Basic microcontroller (msp430) programming using C language

Lab02

Spring 2025

Submitted by: **Mohsin Sajjad**

Registration No: **22pwsce2149**

Class Section: **A**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Mohsin Sajjad

Student Signature: _______________

Submitted to:

Engr. Faheem Jan

Month Day, Year (02 03, 2025)

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

LAB No 2
Basic microcontroller (msp430) programming using C language

TASKS:

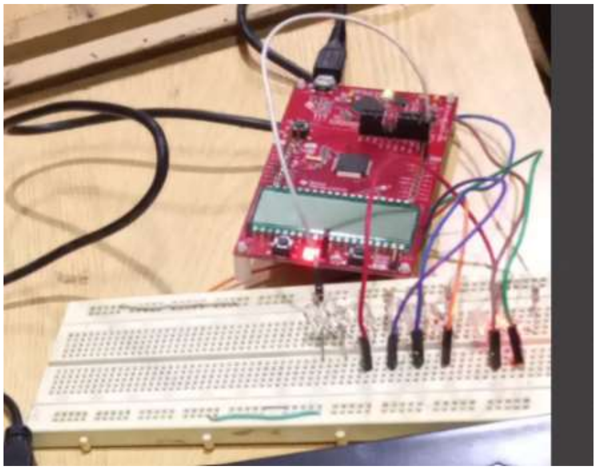
TASK1:

TASK 1 Write C program for Msp430 which toggle P1.0 or any other Pin of Msp430 MCU.

CODE:

```
#include <msp430.h>


/*

 * main.c

 */

int main(void) {

   WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer

   PM5CTL0 &= ~LOCKLPM5;      // Disable high-impedance mode to enable GPIOs

          P1DIR |= 0x01;            //set P1.0 to output direction

          for(;;){

                    volatile unsigned int i;

                    P1OUT ^= 0x01;    //toggle P1.0 using exclusive OR

                    i=10000;

                    do i--;      //do while loop for creating delay

                    while(i!=0);

          }

}
```

OUTPUT:



Conclusion:

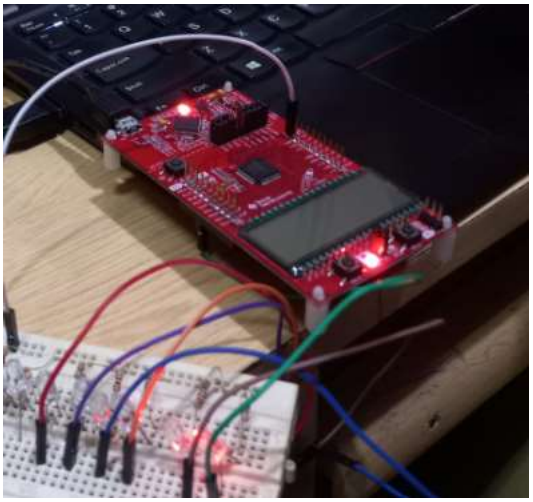
1. The code toggles P1.0 instead of P1.1 due to an incorrect bit mask.
2. It disables the watchdog timer and enables GPIOs.
3. Fix: Change `P1OUT ^= 0x01;` to `P1OUT ^= 0x02;` to toggle P1.1 correctly.

TASK2: Write C program which toggle the LEDS attached with P1.0 and P1.7 at the same time with different delays.

CODE:

```
#include <msp430.h>


int main(void) {
    WDTCTL = WDTPW | WDTHOLD;    // Stop watchdog timer
    PM5CTL0 &= ~LOCKLPM5;        // Disable high-impedance mode to enable GPIOs
    P1DIR |= 0x81;               // Set P1.0 and P1.7 as output


    for(;;) {
        volatile unsigned int i, j;


        P1OUT ^= 0x01;    // Toggle P1.0
        i = 10000;
        do i--;           // Delay after toggling P1.0
        while(i != 0);


        P1OUT ^= 0x80;    // Toggle P1.7
        j = 30000;
        do j--;           // Delay after toggling P1.7
        while(j != 0);
    }
}
```

OUTPUT:



Conclusion:

- The code toggles P1.0 and P1.7 on an MSP430 microcontroller.
- It disables the watchdog timer and enables GPIOs.
- Separate delay loops are used for toggling each pin with different delay durations.

