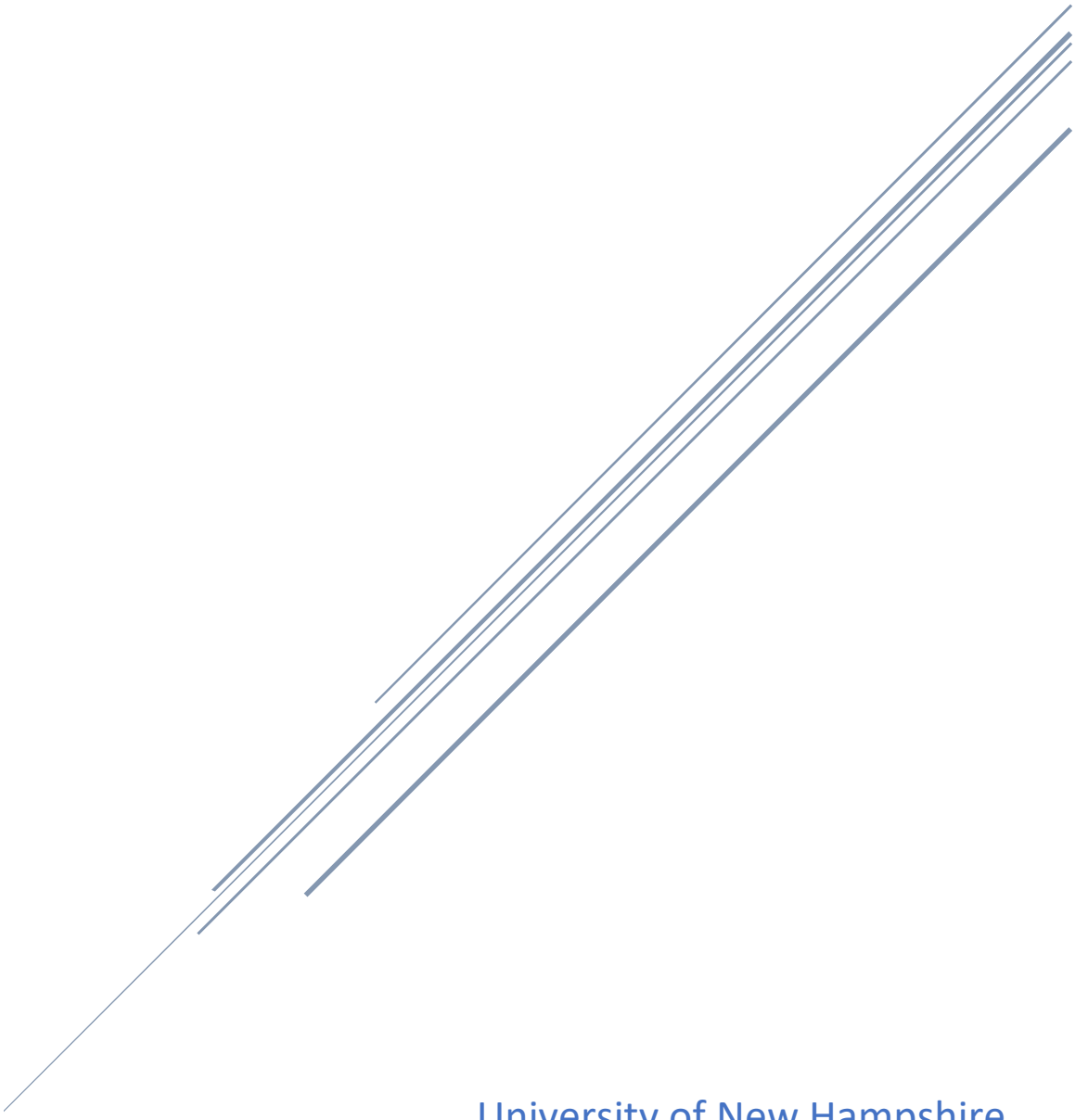
LAB 2: DISPLAY KEYPAD INPUT ON LCD

Nicholas Snyder



University of New Hampshire
ECE 649/796

Objectives:

The purpose of this lab is to get familiar with the capabilities of the liquid crystal display (LCD) on the TI Launchpad board as well as an external 4x4 keypad. This lab also involved the modification of library files, the ASCII table, and served as an extension to lab 1. The first of two tasks was to display one's name to the LCD at the press of a button and clear it after a certain time. The second and more complex task was to implement the keypad and relate the button pressed on the keypad to the matching character appearing on the LCD.

Background:

To prepare for this lab, the lectures included in-depth discussion about how LCDs function and scanning algorithms to interpret a 4x4 keypad. The lectures also strengthened knowledge on the MSP-430 architecture. Previously, my only exposure to LCDs was in simple Arduino projects that sent commands to the LCD using an I2C interface board.

Analysis:

For task 1, first the library files needed to be downloaded and moved into the main directory and then modified. The modifications included adding functionality for the whole alphabet and setting the correct input and output registers for the buttons. Due to this backend setup, the code written in main.c was very short. These included four calls to myLCD_showChar() that commanded the microcontroller to display a single character at some location on the LCD. Task 2 required the user to develop a scanning algorithm to interpret the scan codes coming from the keypad presses. This was done by checking only one column at a time in a loop and looking for a code that matched a known character. Using the keypad necessitated further edits to the library files to assign the correct inputs and outputs.

