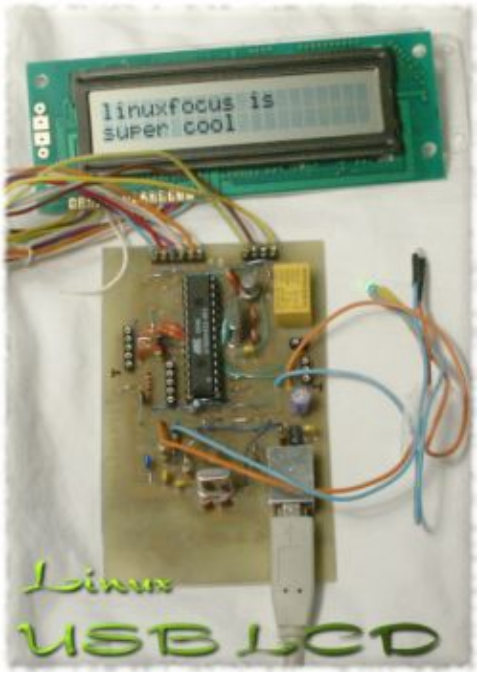
# Linux USB LCD Display mit Watchdog und Tasten




von Guido Socher (homepage)

*Über den Autor:*

Guido mag Linux, weil es ein Paradies für Leute ist, die ihre eigene Hardware und Software entwickeln wollen.

*Übersetzt ins Deutsche von:*
Guido Socher (homepage)

*Zusammenfassung*:

Dieser Artikel ist das Ergebnis der vielen positiven Rückmeldungen zu meinen bisherigen Hardware−Artikeln. Ihr LinuxFocus−Leser seid wirklich großartig! Einige von euch wollten wissen, wie man den USB Bus benutzen kann. Dieser Artikel präsentiert eine einfache und schöne Lösung. Wir benutzen das LCD Display aus dem Mai 2002 Artikel und bringen es mit USB zum Laufen. Das ganze wird "bus powered" sein. Man braucht also keine zusätzliche Stromversorgung.

Für diesen Artikel braucht man wenigstens eine in Teilen installierte Linux AVR Entwicklungsumgebung. Wie man sie aufsetzt, ist in dem Artikel Programmierung des AVR Microcontroller mit GCC, März 2002 beschrieben.

## Einführung

USB ist cool, weil es eine moderne Schnittstelle ist und die Möglichkeit bietet, die Geräte direkt mit Strom über den USB Bus zu versorgen. Die Stecker sind klein und große Datenmengen können über ein dünnes Kable transportiert werden. Das sind die positiven Seiten von USB. Die Nachteile sind eine komplizierte Hardwareentwicklung wegen den hohen Frequenzen und das Protokoll ist sehr kompliziert. Man braucht nur einen Blick auf die Protokollspezifikation (http://www.usb.org/developers/, du brauchst die 1.1 Spezifikation) zu werfen und man ist geschockt. Es sind 327 Seiten Papier und das Dokument ist nicht einfach zu verstehen. Kein Wunder, dass es sooo viele fehlerhafte Implementationen von USB Devices gibt. Eine mehr benutzerfreundliche Einführung kann man unter http://www.beyondlogic.org/ finden, aber die Spezifikation bleibt komplex.

Was soll man tun? Wie können wir unseren Microcontroller an den USB Bus anschließen? FTDI, eine schottische Firma, hat die Lösung (http://www.ftdichip.com). Sie bieten einen Chip, der eine USB zu RS232 Seriell Umsetzung macht. Eine Seite des FT232BM Chips ist rs232 und die andere USB. Mit anderen Worten, man ersetzt einfach den MAX232, den man bisher bei der rs232 Schnittstelle zur Spannungsumsetzung brauchte, mit dem FT232BM Chip und man ist fertig.

# Der Treiber

Der FT232BM ist eine Plattformübergreifende Lösung. Treiber gibt es für viele Betriebsysteme. Das Linux Kernelmodul nennt sich ftdi_sio und ist Open Source. Es ist Teil des Standard Linux Kernel. Der FT232BM bietet mehr als nur eine USB zu rs232 Umsetzung und das Linux Kernelmodul ist noch in der Entwicklung, um auch all die anderen Funktionen zu implementieren. Die reine USB nach rs232 Konvertierung ist aber fertig und ich konnte z.B. einfach einen Standard Redhat 7.3 Kernel (2.4.18) ohne neu kompilieren oder Änderungen benutzen. USB Stecker einfach reinstecken, fertig.

ftdi_sio wird unter folgender Adresse entwickelt: http://ftdi−usb−sio.sourceforge.net/.

