LCD Interfacing with a PIC24
7th Laboratory Report for ECE 383
Microcomputers

Submitted by
Patrick Brooks
11650957

The University of Alabama
Tuscaloosa, AL 35487

4/2/21

Abstract:

The main objectives of this lab are to introduce students to LCD interfacing with the PIC24. They will be working in the PIC24 ecosystem while learning how to implement and use an LCD screen. Task one instructs students to build the schematic given on their breadboard. Task two asks students to output their name and school email to the LCD. Task three requires students to create a counter program that has certain characteristics described in the lab instructions.

Introduction:

The goal of this lab is to introduce students to LCD interfacing with the PIC24. They will be working in the PIC24 ecosystem while creating code to output what is required according to the lab instructions. Task one instructs students to build the schematic, given in the lab instructions, on their breadboard. Task two asks students to output their name and school email to the LCD by creating a C program inside of MPLAB. Task three requires students to create a counter program that has certain characteristics described in the lab instructions.

Procedure/Results:

<u>Task 1</u>: Connecting the LCD to the PIC24

a. Recreate given schematic

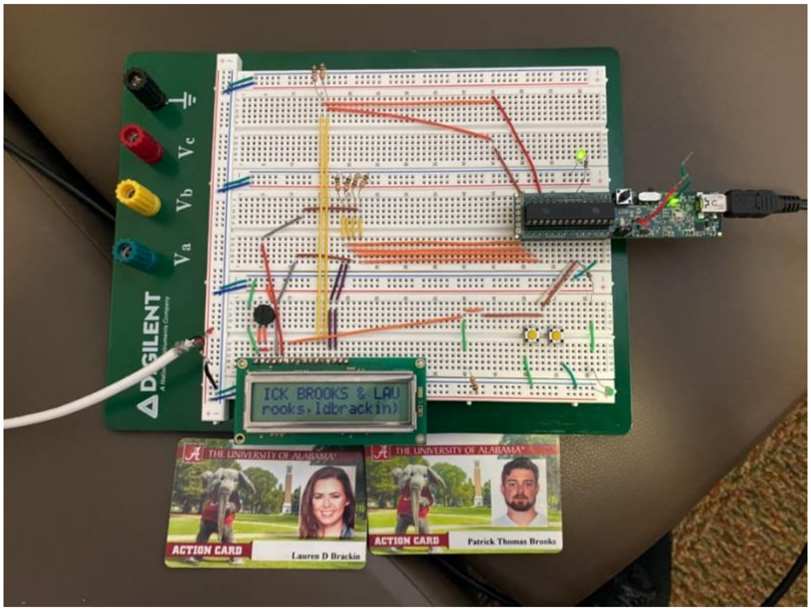


Figure 1. Deliverable 1

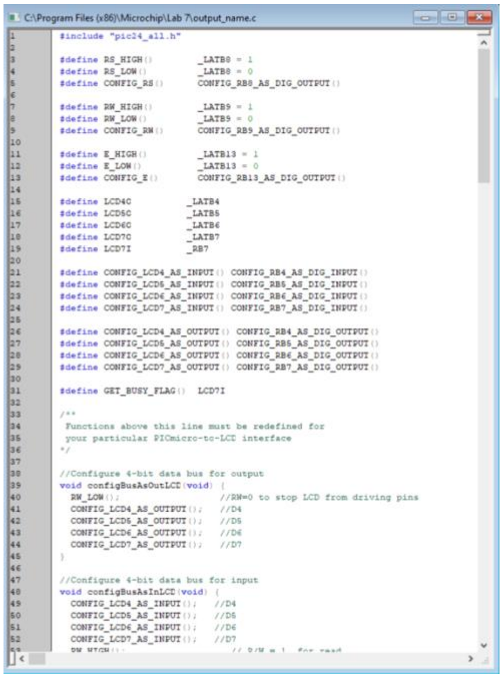
Task 2: Character Output to LCD

a. Use *lcd4bit.mcp* in “chap8” files for a template for this task

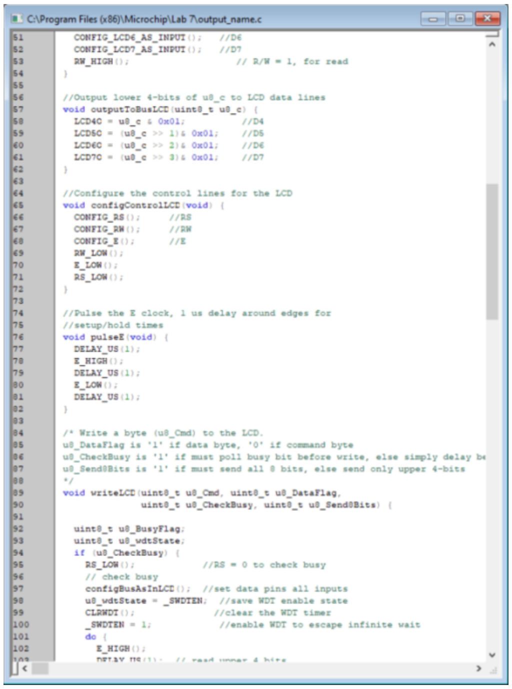
i. Create a program that outputs your name and email to the screen

