Frequenzzähler 1Hz–100Mhz mit LCD Display und RS232 Interface

von Guido Socher (homepage)

Über den Autor:
Guido mag Linux nicht nur, weil es Spaß macht, die großartigen Möglichkeiten, die dieses System bietet zu entdecken, sondern auch wegen der Leute, die an seiner Entwicklung beteiligt sind.

Übersetzt ins Deutsche von:
Guido Socher (homepage)

Zusammenfassung:

Dieser Artikel setzt unsere AT90S4433 Microcontroller Serie fort. Ich schlage vor, die vorangegangenen Artikel über Atmel Microcontroller im Hinblick auf folgende Punkte zu lesen, um diesen Artikel besser verstehen zu können:

1. Wie man die Linux AVR Entwicklungsumgebung installiert und benutzt und wie man einen Programmierer baut:
   März 2002, Den AVR Microcontroller mit GCC programmieren
2. Wie man eine gedruckte Schaltung selbst herstellt:
   Mai 2002, Eine LCD Anzeige und Steuertasten für den Linux Server

Dieses Mal entwickeln wir einen Frequenzzähler, der Frequenzen im Bereich von 1Hz bis 100MHz messen kann. Alternativ kann man mit dem Zähler auch einfach Ereignisse zählen, wie z.B wieviele Leute die Straße überquert haben (oder was immer auch in der Form eines digitalen Pulses verfügbar ist). Der Zähler hat eine LCD Anzeige sowie einen RS232 Anschluß, um die Zählerwerte mit Linux auszulesen.

Einführung

Die Frequenz eines Sinussignals oder Rechtecksignals wird ausgedrückt durch die Anzahl der Schwingungen pro Sekunde. Um die Frequenz zu ermitteln, muß man also nur diese Schwingungen zählen. Auf diese Art ermittelt man die Frequenz der sogenannten ersten Harmonischen Schwingung eines kontinuierlichen Signals. Um die Frequenzen, aus denen ein nicht kontinuierliches "Geräusch" besteht zu ermitteln, braucht man einen Spectrumanalyser. Das ist jedoch ein ganz anderes Stück Hardware. Was wir hier bauen, ist ein Frequenzzähler für kontinuierlich oszillierende Signale. Wir nehmen an, daß sich das Signal während eines Meßintervals nicht ändert.

Der Frequenzzähler arbeitet in zwei Schritten:

1. Entfernen eines möglichen Gleichspannungsanteils aus dem Signal und dann das Signal mit einem Komparator in ein Rechtecksignal verwandeln.
2. Zählen der Pulse pro Zeiteinheit und dann den Wert durch die Zeit teilen.

Was man braucht

Um den Frequenzzähler zu bauen, braucht man folgende Teile:

1 x Atmel At90S4433 Microcontroller
1 x 28pin 7,25 mm IC Sockel
2 x 16pin IC Sockel
1 x 1pin IC Sockel
1 x 14pin IC Sockel
Keine IC Sockel für MAX903 und 74F74. Diese müssen direkt auf die Platine gelötet werden!
1 x MAX232
1 x 4,194304MHz Quarz
1 x LEDs (grün)
1 x BC557 PNP Transistor
4 x 1uF Kondensator (Elko)
2 x 27pF Keramikkondensator
4 x 10nF Mini– Keramikkondensator
3 x 100nF Mini– Keramikkondensator
1 x 200nF Mini– Keramikkondensator
1 x 0,47uF Mini– Keramikkondensator
2 x Widerstand 470 Ohm
1 x Widerstand 470K
2 x Widerstand 100 Ohm
3 x Widerstand 1k
5 x Widerstand 10k
3 x Widerstand 47K
1 x Widerstand 220 Ohm
3 x Widerstand 4K7