Bei meinem Redhat 7.3 wurden alle Module automatisch geladen, wenn ich den USB Stecker einsteckte. Falls das bei dir nicht der Fall ist, dann prüfe, dass du folgende Module hast (für USB−UHCI):

/sbin/lsmod usb−uhci
/sbin/lsmod usbcore
/sbin/lsmod usbserial
/sbin/lsmod ftdi_sio

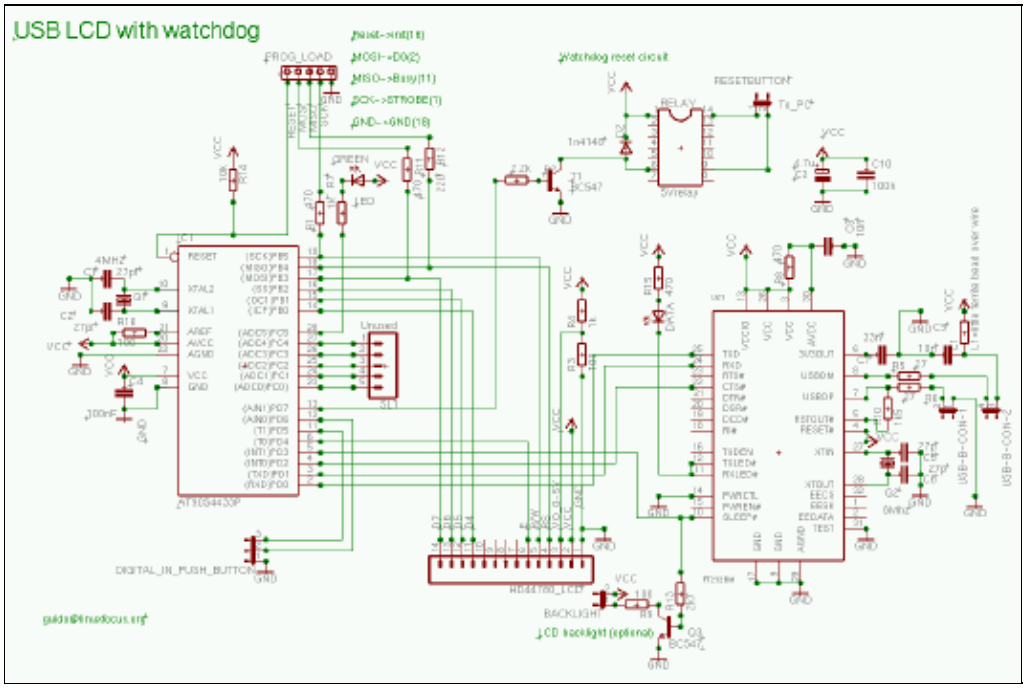
Die Devicedatei ist /dev/ttyUSB0

Die Entwickler von ftdi_sio empfehlen, den letzten Kernel (2.4.20), aber wie schon gesagt, hat 2.4.18 auch funktioniert (jedenfalls für die Dinge, die wir hier brauchen).

# Die Schaltung

Die Schaltung ist ganz einfach aufgebaut. Der FT232BM wird einfach zwischen den Rx/Tx Anschlüssen des Microcontrollers und dem USB Stecker eingebaut. Dazu braucht man ein 6 Mhz Quarz und einige andere

Teile, die in der Design Spezifikation von FTDI beschrieben sind. Das "ferrite bead" (Ferritkern, im Schaltbild rechts) ist eine kleine Spule, die die hohen Frequenzen (der USB Bus arbeitet mit 48Mhz) filtert. Man kann auch einfach 10 Windungen eines dünnen Drahtes über einen 1K Widerstand wickeln und das als Spule nehmen.

