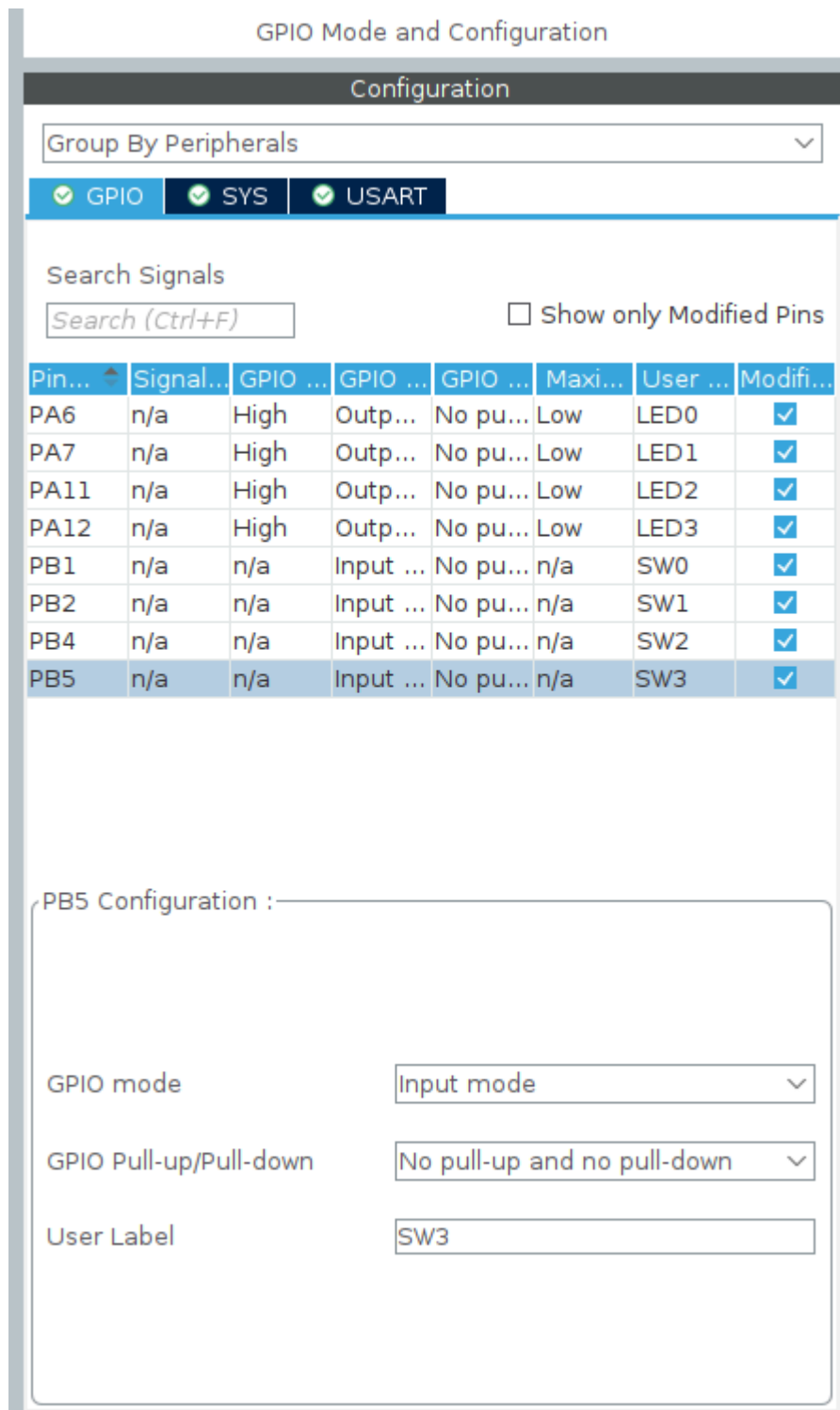




- 마우스 왼쪽 버튼 - GPIO input , 마우스 오른쪽 버튼 - Label 입력
- Pinout & Configuration - System Core - GPIO - Configuration



- LED 동작은 ULN 2003 IC 특성으로 인해 MCU 핀 출력이 high 일 때 LED ON
- 초기에 LED가 켜지도록 GPIO output level High로 수정



5.1.6. GENERATE CODE

- **Project** - **Generate Code**

5.1.7. 빌드 및 실행

- **빌드** 및 **실행** 하여 LED에 불이 들어오는 것을 확인

5.1.8. 소스 코드 작성

- 98번 라인: LED를 모두 OFF시키기 위하여 핀 출력을 LOW로 설정
- while() 루프: 버튼 4개를 순차적으로 읽어 들이는 **Polling** 방식으로 SW 버튼을 누르면 핀 입력값이 **Low** 가 되어 LED 출력 핀을 **High(GPIO_PIN_SET)** 로 변경하여 LED에 불이 켜지도록 함.

