주차별 실험 계획

주차	날짜	실험내용
1	09.02	분반편성 및 Overview
2	09.08	개발 환경 구축 및 개발 장비 교육
3	09.15	디버깅 툴(J-Link) 및 레지스터와 주소 제어를 통한 임베디드 펌웨어 개발
4	09.22	휴강 (추석)
5	09.29	스캐터 로딩 파일 및 플래시 메모리 이해
6	10.06	Polling 방식을 이용한 UART 통신 및 Clock control
7	10.13	Interrupt 방식을 활용한 GPIO 제어 및 UART 통신
8	10.20	휴강 (중간고사 없음)
9	10.27	Bluetooth 및 납땜
10	11.03	TFT-LCD 제어 및 ADC 구현
11	11.10	Timer 및 PWM 구현
12	11.17	DMA 구현
13~	11.24~12.24	텀 프로젝트 진행 (최종 검사일은 미정)
15~16	12.08~12.19	**기말고사 (시험일은 미정)**

차후에 변경 가능

공지

예비 발표 방법 변경

- **발표 영상을 녹화하여 조교 이메일 (chrismail@naver.com) 로 제출**
- **모든 학생은 조교가 PLATO에 업로드한 동영상 각자 시청**

제출 기한

10월 27일	11월 3일	11월 10일	11월 17일
11, 6	5, 2	9, 4	3

Pusan National University

INS LAB
IoT NETWORK & SYSTEMS LAB

임베디드 시스템 설계 및 실험
수요일 분반

October 13, 2021

6주차
Interrupt 방식을 활용한 GPIO 제어 및 UART 통신

조교
김준명

중간고사 주 (10월 20일) 휴강

공지

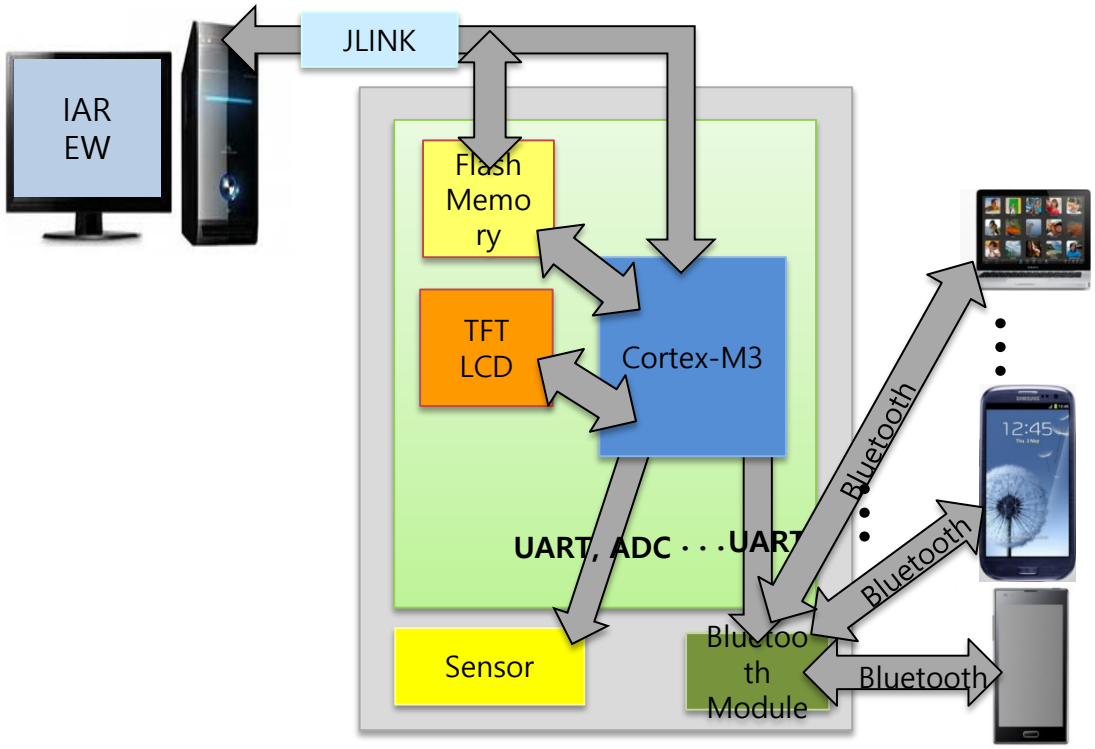
실험 (35)					설계 과제 (65)				
출석 태도	발표	보고서	수업 검사	소계	제안서	최종 보고서	필기 시험	동작 검사	소계
10	5	5	15	**35**	10	10	20	25	**65**

텀 프로젝트 제안서 제출 (11월 2일 23시 59분 까지 e-mail로 PDF)

평가 항목
- 완성도
- 동작안정성
- 구현난이도
- 독창성

- 예시 참고
- 목적과 내용 사용 센서, 시나리오, Flow Chart 작성
- 시스템 구성도 작성
- 사용할 센서의 제품명과 스펙 기재
  - ✓ 블루투스모듈(FB755AC), LCD 모듈(3.2" TFT LCD/SC), 서보모터(SG90), 조도센서 실험 중에 분배 할 예정
- 사용할 센서의 링크 (디마이스마트)와 가격 및 개수, 해외 배송 X