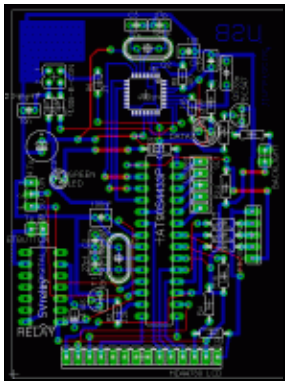



Eine Sache, die man beachten muss, ist die Stromaufnahme. Die Schaltung muss weniger als 100mA verbrauchen, wenn man sie über den USB Bus betreiben will. Zusätzlich muss dein Device den USB Suspend Mode unterstützen. Wenn der Pin, der mit "sleep" man FT232BM bezeichnet ist, auf Null geht, dann muss die ganze Schaltung weniger als 0.5mA verbrauchen. Das ist eine sehr harte Anforderung. Der AVR unterstützt einen "idle mode", in dem er weniger als 2mA braucht und einen "power down" mode, in dem er nur 20uA braucht. Es ist aber viel einfacher, den Microcontroller wieder aus dem "idle mode" aufzuwecken. Ich habe deshalb beschlossen, den "idle mode" zu nehmen, selbst wenn das die USB Spezifikation etwas verletzt. Die optionale Hintergrundbeleuchtung des LCD Display wird im Suspend Mode über den Transistor ausgeschaltet. Die gesamte Schaltung braucht dann etwa 3mA. 3mA ist mehr als 0.5mA, aber USB Host Controller Chips können den Strom nicht so genau messen, um diese Verletzung der Spezifikation zu entdecken. Es sollte funktionieren.

Nachdem ich das gesagt habe, muss ich leider zugeben, dass ich selbst keinen Computer habe, der Suspend unterstützt. Deshalb konnte ich diesen Teil nicht testen. Falls du einen Computer hast, der das kann, vermutlich ein moderner Laptop, dann teste bitte diesen USB Suspend Zustand und berichte mir.

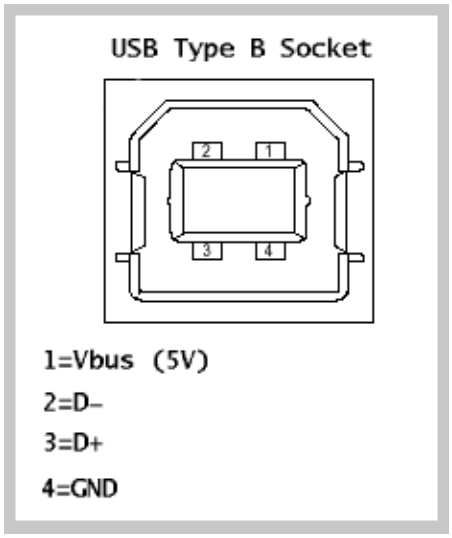
Der Rest der Schaltung ist fast identisch zu dem, was im Mai 2002 Artikel präsentiert wurde. Ich werde deshalb nicht weiter ins Detail gehen.

Du kannst auf das Schaltbild klicken, um ein größeres Bild zu erhalten. Die Eagel Dateien sind zusammen mit der Software am Ende des Artikels downloadbar.

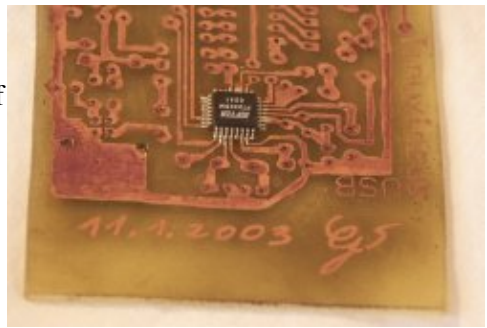


Die Platine ist eine einseitige Platine und nur der blaue Layer sollte geätzt werden. Die roten Linien sind Drähte.

Die USB Type−B Buchse, die man für diese Schaltung braucht, hat folgende Anschlussbelegung:




## Arbeiten mit SMD Chips




SMD Chips haben gute mechanische und elektrische Eigenschaften, aber sie sind ein Albtraum für Hobbyelektronikfans. Man braucht wirklich einiges an Geschick beim Löten und der Teil der Platine, auf der der SMD Chip aufgelötet wird, muss sehr sauber gearbeitet sein. Mit anderen Worten, das ist nichts für Anfänger. Schau dir die Liste mit Alternativen an, wenn du nicht sicher bist, dass du den SMD Chip sauber löten kannst.

Der SMD Chip muss aufgelötet werden, bevor irgendwelche anderen Teile auf die Platine kommen.
Um den Chip aufzulöten, gibt man zunächst etwas Lötzinn auf die Pads auf der Platine. Danach kommt ein dünner Film SMD Lötpaste (auch Löthonig genannt) auf die verzinnten Pads. Man kann auch einfach Lötlack von "Kontakt Chemie" aufsprühen, dann braucht man keinen Löthonig.

