임베디드시스템설계및실습

7 주차 실험 보고서

■ **팀** 4 조

■ **담 당 교 수** 백윤주 교수님

■ **담 당 조 교** 나상진 조교님

■ **실 험 날 짜** 2024-10-24(7 주차)

■ **제 출 일** 2024-10-29

● 목 표

1. Interrupt 방식을 활용한 GPIO 제어 및 UART 통신
2. 라이브러리 함수 사용법 숙지

● 실험 주의사항

1. 실험 장비들을 연결할 시에 **반드시 모든 전원을 끄고 연결해주세요**

2. 장비 반납 시 충격이 가해지지 않게 주의해서 넣어주세요

● 세부 실험 내용

1. Datasheet 및 Reference Manual 을 참고하여 해당 레지스터 및 주소에 대한 설정 이해

2. NVIC 와 EXTI 를 이용하여 GPIO 에 인터럽트 핸들링 세팅

(ISR 동작은 최대한 빨리 끝나야 함)

보드를 켜면 LED 물결 기능 유지 (LED 1->2->3->4->1->2->3->4->1->… 반복) A LED 물결 방향 변경 - 1->2->3->4 B LED 물결 방향 변경 - 4->3->2->1 (물결 속도는 delay 를 이용하여 천천히 동작, **ISR 에서는 delay 가 없어야 합니다**)

3. 버튼 1 : A 동작, 버튼 2 : B 동작

(조이스틱을 눌렸을 때 위 동작이 지연시간 없이 바로 이루어져야 합니다)

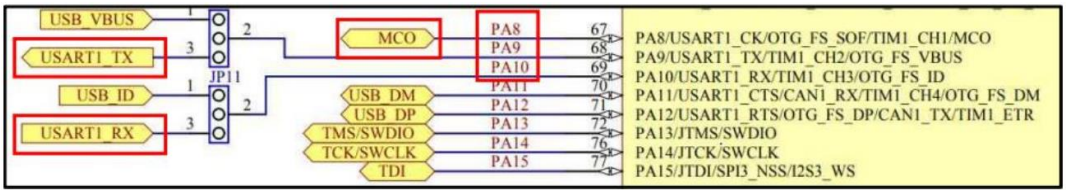
4. PC 의 Putty 에서 a, b 문자 입력하여 보드 제어 (PC -> 보드 명령)

('a' : A 동작, 'b' : B 동작)

5. 버튼 3 을 누를 경우 Putty 로 "TEAMXX.\r\n" 출력

실험 개요:

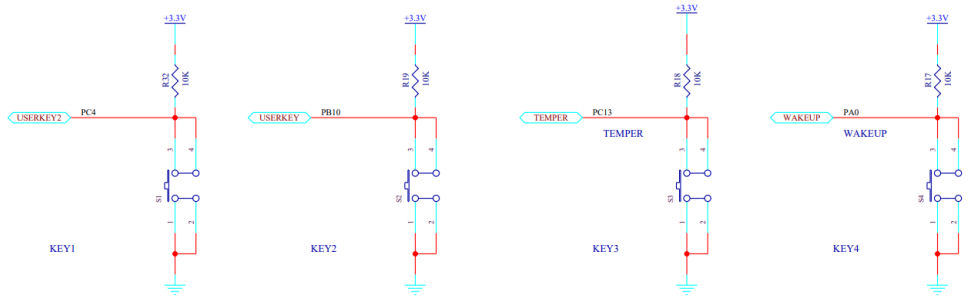
1) 시스템 클럭 및 주변 장치 클럭 설정 (RCC 설정)



<그림 2> USART1_TX, USART1_RX 에 할당된 핀

USART1_TX, USART1_RX 는 각각 PA9, PA10 에 할당되어 있으므로 Port A 에 clock 을 인가한다.

```
/* UART TX/RX port clock enable */
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
```



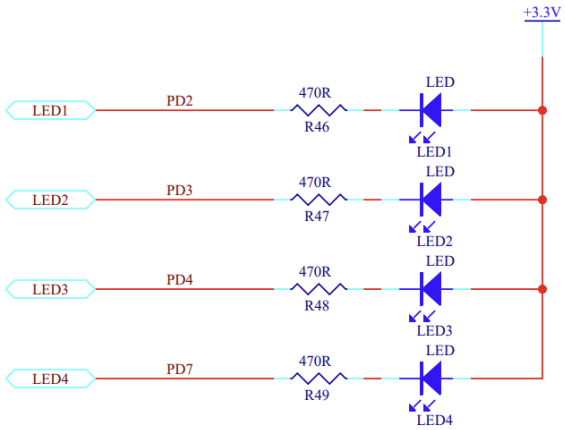
Button

<그림 3> Button 에 할당된 핀 3

PC4, PB10, PC13 에 할당된 버튼 1,2,3 을 사용하므로 Port B 와 Port C 를 enable 시킨다.

```
/* Button 1,2,3 port clock enable */
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
```