텀 프로젝트 제안서 시스템 구성도 예시



텀 프로젝트 아이디어 예시

- 사용자의 자세를 교정해주는 스마트 의자
- 손가락 동작으로 PC에 타이핑 하는 손가락 키보드

텀 프로젝트 제약 사항

- **인터럽트 반드시 활용**
- **센서 간의 의존성 필수 (센서 2개 이상, 각자 폴링 방식으로 동작하지 말고 한 센서 값이 다른 센서 이용을 호출하는 시나리오)**
- **블루투스 연동 필수**
- **차량을 이용하는 시나리오일 경우 릴레이 모듈 말고 모터 드라이버 반드시 이용**
- **센서 및 재료 구매 링크는 "디바이스마트" 만 허용 / 조별 최대 5만원**
  - https://www.devicemart.co.kr/
  - [해외] 적혀 있는 물품은 구매하지 마세요

Tip

- **구매하려는 센서 사용 방법을 꼭 확인하기**
  - i2c, SPI 등등 구현하기 어려운 프로토콜을 이용하는 센서는 지양할 것
- **보기에 깔끔할수록 좋은 점수, 꾸미기 재료도 같이 조사하기**

INS LAB
IoT NETWORK & SYSTEMS LAB

Contents

실험 내용

실험 목적

- **Interrupt 방식을 활용한 GPIO 제어 및 UART 통신**
- **라이브러리 함수 사용법 숙지**

라이브러리 구조체 및 함수 사용

직접 주소로 접근	(*(volatile unsigned int *) 0x40021018) &= ~0x20; (*(volatile unsigned int *) 0x40021018) |= 0x20; (*(volatile unsigned int *) 0x40011000) &= ~0x00000F00; (*(volatile unsigned int *) 0x40011000) |= 0x00000400;
정의된 주소 값 사용	RCC->APB2ENR &= ~(RCC_APB2ENR); RCC->APB2ENR |= RCC_APB2ENR_IOPDEN; GPIOD->CRL &= ~(GPIO_CRL_CNF2 | GPIO_CRL_MODE2); GPIOD->CRL |= GPIO_CRL_MODE2_0;
구조체 및 함수 사용	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);

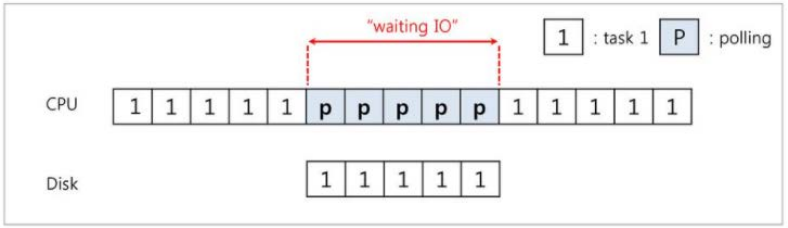
라이브러리 구조체 및 함수 사용

직접 주소로 접근	(*(volatile unsigned int *) 0x40011410) |= 0x04; (*(volatile unsigned int *) 0x40011414) |= 0x04;
정의된 주소 값 사용	GPIOD->BSRR |= GPIO_BSRR_BS2; GPIOD->BRR |= GPIO_BRR_BR2;
구조체 및 함수 사용	GPIO_SetBits(GPIOD, GPIO_Pin_2); GPIO_ResetBits(GPIOD, GPIO_Pin_2);

Polling vs Interrupt

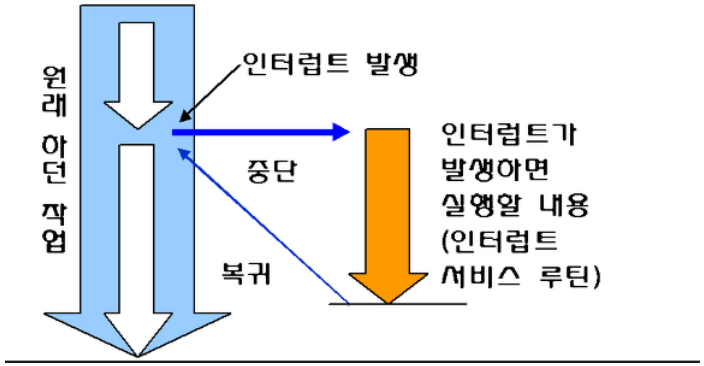
- **Polling**

  CPU 가 특정 이벤트를 처리하기 위해 이벤트가 발생할 때까지 모든 연산을 이벤트가 발생하는지 감시하는 방식



- **Interrupt**

  CPU가 특정 이벤트 발생시 현재 작업을 멈추고 해당 인터럽트 서비스 루틴을 수행 후 다시 이전 작업으로 돌아가는 방식



Interrupt 종류

- **Hardware Interrupt**
  - 비동기식 이벤트 처리로 주변장치의 요청에 의해 발생하는 인터럽트
  - 높은 우선 순위
  - 하드 디스크 읽기 요청/끝, 키보드 입력, 센서 값 업데이트 등의 이벤트에 발생

