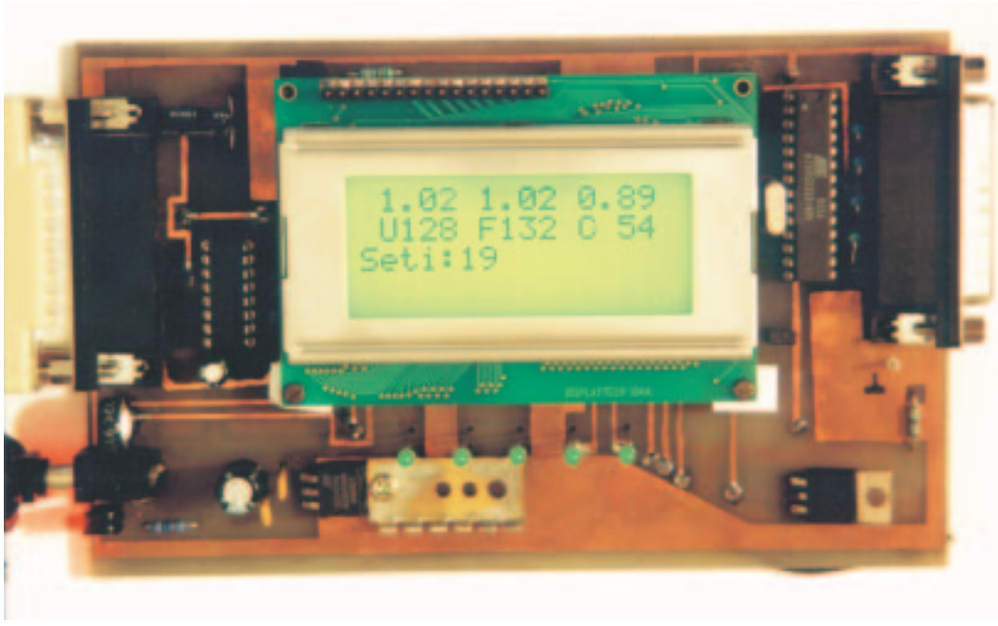
078serial.qxd 31.01.2001 16:09 Uhr Seite 78

Build your own LCD displays

FLACHMANN'S BEAT

MICHAEL MAJUNKE



When it comes to servers it is often not worthwhile connecting a monitor when the administration is done remotely, as there will only be status messages. Further to our serial key project, this time we are going to show you a serial display, with which you can keep an eye on your server.

A look at the circuit board with the components in place. The display was fastened to the circuit board with spacer pins for the casing to be fitted later

Anyone who runs a server or takes part in projects such as SETI@home, knows the problem: To see any information, however small, a monitor is needed. Usually it's just a couple of lines which are really of interest, and they could be quickly read off a display. Since smaller LCD displays are no longer that expensive and almost every computer is equipped with a serial port, such a display can be added relatively easily. The finished device can display 64 alphanumeric characters and has another five LED's for status messages. The cost come to about £20 to £30, depending on the casing and display type selected.

The circuit

The most important thing in the construction of the device is of course the display. Based on the display area needed we decided on a 16x4 display. Naturally it is also possible to select other formats. The built-in LCD Controller used in this project must be a Hitachi HD44780 or compatible.

Controller operation is implemented by means of parallel circuitry, so an interface is needed for connection to the serial port. This interface must also be capable of performing control tasks. These include output of data received and sent to the display, receive and process control codes, drive the LED's and more besides. For this, a small microprocessor, programmed as required, is ideal.

The main criteria for selection of the processor: Easy to program, sufficient number of inputs/outputs and it should not be too expensive. The choice came down to processors from the firm Atmel, which offer precisely what was required. The great advantage is that they are on-board programmable and that free programmer software is available. For the prototype, an AT90S2333 was used having 20 I/Os, 2K of program memory, SRAM, EEPROM, timer, UART and an A/D converter.

The Atmel-2333 is still around in large numbers but is an obsolete model and is no longer in production. Its big brother, AT90S4433, has even more memory and is therefore dearer, but fully hardware and software compatible with the 2333. The stock of CPUs should thus be assured for the next year, even if now and again one may have to put up with delivery bottlenecks.

