

알람시계 ADC

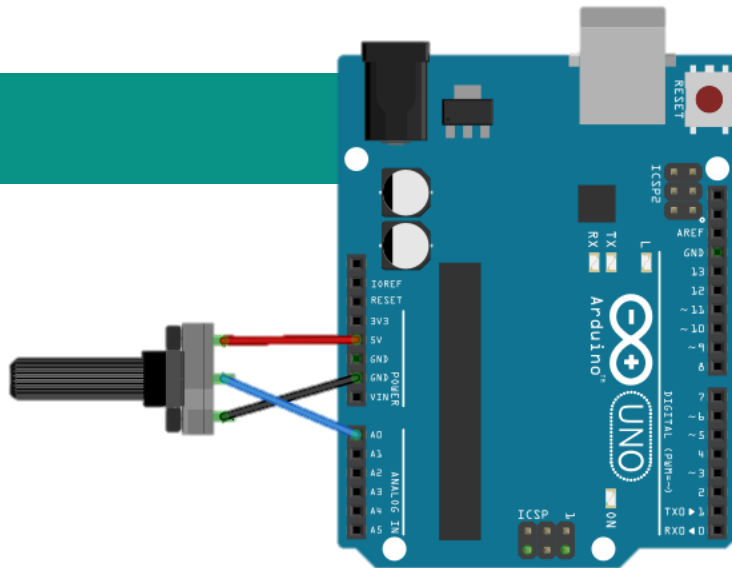


아두이노 ADC

가변 저항

- 3개의 핀으로 구성됨

핀 번호	핀 이름	연결
1번	전원 핀	● 5V 핀
2번	접지 핀	● GND
3번	출력 핀	● A0 핀



[출처: Fritzing]

→ 아두이노에 가변저항을 연결하고 UART로 출력값을 확인하시오.

복습

 오픈플랫폼 입문의 가변저항제어

 아두이노 ADC 설정과 기본 함수 정리

...> <https://cafe.naver.com/ctcemb/2539>

STM32F429의 ADC

STM32F429의 ADC 소개

ADC란?

Analog to Digital Converter

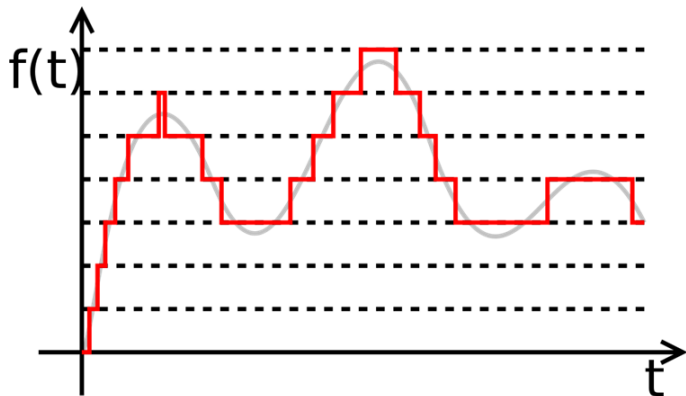
아날로그 신호를 **디지털**로 변환해주는 장치

STM32F429의 ADC

STM32F429의 ADC 소개

ADC의 회로기호

아날로그 신호를 디지털로 변환



ADC 전기기호



STM32F429의 ADC

STM32F429의 ADC 소개

STM32F429의 ADC 특징

- ... 3개의 ADC 컨트롤러
- ... 각각 최대 12비트의 해상도
- ... 12비트는 2의 12승이므로 0~4095까지의 범위로 디지털 값을 얻음
- ... 최대 24개의 채널
 - 동시에 아날로그 신호를 24개까지 처리할 수 있다는 의미
- ... 처리속도는 7.2MSPS
 - MSPS는 Mega Sampling Per Second이며 초당 7.2mega의 속도로 샘플링 가능

STM32F429의 ADC

STM32F429의 ADC 소개

STM32F429의 ADC 특징

13.2 ADC main features

- 12-bit, 10-bit, 8-bit or 6-bit configurable resolution
- Interrupt generation at the end of conversion, end of injected conversion, and in case of analog watchdog or overrun events
- Single and continuous conversion modes
- Scan mode for automatic conversion of channel 0 to channel 'n'
- Data alignment with in-built data coherency
- Channel-wise programmable sampling time
- External trigger option with configurable polarity for both regular and injected conversions
- Discontinuous mode
- Dual/Triple mode (on devices with 2 ADCs or more)
- Configurable DMA data storage in Dual/Triple ADC mode
- Configurable delay between conversions in Dual/Triple interleaved mode
- ADC conversion type (refer to the datasheets)
- ADC supply requirements: 2.4 V to 3.6 V at full speed and down to 1.8 V at slower speed
- ADC input range: $V_{REF-} \leq V_{IN} \leq V_{REF+}$
- DMA request generation during regular channel conversion

Figure 44 shows the block diagram of the ADC.

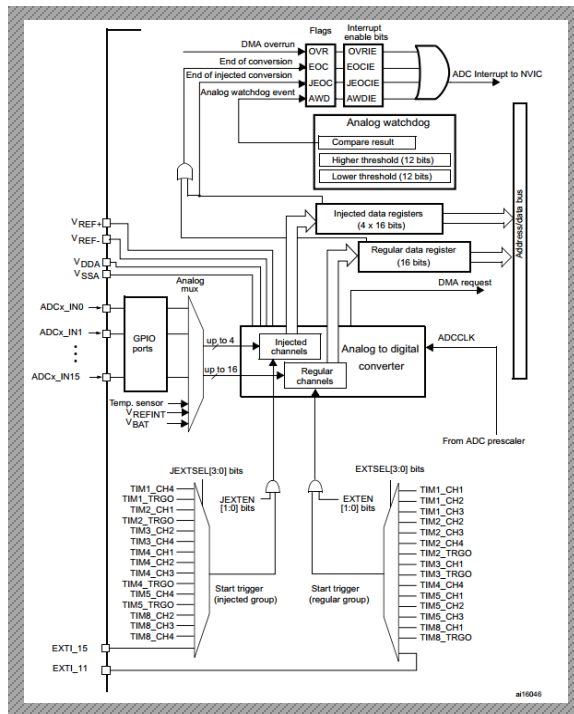
Note: V_{REF-} if available (depending on package), must be tied to V_{SSA} .