LED



<그림 4> LED 에 할당된 핀

마찬가지로 LED 는 각각 PD2, PD3, PD4, PD7 에 할당되어 있으므로 Port D 를 enable 시킨다.

```
/* LED port clock enable */
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);
```

USART1 clock 을 인가하고, EXTI 사용을 위해 AFIO 또한 enable 시킨다.

```
/* USART1 clock enable */
RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);
/* Alternate Function IO clock enable */
RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
```

2) **GPIO 핀 설정 (입력 및 출력 모드 설정)**

버튼을 입력으로 사용할 것이므로, mode 를 input 으로 설정하고 각각 4 번, 10 번, 13 번 핀 할당을 통해 GPIO 설정을 한다.

```
GPIO_InitStructure_btn13.GPIO_Pin = GPIO_Pin_4| GPIO_Pin_13;
GPIO_InitStructure_btn13.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure_btn13.GPIO_Mode = GPIO_Mode_IPU;

GPIO_Init(GPIOC, &GPIO_InitStructure_btn13);

GPIO_InitStructure_btn2.GPIO_Pin = GPIO_Pin_10;
GPIO_InitStructure_btn2.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure_btn2.GPIO_Mode = GPIO_Mode_IPU;
GPIO_Init(GPIOB, &GPIO_InitStructure_btn2);
```

GPIOD 포트의 핀 2, 3, 4, 7 을 푸시풀 출력 모드로 초기화하여 LED 에 전류를 안정적으로 공급하고, 프로그램이 핀을 제어해 LED 를 켜거나 끌 수 있습니다.

```
/* LED pin setting*/
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2 | GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_7;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_Init(GPIOD, &GPIO_InitStructure);
```

UART 통신을 위해 GPIOA 포트의 핀 9(TX)와 핀 10(RX)를 설정하는 코드입니다. TX 와 RX 핀을 설정하여 UART1 을 통해 데이터 송신 및 수신이 가능하도록 합니다.

```
/* UART pin setting */
```

```
//TX
GPIO_InitStructure_TX.GPIO_Pin = GPIO_Pin_9;
GPIO_InitStructure_TX.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure_TX.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_Init(GPIOA, &GPIO_InitStructure_TX);
//RX
GPIO_InitStructure_RX.GPIO_Pin = GPIO_Pin_10;
GPIO_InitStructure_RX.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure_RX.GPIO_Mode = GPIO_Mode_IPD;
GPIO_InitStructure_RX.GPIO_Mode = GPIO_Mode_IPU;
GPIO_Init(GPIOA, &GPIO_InitStructure_RX);
```

3) EXTI (외부 인터럽트) 설정

Button 1, 2, 3 각각의 인터럽트를 설정하고 있습니다. 각 버튼의 GPIO 포트와 핀 번호를 설정한 후, **하강 엣지(Falling Edge)에서 인터럽트를 트리거하도록 설정합니다. GPIO_EXTILineConfig 함수를 통해 각 버튼에 해당하는 GPIO 포트와 핀을 EXTI 라인에 연결하고, EXTI_Init 을 통해 EXTI 라인의 모드, 트리거, 활성화 설정을 적용합니다.**

```
/* Button 1 */
GPIO_EXTILineConfig(GPIO_PortSourceGPIOC, GPIO_PinSource4);
EXTI_InitStructure.EXTI_Line = EXTI_Line4;
EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
EXTI_InitStructure.EXTI_LineCmd = ENABLE;
EXTI_Init(&EXTI_InitStructure);
```

```
/* Button 2 */
GPIO_EXTILineConfig(GPIO_PortSourceGPIOB, GPIO_PinSource10);
    EXTI_InitStructure.EXTI_Line = EXTI_Line10;
    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
  EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
  EXTI_InitStructure.EXTI_LineCmd = ENABLE;
  EXTI_Init(&EXTI_InitStructure);

/* Button 3 */
    GPIO_EXTILineConfig(GPIO_PortSourceGPIOC, GPIO_PinSource13);
    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);
```

4) USART1 초기화

USART_Cmd 함수를 통해 USART1 을 enable 해주고, baudrate, parity bit, stop bit, hardware flow control, mode, word length 에 대해 각각 설정해줌으로써 USART1 을 초기화한다

```
        // USART1_InitStructure.USART_BaudRate = ???;
    USART1_InitStructure.USART_BaudRate = 9600;
    USART1_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
    USART1_InitStructure.USART_WordLength = USART_WordLength_8b;
    USART1_InitStructure.USART_StopBits = USART_StopBits_1;
    USART1_InitStructure.USART_Parity =  USART_Parity_No;
   USART1_InitStructure.USART_HardwareFlowControl =  USART_HardwareFlowControl_None;
// USART_Init(???);
    USART_Init(USART1, &USART1_InitStructure);
// USART_ITConfig(???);
    USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
```