078serial.qxd 31.01.2001 16:10 Uhr Seite 79

For operation we need a working voltage of 5 volts and a quartz crystal component, plus a few resistors which have to be soldered on so that the processor can be programmed in-circuit. You can find out how this self-build programmer functions in detail at *http://www.rowalt.de/mc/*.

For the data channel we will use the RxD and TxD line to the serial port which remained unused in the serial key project. Because of the difference voltage level, it is necessary to make an appropriate adjustment. At this point, we rely on the standard component MAX-232 from Maxim. This is relatively simple to integrate into the circuit and in addition it protects our sensitive interface ports from over-voltage.

Last of all, a power supply is needed for the circuit rated at a maximum of 300mA at 5 volts. For this we will use the popular voltage controller µA7805, which should really be fitted with a small heat sink. Anyone planning to fit the device in the PC case can also draw current direct from the PC power supply and so save a few components.

A completed layout in the Eurocard format can be found on the FTP site at *ftp://ftp.linux-magazin.de/pub/listings/magazin/2001/02/Serielles-display/*. To make the circuit board, we advise using the phototransfer process, and to do this, first print the layout with a high-resolution printer and next copy it twice onto acetate. The two acetate sheets are placed precisely one on top of the other thus producing adequate blackening for the exposure. Because of the relatively broad tracks, track undercutting ought to be easily avoided. Anyway, it won't do any harm to make a visual inspection of the completed circuit board after it's been treated with solderable lacquer.

Anyone who does not wish to perform the etching and drilling process themselves, can turn to the firm Kernel Concepts (*http://www.kernelconcepts.de*), where firstly the order requirements are recorded and over a certain number of pieces, complete circuit boards are manufactured. Whether Kernel Concepts will also be offering construction kits was not clear at the time of going to press.

After building the circuit, you should especially check the solder joints which are located close together for short circuits, but trimming is not necessary.

Programming

Once the circuit has been successfully completed, it is now time to program the circuit according for our purposes. There are a range of options open to us for this: for the purists, Assembly, for C-programmers, certain C compilers and for high-level language

