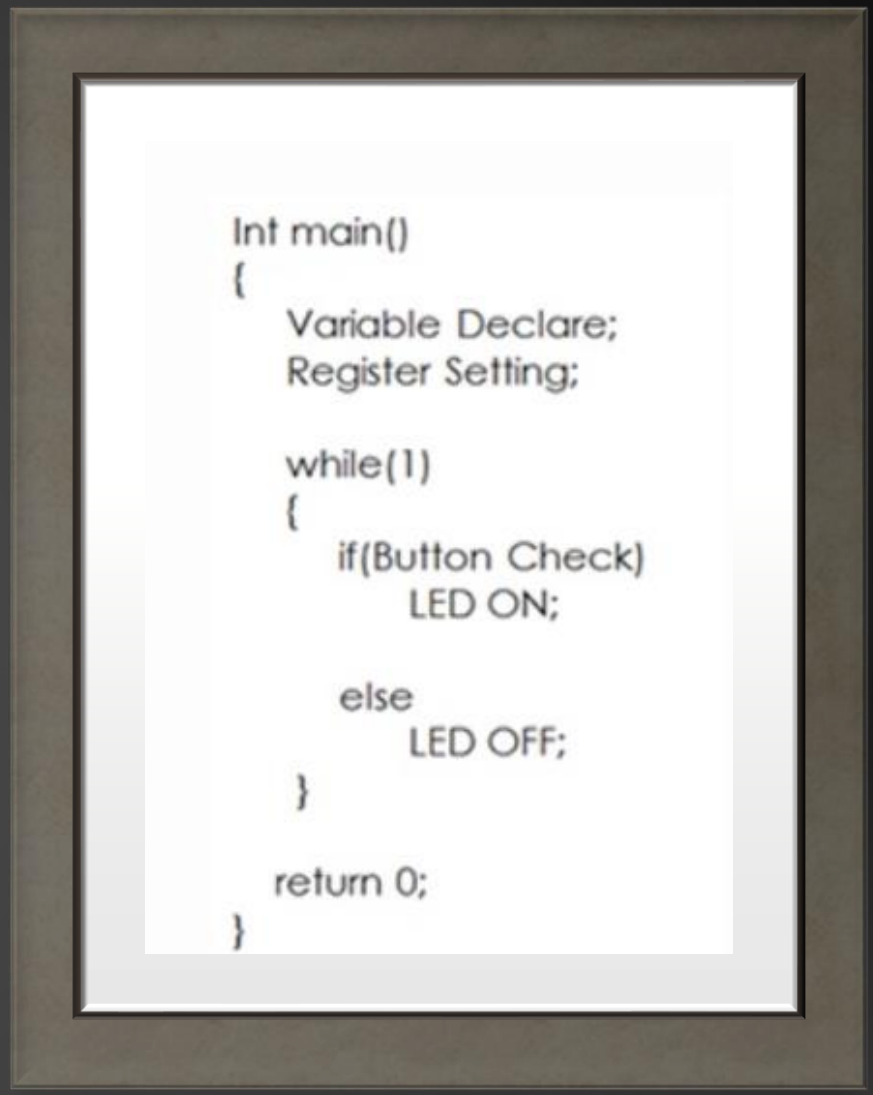
7조 예비실험 발표

강나훈,김호석,송민수,처흠

POLLING

- 폴링 방식은 cpu가 특정 이벤트의 발생을 처리하기 위해 특정 주기마다 신호가 입력되었는지 검사하는 방식

- 구현이 쉽지만 시스템 성능 저하를 일으킨다.

