EE 2361 Microcontrollers
Lab 3 Report: Multiplexing
Andrea Smith and Jason Nowotny

The first time we connected and programmed our device, it did not work. Our biggest struggle was properly reading in the keypad's input and decoding it to determine which key was pressed. Initially we attempted to write the KeyBoard.c bitmask with nested for loops, but that was extremely complex to debug. Instead, we created a helper function, tryColumns to test whether or not the keypress is different than the one before it. This made it much easier to complete the other bit shifting and logical operations necessary to display and hold the value of the keypress on the LED display. Many other simple bugs like having incorrect binary values in display_V001.c and minor wiring errors were resolved just by thoroughly checking values and connections. The fully commented code can be found below.

```c
/*
 * File:   lab3_main.c
 * Author: Owner
 *
 * Updated on March 1, 2020, 5:14 PM
 */


#include "xc.h"
#include "nowot005_displayheader_V001.h"
#include "nowot005_keyboardheader_V001.h"

//CW1: FLASH CONFIGURATION WORD 1

#pragma config ICS = PGx1       // Comm Channel Select
#pragma config FWDTEN = OFF     // Watchdog Timer Disable
#pragma config GWRP = OFF       // General Code Segment Write Protect (Writes to program memory are allowed)
#pragma config GCP = OFF        // General Code Segment COde Protect (Code protection is disabled)
#pragma config JTAGEN = OFF     // JTAG port enable (JTAG port is disabled)

//CW2: FLASH CONFIGURATION WORD 2

#pragma config I2C1SEL = PRI    // I2C1 Pin Location Select
#pragma config IOL1WAY = OFF    // IOLOCK Protection
#pragma config OSCIOFNC = ON    // Primary Oscillator I/O Function
#pragma config FCKSM = CSECME   // Clock Switching and Monitor
#pragma config FNOSC = FRCPLL   // Oscillator Select

#define PERIOD 5

void setup(void) {
    CLKDIVbits.RCDIV = 0;
    AD1PCFG = 0x9fff;
    init7seg();
    initKeyPad();
}

char map_func[4][4] = { //map of the keypad for indexing
    {'E','7','4','1'},
    {'0','8','5','2'},
    {'F','9','6','3'},
    {'D','C','B','A'}
};


int main(void) {
    setup();
    //L_Char and R_Char set to null
    char L_Char = '\0';
    char R_Char = '\0';
    int prevKey = -1;
    while(1){
        int newKey = readKeyPadRaw();
        if (newKey >= 0){
            if(newKey != prevKey){
                R_Char = map_func[(newKey/10)%10][newKey%10];
                //function for mapping the input of the right segment
                if (prevKey >= 0){
                    L_Char = map_func[(prevKey/10)%10][prevKey%10];
                    //function for mapping the input of the left segment
                }
                prevKey = newKey;
            }
        }

        if (R_Char != '\0'){ // Right charachter is sent to 7 seg display
            showChar7seg(R_Char, msb);
            delay(10); //10ms
        }
        if (L_Char != '\0'){ // Right charachter is sent to 7 seg display
            showChar7seg(L_Char, lsb);
            delay(10); //10ms
        }
    }
}
```

```c
/*
 * File:   Display
 * Author: Jason Nowotny, Andrea Smith
 * Comments:
 * Updated on March 1, 2020, 5:16 PM
 */

#include "xc.h"
#include "nowot005_displayheader_V001.h"

void init7seg(void);
void showChar7seg(char myChar, enum DIGIT myDigit);

int pattern(char Char){
  // switch statement to match the input charachter to an output for the display
    switch(Char)
    {
        case '0'://        ABCDEFG.
            return 0b0000000000001100;
        case '1':
            return 0b0000001001111100;
        case '2':
            return 0b0000000010010100;
        case '3':
            return 0b0000000000110100;
        case '4':
            return 0b0000001001100100;
        case '5':
            return 0b0000000100100100;
        case '6':
            return 0b0000000100000100;
        case '7':
            return 0b0000000001111100;
        case '8':
            return 0b0000000000000100;
        case '9':
            return 0b0000000000100100;
        case 'A':
            return 0b0000000001000100;
        case 'B':
            return 0b0000001100000100; //b
        case 'C':
            return 0b0000000110001100;
        case 'D':
            return 0b0000001000010100; //d
        case 'E':
            return 0b0000000110000100;
        case 'F':
            return 0b0000000111000100;
        case 's':
            return 0b0000001001010100; //scorpion
        case 'a':
            return 0b0000001111111100; // all off
    }
}

void init7seg(void) {
    TRISB &= 0b1111000000000011;
    LATB |=  0b0000111111111100; //turns off 7 seg LEDs
    LATB &=  0b1111111111111111; // turns off all displays
}

void showChar7seg(char myChar, enum DIGIT myDigit) {

    LATB &= CLEAR_DIGS_AND_SEGS_BIT_MASK; //located in header
    LATB = myDigit | pattern(myChar); //myDigit is located in header

}
```

Keyboard code:

```
01. //Written by: Jason Nowotny, Andrea Smith
02. //Updated on March 1, 2020, 5:20 PM
03. #include <p24FJ64GA002.h>
04. 
