UNIVERSITY OF CALIFORNIA AT BERKELEY
COLLEGE OF ENGINEERING
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Checkpoint 3
LCD Controller

1.0 Motivation

In this checkpoint you will add a significant extra feature to the base audio player you have created over the past month: track title and artist information.

To do this you will need to create a large Finite State Machine to properly decode incoming track information sent over Ethernet, and send it to the character LCD on the CaLinx2 boards, with the correct timing. You will also need to create pair of cascaded state machines to extract LCD control information such as backlight brightness from the incoming data stream.

As in checkpoint 2, you will be building a controller to match a datasheet, this time for the Hitachi HD44780 LCD controller chip, a standard IC used on nearly every character LCD in the world.

2.0 Introduction

Aside from providing the nifty track title information, this checkpoint will give you experience in working with the very common interface provided by the HD44780. Because this interface was originally designed to connect a microprocessor, you will find most likely find that a general Finite State Machine is the simplest path.

In previous checkpoints we dealt with Ethernet and Audio, both of which are designed to efficiently stream data from one place to another, with a minimum of control logic. However the LCD has a lower bandwidth requirement (it only works with ASCII characters) and we would like the ability to control is somewhat (position our text appropriately), and so the HD44780 interface is slower, and requires more state than a simple counter can easily provide.

2.1 Checkpoint #3 Overview

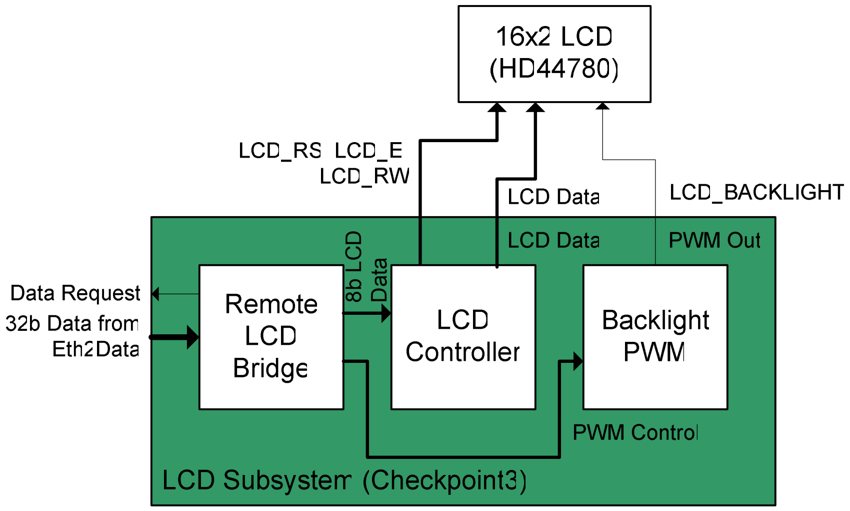


Figure 1: Checkpoint #3 Overview

Shown in Figure 1, above the LCD subsystem consists first of a “Remote LCD Bridge” or simple FSM designed to extract backlight control information from the payload of LCD Ethernet packets, as well as converting from the 32b words received from your Eth2Data module, down to 8b words meant for the LCD controller.

Once the escape sequences (similar to those in C or Java) which control the backlight have been removed, the remaining bytes will be sent to the LCD controller, whose job it is to interact on a cycle to cycle basis with the HD44780, enforcing the timing requirements of that chip, and providing a second set of escape sequences to allow control over the position of the cursor and such on the LCD.

2.2 Pulse Width Modulation

The LCD module on the CaLinx2 boards is equipped with a green LED backlight. In order to make a more visually appealing display (as in the Apple iPod), the brightness of this backlight can be varied.

The traditional method of varying a light bulb or LED’s brightness, which you learned in Physics 7C and EE40, would be to **change the voltage** across the device, or perhaps the current through it. However, given that we have a digital FPGA, this would present a problem: an external ADC would be required to generate the proper voltage.

