# Linux LPLCD

***A Guide for Attaching LCD Panels to Computers With Linux***

*By Andrew J. Kroll <ag784@freenet.buffalo.edu>*
*13 October*, 1998 Revision 0.0.1

# LEGAL NOTICES



Special thanks to LinuxNET members for funding, particularly Rich Sahlender for initializing approval of fund allocation.

Thanks to *ALL* LinuxNET members who encouraged me to do this, especially Alan Cox, because it's all his fault for challenging me to do this in the first place. To bad I didn't make a monetary bet. :-)



Enough of all this legal crap, let's get started!

# Part 1: Panel Specification Guidelines

Character only (non graphical) based panels should have a Hitachi HD44780 compatible controller on board for compatibility with current drivers.  Although nearly all character based LCD panel units fall into this category, there are probably some that are not compatible.  From my personal experience, every panel that I have found is compatible, and has worked without modification to the panel or software.  This does not include any character panels that display greater than 80 characters total, even if the command set is compatible.  There is, however enough room left to add such a panel, or a dual character panel on the same port.  I may do so in the future, but as of this time, I personally do not have a need.  If you think it would be a good idea, send me an email, and I'll proceed to include it in the specifications and update the drivers.  Character based panels can be placed on any type of LP port due to the simplicity of operation.  If there are any other's please let me know and I'll attempt to support it.  Basically, it needs to have a simple 8-bit interface to work, and not need the main CPU to keep the display showing.  I do not want to see any panel that either connects to greater than one LP port, connects to a serial port, or requires the OS to draw the display. Those kinds of displays do not meet the ideas and goals of a self sustaining LCD unit on the LP port.  If you want to support those, I encourage you start your own project. Infact, I would even like to see it.  If your display is NOT LCD but is compatible with the current drivers, great!  Please remember that LCD is low power and better when you have a limited amount of battery hooked to your UPS, and doesn't generate heat. These are other reasons why LCD is the display of choice for this project, and why it's called Linux *LPLCD* in the first place.

Graphical panels should have a Toshiba T6963C compatible controller on board for compatibility with current drivers. I have chosen this chip for very specific reasons, but it does not mean that other panel CPU will not be supported. Firstly, it uses a simple interface that is compatible with bidirectional parallel ports. Second, it can have 32K to 64K of RAM, which eases use of the panel, and virtually eliminates storage overhead on the kernel driver side. Finally, this particular controller is also flexible enough to allow for relocation of the graphics, attribute and character pointers on it's on-board RAM, allowing multiple consoles and such. The choice was kind of obvious once I read the specification sheets about it from Toshiba and started testing. If there are any other types, please let me know and I'll attempt to support it. Basically, it needs to have a simple 8-bit interface to work, and not need the main CPU to keep the display showing. I do not want to see any panel that either connects to greater than one LP port, connects to a serial port, or requires the OS to draw the display. Those kinds of displays do not meet the ideas and goals of a self sustaining LCD unit on the LP port. If you want to support those, I encourage you start your own project. Infact, I would even like to see it. If your display is NOT LCD but is compatible with the current drivers, great! Please remember that LCD is low power and better when you have a limited amount of battery hooked to your UPS, and doesn't generate heat. These are other reasons why LCD is the display of choice for this project, and why it's called Linux *LPLCD* in the first place.

With all the descriptions of what the goals are and what will work, let's move on to the actual connection spec to the LP port.

# Part 2: Standard Connections to LP Port

2a: Character Based Modules:

Listed here are the minimum connection pins used on PC (AT/XT) parallel port used for character based LCD panels. Other platforms may vary.

| Pin | Symbol | Function |
|---|---|---|
| 1 | Strobe | connected to the /CE signal of the LCD |
| 2-9 | DB0-DB7 | connected to DB0-DB7 on LCD |
| 11 | Busy | connected to pin 9 (DB7) for ID and status |
| 14 | ASF | connected to the R/S input of the LCD |
| 16 | Init | connected to R/W input of the LCD |
| 25 | GND | connected to GND on the LCD |

2b: Graphical based Modules:

Listed here are the minimum connection pins used on PC (PS/2 bidirectional) parallel port used for graphical based LCD panels. Other platforms may vary.

