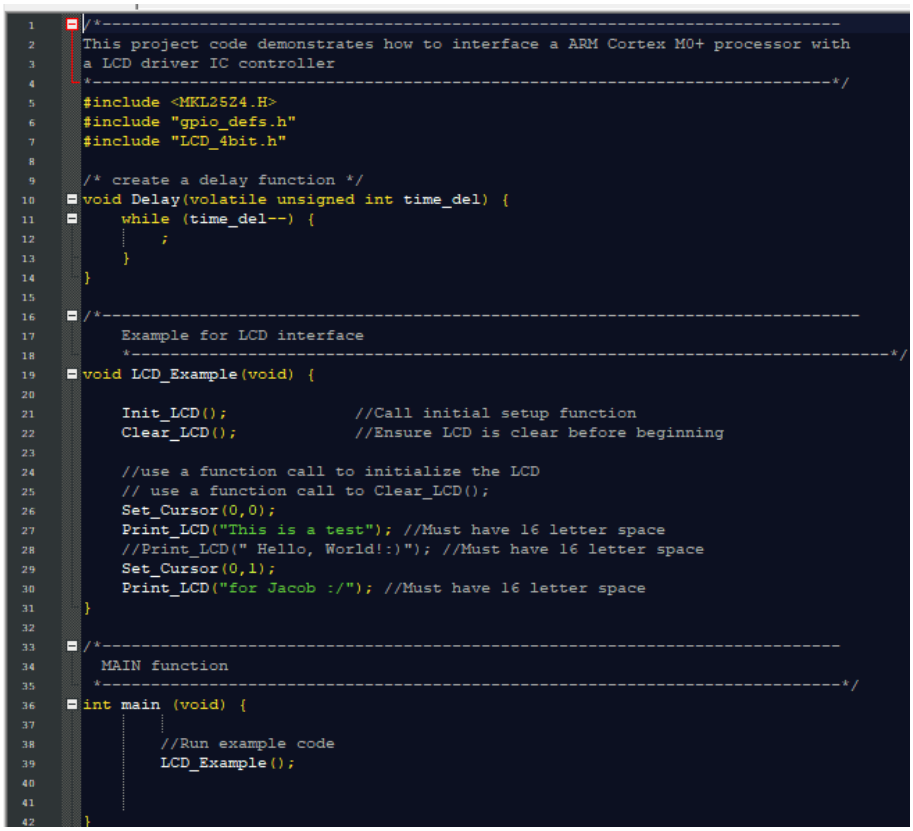
Lab 10
Jacob Hillebrand
CEE-345 Microprocessor System Design
LCD Interfacing Lab

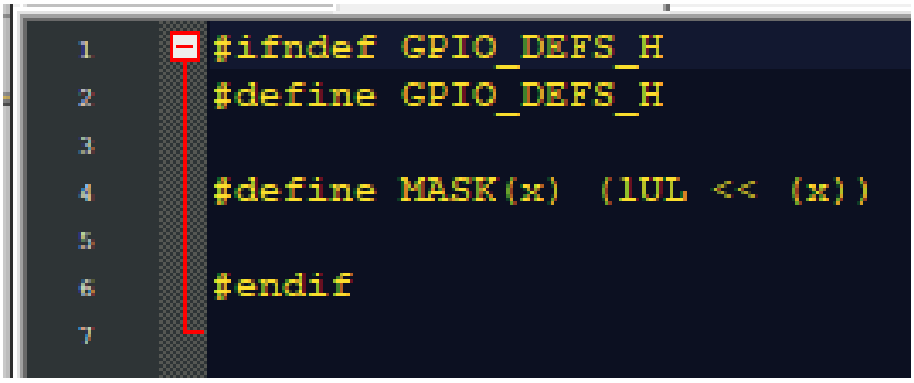
In this lab, we explored interfacing with a 4-bit LCD module using the KL25Z FreedomBoard. Essentially, we looked at how to connect the proper data pins, register select, enable, and write pins, and power and ground pins to the KL25Z. Then, the board was programmed to set up the display, and push a message out to it.

