Class:	CPE 301L - 1001	Semester:	SPRING 2024

Points		Document author:	Abraham Garcia
		Author's email:	garci11@unlv.nevada.edu
		Document topic:	Postlab 8

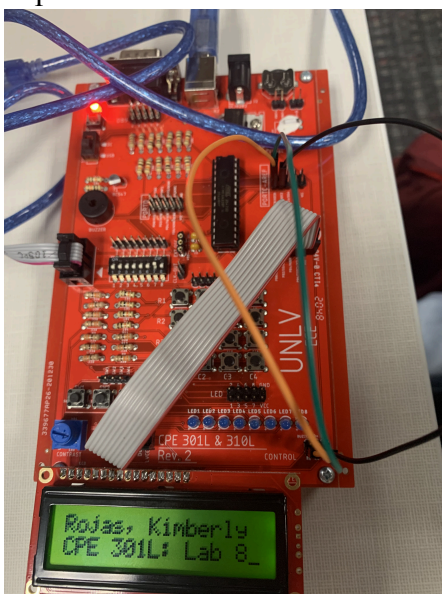
Instructor's comments:

1. Introduction / Theory of operation

This lab had us working with the microcontroller's LCD and push button matrix to display strings and data on the LCD either hard coded in or from the keypad/push button matrix. We first hard coded a string display message, then a string message when a push button in the matrix was pressed, and then when a code was inputted correctly.

2. Experiments

a. Experiment 1:

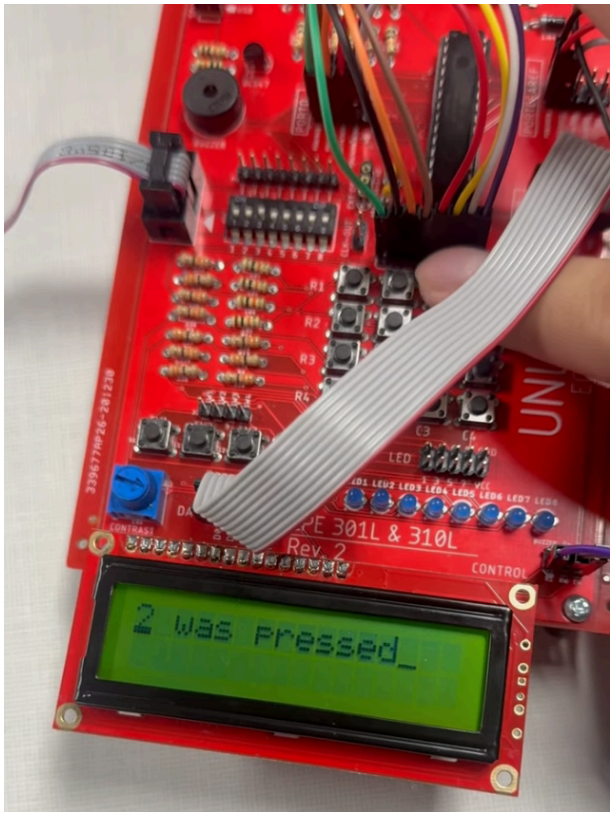



```
#define F_CPU 1000000UL
#include<avr/io.h>
#include<util/delay.h>
//definition for the wire connections
#define rs PC0
#define rw PC1
```

```
#define en PC2
#define data PORTB
//functions necessary
void lcd_init(); //to initialize lcd.
void lcd_cmd(char cmd_out); //to send command to lcd.
void lcd_data(char data_out); //to send data to lcd.
void lcd_str(const char *str); //to send string, basically stripping
each character and sending.

int main()
{
    //set DDR
    DDRB |= 0XFF; //set port b as output
    DDRC |= (1<<rs)| (1<<rw) | (1<<en);
    lcd_init(); // calls init commands
    lcd_str("Rojas, Kimberly");
    lcd_cmd(0xC0); // creates a new line for string
    lcd_str("CPE 301L: Lab 8");
    while(1)
    {}
}

//initializes the command
void lcd_init()
{

    lcd_cmd(0x38); // Initializing to 2 lines & 5x7 font.
    lcd_cmd(0xE);// Display on, cursor on
    lcd_cmd(0x1);// Clear LCD
    lcd_cmd(0x80);// Set cursor position to top row 0x80
}


void lcd_cmd(char cmd_out)
{
    data = cmd_out; //send the cmd_out to data
    PORTC &= ~(1<<rs);//set rs = 0, data is a cmd
    PORTC &= ~(1<<rw);//rw=0, we are writing
    PORTC |= (1<<en);//en = 1, lets lcd read cmd
    _delay_ms(10);//wait for small delay 10ms
```

```
    PORTC &= ~(1<<en);//set en =0, stops reading cmd
    _delay_ms(10);///wait for small delay 10ms
}

void lcd_data(char data_out)
{
    data = data_out;//send the data_out to data
    PORTC |= (1<<rs); // rs = 1 , we are given data not cmd
    PORTC &= ~(1 << rw); //rw = 0, we are writing
    PORTC |= (1<<en); // en = 1, lets lcd read data
    _delay_ms(10);//wait for small delay 10ms
    PORTC &= ~(1 << en); //en = 0, stops lcd from reading data
    _delay_ms(10);//wait for small delay 10ms
}

//reads out each character in the string individually
void lcd_str(const char *str)
{
    unsigned int i=0;
    while(str[i]!='\0')
    {
        lcd_data(str[i]);
        i++;
    }
}
```

