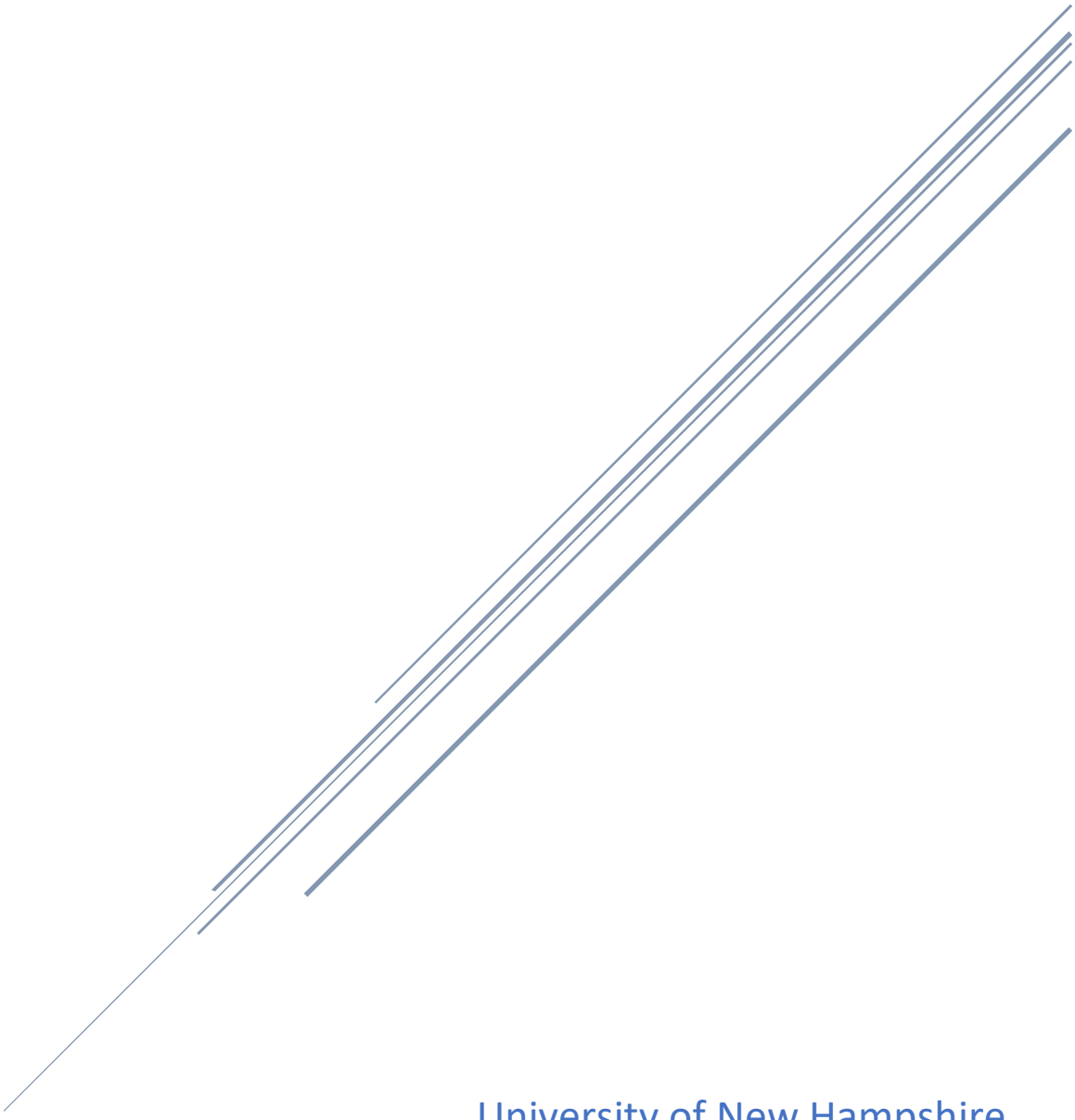
LAB 1: CONTROL LEDS AND PUSH BUTTONS

Nicholas Snyder



University of New Hampshire
ECE 649/796

Objectives:

The purpose of this laboratory was to serve as an introduction to the MSP-430 and TI's Code Composer Studio IDE. This was the fist time students were asked to create a new CCS project, write novel code, and implement said code onto the board. The scope of this assignment included bitwise logical operations and general C programming from a software side as well as mapping the input and output pins and ports from a more hardware perspective.

Background:

Leading up to this lab, the students have been learning about the foundational architecture of the MSP-430. This included the importance of declaring the correct register values for inputs and outputs. In-class exercises have been very helpful for learning the general flow of an embedded program.

Analysis:

The goal of this assignment was to use the two buttons on the board to change the frequency of two flashing LEDs on the board. The two LEDs could not be lit simultaneously and needed to flash at 1 Hz initially. The user could then press either of the pushbuttons to change the frequency to 2 or 4 Hz. This was to be done without using timers or interrupts.

Challenges:

Being the first-time using CCS and the MSP-430, many hurdles presented themselves while completing this assignment. At first, even uploading the code to the board correctly took a few tries. Later, adapting the example from lab0 left a couple of lines that have not been explained fully as of now such as the watchdog and enable pins. Having used all the inputs and output in previous exercises helped with configuring the port registers but still was not as smooth as hoped. Designing a delay function to flash the LEDs at the correct frequency was the most challenging. When first debugging the delay function it was hard to tell if the LEDs were flashing at all or too fast to perceive. Trying to switch between the two LEDs and making sure they were never on at the same time was not as challenging but still took time to get past.

Appendix:

```
#include <msp430.h>

#define GREEN_ON    0x80
#define GREEN_OFF   0x7F
#define RED_ON      0x01
#define RED_OFF     0xFE
#define BUTTON11    0x02
#define BUTTON12    0x04

#define DEVELOPMENT 0x5A80
#define ENABLE_PINS 0xFFFE

int i = 0;
```

```
/**
 * my_delay delays the program a number of milliseconds
 */
int my_delay(int length)
{
	int j;
	for (j = length * 20; j > 0; j--)
	{
		i++;
	}
	return 1;
}

/**
 * main.c
 */
int main(void)
{
	WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer
	PM5CTL0 = ENABLE_PINS;

	P9DIR = GREEN_ON;
	P9REN = 0x00;
	P9OUT = 0x00;

	P1DIR = RED_ON;
	P1REN = BUTTON11 | BUTTON12;
	P1OUT = BUTTON11 | BUTTON12;

	// initial delay is two seconds
	int length = 2000;

	// turn on green for one second
	P9OUT |= GREEN_ON;

	// alternate between red and green for remainder
	while (1)
	{

		// if button 1 is pressed
		if ((P1IN & BUTTON11) == 0)
		{
			length = 1000;
		}

		// if button 2 is pressed
		else if ((P1IN & BUTTON12) == 0)
		{
			length = 500;
		}

		// delay
		my_delay(length);

//		// turn off green
```

```
//        P9OUT ^= GREEN_ON;
//
//        // turn on red
//        P1OUT ^= RED_ON;

            // if red is off
            if ((P1OUT & RED_ON) == 0)
            {
                  // turn off green
                  P9OUT &= GREEN_OFF;

                  // turn on red
                  P1OUT |= RED_ON;
            }

            // if red is on
            else
            {
                  // turn red off
                  P1OUT &= RED_OFF;

                  // turn on green
                  P9OUT |= GREEN_ON;
            }
      }
}
```