To do this, several things had to happen. First, the pins used on the KL25Z had to be set up as GPIO and output. Next, the register select and read/write pins had to be set to initialize the LCD, then set to begin accepting input on the 4 data pins. For each character sent across the data pins, the first 4 bits were sent, then a signal was sent to show that the second 4 bits was being sent, and finally, the last 4 bits were sent. These sets of 4 bits could be combined as 8 bits to reference a specific character in the LCD's character register, which would determine what character to display on screen. This was repeated for each character on the top row, then a signal was sent to the register select pin, and the entire process was repeated to display on the bottom row. The code was split into 4 major files – uv5_LCD.c, LCD_4bit.h, LCD_4bit.c, and gpio_defs.h – which are shown in the screenshots below.



```c
/*----------------------------------------------------------------------------
This project code demonstrates how to interface a ARM Cortex M0+ processor with
a LCD driver IC controller
*----------------------------------------------------------------------------*/
#include <MKL25Z4.H>
#include "gpio_defs.h"
#include "LCD_4bit.h"

/* create a delay function */
void Delay(volatile unsigned int time_del) {
    while (time_del--) {
        ;
    }
}

/*----------------------------------------------------------------------------
    Example for LCD interface
    *----------------------------------------------------------------------------*/
void LCD_Example(void) {

    Init_LCD();             //Call initial setup function
    Clear_LCD();            //Ensure LCD is clear before beginning

    //use a function call to initialize the LCD
    // use a function call to Clear_LCD();
    Set_Cursor(0,0);
    Print_LCD("This is a test"); //Must have 16 letter space
    //Print_LCD(" Hello, World!:)"); //Must have 16 letter space
    Set_Cursor(0,1);
    Print_LCD("for Jacob :/"); //Must have 16 letter space
}

/*----------------------------------------------------------------------------
  MAIN function
 *----------------------------------------------------------------------------*/
int main (void) {

        //Run example code
        LCD_Example();


}
```

Figure 1: Main.c



```
#ifndef GPIO_DEFS_H
#define GPIO_DEFS_H

#define MASK(x)  (1UL << (x))

#endif
```

Figure 2: gpio.h

```
#include <MKL25Z4.H>

#define LCD_COLUMNS   16   // Number of LCD columns in characters
#define LCD_ROWS      2    // Number of LCD rows

/*-------------------- LCD interface hardware definitions --------------------*/

/* Connections from LCD to MCU port bits:
 DB4 through DB8 are contiguous, starting with LSB at bit position PIN_DATA_SHIFT

  For example:
   - DB4 = PTD0
   - DB5 = PTD1
   - DB6 = PTD2
   - DB7 = PTD3

   - E   = PTD4
   - RW  = PTA13
   - RS  = PTD5                                                              */

//The MCU port pins connected to the LCD must first be initialized as GPIO.
//This assigns the four data bus signals to PORTD bit 0 to bit 3
//and the three control signals Port D bit 4 and 5, and Port A bit 13.
//here uses #define macro to simplify code development.

#define PIN_DATA_PORT         PORTD
#define PIN_DATA_PT           PTD
#define PIN_DATA_SHIFT        (0)

#define PIN_E_PORT            PORTD
#define PIN_E_PT              PTD
#define PIN_E_SHIFT           (4)
#define PIN_E                 ( 1 << PIN_E_SHIFT)

#define PIN_RW_PORT           PORTA
#define PIN_RW_PT             PTA
#define PIN_RW_SHIFT          (13)
#define PIN_RW                ( 1 << PIN_RW_SHIFT)

#define PIN_RS_PORT           PORTD
#define PIN_RS_PT             PTD
#define PIN_RS_SHIFT          (5)
#define PIN_RS                ( 1 << PIN_RS_SHIFT)
```

