



Tutorial 4: Computer Components - I/O Interface

1 Introduction

This tutorial gives you experience in interfacing I/O components to a CPU. The LCD display on the NanoBoard will be interfaced to the same CPU which was used for Tutorial 3. There are two possible methods of implementing the LCD display interface: one using a LCD controller provided by Altium and without it. You are expected to implement the first method and if you have time, implement the second.

As part of the preparation for this tutorial, read through the data sheet for the LCD display which can be found in the Lecture Notes Part II Section C1. Also read through the information on the Altium *LCD16X2 LCD Controller* which can be found in the Lecture Notes Part II Section B2.

To access the external data memory, where the LCD display will be interfaced, you need to use a special directive in C. The best way to do this is to declare a pointer to the external data memory space and assign the pointer to the first address. The pointer can then be dereferenced by treating it as a pointer to an array. An example of this is as follows:

```
__xdata volatile char *LCD = 0x0000; // Declare pointer into external
                                   // data memory starting at address 0x0000

LCD[0x0000]=0x12; // Write 0x12 to external data memory at address 0x0000
T=LCD[0x0001];   // Read external data memory at address 0x0001
```

As covered in lectures, a consideration when writing to or reading from I/O devices is that the status of the I/O has to be checked first. With the LCD display, there is a busy flag that needs to be read before writing to the display. You need to take this into account when writing your test code.

More information on the actual controller used in the LCD display can be found at:

- KS0066U 16COM / 40SEG DRIVER & CONTROLLER FOR DOT MATRIX LCD
(<http://www.hantronix.com/download/ks0066u.pdf>)

The LCD display is 16 characters wide by two lines, however the internal controller assumes that there are 40 characters per line. This needs to be taken into account when writing test code for the interface without the Altium LCD controller.

2 LCD Display Interface using the LCD16X2 LCD Controller

The first version of the interface uses the *LCD16X2 LCD Controller* provided by Altium. This connects directly to the LCD display. It handles the LCD initialization at power up and the writing of characters to the LCD display given the position on the screen.

The way in which the controller is connected to the LCD display is shown in Figure 6 of the *LCD16X2 LCD Controller* documentation. However, to interface the controller to the CPU also requires some extra glue logic. A block diagram showing the glue logic to the controller and the LCD display is shown in Figure 1. Note that the LCD controller re-

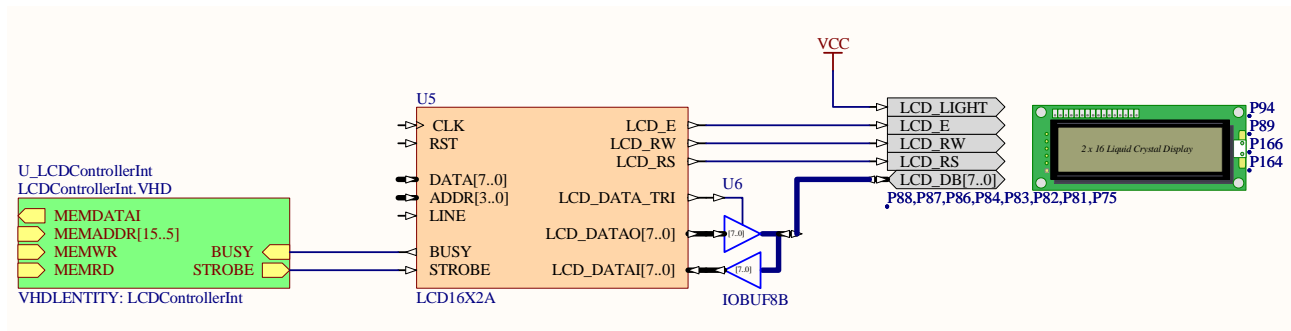


Figure 1: Interface using the LCD controller

quires 32 addresses (16 characters by two lines). The lower order address lines can connect directly to the LCD controller. The higher order address lines have to be decoded to select the correct address range. The same signals as used for the CPU can be used for **CLK** and **RST**.

The data is read by the controller using **CLK** when **STROBE** is asserted.

There is also a **BUSY** output signal which needs to be accessible by the CPU. This could be mapped to one or all of the addresses set aside for the LCD controller. Software needs to sample this signal to check that the next character can be written.