code

```
/* USER CODE BEGIN WHILE */
HAL_GPIO_WritePin (GPIOA, LED0_Pin | LED1_Pin | LED2_Pin | LED3_Pin, GPIO_PIN_RESET);
while (1)
{

    if (HAL_GPIO_ReadPin (SW0_GPIO_Port, SW0_Pin))
        HAL_GPIO_WritePin (LED0_GPIO_Port, LED0_Pin, GPIO_PIN_RESET);
    else
        HAL_GPIO_WritePin (LED0_GPIO_Port, LED0_Pin, GPIO_PIN_SET);

    if (HAL_GPIO_ReadPin (SW0_GPIO_Port, SW1_Pin))
        HAL_GPIO_WritePin (LED0_GPIO_Port, LED1_Pin, GPIO_PIN_RESET);
    else
        HAL_GPIO_WritePin (LED0_GPIO_Port, LED1_Pin, GPIO_PIN_SET);

    if (HAL_GPIO_ReadPin (SW0_GPIO_Port, SW2_Pin))
        HAL_GPIO_WritePin (LED0_GPIO_Port, LED2_Pin, GPIO_PIN_RESET);
    else
        HAL_GPIO_WritePin (LED0_GPIO_Port, LED2_Pin, GPIO_PIN_SET);

    if (HAL_GPIO_ReadPin (SW0_GPIO_Port, SW3_Pin))
        HAL_GPIO_WritePin (LED0_GPIO_Port, LED3_Pin, GPIO_PIN_RESET);
    else
        HAL_GPIO_WritePin (LED0_GPIO_Port, LED3_Pin, GPIO_PIN_SET);
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
}
```

- 라벨: main.h 파일에 define 확인

```

56
57 /* USER CODE END EFP */
58
59 /* Private defines -----
60 #define LED0_Pin GPIO_PIN 6
61 #define LED0_GPIO_Port GPIOA
62 #define LED1_Pin GPIO_PIN 7
63 #define LED1_GPIO_Port GPIOA
64 #define SW0_Pin GPIO_PIN 1
65 #define SW0_GPIO_Port GPIOB
66 #define SW1_Pin GPIO_PIN 2
67 #define SW1_GPIO_Port GPIOB
68 #define LED2_Pin GPIO_PIN 11
69 #define LED2_GPIO_Port GPIOA
70 #define LED3_Pin GPIO_PIN 12
71 #define LED3_GPIO_Port GPIOA
72 #define SW2_Pin GPIO_PIN 4
73 #define SW2_GPIO_Port GPIOB
74 #define SW3_Pin GPIO_PIN 5
75 #define SW3_GPIO_Port GPIOB
76
77 /* USER CODE BEGIN Private defines */
78

```

- 빌드 및 실행
- 버튼을 누르고 있는 동안 LED가 On
- polling 방식은 권하지 않음.
 - 주기적으로 어떤 조건을 체크해서 특정 작업을 수행하는 방식
 - 마이크로컨트롤러가 일정 시간 간격으로 버튼 상태(또는 다른 입력 신호)를 확인하고, 입력값에 따라 다른 동작을 수행하는 방식
- HAL_GPIO_ReadPin 함수와 같은 HAL 드라이버를 사용하여 레지스터를 직접 다루지 않더라도 입출력 제어가 가능

5.2. EXTI

5.2.1. EXTI PushButton

- 이전 프로그램: 버튼 입력 핀 값을 읽어 LED 출력 핀으로 출력하는 프로그램을 polling 방식으로 구현
- 이번 실습: 기존 CubeMX 프로젝트 파일인 ioc 파일을 수정하여 앞선 프로그램을 interrupt 방식으로 변경 구현

5.2.1.1. STM32CubeIDE 프로젝트 생성

- **File** - **New** - **STM32 Project From STM32CubeMX .ioc File**
- **Browse** : 기존 생성한 GPIO 프로젝트 폴더 내 GPIO.ioc 파일 **Open**
- Project Name: EXTI_PushButton
- 파일 잘 선택되었는지 확인 후 **Finish**

STM32 Project From Existing STM32CubeMX Configuration File (.ioc)

IDE

Setup STM32 project

STM32CubeMX .ioc file

File: /home/ahrelee/STM32CubeIDE/workspace/GPIO/GPIO.ioc

Browse...

Project

Project Name: EXTI_PushButton

☒ Use default location

Location: /home/ahrelee/STM32CubeIDE/workspace

Browse...