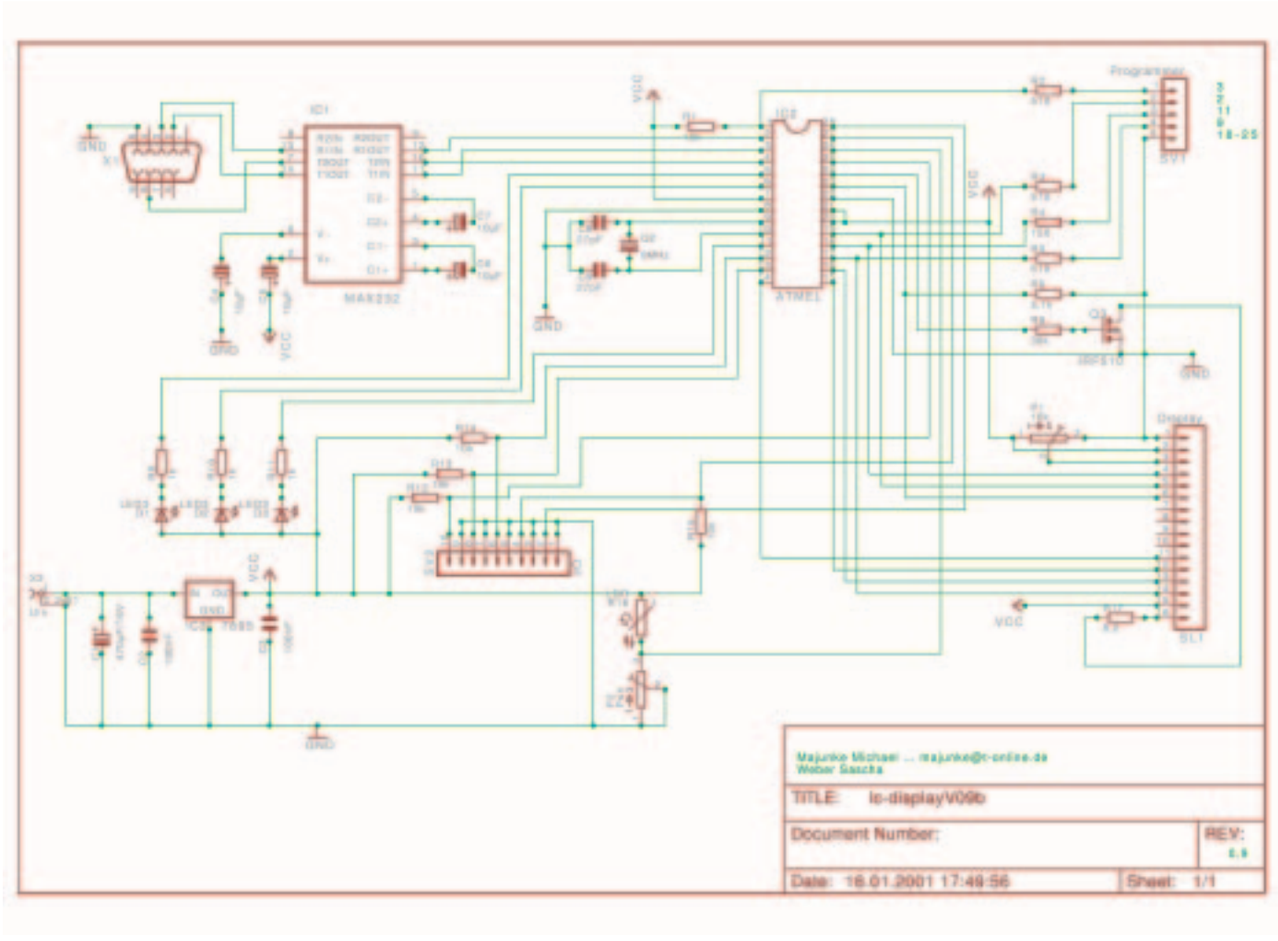


Fig. 1: Circuit diagram

078serial.qxd 31.01.2001 16:10 Uhr Seite 80

enthusiasts, Basic, Pascal, BASCOM and many more.

It is best to start with an assembler, which is the best way to learn the features of the processor. We used the Linux assembler AVRA, which after download is installed as follows:

```
tar xzvf avra-0.5.tar.gz
cd avra-0.5
cp Makefile.linux Makefile
make
su
make install
```

Since AVRA is compatible with the Atmel assembler, you can use its own instructions. You can find the complete assembler instructions for the processor at the home page of Atmel. Here is a simple example for LED control, which switches on the five LEDs of the display:

```
; small assembler example
; Control the LED
.DEVICE AT90S2333     ; Define processor
.EQU PORTD_DDR = 0x11 ; Name of port output d?
irection
.EQU PORTD_D   = 0x12 ; Name of port output d?
ata
.DEF WORK      = r16  ; Name of register R16
ldi WORK,0b11111100   ; LED-Ports to output
out PORTD_DDR,WORK    ; output DataDIR Port D
ldi WORK,0b00000000   ; LED on (neg. control)
out PORTD_D,WORK      ; output data-DataPortD
```

The command *ldi* loads a register with a constant, which is only possible with the registers R16 and above. With *out* a register can then be output in the I/O-address space. This is of course just a simple example.

The source code must now be translated for transfer to the processor. This is done using the command *avra name.asm*. This gives us a HEX file containing the program code for our processor. Anyone programming data for the EEPROM, in his program, will also have received another EEP file, which must be transferred later with the HEX file.

The actual task of transferring data to the processor is handled by the utility SP12, which serves as the basis for all write and read operations on the processor. SP12 is available as source code and binary file and contains a comprehensive set of instructions. In order to use SP12, after unpacking it must be initialised using *sp12 -i*. Then our HEX file can be transferred with the following command line:

```
sp12 -T1 -wpfc name.hex
```

Anyone having to transfer data for the EEPROM must extend the line by *-wefC name.eep*, which will make SP12 write this file into the EEPROM address space. By now we should have our first program in the processor and if everything has been done properly, 5 lit LED's should be visible.