Figure 3: LCD_4bit.h Snippet 1/2

```
44
45  #define PINS_DATA             (0x0F << PIN_DATA_SHIFT)
46
47  /* Enable Clock for peripheral driving LCD pins */
48  #define ENABLE_LCD_PORT_CLOCKS    SIM->SCGC5 |= SIM_SCGC5_PORTD_MASK | SIM_SCGC5_PORTA_MASK | SIM_SCGC5_PORTC_MASK;
49
50  //Three macros manipulate the control lines:
51  /*define macros for LCD Enable, RW, and RS control pins*/
52  #define SET_LCD_E(x)              if (x) {PIN_E_PT->PSOR = PIN_E;} else {PIN_E_PT->PCOR = PIN_E;}
53  #define SET_LCD_RW(x)             if (x) {PIN_RW_PT->PSOR = PIN_RW;} else {PIN_RW_PT->PCOR = PIN_RW;}
54  #define SET_LCD_RS(x)             if (x) {PIN_RS_PT->PSOR = PIN_RS;} else {PIN_RS_PT->PCOR = PIN_RS;}
55
56
57  //Two macros access the 4-bit LCD data bus:
58  #define SET_LCD_DATA_OUT(x)       PIN_DATA_PT->PDOR = (PIN_DATA_PT->PDOR & ~PINS_DATA) | ((x) << PIN_DATA_SHIFT);
59
60  //#define GET_LCD_DATA_IN           (((PIN_DATA_PT->PDIR & PINS_DATA) >> PIN_DATA_SHIFT) & 0x0F)
61  #define GET_LCD_DATA_IN           PIN_DATA_PT->PDIR
62
63  ////Three macros set port pin directions: (_DIR_) below
64  /* Setting all port pins to output mode using the pin definition macros created previously
65  #define SET_LCD_ALL_DIR_OUT       { PIN_DATA_PT->PDDR = PIN_DATA_PT->PDDR | PINS_DATA; \
66                                    PIN_E_PT->PDDR = PIN_E_PT->PDDR | PIN_E; \
67                                    PIN_RW_PT->PDDR = PIN_RW_PT->PDDR | PIN_RW; \
68                                    PIN_RS_PT->PDDR = PIN_RS_PT->PDDR | PIN_RS; }
69
70  /* Setting DATA pins to input mode                                          */
71  #define SET_LCD_DATA_DIR_IN       PIN_DATA_PT->PDDR = PIN_DATA_PT->PDDR & ~PINS_DATA;
72
73  /* Setting DATA pins to output mode                                         */
74  #define SET_LCD_DATA_DIR_OUT      PIN_DATA_PT->PDDR = PIN_DATA_PT->PDDR | PINS_DATA;
75
76  #define LCD_BUSY_FLAG_MASK        (0x80)
77
78  /*****************************************************************************/
79  void Init_LCD (void);
80  void Set_Cursor (uint8_t column, uint8_t row);
81  void Clear_LCD(void);
82  void Print_LCD (char *string);
83  void lcd_putchar (char c);
84
```

Figure 4: LCD_4bit.h Snippet 2/2

```
1   #include "LCD_4bit.h"
2   #include "delay.h"
3   #include <stdint.h>
4
5
6   //with 4-bit data, we need to break
7   //down to two 4-bit operations, upper
8   //nibble and lower nibble.
9
10  //reading 4-bit nibble to
11  //LCD can be done by using
12  //( LCD_DATA_IN & 0x0F) or (LCD_DATA_IN << 4)
13
14  static uint8_t lcd_read_status(void)
15  {
16    uint8_t status;
17
18    SET_LCD_DATA_DIR_IN
19    SET_LCD_RS(0)
20    SET_LCD_RW(1)
21    Delay(1);
22    SET_LCD_E(1)
23    Delay(1);
24    status  = GET_LCD_DATA_IN << 4;
25    SET_LCD_E(0)
26    Delay(1);
27    SET_LCD_E(1)
28    Delay(1);
29    status |= GET_LCD_DATA_IN;
30    SET_LCD_E(0)
31    SET_LCD_DATA_DIR_OUT
32    return(status);
33  }
34
35  void wait_while_busy(void)
36  {
37    for( ; lcd_read_status() & LCD_BUSY_FLAG_MASK; )
38      ;
39  }
40
```

Figure 5: LCD_4bit.c Snippet 1/4

```
41  //To send information to the LCD across the 4 bit data bus,
42  //we use the following function. This function is called twice
43  //to write a byte.
44  void lcd_write_4bit(uint8_t c)
45  {
46    SET_LCD_RW(0)
47    SET_LCD_E(1)
48    SET_LCD_DATA_OUT(c&0x0F)
49    Delay(1);
50    SET_LCD_E(0)
51    Delay(1);
52  }
53
54  //To write an instruction (i.e. command) to the LCD controller,
55  //we use this function which writes the argument while keeping
56  //the RS line at 0, indicating the byte is a command.
57  void lcd_write_cmd(uint8_t c)
58  {
59    wait_while_busy();
60
61    SET_LCD_RS(0)
62    lcd_write_4bit(c>>4);
63    lcd_write_4bit(c);
64  }
65
66  //To write data to the LCD controller, we use this function,
67  //which writes the argument while keeping the RS line at 1,
68  //indicating the byte is data.
69  static void lcd_write_data(uint8_t c)
70  {
71    wait_while_busy();
72
73    SET_LCD_RS(1)
74    lcd_write_4bit(c>>4);
75    lcd_write_4bit(c);
76  }
77
78  void lcd_putchar(char c)
79  {
80    lcd_write_data(c);
81  }
82
```

