피지컬 컴퓨팅

Lec. 5. FND / interrupt

Heenam Yoon

Department of
Human-Centered Artificial Intelligence

E-mail) h-yoon@smu.ac.kr
Room) 0112

공지사항

- 날짜: 2023년 10월 23일 (월)
- 시간: 12:00 ~ 14:00
- 장소: G312
- 범위: 오늘 배운 내용까지

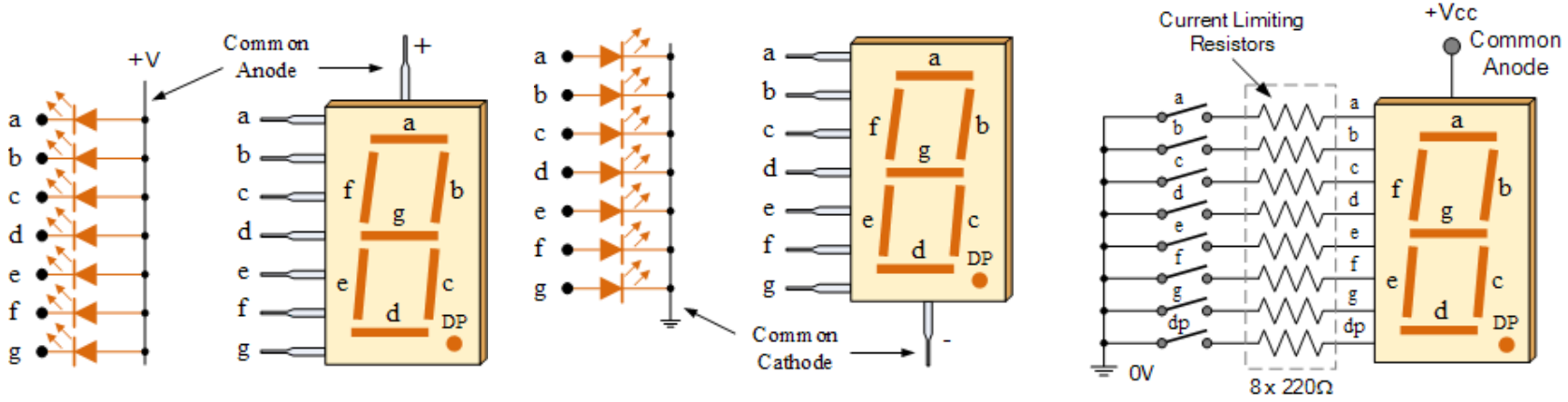
Contents

- GPIO: FND & Switch 실습
- Polling & Interrupt 이론 & 실습

7-세그먼트 (FND)

FND: Flexible Numeric Display

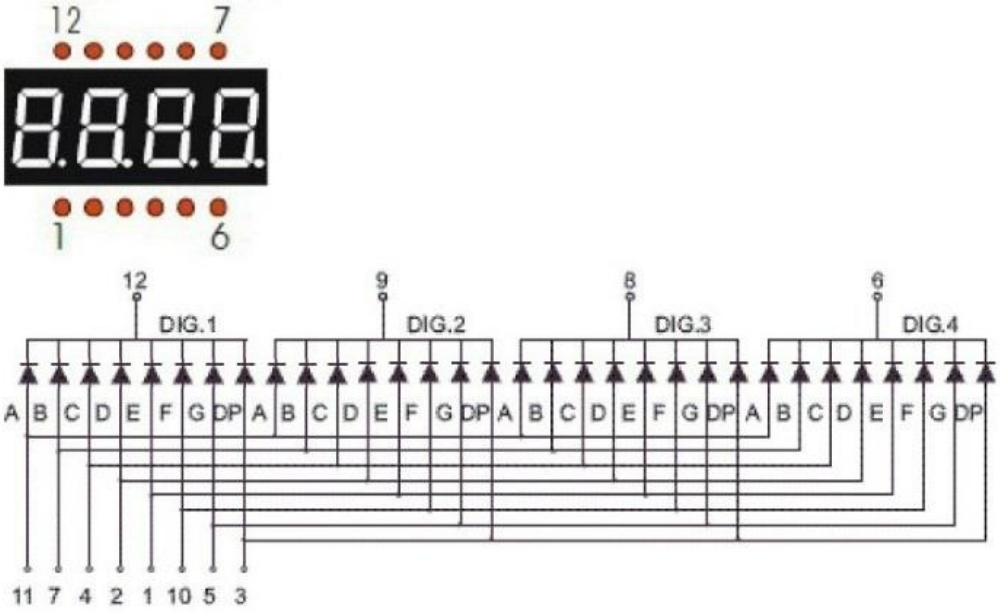
- LED 8개를 그림과 같이 배열해 놓은 것으로서 숫자나 간단한 기호 표현에 많이 사용
- 7-세그먼트는 공통(Common)단자에 인가되는 전원에 따라서 Common Anode(+공통)과 Common Cathode(-공통)으로 분류
- 그림과 같이 공통 단자에 VCC(+5V)에 연결하고 입력 단자에 0V를 인가하였을 때 해당하는 LED에 램프가 들어오는 Anode(+) 공통 형과 공통 단자에 접지(0V)를 연결하고 입력 단자에 +5V를 인가하였을 때, 해당하는 LED에 램프가 들어오는 Cathode(-) 공통형

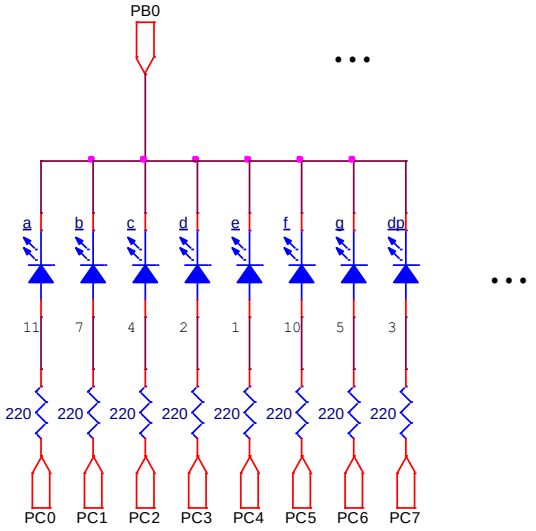


7-세그먼트 (FND)

실습 1: FND의 첫번째 세그먼트 출력하기. HW

- FND의 A,B,C,D,E,F,G,DP를 저항(220Ω)을 연결한 후 PC0~PC7과 연결
- FND의 common인 12,9,8,6핀을 PB0~PB3과 연결





7-세그먼트 (FND)