b. Experiment2:



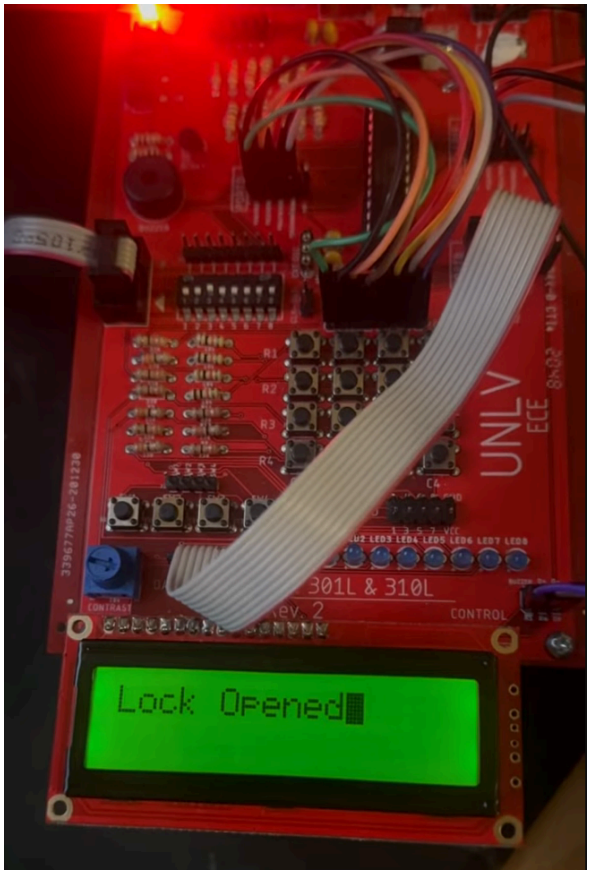

```c
#define F_CPU 16000000UL
#include<avr/io.h>
#include<util/delay.h>
//definition for the wire connections
#define rs PC0
#define rw PC1
#define en PC2
#define data PORTB
//functions necessary
void lcd_init(); //to initialize lcd.
void lcd_cmd(char cmd_out); //to send command to lcd.
void lcd_data(char data_out); //to send data to lcd.
void lcd_str(const char *str); //to send string, basically stripping
each character and sending.
void check();

int main()
{
    //set DDR
    DDRD = 0x0F;
    DDRB |= 0xFF; //set port b as output
```

```
    PORTB = 0x00;
    DDRC |= (1<<rs)| (1<<rw) | (1<<en);
    PORTC = 0xFF;
    while(1)
    {
        lcd_init(); // calls init commands
        _delay_ms(5);//wait for small delay 5ms
        check();
        _delay_ms(5);//wait for small delay 5ms
    }
}

//initializes the command
void lcd_init()
{
    lcd_cmd(0x38); // Initializing to 2 lines & 5x7 font.
    lcd_cmd(0x0F);// Display on, cursor on E
    lcd_cmd(0x01);// Clear LCD
    lcd_cmd(0x80);// Set cursor position to top row 0x80
}

void lcd_cmd(char cmd_out)
{
    data = cmd_out; //send the cmd_out to data

    PORTC &= ~(1<<rs);//set rs = 0, data is a cmd
    PORTC &= ~(1<<rw);//rw=0, we are writing
    PORTC |= (1<<en);//en = 1, lets lcd read cmd

    _delay_ms(10);//wait for small delay 10ms
    PORTC &= ~(1<<en);//set en =0, stops reading cmd
    _delay_ms(10);///wait for small delay 10ms
}

void lcd_data(char data_out)
{
    data = data_out;//send the data_out to data

    PORTC |= (1<<rs); // rs = 1 , we are given data not cmd
    PORTC &= ~(1 << rw); //rw = 0, we are writing
```

```
    PORTC |= (1<<en); // en = 1, lets lcd read data

    _delay_ms(10);//wait for small delay 10ms
    PORTC &= ~(1 << en); //en = 0, stops lcd from reading data
    PORTC &= ~(1 << rs); //en = 0, stops lcd from reading data
    _delay_ms(10);//wait for small delay 10ms
}

//reads out each character in the string individually
void lcd_str(const char *str)
{
    unsigned int i=0;
    while(str[i]!='\0')
    {
        lcd_data(str[i]);
        i++;
    }
}

//string lcd for each push button in 4x4 matrix
void check(void)
{
    PORTD = 0b00000001; //col 1
    if ((PIND & 0b11110000) == 0b00010000)
        lcd_str("A was pressed");
    else if ((PIND & 0b11110000) == 0b00100000)
        lcd_str("4 was pressed");
    else if ((PIND & 0b11110000) == 0b01000000)
        lcd_str("7 was pressed");
    else if ((PIND & 0b11110000) == 0b10000000)
        lcd_str("* was pressed");

    PORTD = 0b00000010; //col2
    if((PIND & 0b11110000) == 0b00010000)
        lcd_str("1 was pressed");
    else if ((PIND & 0b11110000) == 0b00100000)
        lcd_str("5 was pressed");
    else if ((PIND & 0b11110000) == 0b01000000)
        lcd_str("8 was pressed");
    else if ((PIND & 0b11110000) == 0b10000000)
```

```
        lcd_str("0 was pressed");

    PORTD = 0b00000100; //col3
    if ((PIND & 0b11110000) == 0b00010000)
        lcd_str("2 was pressed");
    else if ((PIND & 0b11110000) == 0b00100000)
        lcd_str("6 was pressed");
    else if ((PIND & 0b11110000) == 0b01000000)
        lcd_str("9 was pressed");
    else if ((PIND & 0b11110000) == 0b10000000)
        lcd_str("# was pressed");

    PORTD = 0b00001000; //col4
    if((PIND & 0b11110000) == 0b00010000)
        lcd_str("3 was pressed");
    else if ((PIND & 0b11110000) == 0b00100000)
        lcd_str("B was pressed");
    else if ((PIND & 0b11110000) == 0b01000000)
        lcd_str("C was pressed");
    else if ((PIND & 0b11110000) == 0b10000000)
        lcd_str("D was pressed");
}
```