https://youtu.be/eW3qBe9AUnA

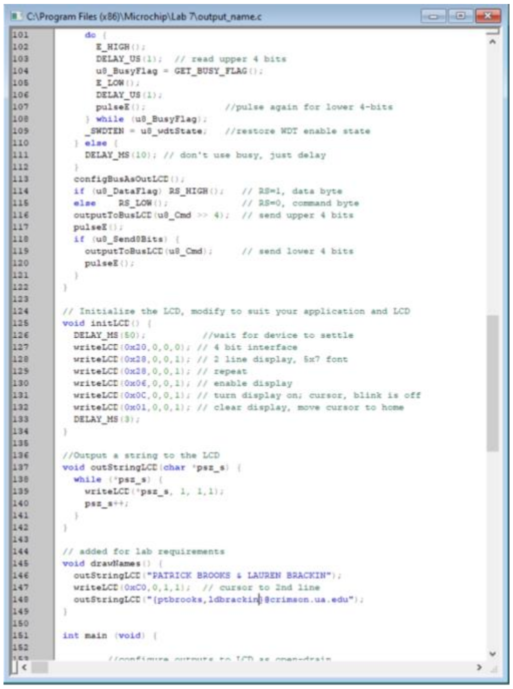
Figure 2. Deliverable 2



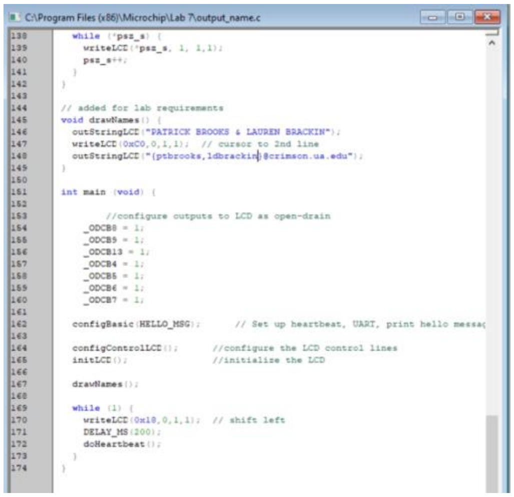
```
#include "pic24_all.h"

#define RS_HIGH()        _LATB8 = 1
#define RS_LOW()         _LATB8 = 0
#define CONFIG_RS()      CONFIG_RB8_AS_DIG_OUTPUT()

#define RW_HIGH()        _LATB9 = 1
#define RW_LOW()         _LATB9 = 0
#define CONFIG_RW()      CONFIG_RB9_AS_DIG_OUTPUT()

#define E_HIGH()         _LATB13 = 1
#define E_LOW()          _LATB13 = 0
#define CONFIG_E()       CONFIG_RB13_AS_DIG_OUTPUT()

#define LCD4O           _LATB4
#define LCD5O           _LATB5
#define LCD6O           _LATB6
#define LCD7O           _LATB7
#define LCD7I           _RB7

#define CONFIG_LCD4_AS_INPUT() CONFIG_RB4_AS_DIG_INPUT()
#define CONFIG_LCD5_AS_INPUT() CONFIG_RB5_AS_DIG_INPUT()
#define CONFIG_LCD6_AS_INPUT() CONFIG_RB6_AS_DIG_INPUT()
#define CONFIG_LCD7_AS_INPUT() CONFIG_RB7_AS_DIG_INPUT()

#define CONFIG_LCD4_AS_OUTPUT() CONFIG_RB4_AS_DIG_OUTPUT()
#define CONFIG_LCD5_AS_OUTPUT() CONFIG_RB5_AS_DIG_OUTPUT()
#define CONFIG_LCD6_AS_OUTPUT() CONFIG_RB6_AS_DIG_OUTPUT()
#define CONFIG_LCD7_AS_OUTPUT() CONFIG_RB7_AS_DIG_OUTPUT()

#define GET_BUSY_FLAG()  LCD7I

/**
 Functions above this line must be redefined for
 your particular PICmicro-to-LCD interface
*/

//Configure 4-bit data bus for output
void configBusAsOutLCD(void) {
  RW_LOW();                  //RW=0 to stop LCD from driving pins
  CONFIG_LCD4_AS_OUTPUT();   //D4
  CONFIG_LCD5_AS_OUTPUT();   //D5
  CONFIG_LCD6_AS_OUTPUT();   //D6
  CONFIG_LCD7_AS_OUTPUT();   //D7
}

//Configure 4-bit data bus for input
void configBusAsInLCD(void) {
  CONFIG_LCD4_AS_INPUT();   //D4
  CONFIG_LCD5_AS_INPUT();   //D5
  CONFIG_LCD6_AS_INPUT();   //D6
  CONFIG_LCD7_AS_INPUT();   //D7
  RW_HIGH();                   // R/W = 1, for read
```



```
  CONFIG_LCD6_AS_INPUT();   //D6