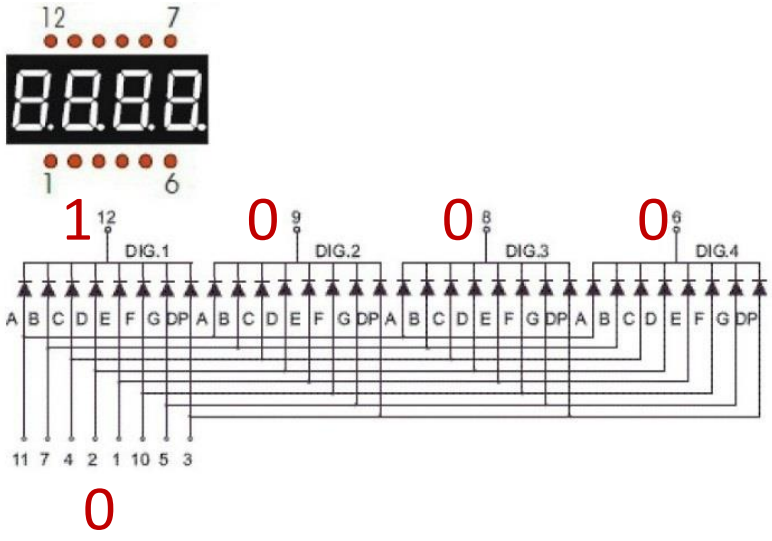
실습 1: FND의 첫번째 세그먼트 출력하기. SW

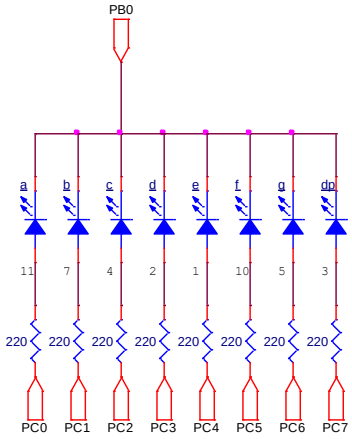
```
#include "stm32f4xx.h"
void Delay(__IO uint32_t nCount)
{
  for(; nCount != 0; nCount--);
}
void FND_Init(void)
{
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);
  GPIO_InitTypeDef GPIO_InitStructure;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2 | GPIO_Pin_3 | GPIO_Pin_4
| GPIO_Pin_5 | GPIO_Pin_6 | GPIO_Pin_7;
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
  GPIO_Init(GPIOC, &GPIO_InitStructure);

  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE);
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2 | GPIO_Pin_3;
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
  GPIO_Init(GPIOB, &GPIO_InitStructure);
}
```

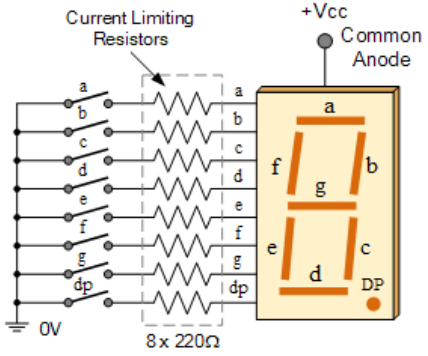
7-세그먼트 (FND)

실습 1: FND의 첫번째 세그먼트 출력하기. SW

```
int main()
{
  int i = 0;
  FND_Init();

  while(1)
  {
    GPIO_SetBits(GPIOB, GPIO_Pin_0);
    GPIO_ResetBits(GPIOB, GPIO_Pin_1);
    GPIO_ResetBits(GPIOB, GPIO_Pin_2);
    GPIO_ResetBits(GPIOB, GPIO_Pin_3);
    GPIO_Write(GPIOC, 0x0000);
  }
  return 0;
}
```



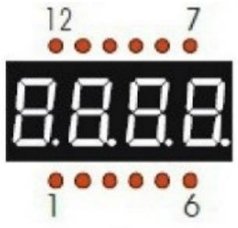


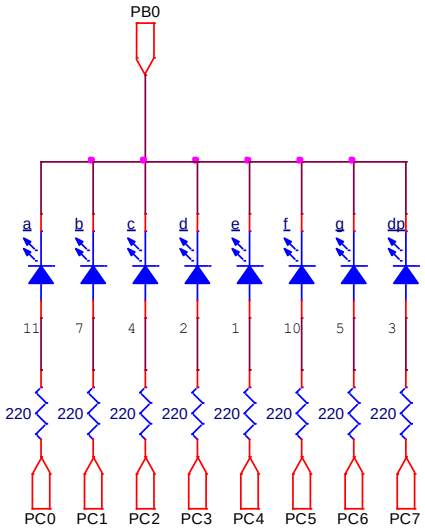


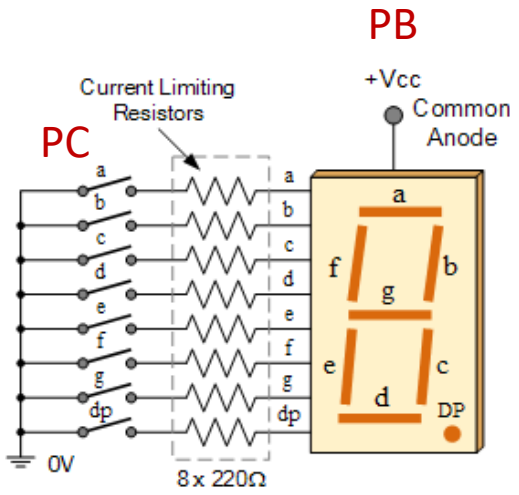
7-세그먼트 (FND)

실습 2: FND의 첫번째 세그먼트에 0에서 9까지 출력하기

- 0을 만들고 싶다면, a, b, c, d, e, f에 불이 들어와야 함
- FND의 11, 7, 4, 2, 1, 10을 조정해야 함
- PB0를 1로 해주고 (=5v공급)
- PC0, PC1 , PC2 , PC3 , PC4 , PC5 를 0으로 해야 함 (=0V)
- PC6, PC7은 1로 하면 g와 dp는 불이 안들어 올 것임
- 1100 0000 = 0xC0







7-세그먼트 (FND)

실습 2: FND의 첫번째 세그먼트에 0에서 9까지 출력하기. SW

```
unsigned char Font[10] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xD8, 0x80, 0x90};

int main()
{
  int i = 0;
  FND_Init();
  while(1)
  {
    GPIO_SetBits(GPIOB, GPIO_Pin_0);
    GPIO_ResetBits(GPIOB, GPIO_Pin_1);
    GPIO_ResetBits(GPIOB, GPIO_Pin_2);
    GPIO_ResetBits(GPIOB, GPIO_Pin_3);

    for(i=0; i<10; i++)
    {
      GPIO_Write(GPIOC, Font[i]);
      Delay(2500000);
    }
  }
   return 0;
}
```

7-세그먼트 (FND)

실습 3: FND에 '1234' 출력하기

```
unsigned char Font[10] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xD8, 0x80, 0x90};
int main()
{
  int i = 0;
  FND_Init();
  while(1)
  {
    GPIO_Write(GPIOB, 0x0001);
    GPIO_Write(GPIOC, Font[1]);

    GPIO_Write(GPIOB, 0x0002);
    GPIO_Write(GPIOC, Font[2]);

    GPIO_Write(GPIOB, 0x0004);
    GPIO_Write(GPIOC, Font[3]);

    GPIO_Write(GPIOB, 0x0008);
    GPIO_Write(GPIOC, Font[4]);
  }
  return 0;
}
```

무엇이 문제일까?

7-세그먼트 (FND)

실습 4: FND에 0000에서 9999까지 출력하기

- 다음 페이지의 예시 코드를 보지 말고 먼저 스스로 해보세요

- 4번째 자리가 9까지 카운트 되면, 3번째 자리가 1로 바뀌어야 함
- 3번째 자리가 9까지 카운트 되면, 2번째 자리가 1로 바뀌어야 함
- 2번째 자리가 9까지 카운트 되면, 1번째 자리가 1로 바뀌어야 함
- 0000에서 9999까지 증가 후 다시 0000으로 돌아와야 함

- Firmware 관점에서 생각하지 말고, software관점으로 생각해보기
  (C언어에서 이런 유사한 문제를 다루어 봤을 것임)

7-세그먼트 (FND)

실습 4: FND에 0000에서 9999까지 출력하기. 예시 코드

```
unsigned char Font[10] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xD8, 0x80, 0x90};

void Count_Progress(int d_3, int d_2, int d_1, int d_0)
{
  GPIO_Write(GPIOB, 0x0008);
  GPIO_Write(GPIOC, Font[d_0]);
  Delay(10000);

  GPIO_Write(GPIOB, 0x0004);
  GPIO_Write(GPIOC, Font[d_1]);
  Delay(10000);

  GPIO_Write(GPIOB, 0x0002);
  GPIO_Write(GPIOC, Font[d_2]);
  Delay(10000);

  GPIO_Write(GPIOB, 0x0001);
  GPIO_Write(GPIOC, Font[d_3]);
  Delay(10000);
}
```

```
int main()
{
  int i = 0;
  FND_Init();

  while(1)
  {
    for(i = 0; i < 10000; i++)
    {
      Count_Progress(i/1000, ((i/100)%10), ((i/10)%10), (i%10));
    }
  }
  return 0;
}
```