- **Software Interrupt**
  - 동기식 이벤트 처리로 소프트웨어가 프로그램 내에서 인터럽트가 발생하도록 설정하는 인터럽트
  - 낮은 우선 순위
  - Trap, Exception 등이 여기에 포함

- **EXTI (External Interrupt)**

  외부에서 신호가 입력될 경우 Device 에 Event나 Interrupt 가 발생되는 기능

  입력 받을 수 있는 신호는 Rising-Edge, Falling-Edge, Rising & Falling-Edge

  각 Port 의 n번 Pine 의 EXTIn 에 연결

- **EXTI 는 Event Mode 와 Interrupt Mode 를 선택하여 설정 가능**

  Interrupt Mode 로 설정할 경우 Interrupt 가 발생해 해당 Interrupt Handler 가 동작

  20개의 Edge Detector Line 으로 구성되어 각 Line 이 설정에 따라 Rising/Falling Trigger 를 감지

- Interrupt Request 는 Mask Register 를 통해 알 수 있다
  Processor 는 Interrupt 를 인지하여 처리하기 전에 Pending Register
  (어떤 Interrupt 가 발생되었는지 저장) 를 검사하여 발생된 Interrupt 중 Prioritry가
  가장 높은 Interrupt 를 처리
- 외부 Interrupt 는 EXTI0 ~ EXTI15까지 각 Port 의 Pin 번호가 Interrupt Pin과 매치

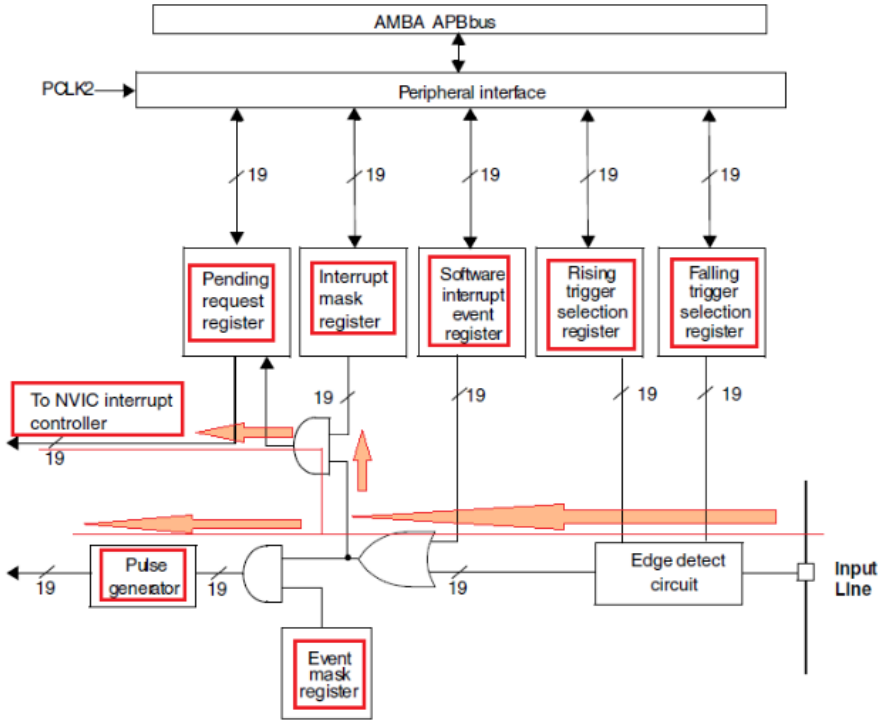
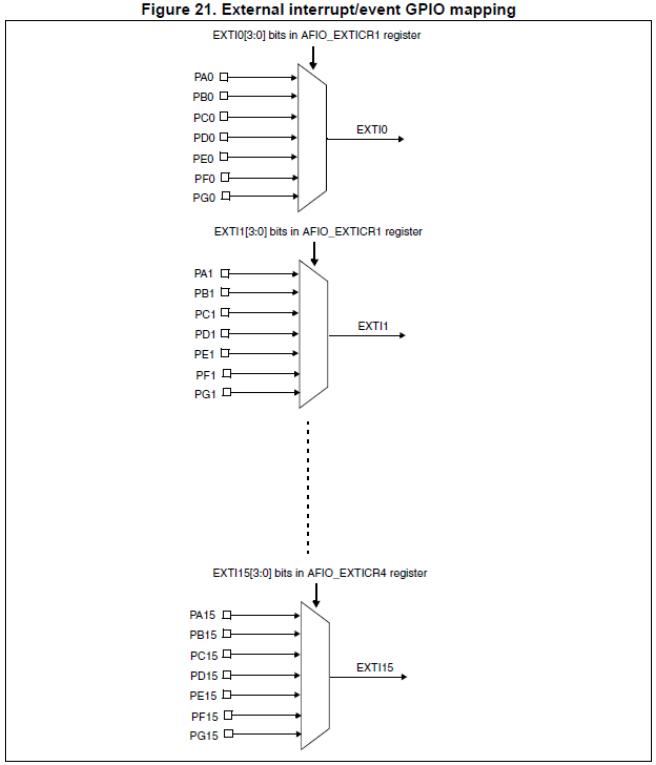