Use the following procedure to implement the interface:

- Draw a timing diagram showing the inputs and outputs for the glue logic. Include a write cycle to the LCD display and a read of the **BUSY** flag.
- Design the glue logic using VHDL. Place the LCD controller in the external memory address range 0x0000 to 0x001f. The busy signal should be mapped to one or all of these addresses. There are three parts to the interface:
 - Address decoding
 - Generation of the **STROBE** signal
 - Reading of the **BUSY** signal
- Simulate the glue logic to ensure correct operation.
- Implement the design on the NanoBoard using the circuit from Tutorial 3 and write suitable test code.

A good test to see that everything is working is to have a message flash on the display continuously.

3 Direct CPU Interface to the LCD Display

Attempt this task if you have time.

It is possible to connect the CPU with some glue logic to the LCD display without using the LCD controller used previously. Study the timing diagram for the CPU again, as you did for Tutorial 4, and the timing diagram for the LCD display. You should also study the table for the LCD display which describes the commands. Note that the busy flag must be checked before writing to the LCD display.

One aspect of the design that you will notice when looking at the timing diagrams is that it is not possible to directly connect the CPU to the LCD display. The **LCD_RS** and **LCD_RW** signals will have to be generated separately. That is, a register (latch) will need to be implemented to drive these signals. The **LCD_DB** and **LCD_E** signals can be generated directly from the CPU.

A block diagram for the interface is shown in Figure 2. Note that the buffering of the data lines is done as part of the

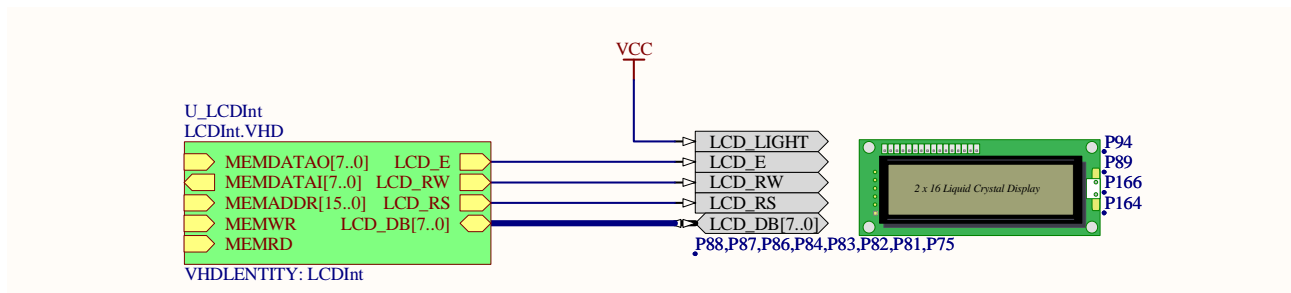


Figure 2: Direct interface between the CPU and LCD display

interface whereas previously the buffer was a separate component,

Use the following procedure to implement the interface:

- Confirm from the timing diagram that the CPU can't be connected directly to the LCD display. State what signals timing cannot be met.
- Draw a timing diagram for a write to the latch which sets **LCD_RS** and **LCD_RW**.
- Draw a timing diagram showing a read and a write cycle from the CPU to the LCD display assuming that **LCD_RS** and **LCD_RW** have been set. Show all the inputs and outputs of the LCD display interface.
- Design the interface using the external data address 0x0000 to access the LCD display and address 0x0001 for the latch to generate the signals **LCD_RS** and **LCD_RW**. There are four parts to the interface:
 - Address decoding
 - Data signal routing
 - Latch to generate **LCD_RS** and **LCD_RW**
 - Generation of **LCD_E** signal

Ensure that there are no clashes on any of the data lines.

- Simulate the design. This includes the operation of the latch and the signals going to the LCD display.
- Implement the design on the NanoBoard by writing a simple test program.

You will have to also set the operating parameters of the LCD display at the start of your program. These can be determined from the documentation (otherwise try writing the following sequence to the instruction register: 0x01, 0x06, 0x0c, 0x38, 0x80).

You will notice that the code required to access the LCD display is much longer than that used for the first interface. It is a good example of how an intelligent interface can reduce the load on the CPU.