STM32F429의 ADC 주요 특징

STM32F429의 ADC

STM32F429의 ADC 소개

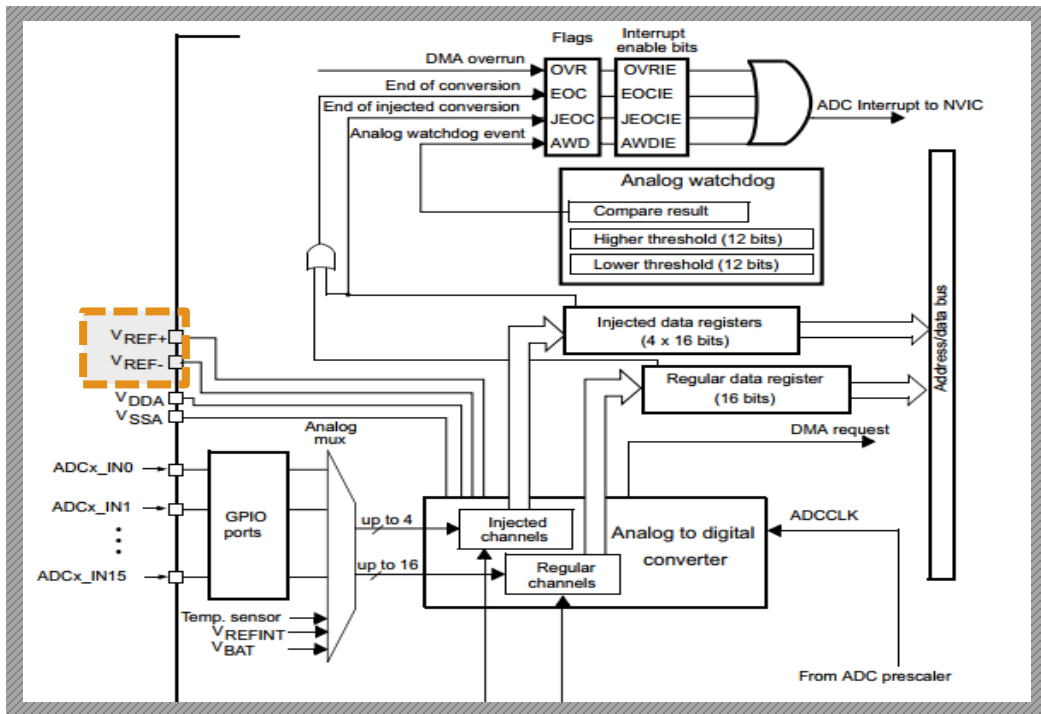
STM32F429의 ADC의 블록도



STM32F429의 ADC

STM32F429의 ADC 소개

STM32F429의 ADC의 블록도



V_{REF+} 와 V_{REF-}

... V_{REF+} 와 V_{REF-} 라는 이름의
기준 전압을 입력 받음

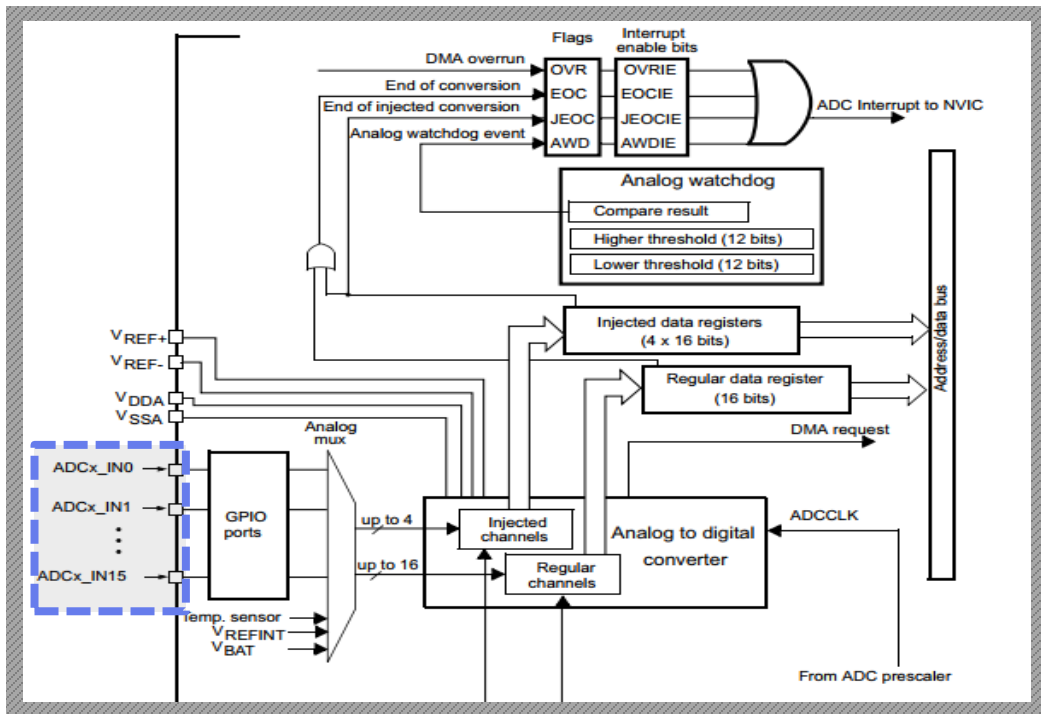
기준 전압

아날로그 신호의 음과 양의 최대 범위

STM32F429의 ADC

STM32F429의 ADC 소개

STM32F429의 ADC의 블록도



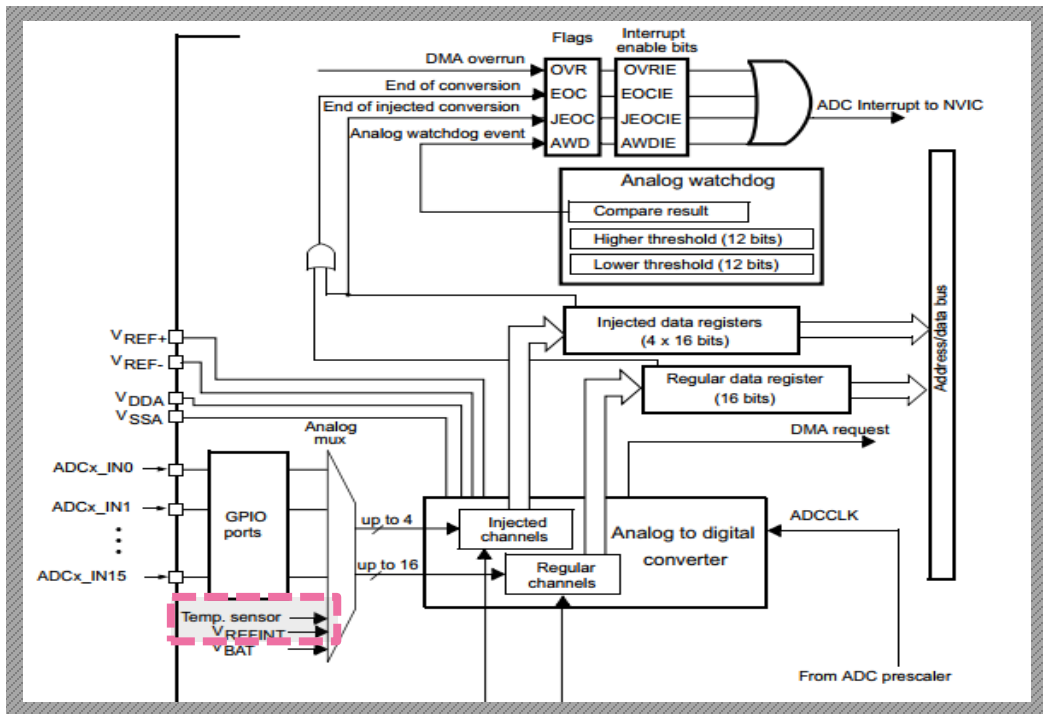
ADCx_IN0~ADCx_IN15

... ADCx_IN0부터
ADCx_IN15까지
16개의 입력

STM32F429의 ADC

STM32F429의 ADC 소개

STM32F429의 ADC의 블록도



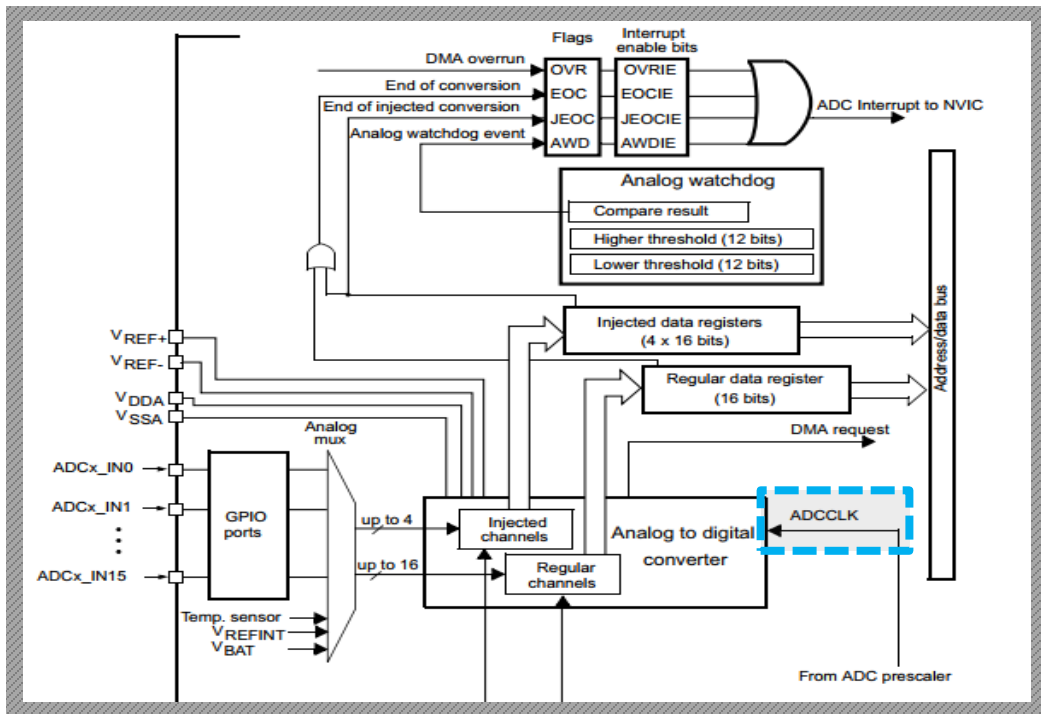
Temp sensor

- Temp sensor는 온도센서로 내부에 온도센서가 있어 ADC를 통해 디지털 온도 값을 받을 수 있음

STM32F429의 ADC

STM32F429의 ADC 소개

STM32F429의 ADC의 블록도



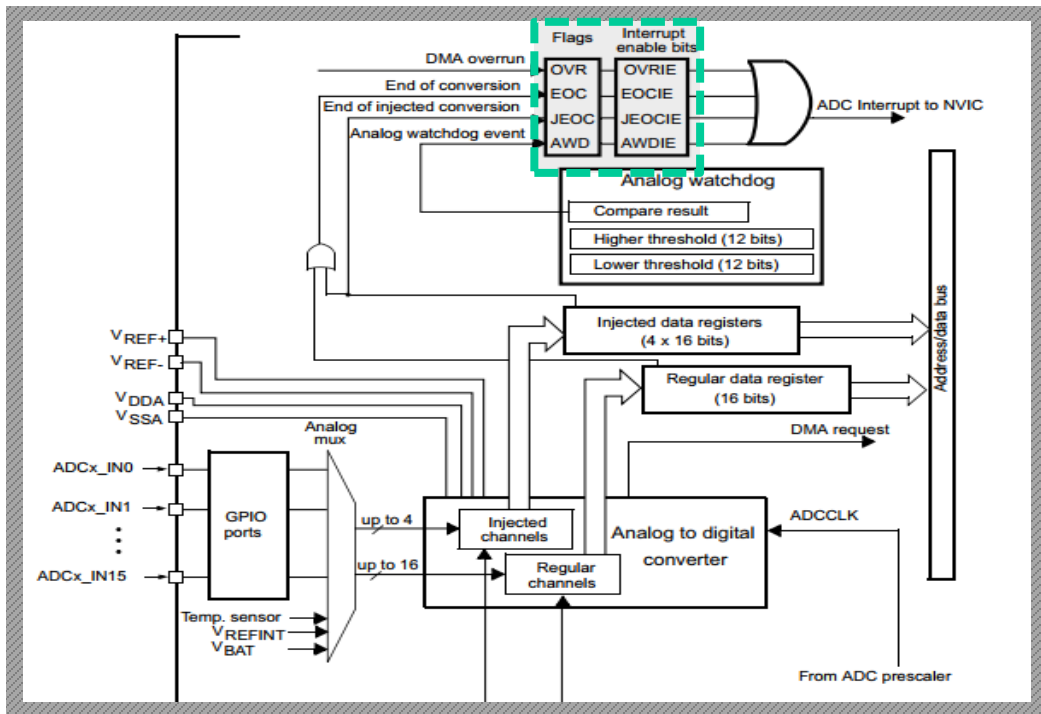
ADCCLK

ADCCLK라는 clock을
입력받아 컨트롤러를 동작시킴

STM32F429의 ADC

STM32F429의 ADC 소개

STM32F429의 ADC의 블록도



OVR, EOC, JEOC, AWD

ADC 인터럽트의 종류

STM32F429의 ADC

STM32F429의 ADC 소개

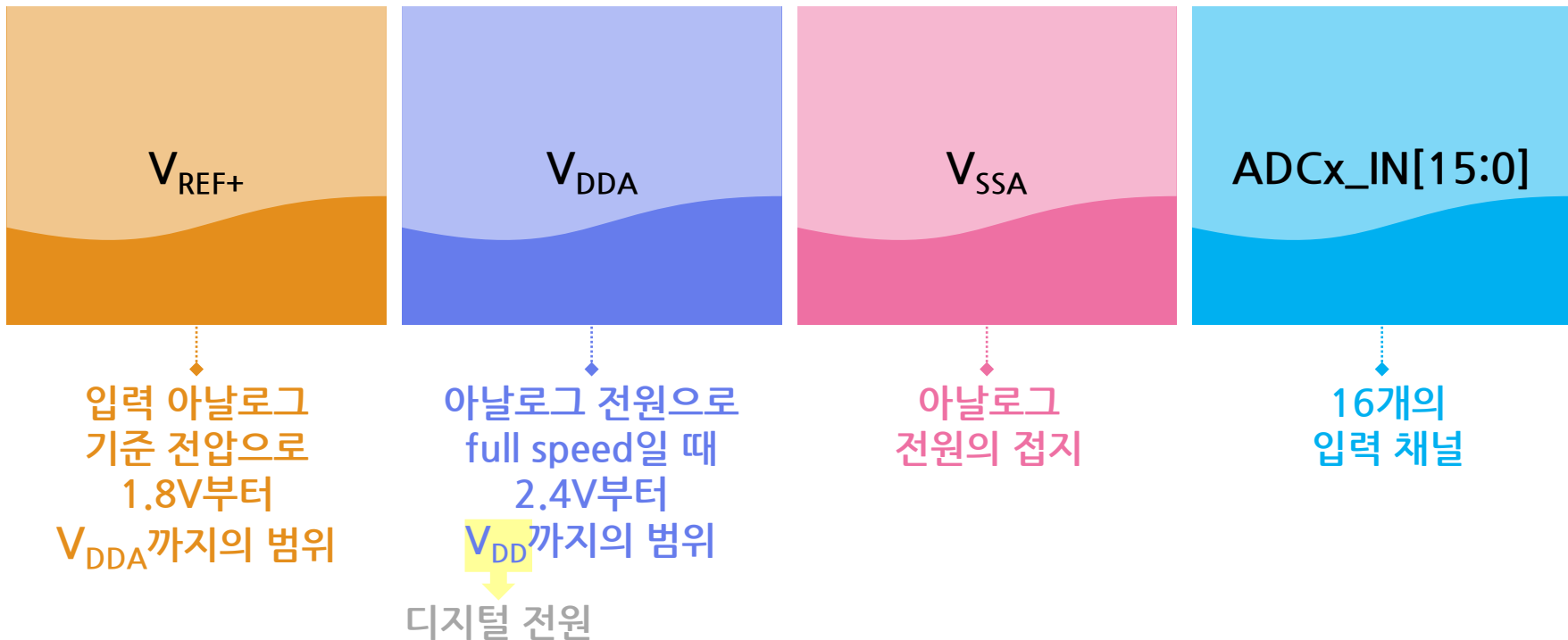
