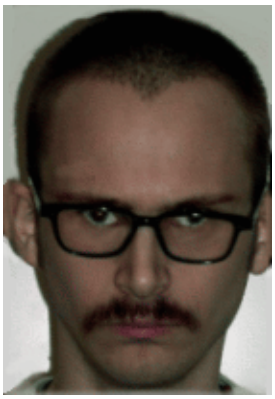
HD44780 kompatible LCD–Displays verstehen

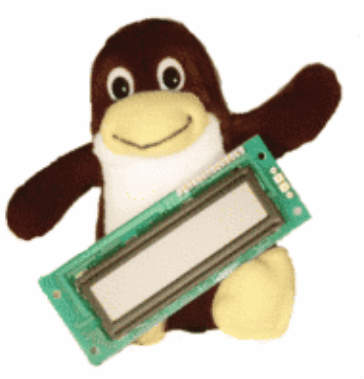


von Jan Svenungson
<jan.svenungson(at)linux.nu>

Über den Autor:

Jan nutzt GNU/Linux seit 1996 und hatte seitdem zwei unbeabsichtigte Reboots (abgesehen von den Reboots aufgrund von Stromproblemen).

Übersetzt ins Deutsche von:
Michael Weinrich
<m.weinrich(at)web.de>



Zusammenfassung:

Dieser Artikel versucht, Dir das eine oder andere über HD44780 kompatible LCD–Displays beizubringen.
Wir werden durchgehen, wie ein HD44780 an Deinen Parallelport angeschlossen wird und wie es mit einem Beispielprogramm namens LCDInfo programmiert wird.
Du sollst nicht nur das Display anschließen, ein Programm starten und alles, was Du willst, auf dem Display darstellen, sondern auch verstehen, wie die Hardware das machen kann, was Du von ihr willst.

Einführung

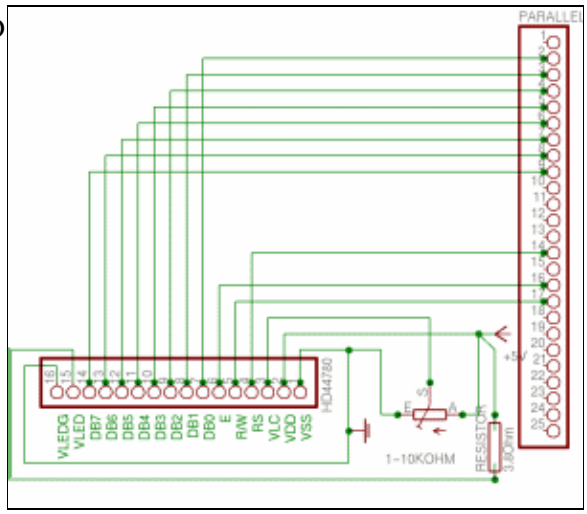
Als erstes brauchst Du etwas Hard– und Software, ich vermute, Du hast bereits einen Computer mit einem Standard Parallel– (Drucker–) Port, auf dem Du GNU/Linux mit gcc und glibc laufen lassen kannst.
Du brauchst auch ein LCD–Display, das HD44780 kompatibel ist, Kabel, um dieses mit Deinem Parallelport zu verbinden und außerdem noch einen Potentiometer, wenn Du den Kontrast des Displays verändern möchtest. Um das Display mit Strom zu versorgen, brauchst Du mit Sicherheit mehr Saft, als Dir Dein Parallelport geben wird, also musst Du den Strom von woanders aus Deinem Computer holen. Am

einfachsten ist es, die Standard +5V Verbindung zu benutzen, die auch verwendet wird, um Diskettenlaufwerke, Festplatten undsoweiter zu versorgen.

Wenn Du das LCD−Display angeschlossen hast, musst Du wissen, wie es funktioniert. Dieses wird meistens in den anderen Artikeln über dieses Thema ausgelassen, aber ich will versuchen, einige Interna des Displays zu erläutern, was Dir bei der Programmierung helfen wird.

