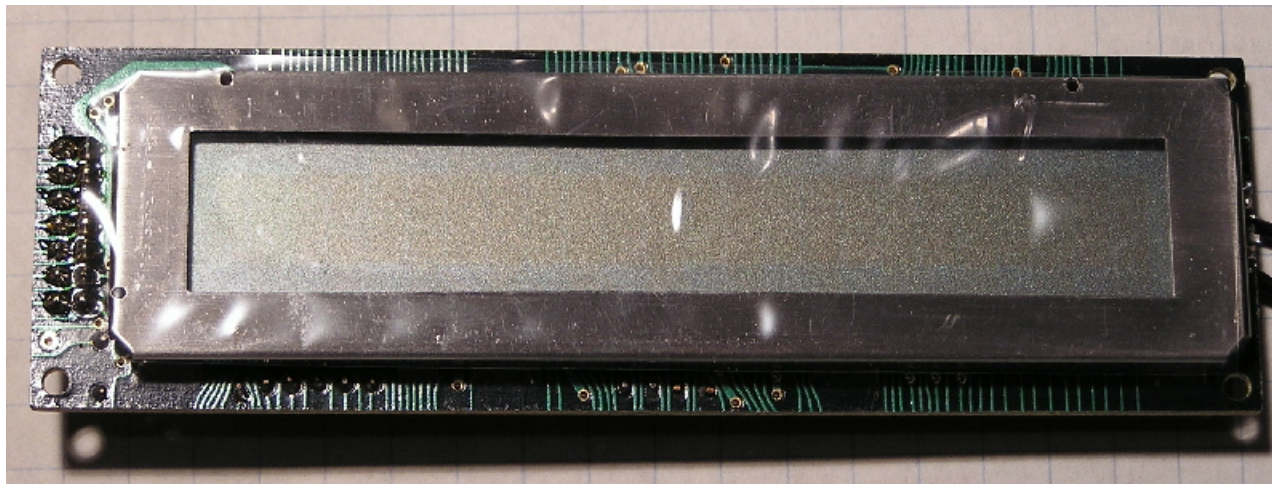


Care and Feeding of Char LCD Displays



Overview

LCD interface mechanical

LCD interface electrical

Hardware & software interface options

Timing considerations

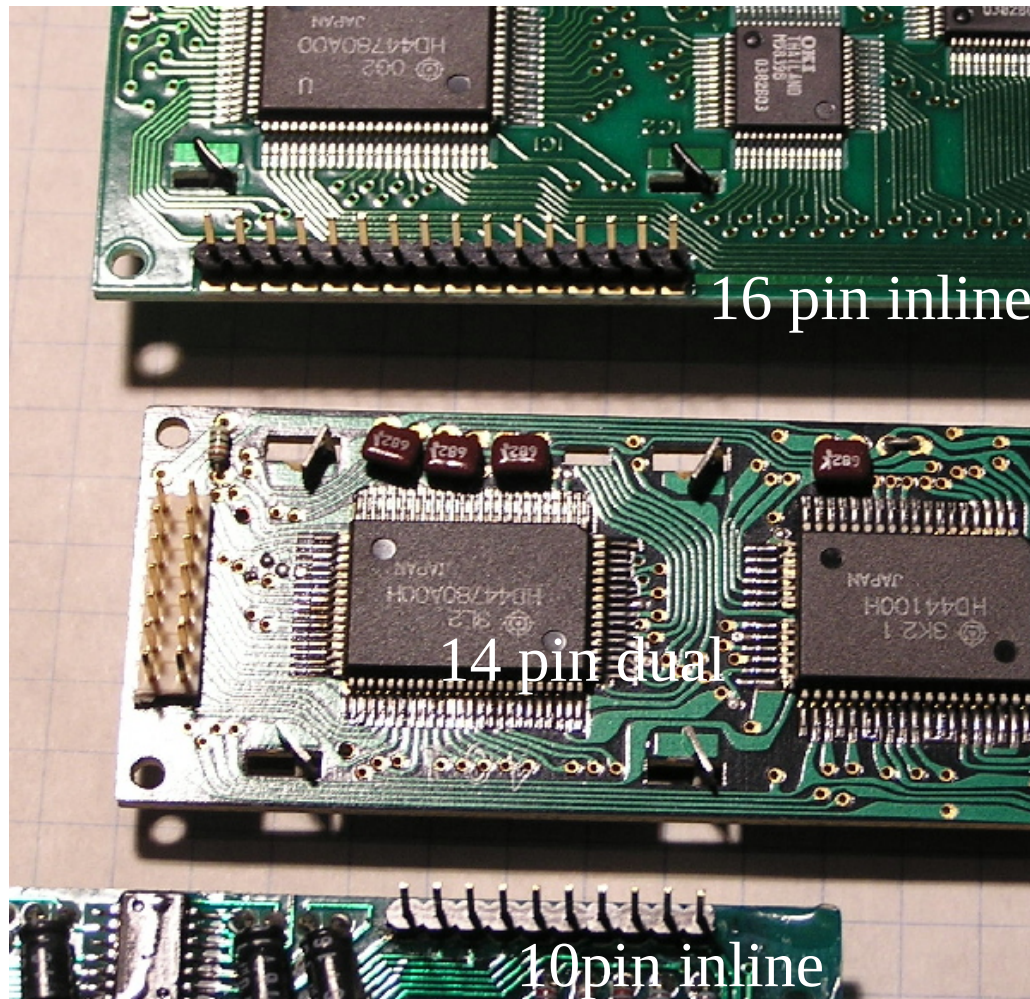
LCD commands

Display addressing

Parallel port options

Software

Interface - Mechanical



There are several different styles of interface layouts

For a given layout the pins will have the same function

The same pin numbers will have the same function across layouts

Interface - Electrical

Pin#	Pin#	Symbol	Level	Function
8 bit	4 bit			
====	=====	=====	=====	=====
1	1	Vss	GND	Ground
2	2	Vcc	+5V	Module power
3	3	Vee	NOTE1	Liquid crystal drive
4	4	RS	H/L	Register select, H=data, L=command
5	5	R/~W	H/L	Read/Write, H=read , L=write
6	6	E	H->L	Edge-sensitive Enable
7	-	DB0		Data bit 0 (not used in 4 bit mode)
8	-	DB1		Data bit 1 (not used in 4 bit mode)
9	-	DB2		Data bit 2 (not used in 4 bit mode)
10	-	DB3		Data bit 3 (not used in 4 bit mode)
11	7	DB4		Data bit 4
12	8	DB5		Data bit 5
13	9	DB6		Data bit 6
14	10	DB7		Data bit 7

NOTE1: On standard modules Vee is between GND and 5V; on temperature extended modules it is between GND and -7V