Software

As the basis for your own projects, you will find a functioning sample program on our FTP server. It allows the display to be controlled via the serial interface, at which point control codes which appear in the text to be output are evaluated. Such a control code is introduced by the start symbol "~", followed by the command code with the

The finishing touch of an aluminium casing is not that easy to do, on the other hand it looks a lot better than the bare circuit board

078serial.qxd 31.01.2001 16:10 Uhr Seite 81

parameters. If pure text is sent to the display, it is simply output with the blank areas of the display ignored. Two examples show how a data output can occur under Linux:

```
echo ~B1~C >/dev/cua1
```

Here we shall first switch on the lights (~B1) and then the "display" is cancelled by (~C)

```
echo ~P206Hello~L01 >/dev/cua1
```

sets the cursor in line 3 and column 7 (~P206, as usual, begins the internal count at zero), writes starting at that position "Hello" and then switches the first LED on (~L01).

You will find an overview of the control codes in the annex to the program. The method of operation is as follows: When a symbol is sent to the processor via the serial interface it triggers an interrupt. This interrupt starts a routine which stores the symbol received in a buffer.

The main program now continually checks whether there is a symbol in the buffer. If so, the symbol is read from the buffer and evaluated, regardless of whether it is a code symbol or simple text. Depending on the evaluation, either the text is output or the command is executed. Since it's possible for the display buffer to become full, an XON/XOFF flow control (alternatively CTS) is programmed in. This stops the data stream from the computer and only continues when sufficient space has been freed in the buffer.

For program analysis the source text is exhaustively commented so you will certainly be able to introduce your own expansions quickly.

Use as status display

One area of application for the self-built display is status messages of all kinds from average loading via free memory or hard disk space to call number display from the ISDN log. You might also like to know whether a daemon has been started, how many users are currently logged on or how far a program has got in a calculation. All this can be output via the display automatically with *cron* and a few basic Linux commands such as *cut*, *grep* or *echo*. The following example from the */etc/crontab* displays the loading of the computer:

```
* * * * * root echo -n ~P001`cut -d" " -f1-3 /
proc/loadavg` >/dev/cua1
```

Large reserves

The circuit has deliberately been equipped with large reserves, so as to be able to take care of future and more complex tasks. Anyone using the more expensive Atmel-4433 will even have at his disposal twice the performance reserves. So it would for example be possible to produce, via the still free inputs and outputs, a data transfer from the display to the computer. Anyone who would like to build in an automatic brightness control can easily do so by

Component list

No.	*Type*	*Value*
1	*electrolytic capacitor*	*470µF, 16V*
4	*electrolytic capacitor*	*10µF, 16V*
2	*tantalum*	*0.1µF, 16V*
2	*capacitor*	*27pF, ceramic*
1	*quartz crystal*	*8MHz*
1	*FET P2N40 or similar*	*min. 300mA*
5	*LED, green*	*max. 20mA*
1		*LED, red*
1	*resistor*	*8.2 Ohm*
3	*resistor*	*510 Ohm*
1	*resistor*	*150 Ohm*
1	*resistor*	*5.1 kOhm*
1	*resistor*	*39 kOhm*
1	*resistor*	*10 kOhm*
6	*resistor*	*1 kOhm*
1	*potentiometer*	*10 kOhm, pre-set*
1	*voltage regulator*	*µA7805, 1A*
1		*ATMEL 2333 or 4433*
1		*MAXIM 232*
1		*display 16x4*
1		*plug connector 16 pin*
2		*socket 25-pin D-Type*
1		*connector power supply*
1		*IC holder, DIL 16 pin*
1		*IC holder, DIL 28 pin*

LCDs via the parallel port

Nils Färber reports on the use of a parallel port display: For anyone who finds controlling an LCD module with the aid of an additional microcontroller circuit board too fiddly or time-consuming, it is possible to achieve the same result with a simple cable solution at the parallel port.

1 IC holder, DIL 28 pin

078serial.qxd 31.01.2001 16:10 Uhr Seite 82

connecting a light dependent resistor (LDR) to one of the A/D converters. But with any expansion, the programming time also increases - so integrating the AVR C-compiler is definitely worthwhile for these purposes.