STM32F429의 ADC 신호들

Name	Signal type	Remarks
V_{REF+}	Input, analog reference positive	The higher/positive reference voltage for the ADC, $1.8V \leq V_{REF+} \leq V_{DDA}$
V_{DDA}	Input, analog supply	Analog power supply equal to V_{DD} and $2.4V \leq V_{DDA} \leq V_{DD}(3.6V)$ for full speed $1.8V \leq V_{DDA} \leq V_{DD}(3.6V)$ for reduced speed
V_{REF-}	Input, analog reference negative	The lower/negative reference voltage for the ADC, $V_{REF-} = V_{SSA}$
V_{SSA}	Input, analog supply ground	Ground for analog power supply equal to V_{SS}
ADCx_IN[15:0]	Analog input signals	16 analog input channels

STM32F429의 ADC

STM32F429의 ADC 소개

STM32F429의 ADC 신호들



STM32F429의 ADC

Nucleo-F429 보드의 ADC

Nucleo-F429 보드의 ADC 포트

Zio커넥터는 아두이노와 호환됨

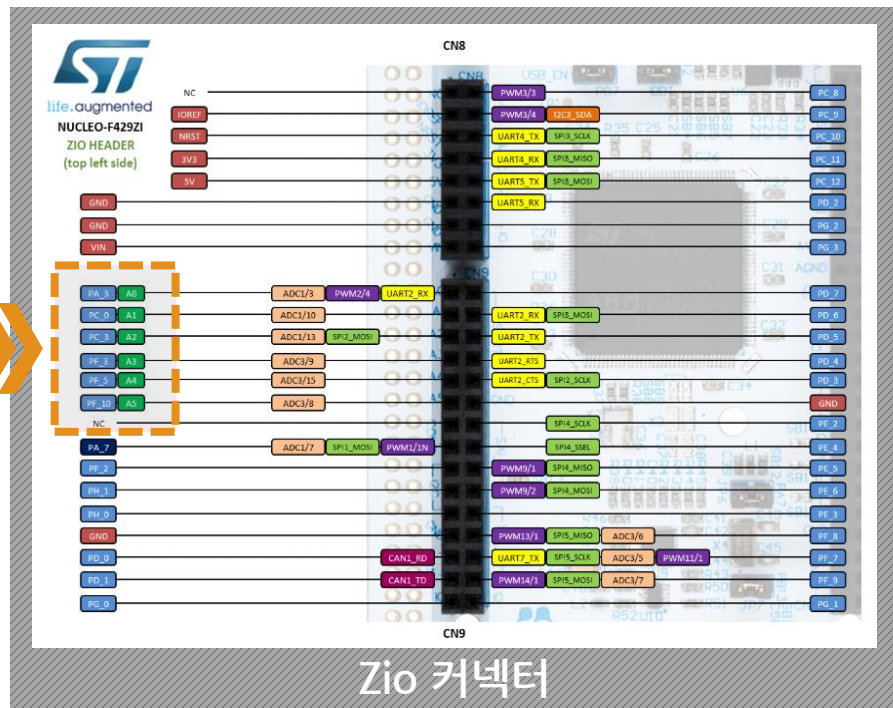


아두이노 아날로그 입력 핀과 동일한 위치에 ADC 포트가 존재함

Nucleo-F429 보드의 ADC

Nucleo-F429 보드의 ADC 포트

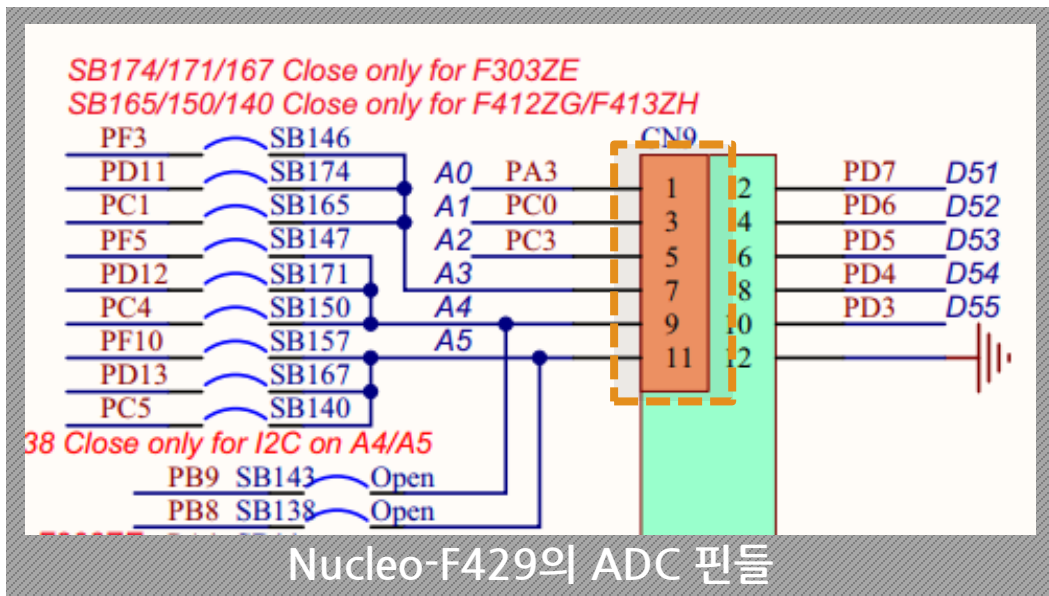
- 아두이노의 아날로그 입력핀과 동일한 위치의 핀들임
- 녹색 블록의 A0~A5로 표시된 핀들임



