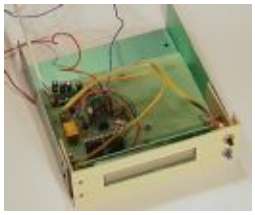
A LCD control panel for your Linux server



by Guido Socher (homepage)

About the author:

Guido loves Linux not only because it is fun to discover the great possibilities of this systems but also because of the people involved in its design.

Abstract:

In this article we will design a LCD control panel based on a Hitachi HD44780 LCD display and the AT90S4433 AVR 8−Bit RISC Microcontroller from Atmel. Both components very common and not expensive. The control panel includes a watchdog to supervise the computer and has two buttons to allow for a dialog with the user. You can set IP address, netmask, gateway address, shutdown the computer, readout statistics, basically anything you want because most of the logic is implemented with a perl script and it can be changed easily. The panel is connected to your computer via RS232 serial line.

For this article you need at least a partial installation of the Linux AVR development environment. How to set it up is described in this article: Programming the AVR Microcontroller with GCC.

Introduction

This device basically combines functionality of hardware already used in previous articles:

- Using Serial Line LCD displays under Linux
- A serial line computer shutdown button and LED

Our new design goes much further than that. It adds buttons for user interaction and includes a hardware watchdog to supervise the server. In addition to that the hardware provides for one analog input line. We don't use it here but you could connect a temperature sensor to it, for example.

To design this you will need some electronic hobby skills. The parts we use are inexpensive and cost all together less than 40 Euro.

The idea behind this panel is that it should allow you to control a server without monitor or keyboard. Linux is a very reliable server operating system and can easily be controlled remotely. However when you connect it to a network the first time you need to set the IP address, gateway and netmask. This panel allows you to set these addresses. It also gives you the ability to shutdown the server while you are still in the server room.

The design of this panel is very generic. All the "server specific" parts are implemented in a perl script. The entire hardware, the state of the buttons, text on the display, LEDs..., can be controlled via ASCII commands. You can therefore use the design to build an mp3 player or control your toaster, what ever you want.

What you need

To build this you need the following parts:

1 x Atmel At90S4433 Microcontroller
1 x 28pin 7.25 mm IC socket
1 x 16pin IC socket
1 x MAX232
1 x small 5V relay
1 x 4MHz crystal
2 x LEDs (green and red)
1 x BC547 NPN transistor
1 x BC557 PNP transistor
4 x 1uF capacitor (common or biased)
2 x 27pF ceramic capacitor
1 x 10nF capacitor
1 x 100nF capacitor
3 x resistor 4k7
2 x resistor 2k2
1 x resistor 10K
1 x resistor 3k3
2 x resistor 100 Ohm
3 x resistor 470 Ohm
3 x resistor 1k
1 x resistor 220 Ohm
1 x 4K7 potentiometer (as small as possible)
1 x Z–diode 4.3V
2 x small touch buttons
1 x small standard diode (e.g 1N4148, any cheap diode)
1 x 2 line 20 character LCD display with HD44780 compatible interface.

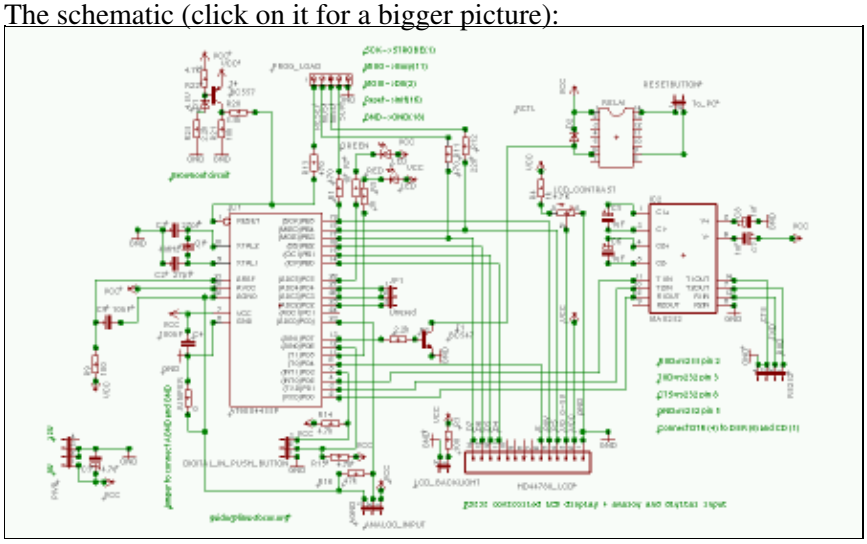
All LCD displays that I have ever seen with 14 or 16 pins on the connector were HD44780 compatible. You can also use a 3 or 4 line display but then you will need to modify the software a bit.