Options

Targeted Language

☒ C ☐ C++


Targeted Device Usage

Targeted Binary Type

☒ Executable ☐ Static Library

Targeted Project Type

☒ STM32Cube ☐ Empty

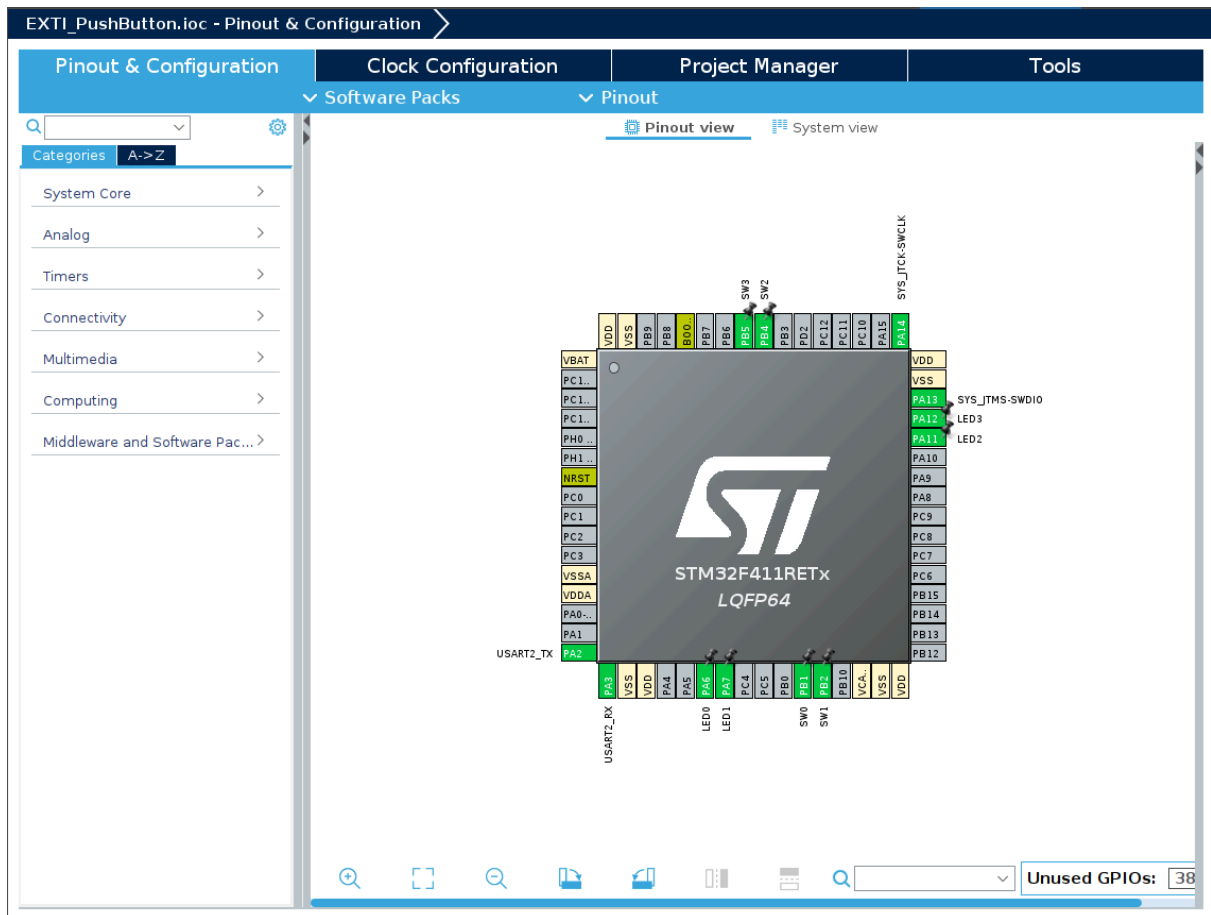


< Back

Next >

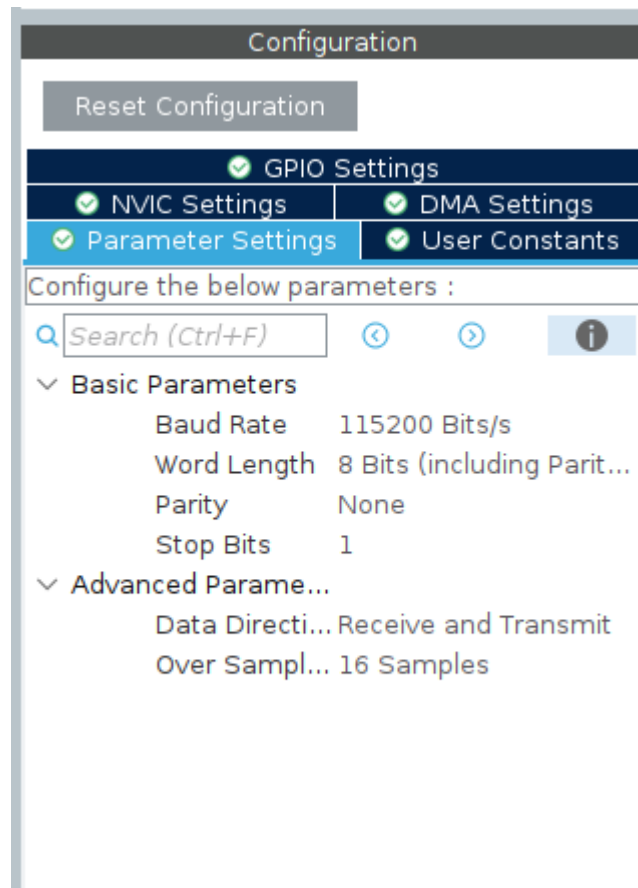
Cancel

Finish



5.2.1.2. Pinout & Configuration 설정

- Categories - System Core - SYS: Debug - Serial Wire
- Categories - Connectivity - USART2: Mode - Asynchronous / 사진 참고



5.2.1.3. Clock Configuration 설정

- 기존과 동일

5.2.1.4. 버튼 Input 설정

- PB1, PB2, PB4, PB5 각 핀에서 **마우스 왼쪽 버튼** 을 눌러 GPIO_EXTI 속성으로 변경

EXTI_PushButton.ioc - Pinout & Configuration

Pinout & Configuration | Clock Configuration | Project Manager

Software Packs | Pinout

GPIO Mode and Configuration

Configuration

Group By Peripherals

GPIO SYS USART NVIC

Search Signals
Search (Ctrl+F) ☐ Show only Modified Pins

Pin Name	Signal on Pin	GPIO output...	GPIO mode	GPIO Pull-u...	Maximum o...	User Label	Modified
PA6	n/a	Low	Output Pus...	No pull-up ...	Low	LED0	<input checked="" type="checkbox"/>
PA7	n/a	Low	Output Pus...	No pull-up ...	Low	LED1	<input checked="" type="checkbox"/>
PA11	n/a	Low	Output Pus...	No pull-up ...	Low	LED2	<input checked="" type="checkbox"/>
PA12	n/a	Low	Output Pus...	No pull-up ...	Low	LED3	<input checked="" type="checkbox"/>
PB1	n/a	n/a	External Int...	No pull-up ...	n/a	SW0	<input checked="" type="checkbox"/>
PB2	n/a	n/a	External Int...	No pull-up ...	n/a	SW1	<input checked="" type="checkbox"/>
PB4	n/a	n/a	External Int...	No pull-up ...	n/a	SW2	<input checked="" type="checkbox"/>
PB5	n/a	n/a	External Int...	No pull-up ...	n/a	SW3	<input checked="" type="checkbox"/>