STM32F429의 ADC

⚙️ Nucleo-F429 보드의 ADC

🌈 Nucleo-F429 보드의 ADC 핀

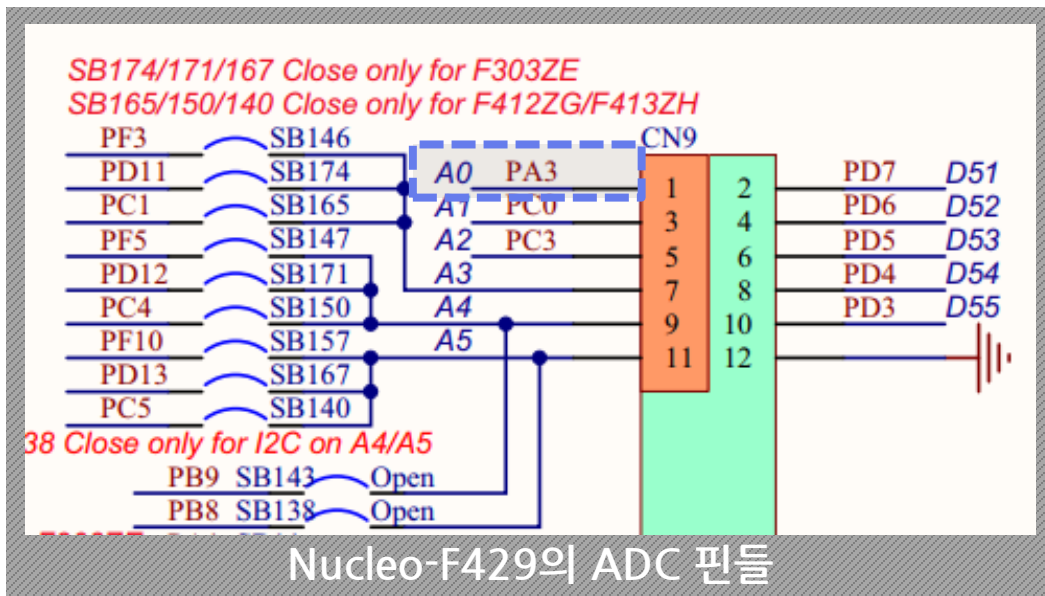


녹색 블록의 A0~A5로
표시된 핀은
Nucleo-F429보드의
회로도를 살펴보면
CN9의 1,3,5,7,9,11번 핀임

STM32F429의 ADC

⚙️ Nucleo-F429 보드의 ADC

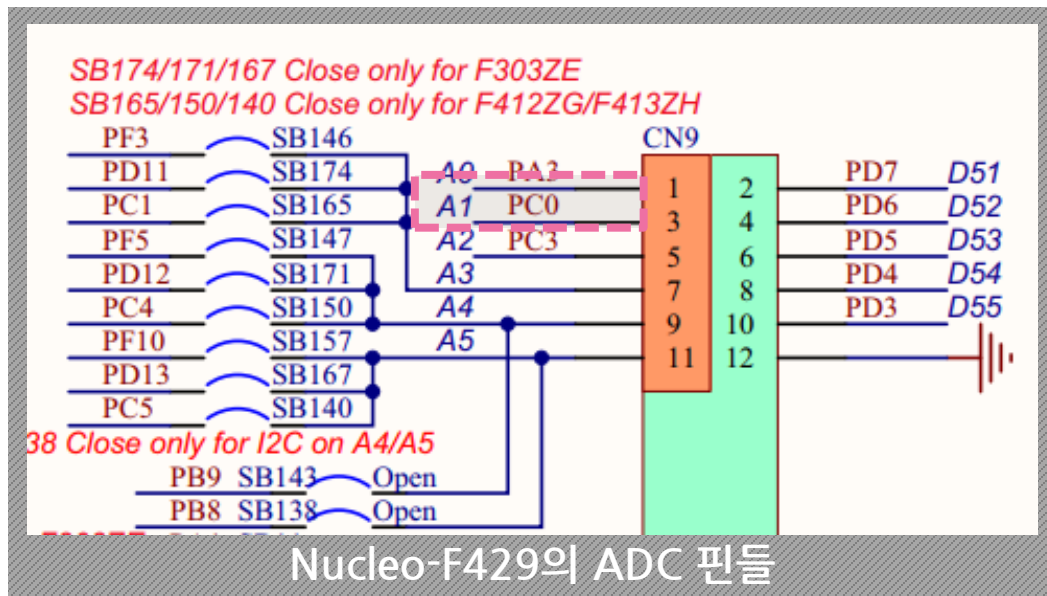
🌈 Nucleo-F429 보드의 ADC 핀



STM32F429의 ADC

⚙️ Nucleo-F429 보드의 ADC

🌈 Nucleo-F429 보드의 ADC 핀

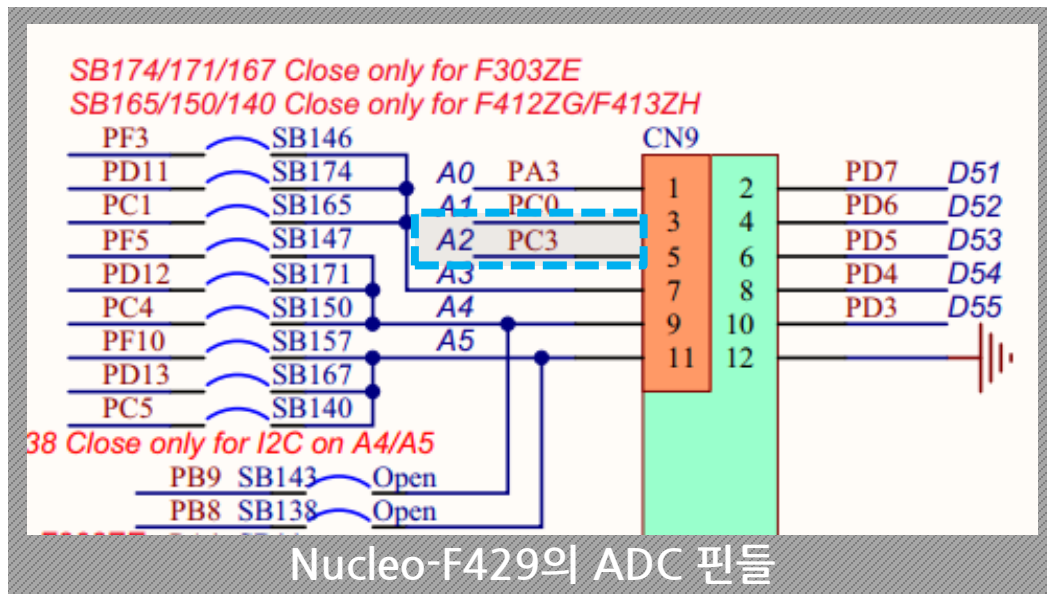


A1로 표시된 핀은
STM32F429의 PC0번 핀

STM32F429의 ADC

⚙️ Nucleo-F429 보드의 ADC

🌈 Nucleo-F429 보드의 ADC 핀

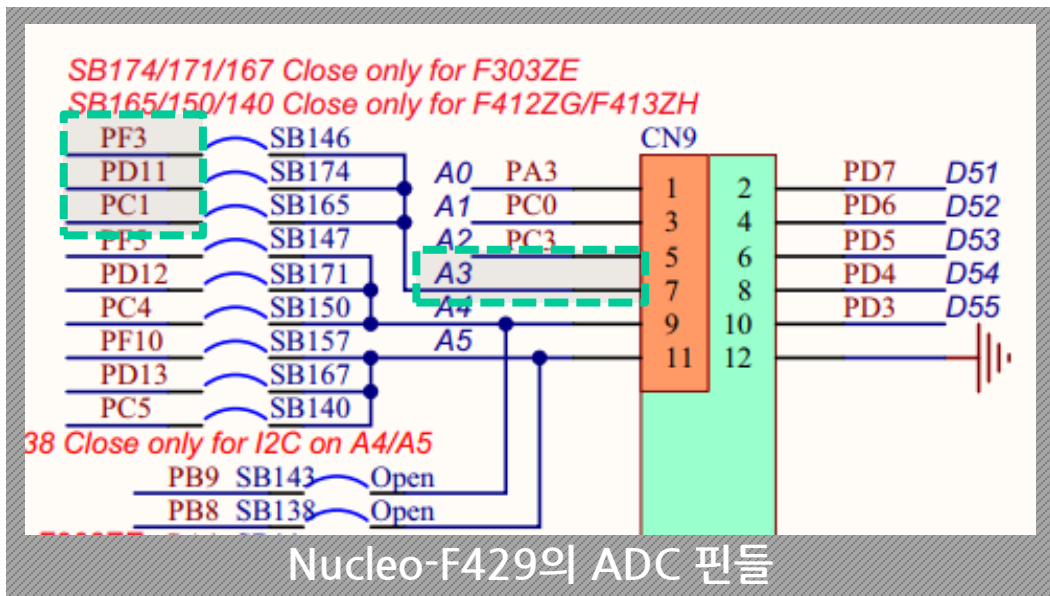


A2로 표시된 핀은
STM32F429의 PC3번 핀

STM32F429의 ADC

⚙️ Nucleo-F429 보드의 ADC

🌈 Nucleo-F429 보드의 ADC 핀

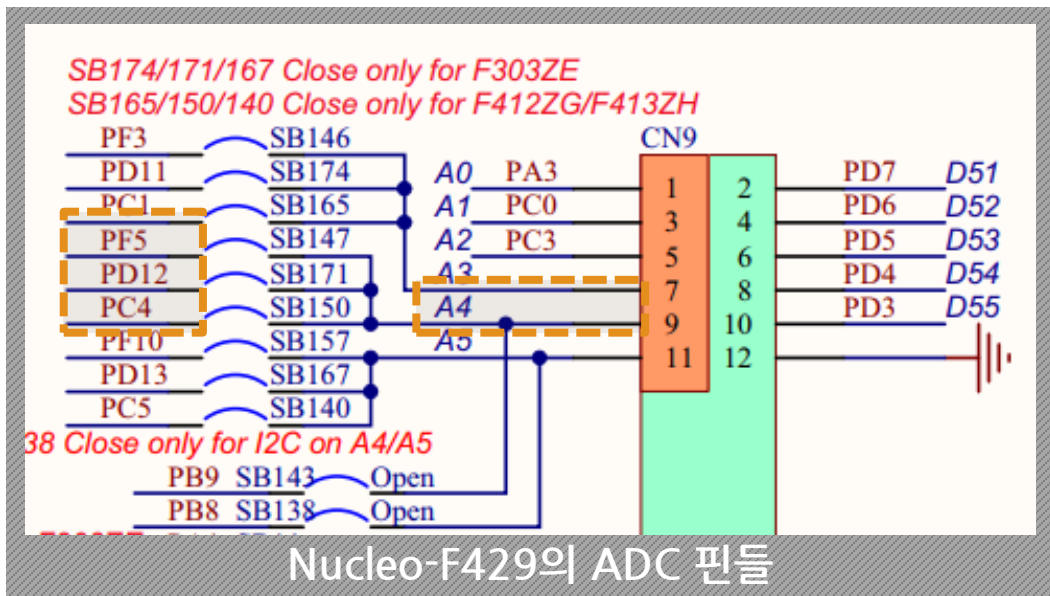


A3로 표시된 핀은
STM32F429의 PF3번 핀이나
PD11, PC1번 핀 중 선택 가능

STM32F429의 ADC

⚙️ Nucleo-F429 보드의 ADC

🌈 Nucleo-F429 보드의 ADC 핀

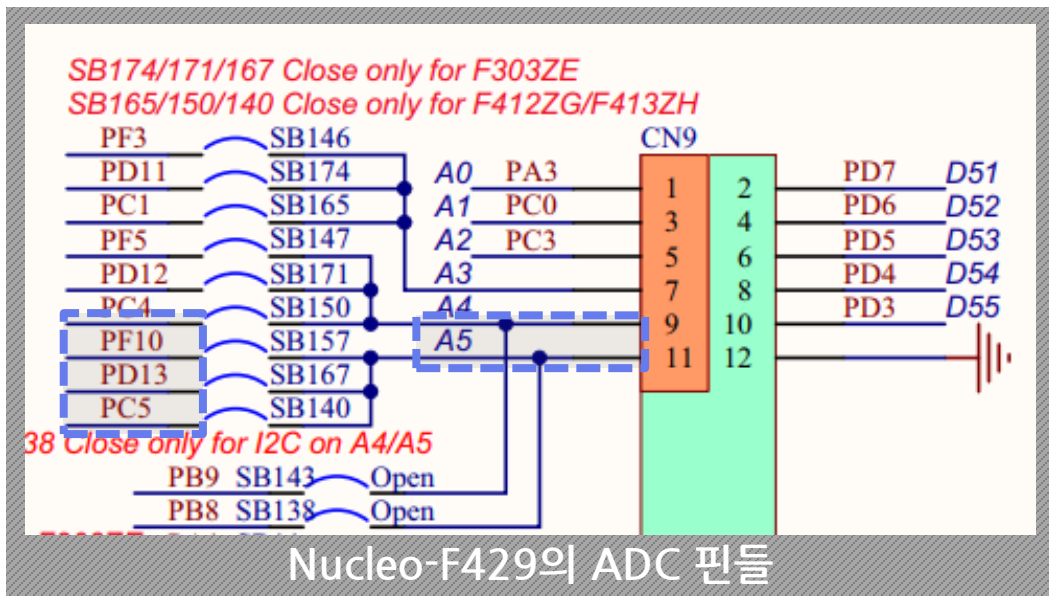


A4로 표시된 핀은
STM32F429의 PF5번 핀이나
PD12, PC4번 핀 중 선택 가능

STM32F429의 ADC

⚙️ Nucleo-F429 보드의 ADC

🌈 Nucleo-F429 보드의 ADC 핀

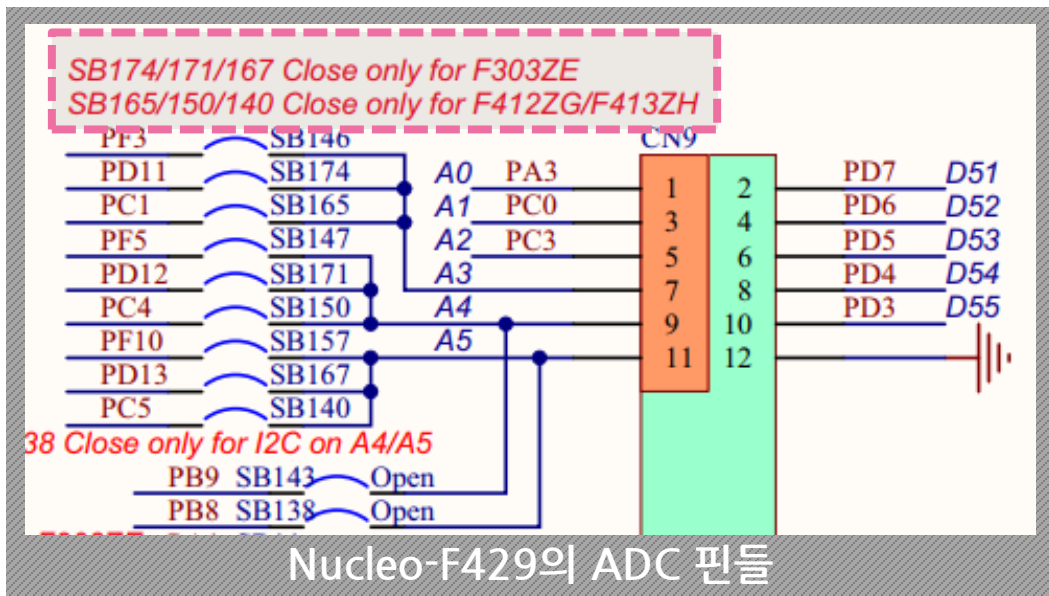


A5로 표시된 핀은
STM32F429의 PF10번 핀이나
PD13, PC5번 핀 중 선택 가능

STM32F429의 ADC

⚙️ Nucleo-F429 보드의 ADC

🌈 Nucleo-F429 보드의 ADC 핀




... 회로도의 SB는 Solder Bridge의 약자로 납으로 간단하게 연결할 수 있는 부분을 말함

... 즉, 납을 사용하여 간단하게 연결하거나 연결을 끊을 수 있음

