CPE 325: Intro to Embedded Computer Systems

Lab03

Digital I/O on Experimenter's Board: LEDs and Switches

Submitted by: Michael Agnew

Date of Experiment: 1/29/2024

Report Deadline: 1/30/2024

Demonstration Deadline: 2/5/2024

Introduction:

This lab details the process by which one can interface with the MSP430 and manipulate the LEDs that are located on the board. This lab talks about how the MSP430 interfaces with other software and programs through its 8-bit parallel ports. The tutorial also lays out the framework for how one can program a code in C which can interact with the MSP430 and inform it to take certain actions specifically involving its LEDs in this case. Speaking of LEDs, this tutorial extensively discusses the process by which one can make the LEDs of this board blink which was a requirement for the assignment. Lastly, the tutorial goes over how to connect and interface with the switches that are on the MSP430. It is with these switches that the user is able to control the LEDs in the way that they were programmed.

Theory:

Debouncing: One of the biggest areas of theory inside of the lab that was done was switch interfacing and debouncing. Debouncing is a very important part of using switches on a circuit board. Debouncing is an intentional software delay implemented by the user which verifies that the switch is indeed pressed and not just noise created by the action of pressing the switch. This makes debouncing extremely important for interfacing switches on a circuit board such as MSP430. Debouncing is done by setting up a for loop to iterate the desired amount of times in order to delay the software enough to verify that the switch is, in fact, pressed.

Software Delay: Software delay, like debouncing, is another very important part of interfacing with circuit boards. Unlike debouncing, which is about switch interfacing, software delay deals with the LEDs that are located on the MSP430. Software delays in this lab are done by utilizing for loops. Each iteration of a for loop takes 10 clock cycles. With this being known, one can

multiply the amount of iterations of a for loop by 10 in order to delay for the desired amount of time. Each 1000 clock cycles is 1ms of delay, which can then be converted to hz. So, when the assignment that was given for this lab asks for 5hz, this can be done by using the __delay_cycles() function that was provided which for *n* given, there will be *n* ms in delay. So to reach 5hz, which is 200ms, one can simply think of 200,000. One needs to be careful however, because in order to blink the LED correctly, one needs to halve the amount of clock cycles in order to keep the LED off and on at the proper rate. So, if wanting to reach 5hz, one should input 100,000 into the __delay_cycles() function.

Problem 01:

What is the Problem?: The lab assignment at hand for this report was to interface the LEDs and switches that are included in the MSP430. This interfacing was done through Code Composer Studio and the C programming language. There was a "truth table" for how the board's LEDs were supposed to interact with the switches. This truth table was shown in the assignment:

Summary	LED1	LED2
No Press (Original State)	OFF	ON
SW1 is held	Blink at 4 Hz	OFF
SW2 is held	ON	Blink at 2 Hz
Both are held	Blink at 5 Hz	Blink at 5 Hz

Figure 1. Truth Table for Switch to LED Interaction.

Based on this table, the default state of the board was to have LED1, the red LED, be off and LED2, the green LED, be on. When the first switch is held, the red LED was meant to blink at 4

Hz while the green LED was meant to be off. When the second switch was held down, the red LED was meant to stay on at a constant rate and the green LED was meant to blink at 2 Hz. For the bonus assignment, which is the last row, it was meant that if the user holds down both switches, both of the LEDs would blink at 5 Hz alternatively. So, taking all of these requirements into consideration, the code was written and was made to perform all of these tasks.

How the Code Works: The code begins by defining the switches and which ports and bits that they will be using, along with defining which bits the red and green LEDs will associate with. After this, the red LED is set to be off while the green LED is set to be on by default. Afterwards, a loop counter is initialized to be used for the debouncing and delay, and the code is sent into an infinite loop. The code has to be in an infinite loop otherwise the LEDs will stop blinking eventually. The code then reaches an if-then-else statement which checks to see which switches are pressed, or if both switches are pressed. If switch one is the only one pressed, then it debounces the program, before turning off the green LED and blinking the red LED at 2 Hz. If switch two is the only one pressed, then it debounces the program, turns on the red LED, and blinks the green LED at 4 Hz. If neither of the switches are pressed, then the green LED is kept on and the red LED is kept off.

How is Debouncing Handled?: Debouncing is handled by a for loop which is set to run for 2000 iterations. This for loop starts an unsigned int i at 2000 and decrements it until it is equal to or less than 0. This loop, when calculating the time, comes out to $10 * (2000) = 20,000$ clock cycles; which is ~20ms. This debounce is done at the beginning of every section of code that controls the LEDs after the code deciphers which switch is being pressed.