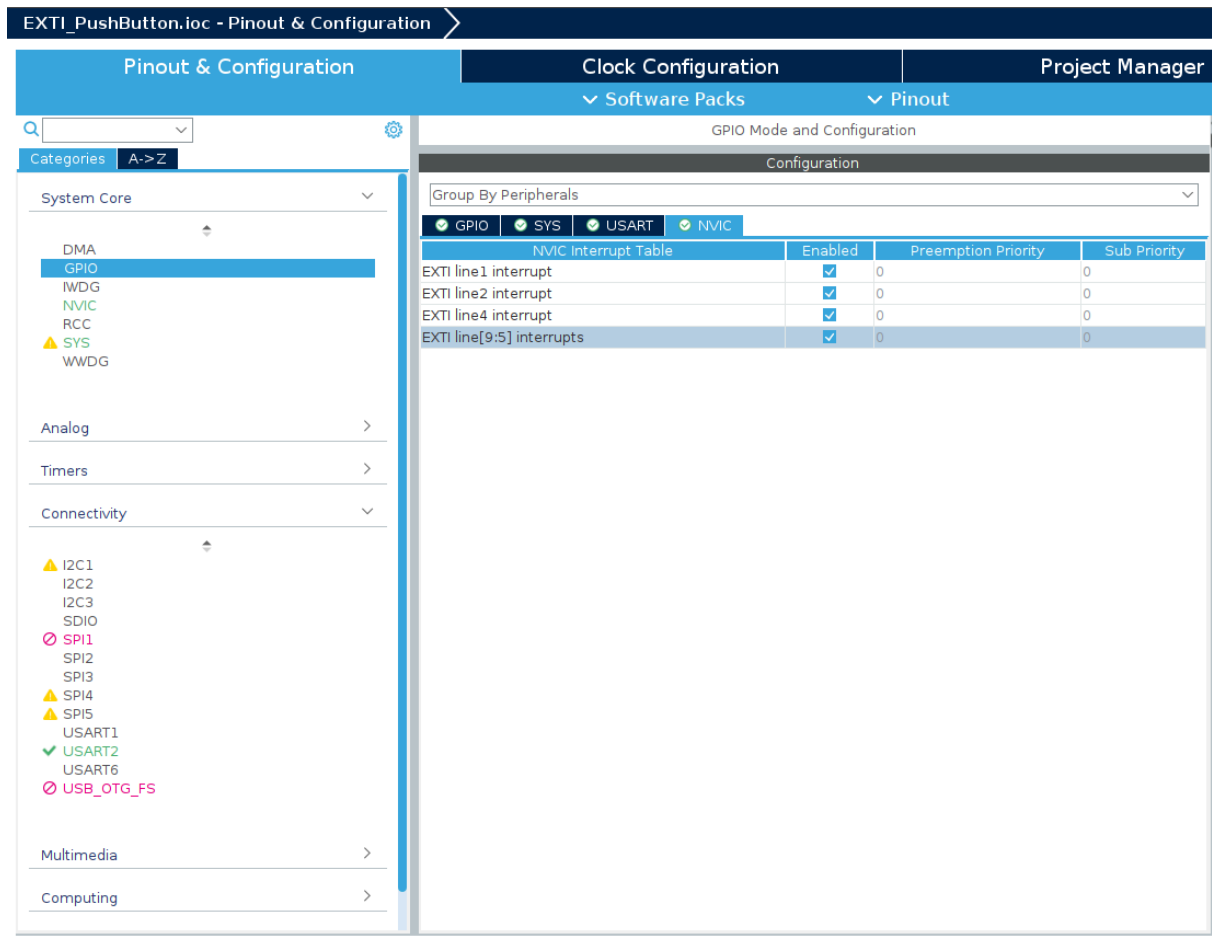
PB5 Configuration :

GPIO mode: External Interrupt Mode with Rising/Falling edge trigger detection

GPIO Pull-up/Pull-down: No pull-up and no pull-down

User Label: SW3

- Configuration - NVIC 모두 Enabled



5.2.1.5. GENERATE CODE

- **Project** - **Generate Code**
- **main.c** - MX_GPIO_init() - 자동 생성된 EXTI 초기화 코드 확인

```

211  /*Configure GPIO pins : SW0 Pin SW1 Pin SW2 Pin SW3 Pin */
212  GPIO_InitStruct.Pin = SW0_Pin|SW1_Pin|SW2_Pin|SW3_Pin;
213  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING_FALLING;
214  GPIO_InitStruct.Pull = GPIO_NOPULL;
215  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
216
217  /* EXTI interrupt init*/
218  HAL_NVIC_SetPriority(EXTI1_IRQn, 0, 0);
219  HAL_NVIC_EnableIRQ(EXTI1_IRQn);
220
221  HAL_NVIC_SetPriority(EXTI2_IRQn, 0, 0);
222  HAL_NVIC_EnableIRQ(EXTI2_IRQn);
223
224  HAL_NVIC_SetPriority(EXTI4_IRQn, 0, 0);
225  HAL_NVIC_EnableIRQ(EXTI4_IRQn);
226
227  HAL_NVIC_SetPriority(EXTI9_5_IRQn, 0, 0);
228  HAL_NVIC_EnableIRQ(EXTI9_5_IRQn);
229
230  /* USER CODE BEGIN MX_GPIO_Init_2 */
231  /* USER CODE END MX_GPIO_Init_2 */
232  }