| Pin | Symbol | Function |
|---|---|---|
| 1 | Strobe | connected to /GW (See Figure 1) |
| 2-9 | DB0-DB7 | connected to DB0-DB7 on LCD |
| 14 | ASF | connected to /GR (See Figure 1) |
| 15 | Error | Connected to pin 9 (DB7) for ID |
| 16 | Init | connected to /RESET input of the LCD |
| 17 | SEL | connected to C/D input of the LCD |
| 25 | GND | connected to GND on the LCD |

By using the 74LS08 AND gate as in Figure 1, /W and /R are gated to form the /CE signal. The setup time for the T6963C chip is automatically provided by the propagation delay of the 74LS08 package, which is needed as per the datasheet specifications provided by Toshiba. The LP port can run from 8 to 10 MHZ. Unfortunately the T6963C can handle 2 MHZ access maximum, it becomes unreliable at higher speeds. I designed this simple circuit to be combined with very small software delays to eliminate the race conditions and panel crashes. This circuit also eliminates the need for a separate control line that the PC's LP port does not have. It actually solves two problems at once with a discrete component that is readily available.

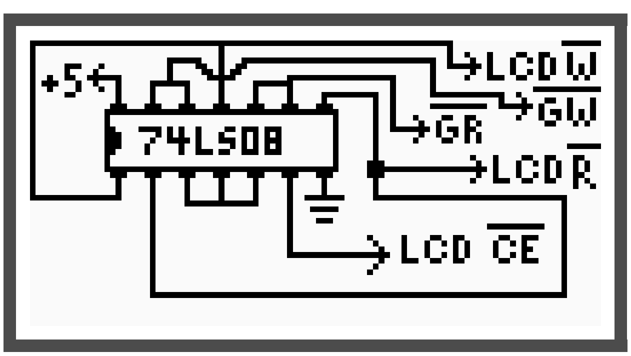


Figure 1

# Part 3: Software Interface Guidelines

3a: Character Based Modules:

At present there is no assigned device major for direct access, so for now, we use Major 23.
```
crw-r--r--   1 root     root      23,   0 Feb 19  1997 /dev/lcd0
```
It is initialized via a command string set to the panel.
ASCII is sent directly to the kernel module for displays.
Commands and data sent to the panel that contain 0x00 are communicated separately by sending a null string to the kernel module via /dev/lcd0. it’s a kludge, but it works, and 0x00 is not often sent to the module, so performance is not hindered.
Since character based LCD is only 80 bytes total, there is no plans on supporting fbcon services, and X is impossible because there are only 8 definable characters.

3b: Graphical based Modules:

At present there is no assigned device major for direct access, so for now, we use Major 23.

```
crw-r--r--   1 root     root      23,   0 Feb 19  1997 /dev/lcd0
```

Direct commands and data are passed as a data stream containing what we want to do in a structure as follows, each part consisting of one byte:

```
  BYTE               BYTE    BYTE    BYTE       BYTE
<bytes in command><command><data 1><data 2>...<data n>
```

Note that the order is the exact opposite order of what the controller expects. This is because the kernel module is smart enough to figure out how many bytes should follow a command. Some of this is done by adding new commands to the T6963C via software by using illegal/unused commands and dealing with them in special ways.

Special commands are as follows:

| To do... | Send | Returns on success |
|---|---|---|
| text fill | 0x00 fill | 0x00 |
| graphics fill | 0x01 fill | 0x00 |
| attribute fill | 0x02 fill | 0x00 |
| fill Start-Count | 0x03 Sl,Sh,Cl,Ch,fill | 0x00 |
| reset panel CPU | 0x04 | CPU status 1 BYTE |
| CPU status read | 0x05 | CPU status 1 BYTE |
| auto RAM write | 0x06 data | 0x00 |
| auto RAM read | 0x07 | LCD RAM data 1 BYTE |
| report RAM size | 0x08 | LCD RAM in K 1 BYTE |
| report panel attrib | 0x09 | panel attrib 4 BYTES |
| Report panel mode | 0x0a | panel mode 1 BYTE |

The following help debugging:
Other illegal commands are trapped and return EBADRQC.
Out of range operands returns EINVAL.
Not enough operands returns EPROTO.
To many operands returns E2BIG.

