Goals:
The goal of this lab is to interface the LCD and Keypad to the AVR microcontroller.

Equipment Usage
For this lab the following equipment will be used:

- Atmega328P
- Keypad (4x4)
  - Datasheet: http://www.actcomponents.com/index_004.htm
- 16x2 LCD screen

IMPORTANT: IF YOU ARE USING A DIFFERENT TYPE OF KEYPAD OR LCD SCREEN THAN SPECIFIED ABOVE, PLEASE CHECK THE MODEL NUMBER ON YOUR DEVICE TO FIND THE PROPER DATASHEET.

Background:

LCD:
Up until this point, you have most likely displayed information using devices such as LEDs or 7-segment displays. Liquid Crystal Displays come in handy when the need for more specific information is required. The biggest difference from the previous devices you have used is that LCDs do not emit light directly. They operate by manipulating the external light that is reflected off of it.

LCDs are far more complex than simple display devices like LEDs. LCDs have several key components.

DDRAM- Data Display RAM contains 128 bytes of RAM (128x8 RAM), and any data written here will be displayed on the LCD.

CGRAM – Character Generator RAM contains 64 bytes of RAM and contains the various font styles accessible to the LCD.
Cursor – acts as an address counter; points to a location in either the DDRAM or CGRAM

Data Register – an 8 bit register that holds any data that is to be written. The location of the cursor decides where the data will be written.

Command Register - register holds the various commands for the LCD. For example, if the cursor location needs to be moved or if the screen needs to be cleared, accessing this register will grant access to those commands. Figure 9.1 lists the various commands that are used.

<table>
<thead>
<tr>
<th>Code (Hex)</th>
<th>Instruction</th>
<th>Code (Hex)</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Clear display screen</td>
<td>2</td>
<td>Return home</td>
</tr>
<tr>
<td>10</td>
<td>Shift cursor position to left</td>
<td>14</td>
<td>Shift cursor position to right</td>
</tr>
<tr>
<td>18</td>
<td>Shift display left</td>
<td>1C</td>
<td>Shift display right</td>
</tr>
<tr>
<td>4</td>
<td>After displaying a character on the LCD, shift cursor to left</td>
<td>6</td>
<td>After displaying a character on the LCD, shift cursor to right</td>
</tr>
<tr>
<td>80-FF</td>
<td>Set cursor position</td>
<td>40-7F</td>
<td>Set CG RAM address</td>
</tr>
<tr>
<td>8</td>
<td>Display off, cursor off</td>
<td>A</td>
<td>Display off, cursor on</td>
</tr>
<tr>
<td>C</td>
<td>Display on, cursor off</td>
<td>E</td>
<td>Display on, cursor on</td>
</tr>
<tr>
<td>F</td>
<td>Display on, cursor blinking</td>
<td>38</td>
<td>Initializing to 2 lines &amp; 5x7 font</td>
</tr>
</tbody>
</table>

Figure 9.1 LCD Commands

Setting the Cursor Position
Based on the size of the LCD screen the value to set the cursor position will vary. The address is an 8-bit number with the upper 4 bits signifying the row and the lower four bits signifying the column.

Once the cursor position is set, the character that is sent to the screen will appear on the current position.
Keypad:
The interface of the key pad is essentially a matrix. By pressing one of the buttons will create a connection between a row and column. To interpret this, the microcontroller will constantly send out high and low pulses throughout the grid and check to see if a connection is made.

Consider Figure 9.2 for the following example. Assume the microcontroller sends a low signal (GND) to the connected rows. When a button is pressed a connection to ground will be made on one of the columns (VCC). When the microcontroller reads that one of its column values has changed from VCC to GND it will interpret the change as a key press. Once the selected column is known the microcontroller should compare it with the currently selected row (i.e. the row receiving GND signal.) With both the row and column known, the microcontroller will access the user created table that corresponds to the keypad and return the value in the array. For this to work properly the rows and columns must be attached to the proper pins according to the code. The find out which pin represents which column and row you will have to refer to the datasheet. If no datasheet is present one method is to measure the resistance between 2 pins. If a button is pressed and a change in resistance occurs then those pins represent the row and column for that key.
#define F_CPU 8000000UL

#include <avr/io.h>  //standard AVR header
#include <util/delay.h>  //delay header

#define LCD_DPRT PORTD //LCD DATA PORT
#define LCD_DDDR DDRD //LCD DATA DDR
#define LCD_DPIN PIND //LCD DATA PIN
#define LCD_CPRT PORTB //LCD COMMANDS PORT
#define LCD_CDDR DDRB //LCD COMMANDS DDR
#define LCD_CPIN PINB //LCD COMMANDS PIN
#define LCD_RS 0 //LCD RS
#define LCD_RW 1 //LCD RW
#define LCD_EN 2 //LCD EN