```

- `stm32f4xx_it.c` - 인터럽트 서비스 루틴 생성 확인

The screenshot shows the STM32CubeIDE interface. The main editor displays the `stm32f4xx_it.c` file, which contains three interrupt service routines (ISRs) for EXTI lines 1, 2, and 4. Each ISR is a void function that calls `HAL_GPIO_EXTI_IRQHandler` with the corresponding pin number (SW0, SW1, and SW2 respectively). The code is annotated with user code begin/end markers. The Outline view on the right lists the project files, including `main.h` and `stm32f4xx_it.h`, and shows the ISRs as functions defined in the project.

```

200
201  /**
202   * @brief This function handles EXTI line1 interrupt.
203   */
204  void EXTI1_IRQHandler(void)
205  {
206      /* USER CODE BEGIN EXTI1_IRQn 0 */
207
208      /* USER CODE END EXTI1_IRQn 0 */
209      HAL_GPIO_EXTI_IRQHandler(SW0_Pin);
210      /* USER CODE BEGIN EXTI1_IRQn 1 */
211
212      /* USER CODE END EXTI1_IRQn 1 */
213  }
214
215  /**
216   * @brief This function handles EXTI line2 interrupt.
217   */
218  void EXTI2_IRQHandler(void)
219  {
220      /* USER CODE BEGIN EXTI2_IRQn 0 */
221
222      /* USER CODE END EXTI2_IRQn 0 */
223      HAL_GPIO_EXTI_IRQHandler(SW1_Pin);
224      /* USER CODE BEGIN EXTI2_IRQn 1 */
225
226      /* USER CODE END EXTI2_IRQn 1 */
227  }
228
229  /**
230   * @brief This function handles EXTI line4 interrupt.
231   */
232  void EXTI4_IRQHandler(void)
233  {
234      /* USER CODE BEGIN EXTI4_IRQn 0 */
235
236      /* USER CODE END EXTI4_IRQn 0 */
237      HAL_GPIO_EXTI_IRQHandler(SW2_Pin);
238      /* USER CODE BEGIN EXTI4_IRQn 1 */

```

5.2.1.6. 소스 코드 작성

- 인터럽트 발생 시, HAL_GPIO_EXTI_IRQHandler() → HAL_GPIO_EXTI_Callback() 함수 호출
- 사용자는 콜백 함수에 실행할 코드를 구현하면 됨.
- `code` - 234 line
- 스위치를 누를 때(Falling edge) - 떨어(Rising edge) 때 모두 인터럽트가 발생
- 스위치 상태 파악을 위해 핀의 현재 상태를 체크

```
/* USER CODE BEGIN 4 */
void
HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin)
{
    switch (GPIO_Pin)
    {
        case GPIO_PIN_1:
            if (HAL_GPIO_ReadPin (SW0_GPIO_Port, SW0_Pin))
                HAL_GPIO_WritePin (LED0_GPIO_Port, LED0_Pin, GPIO_PIN_RESET);
            else
                HAL_GPIO_WritePin (LED0_GPIO_Port, LED0_Pin, GPIO_PIN_SET);
            break;
        case GPIO_PIN_2:
            if (HAL_GPIO_ReadPin (SW0_GPIO_Port, SW1_Pin))
                HAL_GPIO_WritePin (LED0_GPIO_Port, LED1_Pin, GPIO_PIN_RESET);
            else
                HAL_GPIO_WritePin (LED0_GPIO_Port, LED1_Pin, GPIO_PIN_SET);
            break;
        case GPIO_PIN_4:
            if (HAL_GPIO_ReadPin (SW0_GPIO_Port, SW2_Pin))
                HAL_GPIO_WritePin (LED0_GPIO_Port, LED2_Pin, GPIO_PIN_RESET);
            else
                HAL_GPIO_WritePin (LED0_GPIO_Port, LED2_Pin, GPIO_PIN_SET);
            break;
        case GPIO_PIN_5:
            if (HAL_GPIO_ReadPin (SW0_GPIO_Port, SW3_Pin))
                HAL_GPIO_WritePin (LED0_GPIO_Port, LED3_Pin, GPIO_PIN_RESET);
            else
```

```

        HAL_GPIO_WritePin (LED0_GPIO_Port, LED3_Pin, GPIO_PIN_SET);
        break;
    default:
        ;
    }
}
/* USER CODE END 4 */