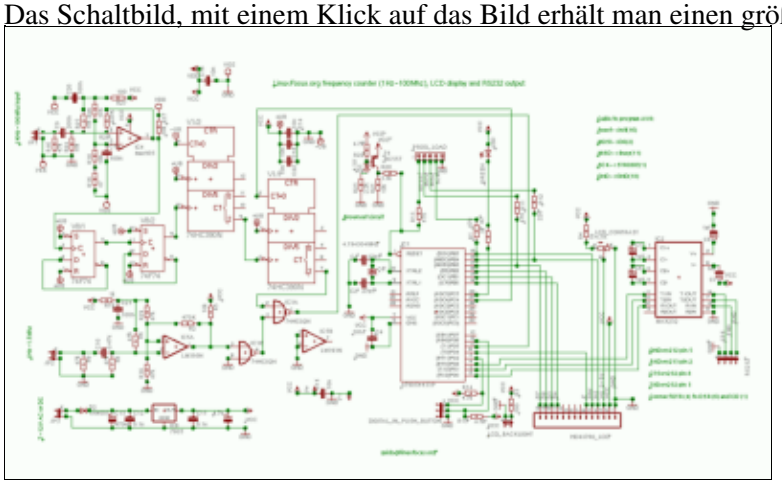
1 x Widerstand 3k3
1 x Widerstand 2k2
1 x Widerstand 47 Ohm
1 x 4K7 Potentiometer (so klein wie möglich). Man kann das Potentiometer auch durch zwei Widerstände ersetzen. Mit dem Potentiometer stellt man den Kontrast des Displays ein. Bei meinem LCD Display ergibt ein 100Ohm Widerstand zusammen mit einem 1K Widerstand einen guten Kontrast.
1 x Z−Diode 4.3V
2 x kleine Taster
1 x 470uF Elko
1 x 4,7uF Elko
1 x 1N4001 Diode
1 x 74HC02 TTL IC
1 x 74F74 fast TTL IC
1 x 74HC390 TTL IC
1 x LM393 voltage Komparator
1 x MAX903 high speed voltage Komparator, 8 Pin plastic DIP package, man kann dieses IC von www.maxim−ic.com bestellen, falls der Laden vor Ort dieses IC nicht auf Lager hat.
1 x 7805 5V regulator
1 x 2 Zeilen 20 Zeichen LCD Display, HD44780 kompatibel mit oder ohne Hintergrundbeleuchtung (16 Pin oder 14 Pin)

Alle LCD Displays mit 14 oder 16 Pins, die ich jemals gesehen habe, waren HD44780 kompatibel. Man kann auch ein 3 oder 4 zeiliges Display benutzen, aber dann muß man die Software etwas ändern.

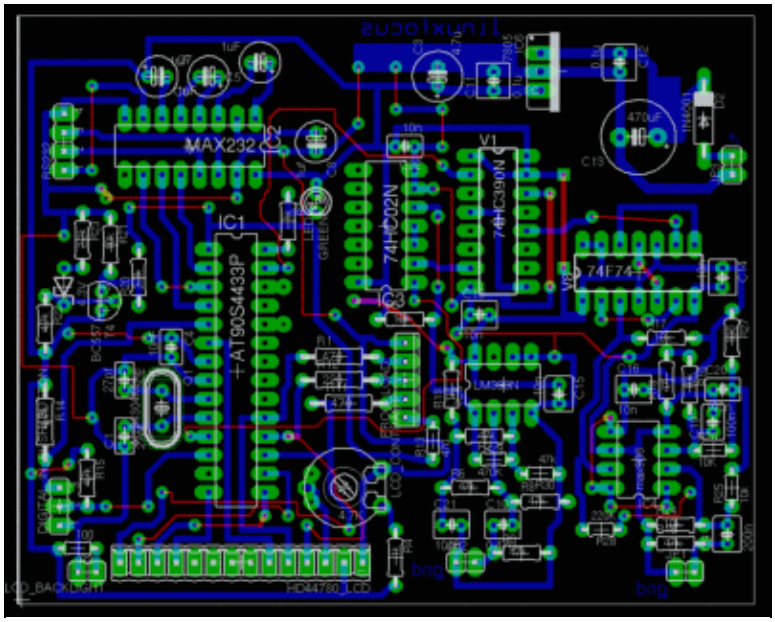
Zusätzlich zu diesen Bauteilen braucht man noch einige Drähte, Stecker (BCD, Stromversorgung, RS232) und einen 9V Transformator oder ein anderes DC oder AC Netzteil mit 150mA. Manchmal findet man sehr preiswerte Stromversorgungen, die sich direkt in die Steckdose stecken lassen und für irgendwelche Unterhaltungselektronik verwendet werden.