STM32F429의 ADC 제어 SW 설계하기

 ADC 제어 실습 환경

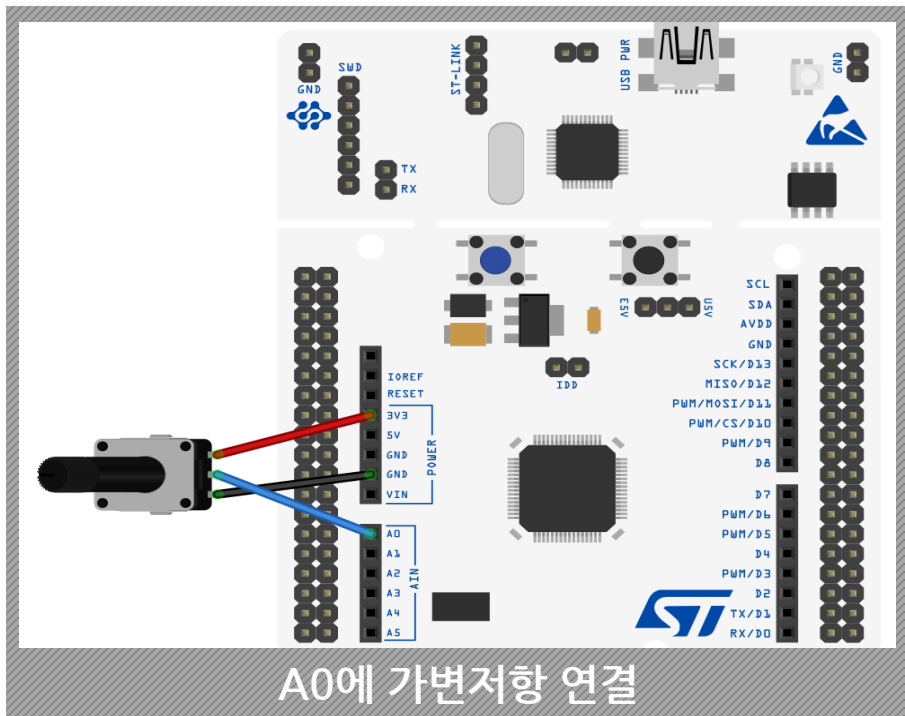
 ADC 포트에 가변저항 연결

ADC 실습을 위해 ADC 포트에 가변저항을 연결하여 실습 진행

STM32F429의 ADC 제어 SW 설계하기

⚙️ ADC 제어 실습 환경

🌈 ADC 포트에 가변저항 연결

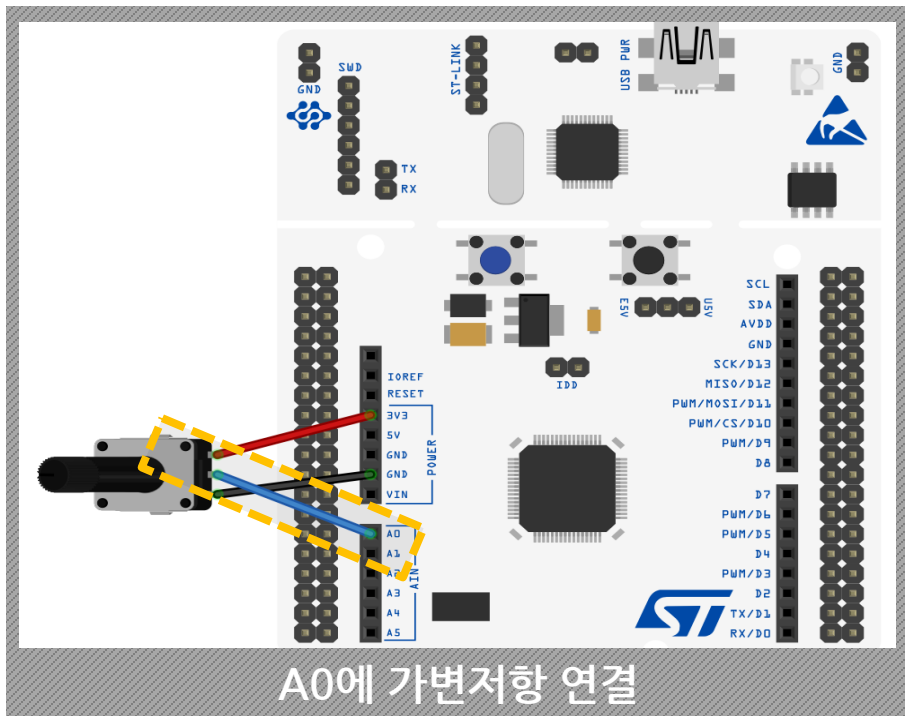


... 그림은 Nucleo-144 보드가 아닌
Nucleo-32 보드이지만
아두이노 핀 배열은 동일함

STM32F429의 ADC 제어 SW 설계하기

⚙️ ADC 제어 실습 환경

🌈 ADC 포트에 가변저항 연결

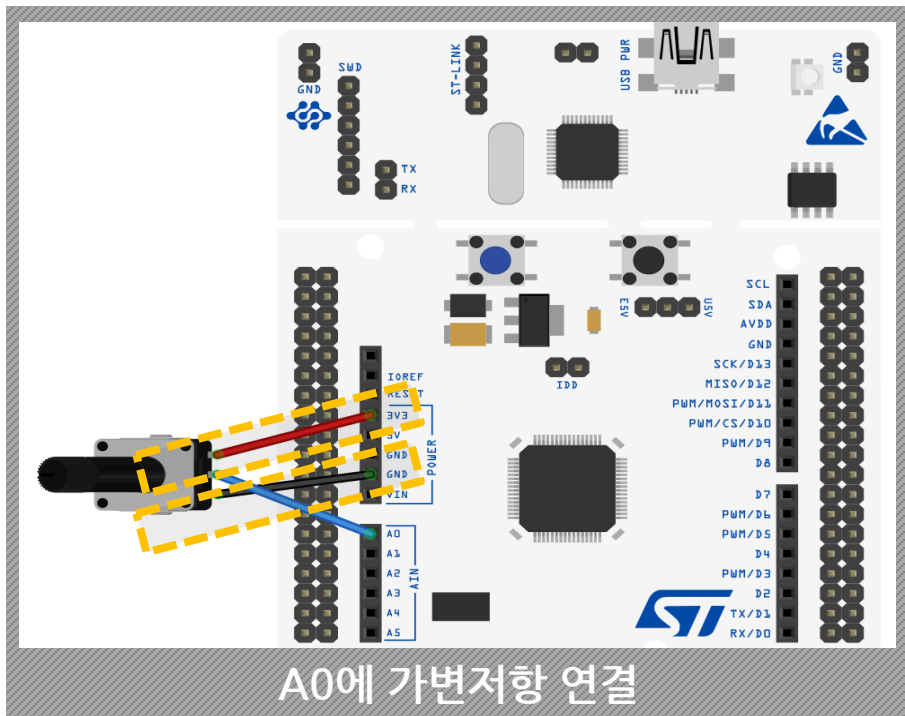


... 아두이노 핀 기준 A0핀인
Nucleo-F429 보드의 CN9커넥터의
1번 핀에 가변저항의 가운데 선과
연결

STM32F429의 ADC 제어 SW 설계하기

⚙️ ADC 제어 실습 환경

🌈 ADC 포트에 가변저항 연결



... 가변저항의 양 끝단 핀은
각각 3.3V 전원과 GND에 연결

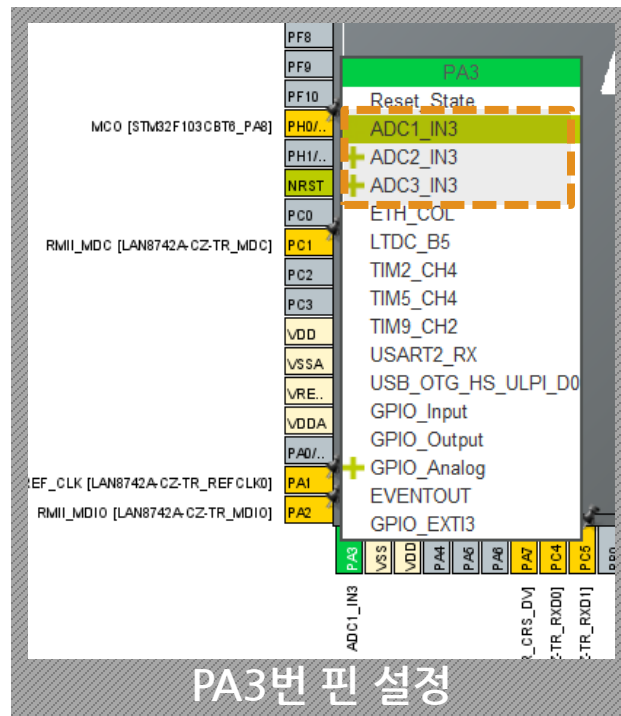
STM32F429의 ADC 제어 SW 설계하기

⚙️ ADC 제어 실습 환경

🌈 ADC 모드의 설정

PA3번 핀

ADC1_IN3, ADC2_IN3,
ADC3_IN3 중 하나를 선택할 수 있음



STM32F429의 ADC 제어 SW 설계하기

 ADC 제어 실습 환경

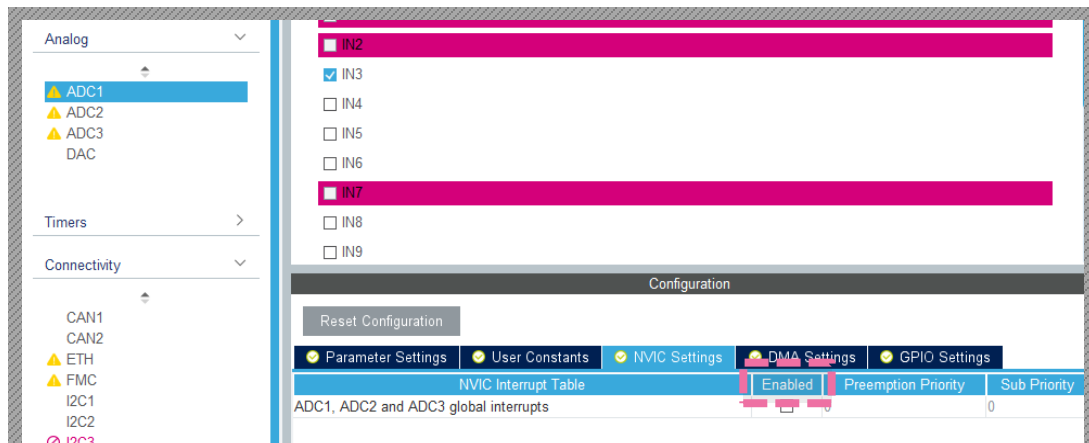
 ADC 모드의 설정

ADC는 폴링 모드나 인터럽트 모드를 선택할 수 있음

STM32F429의 ADC 제어 SW 설계하기

⚙️ ADC 제어 실습 환경

🌈 ADC 모드의 설정



폴링 모드 사용

→ ADC1 Configuration에서
NVIC Settings의
Enabled를 선택하지 않음

인터럽트 모드/ 폴링 모드 선택

인터럽트 모드 사용

→ ADC1 Configuration에서
NVIC Settings의
Enabled 선택

STM32F429의 ADC 제어 SW 설계하기

ADC 제어 SW 코딩 및 테스트

폴링 모드 동영상

- ... CubeMX를 사용하여 코드 생성
- ... Main.c의 main 함수에 ADC 폴링 모드 제어 코드 작성