Interface - Electrical

Back Lighting

Power may be additional interface pins or separate connections

LED

simple, high power

Electro-Luminescent

complex drive circuit, less power

Cold Cathode Fluorescent

**complex drive circuit, less power,
not usually found on char displays**

Interface - Options

Hardware

8 bit

- more interface lines
- faster

4 bit

- 4 less interface lines
- needs 2 write operations

Software

Write and wait

- Write data and wait for max time
- Simple

Check busy

- Write and check status
- May be faster
- Requires bidirectional data bus

Interface - Timing

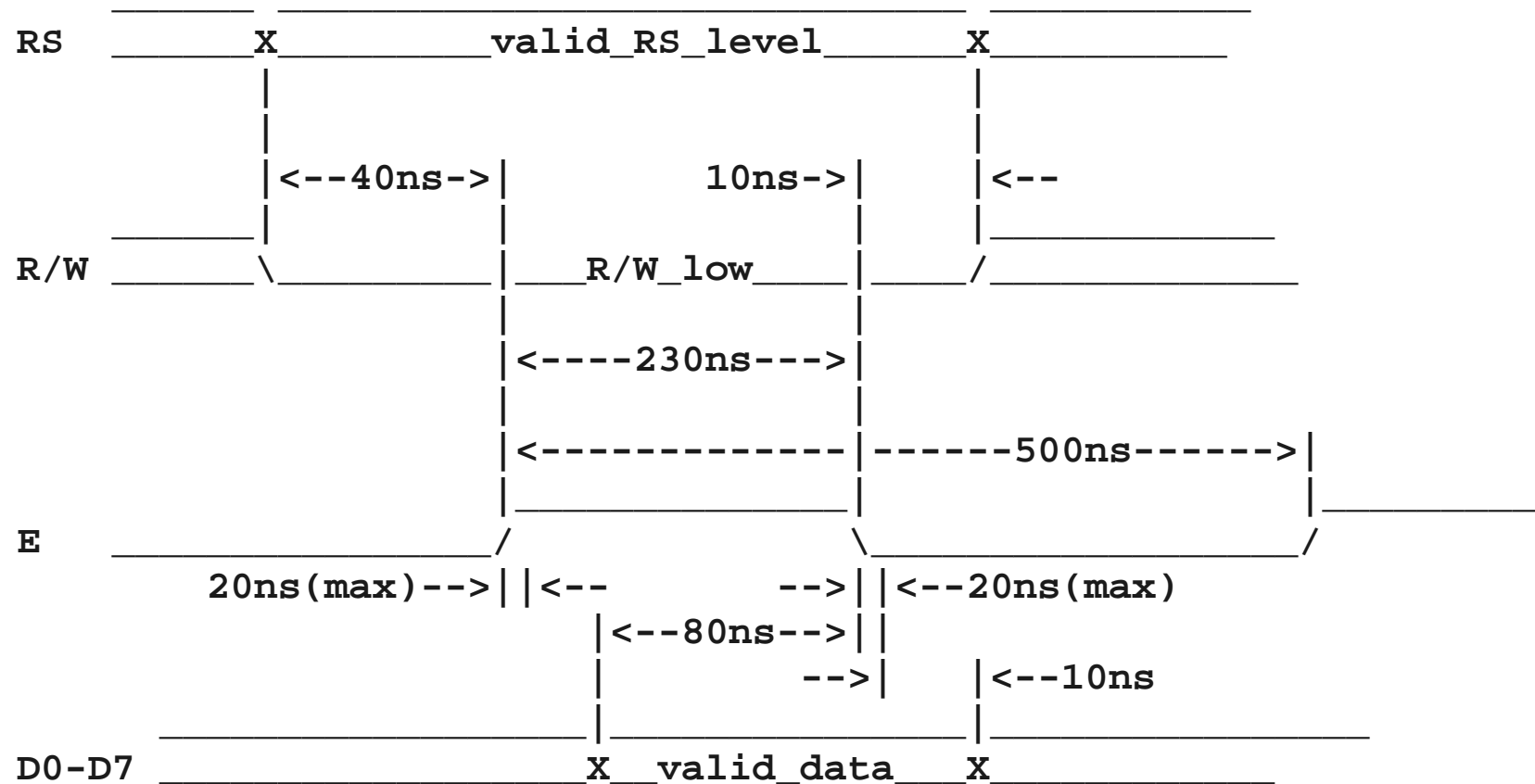
Control Signal Timing

If you look at the spec sheet you will find diagrams like the next page. These are important if you are running at the maximum speed the display can support, such as memory mapped I/O with a micro-processor.

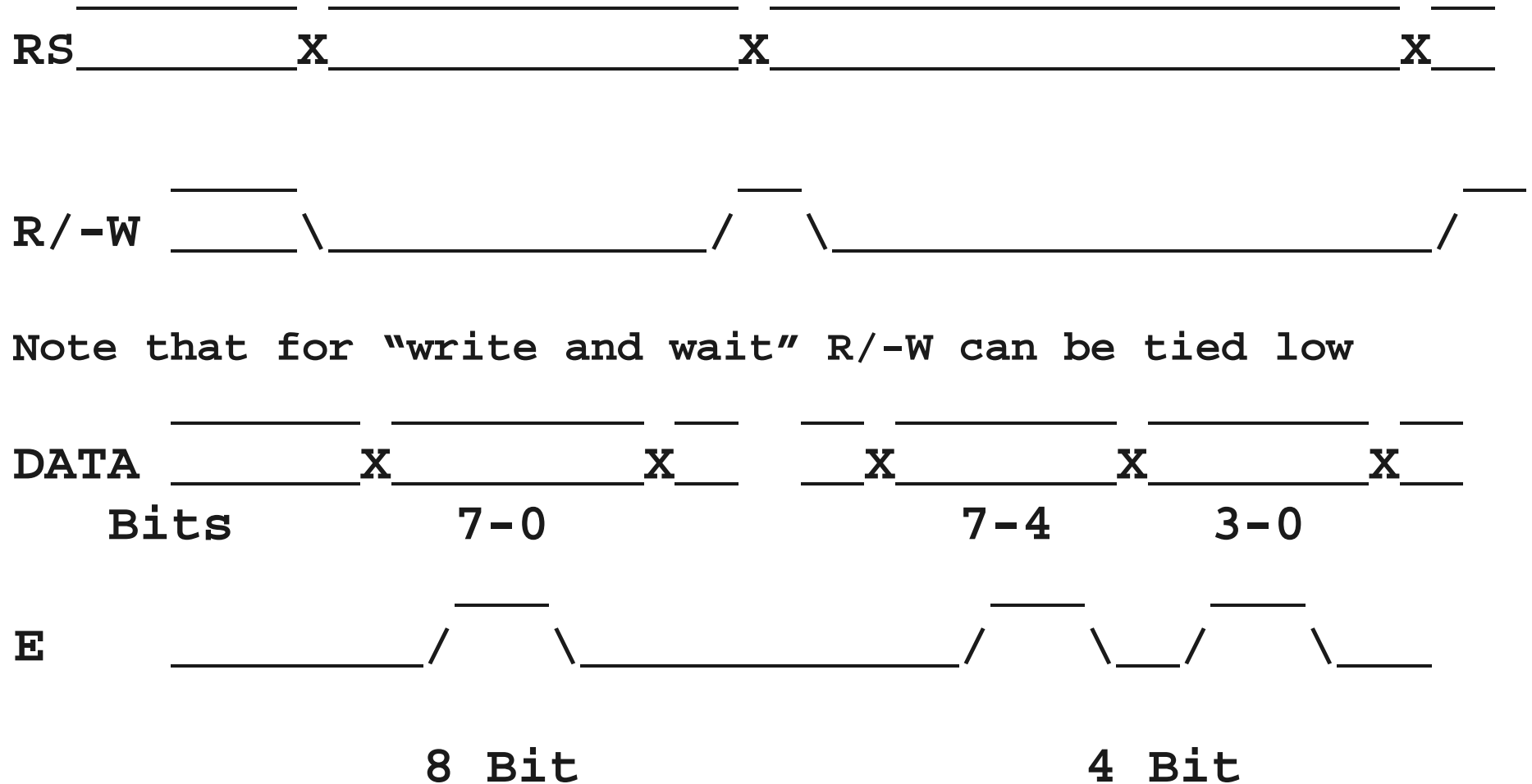
For a parallel port or port based I/O on a micro-processor it is simpler (slower) we usually won't come close to these limits. In compatibility mode, the parallel port is simulating talking to a printer over an 8MHz AT bus. The minimum transition is about 500ns even on a GHz machine.