Figure 21. External interrupt/event GPIO mapping



- **EXTI (External Interrupt)**

  모든 GPIO 핀들은 EXTI line 을 통해 연결되어 있다

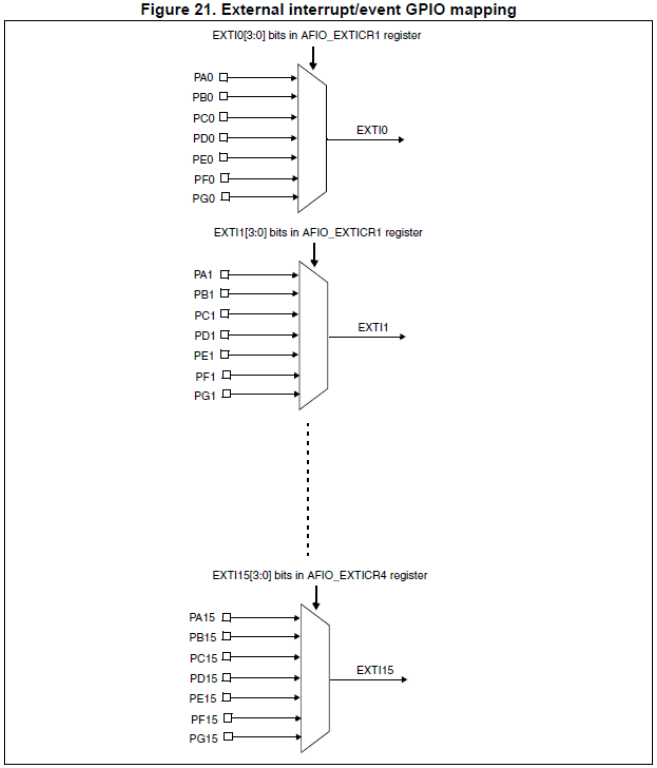
  EXTICR1 레지스터를 통해 입력 받을 포트를 선택 하며 같은 번호의 핀들은 같은 라인을 공유

EXTI MUX에 모든 Port의 Line의 숫자가 같이 들어옴
선언 시에 EXTI()는 사용할 핀 번호를 사용하면 됨

EXTI를 사용할때 Line, Mode, Trigger, Lineconfig 설정
EXTI를 선언 했을 시에는 반드시 Handler 또한 구현 필요

Reference Manual 206,208 참고

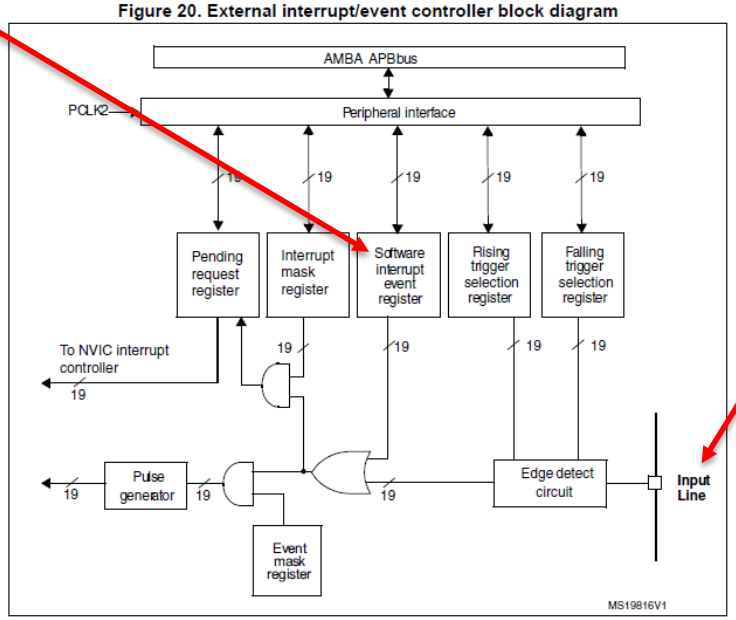
Figure 21. External interrupt/event GPIO mapping