 - HAL_ADC_Start
 - HAL_ADC_PollForConversion
 - HAL_ADC_GetValue
 - HAL_ADC_Stop
- ... 컴파일 후 펌웨어를 보드에 다운로드
- ... 가변저항을 조정하여 ADC 값을 메시지 출력해 봄으로써 ADC 제어 SW 검증

STM32F429의 ADC 제어 SW 설계하기

ADC 제어 SW 코딩 및 테스트

인터럽트 동영상

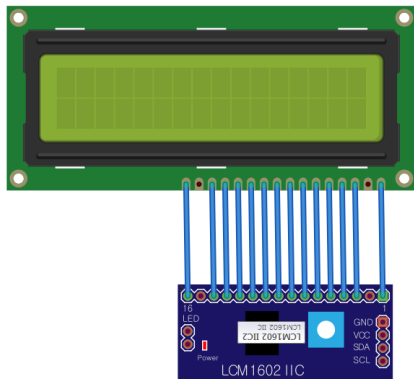
- ... CubeMX를 사용하여 코드 생성
- ... Main.c의 main 함수에 ADC 인터럽트 모드 제어 코드 작성
 - NVIC Settings의 Enabled를 선택하여 인터럽트 모드 enable
 - Src/stm32f4xx_it.c에 ADC 인터럽트 핸들러인 ADC_IRQHandler가 생성됨
 - Main.c에 ADC 인터럽트 callback 함수인 HAL_ADC_ConvCpltCallback 추가
 - Main 함수 while문에서 adc_value 값을 UART로 확인
- ... 컴파일 후 펌웨어를 보드에 다운로드
- ... 가변저항을 조정하여 ADC 값을 메시지 출력해 봄으로써 ADC 제어 SW 검증

STM32F429의 ADC 제어 SW 설계하기

⚙ I2C 버스용 LCD 모듈

🌈 아두이노 LCM1602 IIC 실드

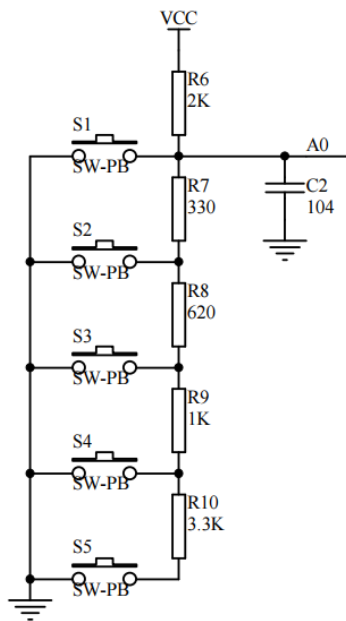
- ... 원래 16개 핀의 인터페이스 필요
- ... 하지만 I2C 인터페이스를 가지는 컨트롤러를 중간에 사용하여 VCC, GND, I2C SDA, I2C SCL의 4개의 핀만을 가지고 제어 가능
- ... <http://www.devicemart.co.kr/goods/view?no=1279486>



STM32F429의 ADC 제어 SW 설계하기

⚙️ I2C 버스용 LCD 모듈

🌈 아두이노 LCM1602 IIC 쉴드 회로도중 switch 회로

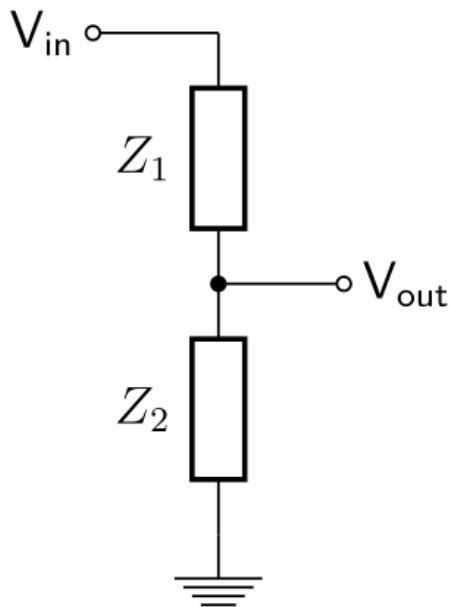


- ... Reset 버튼을 제외한 UP, DOWN, LEFT, RIGHT, SELECT의 5개의 버튼 회로도
- ... A0라는 아날로그 핀의 전압 레벨로 5개 버튼의 눌림 상태를 알 수 있음.
- ... 이 회로를 이해하려면 voltage divider를 이해해야 함.