Schaltplan und Platine

Ich habe eagle für Linux benutzt, um Schaltplan und Platine zu entwickeln. Das Programm hat etwas Probleme, zu verstehen, daß all die unterschiedlichen Stromversorgungsanschlüsse von den ICs immer 5V sind. Man erhält deshalb eine Fehlermeldung, wenn man den "electrical rule check" benutzt. Der Entwurf ist jedoch richtig.
Das Schaltbild, mit einem Klick auf das Bild erhält man einen größeren Plan:



Die Platine, mit einem Klick auf das Bild erhält man eine größere Ansicht:



Die Platine ist speziell für Hobby Elektronik entwickelt. Die blaue Lage soll in die Platine geätzt werden. Die roten Linien sind Drähte. Es ist viel einfacher eine einseitige Platine herzustellen, da die Genauigkeit nicht so gut sein muß. Man kann die roten Drähte so ziehen, daß möglichst kurze Wege entstehen. In Eagle konnte ich die roten Linien nicht so ziehen.
Die Platine mit einem weißen Hintergrund für einen besseren Ausdruck: Platine mit weißem Hintergrund
(Beachte: das ist nicht die Datei, die man braucht um die gedruckte Schaltung herzustellen)

Die Eagle Dateien sind zusammen mit der Software in dem Paket linuxfreqcount−0.4 enthalten, das man am Ende des Artikels herunterladen kann.

Wie es funktioniert

Der AT90S4433 Microcontroller hat zwei interne Zähler. Einer ist 16bit breit und der andere 8bit. Wir benutzen den 8bit Zähler, um eine genaue Zeitbasis aus der Frequenz des Quarzes zu erzeugen. Zu diesem Zweck nehmen wir ein 4194304Hz Quarz und schicken damit über einem internen 1/256 Vorteiler (siehe AT90S4433 Datenblatt, herunterladbar am Ende des Artikels) Pulse an den 8bit Zähler. Der 8bit Zähler ist so konfiguriert, daß er bei einem Überlauf einen Interrupt erzeugt. Mit anderen Worten, wir erhalten eine Zeitbasis von 4194304Hz / (256 * 256) = 64Hz. Über eine Schleifenvariable können wir damit Funktionsaufrufe in 1Hz oder 64Hz Intervallen erzeugen.

Nun haben wir eine Funktion, die in 1Hz oder 64Hz Intervallen aufgerufen wird in Abhängigkeit von dem Modus, in dem der Zähler läuft. Alles, was wir jetzt machen müssen, ist den 16bit Zähler in diesen Intervallen auszulesen und anzuzeigen. Der 16bit Zähler (Pin PD5 am Microcontroller) erhält seine Pulse über das Signal, das wir messen wollen.

Der Microcontroller sampelt seine Eingangssignale, um sie mit dem internen Takt zu synchronisieren. Gemäß des Sampel−Theorems kann man damit Maximal− Frequenzen bis zur Hälfte der Quarzfrequenz messen. Das ist die theoretische Grenze. In der Praxis kann man Signale bis 1,5MHz mit dem Microcontroller (bei einem 4Mhz Quarz) messen.

Um höhere Frequenzen zu messen, braucht man einen Vorteiler. Dieser besteht in unserem Fall aus dem 74F74 und dem 74HC390 IC. Der 74F74 wird als schneller 1/4 Teiler benutzt und der 74HC390 ist ein 1/25

Teiler. Wir können den 74HC390 nicht direkt als 1/100 Teiler benutzen, weil er nur bis 25MHz geht.

Die Schaltung hat zwei Eingänge. Einen über den 1/100 Teiler und einen direkten ungeteilten Eingang. In Abhängigkeit der Frequenz, die du messen möchtest, mußt du einen der beiden Eingänge benutzen (einen, nicht beide).

Falls du nur Frequenzen bis 1.5Mhz messen möchtest (z.B. um den Frequenzgang des Verstärkers deiner Stereoanlage zu messen), kannst du auch eine vereinfachte Version bauen und MAX903, 74F74 und den 74hc390 weglassen. Die Software bleibt die gleiche, die Platine muß man auch nicht unbedingt ändern.

Der Zweck der zwei voltage Komparatoren (MAX903 high speed, und LM393) ist, die Signale zu verstärken und Rechtecksignale zu erzeugen (z.B. Rechteck aus einem Sinussignal).

Die Platine wurde mit großer Sorgfalt entworfen, um zu vermeiden, daß die Komparatoren in der Nähe des Nulldurchgangs schwingen. Das kann besonders dann passieren, wenn dem eigentlichen Eingangssignal noch ein leichtes Rauschen überlagert ist. Hat man z.B. ein 100KHz Signal dem ein verrauschtes 1MHz Signal überlagert ist (z.B. Radiowellen), dann kann es passieren, daß einige 1MHz Pulse in der Nähe des Nulldurchgangs des 100KHz Signals mitgezählt werden.
Der 220K Widerstand an dem MAX903 kann einen gewissen Anteil an Rauschen unterdrücken, indem er für positive Rückkopplung sorgt.