Jetzt reinigt man den Lötkolben und entfernt alles Zinn. Der FT232BM wird genau positioniert und dann

drückt man mit dem Lötkolben jeden Pin einzeln leicht an. Nach dem ersten Pin kann man noch mal checken, dass der Chip auch richtig sitzt. Beim Auflöten gibt man kein Lötzinn hinzu.

Dieses Verfahren funktioniert sehr gut. Es ist nicht so wichtig, dass man einen kleinen Lötkolben hat. Ein normaler Lötkolben tut es. Es ist viel wichtiger, dass der Lötkolben sauber ist. Ich kann nur abraten von irgendwelchen anderen Verfahren. Küchentoaster werden z.B. oft empfohlen. Damit macht man die SMD Chips nur kaputt.

## Der Test

Ich schlage vor, die Schaltung in zwei Schritten zu testen. Zuerst schließt man die Schalung an den Rechner (USB) an, ohne dass der AVR Microcontroller im Sockel steckt. Linux sollte den FTDI Chip erkennen und

man sollte folgenden Eintrag in /proc/bus/usb/devices finden:

T: Bus=02 Lev=01 Prnt=01 Port=00 Cnt=01 Dev#= 2 Spd=12 MxCh= 0
D: Ver= 1.10 Cls=00(>ifc ) Sub=00 Prot=00 MxPS= 8 #Cfgs= 1
P: Vendor=0403 ProdID=6001 Rev= 2.00
S: Manufacturer=FTDI
S: Product=USB <−> Serial
C:* #Ifs= 1 Cfg#= 1 Atr=80 MxPwr= 90mA
I: If#= 0 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=ff Prot=ff Driver=serial
E: Ad=81(I) Atr=02(Bulk) MxPS= 64 Ivl= 0ms
E: Ad=02(O) Atr=02(Bulk) MxPS= 64 Ivl= 0ms

Danach steckt man den AVR Microcontroller in die Fassung und lädt das Testprogramm. Die LED sollte dann blinken. Entpacke das linuxusblcd Tar−File (siehe Download am Ende des Artikels) und tippe:

make testload0

Dabei müssen sowohl der USB Stecker als auch das Programmierkabel eingesteckt sein.
Wenn das Testprogramm funktioniert, dann kann man sicher sein, dass der Microcontroller funktioniert.

Nun kann man die ganze Software laden:

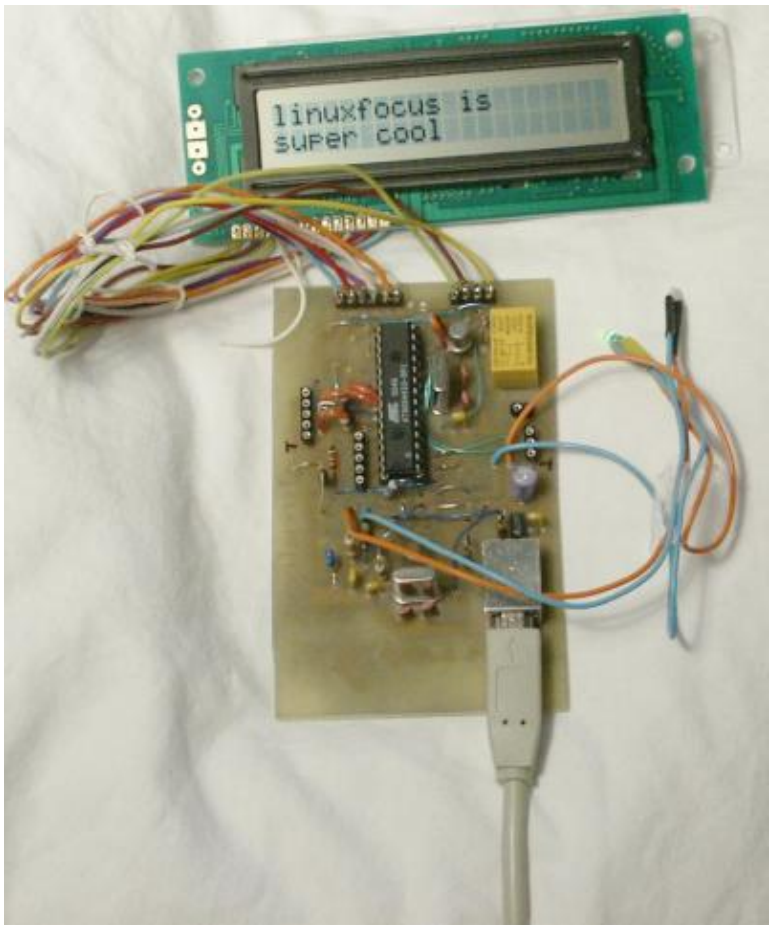
make load

Mit "ttydevinit /dev/ttyUSB0" initialisiert man unter Linux die USB Serial Connection und mit "cat > /dev/ttyUSB0" kann man mit der Schaltung reden.