Write timing – LCD controller

WRITE:



Write timing - Port I/O



LCD Commands

D7	D6	D5	D4	D3	D2	D1	D0	Instruction/Description
0	0	0	0	0	0	0	1	Clear Display
0	0	0	0	0	0	1	*	Return to Home Position
0	0	0	0	0	1	ID	S	Set Cursor Move Direction
0	0	0	0	1	D	C	B	Enable Display/Cursor
0	0	0	1	SC	RL	*	*	Cursor/Display Shift
0	0	1	DL	N	F	*	*	Set Interface Length
0	1	A	A	A	A	A	A	Set CGRAM Address
1	A	A	A	A	A	A	A	Set Display Address

ID 0 = Decrement cursor position

S 0 = No display shift

D 0 = Display off

C 0 = Cursor off

B 0 = Cursor blink off

SC 0 = Move cursor

RL 0 = Shift left

DL 0 = 4-bit interface

N 0 = 1/8 or 1/11 Duty (1 line)

F 0 = 5x7 dots

A Address

1 = Increment cursor position

1 = Display shift

1 = Display on

1 = Cursor on

1 = Cursor blink on

1 = Shift display

1 = Shift right

1 = 8-bit interface

1 = 1/16 Duty (2 lines)

1 = 5x10 dots

LCD Commands

RS is register select 0=command, 1=data.

R/-W is read/write control 0=write,1=read.

Commands are issued with R/S=0 and R/-W=0. Put the command on the data lines, and toggle E from L->H->L.

The Clear Display and Return Home commands can take up to 1.64ms, the rest take ~40usec.

Even with only 8 commands, it can be a bit confusing with all the options
We only need a subset of the commands to initialize the LCD.

1. Set Interface Length - use 0011 1000 8 bit interface, 2 lines, 5x7 dot characters
2. Enable Display/Cursor - use 0000 1100 display on, no cursor
3. Clear Display - use 0000 0001 no options
4. Set Cursor Move Direction - use 0000 0110 increment cursor, no shift

Then we can use Set Display Address and data writes to put characters on the display.

Writing Data

After initializing per the previous slide writing starts at the first position on line 1. To write data, we set RS=1, R/-W=0, put the ASCII character on the data lines, and toggle E from L->H->L.

The controller has 80 bytes of data memory that are organized differently depending on the mode.

In single line mode it starts at 0 offset and is linearly addressed to offset 0x4F. In some documents, you will see memory addressed as 0x80 – 0xCF; this is the actual command to set the memory address (remember command bit 0x80 indicates display address set).