5) NVIC 설정

이 라인은 NVIC 의 우선순위 그룹을 설정합니다. NVIC_PriorityGroup_0 은 프리엠션 우선순위만을 고려하고 서브 우선순위를 사용하지 않는 설정입니다. 이로 인해 각 인터럽트의 우선순위는 프리엠션 우선순위만으로 결정됩니다.

```
// NVIC_PriorityGroupConfig(???);
NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0);
```

이 코드 섹션에서는 각 버튼과 UART1 통신 인터럽트의 우선순위와 활성화 설정을 다룹니다. 모든 인터럽트를 PreemptionPriority = 0x00, SubPriority = 0x01 로 설정하여 높은 우선순위로 처리될 수 있도록 구성하였습니다.

```
// Button1
NVIC_InitStructure_Btn1.NVIC_IRQChannel = EXTI4_IRQn;
NVIC_InitStructure_Btn1.NVIC_IRQChannelPreemptionPriority = 0x00; // TODO
NVIC_InitStructure_Btn1.NVIC_IRQChannelSubPriority = 0x01; // TODO
NVIC_InitStructure_Btn1.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure_Btn1);
// Button2,3
NVIC_InitStructure_Btn2.NVIC_IRQChannel = EXTI15_10_IRQn;
NVIC_InitStructure_Btn2.NVIC_IRQChannelPreemptionPriority = 0x00; // TODO
NVIC_InitStructure_Btn2.NVIC_IRQChannelSubPriority = 0x01; // TODO
NVIC_InitStructure_Btn2.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure_Btn2);

// UART1
// 'NVIC_EnableIRQ' is only required for USART setting
NVIC_EnableIRQ(USART1_IRQn);
NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x00; // TODO
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x01; // TODO
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);
```

6) **USART1 IRQ handler 설정**

PC 의 Putty 에서 a, b 문자를 입력하여 보드를 제어한다. flag 변수를 통해 LED 물결의 방향을 제어할 수 있다.
입력이 'a'이면 flag 변수에 0 을 할당하여 LED 물결의 방향을 1->2->3->4 로 바꾼다.
입력이 'b'이면 flag 변수에 1 을 할당하여 LED 물결의 방향을 4->3->2->1 로 바꾼다.

```
void USART1_IRQHandler() {
    uint16_t word;
    if(USART_GetITStatus(USART1,USART_IT_RXNE)!=RESET){
        // the most recent received data by the USART1 peripheral
        word = USART_ReceiveData(USART1);

        // TODO implement
        if(word == 'a') {idx = 0;}
        else if (word == 'b') {idx = 1;}
        // clear 'Read data register not empty' flag
        USART_ClearITPendingBit(USART1,USART_IT_RXNE);
    }
}
```

7) **EXTI IRQ handler 설정**

이 핸들러는 버튼 2 또는 3 이 눌렸을 때 동작하며, 해당 이벤트 발생 시 idx 를 0 으로 초기화하고, 인터럽트 플래그를 클리어하여 다음 인터럽트가 발생할 수 있도록 합니다.

```
// TODO: Button 1 interrupt handler implement, referent above functio
void EXTI4_IRQHandler(void) { // when the Button 2,3 is pressed

    if (EXTI_GetITStatus(EXTI_Line4) != RESET) {
        if (GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_4) == Bit_RESET) {
            idx = 0;
        }
        EXTI_ClearITPendingBit(EXTI_Line4);
    }
    }
```

8) **LED 동작 구현**

이 코드의 무한 루프는 idx 에 따라 LED 점등 방향을 결정하여 순방향(0,1,2,3) 또는 역방향(3,2,1,0)으로 LED 를 점등합니다.

순방향: led++을 통해 arr 배열의 다음 인덱스로 이동하며, 이전 LED 를 끕니다.

역방향: led--를 통해 이전 인덱스로 이동하며, 범위를 벗어나면 마지막 인덱스(3)로 설정합니다.

Delay() 함수를 통해 딜레이를 추가하여 점등 간격을 조절합니다.

이 코드 구조는 효율적으로 LED 점등 방향을 전환하며, idx 값에 따라 다르게 동작하게 해 주는 간단하면서도 효과적인 방식입니다.

```
while (1) {
    // TODO: implement
  if (idx == 0){
    GPIO_SetBits(GPIOD, arr[led%=4]);
    led ++;
    GPIO_ResetBits(GPIOD, arr[led%=4]);
  }else{
    GPIO_SetBits(GPIOD, arr[led%=4]);
    led --;
    if(led < 0){led = 3;}
    GPIO_ResetBits(GPIOD, arr[led%=4]);
  }     // Delay
    Delay();
}
return 0;
```