Exercise #06 REPORT

Name: 송예린

Student ID: 2171023

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

SysTick Timer 은 STCSR register 에서 24bit 값을 감소시키면서 시간을 측정하는 타이머이다. SysTick Timer 를 이용하여 LED 의 속도를 조절할 수 있다.

2. **Implementation (60%)**

1) Systick Timer 을 이용해 P1 LED 가 빨간색 으로 꺼졌다 켜지는 실습(3 초 -2,5 초 - 2 초 -1.5 초-1.0 초-0.5 초 속도로 깜빡이고 다시 반복됨)

STCSR register 에 24bit 값을 넣고 CTRL 을 5 로 SysTick Timer 을 초기화한다. 빨간 불을 키기 위해 P1->DIR 을 1 로 설정해준다 P1->OUT 에 1 을 넣어서 빨간 불을 키고 SysTick_wait1000ms(3)로 3 초 딜레이를 준다. SysTick_wait1000ms(3) 함수는 SysTick_wait(unsigned int n) 함수를 3 번 반복하게 된다 SYSTEM CLOCK 은 3MHZ 인데 1 초동안 3*10^6 번 clock 이 반복되고 clock 이 한번 바뀔 때 333.333ns 초가 걸린다 333.333 초가 3*10^6 번 반복되면 1 초이므로 n 에 3000000 을 주어서 (SysTick_wait(3000000)) (3*10^6)-1 이 점점 감소되면서 1 초를 측정된다 다시 main 함수로 돌아와서 P1->OUT 에 0 을 넣어서 LED 를 끄고 SysTick_wait1000ms(3)로 3 초 딜레이를 준다. P1->OUT 에 1 을 넣어서 빨간 불을 키고 SysTick_wait1000ms(3)로 3 초 딜레이를 준 . P1->OUT 에 0 을 넣어서 LED 를 끄고 SysTick_wait1000ms(3)로 3 초 딜레이를 준다. 빨간 불을 키고 SysTick_wait1000ms(n)함수로 n 에 딜레이를 줄 시간을 넣어서 딜레이를 주고 다시 LED 를 끄고 딜레이를 주는 코드를 반복한다.

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

1 초 48*10^6 번 cycle clock 이 반복되는 CPU 가 있다면 clock 이 한번 바뀔 때 20.83333*10^(-9)초가 걸린다 이 초가 48*10^6 개 있으면 1 초가 된다. . 48*10^6 값이 1 씩 감소할 때 20.833ns 가 걸리므로 값이 0 이 되면 1 초가 되는 것이다. 이를 이용해 1 초를 측정하고 임베디드 시스템에 시간을 적용할 수 있다. 실습시간에 어려웠던 점은 알고리즘은 짰지만 다른 부분에서 에러가 나서 시간을 많이 소비하였는데 그 와중에 SysTick_wait1000ms(n) 함수를 1 초가 아닌 0.3 초로 잘못 생각하여 계산하였고 딜레이가 3 초보다 시간이 더 걸렸었다. 만약 초반에 작성한 코드에서 SysTick_wait(3000000)가 아닌 SysTick_wait(1000000)로 바꿨다면 이 또한 맞게 돌아갔을 것이다. 정확한 시간으로 임베디드 시스템을 조절할 수 있다는 점이 흥미로웠다.

4. Reference(s)

5. Code (20%)

FILE NAME (ex: main.c) (올바른 코드)

```
#include "msp.h"


/**
 * main.c
 */

void Systic_init(){
    SysTick->LOAD=0x00FFFFFF;
    SysTick->CTRL=0x00000005;
}


void SysTick_wait(unsigned int n){
    SysTick->LOAD=n-1;
    SysTick->VAL=0;
    while((SysTick->CTRL & 0x00010000)==0){
    }
}

void SysTick_wait10ms(unsigned int delay){
    int i;
    for(i=0; i<delay; i++){
        SysTick_wait(3000000);
    }
}

void main(void)
{
    int i;
    Systic_init();
    P1->DIR|=0x01;
    while(1){
    P1->OUT=0x01;
    SysTick_wait1000ms(3);
    P1->OUT=0;
    SysTick_wait1000ms(3);
    P1->OUT=0x01;
    SysTick_wait1000ms(2.5);
    P1->OUT=0;
    SysTick_wait1000ms(2.5);
    P1->OUT=0x01;
     SysTick_wait1000ms(2);
     P1->OUT=0;
     SysTick_wait1000ms(2);
     P1->OUT=0x01;
      SysTick_wait1000ms(1.5);
      P1->OUT=0;
      SysTick_wait1000ms(1.5);
```

```
        P1->OUT=0x01;
        SysTick_wait1000ms(1);
        P1->OUT=0;
        SysTick_wait1000ms(1);
    for(i=0;  i<9; i++){
        P1->OUT=0x01;
        SysTick_wait10ms(0.5);
        P1->OUT=0;
        SysTick_wait10ms(0.5);
    }


    }

        WDT_A->CTL  = WDT_A_CTL_PW  | WDT_A_CTL_HOLD;                 // stop watchdog  timer
}
```

FILE NAME (ex: main2.c) (틀린 부분)

```
    P1->OUT=0x01;
    SysTick_wait10ms(9);
    P1->OUT=0;
    SysTick_wait10ms(9);
    P1->OUT=0x01;
    SysTick_wait10ms(7.5);
    P1->OUT=0;
    SysTick_wait10ms(7.5);
    P1->OUT=0x01;
     SysTick_wait10ms(6);
     P1->OUT=0;
     SysTick_wait10ms(6);
     P1->OUT=0x01;
      SysTick_wait10ms(4.5);
      P1->OUT=0;
      SysTick_wait10ms(4.5);
      P1->OUT=0x01;
       SysTick_wait10ms(3);
       P1->OUT=0;
       SysTick_wait10ms(3);
    for(i=0;  i<9; i++){
        P1->OUT=0x01;
        SysTick_wait10ms(1.5);
        P1->OUT=0;
        SysTick_wait10ms(1.5);
    }
```

Code2(틀린 코드)에서 code 1(올바른 코드)로 바뀐 부분은 빨간색으로 표시했습니다.