Instead we will use **Pulse Width Modulation**. By turning the backlight on and off very rapidly, we can take advantage of the fact that the human eye has a **relatively slow response rate** to make the backlight look dimmer. In order to do this we will feed a fixed **frequency but variable duty cycle signal** into the backlight.

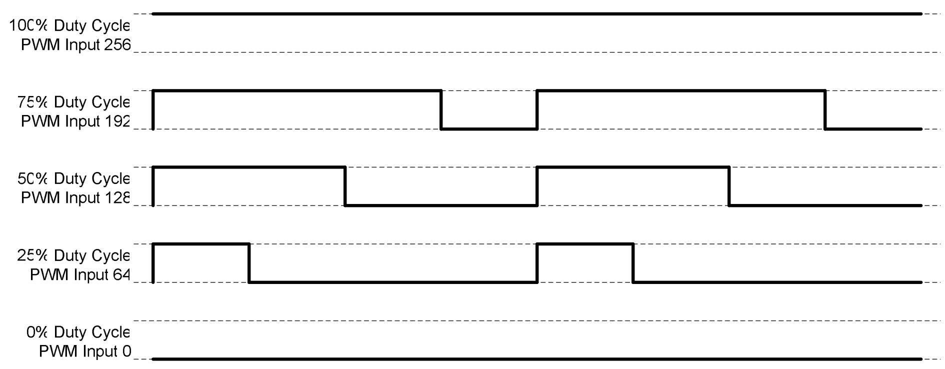


Figure 2: Example PWM Outputs

Shown in Figure 2, above, are **example PWM outputs**. The idea behind the PWM module is that it should be capable of taking a **9bit input** and generating a variable width pulse based on this input.

The reason for this 9bit input will become apparent upon closer examination: if we use an **8bit counter to divide our period** (whatever it may be) **into 256 pieces, then it will require 9bits to allow the pulse width** (and brightness) **to vary from always off,** as shown at the bottom of Figure 2, **to always on**, shown at the top of Figure 2. Notice that **when the MSB is `1'b1`, none of the other bits matter**: the output should always be `1'b1`.

2.4 HD44780

First off, you should not rely solely on the text of this checkpoint write up for information about the HD44780, you should read the datasheet thoroughly, as it provides a complete technical description and full details. This text is merely meant to highlight those issues you should be aware of.

