Exercise #07 REPORT

Name: 송예린

Student ID: 2171023

1. **Exercise Objective (10%)**

일상생활 중 interrupt 가 발생하면 문제를 해결하고 다시 하던 일을 마저 한다. 소프트웨어에서도 interrupt 가 발생하면 동일한 방법으로 수행된다. 함수 호출과 달리 언제, 어디서나 interrupt 를 발생시킬 수 있고 interrupt 는 우선순위를 바꿔치기해 interrupt 가 먼저 수행되게 한다. 임베디드 시스템 상에서 interrupt 가 발생하는 코드를 짜고 실습해 볼 예정이다.

2. **Implementation (60%)**

S1, S2 를 input 으로 사용하기 위해서 p1 port 핀 번호 1,4 에 0 을 넣는다 Pull up 을 enable 할 핀 번호(1,4)에 1 을 넣어주고 p1 out 의 1,4 핀번호에 1 을 넣어서 pull up 을 사용한다는 것을 시스템에 알려준다. Pull up 은 버튼이 열려있을 때 1 을 가지고 있기 때문에 high 에서 low 로 변경될 때 interrupt 를 수행하게 한다면 스위치를 누를 때를 체크할 수 있다. 버튼이 눌릴 때 수행되기를 원하기 때문에 IES Register 해당 핀에 1 을 설정하고 p1 interrupt flags 를 모두 0 으로 만들어준다. P1 port 1, 4 에 enable signal 을 1 로 설정하고 p2 port 0,1,2 를 output 으로 사용할 수 있도록 설정해준다. P2 가 빨간 불을 키게 설정하고 INTISR[35]를 의미하는 곳에 1 을 넣어주어서 port1 interrupt 가 enable 할 수 있게 한다. 만약 s1,s2 의 버튼이 눌리는 interrupt 가 발생한다면 p1 interrupt flag 해당 핀 번호가 1 로 바뀌게 되고 함수에서 if 문을 통해 p2 의 led 색을 변경할 수 있다.

3. **Discussion & Conclusion (10%)**

While 문이 아닌 interrupt 를 이용해 스위치를 제어해보았다. Interrupt 가 발생하면 현재 주소를 저장하고 interrupt 를 해결 후 다시 저장한 주소로 돌아오는 식으로 구현된다

.

4. **Reference(s)**

5. **Code (20%)**

FILE NAME (ex: main.c)

```
#include "msp.h"


void PORT1_IRQHandler(){

    if(P1->IFG&BIT1){
       P2->OUT=BIT1;
    }
    P1->IFG&=~BIT1;
    if(P1->IFG&BIT4){
        P2->OUT=BIT2;
    }
    P1->IFG&=~BIT4;
}


void main(void)
{

    P1->DIR  = ~(uint8_t)(BIT1|BIT4);
    P1->REN  = (BIT1|BIT4);
    P1->OUT  = (BIT1|BIT4);

    P1->IES=(BIT1|BIT4);
    P1->IFG=0;
    P1->IE=(BIT1|BIT4);

    P2->DIR=BIT0|BIT1|BIT2;
    P2->OUT=BIT0;
    NVIC->ISER[1]=1<<3;
    while(1){}
}
```