In addition to that you need some wires and connectors for power and RS232. If you have a 2 line display then you can mount it onto a thin aluminum metal sheet and fit it into a 5.25" bay on your server.

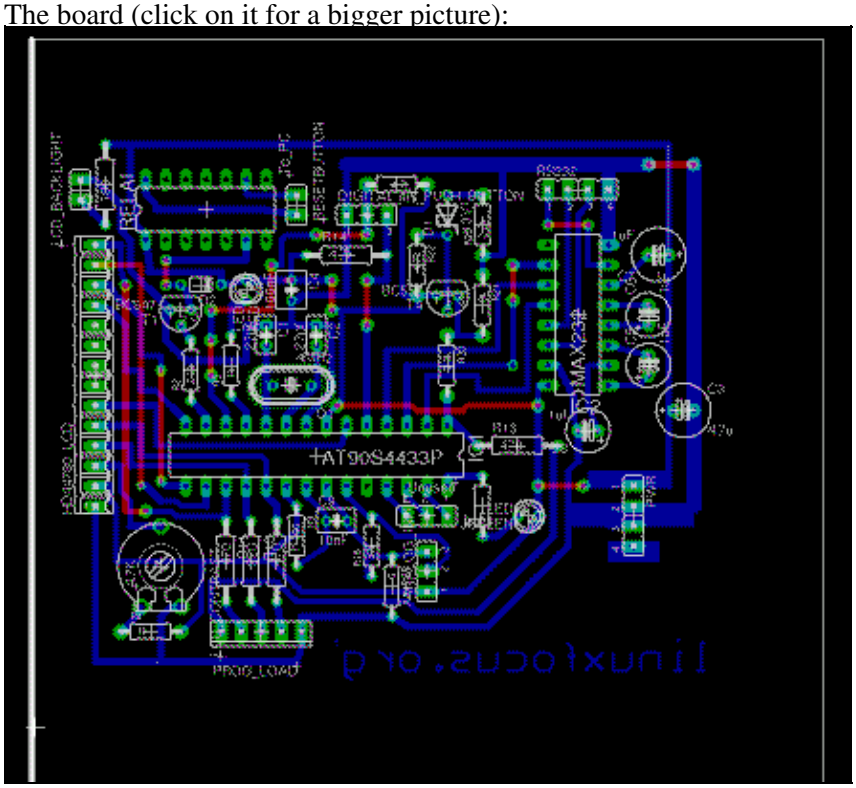
Schematic and board

I used eagle for Linux to design the schematic and board. It's an excellent software but you will need some time to learn how to use it. You can get a free version for private use at cadsoftusa.com.

The schematic (click on it for a bigger picture):



The board (click on it for a bigger picture):



The board layout with white background for better printing: board with white background (Note: This is not the file you need to make the printed circuit board.)

The eagle files (compressed with gzip, note some smart browsers uncompress already during download):

- linuxlcdpanel.brd.gz

- linuxlcdpanel.sch.gz

The circuit

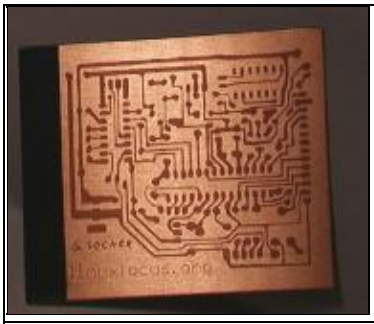
I will quickly explain the above circuit diagram. The AT90S4433 has 3 ports: PB, PC and PD. PC can be used as analog or digital input port. All ports can be used as digital input and output lines. This is controlled from the software via the DDR (Data direction register). We use all pins except 23 for digital lines (0 or 5 V). The Max232 is a voltage level converter. The RS232 interface uses +−10V and the Max232 converts this to 0−5V. On pin 1 (reset pin) of the AT90S4433 you find something called Brownout circuit. This circuit keeps the reset low (active) during times of insufficient power supply to prevent the CPU from executing wrong instructions or general malfunction. This can happen for a few milliseconds during power−up or power−down. It ensures basically that the program on the Microcontroller is started correctly.