Should the panel CPU lockup at anytime during use, all commands will return EIO, except reset and CPU status read. Resetting of the panel's CPU should be handled by the user space program should this situation occur.

# Part 4: Software Module Identification

4a: Identification of display type:

Since we are dealing with at least two types of displays, I thought it would be good to tell what we are connected to in software. This is done in a simple manner, has room for literally hundreds of display types to be identified, quickly in software. Generic detection method works as follows:

Using the output databits, strobe each till we get an answer on one of the LP port inputs.
For detecting an unknown display size I have devised 3 generic identifications, which require display parameters to be hard coded into the user application.

First, no response on any of the input pins means that there is *possibly* an old style character module attached, and it's using the obsolete connection devised by Stephen S. Richardson. I do not recommend this attachment, but it is supported by the driver. Anyone using his connection is advised to add the three new wires or at least the generic wire, because it makes the display much more reliable.

If BUSY tracks DB7, we have a character based module.

If ERROR tracks DB7, we have a graphical based module.

4b: Character Based Module Identification:

| Display width size | Tracking bits |
|---|---|
| UNKNOWN | NONE |
| 8 | PE tracks DB0 |
| 12 | PE tracks DB1 |
| 16 | PE tracks DB2 |
| 20 | PE tracks DB3 |
| 32 | PE tracks DB4 |
| 24 | PE tracks DB5 |
| 40 | PE tracks DB6 |
| 80 | PE tracks DB7 |

| Display height size | Tracking bits |
|---|---|
| UNKNOWN | NONE |
| 1 | ACK tracks DB0 |
| 2 | ACK tracks DB1 |
| 3 | ACK tracks DB2 |
| 4 | ACK tracks DB3 |
| 5 | ACK tracks DB4 |
| 6 | ACK tracks DB5 |

4c: Graphical based Modules Identification:

| CPU type | Tracking bits |
|---|---|
| T6963C LCDI mode | BUSY tracks DB0 |
| T6963C LCDII mode | BUSY tracks DB1 |
| RESERVED | BUSY tracks DB2 |
| RESERVED | BUSY tracks DB3 |
| RESERVED | BUSY tracks DB4 |
| RESERVED | BUSY tracks DB5 |

| Display character width size | Tracking bits |
|---|---|
| UNKNOWN | NONE |
| 30 | PE tracks DB0 |
| 32 | PE tracks DB1 |
| 40 | PE tracks DB2 |
| 48 | PE tracks DB3 |
| 64 | PE tracks DB4 |
| 80 | PE tracks DB5 |
| 128 | PE tracks DB6 |
| RESERVED | PE tracks DB7 |
| RESERVED | ERROR tracks DB0 |
| RESERVED | ERROR tracks DB1 |
| RESERVED | ERROR tracks DB2 |

| Character width in pixels | Tracking bits |
|---|---|
| UNKNOWN | NONE |
| 5 | ACK tracks DB0 |
| 6 | ACK tracks DB1 |
| 7 | ACK tracks DB2 |
| 8 | ACK tracks DB3 |
| RESERVED | ACK tracks DB4 |
| RESERVED | ACK tracks DB5 |

| Display character height size | Tracking bits |
|---|---|
| UNKNOWN | NONE |
| 2 | SELIN tracks DB0 |
| 4 | SELIN tracks DB1 |
| 6 | SELIN tracks DB2 |
| 8 | SELIN tracks DB3 |
| 10 | SELIN tracks DB4 |
| 12 | SELIN tracks DB5 |
| 14 | SELIN tracks DB6 |
| 16 | SELIN tracks DB7 |
| 20 | ERROR tracks DB3 |
| 24 | ERROR tracks DB4 |
| 28 | ERROR tracks DB5 |
| 32 | ERROR tracks DB6 |
| RESERVED | BUSY tracks DB6 |
| RESERVED | ACK tracks DB6 |
| RESERVED | ACK tracks DB7 |

Display Resolution is calculated as follows:

X pixels =
<Display character width size> * <Character width in pixels>

Y pixels =
<Display character height size> * <CPU DEFAULT CHARACTER HEIGHT>

The T6963C always has 8 pixel character height. Other CPU's may not, that is why there are reserved entries.

Databit isolation is simply done with diodes, if more than one DB connects to the ID inputs. this minimizes cost of the entire project, yet leaves enough room for flexibility.