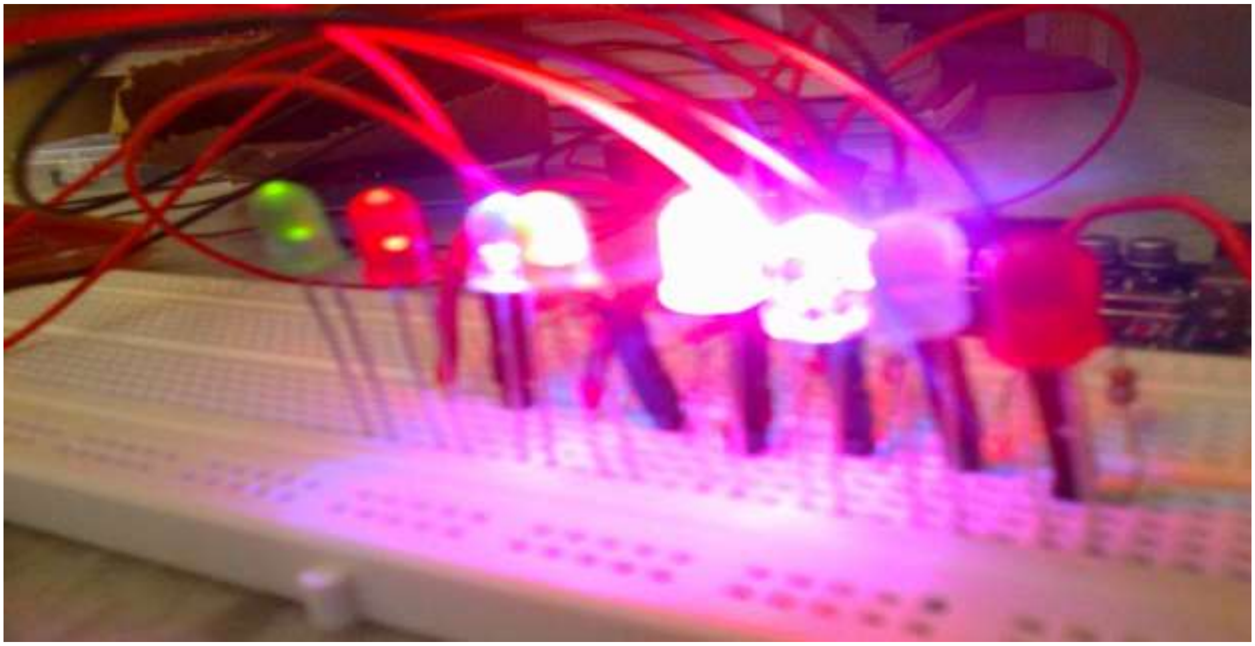
TASK3: Write C program which toggle all the LEDs attached with P1 or any other PORT

CODE:

```
#include <msp430.h>


/*
 * main.c
 */
```

```
int main(void) {
    WDTCTL = WDTPW | WDTHOLD;         // Stop watchdog timer
    PM5CTL0 &= ~LOCKLPM5;        // Disable high-impedance mode to enable GPIOs
            P1DIR |= 0xFF;               //set P1 to output direction
            for(;;){
                        volatile unsigned int i;
                        P1OUT ^= 0xFF;    //toggle P1 using exclusive OR
                        i=10000;
                        do i--;        //do while loop for creating delay
                        while(i!=0);
            }
}
```

OUTPUT:



Conclusion:

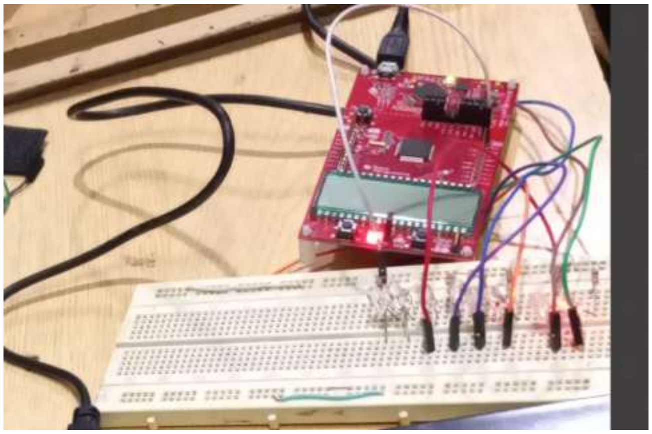
1. The code toggles all pins of Port 1 on an MSP430 microcontroller.
2. It disables the watchdog timer and enables GPIOs.
3. A delay loop controls the blink rate of all output pins.

TASK4: Display the pattern using C language 00000001 00000010 00000100 …. 10000000 00000001 00000010 Continuously

CODE:

```
#include <msp430.h>


/*

 * main.c

 */

int main(void) {

    WDTCTL = WDTPW | WDTHOLD;    // Stop watchdog timer

    PM5CTL0 &= ~LOCKLPM5;        // Disable high-impedance mode to enable GPIOs

    P1DIR |= 0xFF;               // Set all P1 pins as output


    unsigned char pattern = 0x01;  // Start from P1.0


    while(1) {

        P1OUT = pattern;          // Output the pattern

        __delay_cycles(100000);   // Delay for visibility


        pattern <<= 1;            // Shift left


        if(pattern == 0x00) {     // If pattern goes beyond P1.7, reset

            pattern = 0x01;

        }

    }
}
```

OUTPUT:



Conclusion:

1. The code shifts an LED pattern across all P1 pins on an MSP430.
2. It disables the watchdog timer and enables GPIOs.
3. A delay loop ensures visibility, and the pattern resets after reaching P1.7.