Figure 6: LCD_4bit.c Snippet 2/4

```
83  //The MCU port pins connected to the LCD must
84  //first be initialized as GPIO.
85  void lcd_init_port(void) {
86    /* Enable clocks for peripherals        */
87    ENABLE_LCD_PORT_CLOCKS
88
89    /* Set Pin Mux to GPIO */
90    PIN_DATA_PORT->PCR[PIN_DATA_SHIFT] = PORT_PCR_MUX(1);
91    PIN_DATA_PORT->PCR[PIN_DATA_SHIFT+1] = PORT_PCR_MUX(1);
92    PIN_DATA_PORT->PCR[PIN_DATA_SHIFT+2] = PORT_PCR_MUX(1);
93    PIN_DATA_PORT->PCR[PIN_DATA_SHIFT+3] = PORT_PCR_MUX(1);
94    PIN_E_PORT->PCR[PIN_E_SHIFT] = PORT_PCR_MUX(1);
95    PIN_RW_PORT->PCR[PIN_RW_SHIFT] = PORT_PCR_MUX(1);
96    PIN_RS_PORT->PCR[PIN_RS_SHIFT] = PORT_PCR_MUX(1);
97  }
98
99  //We use these pieces to initialize the HD44780 LCD controller
100 //as directed in the datasheet
101 void Init_LCD(void)
102 {
103   /* initialize port(s) for LCD */
104   lcd_init_port();
105
106   /* Set all pins for LCD as outputs */
107   SET_LCD_ALL_DIR_OUT
108   Delay(100);
109   SET_LCD_RS(0)
110
111   //0x03 allows to shift 4-bit data to LCD
112   //see SET_LCD_DATA_OUT(x) when x=0x03
113   lcd_write_4bit(0x03);
114   Delay(100);
115   lcd_write_4bit(0x03);
116   Delay(10);
117  // lcd_write_4bit(0x3);
118  // lcd_write_4bit(0x2);
119   //Function Set: 4-bit, 2 Line, 5x7 Dots
120   lcd_write_cmd(0x28);
121  // Display on Cursor off
122   lcd_write_cmd(0x0C);
123  // Entry Mode
124   lcd_write_cmd(0x06);
125  //Set DDRAM address or coursor position on display
126   lcd_write_cmd(0x80);
127 }
```

Figure 7: LCD_4bit.c Snippet 3/4

```
129  void Set_Cursor(uint8_t column, uint8_t row)
130  {
131    uint8_t address;
132
133    address = (row * 0x40) + column;
134    address |= 0x80;
135    lcd_write_cmd(address);
136  }
137
138  void Clear_LCD(void)
139  {
140    lcd_write_cmd(0x01);
141    Set_Cursor(0, 0);
142  }
143
144  void Print_LCD(char *string)
145  {
146    while(*string)  {
147      lcd_putchar(*string++);
148    }
149  }
150
```

Figure 8: LCD_4bit.c Snippet 4/4

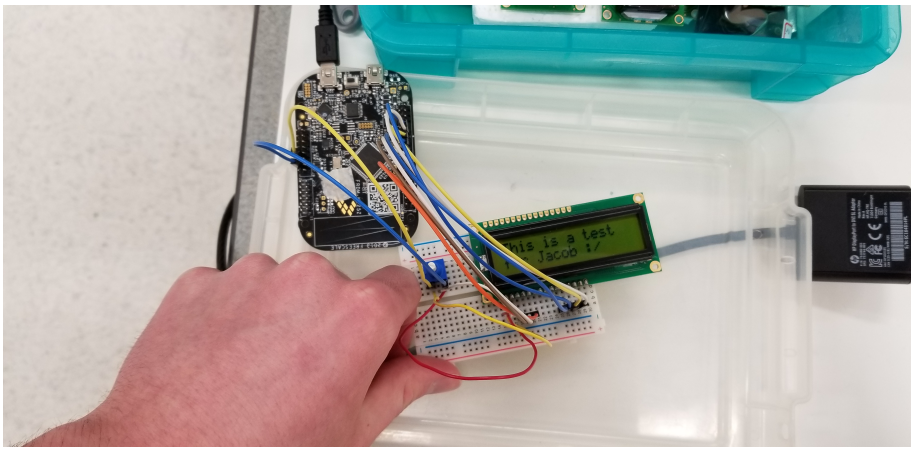


Figure 9: Wiring Configuration