Lab 5
Jacob Hillebrand
CEE-345 Microprocessor System Design
LCD Interfacing Lab

This lab was our final exploratory lab with the Atmelmega8515, and we used it to interface with an LCD module. This assignment was written in C.
The code loaded the board's registers, set up PORTC and PORTA (for LCD control and data respectively), then initialized the LCD and set up the registers, then used a while loop to repeatedly refresh the display and continuously write "I Love Processor System Design." The code for this program is shown on the next pages.

```
//CEE-345
//THIS EXAMPLE IS TO INTERFACING LCD TO THE ATMEGA8515 MICROCONTROLOR
//to a LCD built-in controller/processor (SPLC780D or ST7066U)
//PORTA IS CONNECTED TO THE LCD DATA
//PORTC IS CONNECTED TO LCD COMMAND(RS, R/W, E PINS)


#include <avr/io.h>
#define F_CPU 4000000UL

//This will include the matching header file for your
//AVR micro automatically

#include <util/delay.h>

#define lcd_dprt PORTA  //LCD Data Port
#define lcd_dddr DDRA   //LCD Data Register
#define lcd_dpin PINA   //LCD data Pin input to LCD
#define lcd_cprt PORTC  //LCD command port
#define lcd_cddr DDRC   //LCD command Register
#define lcd_cpin PINC   //LCD command input port to LCD
#define lcd_rs 0        //LCD Register Select (rs)
#define lcd_rw 1        //LCD Read/Write (rw)
#define lcd_en 2        //LCD Enable (en)
```

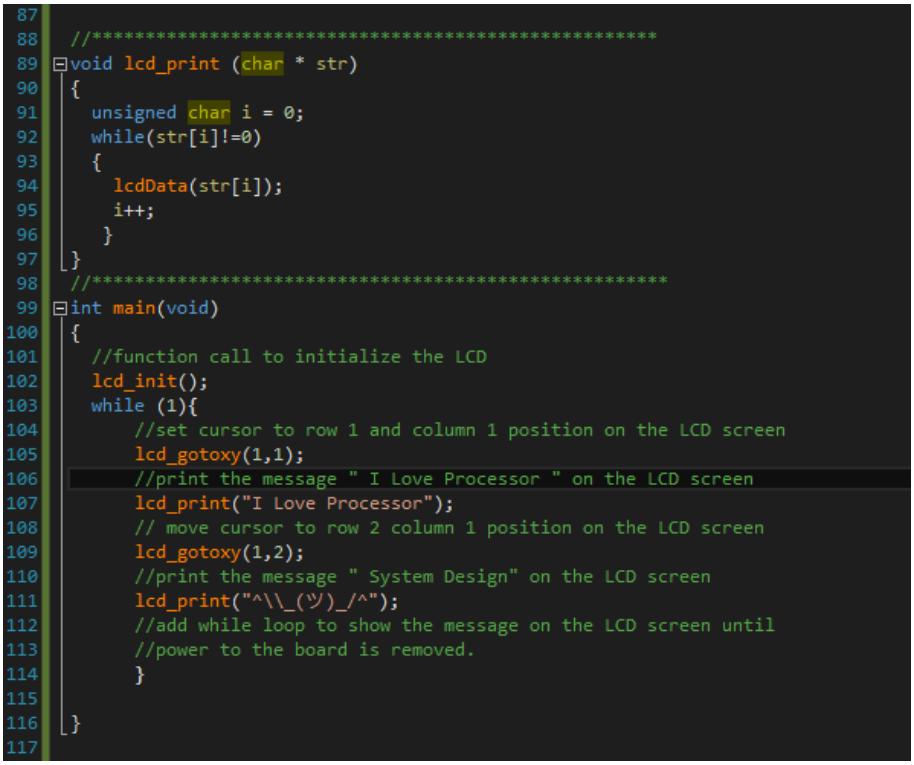
Figure 1: First Snippet

```
void lcdCommand(unsigned char cmnd) //send command to data/command registers
{
    lcd_dprt = cmnd;    //send command in rs mode

    //lcd_cprt = lcd_cprt | (1<<lcd_en);
    lcd_cprt = 0b00000100; // bit 2 (en) ='1', bit 1 (r/w)='0'(write); bit 0 (rs)='0'(command)

    _delay_us(1);

    //lcd_cprt = lcd_cprt & ~ (1<<lcd_en);

    lcd_cprt = 0b00000000; //make EN pin to stay 'high' longer

    _delay_us(100);
}

void lcdData(unsigned char data)
{
    lcd_dprt = data;          //send data to data port
    lcd_cprt |= (1<<lcd_rs);  //RS = 1 sending data to LCD
    lcd_cprt &= ~ (1<<lcd_rw); //RW = 0 writing data to LCD
    lcd_cprt |= (1<<lcd_en);   //en= 1

    _delay_us(1);

    lcd_cprt &= ~ (1<<lcd_en); //en=0 to complete a syn. cycle

    _delay_us(100);

}
```

Figure 2: Second Snippet

```
void lcd_init()
{
    lcd_dddr = 0xff;
    lcd_cddr = 0xff;

    lcd_cprt &= ~(1<<lcd_en);    //LCD en = 0

    _delay_us(2000);
    lcdCommand(0x38);          //configure LCD to 8-bit/2-line mode
    lcdCommand(0x0c);          //display ON, cursor ON
    lcdCommand(0x01);          //clear LCD screen

    _delay_us(2000);           //wait

    lcdCommand(0x06);          //move cursor to right side of the LCD screen
}


//*************************************************
void lcd_gotoxy(unsigned char x, unsigned char y)
{

  //unsigned char firstCharAdr[] = {0x80, 0xC0, 0x94, 0xD4};
  unsigned char firstCharAdr[] = {0x80, 0xC0};
  lcdCommand(firstCharAdr[y-1]+x-1);
  _delay_us(100);
}

```

Figure 3: Third Snippet



```
87
88  //*****************************************************
89  void lcd_print (char * str)
90  {
91    unsigned char i = 0;
92    while(str[i]!=0)
93    {
94      lcdData(str[i]);
95      i++;
96     }
97  }
98  //*****************************************************
99  int main(void)
100 {
101   //function call to initialize the LCD
102   lcd_init();
103   while (1){
104       //set cursor to row 1 and column 1 position on the LCD screen
105       lcd_gotoxy(1,1);
106       //print the message " I Love Processor " on the LCD screen
107       lcd_print("I Love Processor");
108       // move cursor to row 2 column 1 position on the LCD screen
109       lcd_gotoxy(1,2);
110       //print the message " System Design" on the LCD screen
111       lcd_print("^\\_(ツ)_/^");
112       //add while loop to show the message on the LCD screen until
113       //power to the board is removed.
114       }
115
116 }
117
```

Figure 4: Fourth Snippet