  CONFIG_LCD7_AS_INPUT();   //D7
  RW_HIGH();                   // R/W = 1, for read
}

//Output lower 4-bits of u8_c to LCD data lines
void outputToBusLCD(uint8_t u8_c) {
  LCD4O = u8_c & 0x01;          //D4
  LCD5O = (u8_c >> 1)& 0x01;    //D5
  LCD6O = (u8_c >> 2)& 0x01;    //D6
  LCD7O = (u8_c >> 3)& 0x01;    //D7
}

//Configure the control lines for the LCD
void configControlLCD(void) {
  CONFIG_RS();      //RS
  CONFIG_RW();      //RW
  CONFIG_E();       //E
  RW_LOW();
  E_LOW();
  RS_LOW();
}

//Pulse the E clock, 1 us delay around edges for
//setup/hold times
void pulseE(void) {
  DELAY_US(1);
  E_HIGH();
  DELAY_US(1);
  E_LOW();
  DELAY_US(1);
}

/* Write a byte (u8_Cmd) to the LCD.
u8_DataFlag is '1' if data byte, '0' if command byte
u8_CheckBusy is '1' if must poll busy bit before write, else simply delay be
u8_Send8Bits is '1' if must send all 8 bits, else send only upper 4-bits
*/
void writeLCD(uint8_t u8_Cmd, uint8_t u8_DataFlag,
              uint8_t u8_CheckBusy, uint8_t u8_Send8Bits) {

  uint8_t u8_BusyFlag;
  uint8_t u8_wdtState;
  if (u8_CheckBusy) {
    RS_LOW();              //RS = 0 to check busy
    // check busy
    configBusAsInLCD();  //set data pins all inputs
    u8_wdtState = _SWDTEN;  //save WDT enable state
    CLRWDT();               //clear the WDT timer
    _SWDTEN = 1;             //enable WDT to escape infinite wait
    do {
      E_HIGH();
      DELAY_US(1);  // read upper 4 bits
```



```
    do {
      E_HIGH();
      DELAY_US(1);  // read upper 4 bits
      u8_BusyFlag = GET_BUSY_FLAG();
      E_LOW();
      DELAY_US(1);
      pulseE();              //pulse again for lower 4-bits
    } while (u8_BusyFlag);
    _SWDTEN = u8_wdtState;   //restore WDT enable state
  } else {
    DELAY_MS(10); // don't use busy, just delay
  }
  configBusAsOutLCD();
  if (u8_DataFlag) RS_HIGH();   // RS=1, data byte
