Anzeigen und Displays

LED-Anzeigen und Displays aller Art

Jürgen Plate, 31. Oktober 2016

Inhaltsverzeichnis

1

Anzeigen und Displays

Dieses Skript wird sich nach einem kurzen Abstecher zu den LED-Anzeigen hauptsächlich mit Text- und Grafikanzeigen auf Flüssigkristall- und OLED-Basis beschäftigen und zeigen, wie diese angesteuert werden.

Die Frage „Wozu ein Display an einen PC anschließen, man hat ja schließlich einen Bildschirm?“ ist auf den ersten Blick berechtigt. Setzt man einen Linux-Rechner beispielsweise als Server oder Netzwerkrouter ein, werden unter Umständen die Tastatur, der Bildschirm und andere Peripheriegeräte überflüssig. Um dennoch bestimmte Systeminfos vom PC zu erhalten, hat man sich diese einfache Lösung ausgedacht; auch für spezielle Steuersysteme bietet sich das Extra-Display an.

Bei Mikrocontrollern stellt sich die Frage genau anders herum: „Womit soll das Board mit dem menschlichen Bediener kommunizieren?“ Da sind dann einige LEDs ganz nützlich. Uns Menschen ist halt immer noch ein Gerät etwas unheimlich, das ohne jede äußeren Anzeichen seine Tätigkeit verrichtet. Schon eine einzige blinkende LED kann ausreichend sein – wie beispielsweise bei meinem Notebook im Ruhezustand. Manche Entwickler planen bei jeder Appliance sogar eine Blink-LED als Betriebsanzeige ein.

1.1 LED-Anzeigen

Dieser Abschnitt beschäftigt sich nochmals ganz knapp mit LEDs (Leuchtdiode, light emitting diode), LED-Zeilen und Siebensegmentanzeigen. Eine LED (Bild 1.1) ist ein Halbleiter mit PN-Übergang, also eine Diode, die Licht aussendet, wenn sie in Durchlassrichtung betrieben wird. Das Licht wird durch Elektroluminiszenz bei der Elektronen-Löcher-Rekombination an der Sperrschicht hervorgerufen. Helligkeit und Lichtfarbe hängen von den verwendeten Materialien und vom Strom durch die LED ab. Da es sich um eine Diode handelt, die in Durchlassrichtung betrieben wird, muss der Strom schaltungstechnisch begrenzt werden. In der Regel geschieht dies durch einen Vorwiderstand, bei LEDs höherer Leistung aber auch durch eine elektronische Stromquelle.

Über der LED fällt einen Spannung ab, deren Höhe bauartbedingt und wesentlich höher als bei normalen Dioden ist. Vom Betrag dieser Durchlass-Spannung V_d und dem Strom durch die LED I_d hängt der Wert des Vorwiderstandes ab. Berechnet wird er nach dem Ohmschen Gesetz:

$$R = \frac{V_{cc} - V_d}{I_d} \tag{1.1}$$

Die Durchlassspannung hängt von der LED-Farbe und -Bauart ab, in der Regel gelten die Angaben aus Tabelle 1.1. Bild 1.2 zeigt das Verhältnis von Durchlass-Strom und -Spannung für die verschiedenen Farben.

Bei Hintereinanderschaltung mehrerer LEDs addieren sich deren Durchlassspannungen, was insbesondere bei weißen LEDs schnell zu sehr hohem Spannungsbedarf führt. Übersteigt der Spannungsbedarf die zur Verfügung stehende Versorgungsspannung und kann man nicht auf Parallelschaltung