Some of you might wonder why there is a diode in parallel to the coil of the relay and the polarity such that looks like it will never have anything to do. This diode is very important! When you switch of the relay then a very high voltage is generated by the coil. It can destroy the Microcontroller. This voltage has opposite polarity to the supply voltage on the coil. The diode can be any cheap small diode nothing special but it is important to have the diode.

The two push buttons need to be connected to the connector marked in the schematic as "DIGITAL IN PUSH BUTTON". They connect PD3 or PD6 to ground when pressed.

How to make the printed circuit board

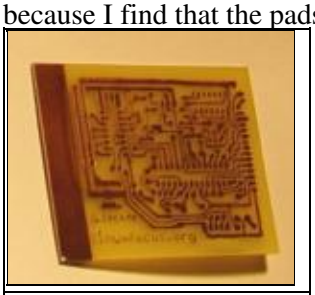
To etch the printed circuit board you first need to print this postscript file (linuxlcdpanel.ps.gz) onto transparent foil. In shops for architects you can get some semi−transparent plastic foil called Sinolit. Its produced by Regulus and is normally used for offset printing. Another good alternative is 60g paper + transparent spray (pausklar 21 from Kontakt Chemie). The advantage of paper and Sinolit is that the toner from Laser printers really sticks to the paper/foil and provides for good contrast.
I have converted the postscript file to PDF in case you don't have a postscript printing system. The quality is however rather poor.
The exposure time for copper clad photo boards depends on the light source. For a normal home solaria it is between 1−2 Minutes. You can also use daylight but try to avoid direct sun light (it's too strong). You should experiment a bit with small stripes of photo boards to get the best exposure time before you use the real board.



The exposed and developed board before etching

The exposed board needs then to be developed for several minutes in NaOH (Natriumhydroxid). After that

you should carefully check the result and make corrections with a Edding 780 black paint marker (not the permanent markers for overheads, it is a marker with real think paint). I usually make the pads a bit bigger because I find that the pads from Eagle are too small for hobby use.



The ready made board before drilling the holes

Note: Somehow it seems impossible for the manufactures to agree on a common pinout and naming conventions for relays. I used a small 5V relay manufactured by Matsushita. Your relay might have a different pinout therefore change the board as needed (with an etch resistant paint marker).

When you are satisfied you can etch the board in FeCl3 (Ferric Cloride). FeCl3 has good etching rates at room temperature. It it very easy to use and therefore good for use at home. You get the best results if the board is standing upright in a high container. Copper ions are heavier than iron ions and if you fill the FeCl3 into a small flat tub then the copper ions will accumulate at the bottom where the board is.

When the board is ready wipe off the Edding paint marker ink with turpentine. You can leave the photo resist "ink" on. It will evaporate when you solder on it and protects the copper.

The software for the Microcontroller

The software for the Microcontroller is organized in the following files:

- lcd.c, lcd.h, lcd_hw.h: This is a generic avr LCD library. It is based on the work of Peter Fleury (http://jump.to/fleury). This version is a bit modified and more flexible. It allows you to connect the LCD hardware to any of the pins on the Microcontroller. You just need to change the definition in the file lcd_hw.h.
- avr−util.c, avr−util.h: functions to get various delay times.
- uart.c, uart.h: This is a library for the RS232 interface. It uses hardware interrupts. Every time a character is received from the computer the function SIGNAL(SIG_UART_RECV) will be executed and the data is copied from the receive buffer to a string buffer. The command language for our LCD panel is designed such that each command ends with with newline character. When a new line is found then then a flag (uart_rx_linecomplete) is set and the data is made available. This also means that you must not send commands to the display as fast as possible but wait a bit (a millisecond) after each line. Each command will be acknowledged by a result, ok, or err (for error). The driving perl program can therefore use the result as a trigger for to send the next command.
- analog.c, analog.h: Code for the analog to digital converter. It's again interrupt driven. A single analog to digital conversion is started and then the program waits for the SIG_ADC interrupt to read out the result from the ADC register.
- hardwarewd.c, hardwarewd.h: This is the watchdog. We use the internal divider (divide by 1024) to pulse the timer. The timer is a 16 bit register when we get an overflow we count down a 8 bit variable. With a 4MHz crystal we will count down our variable approximately every 16 seconds. The perl program signals that the computer that it is alive by periodically setting the variable back to a high value. If it fails to do so (because the computer hangs) that then the variable will decrease in its value

all the time and when it reaches zero the relay will go click−clack and do a hardware reset of our server.
- linuxlcdpanel.c: This is the main program. It will continuously check for commands from the RS232 interface and button press events.