STM32F429의 ADC 제어 SW 설계하기

⚙️ I2C 버스용 LCD 모듈

🌈 Voltage divider



... V_{out} 의 값은 아래 공식과 같이 Z_1 과 Z_2 저항의 값으로 정해진다.

$$V_{out} = \frac{Z_2}{Z_1 + Z_2} \cdot V_{in}$$

... $V_{in} = 5V$, $Z_1 = 100\Omega$, $Z_2 = 200\Omega$

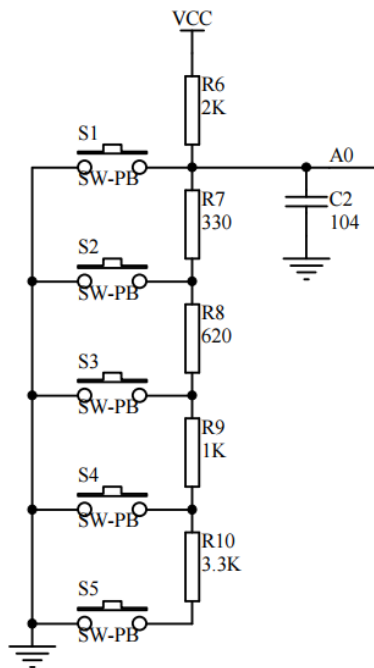
... $V_{out} = \frac{200}{100+200} * 5 = 3.33V$

... https://en.wikipedia.org/wiki/Voltage_divider

STM32F429의 ADC 제어 SW 설계하기

⚙️ I2C 버스용 LCD 모듈

🌈 아두이노 LCM1602 IIC 쉴드 회로도중 switch 회로



- ... 어떤 스위치도 누르지 않으면 R6은 pull-up저항역할이 되어 A0에 걸리는 전압은 5V
- ... S1을 누르면 접지와 연결되므로 0V
- ... S2를 누르면 R6과 R7의 voltage divider에 의해 $5V * (300 / (2000 + 300)) = 0.7V$
- ... S3을 누르면 R6과 R7+R8의 voltage divider에 의해 $5V * ((300 + 620) / (2000 + (300 + 620))) = 1.6V$
- ... S4는 $5V * (\quad / \quad + \quad) = 2.46V$
- ... S5는 $5V * (\quad / \quad + \quad) = 3.6V$

STM32 ADC+Timer

ADC + Timer 인터럽트

- ... ADC는 폴링 방식이나 인터럽트 방식을 선택하여 사용할 있음.
- ... 폴링 방식은 HAL_Delay()를 사용하여 샘플링 주기를 조절할 수 있으나 코딩이 복잡하고 비효율적이 됨.
- ... ADC 인터럽트 방식을 사용하면 샘플링 주기를 조절하기가 어려워 너무 자주 인터럽트가 발생함.
- ... 샘플링 주기를 제어하기 위해 Timer인터럽트와 ADC를 조합하여 사용할 수 있음.
- ... Timer 인터럽트의 주기대로 ADC값을 얻을 수 있음.
- ... 다음에 설명하는 예는 100ms의 주기로 ADC 샘플링

심화 학습

버튼 다루기

ADC + Timer 인터럽트

EXTI 인터럽트로 Long, Normal, Double click 구분하기



융합기술교육원 임베디드시스템



[8강 STM32 ADC+Timer 인터럽트, EXTI](#)
[심화 – YouTube](#)

8강 STM32 ADC+Timer 인터럽트, EXTI 심화

조회수 787회 • 2020. 3. 30.



8



0



공유

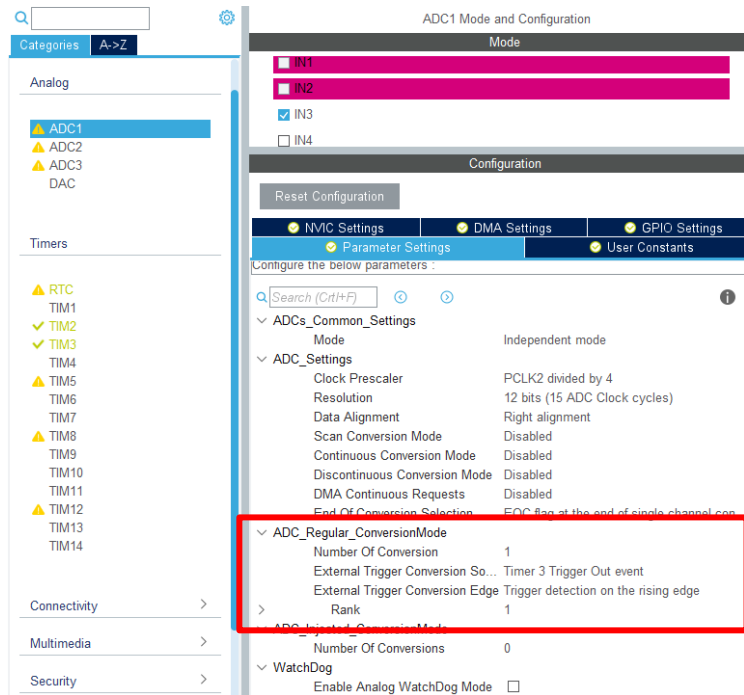


저장



STM32 ADC+Timer

ADC + Timer 인터럽트



ADC 셋팅

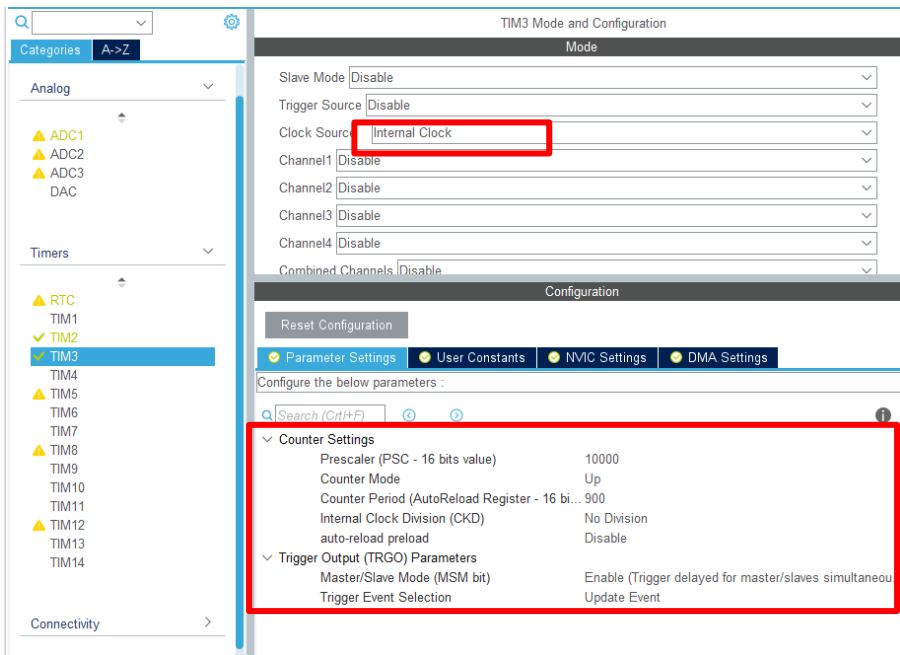
- A0인 PA3번 핀을 ADC1 컨트롤러의 IN3번 채널 사용
- Parameter Settings에서 ADC_Regular_ConversionMode에서 External Trigger Conversion Source를 Timer 3 Trigger Out event를 선택
- External Trigger Conversion Edge를 Trigger detection on the rising edge를 선택



STM32 ADC+Timer



ADC + Timer 인터럽트



Timer 셋팅

- TIM3를 사용
- Clock Source를 Internal Clock사용
- 100ms 주기를 위해 Prescaler 를 10000, Period를 900으로 셋팅
- $0.1\text{초} = 10000 * (1/90\text{M}) * 900$
- Trigger Output Parameters의 Master/Slave Mode는 Enable
- Trigger Event Selection은 Update Event
- NVIC의 TIM3 global interrupt enable



심화 학습



STM32 ADC+Timer



ADC + Timer 인터럽트

```
#define UP_KEY_MIN 0
#define UP_KEY_MAX 10

#define DOWN_KEY_MIN 850
#define DOWN_KEY_MAX 870

#define LEFT_KEY_MIN 1940
#define LEFT_KEY_MAX 1960

#define RIGHT_KEY_MIN 2980
#define RIGHT_KEY_MAX 3010
```



전압 레벨을 통한 ADC 변환 값의 범위

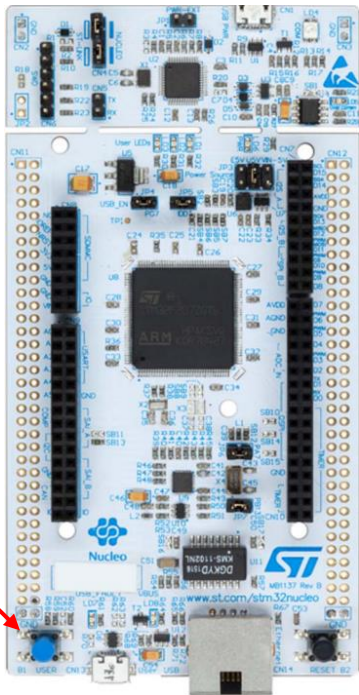
```
uint32_t ADC_value;

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if(htim->Instance==TIM3)
    {
        ADC_value = HAL_ADC_GetValue(&hadc1);
        if(ADC_value<=UP_KEY_MAX)
        {
            printf("UPWrWn");
        }
        if(ADC_value>= DOWN_KEY_MIN && ADC_value<=DOWN_KEY_MAX)
        {
            printf("DOWNWrWn");
        }
        if(ADC_value>= LEFT_KEY_MIN && ADC_value<=LEFT_KEY_MAX)
        {
            printf("LEFTWrWn");
        }
        if(ADC_value>= RIGHT_KEY_MIN && ADC_value<=RIGHT_KEY_MAX)
        {
            printf("RIGHTWrWn");
        }
    }
}
```