- **EXTI (External Interrupt)**

  EXTI line (Input Line) 을 통해 입력 받은 신호와 레지스터 설정들을 비교하여 NVIC controller 로 보냄

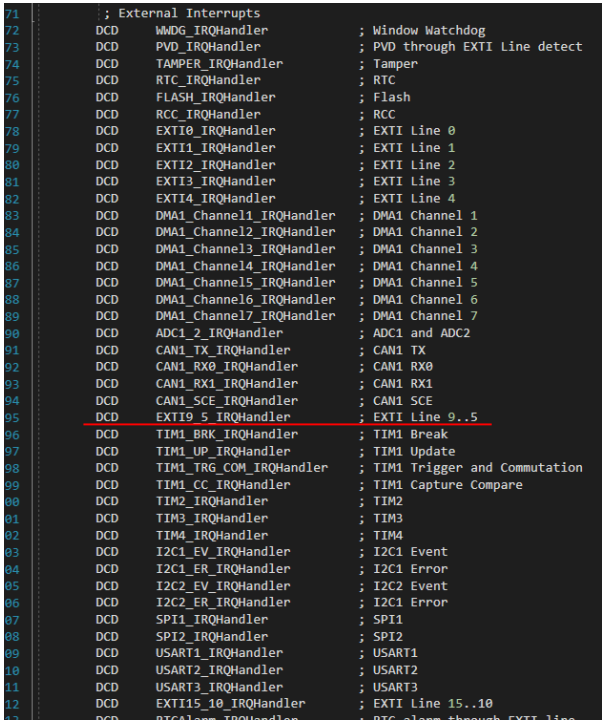
  소프트웨어 인터럽트도 중간에 비교되는 것을 확인 가능



Reference Manual 206,208 참고

실험 과정

- Libraries₩CMSIS₩DeviceSupport₩Startup ₩startup_stm32f10x_cl.s 내용을 참고
- 각 인터럽트 핸들러에서 호출되는 함수의 프로토타입이 정의 되어 있음
- 정의된 이름을 그대로 사용하여 원하는 함수 구현
- 예) 8번 핀을 사용 시에 Handler 이름은 EXTI9_5_IRQHandler 로 선언



```
71          ; External Interrupts
72          DCD     WWDG_IRQHandler           ; Window Watchdog
73          DCD     PVD_IRQHandler            ; PVD through EXTI Line detect
74          DCD     TAMPER_IRQHandler         ; Tamper
75          DCD     RTC_IRQHandler            ; RTC
76          DCD     FLASH_IRQHandler          ; Flash
77          DCD     RCC_IRQHandler            ; RCC
78          DCD     EXTI0_IRQHandler          ; EXTI Line 0
79          DCD     EXTI1_IRQHandler          ; EXTI Line 1
80          DCD     EXTI2_IRQHandler          ; EXTI Line 2
81          DCD     EXTI3_IRQHandler          ; EXTI Line 3
82          DCD     EXTI4_IRQHandler          ; EXTI Line 4
83          DCD     DMA1_Channel1_IRQHandler  ; DMA1 Channel 1
84          DCD     DMA1_Channel2_IRQHandler  ; DMA1 Channel 2
85          DCD     DMA1_Channel3_IRQHandler  ; DMA1 Channel 3
86          DCD     DMA1_Channel4_IRQHandler  ; DMA1 Channel 4
87          DCD     DMA1_Channel5_IRQHandler  ; DMA1 Channel 5
88          DCD     DMA1_Channel6_IRQHandler  ; DMA1 Channel 6
89          DCD     DMA1_Channel7_IRQHandler  ; DMA1 Channel 7
90          DCD     ADC1_2_IRQHandler         ; ADC1 and ADC2
91          DCD     CAN1_TX_IRQHandler        ; CAN1 TX
92          DCD     CAN1_RX0_IRQHandler       ; CAN1 RX0
93          DCD     CAN1_RX1_IRQHandler       ; CAN1 RX1
94          DCD     CAN1_SCE_IRQHandler       ; CAN1 SCE
95          DCD     EXTI9_5_IRQHandler        ; EXTI Line 9..5
96          DCD     TIM1_BRK_IRQHandler       ; TIM1 Break
97          DCD     TIM1_UP_IRQHandler        ; TIM1 Update
98          DCD     TIM1_TRG_COM_IRQHandler   ; TIM1 Trigger and Commutation
99          DCD     TIM1_CC_IRQHandler        ; TIM1 Capture Compare
00          DCD     TIM2_IRQHandler           ; TIM2
01          DCD     TIM3_IRQHandler           ; TIM3
02          DCD     TIM4_IRQHandler           ; TIM4
03          DCD     I2C1_EV_IRQHandler        ; I2C1 Event
04          DCD     I2C1_ER_IRQHandler        ; I2C1 Error
05          DCD     I2C2_EV_IRQHandler        ; I2C2 Event
06          DCD     I2C2_ER_IRQHandler        ; I2C1 Error
07          DCD     SPI1_IRQHandler           ; SPI1
08          DCD     SPI2_IRQHandler           ; SPI2
09          DCD     USART1_IRQHandler         ; USART1
10          DCD     USART2_IRQHandler         ; USART2
11          DCD     USART3_IRQHandler         ; USART3
12          DCD     EXTI15_10_IRQHandler      ; EXTI Line 15..10
```

실험 과정

- **NVIC (Nested Vectored Interrupt Controller)**

  인터럽트 처리 중 또다른 인터럽트 발생시 우선순위를 사용

  우선순위가 높은 인터럽트부터 처리 후 다른 인터럽트 처리

  ARM 보드에서 인터럽트 사용시 NVIC 통하여 우선순위를 결정

  값이 작을수록 우선순위가 높음

```
@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
 ============================================================================================================
@endcode
```