Reaching the Correct Delay: One may reach the desired delay by using the __delay_cycles() function that was made available. This function allows one to set the exact amount of cycles that one would like to delay the program by. For example: if one would like to delay the number of

cycles by 60000, they could simply call __delay_cycles(60000);. This is the equivalent of making a for loop that loops for 6000 iterations. It is with this function that the correct Hz that was desired by the assignment criteria was reached.

Program Flowchart: The assignment requested that a flowchart be made for the written code. Therefore a flowchart was made that details the flow and logic of the program:

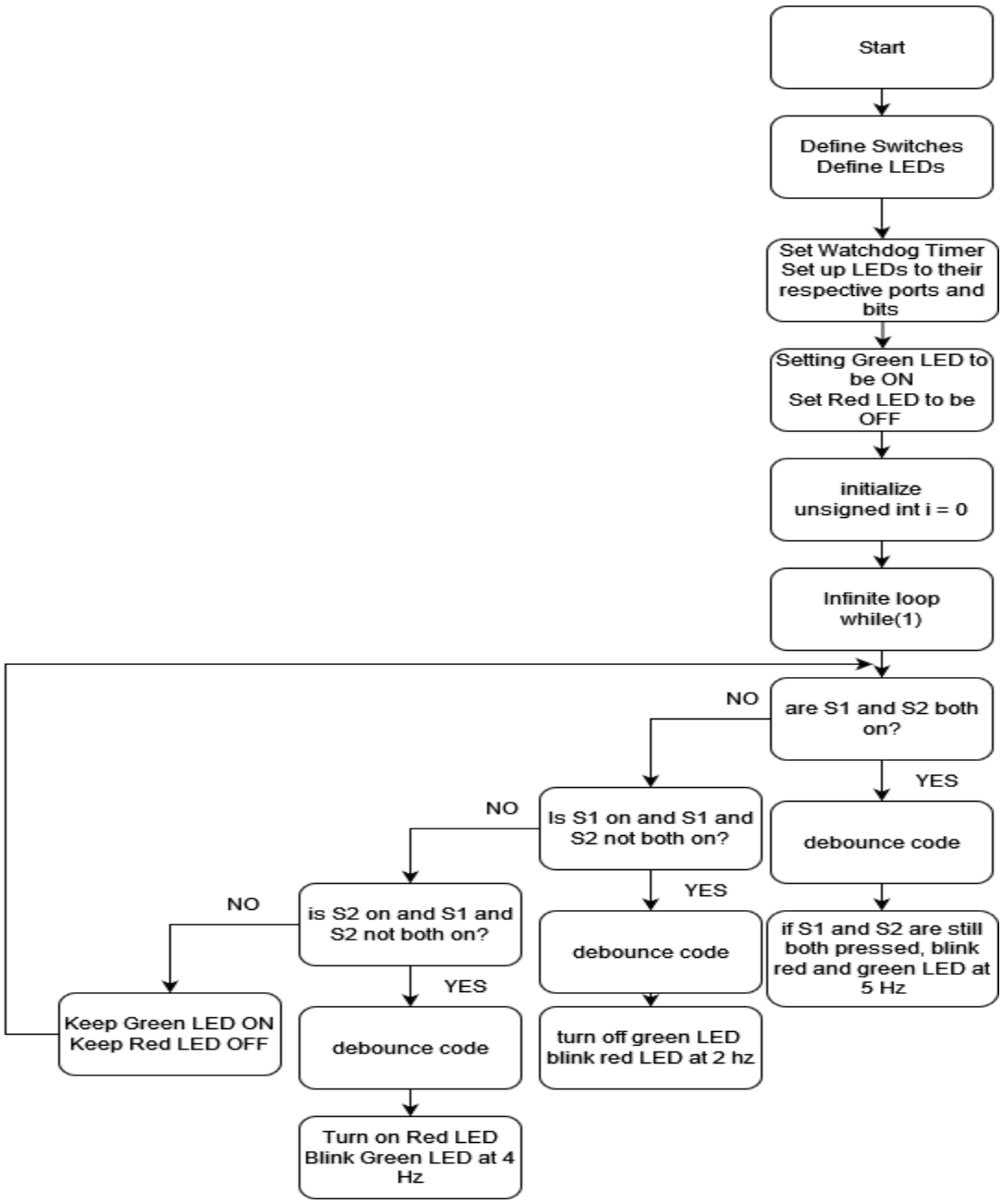

Figure 2. Flowchart for Program.