05. #include "xc.h"
06. #include "nowot005_keyboardheader_V001.h"
07. 
08. void initKeyPad(void);
09. unsigned int readKeyPadRaw(void);
10. 
11. void initKeyPad(void){
12.     TRISA |= 0x000F;//ra3-ra0 input
13.     TRISB &= 0x0FFF;//rb15-rb12 output
14.     CNPU1bits.CN2PUE = 1;
15.     CNPU1bits.CN3PUE = 1;
16.     CNPU2bits.CN29PUE = 1;
17.     CNPU2bits.CN30PUE = 1;
18. }
19. 
20. void delay(unsigned int milli)//Delay for milli in milliseconds
21. {
22.  int a;
23.     for (a = 0; a < milli; a++) {
24.         asm("repeat #15993");
25.         asm("nop");
26.     }
27.  return;
28. }
29. 
30. int tryColumns(int row){
31.     unsigned int bitmask = 0xFFF7;
32.     int jj;
33. 
34.     for(jj = 0; jj < 4; jj++){
35.         if ((PORTA | bitmask) < 0xFFFF){
36.             return jj;
37.         }
38.         bitmask = (bitmask >> 1);
39.         bitmask += 0x8000;
40.     }
41.     return -1;
42. }
43. /*try columns starts with the bitmask being
44.  * 0b1111 1111 1111 0111 for jj = 0
45.  * 0b1111 1111 1111 1011 for jj = 1
46.  * 0b1111 1111 1111 1101 for jj = 2
47.  * 0b1111 1111 1111 1110 for jj = 3
48.  *
49.  * this allows for us to check rb15-rb12 for the output from the keypad*/
50. 
51. unsigned int readKeyPadRaw(void) {
52.     unsigned int bitmask = 0x7FFF;
53.     int i;
54.     int result;
55.     static int previous = -1;
56. 
57.     //bottom to top
58.     for(i = 0; i < 4; i++) {
59.         LATB |= 0xF000;
60.         LATB &= bitmask;
61.         result = tryColumns(i);
62. 
63.         if (result != previous) {
64.             result = tryColumns(i);
65.             if (result >= 0) {
66.                 LATB &= 0x0FFF;
67.                 return (result*10) + i; // the reason we are doing this is for indexing reasons
68.             }
69.         }
70.         previous = result;
71.         bitmask = bitmask >> 1;
72.         bitmask += 0x8000;
73.     }
74.     return -1;
75. }
```

```
/*
 * File:   Display Header
 * Author: Jason Nowotny, Andrea Smith
 * Comments:
 * Updated on March 1, 2020, 5:18 PM
 */

// This is a guard condition so that contents of this file are not included
// more than once.
#ifndef nowot005_displayheader_V001_H
#define nowot005_displayheader_V001_H

#include <xc.h> // include processor files - each processor file is guarded.


#ifdef  __cplusplus
extern "C" {
#endif /* __cplusplus */

#define CLEAR_DIGS_AND_SEGS_BIT_MASK 0b0000000111111100// bitmask
//definitions
enum DIGIT{ // corresponds to a charachter on a certain side of the display
    msb = 0b0000100000000000, //Right
    lsb = 0b0000010000000000  //Left
};

void init7seg(void);
void showChar7seg(char myChar, enum DIGIT myDigit);

#ifdef  __cplusplus
}
#endif /* __cplusplus */

#endif  /* XC_HEADER_TEMPLATE_H */
```

```
/*
 * File:   nowot005_keyboardheader_V001.h
 * Author: Owner
 *
 * Updated on March 1, 2020, 5:22 PM
 */

#ifndef NOWOT005_KEYBOARDHEADER_V001_H
#define NOWOT005_KEYBOARDHEADER_V001_H

#ifdef  __cplusplus
extern "C" {
#endif
//definitions
void initKeyPad(void);
int tryColumns(int row);
void delay(unsigned int milli);
unsigned int readKeyPadRaw(void);


#ifdef  __cplusplus
}
#endif

#endif  /* NOWOT005_KEYBOARDHEADER_V001_H */
```