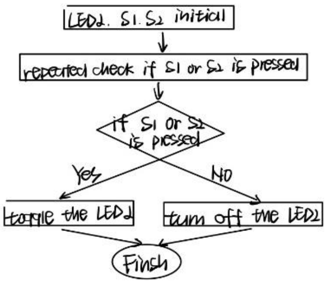
EE 215 Microprocessors LAB #6

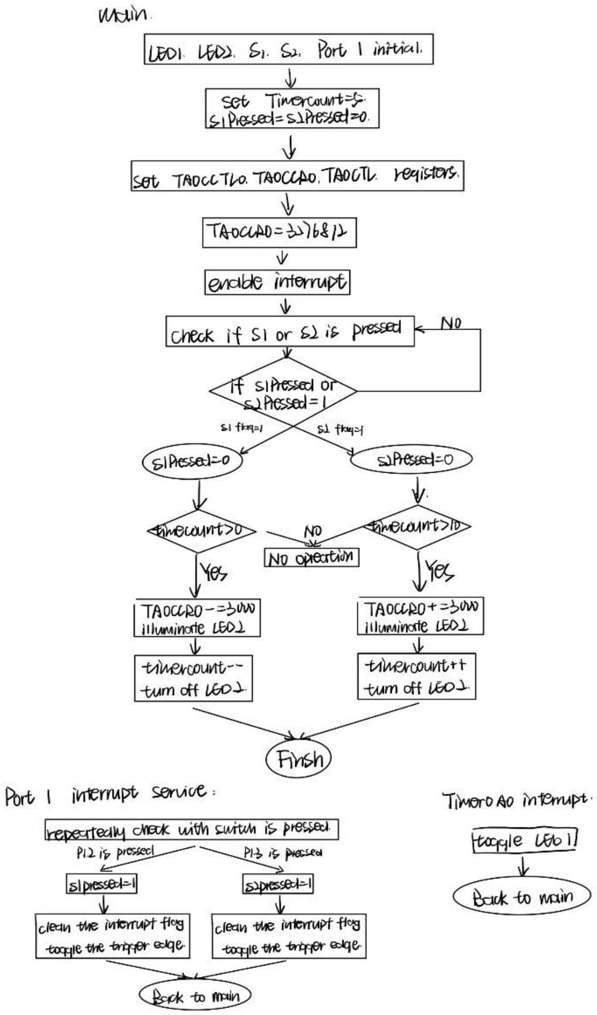
Student name：Li Xianzhe
Student ID number：2022214880

Flowchart

Lab6a



Lab6b



Code

Lab6a

```c
#include <msp430.h>

#define SWITCH_PIN1

#define CPU_F ((double)1000000)
#define delay_ms(x) __delay_cycles((long)(CPU_F*(double)x/1000.0))

int main(void) {
    WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer

    P3DIR |= BIT7;          // Set P3.7 as output for the LED
    P1DIR &= ~(BIT2+BIT3);      // Set P1.3 as input for the switch
    P1REN |= BIT2+BIT3;       // Enable pull-up resistor for P1.3
    P1OUT |= BIT2+BIT3;       // Set pull-up resistor as active for P1.3

    while(1)
    {
        if (((P1IN & BIT2) == 0)||((P1IN & BIT3) == 0)) // Switch is pressed
        {
            delay_ms(20); // Debounce delay
            if (((P1IN & BIT2) == 0)||((P1IN & BIT3) == 0))// Switch is still pressed after debounce, toggle LED state
            {
                P3OUT |= BIT7; // Toggle the LED state
                while (((P1IN & BIT2) == 0)||((P1IN & BIT3) == 0)); // Wait for the switch to be released
            }
        }
        else
            P3OUT &= ~BIT7;
    }
    return 0;
}
```

Lab6b

```c
#include <msp430.h>

#define CPU_F ((double)1000000)
```

```
#define delay_ms(x) __delay_cycles((long)(CPU_F*(double)x/1000.0))

#define LED1_PIN    BIT1
#define LED2_PIN    BIT7
#define S1_PIN      BIT2
#define S2_PIN      BIT3

volatile unsigned int timerCount = 5;
volatile unsigned char s1Pressed = 0;
volatile unsigned char s2Pressed = 0;

void main(void)
{
    WDTCTL = WDTPW | WDTHOLD;       // Stop the watchdog timer

      P1DIR &= ~(S1_PIN | S2_PIN);    // Set buttons as inputs
      P1REN |= S1_PIN | S2_PIN;        // Enable pull-up resistors
      P1OUT |= S1_PIN | S2_PIN;

      P8DIR |= LED1_PIN;              // Set LED1 as output
      P3DIR |= LED2_PIN;              // Set LED2 as output

      // Configure Timer A
      TA0CTL = TASSEL_1 | MC_1;       // ACLK, Up mode
      TA0CCR0 = 32768/2;               // 1 Hz at 32,768 Hz ACLK

      // Enable Timer A interrupt
      TA0CCTL0 |= CCIE;

    // Enable Port 1 interrupt for switches S1 and S2
    P1IE |= S1_PIN | S2_PIN;
    P1IES |= S1_PIN | S2_PIN;   // Set interrupt to trigger on
falling edge

    __enable_interrupt();       // Enable global interrupt

    while (1)
    {
        if (s1Pressed)   // Check if S1 is pressed
        {
            delay_ms(20);
            if ((P1IN & S1_PIN) == 0)
            {
                s1Pressed = 0;
            if (timerCount > 0)
            {
                TA0CCR0 += 3000;     // Decrease the frequency by a
reasonable amount
                P3OUT |= LED2_PIN;  // Illuminate LED2
                while((P1IN & S1_PIN) == 0);
                {
                    delay_ms(20);
                    while((P1IN & S1_PIN) == 0);
                    {
                        timerCount--;
                        delay_ms(120);
                        P3OUT &= ~LED2_PIN;// Turn off LED2
                    }
                }
            }
            }
        }

        if (s2Pressed)   // Check if S2 is pressed
        {
            delay_ms(20);
            if ((P1IN & S2_PIN) == 0)
            {
                s2Pressed = 0;
            if (timerCount < 10)
            {
                TA0CCR0 -= 3000;     // Increase the frequency by a
reasonable amount
                P3OUT |= LED2_PIN;  // Illuminate LED2
                while((P1IN & S2_PIN) == 0);
                {
                    delay_ms(20);
                    while((P1IN & S2_PIN) == 0);
                    {
                        timerCount++;
                        delay_ms(120);
                        P3OUT &= ~LED2_PIN;// Turn off LED2
                    }
                }
            }
            }
        }
    }
```

```
}

#pragma vector=PORT1_VECTOR
__interrupt void Port1_ISR(void)
{
    if (P1IFG & S1_PIN)
        {
            s1Pressed = 1;
            P1IFG &= ~S1_PIN;  // Clear the interrupt flag
            P1IES ^= S1_PIN; //toggle the trigger edge
        }

        if (P1IFG & S2_PIN)
        {
            s2Pressed = 1;
            P1IFG &= ~S2_PIN;  // Clear the interrupt flag
            P1IES ^= S2_PIN; //toggle the trigger edge
        }
}

#pragma vector=TIMER0_A0_VECTOR
__interrupt void Timer_A(void)
{
    P8OUT ^= LED1_PIN;             // Toggle LED1
}
```