7-세그먼트 (FND)

실습 5: User Switch를 누른 횟수로 FND 증가시키기

- Delay()
- FND_Init()
- SW_Init(): 지난 시간에 배운 user switch 초기화 함수
- Count_Progress()

- Main()
  - OOO_Init() 함수들 호출
  - 스위치 감지 코드
  - 스위치를 누르면, Count_Progress() 호출

7-세그먼트 (FND)

실습 5: User Switch를 누른 횟수로 FND 증가시키기. 예시 코드

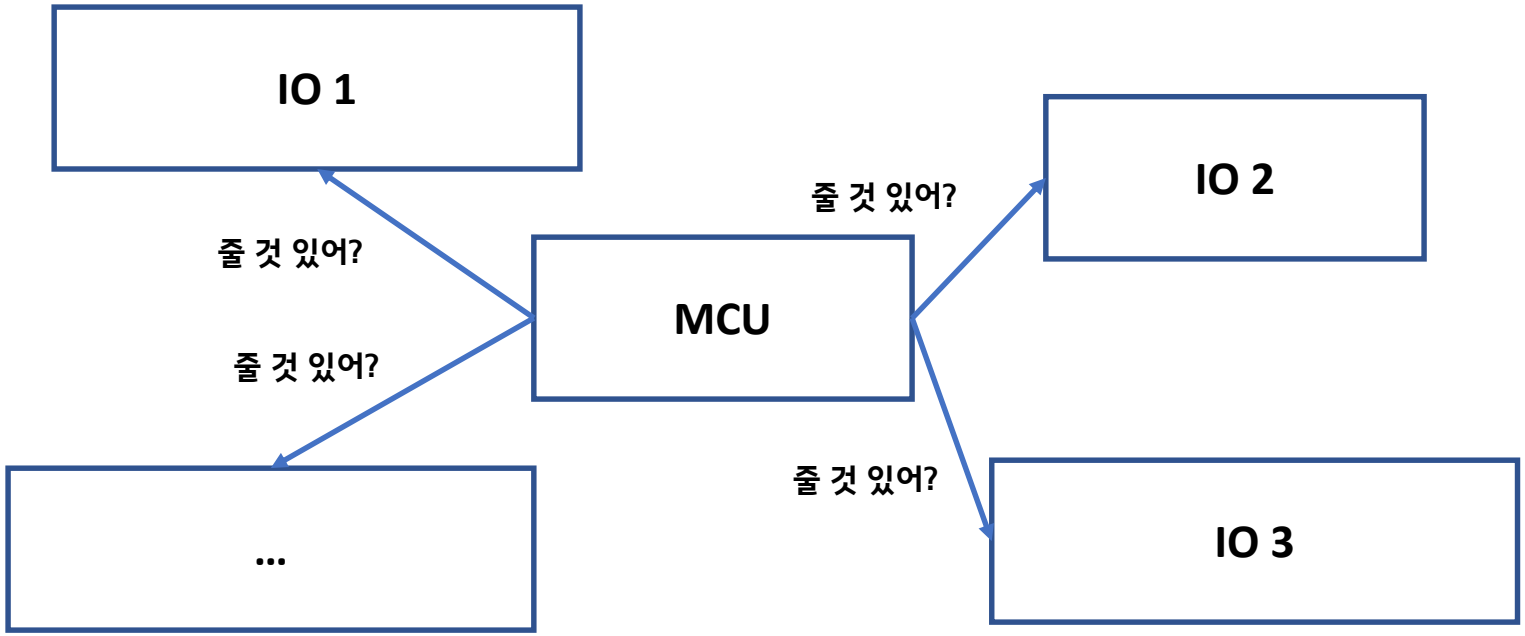
```
…
void Switch_Init(void)
{
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);
  GPIO_InitTypeDef GPIO_InitStructure;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_13;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;  GPIO_Init(GPIOC, &GPIO_InitStructure);
}
…
```

```
int main()
{
  int i = 0;
  int sw_new = 1;
  int sw_old = 1;
  FND_Init();
  Switch_Init();
  Count_Progress(0, 0, 0, 0);
```

```
while(1)
  {
    sw_new = GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_13);
   if ( (sw_old == 0) & (sw_new == 1) )
    {
      i++;
      if (i >= 10000)
      {
        i = 0;
      }
    }
    sw_old = sw_new;
    Count_Progress(i/1000, ((i/100)%10), ((i/10)%10), (i%10));  }

return 0;
}
```

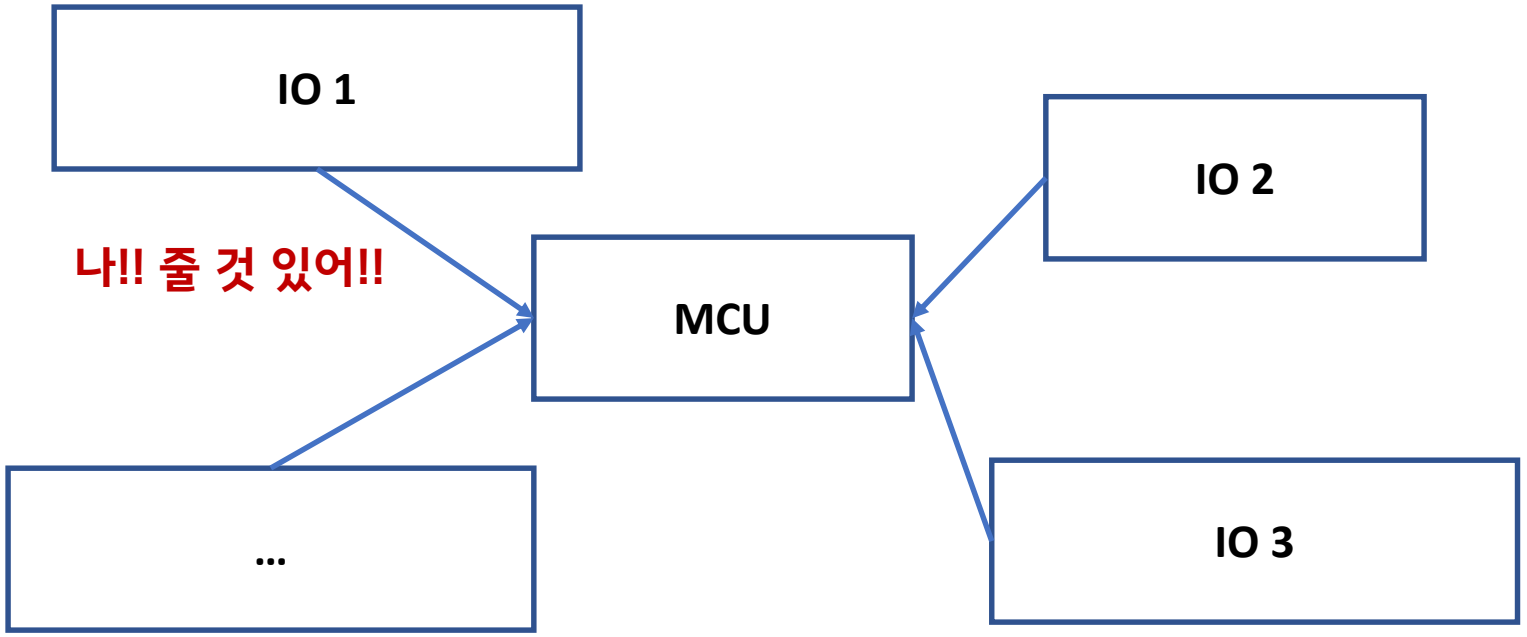
Polling & Interrupt

- MCU는 다양한 IO와 상호작용을 함
- 어떻게 상호작용을 할 것인가에 대해 고민해야 함



Polling & Interrupt

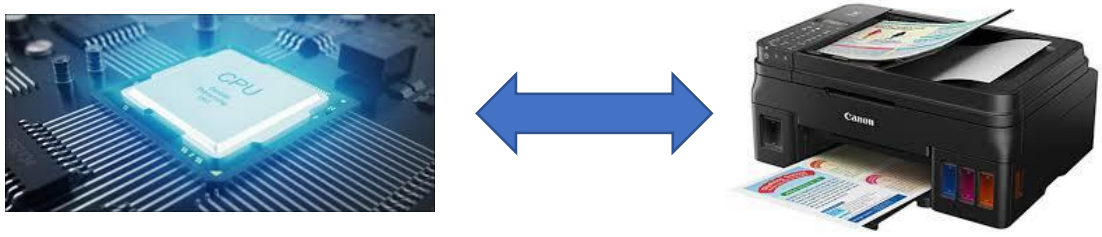
- MCU는 다양한 IO와 상호작용을 함
- 어떻게 상호작용을 할 것인가에 대해 고민해야 함