Als letztes müssen wir das Display dazu bringen, etwas sinnvolles auszugeben. Als Beispiel werde ich ein kleines Programm namens LCDInfo verwenden, was die meisten Funktionen des HD44780 unterstützt, aber im Moment noch nicht viel ausdruckt. Dieses Programm ist noch im Alpha−Stadium und ich arbeite daran, wenn ich etwas Zeit übrig habe.
Wenn Du noch nie in C programmiert hast, magst Du Dir überlegen, ein bißchen über C zu lesen, ich gehe aber davon aus, dass Du ein C−Anfänger bist, weil ich mich da selbst auch noch zu zähle.

Wie es angeschlossen wird



Lass uns zuerst die verschiedenen Pins ansehen, die das LCD zur Verfügung stellt und erklären, was sie tun.
Pin 1 wird als VSS bezeichnet und soll an die Masse (GND) angeschlossen werden.
Pin 2 wird als VDD bezeichnet und ist der Stromversorgungs−Pin, also +5V
Pin 3 wird als VLC bezeichnet und wird mit dem Potentiometer verbunden, um den Kontrast des Displays einzustellen.
Pin 4 ist der RS Pin und in Abhängigkeit dieses Pins bereitet sich das Display darauf vor, *Befehle* oder *Daten* zu bekommen.
Pin 5 ist der R/W Pin und entscheidet, ob das LCD *sendet* oder *empfängt*.
Pin 6 ist der Freigabe Pin. Wenn er von runter auf rauf und dann wieder auf runter geht, liest das LCD die Pins 4,5 und 7−14.
Die Pins 7−14 sind der *Daten−Bus−Strang* und werden mit DB0−DB7 bezeichnet. Diese sind die eigentlichen Datenbits, die an das LCD gesendet werden und legen fest, wo und was auf das Display geschrieben wird.
Die Pins 15 und 16 gibt es nur auf Displays mit Hintergrundbeleuchtung und werden einfach an +5V und Masse (GND) mit einem 3,8 Ohm Widerstand zwischen Pin 15 und +5V angeschlossen.

Um herauszufinden, wo Du diese Pins mit dem Druckerport verbinden musst, kannst Du Dir das Schaltbild rechts ansehen, in dem ich versucht habe, dies so klar wie möglich darzustellen. Klick auf das Schaltbild, um ein größeres Bild zu bekommen.
Dieses Schaltbild trifft nur zu, wenn Du den Kontrast des Displays einstellen willst. Ich habe einfach Pin 3 und Pin 1 an die Masse angeschlossen und das Ergebnis ist okay für mich − wenn Du eine seltsame Beleuchtung im Raum hast, kannst Du in Erwägung ziehen, einen Potentiometer einzubauen.

Sei bitte vorsichtig, wenn Du die Stromversorgung vom PC verwendest. Wenn Du die Spannung von dem falschen Kabel nimmst, bekommst Du +12V, was Dein LCD abrauchen lassen wird. Das Kabel, das Du brauchst, ist das rote. Gelb ist +12V und schwarz ist die Masse (GND).
Wenn Du dies richtig gemacht hast, sollte das LCD die erste (und dritte, falls es sie gibt) Zeile schwarz anzeigen, wenn Du den PC einschaltest.

Wie das LCD arbeitet

Das LCD macht gar nichts, bis Du ihm sagst, dass es etwas tun soll, es wartet einfach bis es einen gültigen rauf runter Impuls bekommt (das heißt, wenn wir den Freigabe Pin rauf setzten, eine Weile warten und ihn wieder runter setzen). Dann liest das Display, ob es Befehle oder Daten zur Verarbeitung bekommt und danach, ob es Informationen erhält oder sendet und schließlich die Datenbits, die gesendet oder empfangen werden.
In diesem Artikel werden wir niemals Informationen von dem LCD erhalten, also wird der R/W Pin immer runtergesetzt sein, was "schreiben" bedeutet.
Der RS Pin wird immer runter sein, außer wenn wir Zeichen schreiben, alles andere wird als Befehl angesehen.
Dies macht es wirklich einfach, das Display zu programmieren.