//****************************************************************************
void delay_us(unsigned int d)
{
  _delay_us(d);
}
//****************************************************************************
void lcdCommanda (unsigned char cmnd)
{
  LCD_DPRT = cmnd; //send cmnd to data port
  LCD_CPRT &= ~(1<<LCD_RS); //RS = 0 for command
  LCD_CPRT &= ~(1<<LCD_RW); //RW = 0 for write
  delay_us(1); //wait to make enable wide
  LCD_CPRT &= ~(1<<LCD_EN); //EN = 0 for H-to-L pulse
  delay_us(100); //wait to make enable wide
}

void lcdData(unsigned char data)
{
  LCD_DPRT = data; //send data to data port
  LCD_CPRT |= (1<<LCD_RS); //RS = 1 for data
  LCD_CPRT &= ~(1<<LCD_RW); //RW = 0 for write
  delay_us(1); //wait to make enable wide
  LCD_CPRT &= ~(1<<LCD_EN); //EN = 0 for H-to-L pulse
  delay_us(100); //wait to make enable wide
}

void lcd_init()
{
  LCD_DDDR = 0xFF;
  LCD_CDDR = 0xFF;
  LCD_CPRT &=~(1<<LCD_EN); //LCD EN = 0
  delay_us(2000); //wait for init
  lcdCommanda(0x38); //init. LCD 2 line, 5x7
  lcdCommanda(0x0E); //display on, cursor on
  lcdCommanda(0x01); //clear LCD
  delay_us(2000); //wait
  lcdCommanda(0x06); //shift cursor right
}

//****************************************************************************
void lcd_gotoxy(unsigned char x, unsigned char y)
{
  unsigned char firstCharAdr[] = {0x80, 0xC0, 0x94, 0xD4};
  lcdCommanda(firstCharAdr[y-1] + x - 1);
  delay_us(100);
}

void lcd_print(char * str)
{
  unsigned char i = 0;
  while (str[i]!=0)
  {
    lcdData(str[i]);
    i++;
  }
}

int main(void)
{
  lcd_init();
  lcd_gotoxy(1,1);
  lcd_print("THE KEY ENTR IS");
  lcd_gotoxy(1,2);
  lcd_print("NOW GET IT");
  while(1); //stay here forever
  return 0;
}
#include <avr/io.h>
#include <util/delay.h>

#define F_CPU 8000000UL

#define KEY_PRT PORTD  //keyboard PORT
#define KEY_DDR DDRD  //keyboard DDR
#define KEY_PIN PIND  //keyboard PIN

void delay_ms(unsigned int d)
{
  _delay_ms(d);
}

unsigned char keypad[4][4] = {
  {'0','1','2','3'},
  {'4','5','6','7'},
  {'8','9','A','B'},
  {'C','D','E','F'},
};

int main(void)
{
  unsigned char colloc, rowloc;
  //keyboard routine. This sends the ASCII
  //code for pressed key to port c
  DDRD = 0xFF;
  KEY_DDR = 0xF0;
  KEY_PRT = 0xFF;

  while(1)
  {
    do
    {
      KEY_PRT &= 0x0F;  //ground all rows at once
      colloc = (KEY_PIN & 0x0F);  //read the columns
      //check until all keys released
    } while (colloc != 0x0F);

    delay_ms(20);  //call delay
    colloc = (KEY_PIN & 0x0F);  //see if any key is pressed
    while (colloc == 0x0F);  //keep checking for key press

    delay_ms(20);  //call delay for debounce
    colloc = (KEY_PIN & 0x0F);  //read columns
    while (colloc == 0x0F);  //wait for key press

    while(1)
    {
      KEY_PRT = 0xEF;  //ground row 0
      colloc = (KEY_PIN & 0x0F);  //read the columns
      if (colloc != 0x0F)  //column detected
      {
        rowloc = 0;  //save row location
        break;  //exit while loop
      }

      KEY_PRT = 0xDF;  //ground row 1
      colloc = (KEY_PIN & 0x0F);  //read the columns
      if (colloc != 0x0F)  //column detected
      {
        rowloc = 1;  //save row location
        break;  //exit while loop
      }

      KEY_PRT = 0xBF;  //ground row 2
      colloc = (KEY_PIN & 0x0F);  //read the columns
      if (colloc != 0x0F)  //column detected
      {
        rowloc = 2;  //save row location
        break;  //exit while loop
      }

      KEY_PRT = 0x7F;  //ground row 3
      colloc = (KEY_PIN & 0x0F);  //read the columns
      rowloc = 3;  //save row location
      break;  //exit while loop
    }

    //check column and send result to Port D
    if (colloc == 0x0E) PORTB = (keypad[rowloc][0]);
    else if (colloc == 0x0D) PORTB = (keypad[rowloc][1]);
    else if (colloc == 0x0B) PORTB = (keypad[rowloc][2]);
    else    PORTB = (keypad[rowloc][3]);

    return 0;
  }
}
Lab Experiments: Demonstrate the following experiments to the TA

Experiment 1: Display on the following on the LCD screen:

First Row “Last Name, First Name”

Second Row: CPE 301L: Lab 9

Experiment 2: Create a program that will display the word of the integer value of the buttons on the keypad when they are pressed. For example, when the ‘4’ key is pressed the LCD should display “4 was pressed.”

NOTE: (The sample code assumes that PD0-PD3 connected to columns & PD4-PD7 connected to rows. Connecting the Keypad differently will require modification of grid values.)

Experiment 3: Create a combinational lock system using the keypad and LCD screen. Your code should accept a 4-digit number. The correct number sequence should display “Lock Opened” while the incorrect number sequence should display “Incorrect Sequence”. The program should accept all four digits and the “enter” button before displaying a response.

Post-Lab Deliverables:

1) Submit your completed project report including your working code.
2) Submit a PCB layout Design of your circuit using Altium