Libraries₩STM32F10x_StdPeriph_Driver_v3.5₩inc₩misc.h 참고

실험 과정

```
typedef struct
{
  uint8_t NVIC_IRQChannel;

  uint8_t NVIC_IRQChannelPreemptionPriority;

  uint8_t NVIC_IRQChannelSubPriority;

  FunctionalState NVIC_IRQChannelCmd;

} NVIC_InitTypeDef;
```

- Pre-emption : 우선순위가 높은 interrupt가 들어오면, 현재 작업을 멈추고 해당 interrupt를 진행 (선점)
- Pre-emption priority 로 선점 우선순위 결정
- sub priority로 아직 대기 중인 ISR들의 순서가 결정

Libraries₩STM32F10x_StdPeriph_Driver_v3.5₩inc₩misc.h 참고

실험 과정

- 레지스터 설정에 구조체를 사용

  이전 실험 까지는 stm32f10x.h 라이브러리 사용하여 주소를 직접 쓰지 않고

  정의된 상수를 사용

- 이번 실험은 추가 라이브러리의 구조체와 함수를 사용 할 것

  함수에 구조체를 넣어서 시행하면 해당 레지스터에 직접 값을 넣는것과 같은 설정 수행

- 구조체 및 함수 동작 숙지 필요

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

int main{
    GPIO_InitTypeDef  GPIO_InitStructure;
    RCC_APB2PeriphClockCmd(RCC_APB2Priph_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, &GPIOD_InitStructure);

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

- Clock Enable 수행

```
void RCC_configuration() {

    RCC_APB2PeriphClockCmd( RCC_APB2Periph_AFIO, ENABLE);
    /*TODO : APB2PeriphClockEnable */

}
```

- GPIO Configuration 수행

```
void GPIO_configuration() {

    GPIO_InitTypeDef GPIOD_init;

    GPIOD_init.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIOD_init.GPIO_Pin = (GPIO_Pin_2 | GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_7);
    GPIOD_init.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOD, &GPIOD_init);

    /*TODO: USART1, JoyStick Config */


    /*TODO: GPIO EXTILineConfig*/


}
```

실험 과정

- EXTI Configuration - 사용할 EXTILine 을 어떤 설정으로 Enable 할 것인지 결정

```
void EXTI_configuration() {
    /*TODO: EXTI configuration [ mode interrupt ] [Trigger_falling] */

}
```

- USART 직렬통신 설정의 정의

```
void USART_configuration() {
    /*TODO: USART1 configuration*/

    /*TODO: USART1 cmd ENABLE*/

    /*TODO: USART1 IT Config*/
}
```

- NVIC Configuration – 각 Interrupt 의 우선순위를 설정

```
void NVIC_configuration() {
    /*TODO: NVIC_configuration */
}
```

