Lab 8 LCD Display Interface

This laboratory assignment accompanies the book, Embedded Microcomputer Systems: Real Time Interfacing, by Jonathan W. Valvano, published by Brooks-Cole, copyright © 2000.

Goals
- Design the hardware interface between a LCD display to a microcomputer,
- Create the low-level device driver that can be used in other applications.

Review
- Valvano Chapter 3 on Basic Handshake Mechanisms,
- Valvano Section 6.2.1 and 6.2.2 on Output Compare,
- Valvano Section 8.3 on LCD fundamentals,
- The chapter on the parallel port and output compare in the Motorola Reference Manual.

Starter files
- LCD12.C, LCD12.H, LCDTEST.C

Background

Microprocessor controlled LCD displays are widely used, having replaced most of their LED counterparts, because of their low power and flexible display graphics. This experiment will illustrate how a handshaked parallel port of the microcomputer will be used to output to the LCD display. The hardware for the display uses an industry standard HD44780 controller. The low-level software initializes and outputs to the HD44780 controller.

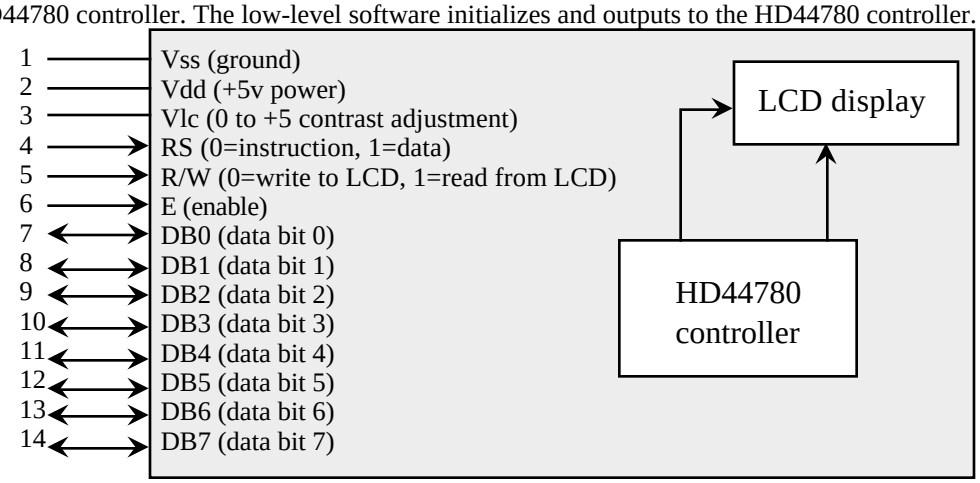

Figure 8.1. 1 by 16 LCD display.

There are four types of access cycles to the HD44780 depending on RS and R/W

RS	R/W	Cycle
0	0	Write to Instruction Register
0	1	Read Busy Flag (bit 7)
1	0	Write data from µP to the HD44780
1	1	Read data from HD44780 to the µP

Two types of synchronization can be used, blind cycle and gadfly. Most operations require 40 µs to complete while some require 1.64 ms. The example implementation shown in the LCD12.H, LCD12.C uses OC5 to create the blind cycle wait. A gadfly interface provides feedback to detect a faulty interface, but has the problem of creating a software crash if the LCD never finishes. The best interface utilized both gadfly and blind cycle, so that the software can return with an error code if a display operation does not finish on time (due to a broken wire or damaged display.)

In embedded systems like we use, it is OK to provide LCD12.H and LCD12.C files which the user can compile with their application. In our embedded system, linking will performed by the compiler. You are encouraged to modify/extend this example, and define/develop/test your own format. Normally, we group the device driver software into four categories. We will use interrupts in the later labs.

1. Data structures: global, protected (accessed only by the device driver, not the user.)

`OpenFlag` boolean that is true if the display port is open
initially false, set to true by `LCDOpen`, set to false by `LCDClose`
static storage (or dynamically created at bootstrap time, i.e., when loaded into memory)

2. Initialization routines (called by user)

```
#define LCDscroll 8
#define LCDnoscroll 0
#define LCDleft 0
#define LCDright 4
```

`LCDOpen` Initialization of display port
Sets `OpenFlag` to true
Initialize hardware, other data structures
Returns an error code if unsuccessful
hardware non-existent, already open, out of memory, hardware failure, illegal parameter
Input Parameters(mode) see the LCD data sheets for various options, e.g., scrolling
Output Parameter(none)
Typical calling sequence

```
if(!LCDOpen(LCDscroll|LCDright)) error();
```

`LCDClose` Release of display port
Sets `OpenFlag` to false
Release any dynamically allocated memory
Returns an error code if not previously open
Output Parameter(error code)
Typical calling sequence

```
if(!LCDClose()) error();
```

3. Regular I/O calls (called by user to perform I/O)

`LCDPutChar` Output an ASCII character to the LCD port
Returns an error code if unsuccessful
device not open, hardware failure (happens when a wire is loose)
Input Parameter(ASCII character)
Output Parameter(error code)
Typical calling sequence (you are free to change)

```
if(LCDPutChar(letter)) error();
```

4. Support software (protected, not directly accessible by the user).
None in this category for this lab, but there will be in later labs.

Preparation

Show the required hardware connections. Label all hardware chips, pin numbers, and resistor values. Write the low-level LCD device driver. You must have a separate LCD12.H and LCD12.C files to simplify the reuse of these routines. Write a main program that tests all features of the interface.

Procedure

You should look at the +5 V voltage versus time signal on a scope when power is first turned on to determine if the LCD "power on reset" circuit will be properly activated. The LCD data sheet specifies it needs from 0.1 ms to 10 ms rise time from 0.2 V to 4.5 V to generate the power on reset. Connect the LCD to your microcomputer. Use the scope to verify the sharpness of the digital inputs/outputs. Adjust the contrast potentiometer for the best looking display. Test the device driver software and main program in small pieces.

Checkout

You should be able to demonstrate all the "cool" features of your LCD display system.

Hints

1) Make sure the 14 wires are securely attached to your board.
2) One way to test for the first call to open is to test the direction register. After reset, the direction registers are usually zero, after a call to open, some direction register bits will be one.
3) Download from the class web site the files LCDTEST.C LCD12.H and LCD12.C files. These C language routine to do low level LCD output to Port H/J. Notice that it does not perform any input (either status or data), therefore it leaves DDRJ=0xFF, DDRH=0xFF. If you wish to include inputs, then you will have to toggle DDRH, so that PORTH is an output for writes and an input for reads.
4) Although many LCD displays use the same HD44780 controller, the displays come in various sizes ranging from 1 row by 16 columns up to 4 rows by 40 columns.