In 2 line mode the display memory is split with the first 40 bytes starting at offset 0 (0x80) and the second line at offset 0x40 (0xC0).

Display Addressing

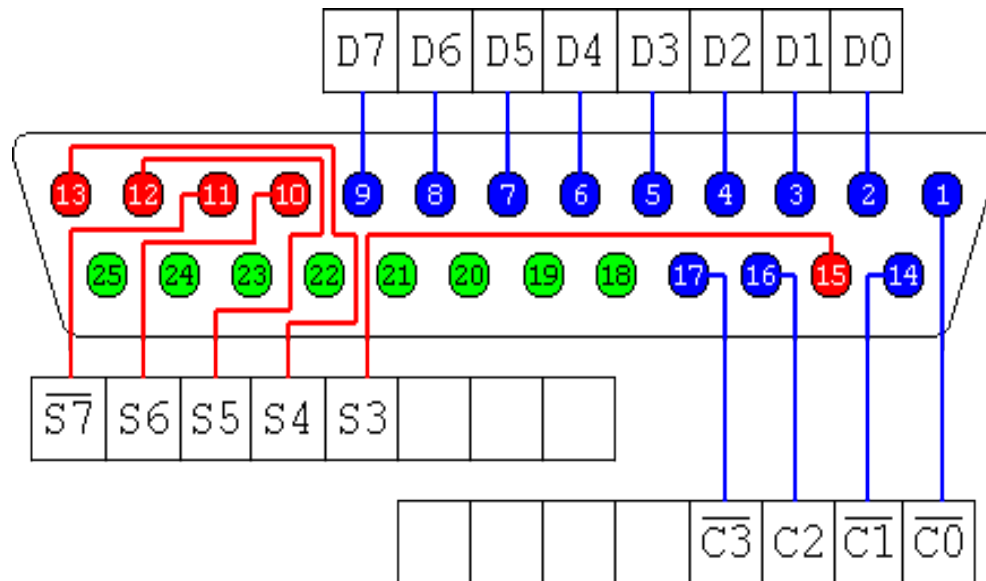
The base controller can display 2 lines x 8 char. For each additional 8 characters, a display driver is added to handle another 2 lines x 8. For a 1 line x 16 characters, there are two possibilities. There could be the controller and a display driver, in which case it would look like a single line display. The other case is to use only the controller and treat it as a 2 line x 8 character display with the 2 “lines” on one line. This makes a difference in how you address the second 8 characters!

Four line displays, for example 4 lines x 20 characters, the other lines are mapped onto lines 1 and 2. Line 3 is the second half of line 1 and line 4 is the second half of line 2.

Line 1	0x80 – 0x93
Line 2	0xC0 – 0xD3
Line 3	0x94 – 0xA7
Line 4	0xD4 - 0xE7

Parallel port signals

So how do we connect to the parallel port?