ttydevinit /dev/ttyUSB0
cat > /dev/ttyUSB0
D=hello world

Das wird "hello world" auf das Display schreiben. Mehr Details dazu finden sich im Mai 2002 Artikel. Der Code für den Mai 2002 Artikel enthält auch ein Programm namens llp.pl, mit dem ein interaktiver Dialog über das Display und die Tasten realisiert wird. Dieses Programm kann man hier wieder verwenden.

... und hier ist das funktionstüchtige Display (die Taster waren nicht angeschlossen als das Foto gemacht wurde. Der FT232BM ist unter der Platine):




## Alternativen

Obwohl die hier präsentierte Schaltung sehr einfach ist, ist sie nichts für Anfänger, da man viel Geschick für den SMD Chip braucht. Man sollte sich deshalb überlegen, ob man nicht ein fertiges Produkt kauft. Der Nachteil ist, dass man normalerweise nicht diesen Funktionsumfang mit Tasten und Watchdog bekommt. Normalerweise erhält man nur das LCD Display. Die Preise für fertige Lösungen sind in Ordnung. Alleine die Bauteile für diese Schaltung kosten etwa 30 EUR und fertige Displays liegen in derselben Größenordnung.

Leider benutzen die meisten kommerziellen Produkte eigene Vendor IDs, selbst wenn der gleiche FTDI Chip verwendet wird. Das bedeutet, dass das Kernel Modul die Schaltung nicht erkennen wird, da USB Treiber von diesen IDs abhängig sind. Man muss den Kernel−code editieren und neu kompilieren. Zukünftige Kernel funktionieren vielleicht schon, wenn jemand anderes die IDs im Code eingetragen hat.

- http://www.matrixorbital.com/ Sie benutzen auch den FTDI 232bm, aber mit eigenen Vendor IDs. Das Display nennt sich LK202−24−USB.

- http://www.usblcd.de/ Diese Lösung hat ihr eigenes Kernelmodul. Es ist jedoch Teil des Standard Linux Kernel. Es wird ohne große Änderung mit jedem 2.4.x Kernel funktionieren. Vermutlich eine sehr gute Lösung.

- http://crystalfontz.com/ Ihre USB Displays (632 und 634) benutzen den FT232AM mit eigenen Product IDs.

- http://www.cwlinux.com/eng/products/products_lcd.php Ich habe diese Seite erst kürzlich entdeckt. Sie scheinen LCD Displays mit kleiner Tastatur zu haben. Die Displays sind jedoch doppelt so teuer wie die Lösung aus diesem Artikel.

## Referenzen


- Die Software und Dokumentation, die hier erwähnt wurde. (Updates für linuxusblcd werden auf dieser Seite zu finden sein.)
- Wie man den AVR Microcontroller programmiert: Den AVR Microcontroller mit GCC programmieren, März 2002
- Der Mai 2002 Artikel mit dem linuxlcdpanel. Das Perl Programm namens llp.pl aus diesem Artikel kann man wieder verwerten: Mai 2002 Artikel
- Die FTDI Webseite: www.ftdichip.com
- Das Datenblatt für den FT232BM (von http://www.ftdichip.com): ftdichip_ds232b11.pdf, 820Kb
- Eagle für Linux cadsoftusa.com
- Eagle Library für die FTDI Chips (von http://www.elektronik−projekt.de) ftdi.lbr.gz


---


| | |
|---|---|
| Der LinuxFocus Redaktion schreiben<br>© Guido Socher<br>"some rights reserved" see linuxfocus.org/license/<br>http://www.LinuxFocus.org | Autoren und Übersetzer:<br>en −−> −− : Guido Socher (homepage)<br>en −−> de: Guido Socher (homepage) |


2005−02−12, generated by lfparser_pdf version 2.51