University of Texas at El Paso
Electrical and Computer Engineering Department

EE 3176 – Laboratory for Microprocessors I

Spring 2022

LAB 01

Basic Debugging and Troubleshooting Techniques

Goals:

- Using general purpose input output ports of a microcontroller
- Analyzing a C program using TI CCS integrated development environment (IDE) editor, compiler, and debugger.
- Stepping through code to determine an implemented algorithm.
- Observing the program control flow and data stored in hardware registers.

Pre Lab Questions:

- What is a microcontroller?
- How are microcontroller registers read or modified from a C program?
- How can you find syntax error in a program and fix them?
- How can you find logical errors in a program and fix them?
- Write 0 through 15 in hexadecimal and binary format.
- What are the truth tables for bitwise and (&), or (|), XOR (^) and complement (~)?

Basic Debugging and Troubleshooting Techniques Lab Guide

1. Run Code Composer Studio and create an empty C program project.
2. Replace the source code in your “main.c” file with the following program:

```
//******************************************************************************
//  MSP432P401 Demo - Software Poll Switches S1 and S2
//
//  Description:
//  ACLK = n/a, MCLK = SMCLK = default DCO
//
//              MSP432P401x
//            -----------------
//        /|\|              XIN|-
//         | |                 |
//         --|RST          XOUT|-
//     /|\   |                 |
//      --o--|P1.0         P1.0|-->LED
//     \|/
//
//   Dung Dang
//   Texas Instruments Inc.
//   June 2016 (updated) | November 2013 (created)
//   Built with CCSv6.1, IAR, Keil, GCC
//******************************************************************************
#include "msp.h"
int main(void){

    WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD; // Stop watchdog timer

    /* Configure GPIO */
    /* RGB LEDs at P2.0, P2.1, and P2.2, respectively */
    P2->DIR |= 0x07;                // All set as outputs
    P2->OUT &= ~0x07;
    P2->OUT |= BIT2;

    P1->DIR |= BIT0;                // Red Led at P1.0 set for output direction
    P1->OUT &= ~BIT0;

    /* Switch S1 */
    P1->DIR &= ~BIT1;               // Switch 1 at P1.1 set for input direction
    P1->REN |= BIT1;                // Enable Input Resistor
    P1->OUT |= BIT1;                // Set resistor to pull up

    /* Switch S2      */
    P1->DIR &= ~BIT4;
    P1->REN |= BIT4;
    P1->OUT |= BIT4;
```

```c
    P2->OUT &= ~BIT2;
    while (1) {
       P2->OUT |= BIT0;
       if (P1->IN & BIT1){
          P2->OUT &=~BIT0;
       }else {
          P2->OUT |= BIT0;         // Turn on LED if switch is pressed
       }
       P2->OUT |= BIT1;
       if(P1->IN & BIT4)
          P2->OUT &= ~BIT1;
       else
          P2->OUT |= BIT1;
          P2->OUT &= ~BIT1;
    }

}
```

3. Compile and run the program.
4. Provide a single sentence description of Launchpad behavior when the program is running.
5. Set a breakpoint at the second executable line of the while body and start debugging the program.
6. When the program stops at the breakpoint, use the “step over” debugger function to single-step through the while body. What function key invokes step over? ______ Step Into? ______ What is the difference between step over and step into within the while body?
7. What is the value of the P1->IN register when switch 2 is pressed and why is it set to that value?
8. Based on observed program behavior, add or modify comments to document the program so you can explain it to your T.A. Once you are ready, explain the program to your T.A.
9. Analyze the code and reduce its number of executable statements by 20% while keeping its functionality while executing without the debugger controlling it. Comment out unneeded lines and add comments to justify your modifications.
10. Create a flow chart to describe the algorithm implemented in the program.
11. Draw an updated schematic as program documentation to include all switches, LEDs, resistors, etc., labeled with their pin connection and function.
12. Once your program is running, use a voltmeter to measure the output voltages driving the output LEDs when the output is 0 and when the output is 1. Write in your lab report the actual voltage you measured for each LED.

Basic Debugging and Troubleshooting Techniques Using Measuring Instruments

13. Measure the voltage for zeros output to LED1 and the blue RGB LED. Change the program to send a 1 to each of these LEDs to read the voltage they are getting. Show your measurements to the T.A. and also write the values you obtained in each measurement in your lab report.

14. Use an oscilloscope to monitor the signals driving LED1 and the blue RGB LED. What are the driving voltages measured by the scope for each of these LEDs.

15. Include photographs of oscilloscope outputs for the LED1 and the blue RGB LED in your lab report.