CPE 490 Lab 7

## Goals

1. Learn how to interface with a small inexpensive LCD display module using C code and the dsPIC33 processor.
2. Use this example to explore communications between two microcontrollers.

## Equipment

You will be using the explorer 16 board with the MPLAB IDE and the MPLAB C30 compiler

## Overview

The interface between the dsPIC33 and the LCD module is a parallel communication port. The LCD module is from a company called Truly the model number is TSB1G7000. This module consists of the LCD, electronics to drive it, and a LCD Controller/Drive. The Controller/Driver is made by a company called Novatek it has a model number of NT7603. Both the Truly and Novatek data sheets are on Blackboard. The Novatek part is mimicking the original part made by Hitachi, that data sheet is also posted.

The wiring between the dsPIC33 and the LCD is detailed in the Explorer 16 user guide which has been posted on Blackboard under Microchip Documentation. The following description if from the user guide:

“The Explorer 16 board includes an alphanumeric LCD display with two lines of 16 characters each. The display is driven with three control lines (RD4, RD5 and RD15) and eight data lines (RE7:RE0). On PIC24 devices, the LCD is driven by the PMP module, not the I/O port. “

The Novatek data sheet gives all the detail on the electrical / communication interface. The following table is taken from the data sheet. The communications are broken into two types one is for commands and one is for data.

If the signal RS is cleared then you are sending a command to the LCD module. Commands are like clear the display, blink the display, set the RAM address that I will write to next. Commands are used to control the LCD module.

If the signal RS is set then you are reading (RW =1) or writing (RW=0) to the data memory. Typically writing an ASCII code to the Display Data RAM (DD RAM) the LCD will display that character. More than the standard ASCII characters are available see the data sheet for more codes. The DD RAM has a one to one correspondence to the characters shown on the LCD. The code found at address 0 will be displayed on the top most left character space in the LCD. Incrementing the address will move us to the next character to the right. Our display has 16 characters per line. Address spaces after 0x0F will not do anything unless we use shift commands (we are not for this lab). To write to the first character on the 2nd line use address 0X40.



A second memory space called Character Generator RAM is available. It allows you to describe 8 characters of your own design, so some graphic capabilities exist. We will not be using CR RAM in this lab.

To allow us to write messages from C code we will need a set of functions that will control the signals to allow us to issue commands, and write and read data memory. Lucky for us we have a start on this task. On blackboard there are some files that can be used. The C source file LCD.C gives us some functions we can use (prototypes copied from lcd.h):

/\*\*\*\*\*\* LCD FUNCTION PROTOYPES \*\*\*\*\*\*/

void Init\_LCD( void ); // initialize display

void lcd\_cmd( char cmd ); // write command to lcd

void lcd\_data( char data ); // write data to lcd

void puts\_lcd ( unsigned char \*data, unsigned char count );

/\*\*\*\*\* LCD COMMAND FUCNTION PROTOTYPES \*\*\*\*\*/

#define cursor\_right() lcd\_cmd( 0x14 )

#define cursor\_left() lcd\_cmd( 0x10 )

#define display\_shift() lcd\_cmd( 0x1C )

#define home\_clr() lcd\_cmd( 0x01 )

#define home\_it() lcd\_cmd( 0x02 )

#define line\_2() lcd\_cmd( 0xC0 ) // (0xC0)

The function init\_LCD() will initialize the LCD and set it up to be two a two line display. This function must be called before the LCD can be used.

The function lcd\_cmd(char cmd) will send the character cmd to the LCD as a command.

The function lcd\_data(char data) will send the character data to the LCD to be written in its data memory. The LCD controller will automatically increase the address to the next data memory location if the “Entry Mode Set” command is given with I/D =1 , it will decrement if I/D=0

The function put\_lcd(\*data, count) will display a string of characters directly, you just pass a pointer to the first character to display and then make count equal to the number of letters you want to display. A useful C call to this function would be :

const char mytext[] = "Embedded Systems";

const char mytext1[] = "Are Great";

puts\_lcd( mytext, sizeof(mytext) -1 );

line\_2();

puts\_lcd( mytext1, sizeof(mytext1) -1 );

The set of define statements just use functions to send certain commands to the LCD module. Refer to the previous table for details.

In order to take advantage of these already written routines , from blackboard copy the following files:

lcd.h

lcd.c

delay.h (needed by code in lcd.c)

delay.c

Make sure to include lcd.h in any source file that call these functions.

## Design 1

Design a C language program that displays “Embedded Systems” on line one and on line two it displays “Are Great”

### Verify

Have an instructor witness this. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Design 2

Starting with the previous program, have the display go blank when switch S3 is pressed.

Pressing S3 again will cause the first letter of the first line (‘E’) to be displayed. Pressing S3 again will display the first two letters of the display (‘EM’) and so on to the end of the second line.

Pressing S6 will cause the display to lose one character, until there is no more display.

Have an instructor witness this. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In you lab report, be sure to state how you verified that the code is working.