```

5.2.1.7. 빌드 및 실행

5.2.2. EXTI_Encoder

- 외부 인터럽트를 이용한 엔코더 스위치 응용 프로그램 구현

5.2.2.1. 엔코더 스위치

- NUCLEOEVB 보드에 사용되는 엔코더 스위치는 BOURNS사의 PEC11R-4020K-S0024
- 1회전 시 펄스가 24개 발생

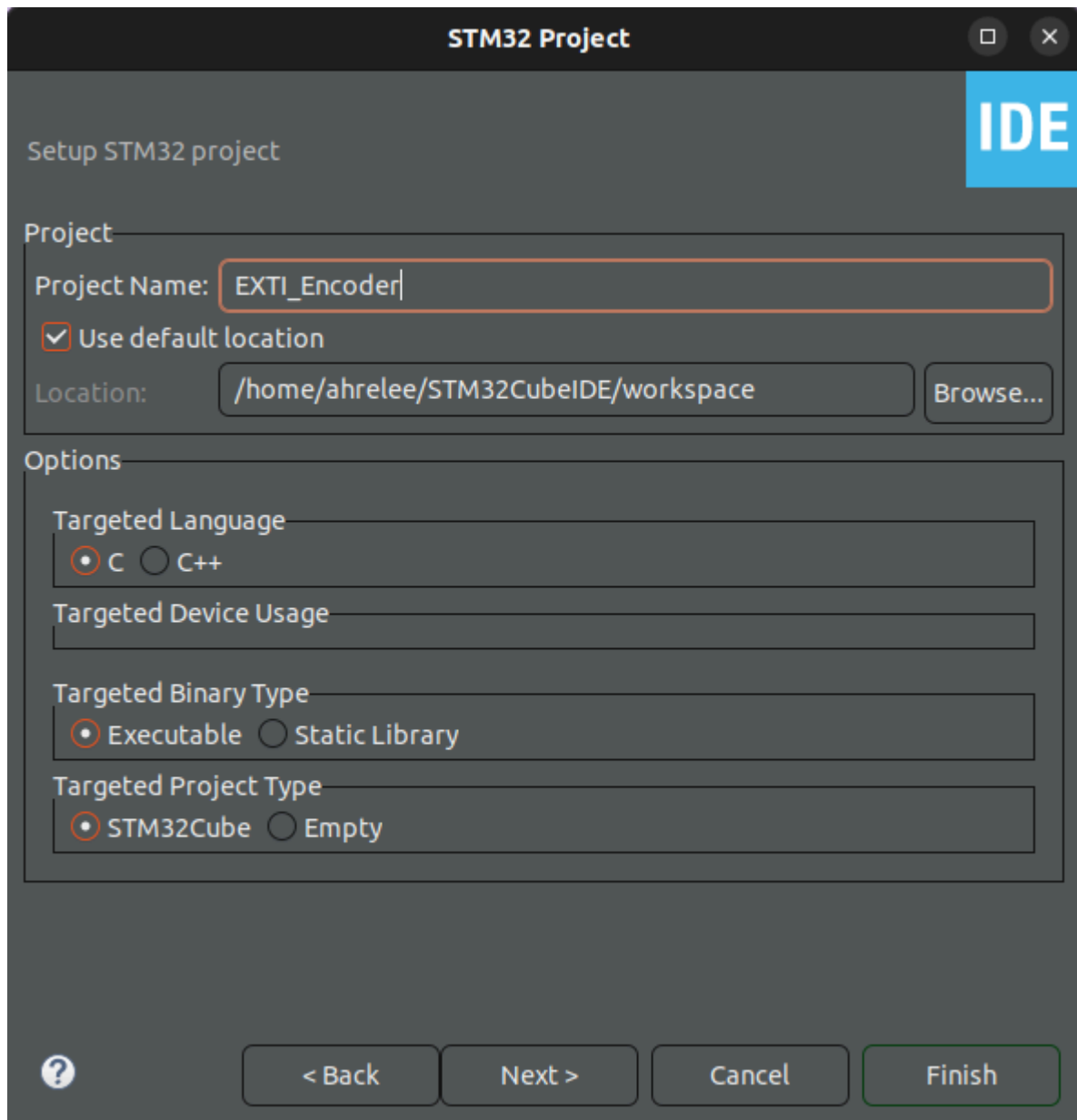
+ encoder

- 회전축이나 선형 이동을 측정하고 이를 전기 신호로 변환하는 장치
- CW 방향: A signal 에서 rising edge 먼저 발생
- CCW 방향: B signal 에서 rising edge 먼저 발생