To understand the software in detail I recommend you to read the datasheet for the Microcontroller. It's available from the references section at the end of the article (or from www.atmel.com)
However to use this panel you don't need to understand the software you just need to unpack the source code archive (get the linuxlcdpanel−0.7.tar.gz from the download page) and type:

make
make load

or you can even use the pre−compiled software and load it with the command
make loadprebuild

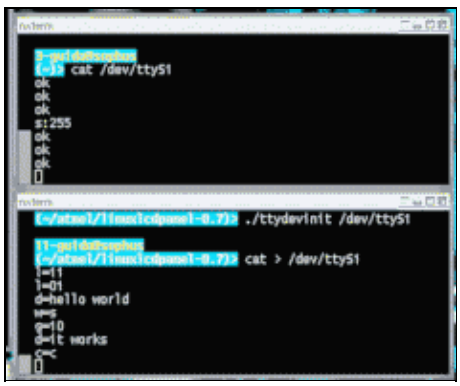
Very easy. You find a description on how to program the Microcontroller in the first article: Programming the AVR Microcontroller with GCC.

Testing the LCD panel

The LCD panel is designed to work from the 5V internal power supply of your server (red cable=5V, black cable=ground). However you should never connect it already the first time to the computer power supply. You might have made a small mistake during the soldering and assembly process. The computer power supply is very strong and both your computer and the board can go off in a cloud of smoke if you made a mistake. Test is first with an external electronically stabilized and current limited power supply! Now you download the software too the EEPROM as described above. After that you should see a scrolling "linuxfocus.org" banner in the LCD display. Now connect the RS232 interface:

MAX232 pin 14 to CTS (DB−9 pin 8)
MAX232 pin 7 to RXD (DB−9 pin 2)
MAX232 pin 13 to TXD (DB−9 pin 3)
You also need to connect DTR, DSR and CD which each other (DB−9 pin 4, 6 and 1)

This is documented in the circuit diagram above as well.



To use the serial line you need to initialize it. The linuxlcdpanel−0.7.tar.gz source code archive contains a program called ttydevinit which does exactly this initialization. Let's say you have connected the panel to COM2 (ttyS1) then you need to run once the command:

./ttydevinit /dev/ttyS1

Now the serial line driver is initialized to use 9600 Baud and you can start to "talk" to your LCD panel. Open 2 xterm windows. In one you type "cat /dev/ttyS1" and in the other you type "cat > /dev/ttyS1". Now you can e.g type the command l=11 (switch on LED 1) or l=10 (switch off LED 1). You can see in the first xterm that our LCD panel acknowledges your commands with "ok".

All available commands are explained in the README.commands file.

The source code archive contains a perl script called ttytest.pl that does nothing but switch on and off the red led in intervals. It is meant to be used as a simple example program that shows how to drive the LCD panel. You can use it as a base for your own program. Take a look at the source code. It requires some knowledge of perl but it is a rather short program.

Connecting the watchdog

The watch dog is off by default. You switch it on with the command w=1 and you set the timeout with s=x where x is a value between 1 and 255. E.g s=10 will let the watchdog timeout in 10*16sec=160sec. The driver program needs set the timeout periodically to avoid that the watchdog hits. If your server should ever lock up then the driver program will no longer set the timeout and the watchdog hits. I know that Linux servers almost never hang up. However if they hang up and get stuck then there is usually nobody at site to press the reset or nobody knows where the server is because there where no problem with it for the last 2 years.

The watchdog is designed such that it will hit only once. This is to avoid it hitting again during the file system check which will probably follow after the reset. When the server comes up again the driver program needs to re−enable it.

To physically connect the watchdog you need to find the two wires that go the reset button of your server. In parallel to this reset button you need to connect the relay from our watchdog.

How to use the watchdog

The watchdog guarantees that the system is always able to execute programs. It does not guarantee that a webserver or a database application is still running and responding. To check such things you should use a crontab entry or other programs. You can be confident that the crontab will be working because the watchdog ensures that software in general is still executing.

For example you can design a script that is triggered by a cronjob and downloads a webpage from your webserver every 15 minutes, but you have to be careful with that: A webserver can get heavily loaded by a lot of requests and then it is normal that it does not answer all of them. Therefore you should count how often the webserver did not answer. If it did not answer at all in the last 10 checks then you can restart the webserver or trigger a normal reboot (not a hard reset via watchdog).

Apart from the application you should also monitor the disk usage. The following shell command will return something if one of the disk partitions becomes more than 80% full:

```
df | egrep ' (8.%|9.%|100%) '
```

This can again be used in combination with a crontab entry to regularly check the disk usage.

