Junchi Feng
Lab 5 Report
4/16/19

**Description of LCDscreen.h**

A library which gives functionality to an LCD screen, and allows it to print strings in fancy ways.

A copy of your library (.c, and .h files zipped)
State the supported microprocessor and LCD

Supporting the PIC24 and the AQM0802A-RN-GBW LCD using the ST7032 controller chip.

**Dependencies**

p24FJ64GA002.h
This library makes it so that the PIC24 can work, and defines everything relating to the PIC24's pinout.

xc.h
Adds the MPLAB X16 C complier.

string.h
Adds string functionality to C.

stdbool.h
Adds boolean functionality to C.

feng_lab2b_asmLib_v001.h
Adds delay functions using assembly which are more accurate than one written in pure C.

**Function Documentation**

void setup(void);
Sets up the PIC24 Microcontroller for use. Set up TRISB2 and TRISB3 as output. Set up I2C.

```
int main(void) {
    setup();
    lcd_init();
    lcd_clear();
    while(1) {
        scroll_left("Hello World!");
        scroll_right("Hello World!");
    }
    return 0;
}
```

void delay(int delay_in_ms);
Puts the assembly code into a C function for a delay. It takes an integer number of milliseconds, and that is how long the delay is.

```
void lcd_init(void) {
    delay(50);
    lcd_cmd(0b00111000);
    delay(1);

    lcd_cmd(0b00111001);
    delay(1);

    lcd_cmd(0b00010100);
   delay(1);

   lcd_cmd(0b01110000);
   delay(1);

    lcd_cmd(0b01011110);
    delay(1);

   lcd_cmd(0b01101100);
    delay(200);

   lcd_cmd(0b00111000);
    delay(1);

   lcd_cmd(0b00001100);
   delay(1);

    lcd_cmd(0b00000001);


    delay(2);
}
```

void microDelay(int delay_in_ms);
Puts the assembly code into a C function for a delay. It takes an integer number of microseconds, and that is how long the delay is

void lcd_cmd(char command);
Handles the I2C to send the data of a single binary number through the I2C. Send a single char command to the LCD.

```
void lcd_setCursor(char x, char y) {
    if (x < 0 || x > 1 || y < 0 || y > 7) {
        return;
    }
    else {
        char cursorLocation = 0b10000000;
        cursorLocation |= y;
        if (x == 1) {
            cursorLocation |= 0b01000000;
        }
        lcd_cmd(cursorLocation);
    }
```

```
}
```

void lcd_init(void);
Uses the lcd_cmd function and the delay function to initialize the LCD by sending the binary numbers specified in the datasheet for normal functioning.

```
int main(void) {
    setup();
    lcd_init();
    lcd_clear();
    while(1) {
        scroll_left("Hello World!");
        scroll_right("Hello World!");
    }
    return 0;
}
```

void lcd_printChar(char myChar);
Prints a single character specified by the parameter, myChar, a character in ASCII. Similar to lcd_cmd in that it sends the data through I2C.

void lcd_setCursor(char x, char y);
Sets the location of the cursor given two character coordinates, i.e. where the LCD will write, and uses lcd_cmd to send that cursor location through I2C to the LCD.

char getCoordinates(char x, char y);
Similar to lcd_setCursor, given two character coordinates, it returns the binary number which would normally later be used in the lcd_cmd function to be sent if that's where the cursor should be located.

```
    for (i = 0; i < 8; i++) {
        if (i == 7) { //last char in string
            I2C2TRN = 0b10000000;
            while(!IFS3bits.MI2C2IF);
            IFS3bits.MI2C2IF = 0;
             char location = getCoordinates(row,column);
             I2C2TRN = location;
             while(!IFS3bits.MI2C2IF);
             IFS3bits.MI2C2IF = 0;
             I2C2TRN = 0b01000000;   // Control Byte: CO = 0 RS =1 ?last byte?
             while(!IFS3bits.MI2C2IF);
             IFS3bits.MI2C2IF = 0;
             I2C2TRN = s[i];  //data byte
             while(!IFS3bits.MI2C2IF);
             IFS3bits.MI2C2IF = 0;
        }
```

void lcd_printStr(const char *s);
Prints each character in a string to the LCD. Communicates the I2C directly in doing so.

```
void scroll_right(const char *s) {
   if (strlen(s) < 9) {
      lcd_printStr(s);
   }
   else {
      int i = 0;
      for (i = strlen(s)-1; i >= 7; i--) {
         char temp[8];
         int k = 0;
         int tempPos = 7;
         for (k = i; k > i-8; k--) {
            temp[tempPos] = s[k];
            tempPos--;
         }
         lcd_printStr(temp);
         int j = 0;
         for (j = 0; j < 500; j++) {
            ms_wait();
         }
      }
   }
}
```

void scroll_left(const char *s);
uses lcd_printStr to print a string in such a way so that it scrolls left across the LCD screen.

```
int main(void) {
   setup();
   lcd_init();
   lcd_clear();
   while(1) {
      scroll_left("Hello World!");
      scroll_right("Hello World!");
   }
   return 0;
}
```

void scroll_right(const char *s);
Does the same as scroll left but right.

```
int main(void) {
   setup();
   lcd_init();
   lcd_clear();
   while(1) {
      scroll_left("Hello World!");
```

```
        scroll_right("Hello World!");
    }
    return 0;
}
```

void lcd_clear(void);
Clears the LCD by sending an empty string to lcd_printSTR

```
int main(void) {
    setup();
    lcd_init();
    lcd_clear();
    while(1) {
        scroll_left("Hello World!");
        scroll_right("Hello World!");
    }
    return 0;
}
```