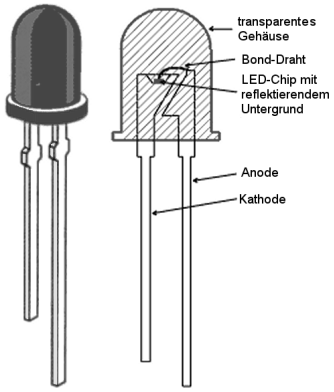

Bild 1.1: Aufbau einer LED

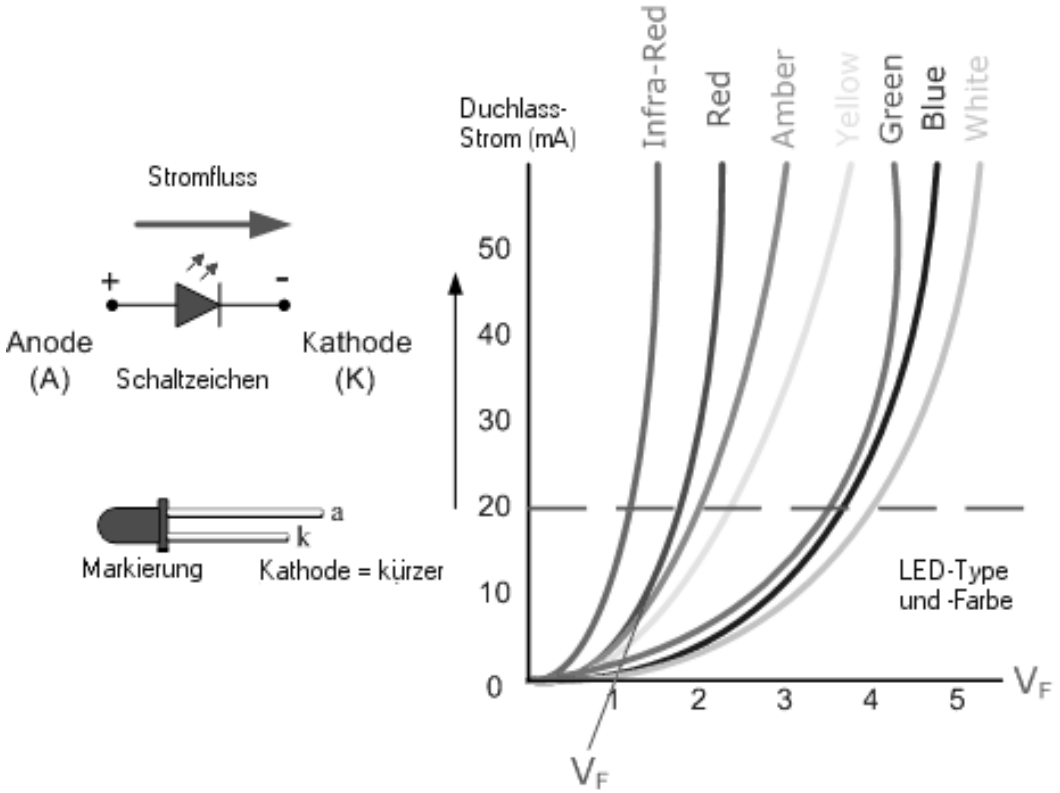

Bild 1.2: Durchlass-Strom und -Spannung verschiedener LED-Typen

Tabelle 1.1: Durchlassspannungen von LEDs

infrarot	1,5 V	rot	1,6 V	gelb	2,2 V
grün	2,1 V	blau/uv	2,9 V	weiß	4,0 V

ausweichen, hilft ein DC-DC-Wandler mit integriertem Stepup-Regler – der dann auch gleich die Stromregelung übernimmt.

Schon mit einzelnen LEDs kann man oft genügend Rückmeldung über die Tätigkeit einer Appliance erzeugen, so dass sich eventuell komplexere Anzeigen erübrigen. Oft ist auch die Schaltung an sich räumlich beschränkt und es lassen sich sowieso nur einzelne SMD-LEDs unter bringen. Manchmal ist auch die Zahl der für die Ausgabe verfügbaren E/A-Ports des Controllers recht gering. Das alles kann aber zu kreativen Anzeigelösungen mit wenigen LEDs führen.

Mit Zweifarben-LEDs lassen sich alternierende Anzeigen realisieren, die mit weniger Ausgangsleitungen des Controllers auskommen. So zeigt die linke Schaltung im Bild 1.3, wie mit einer Leitung zwei Zustände aktiv signalisiert werden können. Je nachdem, ob das Eingangssignal 0 oder 1 ist,

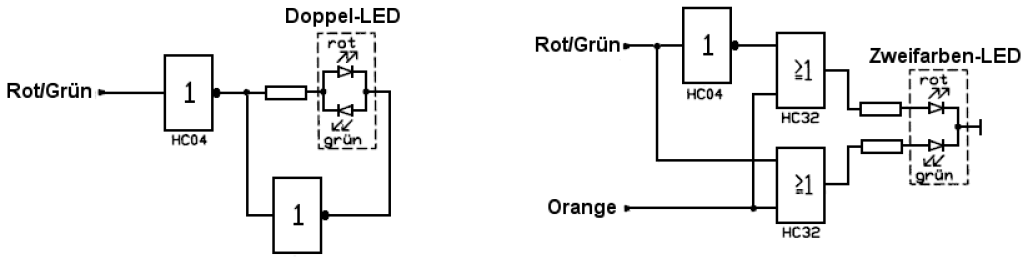

Bild 1.3: Ansteuerung einer Zweifarben-LED

leuchtet die LED rot oder grün. Das Leuchten der LED an sich zeigt gleichzeitig z. B. die Betriebsbereitschaft an, mit der Farbe werden dann zwei weitere Zustände signalisiert.[1] Die verwendete LED ist eine sogenannte Doppel-LED mit zwei Anschlussbeinen, wobei je nach Richtung des Stroms eine der beiden internen LEDS leuchtet.

Die rechte Schaltung erweitert die Anzeige um einen weiteren Eingang. Hier kommt eine Duo-LED zum Einsatz, bei der die beiden Kathodenanschlüsse intern verbunden, die Anoden aber getrennt herausgeführt sind. Mit zwei Steuerleitungen können nun – neben „Aus“ – folgende Zustände angezeigt werden:

- Rot: „Rot/Grün“ = 0, „Orange“ = 0
- Grün: „Rot/Grün“ = 1, „Orange“ = 0
- Orange: „Rot/Grün“ = x, „Orange“ = 1

Das Spiel kann natürlich noch weiter getrieben werden. Bei Einsatz einer RGB-LED, die im Gehäuse die Farben Rot, Grün und Blau vereint, lassen sich durch Farbmischung mit drei Leitungen sieben Farben (= Zustände) darstellen. Hier ist aber die Lernkurve beim Benutzer schon fast zu steil.