Nun, wo wir dies alles wissen, wollen wir damit beginnen, das Display einzuschalten und bereit zu machen, Informationen zu erhalten. Dies geschieht in der Initialisierungssequenz, bei der wir dem Display sagen, dass es eingeschaltet wird, welches Funktionsset genutzt werden sollen undsoweiter.
Der Strom sollte bereits eingeschaltet sein, wenn Du den Strom von einem freien Spannungskabel Deines PC bekommst, anderenfalls ist das das erste, was getan werden muss.
Als nächstes ist das "Funktionsset" dran, welches von Deinem Display abhängt.

Um es leichter verständlich zu machen, werde ich genau beschreiben, was wir tun, wenn wir das Funktionsset benutzen.
DB2 ist das Schriftart Bit und sollte *runter* gesetzt sein, das heißt 5x7 Matrix.
BD3 ist das Display ZeilenBit und sollte *rauf* gesetzt sein, was 2 Zeilen bedeutet. Was passiert, wenn Du 4 Zeilen auf dem Display hast? Keine Angst, die erste und dritte Zeile sind im Display Speicher gleich, also solltest Du BD3 auf rauf setzen.
DB4 ist das Daten Längen Bit und entscheidet, ob Du 4 oder 8 DB hast. Wenn Du das Display wie in meinem Schaltbild hast, solltest Du dieses DB auf *rauf* setzen.
Dann setze *DB5* rauf, um dem Display zu sagen, dass dieses tatsächlich eine "Funktionsset"−Anweisung ist und vergewissere Dich, dass *RS* und *R/W runter* gesetzt sind und löse eine Freigabe mit rauf runter aus. Was Zeiten angeht, lies das Handbuch, ich gehe davon aus, dass wir nur für Mikrosekunden anhalten, wenn wir auf das Display warten, was mehr als genug sein sollte.

Was ist mit dem Code?

Hier werden wir die Teile des LCDInfo Programms diskutieren, die Du brauchst, wenn Du verstehen willst, wie das HD44780 arbeitet. Du kannst das LCDInfo Programm am Ende des Artikels herunterladen oder direkt einen Blick auf die C−Code Dateien iolcd.c und lcdinfo.c werfen, indem Du hier klickst.

Jetzt müssen wir die oben genannten Anweisungen in C schreiben, und glaub' mir, das ist wirklich einfach. Ich werde Schritt für Schritt durch den Code gehen und Du wirst ihn sogar verstehen, wenn Du ein C−Anfänger bist.
Zuerst binden wir ein paar Header−Dateien ein und definieren Funktionen (siehe Quelltext). Dann kommt der spaßige Teil.

```
#define D_REGISTER 0
#define I_REGISTER 2
#define WRITE_DATA 8
#define BASE 0x378

int main(void)
{
  ioperm(BASE,3,1);
  [CUT]
}
```

Dies ist die erste Anweisung in der Hauptfunktion, die uns die Berechtigung für den Parallelport erteilt. BASE sollte 0x378 oder so sein und die "3" bedeutet, dass wir Zugriff zu 0x378, 0x379 und 0x380 erhalten, was im Grunde der gesamte Druckerport ist.
Der Grund für die drei Adressen liegt darin, dass der Port in Daten, Status, und Kontrolle eingeteilt ist. Für uns bedeutet das, dass wir zuerst die Data Pins setzen müssen und dann die Kontroll Pins und dies nicht in einem einzigen Befehl tun können.
Als nächstes müssen wir uns das Funktionsset vornehmen, das wir oben beschrieben haben.

```
void function_set(void)
{
  outb(56, BASE);
```

Dieses setzt die DB Pins auf 5x7 Punkt Matrix, zwei Zeilen undsoweiter.

```
  outb(I_REGISTER + WRITE_DATA, BASE + 2);
```