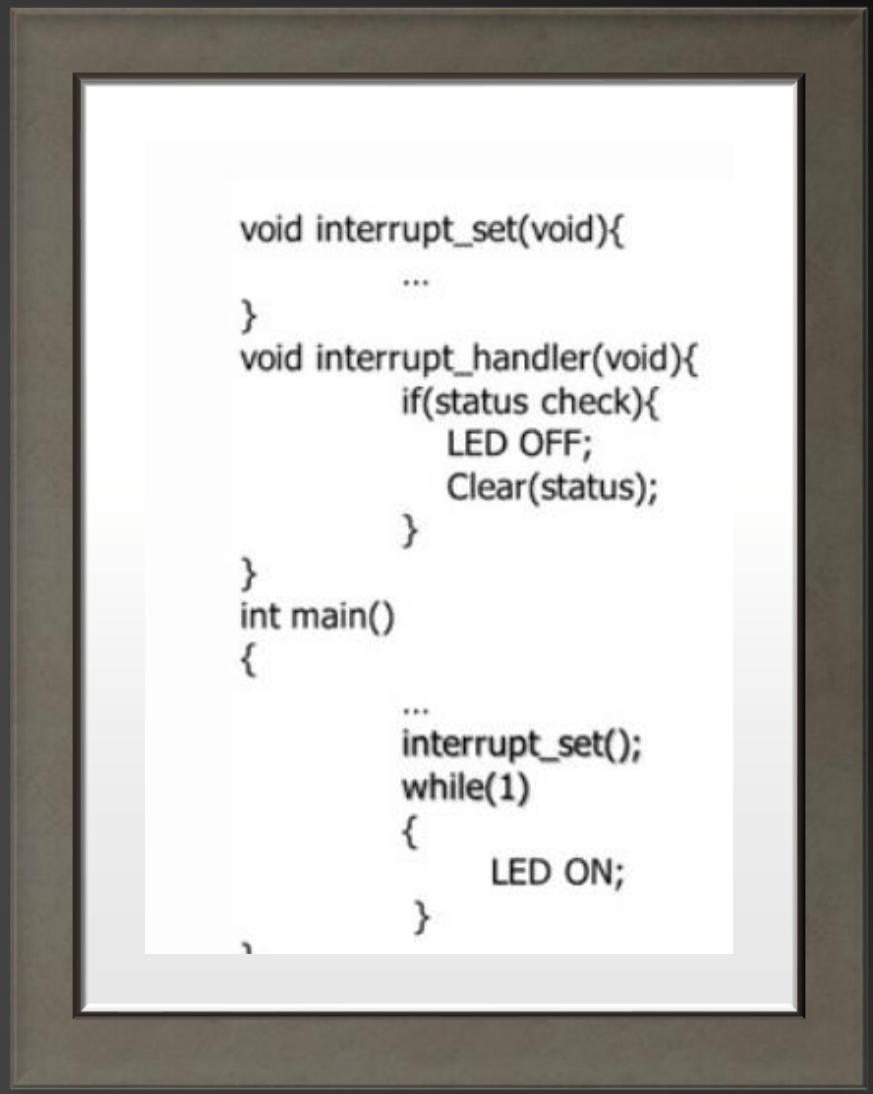


```
Int main()
{
    Variable Declare;
    Register Setting;

    while(1)
    {
        if(Button Check)
            LED ON;

        else
            LED OFF;
    }

    return 0;
}
```

INTERRUPT

- Interrupt 방식은 CPU가 다른 연산을 수행하고있던 도중이라도 특정 이벤트가 발생하면 인터럽트 서비스 루틴을 수행함.

- 구현은 복잡하지만 처리가 정확해 시스템 부하를 줄일수있다.



```
void interrupt_set(void){
        ...
}
void interrupt_handler(void){
        if(status check){
            LED OFF;
            Clear(status);
        }
}
int main()
{
        ...
        interrupt_set();
        while(1)
        {
            LED ON;
        }
```

INTERRUPT의 종류

- 1. Software Interrupt
- software interrupt 사용자가 프로그램 내에서 interrupt 가 발생하도록 설정하는 것이다. 즉, 현재 실행하고 있는 instruction 으로부터 interrupt 발생 요청을 받는다.
- 2. Hardware Interrupt
- hardware interrupt 는 비동기식 event 처리로 peripheral(주변장치) 의 요청에 의해 발생하는 interrupt 이다.

NVIC

NVIC 는 중첩된 interrupt 를 제어하는 기능이다. 모든 exception 에 대해 priority 가 설정되어 있고 이 우선순위에 따라 interrupt 를 처리하는데 두가지 종류가있다.