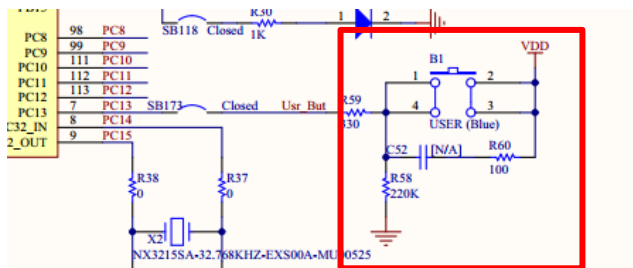
버튼 다루기

⚙ User button을 Select키로 사용

🌈 Select키는 Long click, Double click, Normal click 구분



Select키로 사용

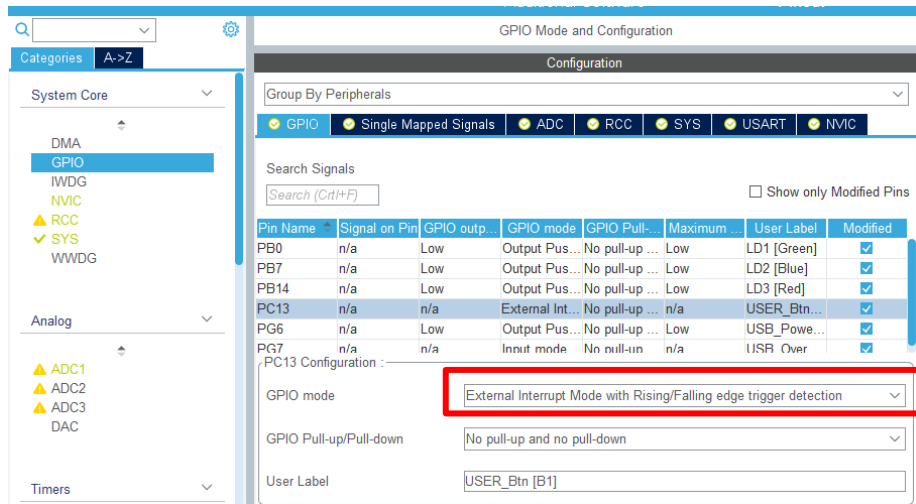


- ... User 버튼의 회로도
- ... 평상시에는 220K pull-down저항에 의해 low상태
- ... 버튼을 누르면 VDD와 연결되어 high상태

버튼 다루기

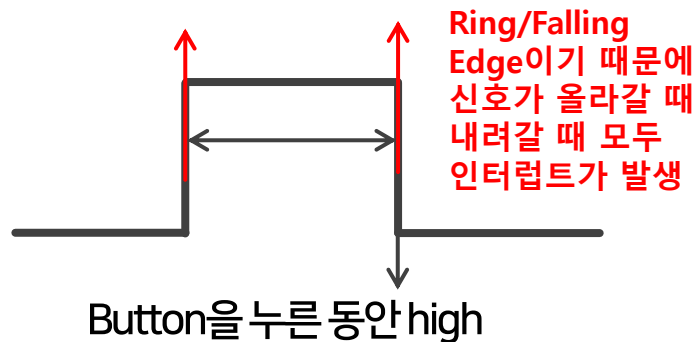
⚙ User button을 Select키로 사용

🌈 Select키는 Long click, Double click, Normal click 구분



... GPIO Configuration

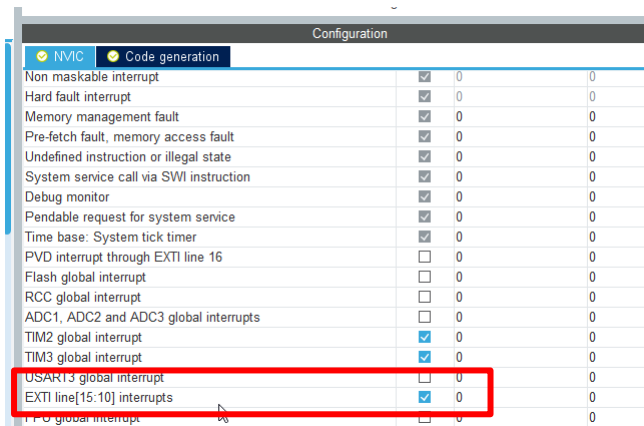
- PC13번을 External Interrupt모드로 설정
- Rising/Falling edge trigger로 설정



버튼 다루기

⚙ User button을 Select키로 사용

🌈 Select키는 Long click, Double click, Normal click 구분



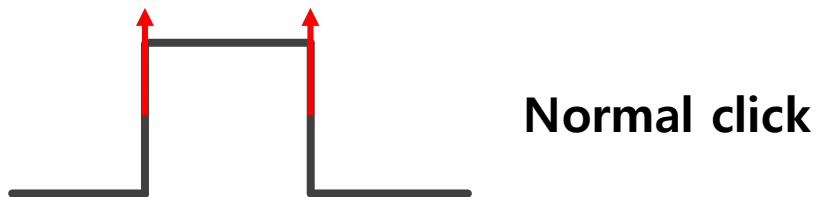
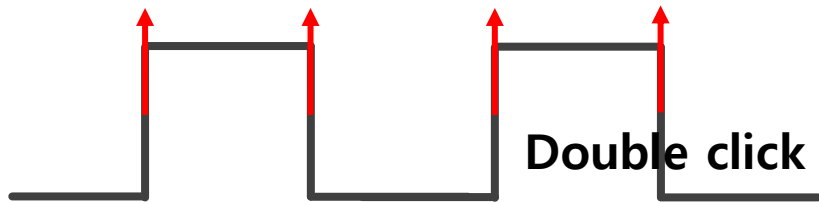
...> NVIC Configuration

➤ EXTI line[15:10] interrupts enable

버튼 다루기

⚙ User button을 Select키로 사용

🌈 Select키는 Long click, Double click, Normal click 구분



- ... Long click은 high인 구간의 길이로 구분
- ... Double click은 high, low, high인 구간의 길이로 구분
- ... Normal click은 high인 구간의 길이로 구분
- ... 그리고 테스트를 해보면 Noise에 해당하는 아주 짧은 high 구간도 있음. 이를 걸러 주어야 함.
- ... 빨간색 화살표 순간이 모두 EXTI 인터럽트 발생하는 시점으로 지난 번 발생한 시간에서 현재 발생한 시간의 차를 계산하면 펄스의 폭을 계산할 수 있음.
- ... 인터럽트 시점이 Rising인지 falling인지는 HAL_GPIO_ReadPin()으로 알 수 있음.
- ... Rising때는 high레벨, Falling일 때는 low 레벨

버튼 다루기



User button을 Select키로 사용



Select키는 Long click,
Double click, Normal
click 구분

```
typedef struct{  
    int32_t time;      펄스의 길이와 핀의  
    GPIO_PinState level; 레벨  
}ClickInfoDef;
```

ClickInfoDef click[3]; 3개의 click정보

```
#define LONG_CLICK_MIN 1500 //1.5sec  
#define LONG_CLICK_MAX 5000 //5sec
```

```
#define DOUBLE_CLICK_MIN 40  
#define DOUBLE_CLICK_MAX 120
```

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)  
{  
    GPIO_PinState pin;  
    int i;  
  
    if(GPIO_Pin == GPIO_PIN_13)  
    {  
        current_time = HAL_GetTick();  
        time_interval = current_time - last_time;  
        last_time = current_time;  
  
        pin = HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_13);  
  
        if(time_interval<=2) // noise  
            printf("Noise %d,%dWrWn",pin,time_interval);  
        else  
        {  
  
            click[2].time = click[1].time;  
            click[2].level = click[1].level;  
  
            click[1].time = click[0].time;  
            click[1].level = click[0].level;  
  
            click[0].time = time_interval;  
            click[0].level = pin;
```

펄스를 3개까지 저장

버튼 다루기

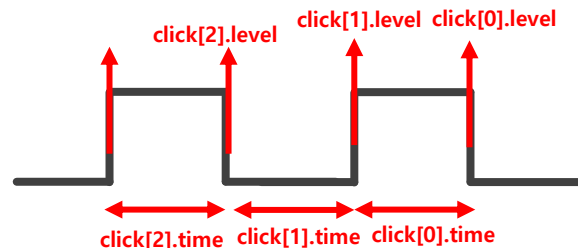
⚙ User button을 Select키로 사용

```
if( click[2].level ==GPIO_PIN_RESET && click[1].level == GPIO_PIN_SET && click[0].level ==GPIO_PIN_RESET)
{
    for(i=0;i<3;i++)
    {
        if(click[i].time>= DOUBLE_CLICK_MIN && click[i].time <= DOUBLE_CLICK_MAX)
        {
            continue;
        }
        else
        {
            break;
        }
    }
    if(i==3)
    {
        printf("Double Click\r\n");
    }
}
if(click[0].level == GPIO_PIN_RESET && click[0].time >=LONG_CLICK_MIN) // long click
{
    printf("Long Key\r\n");
}
else if(click[0].level == GPIO_PIN_RESET && click[0].time < LONG_CLICK_MIN && click[0].time > DOUBLE_CLICK_MAX)
{
    printf("Select Key, %d\r\n",click[0].time);
}
```

현재 펄스는 low, 지난번 펄스는 high, 그전 pulse는 low이면

Double click에 해당하는 pulse 폭을 유지하면

3개의 pulse 모두 위의 조건을 만족하면



Button을 누르다가 떼는 순간의 레벨이므로 Low

Q & A
Thank you