The scripts on the server

Almost all the logic of our LCD panel is implemented in a perl script called llp.pl copy this file to /usr/sbin/. Next copy the program ttydevinit to /usr/bin and the file ifconfig_llp.txt (from the etc directory of the source code archive) to /etc. Now edit the ifconfig_llp.txt and changed the addresses as needed:

```
NETMASK=255.255.255.0
IPADDR=10.0.0.4
```

GATEWAY=10.0.0.2

Now make a backup of your original /etc/rc.d/init.d/network script and copy the etc/network script from source code archive to /etc/rc.d/init.d/network. This script and the directory names are only valid for Redhat and Mandrake. The script etc/network_all_distributions is more basic and will work with any Linux distribution but you need to figure out where your Linux distribution has its init−rc directories. It is slightly different from distribution to distribution.

Edit the file /etc/rc.d/init.d/network and change the line

/usr/sbin/llp.pl /dev/ttyS1&

if you do not use COM2.

Now you can run

/etc/rc.d/init.d/network start

and see your LCD panel in action. Note: It is save to play around and modify the IP address. The changes take only affect after the next reboot. Therefore test it and then change it back before you shutdown the server (you can also edit /etc/ifconfig_llp.txt to undo your changes).

Log files

The llp.pl script writes a logfile to /var/log/llp.log. This log will only grow very slowly. There should normally be no need to rotate it automatically. You can rotate it with a program like logrotate if you want. There is no post rotate action needed. A config file line for logrotate could look like this:

```
/var/log/llp.log {
nocompress
monthly
}
```

The log will contain entries when the system was manually shutdown, an IP address was changed (IP, GW, netmask) or when the hardware watchdog triggered a reset. Naturally you can not log the watchdog timeout when it happens (because the system hangs) but instead it will be logged at the next startup.

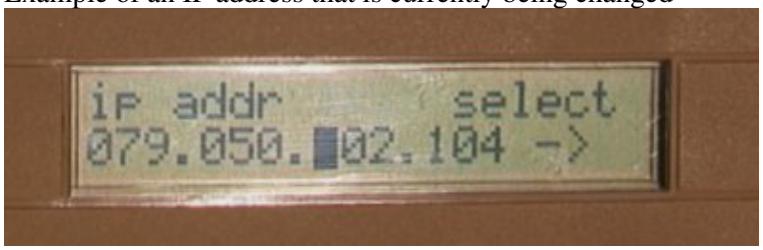
The panel in operation

Here are some "screen shoots" of the LCD panel in operation. This is not all of the functions offered by this panel. There are more and you can add your own.

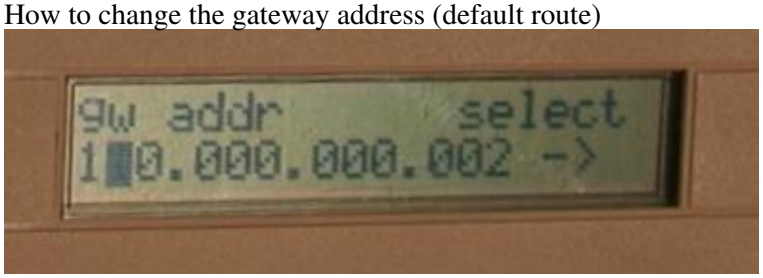
The main screen. Showing some name (linuxfocus in this case) and uptime and load. This updates periodically.

The IP configuration menu

Example of an IP address that is currently being changed



How to change the gateway address (default route)



Conclusion

To build this LCD panel requires some electronic hobby skills but it is not a very complex circuit. Our Linux LCD panel offers more functionally than any other control panel that I have ever seen and is very generic and cheap.

Happy soldering :−)

References

- The uisp AVR programmer software: www.amelek.gda.pl/avr/
- The source code for this article linuxlcdpanel−0.7.tar.gz . The circuit diagram, the Eagle files and screen shoots are as well included.

- All software and documents mentioned in this article (any updates of the linuxlcdpanel software will show up on this page)
- Datasheet for MAX232 MAX220−MAX249.pdf 448K
- Datasheet for ST232, a cheap variant, often sold instead of the real MAX232 st232.pdf 100K
- Datasheet for Atmel AT90S4433 avr4433.pdf 2356K
- The atmel website: www.atmel.com/
- Eagle for Linux cadsoftusa.com

2005−02−12, generated by lfparser_pdf version 2.51