Polling & Interrupt

Interrupt의 필요성

- 고속 CPU와 저속 주변장치



CPU가 1개 문자를 프린터로
전송하는 시간

프린터가 1개 문자를 출력하는 시간

CPU는 프린터에 문자 하나가 모두 출력될 때까지 지켜봐야 하는가?

CPU는 프린터에 문자 하나를 전달하고,
다른 할 일을 하다가 문자 출력이 완료되면 프린터가 CPU에게 완료되었다고 알려주는 것이 좋지 않을까?

Polling & Interrupt

Interrupt의 필요성

- 화제 경보 알람 예.

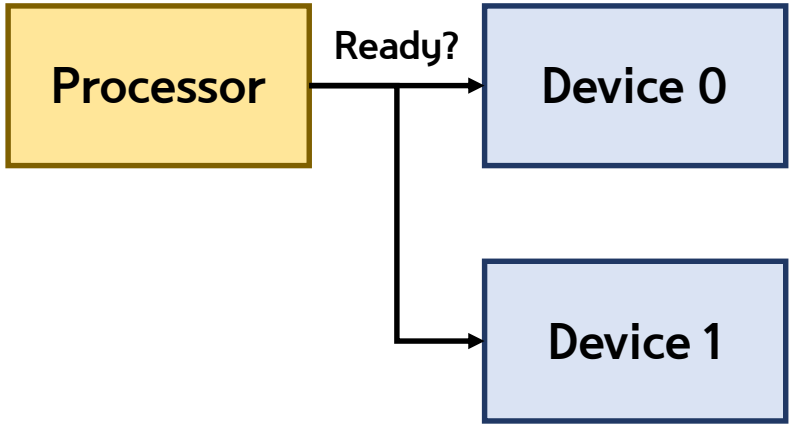
```
while(1)
{
   1번째 방 상태 확인 // 10초 소요
    …
   100번째 방 상태 확인
   …
   복도 사람 감지 후 사람이 없으면 조명 끄기 // 100초 소요
   …
   문이 잠겼는지 확인 // 1000초 소요
}
```

50번째 방에 불이 났다면, 언제 알람을 받을 수 있을까?

Polling & Interrupt

Polling

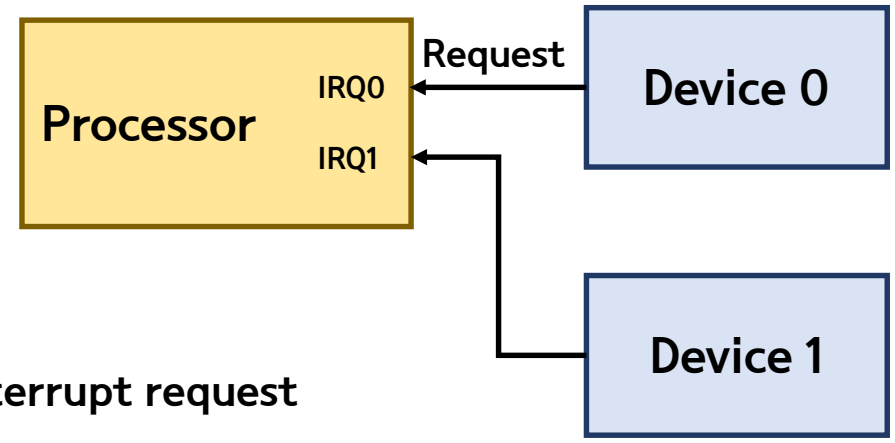
- refers to actively sampling the status of an external device as a synchro-nous activity
- refers to the situation where a device is repeatedly checked for readiness
- Simple, but inefficient



Polling & Interrupt

Interrupt

- An synchronous signal from hardware indicating the need for attention
- A way to avoid wasting the processor's valuable time in polling loops, waiting for external events
- A hardware interrupt causes the processor to save its state of execution via a context switch, and begin execution of an interrupt handler
- Efficient, but hardware is required
- The program is not sequentially executed

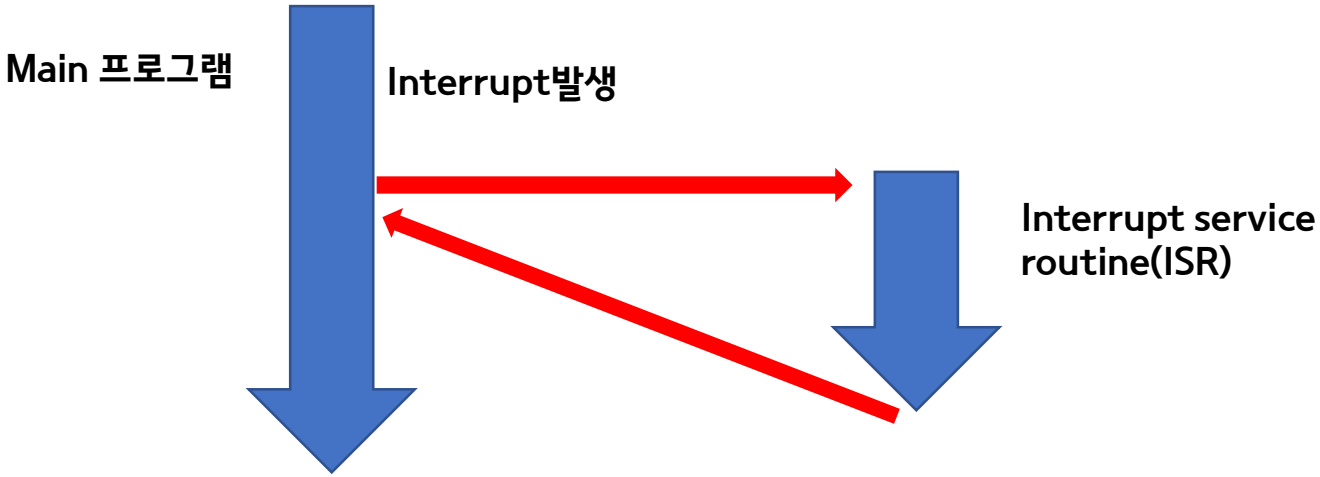


IRQ: Interrupt request

Polling & Interrupt

Interrupt

- 정의
  - 현재 수행중인 프로그램보다 우선순위가 높은 프로그램을 Interrupt에 의해 우선적으로 실행 시켜 주는 기능
- 처리방법
  - Interrupt handler or Interrupt service routine(ISR)