  else    RS_LOW();             // RS=0, command byte
  outputToBusLCD(u8_Cmd >> 4);  // send upper 4 bits
  pulseE();
  if (u8_Send8Bits) {
    outputToBusLCD(u8_Cmd);     // send lower 4 bits
    pulseE();
  }
}

// Initialize the LCD, modify to suit your application and LCD
void initLCD() {
  DELAY_MS(50);          //wait for device to settle
  writeLCD(0x20,0,0,0); // 4 bit interface
  writeLCD(0x28,0,0,1); // 2 line display, 5x7 font
  writeLCD(0x28,0,0,1); // repeat
  writeLCD(0x06,0,0,1); // enable display
  writeLCD(0x0C,0,0,1); // turn display on; cursor, blink is off
  writeLCD(0x01,0,0,1); // clear display, move cursor to home
  DELAY_MS(3);
}

//Output a string to the LCD
void outStringLCD(char *psz_s) {
  while (*psz_s) {
    writeLCD(*psz_s, 1, 1,1);
    psz_s++;
  }
}

// added for lab requirements
void drawNames() {
  outStringLCD("PATRICK BROOKS & LAUREN BRACKIN");
  writeLCD(0xC0,0,1,1);  // cursor to 2nd line
  outStringLCD("{ptbrooks,ldbrackin}@crimson.ua.edu");
}

int main (void) {

        //configure outputs to LCD as open-drain
```



```
  while (*psz_s) {
    writeLCD(*psz_s, 1, 1,1);
    psz_s++;
  }
}

// added for lab requirements
void drawNames() {
  outStringLCD("PATRICK BROOKS & LAUREN BRACKIN");
  writeLCD(0xC0,0,1,1);  // cursor to 2nd line
  outStringLCD("{ptbrooks,ldbrackin}@crimson.ua.edu");
}

int main (void) {

        //configure outputs to LCD as open-drain
    _ODCB8 = 1;
    _ODCB9 = 1;
    _ODCB13 = 1;
    _ODCB4 = 1;
    _ODCB5 = 1;
    _ODCB6 = 1;
    _ODCB7 = 1;

  configBasic(HELLO_MSG);      // Set up heartbeat, UART, print hello messa

  configControlLCD();      //configure the LCD control lines
  initLCD();               //initialize the LCD

  drawNames();

  while (1) {
    writeLCD(0x18,0,1,1);  // shift left
    DELAY_MS(200);
    doHeartbeat();
  }
}
```