- IRQHandler 정의 – 각 Interrupt 들이 발생 하였을 때 처리할 작업을 정의

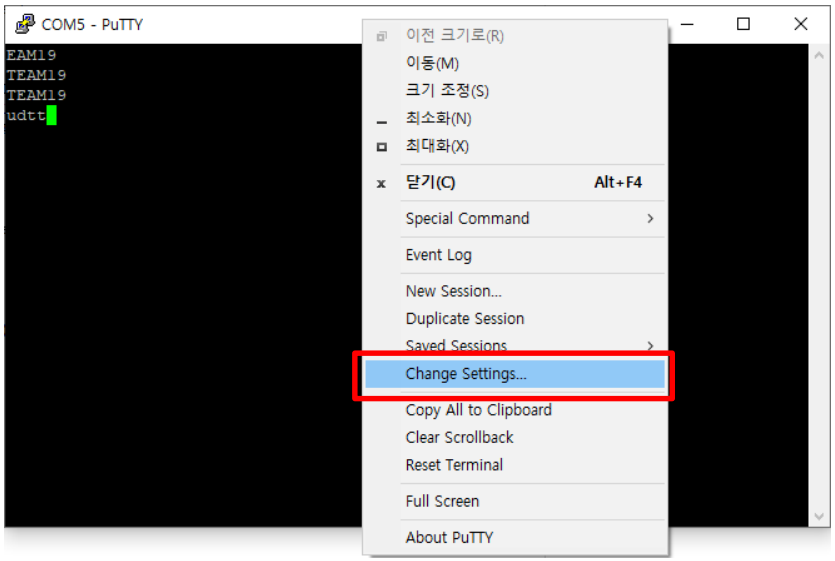
  (주의사항) Interrupt 동작에서는 딜레이가 없어야 함

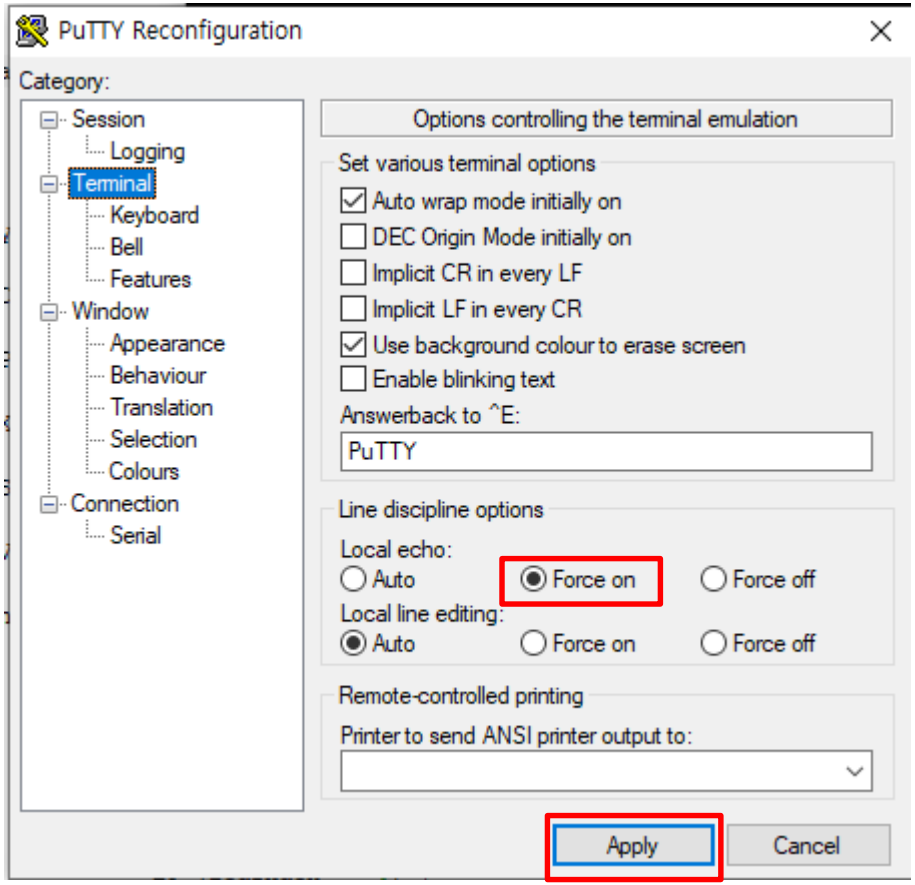
```
/*TODO: IRQHandler */
```

실험 주의사항

- Putty 에서 PC 입력이 콘솔에 보이게 하고 싶을 때

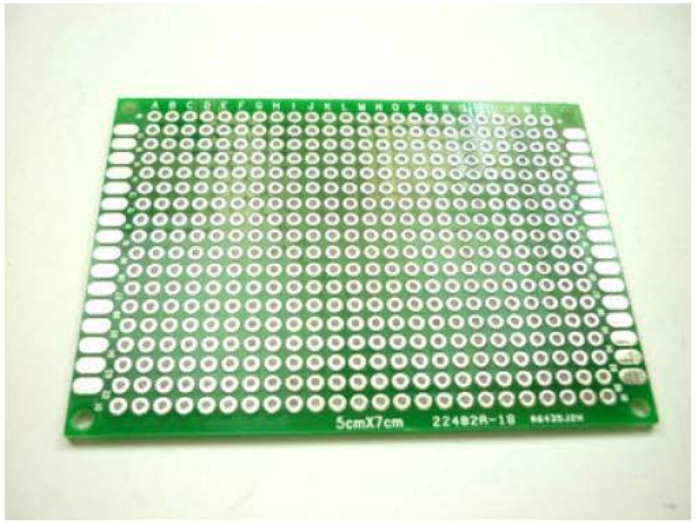
Putty 외의 프로그램들은 echo기능 활성화로 검색

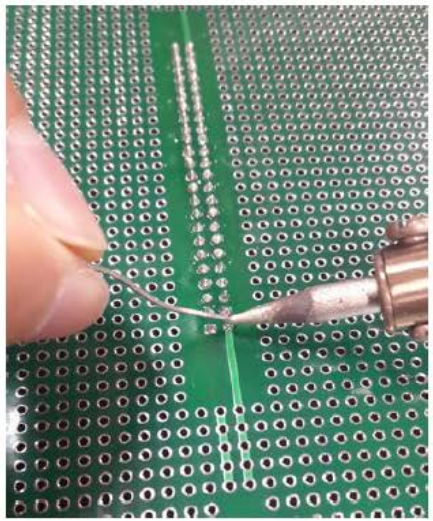




납땜

- 만능기판과 헤더핀을 납땜





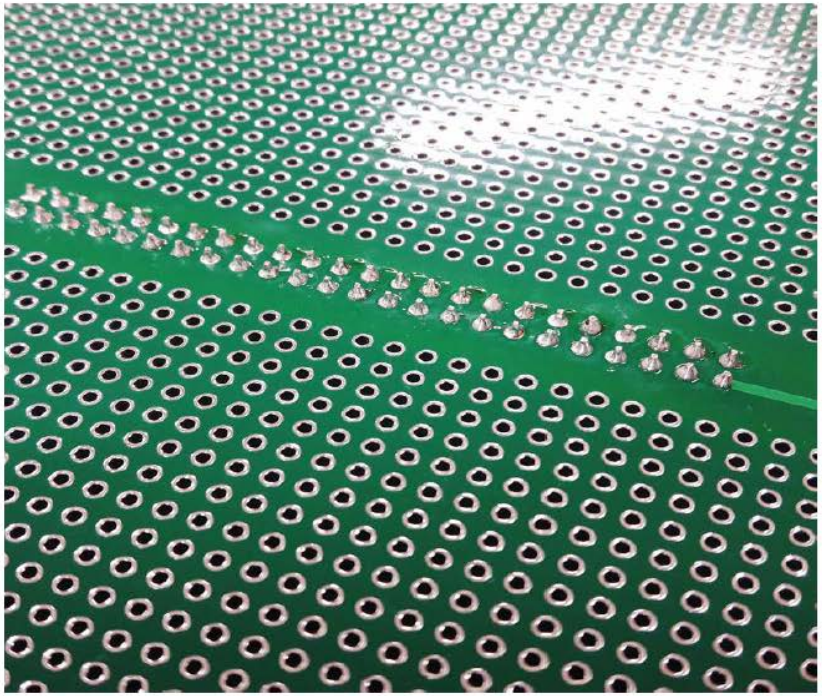
납땜 방법

인두기

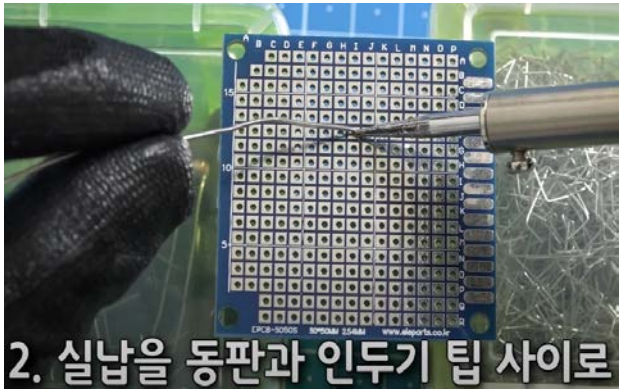
실납

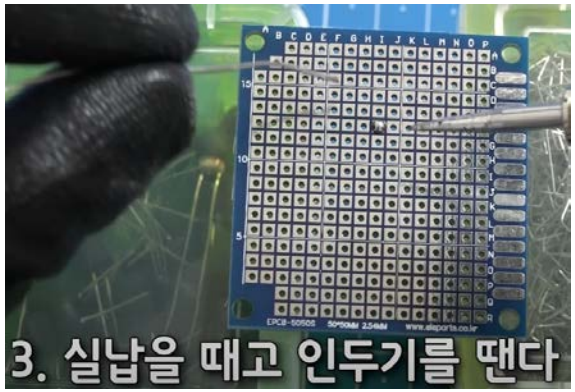
납 흡입기

인두팁 클리너



납땜 방법





실험 주의사항

- 실험 장비들을 연결 및 분리할 때 반드시 모든 전원을 끄고 연결해주세요.
- 장비 사용시 충격이 가해지지 않도록 주의해주세요.
- 자리는 항상 깔끔하게 유지하고 반드시 정리 후 퇴실해주세요.