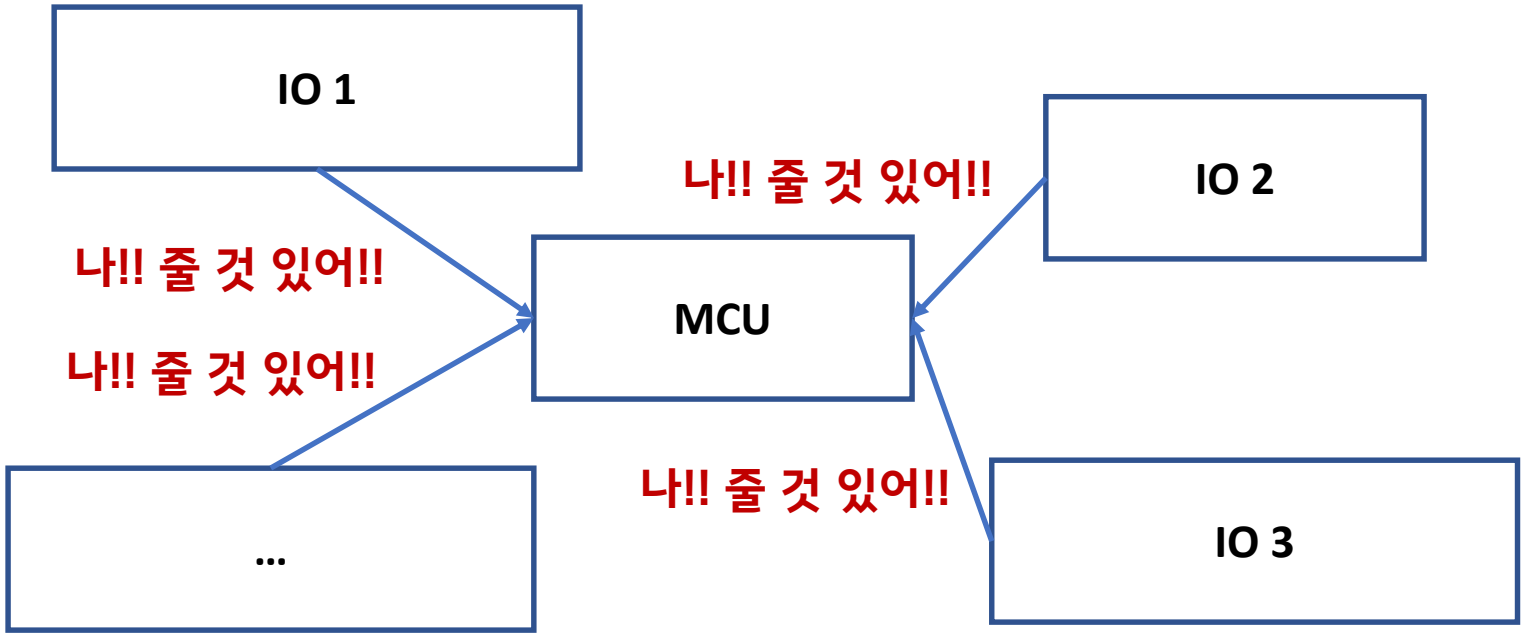
Polling & Interrupt

Interrupt 종류

- Interrupt 발생 원인에 따른 분류
  - 하드웨어 인터럽트 : 내부 인터럽트, 외부 인터럽트
  - 소프트웨어 인터럽트
- Interrupt 발생 시 마이크로프로세서의 반응 방식에 따른 분류
  - 차단 가능 인터럽트(INT : Maskable Interrupt)
  - 차단 불가능 인터럽트(NMI : Non-Maskable Interrupt)
- Interrupt 를 요구한 입출력 기기를 확인하는 방법에 따른 분류
  - 백터형 인터럽트(Vectored Interrupt)
  - 조사형 인터럽트(Polled Interrupt)

Polling & Interrupt

- 어떤 request부터 처리해야 하는가?



Polling & Interrupt

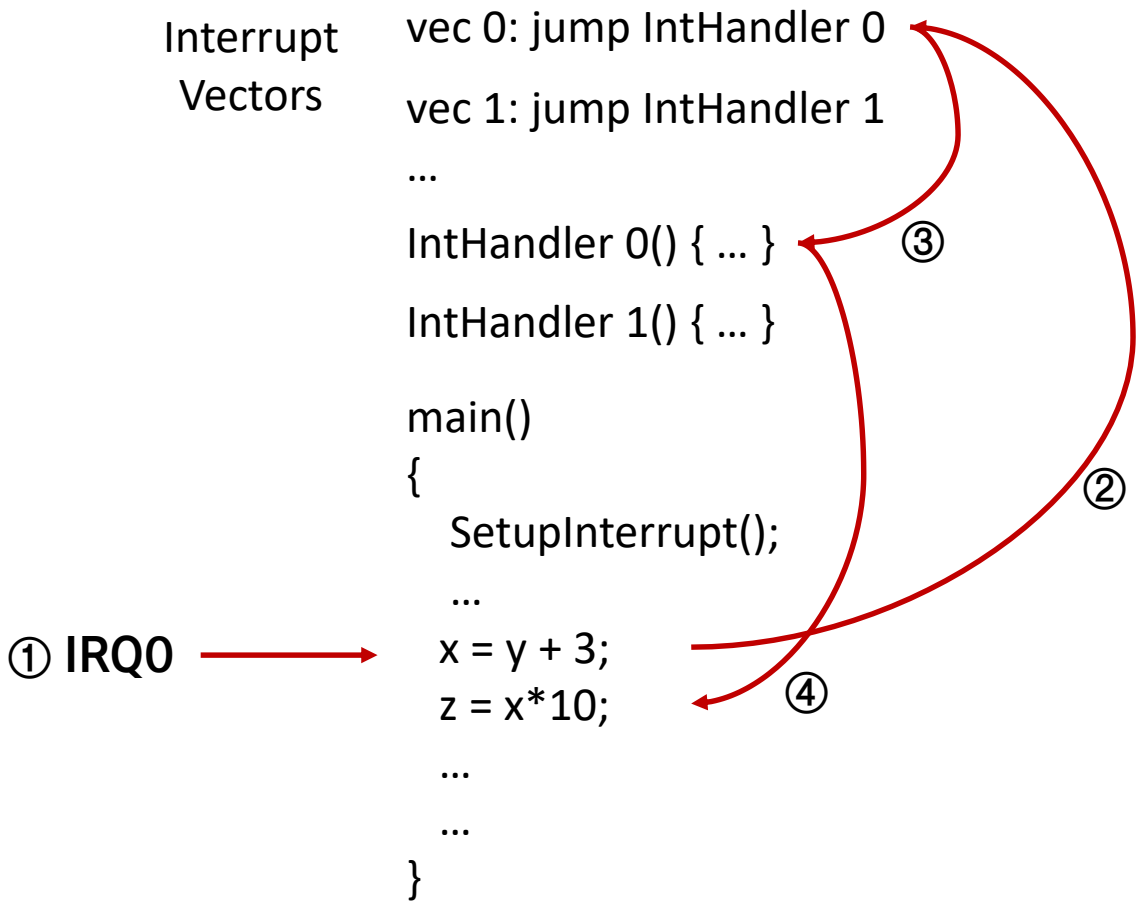
Interrupt

- Vector table

Position	Priority	Type of priority	Acronym	Description	Address
	-	-	-	Reserved	0x0000 0000
	-3	fixed	Reset	Reset	0x0000 0004
	-2	fixed	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000 0008
	-1	fixed	HardFault	All class of fault	0x0000 000C
	0	settable	MemManage	Memory management	0x0000 0010
	1	settable	BusFault	Pre-fetch fault, memory access fault	0x0000 0014
	2	settable	UsageFault	Undefined instruction or illegal state	0x0000 0018
	-	-	-	Reserved	0x0000 001C - 0x0000 002B
	3	settable	SVCall	System service call via SWI instruction	0x0000 002C
	4	settable	Debug Monitor	Debug Monitor	0x0000 0030
	-	-	-	Reserved	0x0000 0034
	5	settable	PendSV	Pendable request for system service	0x0000 0038
	6	settable	SysTick	System tick timer	0x0000 003C
0	7	settable	WWDG	Window Watchdog interrupt	0x0000 0040
1	8	settable	PVD	PVD through EXTI line detection interrupt	0x0000 0044
2	9	settable	TAMP_STAMP	Tamper and TimeStamp interrupts through the EXTI line	0x0000 0048
3	10	settable	RTC_WKUP	RTC Wakeup interrupt through the EXTI line	0x0000 004C
4	11	settable	FLASH	Flash global interrupt	0x0000 0050
5	12	settable	RCC	RCC global interrupt	0x0000 0054
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000 0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000 005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000 0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000 0064
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000 0068
11	18	settable	DMA1_Stream0	DMA1 Stream0 global interrupt	0x0000 006C