Figure 3. Deliverable 3

Task 3: Counter Output to LCD

a. Use *lcd4bit.mcp* in “chap8” files for a template for this task

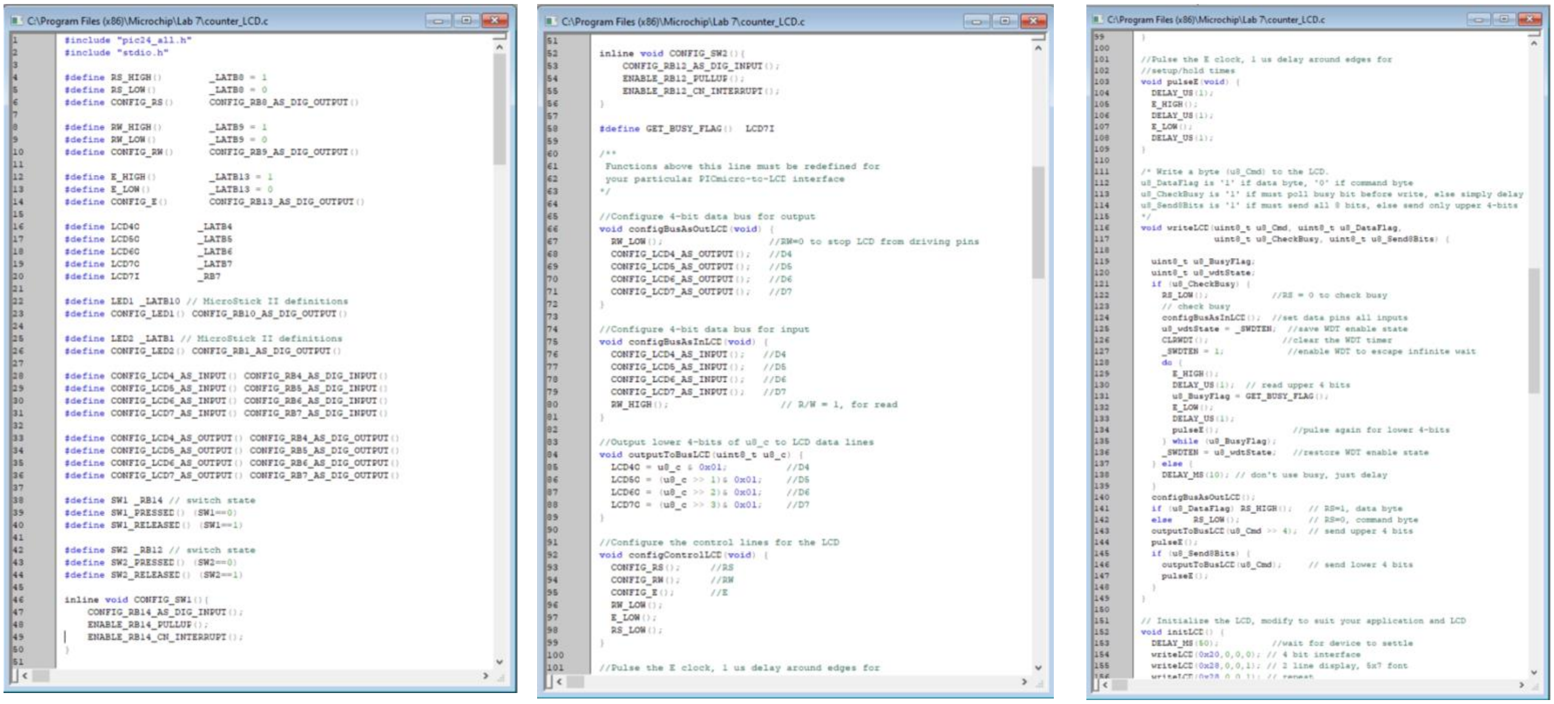
i. Create a program that outputs a counter from *000-015*