Dieses setzt die RS und R/W pins auf "Befehl" und "Schreiben". Ich habe globale Variablen aus I_REGISTER und WRITE_DATA gemacht und sie auf 2 und 8 gesetzt.
Als nächstes kommt das rauf und runter für die Freigabe.

```
  outb(ENABLE + I_REGISTER + WRITE_DATA, BASE + 2);
  usleep(0);
  outb(I_REGISTER + WRITE_DATA, BASE + 2);
}
```

Dieser Code macht grundsätzlich nichts anderes, als die Freigabe raufzusetzen, zu warten und sie dann wieder runterzusetzen. Der Befehl für usleep(0); ist nicht wirklich ideal, aber ich habe den Code für das Timing des Displays noch nicht fertiggestellt.
Viele von Euch werden sich fragen, warum ich RS und R/W im Code auf *rauf* setze, wenn ich in den Befehlen sage, dass ich sie runtersetze. Das liegt daran, dass die Pins 1, 14 und 17 "Hardware invertiert" sind, das heißt, wenn Pin 14 "aus" ist, was den Drucker Port betrifft, so ist der Pin auf "rauf" gesetzt.
Nun, ich hab' doch gesagt, dass es einfach ist, oder?

Wie ich Zeichen angezeigt bekomme

Vielleicht möchtest Du ja auch einen praktischen Nutzen für Dein Display haben, zum Beispiel Text anzeigen? Kein Problem.
Der Code (Code wie in Befehlen) ist das gleiche wie Schreiben eines Zeichens und Setzen der Funktionen. Das einzige, was wir tun müssen, ist, einige Variablen zu ändern. Zuerst einmal wollen wir den RS nicht auf Befehl setzen, sondern auf Daten. Das heißt, die Funktion print_character() sieht so aus:

```
void print_character(int character)
{
  outb(D_REGISTER + WRITE_DATA, BASE + 2);
  outb(character, BASE);
  outb(ENABLE + D_REGISTER + WRITE_DATA, BASE + 2);
  usleep(0);
  outb(D_REGISTER + WRITE_DATA, BASE + 2);
}
```

Wie Du siehst, haben wir "I_REGISTER" in "D_REGISTER" geändert und "56" in "character", aber was bedeutet das? Wenn Du Dir die Zeichencodes in Deinem Handbuch ansiehst, wirst Du es verstehen.
Wir brauchen der Funktion nur ein Zeichen zu übergeben (da wir C benutzen, brauchen wir uns noch nicht einmal darum zu kümmern, es erst zu einem Integer zu machen) und schon wird das Zeichen auf dem Display angezeigt. Ist doch nett, oder?

Mit diesem Code hast Du die Grundlagen eines LCD Programms, benutze sie, um sie Deinen Wünschen entsprechend anzupassen, zeige freien Speicher an, gib die aktive http−Verbindung aus oder was auch immer. Einige Beispiele sind im LCDInfo Programm, welches einige Dinge anzeigt, die im Proc Dateisystem auf einem GNU/Linux−Computer zur Verfügung stehen.

Referenzen

- Für Infos über den Druckerpsort schau Dir folgendes http://et.nmsu.edu/~etti/fall96/computer/printer/printer.html an, wo es einige Beispiele gibt. (Eine lokale Kopie dieses Artikels ist >hier<) zu finden.
- Für Infos zum LCD Programm geh' zu http://lcdproc.omnipotent.net/, welches ein gutes LCD Programm ist

Dank an Sven and Reinhold für einige Hinweise.

- Der Quellcode des lcdinfo Programms: lcdinfo−0.02.tar.bz2.
  Updates werden unter http://savannah.gnu.org/download/lcdinfo/ erreichbar sein.

Der LinuxFocus Redaktion schreiben © Jan Svenungson "some rights reserved" see linuxfocus.org/license/ http://www.LinuxFocus.org	Autoren und Übersetzer: en −−> −− : Jan Svenungson <jan.svenungson(at)linux.nu> en −−> de: Michael Weinrich <m.weinrich(at)web.de>

2005−02−12, generated by lfparser_pdf version 2.51