⇒ signal의 인터럽트가 먼저 발생했을 때, 나머지 signal의 신호가 High인지 Low 인지 확인하여 방향 파악 가능

5.2.2.2. STM32CubeIDE 프로젝트 생성

- New - STM32 Project - STM32F411RET6 선택
- Project Name: EXTI_Encoder



5.2.2.3. Pinout & Configuration

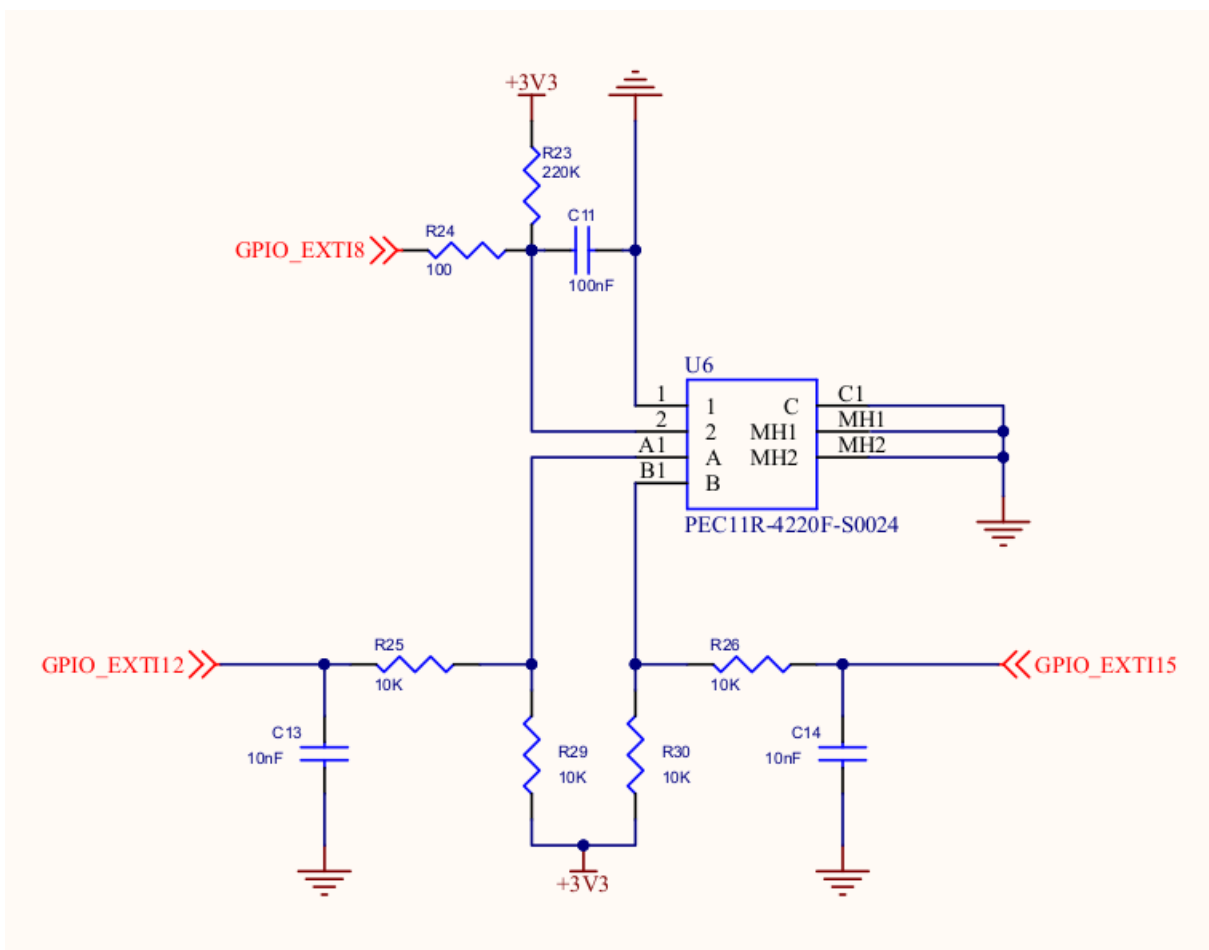
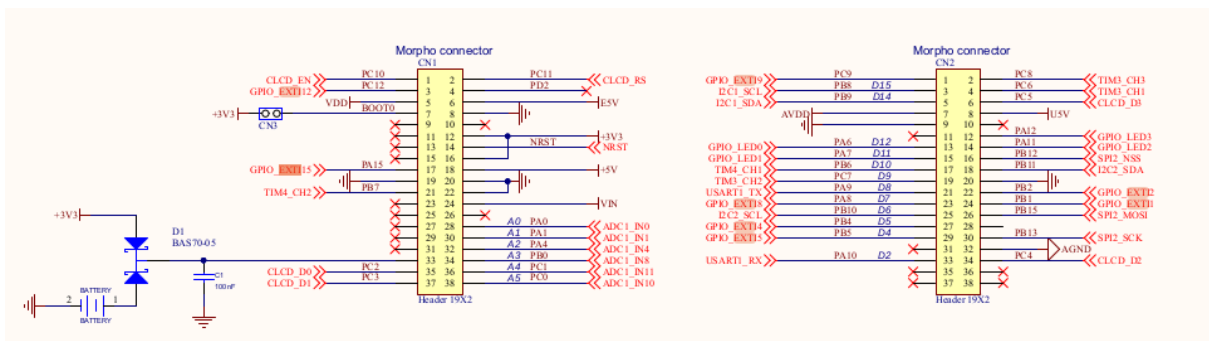
- Categories - System Core - SYS: Debug - Serial Wire
- Categories - Connectivity - USART2: Mode - Asynchronous

5.2.2.4. Clock Configuration 설정

- 위와 동일하게 ?
- PLL 클럭을 분주시켜 내부 클럭의 최대 클럭인 64MHz 32MHz 사용하기 위해
- PLLM = 16, PLLN = 128, PLLP = 4 선택