ii. Once the counter hit *015*, counter resets

iii. If SW1 is pressed the counter will reset to *000*

https://youtu.be/Sg8ta7pwU5M

Figure 4. Deliverable 4

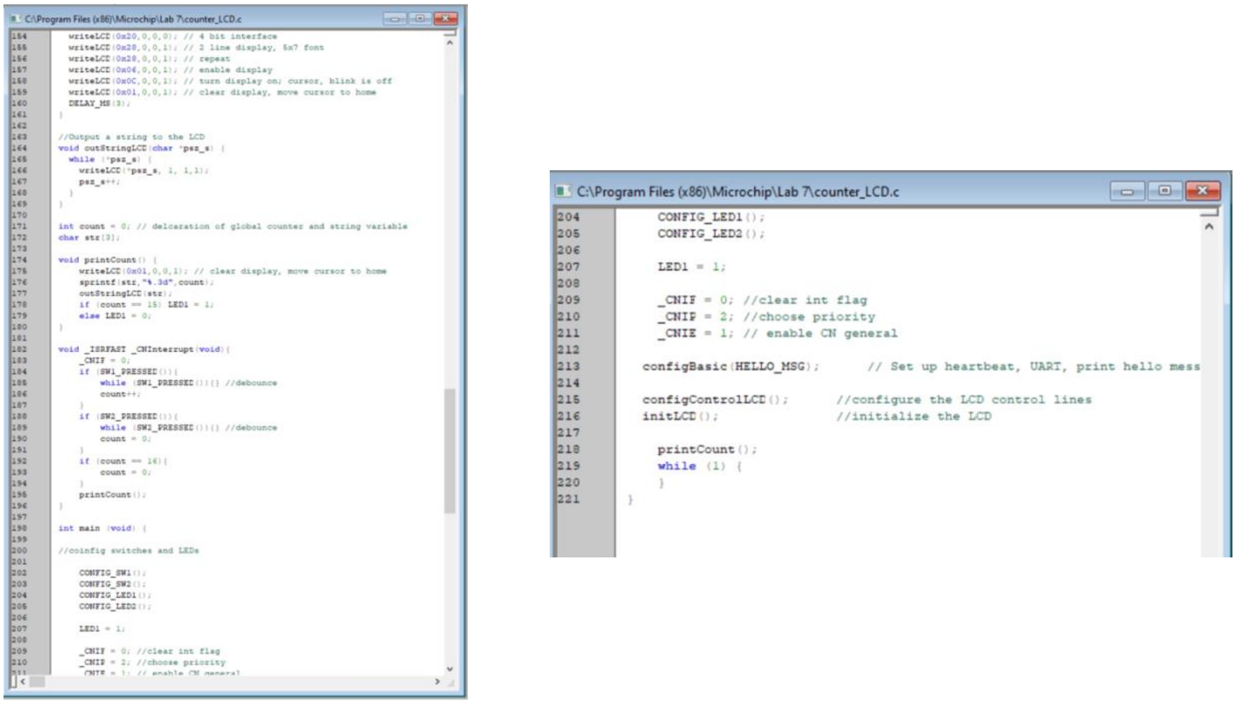


```c
#include "pic24_all.h"
#include "stdio.h"

#define RS_HIGH()        _LATB8 = 1
#define RS_LOW()         _LATB8 = 0
#define CONFIG_RS()      CONFIG_RB8_AS_DIG_OUTPUT()

#define RW_HIGH()        _LATB9 = 1
#define RW_LOW()         _LATB9 = 0
#define CONFIG_RW()      CONFIG_RB9_AS_DIG_OUTPUT()

#define E_HIGH()         _LATB13 = 1
#define E_LOW()          _LATB13 = 0
#define CONFIG_E()       CONFIG_RB13_AS_DIG_OUTPUT()

#define LCD4O          _LATB4
#define LCD5O          _LATB5
#define LCD6O          _LATB6
#define LCD7O          _LATB7
#define LCD7I          _RB7

#define LED1 _LATB10 // MicroStick II definitions
#define CONFIG_LED1() CONFIG_RB10_AS_DIG_OUTPUT()

#define LED2 _LATB1 // MicroStick II definitions
#define CONFIG_LED2() CONFIG_RB1_AS_DIG_OUTPUT()

#define CONFIG_LCD4_AS_INPUT() CONFIG_RB4_AS_DIG_INPUT()
#define CONFIG_LCD5_AS_INPUT() CONFIG_RB5_AS_DIG_INPUT()
#define CONFIG_LCD6_AS_INPUT() CONFIG_RB6_AS_DIG_INPUT()
#define CONFIG_LCD7_AS_INPUT() CONFIG_RB7_AS_DIG_INPUT()

#define CONFIG_LCD4_AS_OUTPUT() CONFIG_RB4_AS_DIG_OUTPUT()
#define CONFIG_LCD5_AS_OUTPUT() CONFIG_RB5_AS_DIG_OUTPUT()
#define CONFIG_LCD6_AS_OUTPUT() CONFIG_RB6_AS_DIG_OUTPUT()
#define CONFIG_LCD7_AS_OUTPUT() CONFIG_RB7_AS_DIG_OUTPUT()

#define SW1 _RB14 // switch state
#define SW1_PRESSED() (SW1==0)
#define SW1_RELEASED() (SW1==1)

#define SW2 _RB12 // switch state
#define SW2_PRESSED() (SW2==0)
#define SW2_RELEASED() (SW2==1)

inline void CONFIG_SW1(){
    CONFIG_RB14_AS_DIG_INPUT();
    ENABLE_RB14_PULLUP();
    ENABLE_RB14_CN_INTERRUPT();
}
```

```c

inline void CONFIG_SW2(){
    CONFIG_RB12_AS_DIG_INPUT();
    ENABLE_RB12_PULLUP();
    ENABLE_RB12_CN_INTERRUPT();
}

#define GET_BUSY_FLAG()  LCD7I

/**
 Functions above this line must be redefined for
 your particular PICmicro-to-LCD interface
*/

//Configure 4-bit data bus for output
void configBusAsOutLCD(void) {
  RW_LOW();                   //RW=0 to stop LCD from driving pins
  CONFIG_LCD4_AS_OUTPUT();    //D4
  CONFIG_LCD5_AS_OUTPUT();    //D5
  CONFIG_LCD6_AS_OUTPUT();    //D6
  CONFIG_LCD7_AS_OUTPUT();    //D7
}

//Configure 4-bit data bus for input
void configBusAsInLCD(void) {
  CONFIG_LCD4_AS_INPUT();    //D4