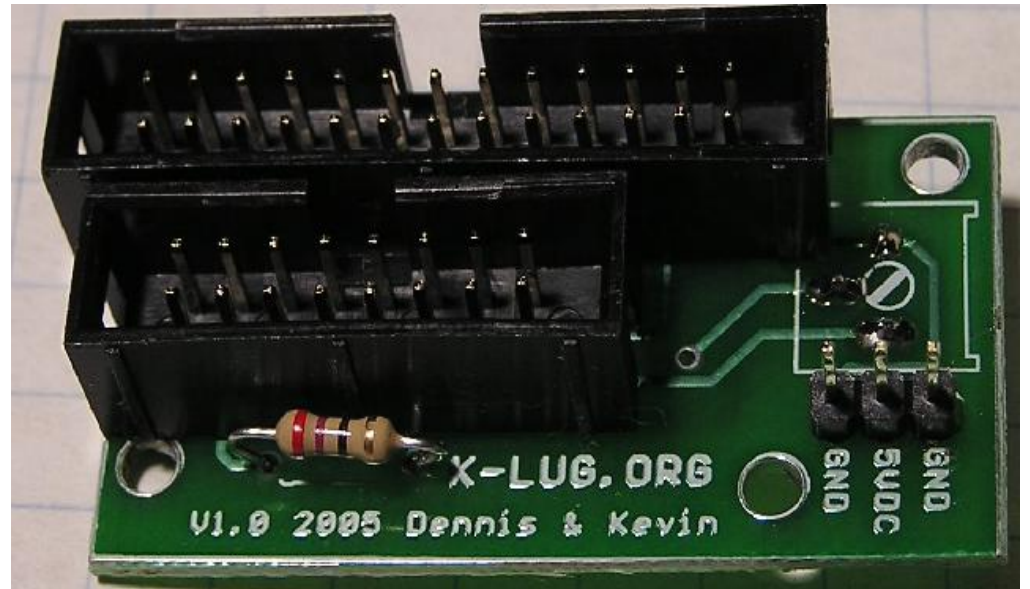
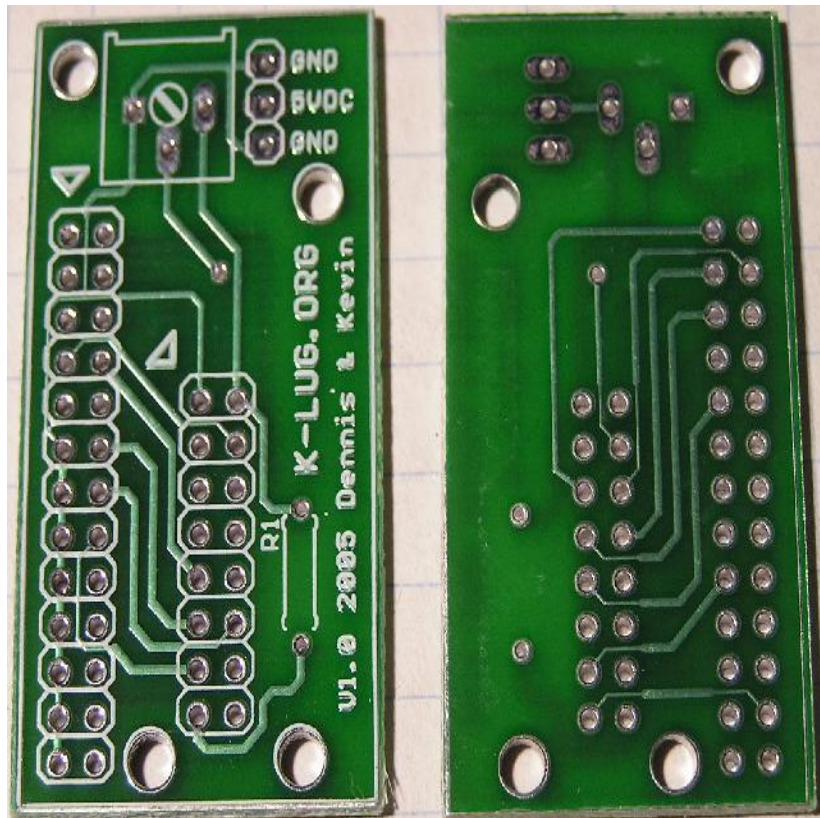
Data is easy just connect the parallel port data to LCD data.

The LCD requires 2 or 3 control lines, the parallel port has 4 available; so there are several possibilities. We will use the connection used by Winamp.

Parallel port connection

D-Type Pin No	Name	Direction	Hardware Register	Inverted	LCD Sig	LCD pin
1	-Strobe	Out	Control	Yes	E	6
2	Data 0	Out	Data		Data0	7
3	Data 1	Out	Data		Data1	8
4	Data 2	Out	Data		Data2	9
5	Data 3	Out	Data		Data3	10
6	Data 4	Out	Data		Data4	11
7	Data 5	Out	Data		Data5	12
8	Data 6	Out	Data		Data6	13
9	Data 7	Out	Data		Data7	14
14	-Linefeed	Out	Control	Yes	R/-W	5
16	Initialize	Out	Control		RS	4
18 – 25	Ground	Gnd			Gnd	2

K-LUG Printer Port Board



K-LUG 26 pin connector

	1	2	E
R/~W	3	4	D0
	5	6	D1
RS	7	8	D2
	9	10	D3
GND	11	12	D4
GND	13	14	D5
GND	15	16	D6
GND	17	18	D7
GND	19	20	
GND	21	22	
GND	23	24	
GND	25	26	

So what about the code?

There are several ways to talk to the parallel port

1. Direct port I/O – requires root authority
2. Write a kernel device driver – complex (although Mark Harris has done most of the leg work for us) and non standard
3. Use the existing parallel port device driver for device `/dev/parport0`
This can run in user state and it is commonly available. I used this option.

File device drivers support open, close, read, and write for normal operation. Unfortunately toggling control port bits is not “normal operation”. For hardware specific operations, the `ioctl` (i/o control) operation is provided. We will use this for both data and control writes. The constants are in `include/linux/ppdev.h`