preemtion : priority 가 높은 interrupt 가 들어올 때 현재 작업을 멈추고 해당 interrupt 를 진행

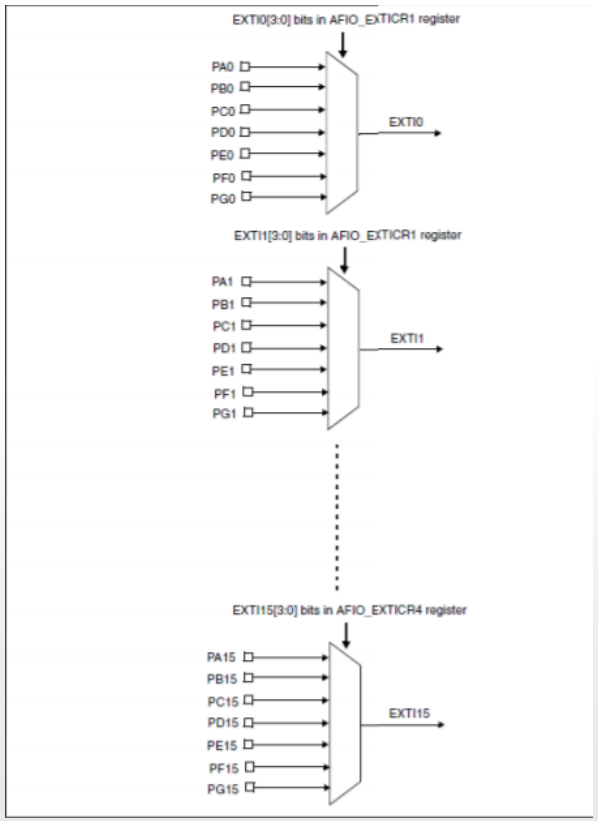
subpriority : interrupt 수행 중이 아닌, 대기 중에 있어서 priority 가 높은 것을 먼저 수행

@code
The table below gives the allowed values of the pre-emption priority and subpriority according to the Priority Grouping configuration performed by NVIC_PriorityGroupConfig function

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

EXTI

- EXTI 는 외부에서 신호가 입력될 경우 device 에 event 나 interrupt 가 발생되는 기능이다.
- 입력 받을 수 있는 신호는 rising edge, falling edge, rising & falling edge 이다
- Ex)pa15번으로 외부 인터럽트를 받고싶다면 EXTI15를 이용해야함



예비실험 과정

- USART를 위해 저번주차에 CLOCK설정을 위한 함수 2개
- (SysInit, SetSysClock)필요.

```
#include <misc.h>
#include <stm32f10x.h>
#include <stm32f10x_exti.h>
#include <stm32f10x_gpio.h>
#include <stm32f10x_rcc.h>
#include <stm32f10x_usart.h>
```

- 라이브러리 안의 함수를 검색하며 진행

- EXTI를 통해 Interrupt 설정, NVIC로 우선순위 설정
- 버튼 입력 및 UART 입력이 들어오면 Interrupt Handler 가 호출되어 처리
- Joystick select 버튼을 눌렀다 뗄 때 4개의 LED가 순서대로 점멸 반복 (물
- 다시 Joystick select 버튼을 눌렀다 때면 모든 LED 꺼짐
- User S1 버튼 누르면 PUTTY 터미널에 'ABCD₩r₩n' 출력
- Joystick select와 User S1 동작은 인터럽트 핸들러에서 구현

주소값사용 -> 구조체사용

이번 실험부터는 주소값을 직접
이용하는 방식이 아닌 헤더파일에 의해
정의되어있는 구조체와 함수를 이용한다.

- 정의된 주소 값 사용 vs Structures, functions 사용

```
static int i;

int main()
{
   RCC->APB2ENR &= ~(RCC_APB2ENR);
   RCC->APB2ENR |= RCC_APB2ENR_IOPDEN;
   GPIOD->CRL &= ~(GPIO_CRL_CNF2 | GPIO_CRL_MODE2);
   GPIOD->CRL |= GPIO_CRL_MODE2_0;



   while(1) {
      (*(volatile unsigned int *) 0x40011410) |= 0x04;
      for(i = 0 ; i < 1000000 ; i++);
      (*(volatile unsigned int *) 0x40011414) |= 0x04;
      for(i = 0 ; i < 1000000 ; i++);
   }
}
```

```
int main()
{
   GPIO_InitTypeDef GPIO_InitStructure;
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);

   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
   GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
   GPIO_Init(GPIOD, &GPIO_InitStructure);

   while (1)
   {
      GPIO_SetBits(GPIOD, GPIO_Pin_2);
      Delay(1000);
      GPIO_ResetBits(GPIOD, GPIO_Pin_2);
      Delay(1000);
   }
}
```

EXTI 구조체 예제

- void EXTI_Configure(void){
- EXTI_InitTypeDef   EXTI_InitStructure;
- EXTI_InitStructure.EXTI_Line=EXTI_Line11;
- EXTI_InitStructure.EXTI_LineCmd = ENABLE;
- EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
- EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising;
- EXTI_Init(&EXTI_InitStructure);
- }
- Nvic,USART,Gpio 모두 라이브러리에 들어가 검색후 진행

인터럽트 루틴

- 전역 변수 설정 후 인터럽트가 호출될때마다 전역변수의 값을 바꿔
- 인터럽트 서비스 루틴 후 복귀했을때
- 메인 함수에서의 루틴이 달라지는 방식을 이용해 예비실험 해결