EE 215 Microprocessors LAB #5

Student name：Li Xianzhe
Student ID number：2022214880

Explanation of Approach

QUESTIONS:

1.Explain what port control register commands are needed for an input switch.

ANSWERS:

a. Direction (DIR) Register (PxDIR):
PxDIR is used to set the direction of a port pin. To configure a pin as an input for a switch, you would clear the corresponding bit in the PxDIR register.
b. Pull-Up/Pull-Down Resistor Enable (REN) Register (PxREN):
To enable the pull-up or pull-down resistor for the switch, set the corresponding bit in the PxREN register.
c. Pull-Up/Pull-Down Resistor Selection (OUT) Register (PxOUT):
The PxOUT register is used to select the direction of the pull-up/pull-down resistor. For a pull-up configuration, set the corresponding bit; for a pull-down configuration, clear the bit.
d. Input (IN) Register (PxIN):
PxIN is used to read the status of the input pin. It allows you to check whether the switch is pressed or not by examining the state of the corresponding bit.

Code

Lab5a

```
#include <msp430.h>
#define CPU_F ((double)1000000) // Define CPU frequency for delay
calculation
#define delay_ms(x) __delay_cycles((long)(CPU_F * (double)x / 1000.0)) //
Delay function in milliseconds

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

    P8DIR |= BIT1; // Set P8.1 as output (LED)

    P1REN |= BIT2; // Enable pull-up resistor on P1.2 (Switch)
    P1OUT |= BIT2; // Set pull-up resistor as active for P1.2

    while (1)
    {
        if (P1IN & BIT2)// If the switch is not pressed, wait for debounce
and turn off the LED
```

```
        {

            delay_ms(20);
            if (P1IN & BIT2)
                P8OUT &= ~BIT1;
        }
        else// If the switch is pressed, wait for debounce and turn on the
LED
        {

            delay_ms(20);
            if ((P1IN & BIT2) == 0)
                P8OUT |= BIT1;
        }
    }

    return 0;
}
```

Lab5b

```
#include <msp430.h>

#define LED_PIN BIT7
#define SWITCH_PIN BIT3

#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 |= LED_PIN;          // Set P3.7 as output for the LED
    P1DIR &= ~SWITCH_PIN;       // Set P1.3 as input for the switch
    P1REN |= SWITCH_PIN;        // Enable pull-up resistor for P1.3
    P1OUT |= SWITCH_PIN;        // Set pull-up resistor as active for P1.3

    while(1)
    {
        if ((P1IN & SWITCH_PIN) == 0) // Switch is pressed
        {
            delay_ms(20); // Debounce delay
            if ((P1IN & SWITCH_PIN) == 0) // Switch is still pressed after
```

```
debounce, toggle LED state
            {
                P3OUT ^= LED_PIN; // Toggle the LED state
                while ((P1IN & SWITCH_PIN) == 0); // Wait for the switch to
be released
            }
        }
    }
    return 0;
}
```