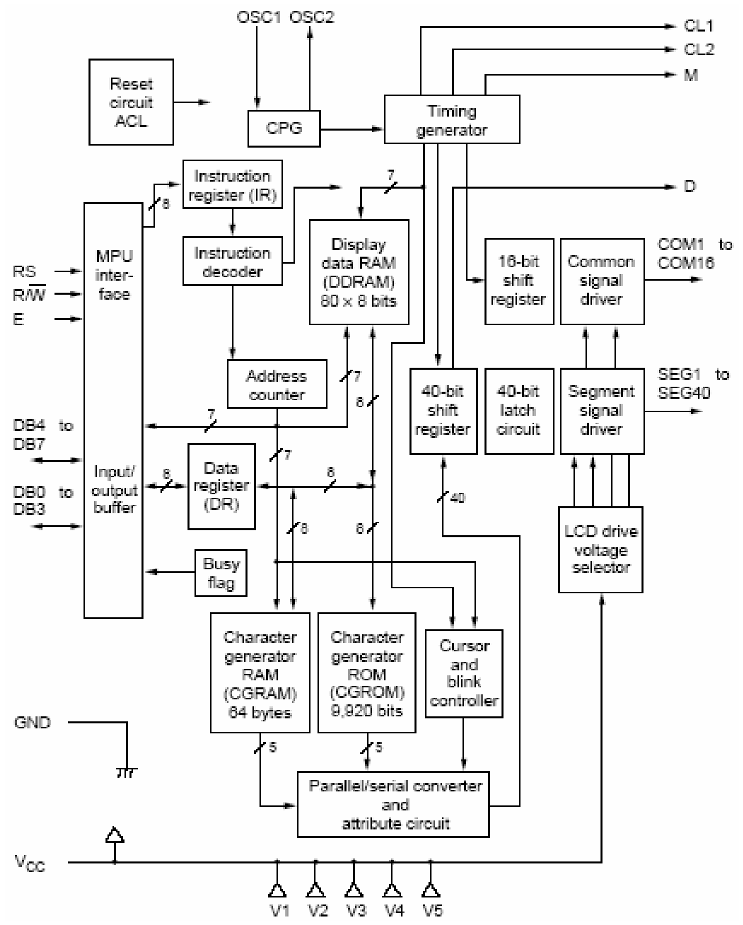


Figure 3: HD44780 Block Diagram

2.4.1 Theory of Operation

The HD44780 is based on a simple interface: an 8-bit data bus and three control lines. The RS and RW control lines provide command information, telling the HD44780 what operation to perform. The E control line acts as a combined Clock and Enable input, often called a Strobe. The HD44780 will perform the requested command on the falling edge of the E control line.

For detailed timing diagram please refer to Figures 4 & 5, below which show both the timing diagram and parameters for the control and strobe. You will need to pay close attention to these timing constraints, as failure to do so will cause the LCD to behave incorrectly and in strange ways. You should also note that the LCD is very slow, with timing numbers measured in microseconds, where your circuit operates in nanoseconds (1/27MHz = 37ns).

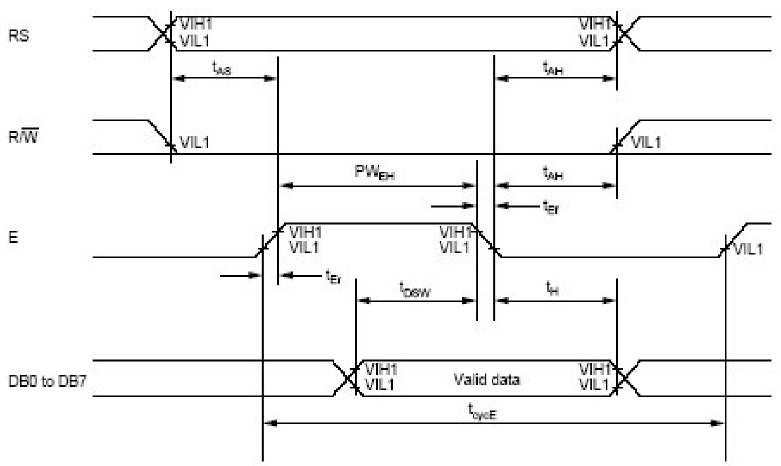
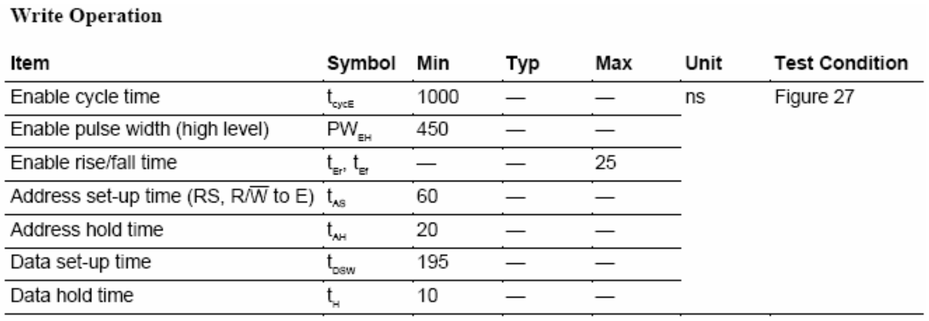