Principle

Most commercial alphanumeric LCD modules are based on the LCD controller HD44780 from Hitachi or compatibles. This can be controlled very easily via its parallel input. To do this, four or eight data lines for the four or eight-bit data mode and two control lines are necessary. Parallel ports on PCs have eight data lines and eight additional control lines, more than enough to control the LCD modules. The supply of power to the modules requires a bit of ingenuity. Since the parallel port works with 5 volt TTL levels and the display needs exactly 5 volts, the operating voltage could be obtained here. But to protect the output drivers of the port, it would be better not to do so. In the case of an internal installation, the PC power supply can be used, and if connected externally the display can be powered via the joystick port or an external power supply.

Hardware

As a rule, LCD modules have a 16 pin connector, which almost always has the same configuration. For connection to the parallel port, a 25-pin D-type plug and a length of 14 core cable is needed. If the joystick port is to be used for power supply purposes, another 15 pin D-type plug will be needed. The connection is described in Table 1.This configuration was selected so a ribbon cable could be connected almost 1:1. Those using the joystick port to supply power will obtain 5 volts from pins 1, 8, 9 or 15, and ground from pins 4, 5 or 12. Now all that's missing is the contrast regulator at pin 3 of the LCD modules.

Depending on the type of the modules, it can now become a bit more difficult. With some modules this pin can be bridged directly to earth, which gives maximum contrast and in many displays delivers a very good result. If the maximum contrast is too dark, a voltage divider must be connected here, i.e. a potentiometer with a range 100 to 500 Ohm, with one side at +5V, the other to earth and the middle tap to pin 3 of the LCD. Then the contrast can be adjusted by this means.

Table 1: Wiring of parallel port/LCD display

PC	LCD Function	LCD Pin
18-25	GND	1
	+5V	2
	DRV	3
18	R/W	5
1	EN	6
2	DB1	7
3	DB2	8
4	DB3	9
5	DB4	10
6	DB5	11
7	DB6	12
8	DB7	13
9	DB8	14
14	RS	4

Driver

The current version of the driver for Linux kernels from version 2.0 to version 2.4 can be found at *http://sourceforge.net/projects/lcd/* and can be downloaded there direct from the FTP or CVS domain. In the *driver/* directory of the driver, before compiling, take a look into the *Makefile*, because this is where it is possible to make a few adjustments to your own system. A subsequent *make* creates the driver module *lcd.o*, which can then be loaded with *insmod ./lcd.o*. In the *driver/* subdirectory there is also the small script *mkdevice*, which creates the appropriate device under */dev/*. The driver can also be attributed additional parameters, described in Table 2, when loading.

The two waiting times *t_short* and *t_long* are the intervals for which the driver waits for the LCD display. If the output appears on the display garbled, these times should be increased.

Application

If the cable connection is correct and the driver module has been loaded, then the display has been initialised and the cursor will be visible in the top left corner. A simple *echo "Hello" > /dev/lcd* will make the text "Hello" appear on the display.

The driver also supports, apart from normal text output, a few escape sequences, which are comprehensively described in the *README* file of the driver. These include: delete display, move cursor up and left, cursor on/off and self-definable graphic symbols. The latter are especially interesting for displaying bar graphics. ■

Table 2: Parameters of the LCD-modules

Parameter	Meaning	Default
io	I/O Address of the parallel port	0x378
cols	Number of display columns	20
lines	Number of display lines	4
t_short	short waiting time	40
t_long	long waiting time	100

The author:

Michael Majunke works as a communications electronics engineer and mainly spends his spare time tinkering on the computer and programming. To balance this he likes listening to e-music and still takes photographs the old-fashioned way with a film camera.

Info

HITACHI display controller
http://semiconductor.hitachi.com/
Maplin Electronics
http://www.maplin.co.uk
Homepage of the firm ATMEL
http://www.atmel.com
In-circuit programming of ATMEL processors
http://www.rowalt.de/mc/
Homepage of the firm Maxim
http://www.maxim-ic.com
AVR C-compiler overview
http://www.omegav.ntnu.no/~karlto/avr/ccomp.html
AVRA-Assembler
http://tihlde.org/~jonah/el/avra.html
Programming SP12-ATMEL processors under Linux
http://www.xs4all.nl/~sbolt/e-spider_prog.html
LCD parallel port drivers:
http://sourceforge.net/projects/lcd/

■