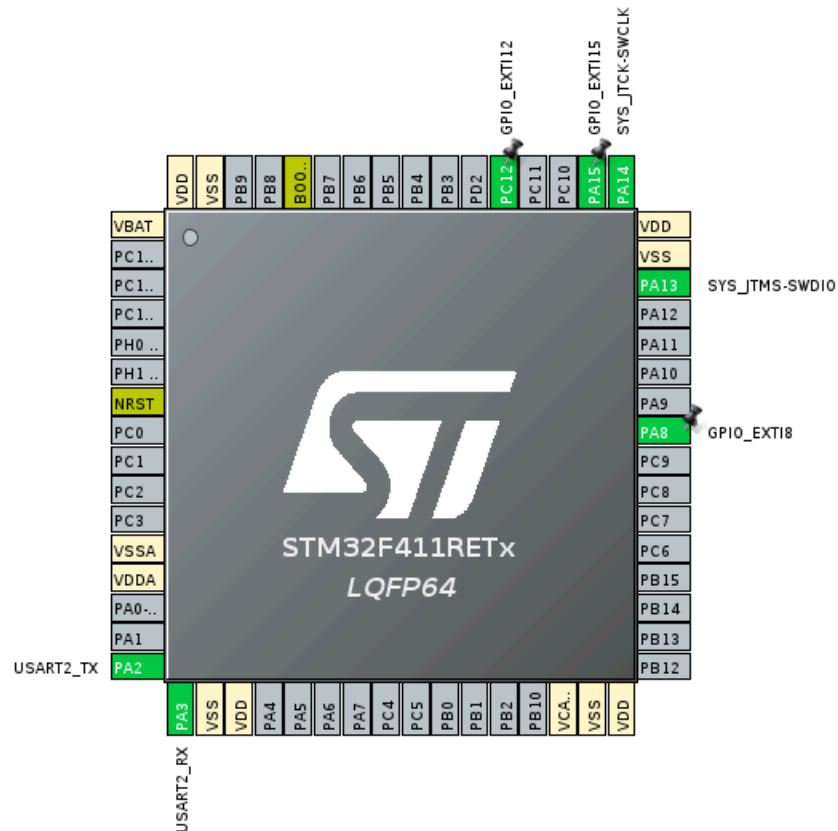
5.2.2.5. Encoder 설정

1. Pinout 설정



A signal - PC12, B signal - PA15

엔코더 스위치 노드를 누르면 PA8 신호가 High - Low



PC12, PA15, PA8을 EXTI로 설정

2. GPIO Configuration 설정

GPIO Configuration

- PA8: Falling - SW
- PA15: Rising edge trigger - B_SIG
- PC12: Rising edge trigger - A_SIG

EXTI_Encoder.ioc - Pinout & Configuration

Pinout & Configuration

Clock Configuration

Software Packs

Pinout

Categories

A->Z

System Core

DMA

GPIO

IWDG

NVIC

▲ RCC

▲ **SYS**

WWDG

Analog

Timers

Connectivity

Multimedia

Computing

Middleware and Software Packs

GPIO Mode and Configuration

Configuration

Group By Peripherals

GPIO

SYS

USART

NVIC

Search Signals

Search (Ctrl+F)

☐ Show only Modified Pins

Pin N...	Signal on...	GPIO out...	GPIO mo...	GPIO Pull...	Maximu...	User Label	Modified
PA8	n/a	n/a	External ...	No pull-u...	n/a	SW	<input checked="" type="checkbox"/>
PA15	n/a	n/a	External ...	No pull-u...	n/a	B_SIG	<input checked="" type="checkbox"/>
PC12	n/a	n/a	External ...	No pull-u...	n/a	A_SIG	<input checked="" type="checkbox"/>

PC12 Configuration :

GPIO mode

External Interrupt Mode with Rising edge trigger detection

GPIO Pull-up/Pull-down

No pull-up and no pull-down

User Label

A_SIG

NVIC - 인터럽트 테이블 활성화


```

/* USER CODE BEGIN Includes */
#include <stdio.h>
/* USER CODE END Includes */

```

- 53 line

code

```

/* USER CODE BEGIN PFP */
#ifdef __GNUC__
#define PUTCHAR_PROTOTYPE int __io_putchar(int ch)
#else
#define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
#endif /* __GNUC__ */

/**
 * @brief Retargets the C library printf function to the USART.
 * @param None
 * @retval None
 */
PUTCHAR_PROTOTYPE
{
    /* Place your implementation of fputc here */
    /* e.g. write a character to the USART1 and Loop
       until the end of transmission */
    if (ch == '\n')
        HAL_UART_Transmit (&huart2, (uint8_t*) "\r", 1, 0xFFFF);
    HAL_UART_Transmit (&huart2, (uint8_t*) &ch, 1, 0xFFFF);

    return ch;
}
/* USER CODE END PFP */

```

2. 인터럽트 콜백 함수

HAL_GPIO_EXTI_Callback() 함수를 통해 노브를 누르면 발생하는 인터럽트에서 변수값을 초기화

A signal 인터럽트가 발생했을 때, B signal 신호 레벨 = Low면 시계 방향으로 회전 → 변수값 증가

B signal 인터럽트가 발생했을 때, A signal 신호 레벨 = Low면 반시계 방향으로 회전 → 변수값 감소

- 45 line

code

```
/* USER CODE BEGIN PV */  
volatile int gEncoderCnt;  
/* USER CODE END PV */
```

- 259 line

code

```
/* USER CODE BEGIN 4 */  
void  
HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin)  
{  
    switch (GPIO_Pin)  
    {  
        case GPIO_PIN_8: // SW_Pin  
            gEncoderCnt = 0;  
            break;  
  
        case GPIO_PIN_12: // A_SIG_Pin  
            if (!HAL_GPIO_ReadPin (B_SIG_GPIO_Port, B_SIG_Pin))  
                gEncoderCnt++;  
            break;  
  
        case GPIO_PIN_15: // B_SIG_Pin  
            if (!HAL_GPIO_ReadPin (A_SIG_GPIO_Port, A_SIG_Pin))  
                gEncoderCnt--;
```

```

        break;

    default:
        ;
    }
}
/* USER CODE END 4 */

```

3. 엔코더 카운터 변수값 출력

- 시리얼 디버깅을 이용하여 엔코더 카운터 변수값을 출력하는 루틴
- 변수값이 변경될 때만 시리얼이 출력되도록 구현

- 119 line

code

```

/* USER CODE BEGIN WHILE */
int preEncoderCnt = 0;

while (1)
{
    if (preEncoderCnt != gEnCoderCnt)
    {
        preEncoderCnt = gEnCoderCnt;
        printf ("%d\n", preEncoderCnt);
    }
}
/* USER CODE END WHILE */
/* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
}

```

5.2.2.8. 빌드 및 실행

5.2.2.9. 디버깅

- `Run` - `Debug` - `Switch`
- `>> 3` - Live Expressions
- Add new expression - gEncoderCnt
- 265 line - Toggle Breakpoint