Der Microcontroller kann über RS232 und zwei Taster gesteuert werden (ein "clear" Taster und ein "change counting mode" Taster).

Die zwei Eingänge werden über ein Odergatter (74hc02) kombiniert. Das Odergatter würde natürlich den Eingang Blockieren, wenn man den Stecker aus dem 1/100 Eingang zieht, während noch eine logisch "1" an dem Ausgang des 74hc390 liegt. Der 74hc390 wird daher gelöscht (pin 2), wenn man den "counting mode" wechselt oder auf "clear" drückt.

Die Software

Die Software für den Microcontroller konfiguriert zwei Zähler in Interruptmode. Wie das geht, ist in dem Datenblatt des AT90S4433 beschrieben (siehe Referenzen). Man muß dazu einige Register setzen. Es ist ein wichtiger Teil, aber sehr trocken. Ich werde es deshalb hier nicht wiederholen. Der größte Teil der Frequenzzähler−Logik ist in der Datei linuxfreqcount.c implementiert. Alle anderen Dateien sind "Bibliotheken" für LCD UART, etc. ... Wenn man Software für den Microcontroller schreibt, muß man darauf achten, daß man nicht mehr als 128Bytes Ram benutzt. Das ist alles, was wir haben. Man sollte geschachtelte Aufrufe vermeiden und wo sinnvoll, globale Variablen benutzen.

Wenn die Zählerregister richtig gesetzt sind, dann wird die Funktion SIGNAL(SIG_OVERFLOW0) im 64Hz Takt aufgerufen. Hier lesen wir den 16bit Zähler und setzen ein Flag (hflag), um später das Ergebnis in der Funktion handlecounterresult() weiter zu bearbeiten. Bei einer Gatterfrequenz von 1Hz kann man das Ergebnis direkt anzeigen. Bei 64Hz muß man mit 64 multiplizieren. Die Mathematikfunktionen, die man mit einem Microcontroller ausführen kann, sind begrenzt (wir haben keine schönen 32bit Register wie in einer Intel Pentium CPU). Glücklicherweise ist ein "mal 64" nur ein Linksverschieben um 6 im Binärsystem. Wir speichern das Endergebnis in 3x 8bit Variablen (counterval[3]).

Damit hätten wir den richtigen Wert als eine 24bit Zahl. Um sie anzuzeigen, müssen wir sie noch in dezimal ASCII konvertieren. Printf würde das normalerweise machen, aber wir haben kein printf. Wenn wir printf hätten, würde es all unseren Speicher aufbrauchen. Wir müssen deshalb unser eigenes ganz spezielles printf schreiben. Für die Binär− zu Dezimal− Konvertierung teilen wir die 24Bit Zahl durch 10 mit Rest. Der Rest ergibt unsere Dezimalzahlen. Unsere CPU kann keine 24bit Mathematikoperationen ausführen. Deshalb

müssen wir die Zahl in 8bit Stücken verarbeiten (Funktion divby10()). Wenn man am Ende zu jeder Zahl den ASCII Wert der Zahl "0" addiert, erhält man schließlich eine ASCII Darstellung der Zahl (Funktion longtoascii()).

Dieser ASCII String kann dann über die RS232 Schnittstelle geschickt werden und im LCD angezeigt werden.

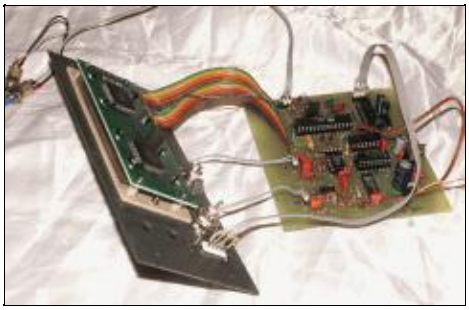
Erstellen einer gedruckten Schaltung

Das Softwarepaket enthält eine Postscriptdatei (linuxfcount.ps) für die gedruckte Schaltung. Persönlich finde ich, daß die Lötaugen immer etwas zu klein sind. Ich empfehle daher, alle Lötaugen vor dem Ätzen mit einem Edding Paint Marker nachzuzeichnen. Wie man eine Platine erstellt, ist in Mai 2002, Eine LCD Anzeige und Steuertasten für den Linux Server beschrieben. Einige Leser schrieben mir, daß sie Kontakt Pausklar−21 Spray nicht kaufen können. Man kann statt diesem Spray auch einfach Petroleum, wie es für Lampen benutzt wird, verwenden. Das Petroleum muß mit etwas Seife abgewaschen werden, bevor man die Platine in den Entwickler steckt.