Figure 4: HD44780 Write Timing



Write Operation

Item	Symbol	Min	Typ	Max	Unit	Test Condition
Enable cycle time	t_{cycE}	1000	—	—	ns	Figure 27
Enable pulse width (high level)	PW_{EH}	450	—	—		
Enable rise/fall time	t_{Er}, t_{Ef}	—	—	25		
Address set-up time (RS, R/W̄ to E)	t_{AS}	60	—	—		
Address hold time	t_{AH}	20	—	—		
Data set-up time	t_{DSW}	195	—	—		
Data hold time	t_H	10	—	—		

Figure 5: HD44780 Write Operation Timing

There are two details which are not covered by the above timing diagram:

- **After any operation is issued** using the timing in Figure 4, **you must continuously check the busy flag**. Your LCD controller may **not** issue a second operation to the HD44780 **until after the busy flag returns to 1'b0**.
- Even when the busy flag has returned to 1'b0, **you will need to wait 4us, before issuing another command**. This is detailed in **Table 6 and Figure 10 on page 192 of the HD44780 datasheet**.

2.4.2 Initialization

Most of the complexity of interfacing with the HD44780 has been removed by our use of escape characters allowing the audio server to control the display, as detailed in section 4.3 LCD.v. However the LCD controller you build for this checkpoint must still properly initialize the display or it will not operate at all.

Shown in Figure 6 below is the initialization sequence taken from page 212 of the HD44780 datasheet

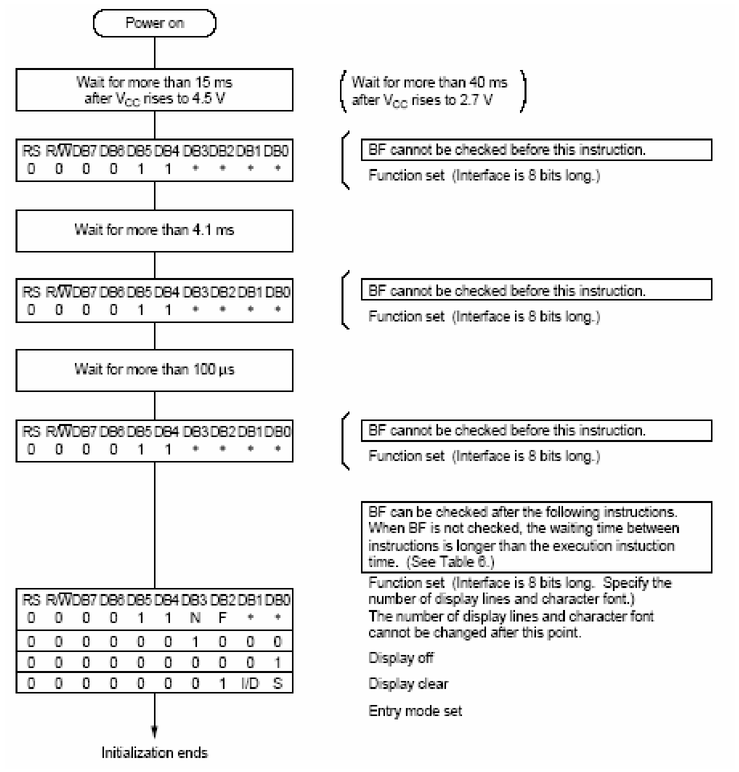

Power on
Wait for more than 15 ms after V_{CC} rises to 4.5 V
Wait for more than 40 ms after V_{CC} rises to 2.7 V
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0 0 0 0 1 1 * * * *
BF cannot be checked before this instruction.
Function set (Interface is 8 bits long.)
Wait for more than 4.1 ms
Wait for more than 100 μs
BF can be checked after the following instructions. When BF is not checked, the waiting time between instructions is longer than the execution instuction time. (See Table 6.)
Function set (Interface is 8 bits long. Specify the number of display lines and character font.) The number of display lines and character font cannot be changed after this point.
0 0 0 0 1 1 N F * *
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1 I/D S
Display off
Display clear
Entry mode set
Initialization ends