Polling & Interrupt

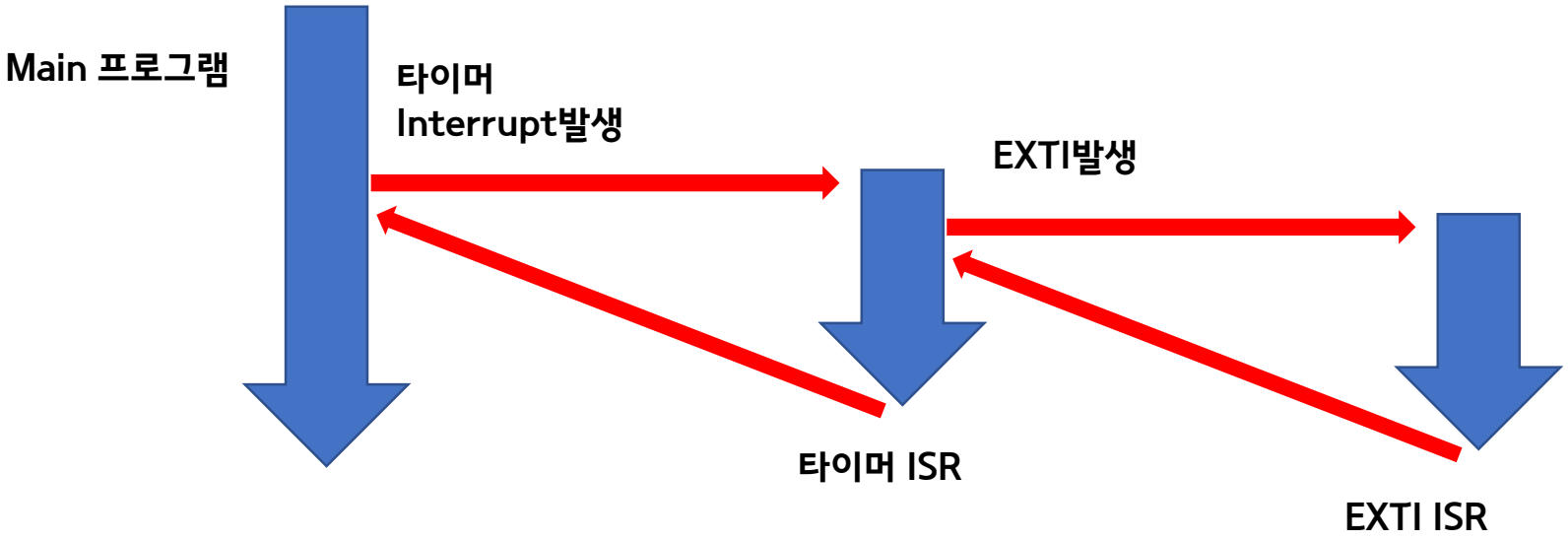
Interrupt



Polling & Interrupt

Interrupt 우선 순위

- 다중 interrupt 발생시 우선순위가 높은 interrupt를 우선 처리
- Nested Vectored Interrupt Controller (NVIC)



Polling & Interrupt

Interrupt 우선 순위

- 다중 interrupt 발생시 우선순위가 높은 interrupt를 우선 처리
- Nested Vectored Interrupt Controller (NVIC)

NVIC_PriorityGroup	NVIC_IRQChannelPreemptionPriority	NVIC_IRQChannelSubPriority	Description
NVIC_PriorityGroup_0	0	0-15	0 bits for pre-emption priority 4 bits for subpriority
NVIC_PriorityGroup_1	0-1	0-7	1 bits for pre-emption priority 3 bits for subpriority
NVIC_PriorityGroup_2	0-3	0-3	2 bits for pre-emption priority 2 bits for subpriority
NVIC_PriorityGroup_3	0-7	0-1	3 bits for pre-emption priority 1 bits for subpriority
NVIC_PriorityGroup_4	0-15	0	4 bits for pre-emption priority 0 bits for subpriority

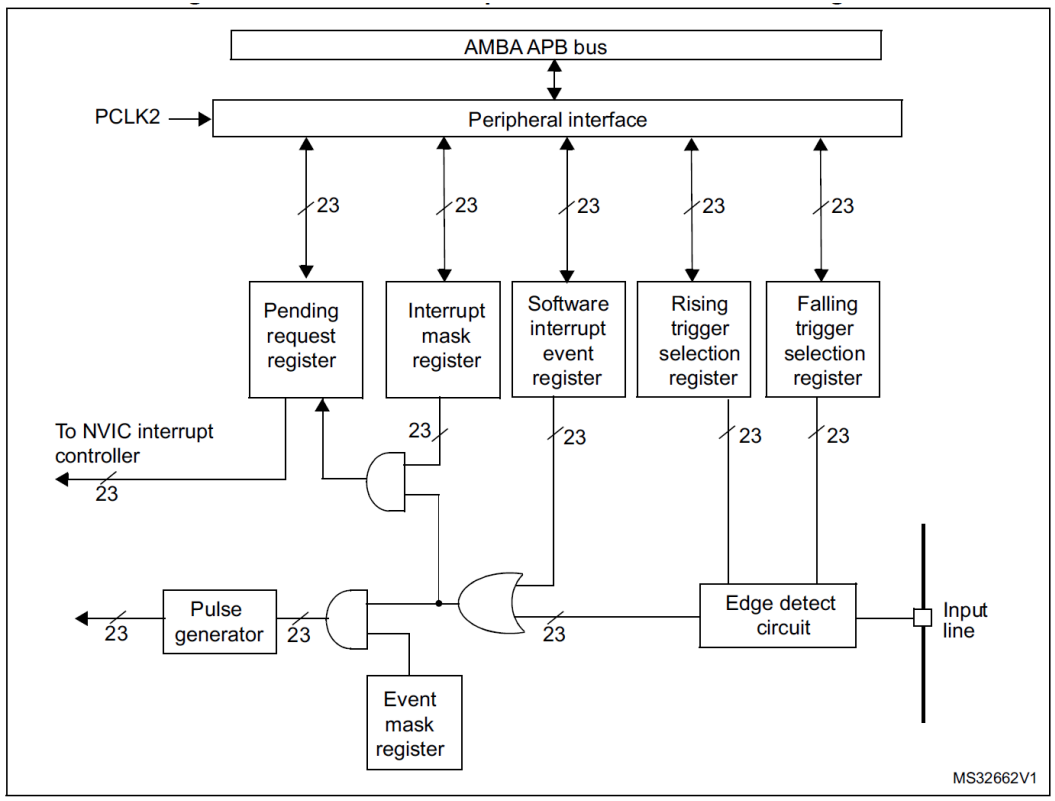
Polling & Interrupt

STM32 외부 인터럽트

- STM32F4의 외부 인터럽트 / 이벤트 컨트롤러를 생성하기 위하여 최대 23 개의 엣지 검출기로 구성
  - Falling edge(high to low)
  - Rising edge(low to high)
- 각각의 입력 라인이 독립적으로 선택 할 수 있도록 구성
- 외부 인터럽트 주요 기능
  - 각 인터럽트 라인을 위한 전용 상태 비트
  - 독립적인 트리거 표기
  - 최대 23개의 소프트웨어 이벤트/인터럽트 요청의 생성