- 실험 소스 코드와 프로젝트 폴더는 백업 후 반드시 삭제해주세요.
- 장비 관리, 뒷정리가 제대로 되지 않을 경우 해당 조에게 감점이 주어집니다.

- 동작 중 케이블 절대 뽑지 말것
- 보드는 전원으로 USB Port나 어댑터(5V,1A)를 사용할 것 (5V 5A 어댑터(비슷하게 생김)와 혼동하지 말 것, 사용시 보드가 타 버림 -> 감점)
- 디버깅 모드 중에 보드 전원을 끄거나 연결 케이블을 분리하지 말 것!!!

- ->지켜지지 않을 시 해당 조 감점

실험미션

미션 !
별도 미션지 참고

실험 검사

1. 레지스터 및 주소 설정 이해 확인, 구조체 및 함수를 이용하여 코드 작성하였는지 확인
2. 조이스틱 및 LED 동작 및 인터럽트 구현 검사
3. 코드 template 주석상 코멘트로 제약사항으로 있는 부분 잘 읽어보고 코드 제작
   위반시 코드 동작이 정상이더라도 보고서상 감점
4. 땜납 검사(냉납주의)

이번 주 실험 결과 보고서

A. **이론부터 실습까지 전반적인 내용을 포함하도록 작성 (실험 과정 사진 찍으시면 좋아요)**

B. **다음 실험전날 자정전까지 e-mail 제출**

나가실 때, 만드신 코드 및 프로젝트 폴더는 모두 백업하시고 삭제해주세요.
다른 분반 파일은 만지지 마시고 조교에게 알려주세요.
자리 정리정돈 안 되어 있으면 감점합니다!!!