Calculations for Timing Delays: According to the lab tutorial that was given, a for loop can be used to find the correct time based on the number of iterations that it is given. The tutorial says that one iteration of a for loop takes 10 clock cycles. The tutorial says that the clock cycle time is $1\mu s$ where if you have 500,000 clock cycles, you can multiply it by $1\mu s$ to get 0.5s. Using this formula, one can conclude that for 100,000 clock cycles, it is 100ms. With this information, one can make the conversion from ms to Hz, $Hz = 1/s$. Therefore, when the assignment asked for 4 Hz, this can be converted to 250,000/2 (125ms) clock cycles; and for 2 Hz, it was 500,000/2 (250ms) clock cycles.

Bonus Problem:

The bonus problem for this code was an attachment for the main code. This was written by checking to see if switch 1 and 2 are on, and if so, debounce and delay for 5 Hz, (250,000 clock cycles) before blinking both the red and green LEDs. This successfully blinks the LEDs alternatively at 5 Hz, as requested by the assignment.

Results:

The results of this lab were that the LEDs were correctly able to be blinked at the proper Hz and all of the requested parameters of the assignment were fulfilled. Therefore, the lab experiment was a success and everything that was set out to be completed was done so successfully.

Conclusion:

Fortunately, there were no issues that were encountered over the course of the lab. There were a lot of important points and abilities that were learned during this lab assignment. Most notably, how to interface with the MSP430's ports and how to control the LEDs associated with the board. The assignment that was given gave the student the ability to learn exactly how to assign tasks to the LEDs and the correct ports associated with them in order to complete what the assignment laid out. Overall, learning how to assign ports, their designated bits and the LEDs in C, allowing them to be manipulated and put into effect by means of the MSP430's switches was completed.

Appendix:

Problem 01 (With Bonus):

```
/*-------------------------------------------------------------------------
 * File:        Lab3Code.c
 * Function:    Blinks LED lights on the MSP430 in the ways assigned.
 * Description: This code performs LED blinking and alternation based on the given
requirements of the assignment.
 * Input:       None
 * Output:      LEDs on the circuit board.
 * Author(s):   Michael Agnew, ma0133@uah.edu
 * Date:        Jan 29, 2024
 *-------------------------------------------------------------------------*/
#include <msp430.h>
```

```
#define S2 P1IN&BIT1
#define S1 P2IN&BIT1
#define GREENLED 0x80            // Mask for BIT7 = 1000_0000b
#define REDLED 0x01              // Mask for BIT0 = 0000_0001b

void main(void)
{
   WDTCTL = WDTPW + WDTHOLD;   // Stop watchdog timer
   P4DIR |= GREENLED;          // Set P4.7 to output direction
   P4OUT |= GREENLED;          // Set P4OUT to 1000_0000b (LED1 is ON)

   P1DIR |= REDLED;  // assign red led
   P1OUT &= ~REDLED; // make sure that it's off

   P2DIR &= ~BIT1;    // Set P2.1 as input for S1 input
   P2REN |= BIT1;     // Enable the pull-up resistor at P2.1
   P2OUT |= BIT1;     // Required for proper IO


   P1DIR &= ~BIT1;    // Set P1.1 as input for S2 input
   P1REN |= BIT1;     // Enable the pull-up resistor at P1.1
   P1OUT |= BIT1;     // Required for proper IO
```

```
unsigned int i = 0;


while(1) // infinite loop
{
   if (((S1) == 0) && ((S2) == 0)) // if switch 1 and 2 are pressed
   {


      for (i = 2000; i > 0; i--) // debounce


      if (((S1) == 0) && ((S2) == 0))
      {
         __delay_cycles(100000); // 5 hz delay


         P1OUT ^= REDLED;              // Toggle LED1
         P4OUT ^= GREENLED;            // Toggle LED2


      }


   }
   else if (((S1) == 0) && !(((S1) == 0) && ((S2) == 0))) // if the switch is pressed enter
   {
```

```
            for (i = 2000; i > 0; i--) // debounce ~20 ms


            P4OUT &= ~GREENLED; // turn LED2 off
            __delay_cycles(250000); // delay of 2 hz
            P1OUT ^= REDLED; // blink red LED
        }
        else if (((S2) == 0) && !(((S1) == 0) && ((S2) == 0))) // if switch 2 is pressed
        {
            for (i = 2000; i > 0; i--) // debounce


            P1OUT |= REDLED; // turn on red LED
            __delay_cycles(125000); // delay for 4 hz
            P4OUT ^= GREENLED; // blink green LED
        }
        else // if no buttons pressed (default state)
        {
            P4OUT |= GREENLED; // turn LED2 on
            P1OUT &= ~REDLED; // make sure that LED1 is off
        }
    }
}
```