Wie man ein Gehäuse für den Frequenzzähler baut

Ein großes Problem für den Hobbyelektroniker ist oft das Gehäuse. Vorgefertigte Aluminiumgehäuse sind sehr teuer. Ein genaues Biegen von Blechen ist mit den Mitteln der Heimwerkstatt oft nicht möglich. Ich habe eine, wie ich finde, sehr gute Lösung dieses Problems gefunden. Ich benutze Fichtenholz für die Seiten und Kanten. Man kann das Holz etwas beizen, damit es "teuer" aussieht. Hier sollte man nur wasserlösliche Beizen benutzen. Für die Front, oben und unten, nimmt man gerade Aluminiumbleche und man braucht nichts zu biegen. Einfach mit einer Säge auf die richtige Größe zuschneiden. Wie der Holzrahmen aussieht, kann man auf der rechten Seite sehen. Links ist das fertige Gehäuse (die obere Abdeckung ist noch offen).

Zusammenlöten der Platine



Beim Zusammenlöten sollte man den Hochfrequenzbauteilen etwas Aufmerksamkeit schenken (MAX903, 74F74 mit Widerständen und Kondensatoren). Ich empfehle normalerweise. Sockel zu benutzen, da sie die Fehlersuche enorm erleichtern. In diesem Fall sollte man die Bauteile aber direkt auf die Platine löten, um parasitäre Kapazitäten zu vermeiden. Für die Entkopplungskondensatoren zwischen Masse und Spannungsversorgung sollte man kleine keramische Kondensatoren mit Werten zwischen 10nF und 100nF benutzen.

Das ist die komplizierteste Schaltung, die wir bisher in LinuxFocus gebaut haben. Ich empfehle, die Schaltung in Schritten aufzubauen und dazwischen immer zu testen:

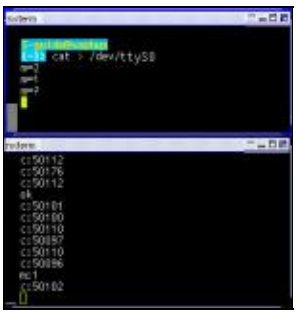
1. Löte erst die Teile für die Spannungsversorgung (7805 etc...) auf die Platine und teste dann, ob sie

funktioniert.
2. Löte alle Teile auf die Platine, aber stecke nur den Microcontroller in den Sockel.
3. Das linuxfreqcount−0.4 Paket enthält ein Testprogramm, avr_led_lcd_test.c, zum Testen des Microcontrollers, der LED und der Anzeige. Verbinde die Schaltung mit dem Programmierkabel wie in dem ersten AVR Artikel beschrieben (März 2002, Den AVR Microcontroller mit GCC programmieren) und tippe dann "make testload", um die avr_led_lcd_test Software aus dem linuxfreqcount−0.4 Paket zu laden. Die LED sollte blinken und auf dem LCD sollte "Hello" zu sehen sein.
4. Stecke nun alle ICs in die Sockel. Lade die endgültige Software (make load) und teste erst den Niederfrequenzeingang. Man braucht irgendeinen Frequenzgenerator (Sinus oder Rechteck vorzugsweise).

Den Zähler benutzen

Der Zähler kann in verschiedenen "modes" arbeiten:

1. Kontinuierlich hochzählen
2. Frequenzzählen mit einer Gatterfrequenz von 1Hz. In diesem "mode" kann man Frequenzen bis 16bit messen (65535Hz) oder 6553500Hz mit dem 1/100 Teiler.
3. Frequenzzählen mit einer Gatterfrequenz von 64Hz. In diesem "mode" kann man Frequenzen bis 1,5 MHz oder bis 100MHz mit dem 1/100 Teiler messen.



RS232 ASCII Kommandointerface.