Figure 6: Initialization Sequence

However this initialization sequence can be simplified for our purposes, for example we need not issue the display control commands shown at the bottom: the audio server will take care of those. As such the recommended initialization sequence is:

1. Issue a Function Set Command
   a. Interface is 8bits
   b. Display is 2 lines
   c. 5x8 Character Font
2. Wait 4.1ms
3. Issue a Function Set
4. Wait 4.1ms
5. Issue a Function Set
6. Wait for the busy flag to go low (as detailed in section 2.4.1 Theory of Operation)
7. Wait for the address change (as detailed in section 2.4.1 Theory of Operation)
8. LCD is ready for use, begin handling incoming data

3.0 Prelab

Please make sure to complete the prelab before you attend your lab section. **You will not be able to finish this lab in 3hrs.**

1. **Read this handout thoroughly**. Pay particular attention to sections **2.0 Introduction** and **4.0 Lab Procedure**.
2. **Read the HD44780 Datasheet**

   a. You can find this on the website: http://www-inst.eecs.berkeley.edu/~cs150/sp05/Documents.htm.
3. **Examine the Verilog** provided for this weeks lab.
4. **Take your design review seriously**.
   a. Again we are expecting you to design your own circuits from scratch.
   b. You will need your design review to get feedback on the things you did well, as well as the problems your design has.
5. **Start early**!
   a. While this is a one week checkpoint, there are a number of modules you need to build. Each one may be simple, but each one may have bugs.

4.0 Lab Procedure

Remember to **manage your Verilog, projects and folders well**. Doing a poor job of managing your files can cost you **hours of rewriting code**, if you accidentally delete your files.

Below are sections describing the various modules you may work with for this lab. Note that you will need at least one instance of each of these modules.

4.1 PWM.v

This module should be very simple to design and build. A recommended set of I/O ports is listed below. You should feel free to modify this as you please. For example you may wish to incorporate the fading functionality directly into this module, in which case you may want a `FadeStart` input.

Signal	**Width**	**Dir**	**Description**
`Clock`	1	I	`Clock` input
`Reset`	1	I	`Reset` input
`Load`	1	I	`Load` a new PWM value into an internal register
`In`	9	I	New PWM value
`Out`	1	O	PWM output (See Figure 2)

Table 1: Example Port List for PWM.v

4.2 RemoteLCD.v

This module has two duties as outlined below and in lab lecture:

1. This module is responsible for requesting words from the Eth2Data module and converting them from 32b wide to 8b wide, as needed by the LCD.v module.
2. RemoteLCD.v should also translate the 0xFE and 0xFF backlight control escape sequences, and properly issue commands to the backlight PWM.
   a. 0xFE marks the start of a song, and should be use to start fading the backlight.
   b. 0xFF should set the backlight to the value specified by the next byte. Remember the PWM will accept 9bit inputs, therefore this

escape sequence cannot turn the backlight full on (full on is 9'h100, this can only go to 8'hFF).