Challenges:

Not many challenges were encountered while completing task 1 pertained to the task itself. One error was that main.c could not find a few of the library file because they were in directories and not with main.c. That was remedied quickly by copying all the library files to the parent directory. Task 2 had more functionality which meant more things had to be correct for the code to work right. An ongoing problem I am having is that nothing is showing on the LCD when testing. I am not able to say for sure if the problem lies in the scanning algorithm or the library definitions.

Appendix:

main.c

```
<msp430.h>
<driverlib.h>  // Required for the LCD
"myGpioLab3.h" // Required for the LCD
"myClocks.h"   // Required for the LCD
"myLcdLab3.h"  // Required for the LCD
//Note, myLcd.c file has been changed to display * and #
//myGpio.c file has been changed as well, to configure the port direction, pull-down resistor, and
input/outport pins

/*
 * main.c
 */
int main(void)
```

```
{
    WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer

    initGPIO(); // Initializes General Purpose
    // Inputs and Outputs for LCD
    initClocks(); // Initialize clocks for LCD
    myLCD_init(); // Prepares LCD to receive commands

    int i = 0;

    while(1)
    {
        // if button 1 is pressed
        if (~P1IN & 0x02)
        {
            myLCD_showChar('N', 1);
            myLCD_showChar('I', 2);
            myLCD_showChar('C', 3);
            myLCD_showChar('K', 4);

            __delay_cycles(10000000);
        }

        // if button 2 is pressed
        if (~P1IN & 0x04)
        {
            myLCD_showChar(' ', 1);
            myLCD_showChar(' ', 2);
            myLCD_showChar(' ', 3);
            myLCD_showChar(' ', 4);
        }

        switch(i)
        {
            case 0:
                P9OUT = 0x01;

                if((P8IN & 0x80) == 0x80)
                {
                    while((P8IN & 0x80) == 0x80)
                    {
                        myLCD_showChar( '*', 3 );
                    }
                }
                else if((P8IN & 0x40) == 0x40)
                {
                    while((P8IN & 0x40) == 0x40)
                    {
                        myLCD_showChar( '7', 3 );
                    }
                }
                else if((P8IN & 0x20) == 0x20)
                {
                    while((P8IN & 0x20) == 0x20)
                    {
                        myLCD_showChar( '4', 3 );
                    }
                }
                else if((P8IN & 0x10) == 0x10)
                {
                    while((P8IN & 0x10) == 0x10)
                    {
                        myLCD_showChar( '1', 3 );
                    }
                }

                i++;
                break;
```

```
case 1:
    P9OUT = 0x02;

    if((P8IN & 0x80) == 0x80)
    {
        while((P8IN & 0x80) == 0x80)
        {
            myLCD_showChar( '2', 3 );
        }
    }
    else if((P8IN & 0x40) == 0x40)
    {
        while((P8IN & 0x40) == 0x40)
        {
            myLCD_showChar( '8', 3 );
        }
    }
    else if((P8IN & 0x20) == 0x20)
    {
        while((P8IN & 0x20) == 0x20)
        {
            myLCD_showChar( '5', 3 );
        }
    }
    else if((P8IN & 0x10) == 0x10)
    {
        while((P8IN & 0x10) == 0x10)
        {
            myLCD_showChar( '2', 3 );
        }
    }

    i++;
    break;

case 2:
    P9OUT = 0x20;

    if((P8IN & 0x80) == 0x80)
    {
        while((P8IN & 0x80) == 0x80)
        {
            myLCD_showChar( '3', 3 );
        }
    }
    else if((P8IN & 0x40) == 0x40)
    {
        while((P8IN & 0x40) == 0x40)
        {
            myLCD_showChar( '6', 3 );
        }
    }
    else if((P8IN & 0x20) == 0x20)
    {
        while((P8IN & 0x20) == 0x20)
        {
            myLCD_showChar( '9', 3 );
        }
    }
    else if((P8IN & 0x10) == 0x10)
    {
        while((P8IN & 0x10) == 0x10)
        {
            myLCD_showChar( '#', 3 );
        }
    }

    i++;
    break;
```

```
            case 3:
                P9OUT = 0x40;

                if((P8IN & 0x80) == 0x80)
                {
                    while((P8IN & 0x80) == 0x80)
                    {
                        myLCD_showChar( 'a', 3 );
                    }
                }
                else if((P8IN & 0x40) == 0x40)
                {
                    while((P8IN & 0x40) == 0x40)
                    {
                        myLCD_showChar( 'b', 3 );
                    }
                }
                else if((P8IN & 0x20) == 0x20)
                {
                    while((P8IN & 0x20) == 0x20)
                    {
                        myLCD_showChar( 'c', 3 );
                    }
                }
                else if((P8IN & 0x10) == 0x10)
                {
                    while((P8IN & 0x10) == 0x10)
                    {
                        myLCD_showChar( 'd', 3 );
                    }
                }

                i = 0;
                break;
        }
    }
}
```