  CONFIG_LCD5_AS_INPUT();    //D5
  CONFIG_LCD6_AS_INPUT();    //D6
  CONFIG_LCD7_AS_INPUT();    //D7
  RW_HIGH();                     // R/W = 1, for read
}

//Output lower 4-bits of u8_c to LCD data lines
void outputToBusLCD(uint8_t u8_c) {
  LCD4O = u8_c & 0x01;          //D4
  LCD5O = (u8_c >> 1)& 0x01;    //D5
  LCD6O = (u8_c >> 2)& 0x01;    //D6
  LCD7O = (u8_c >> 3)& 0x01;    //D7
}

//Configure the control lines for the LCD
void configControlLCD(void) {
  CONFIG_RS();     //RS
  CONFIG_RW();     //RW
  CONFIG_E();      //E
  RW_LOW();
  E_LOW();
  RS_LOW();
}

//Pulse the E clock, 1 us delay around edges for
```

```c
}

//Pulse the E clock, 1 us delay around edges for
//setup/hold times
void pulseE(void) {
  DELAY_US(1);
  E_HIGH();
  DELAY_US(1);
  E_LOW();
  DELAY_US(1);
}

/* Write a byte (u8_Cmd) to the LCD.
u8_DataFlag is '1' if data byte, '0' if command byte
u8_CheckBusy is '1' if must poll busy bit before write, else simply delay
u8_Send8Bits is '1' if must send all 8 bits, else send only upper 4-bits
*/
void writeLCD(uint8_t u8_Cmd, uint8_t u8_DataFlag,
              uint8_t u8_CheckBusy, uint8_t u8_Send8Bits) {

  uint8_t u8_BusyFlag;
  uint8_t u8_wdtState;
  if (u8_CheckBusy) {
    RS_LOW();             //RS = 0 to check busy
    // check busy
    configBusAsInLCD();  //set data pins all inputs
    u8_wdtState = _SWDTEN;  //save WDT enable state
    CLRWDT();              //clear the WDT timer
    _SWDTEN = 1;            //enable WDT to escape infinite wait
    do {
      E_HIGH();
      DELAY_US(1);  // read upper 4 bits
      u8_BusyFlag = GET_BUSY_FLAG();
      E_LOW();
      DELAY_US(1);
      pulseE();              //pulse again for lower 4-bits
    } while (u8_BusyFlag);
    _SWDTEN = u8_wdtState;   //restore WDT enable state
  } else {
    DELAY_MS(10); // don't use busy, just delay
  }
  configBusAsOutLCD();
  if (u8_DataFlag) RS_HIGH();   // RS=1, data byte
  else    RS_LOW();             // RS=0, command byte
  outputToBusLCD(u8_Cmd >> 4);  // send upper 4 bits
  pulseE();
  if (u8_Send8Bits) {
    outputToBusLCD(u8_Cmd);     // send lower 4 bits
    pulseE();
  }
}

// Initialize the LCD, modify to suit your application and LCD
void initLCD() {
  DELAY_MS(50);          //wait for device to settle
  writeLCD(0x20,0,0,0); // 4 bit interface
  writeLCD(0x28,0,0,1); // 2 line display, 5x7 font
  writeLCD(0x28,0,0,1); // repeat
```