Man kann den "mode" über den Taster oder über RS232 ändern. Die RS232 ASCII Befehle sind in README.commands beschrieben. Wie man das RS232 ASCII Interface benutzt, ist in Mai 2002, Eine LCD Anzeige und Steuertasten für den Linux Server beschrieben.

Die grüne Leuchtdiode zeigt an, daß das Gatter offen ist. Das heißt, jetzt können Pulse in den Zähler kommen.

Der zweite Taste ist gedacht, um den Zählerwert zu löschen (hauptsächlich, wenn der Zähler zum kontinuierlichen Zählen benutzt wird). Außerdem kann man damit die Anzeigendarstellung zwischen mit "100 multipliziert" und "normal" wechseln. Damit erspart man sich kompliziertes Kopfrechnen, falls man den 1/100 Teiler benutzt :−).

Wie genau?

Jeder digitale Zähler hat eine maximale Genauigkeit von +/−1. Die letzte Zahl wird sich immer etwas ändern, selbst wenn das Eingangssignal sehr stabil ist. Bei einer Gatterfrequenz von 64Hz wird das Ergebnis intern mit 64 multipliziert. Damit ergibt sich eine Genauigkeit von +/−64. Auf die gleiche Art hat der 1/100 Teiler Einfluß auf die Genauigkeit. Der Zähler ist jedoch sehr gut. Das Hauptproblem ist der Temperaturdrift des

4194304Hz Quarzes. Ein Quarz oszilliert langsamer, wenn es wärmer wird. Das ist Physik und man kann das nur über eine Temeraturregelung in den Griff bekommen.

Es ist möglich, den Zähler zu kalibrieren, wenn man ein gutes Referenzsignal hat. In der Software kann man kalibrieren, indem man kleine Verzögerungen in der Funktion handlecounterresult() einbaut. Ich habe meinen Zähler mit einem Referenzsignal verglichen und er war fast 100% exakt. Der Kalibrierungscode ist deshalb momentan auskommentiert. Man kann auch die Frequenz des Quarzes etwas ändern, indem man einen der 27pF Kondensatoren ändert (man muß mit verschiedenen Werten experimentieren 10pF, 50pF etc..).

Für den normalen Hausgebrauch sehe ich jedoch keine Notwendigkeit, etwas zu kalibrieren. Normale 4194304Hz Quarze sind schon sehr genau.

Der Zähler im Betrieb

Hier ist ein Foto des Zählers:

Weitere Projekte

In diesem Artikel wurde keine Auswertungssoftware geschrieben. Man könnte sich z.B. vorstellen, daß man mit dem Zähler Windgeschwindigkeit oder andere Dinge an einem entfernten Ort mißt. Um Langzeitmessungen zu machen, kann man z.B die Software aus Mai 2002, Eine LCD Anzeige und Steuertasten für den Linux Server ändern. Wenn man die Daten jedoch einfach in eine Datei schreiben möchte, dann reichen folgende Befehle:

```
ttydevinit /dev/ttyS0
cat /dev/ttyS0 > your_logfile.txt
```

Hier wird angenommen, daß der Zähler an COM1 (=ttyS0) angeschlossen ist.
Wirklich einfach, nicht wahr:−)?

Referenzen

- Die uisp AVR Programiersoftware: www.amelek.gda.pl/avr/
  Lokale Kopie: uisp−20011025.tar.gz
- Wie man die Linux AVR Entwicklungsumgebung installiert und benutzt und wie man einen Programmierer baut:
  März 2002, Den AVR Microcontroller mit GCC programmieren

- Der Quellcode für diesen Artikel: linuxfreqcount−0.4.tar.gz Der Schaltplan, die Eagle Dateien und einige Bilder sind auch enthalten.
- Alle Software (neuere Versionen werden auf dieser Seite auftauchen) und Dokumente:software/datasheets
- Datenblatt für 74hc390 74hc390.pdf 48K
- Datenblatt für 74f74 sn54f74.pdf 84K
- Datenblatt für LM393 LM193.pdf 348K
- Datenblatt für MAX903 MAX900−MAX903.pdf 164K
- Datenblatt für MAX232 MAX220−MAX249.pdf 448K
- Datenblatt für ST232, eine billige Variante, oft anstelle des echten MAX232 zu haben st232.pdf 100K
- Datenblatt für Atmel AT90S4433 avr4433.pdf 2356K
- Die Atmel Webseite: www.atmel.com/
- Eagle für Linux cadsoftusa.com

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

2005−02−12, generated by lfparser_pdf version 2.51