myGpioLab3.c

```
// -------------------------------------------------------------------------
// myGpio.c  ('FR6989 Launchpad)
// -------------------------------------------------------------------------

#include <driverlib.h>
#include "myGpioLab3.h"

#define RED_ON              0x0001      // Enable and turn on the red LED
#define RED_OFF             0xFFFE      // Turn off the red LED
#define GREEN_ON                 0x0080          // Enable and turn on the green LED
#define GREEN_OFF                        0xFF00          // Turn off the green LED
#define DEVELOPMENT         0x5A80      // Stop the watchdog timer
#define ENABLE_PINS         0xFFFE      // Required to use inputs and outputs
#define BUTTON1                          0x0002          // P1.1 is button 1
#define BUTTON2                          0x0004          // P1.2 is button 2


//***************************************************************************
// Initialize GPIO
//***************************************************************************
void initGPIO(void)
{
        P1DIR = RED_ON;         // Red LED pin will be an output

        P1OUT &= RED_OFF;           // Start with red LED off
```

```
    P9DIR = GREEN_ON | 0x63;    // Green LED pin will be an output configuring 9.5, 9.6, 9.1, 9.0 as
outputs for keypad 0110 0011
    P9REN = 0x00;
    P9OUT &= GREEN_OFF;          // Start with green LED off

    P8DIR = 0xF0;               // Enabling port 8 pins as inputs
    P8REN = 0x00;               // Set upper 4bits of port 8 as inputs with pulldown
    P8OUT = 0x00;               // enable pull up/down resistor configuration for 8.7,8.6,8.5,8.4 1111
0000

    PM5CTL0 = ENABLE_PINS;      // Enable to turn on LEDs

        P1REN = 0x02;                           // Set P1.1 as input with pull-up
        P1OUT = 0x02;                           // enable pull up/down resistor configuration (for push
button 1)

        P1REN |= 0x04;                  // Set P1.2 as input with pull-up
        P1OUT |= 0x04;                  // enable pull up/down resistor configuration (for push button 2)

    // Set LFXT (low freq crystal pins) to crystal input (rather than GPIO)
    GPIO_setAsPeripheralModuleFunctionInputPin(
            GPIO_PORT_PJ,
            GPIO_PIN4 +                 // LFXIN  on PJ.4
            GPIO_PIN5 ,                 // LFXOUT on PJ.5
            GPIO_PRIMARY_MODULE_FUNCTION
    );
}
```

myLcdLab3.c (condensed for relevancy)

```
// LCD memory map for numeric digits
const char digit[10][2] =
{
    {0xFC, 0x28},  /* "0" LCD segments*/
    {0x60, 0x20},  /* "1" */
    {0xDB, 0x00},  /* "2" */
    {0xF1, 0x00},  /* "3" */
    {0x67, 0x00},  /* "4" */
    {0xB7, 0x00},  /* "5" */
    {0xBF, 0x00},  /* "6" */ // 1011 1111
    {0xE0, 0x00},  /* "7" */ // 1110 0000
    {0xFF, 0x00},  /* "8" */ // 1111 1111
    {0xF7, 0x00}   /* "9" */ // 1111 0111
};

// LCD memory map for uppercase letters
const char alphabetBig[26][2] =
{
    {0xEF, 0x00},  /* "A" LCD segments*/
    {0xF1, 0x50},  /* "B" */
    {0x9C, 0x00},  /* "C" */
    {0xF0, 0x50},  /* "D" */ // 1111 0000 0101 0000
    {0x9E, 0x00},  /* "E" */ // 1001 1110 0000 0000
    {0x8E, 0x00},  /* "F" */ // 1000 1110 0000 0000
    {0xBD, 0x00},  /* "G" */ // 1011 1101 0000 0000
    {0x6F, 0x00},  /* "H" */
    {0x90, 0x50},  /* "I" */
    {0x78, 0x00},  /* "J" */
    {0x0E, 0x22},  /* "K" */
    {0x1C, 0x00},  /* "L" */
    {0x6C, 0xA0},  /* "M" */ // 0110 1100 1010 0000
    {0x6C, 0x82},  /* "N" */ // 0110 1100 1000 0010
    {0xFC, 0x00},  /* "O" */ // 1111 1100 0000 0000
    {0xCF, 0x00},  /* "P" */
    {0xFC, 0x02},  /* "Q" */
    {0xCF, 0x02},  /* "R" */
    {0xB7, 0x00},  /* "S" */
    {0x80, 0x50},  /* "T" */
```

```
    {0x7C, 0x00},  /* "U" */
    {0x0C, 0x28},  /* "V" */
    {0x6C, 0x0A},  /* "W" */
    {0x00, 0xAA},  /* "X" */ // 0000 0000 1010 1010
    {0x00, 0xB0},  /* "Y" */ // 0000 0000 1011 0000
    {0x90, 0x28},  /* "Z" */ // 1001 0000 0010 1000
};
 const char star[1][2]={
         {0x03, 0xAA}
 };
 const char hash[1][2]={
         {0xFF, 0x50}
  };
```