Figure 5. Deliverable 5.1


C:\Program Files (x86)\Microchip\Lab 7\counter_LCD.c
C:\Program Files (x86)\Microchip\Lab 7\counter_LCD.c

```
    writeLCD(0x20,0,0,0); // 4 bit interface
    writeLCD(0x28,0,0,1); // 2 line display, 5x7 font
    writeLCD(0x28,0,0,1); // repeat
    writeLCD(0x06,0,0,1); // enable display
    writeLCD(0x0C,0,0,1); // turn display on; cursor, blink is off
    writeLCD(0x01,0,0,1); // clear display, move cursor to home
    DELAY_MS(3);
  }

  //Output a string to the LCD
  void outStringLCD(char *psz_s) {
    while (*psz_s) {
      writeLCD(*psz_s, 1, 1,1);
      psz_s++;
    }
  }

  int count = 0; // delcaration of global counter and string variable
  char str[3];

  void printCount() {
      writeLCD(0x01,0,0,1); // clear display, move cursor to home
      sprintf(str,"%.3d",count);
      outStringLCD(str);
      if (count == 15) LED1 = 1;
      else LED1 = 0;
  }

  void _ISRFAST _CNInterrupt(void){
      _CNIF = 0;
      if (SW1_PRESSED()){
          while (SW1_PRESSED()){} //debounce
          count++;
      }
      if (SW2_PRESSED()){
          while (SW2_PRESSED()){} //debounce
          count = 0;
      }
      if (count == 16){
          count = 0;
      }
      printCount();
  }

  int main (void) {

  //coinfig switches and LEDs

      CONFIG_SW1();
      CONFIG_SW2();
      CONFIG_LED1();
      CONFIG_LED2();

      LED1 = 1;

      _CNIF = 0; //clear int flag
      _CNIP = 2; //choose priority
      _CNIE = 1; // enable CN general
```

```
      CONFIG_LED1();
      CONFIG_LED2();

      LED1 = 1;

      _CNIF = 0; //clear int flag
      _CNIP = 2; //choose priority
      _CNIE = 1; // enable CN general

    configBasic(HELLO_MSG);      // Set up heartbeat, UART, print hello mess

    configControlLCD();      //configure the LCD control lines
    initLCD();               //initialize the LCD

      printCount();
      while (1) {
      }
  }
```

Figure 5. Deliverable 5.2 (cont.)

Conclusion:

After completion of the lab students learned how to connect the LCD screen to the PIC24 system, revisited important foundational C language techniques, learned how to incorporate interrupts, and learned how to critically think through these problems. Students also became familiar with the pins on the PIC24 and how they can be used to perform different tasks. Students also revisited useful software tools implemented in MPLAB. Students also deepened their understanding of the lab report format which will be used throughout their academic endeavors.