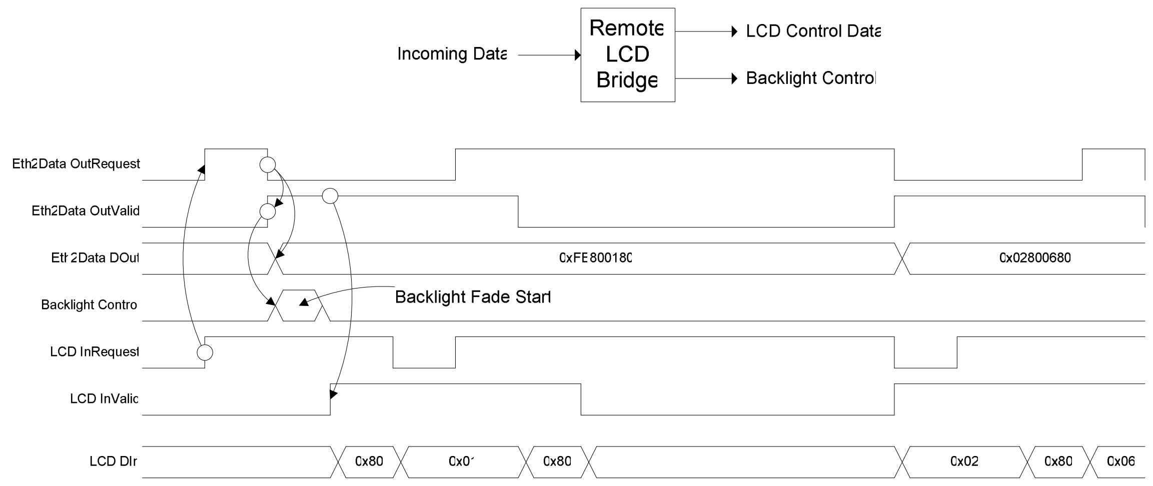


Figure 7: Operation of RemoteLCD.v

Signal	Width	Dir	Description
`DIn`	`32`	I	32b Data word in from Eth2Data
`InValid`	`1`	I	Indicates that `InRequest` was successful
`InRequest`	`1`	O	Request a new word from Eth2Data
`Reset`	`1`	I	`Reset` input
`Clock`	`1`	I	`Clock` input (27MHz)
`LCD_DB`	`8`	I/O	Bidirectional data bus to the HD44780
`LCD_E`	`1`	O	Strobe output to HD44780
`LCD_RW`	`1`	O	Read/Write control bit
`LCD_RS`	`1`	O	Register (Command/Data) Select
`LCD_BACKLIGHT`	`1`	O	LED Backlight PWM Output

Table 1: Port Specification for RemoteLCD.v

This should also be the top module of your design instantiating the PWM and LCD.v modules, and properly connecting them.

4.3 LCD.v

Most of your design and debugging time will be spend working on this module: the LCD controller which must handle initialization, LCD escape sequences and command timing to interface between your project and the HD44780.

Signal	Width	Dir	Description
`DIn`	`8`	I	8b Data word in from RemoteLCD.v
`InValid`	`1`	I	Indicates that `InRequest` was successful
`InRequest`	`1`	O	Request a new word from Eth2Data
`Reset`	`1`	I	`Reset` input
`Clock`	`1`	I	`Clock` input (27MHz)

LCD_DB	8	I/O	Bidirectional data bus to the HD44780
LCD_E	1	O	Strobe output to HD44780
LCD_RW	1	O	Read/Write control bit
LCD_RS	1	O	Register (Command/Data) Select

Table 2: Port Specification for LCD.v

Aside from issuing the initialization sequence as listed in section 2.4.2 Initialization, the main duties of this module are to ensure the command timing detailed in the HD44780 datasheet and to translate escape sequences.

Most of the bytes appearing at DIn, should be written, unmolested to the HD44780, by placing them on the LCD_DB bus and strobing the LCD_E wire to write the bytes in DDRAM at the current cursor location.

However, any incoming byte where the top six bits are 6'h20, should be treated as the first half of an escape sequence. The lowest two bits of this byte must be registered and saved. When the second byte arrives, the lowest two bits of the first byte should be sent to LCD_RS and LCD_RW while the second byte is put on LCD_DB and LCD_E is pulsed. This will allow the server to issue various display commands.

For example escape sequences, please refer to the LCDTestbench.v file included in this Checkpoint.

5.0 Checkpoint3 Checkoff

Name: ______________________________ SID: ______________________________
Name: ______________________________ SID: ______________________________
Section: ______________________________

I	Track Titles & Artist	__________ (35%)
II	Backlight Control	__________ (35%)
III	Clean Verilog & Interfaces	__________ (10%)

IV	Hours Spent:	__________
V	Total:	__________
VI	TA:	__________

RevA – 3/7/2005	Greg Gibeling	Created a new checkpoint