c. Experiment 3:




```c
#define F_CPU 16000000UL
#include <stdio.h>
#include <string.h>
#include<avr/io.h>
#include<util/delay.h>

//definition for the wire connections
#define rs PC0
#define rw PC1
#define en PC2
#define data PORTB

//functions necessary
void lcd_init();        //to initialize lcd.
void lcd_cmd(char cmd_out);     //to send command to lcd.
void lcd_data(char data_out);   //to send data to lcd.
void lcd_str(const char *str);  //to send string, basically
stripping each character and sending.
```

```
void lcd_key(char keyin);   // user input and calls comparator
function
void seqcompare();      // comparator function
void check();           // connects buttons pushed to each letter,
number, and symbol

//variables used
char sequence[6] = "4214D";     //password to our secret crabby
patty formula
char seqin[6] = "";     // user input
int digit = 0;          // comparison digit
char keyin = ' ';       // variable to compare

int main()
{

    DDRD = 0x0F;                //sets lower 4 bits to output and
upper 4 to input
    DDRB |= 0xFF;               //set port b as output
    PORTB = 0x00;               //initialize to all logic low
    DDRC |= (1<<rs)| (1<<rw) | (1<<en); // intialized from beginning
    PORTC = 0xFF;               //sets all pin c to logic high

    lcd_init();                             // calls init commands
    while(1)
    {
        check();            // performs lock combo operation
    }
}

//initializes the command
void lcd_init()
{
    lcd_cmd(0x38); // Initializing to 2 lines & 5x7 font.
    lcd_cmd(0x0F);// Display on, cursor on E
    lcd_cmd(0x01);// Clear LCD
    lcd_cmd(0x80);// Set cursor position to top row 0x80
}

void lcd_cmd(char cmd_out)
```

```
{
    data = cmd_out; //send the cmd_out to data

    PORTC &= ~(1<<rs);//set rs = 0, data is a cmd
    PORTC &= ~(1<<rw);//rw=0, we are writing
    PORTC |= (1<<en);//en = 1, lets lcd read cmd

    _delay_ms(10);//wait for small delay 10ms
    PORTC &= ~(1<<en);//set en =0, stops reading cmd
    _delay_ms(10);///wait for small delay 10ms
}

void lcd_data(char data_out)
{
    data = data_out;//send the data_out to data

    PORTC |= (1<<rs); // rs = 1 , we are given data not cmd
    PORTC &= ~(1 << rw); //rw = 0, we are writing
    PORTC |= (1<<en); // en = 1, lets lcd read data

    _delay_ms(10);//wait for small delay 10ms
    PORTC &= ~(1 << en); //en = 0, stops lcd from reading data
    PORTC &= ~(1 << rs); //en = 0, stops lcd from reading data
    _delay_ms(10);//wait for small delay 10ms
}

//reads out each character in the string individually
void lcd_str(const char *str)
{
    unsigned int i=0;
    while(str[i]!='\0')
    {
        lcd_data(str[i]);
        i++;
    }
}

void lcd_key(char keyin)
{
    seqin[digit] = keyin;       //sets the user input into an array
```

```
    char str[2];            //initializes the char array
    str[0] = keyin;         //inputs into position 0