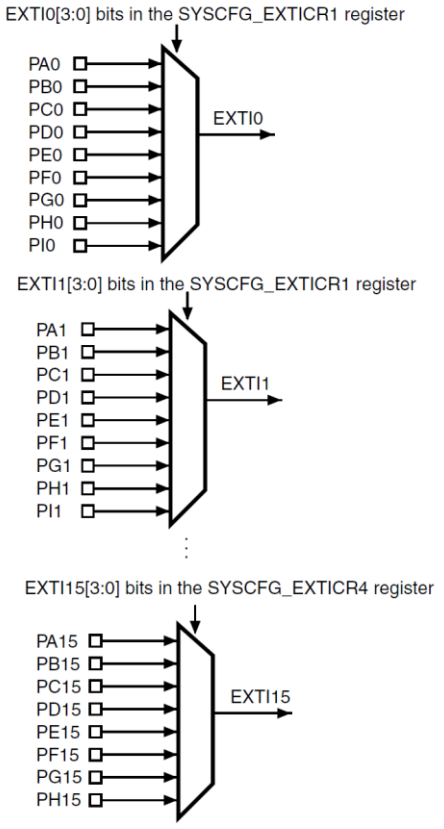
Polling & Interrupt

STM32 외부 인터럽트



STM32 외부 인터럽트

- 외부 인터럽트/이벤트 라인 맵핑
  - 140개의 GPIO는 다음의 16개 EXTI 라인이 연결되며 외부 인터럽트/이벤트 라인에 접속
  - EXTI 라인 16은 PVD 출력에 접속
  - EXTI 라인 17은 RTC 알람 이벤트에 접속
  - EXTI 라인 18은 USB OTG FS Wakeup 이벤트에 접속
  - EXTI 라인 19은 이더넷 Wakeup 이벤트에 접속
  - EXTI 라인 20은 USB (FS 구성) OTG HS Wakeup 이벤트
  - EXTI 라인 21은 RTC 임의 변경 및 타임 스탬프 이벤트
  - EXTI 라인 22은 RTC Wakeup 이벤트에 접속



Polling & Interrupt

STM32 외부 interrupt handler

Irq	Handler	Description
EXTI0_IRQn	EXTI0_IRQHandler	Handler for pins connected to line 0
EXTI1_IRQn	EXTI1_IRQHandler	Handler for pins connected to line 1
EXTI2_IRQn	EXTI2_IRQHandler	Handler for pins connected to line 2
EXTI3_IRQn	EXTI3_IRQHandler	Handler for pins connected to line 3
EXTI4_IRQn	EXTI4_IRQHandler	Handler for pins connected to line 4
EXTI9_5_IRQn	EXTI9_5_IRQHandler	Handler for pins connected to line 5 to 9
EXTI15_10_IRQn	EXTI15_10_IRQHandler	Handler for pins connected to line 10 to 15

Polling & Interrupt

STM32 Interrupt register

- **Interrupt mask register (EXTI_IMR)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									MR22	MR21	MR20	MR19	MR18	MR17	MR16
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.
Bits 22:0 **MRx:** Interrupt mask on line x
0: Interrupt request from line x is masked
1: Interrupt request from line x is not masked

Polling & Interrupt

STM32 Interrupt register

- **Event mask register (EXTI_EMR)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									MR22	MR21	MR20	MR19	MR18	MR17	MR16
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.
Bits 22:0 **MRx:** Event mask on line x
0: Event request from line x is masked
1: Event request from line x is not masked

Polling & Interrupt

STM32 Interrupt register

- **Rising trigger selection register (EXTI_RTSR)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									TR22	TR21	TR20	TR19	TR18	TR17	TR16
									rw	rw	rw	rw	rw	rw	rw

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

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

Bits 22:0 **TRx:** Rising trigger event configuration bit of line x

0: Rising trigger disabled (for Event and Interrupt) for input line

1: Rising trigger enabled (for Event and Interrupt) for input line

Polling & Interrupt

STM32 Interrupt register

- **Falling trigger selection register (EXTI_FTSR)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									TR22	TR21	TR20	TR19	TR18	TR17	TR16
									rw	rw	rw	rw	rw	rw	rw

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

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

Bits 22:0 **TRx:** Falling trigger event configuration bit of line x

0: Falling trigger disabled (for Event and Interrupt) for input line

1: Falling trigger enabled (for Event and Interrupt) for input line.

Polling & Interrupt

STM32 Interrupt register

- **Software interrupt event register (EXTI_SWIER)**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWIER 15	SWIER 14	SWIER 13	SWIER 12	SWIER 11	SWIER 10	SWIER 9	SWIER 8	SWIER 7	SWIER 6	SWIER 5	SWIER 4	SWIER 3	SWIER 2	SWIER 1	SWIER 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 22:0 **SWIERx:** Software Interrupt on line x

If interrupt are enabled on line x in the EXTI_IMR register, writing '1' to SWIERx bit when it is set at '0' sets the corresponding pending bit in the EXTI_PR register, thus resulting in an interrupt request generation.

This bit is cleared by clearing the corresponding bit in EXTI_PR (by writing a 1 to the bit).

Polling & Interrupt

STM32 Interrupt register

- **Pending register (EXTI_PR)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									PR22	PR21	PR20	PR19	PR18	PR17	PR16
									rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

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

Bits 22:0 **PRx:** Pending bit

0: No trigger request occurred

1: selected trigger request occurred

This bit is set when the selected edge event arrives on the external interrupt line.

This bit is cleared by programming it to ‘1’.

Polling & Interrupt

실습 0: User SW를 이용한 User LED의 On/Off (Polling 방법)

- 기존에 실습한 코드가 polling 방식으로 구현한 것임

```
void LED_init(void)
{
   RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);

   GPIO_InitTypeDef GPIO_InitStructure;

   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
   GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
   GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;

   GPIO_Init(GPIOA, &GPIO_InitStructure);
}

void SW_init(void)
{
   RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);

   GPIO_InitTypeDef GPIO_InitStructure;

   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_13;
   GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;

   GPIO_Init(GPIOC, &GPIO_InitStructure);
}
```

```
int main()
{
   LED_init();
   SW_init();

   while(1)
   {
    if(GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_13) == 1)
      GPIO_SetBits(GPIOA, GPIO_Pin_5);
    else
      GPIO_ResetBits(GPIOA, GPIO_Pin_5);
   }

  return 0;
}
```

Polling & Interrupt

실습 1: User SW를 이용한 User LED의 On/Off (Interrupt 방법)

- User 스위치가 눌리면 User LED가 켜지게 구현하는데,
  Interrupt방식으로 구현할 것

```
int main()
{
   LED_init();
   SW_init();

   while(1)
   {
    if(GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_13) == 1)
      GPIO_SetBits(GPIOA, GPIO_Pin_5);
    else
      GPIO_ResetBits(GPIOA, GPIO_Pin_5);
    }

  return 0;
}
```

메인 함수에서
User 스위치가 눌렸는지
반복적으로 체크하는 것이 아니고

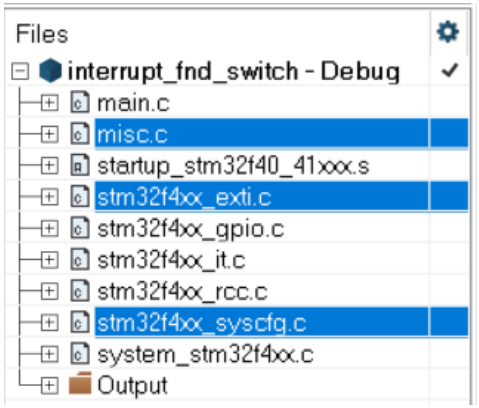
User 스위치가 눌리면 (GPIOC 13번 핀),
interrupt request가
발생하고, LED가 동작하는 방식!

Polling & Interrupt

실습 1: User SW를 이용한 User LED의 On/Off (Interrupt 방법)

- 프로젝트 새로 구성하시오 (lec3 자료 참고)

- 프로젝트 파일 추가
  - 폴더 : STM32F4xx_DSP_StdPeriph_Lib_V1.8.0₩Libraries₩ STM32F4xx_StdPeriph_Driver₩src
  - 파일 : misc.c, stm32f4xx_exti.c, stm32f4xx_syscfg.c



Polling & Interrupt

실습 1: User SW를 이용한 User LED의 On/Off (Interrupt 방법)

```
int cnt = 0; // 글로벌 변수 하나 설정

void LED_init(void)
{
   RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);

   GPIO_InitTypeDef GPIO_InitStructure;

   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
   GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
   GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;

   GPIO_Init(GPIOA, &GPIO_InitStructure);
}
```

```
void SW_init(void)
{
   RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);

   GPIO_InitTypeDef GPIO_InitStructure;

   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_13;
   GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;

   GPIO_Init(GPIOC, &GPIO_InitStructure);
}
```

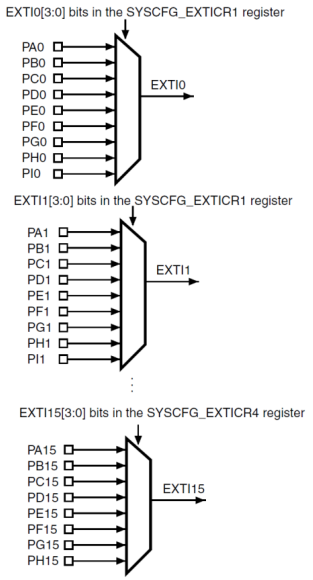
Polling & Interrupt

실습 1: User SW를 이용한 User LED의 On/Off (Interrupt 방법)

Position	Priority	Type of priority	Acronym	Description	Address
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000 0058
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000 005C
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000 0060
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000 0064
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000 0068
11	18	settable	DMA1_Stream0	DMA1 Stream0 global interrupt	0x0000 006C

```
void EXTI13_Configuration(void)
{
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_SYSCFG, ENABLE);
  SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOC, EXTI_PinSource13);

  EXTI_InitTypeDef EXTI_InitStructure;
  NVIC_InitTypeDef NVIC_InitStructure;

  EXTI_InitStructure.EXTI_Line = EXTI_Line13;
  EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
  EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
  EXTI_InitStructure.EXTI_LineCmd = ENABLE;
  EXTI_Init(&EXTI_InitStructure);

  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);
  NVIC_InitStructure.NVIC_IRQChannel = EXTI15_10_IRQn;
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);
}
```

Port C의 13번 핀이 스위치임
이 핀에 변화가 생김을 감지해야 함



Polling & Interrupt

실습 1: User SW를 이용한 User LED의 On/Off (Interrupt 방법)

Irq	Handler	Description
EXTI0_IRQn	EXTI0_IRQHandler	Handler for pins connected to line 0
EXTI1_IRQn	EXTI1_IRQHandler	Handler for pins connected to line 1
EXTI2_IRQn	EXTI2_IRQHandler	Handler for pins connected to line 2
EXTI3_IRQn	EXTI3_IRQHandler	Handler for pins connected to line 3
EXTI4_IRQn	EXTI4_IRQHandler	Handler for pins connected to line 4
EXTI9_5_IRQn	EXTI9_5_IRQHandler	Handler for pins connected to line 5 to 9
EXTI15_10_IRQn	EXTI15_10_IRQHandler	Handler for pins connected to line 10 to 15

```
void EXTI13_Configuration(void)
{
 RCC_APB2PeriphClockCmd(RCC_APB2Periph_SYSCFG, ENABLE);
 SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOC, EXTI_PinSource13);

 EXTI_InitTypeDef EXTI_InitStructure;
 NVIC_InitTypeDef NVIC_InitStructure;

 EXTI_InitStructure.EXTI_Line = EXTI_Line13;
 EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
 EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
 EXTI_InitStructure.EXTI_LineCmd = ENABLE;
 EXTI_Init(&EXTI_InitStructure);

 NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);
 NVIC_InitStructure.NVIC_IRQChannel = EXTI15_10_IRQn;
 NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
 NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
 NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
 NVIC_Init(&NVIC_InitStructure);
}
```

NVIC_PriorityGroup	NVIC_IRQChannelPreemptionPriority	NVIC_IRQChannelSubPriority	Description
NVIC_PriorityGroup_0	0	0-15	0 bits for pre-emption priority 4 bits for subpriority
NVIC_PriorityGroup_1	0-1	0-7	1 bits for pre-emption priority 3 bits for subpriority
NVIC_PriorityGroup_2	0-3	0-3	2 bits for pre-emption priority 2 bits for subpriority
NVIC_PriorityGroup_3	0-7	0-1	3 bits for pre-emption priority 1 bits for subpriority
NVIC_PriorityGroup_4	0-15	0	4 bits for pre-emption priority 0 bits for subpriority

Polling & Interrupt

실습 1: User SW를 이용한 User LED의 On/Off (Interrupt 방법)

Request가 오면, 어떤 동작을 수행할 것인가?

```
void EXTI15_10_IRQHandler(void)
{
  /* Make sure that interrupt flag is set */
                                                    0
  if (EXTI_GetITStatus(EXTI_Line13) != RESET)
  {
    cnt ++;
    if(cnt % 2 == 1)
    {
      GPIO_SetBits(GPIOA, GPIO_Pin_5);
    }
    else
    {
      GPIO_ResetBits(GPIOA, GPIO_Pin_5);
    }
   }
     /* Clear interrupt flag */
   EXTI_ClearITPendingBit(EXTI_Line13);
  }
}
```

이 부분에 동작 코드를 구현하면 됨

Irq	Handler	Description
EXTI0_IRQn	EXTI0_IRQHandler	Handler for pins connected to line 0
EXTI1_IRQn	EXTI1_IRQHandler	Handler for pins connected to line 1
EXTI2_IRQn	EXTI2_IRQHandler	Handler for pins connected to line 2
EXTI3_IRQn	EXTI3_IRQHandler	Handler for pins connected to line 3
EXTI4_IRQn	EXTI4_IRQHandler	Handler for pins connected to line 4
EXTI9_5_IRQn	EXTI9_5_IRQHandler	Handler for pins connected to line 5 to 9
EXTI15_10_IRQn	EXTI15_10_IRQHandler	Handler for pins connected to line 10 to 15

Request가 발생하면 담당하는 handler

```
int main()
{
  LED_init();
  SW_init();
  EXTI13_Configuration();

  while(1)
  {

  }
 return 0;
}
```

과제

- <실습 5. User Switch를 누른 횟수로 FND 증가시키기> 를 Interrupt 방법으로 동작시키는 코드 작성
- FND의 A,B,C,D,E,F,G,DP를 저항(220Ω)을 연결한 후 PC0~PC7과 연결
- FND의 common인 12,9,8,6핀을 PB0~PB3과 연결
- 스위치는 port C 15번 핀 사용
- 기본 필요 변수 & 함수
  - Interrupt 설정 함수
  - FND 설정 함수
  - Delay 함수
  - Switch 설정함수
  - FND 출력 함수
  - Font 배열
  - 글로벌 변수: count

과제

```
void EXTI13_Configuration(void)
{
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_SYSCFG, ENABLE);
  SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOC, EXTI_PinSource13);

  EXTI_InitTypeDef EXTI_InitStructure;
  NVIC_InitTypeDef NVIC_InitStructure;

  EXTI_InitStructure.EXTI_Line = EXTI_Line13;
  EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
  EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
  EXTI_InitStructure.EXTI_LineCmd = ENABLE;
  EXTI_Init(&EXTI_InitStructure);

  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);
  NVIC_InitStructure.NVIC_IRQChannel = EXTI15_10_IRQn;
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);
}
```

EXTI_Trigger_Rising
EXTI_Trigger_Falling
EXTI_Trigger_Rising_Falling
으로 바꾸어서 코딩해보고,
결과가 어떻게 달라지는지도
확인해보기

Thank you.