    str[1] = '\0';          //inputs into position 1
    lcd_str(str);           //goes through the whole string
    if (seqin[digit] == 'D')    //if user enters done then check the
combo
        seqcompare();       //send to comparason function
    digit++;            //increment for iteration
}

void seqcompare(void)
{
    if (strncmp(seqin, sequence, 5) == 0)   //compares first 5
characters in string
    {   //if equal
        lcd_cmd(0x01);  //clear display
        lcd_str("Lock Opened");//output success message
    }
    else
    {
        lcd_cmd(0x01);          //clear display
        lcd_str("Incorrect Code");  //output failure message
    }
}

void check(void)
{
    PORTD = 0b00000001; //sets portD to 1
    if ((PIND & 0b11110000) == 0b00010000) //checks for button a
    lcd_key('A');
    else if ((PIND & 0b11110000) == 0b00100000) //checks for button
4
    lcd_key('4');
    else if ((PIND & 0b11110000) == 0b01000000) //checks for button
7
    lcd_key('7');
    else if ((PIND & 0b11110000) == 0b10000000) //checks for button
*
    lcd_key('*');
```

```
    PORTD = 0b00000010;
    if((PIND & 0b11110000) == 0b00010000) //checks for button 1
    lcd_key('1');
    else if ((PIND & 0b11110000) == 0b00100000) //checks for button
5
    lcd_key('5');
    else if ((PIND & 0b11110000) == 0b01000000) //checks for button
8
    lcd_key('8');
    else if ((PIND & 0b11110000) == 0b10000000) //checks for button
0
    lcd_key('0');

    PORTD = 0b00000100;
    if ((PIND & 0b11110000) == 0b00010000) //checks for button 2
    lcd_key('2');
    else if ((PIND & 0b11110000) == 0b00100000) //checks for button
6
    lcd_key('6');
    else if ((PIND & 0b11110000) == 0b01000000) //checks for button
9
    lcd_key('9');
    else if ((PIND & 0b11110000) == 0b10000000) //checks for button
#
    lcd_key('#');

    PORTD = 0b00001000;
    if((PIND & 0b11110000) == 0b00010000) //checks for button 3
    lcd_key('3');
    else if ((PIND & 0b11110000) == 0b00100000) //checks for button
B
    lcd_key('B');
    else if ((PIND & 0b11110000) == 0b01000000) //checks for button
C
    lcd_key('C');
    else if ((PIND & 0b11110000) == 0b10000000) //checks for button
D
    lcd_key('D');

    _delay_ms(20);
```

```
}
```

3. Questions

a. We connected the data connections (D4-D7) to PORTB for the visual display. We connected RS, RW, and EN to PORTC's PC0, PC1, PC2 respectively.
b. The keypad is organized in a 4x4 matrix similar to an array matrix in c++ in how first you would need to access the column then the row.
c. The difference between 16x2 and 40x2 is the amount of char symbols, font size, and lines that can be displayed. The16x2 has 16 rows and 2 columns and the 40x2 displays 40 rows and 2 columns.

4. Conclusions

We had difficulty understanding how to get the push button working correctly as at first it wouldn't work so I tested it on the dip switch above the 4x4 push button matrix and it worked. I changed how I masked the input and it eventually worked.