Es müssen auch nicht unbedingt digitale Bausteine zur Ansteuerung verwendet werden. In Bild 1.4 werden zwei LEDs mit gemeinsamen Vorwiderstand verwendet. Die obere LED muss immer die mit höherer Durchlass-Spannung sein (in der Regel grün, blau, weiß). Die untere LED hat eine geringere Durchlass-Spannung. Sie wird über einen PNP-Transistor geschaltet. Ist der Transistor gesperrt, leuchtet die obere LED. Ist er durchgeschaltet, leuchtet die untere LED. Will man zwei gleiche LEDs umschalten, wird zur oberen LED eine Diode in Reihe geschaltet, die die Schaltschwelle um ca. 0,6 V erhöht. Der Widerstand R wird so gewählt, dass ca. 10 - 20 mA fließen.

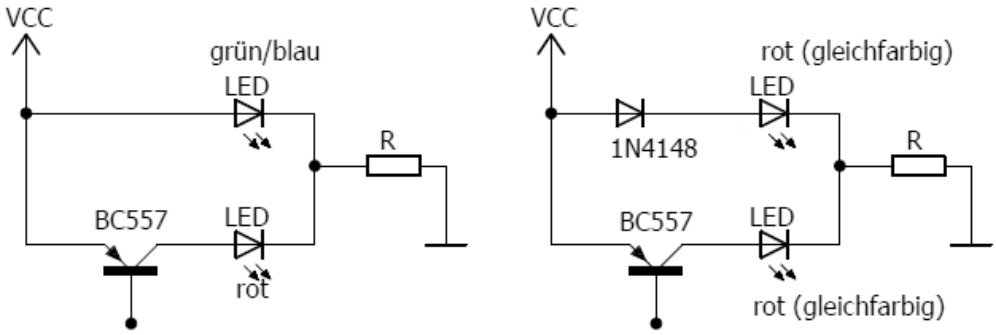

Bild 1.4: Umschalten zweier LEDS mit einem Signalport

Eine weitere Möglichkeit, die Anzeige zu variieren wäre das Blinken der LED mit unterschiedlicher Frequenz oder in unterschiedlichen Rhythmen.

[1] Einen Nachteil hat die Schaltung jedoch: Wenn jemand Rot-Grün-blind ist, kann eventuell nicht zwischen den beiden Zuständen unterschieden werden.

Zur digitalen Ansteuerung einer Leuchtpunktanzeige kann man einen 1-aus-n-Decoder einsetzen. Dabei wird diejenige LED eingeschaltet, die am selektierten Ausgang angeschlossen ist. Damit ist es beispielsweise möglich, mit vier Ausgangsleitungen des Controllers bis zu 16 LEDs anzusteuern – wenn auch nur immer eine einzige leuchten kann. In Bild 1.5 ist eine Realisierung mit dem 1-aus10-Decoder 74LS42 gezeigt, der den LED-Strom ohne weiteren Verstärker liefern kann. Da immer nur eine LED brennt, reicht auch ein einziger Widerstand zur Strombegrenzung aus.

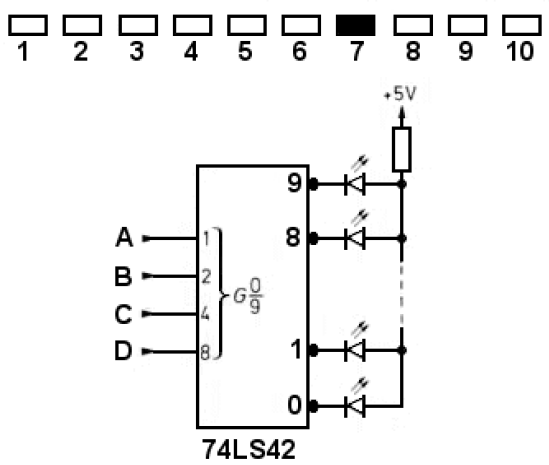

Bild 1.5: Ansteuerung einer LED-Zeile

Eine ständig wiederkehrende Aufgabe beim Einsatz von LEDs ist die Berechnung des strombegrenzenden Vorwiderstandes. Bei einer LED kann man sich mit der Faustregel für den Betrieb an 5 V merken: 470 Ohm für gewöhnliche LEDs und 1 k Ohm für Low-Power-LEDs. Bei der Reihenschaltung von LEDs und entsprechend höherer Betriebsspannung muss man nicht nur die an jeder LED abfallende Spannung kennen, sondern auch entsprechend rechnen. Mit nur zwei Transistoren kann man sich aber eine Stromquelle aufbauen, welche die Rechnerei auf die Dimension eines Widerstands reduziert und der es auch nichts ausmacht, wenn später LEDs hinzugefügt oder entfernt werden. Insofern ideal auch für LED-Streifen. Die Schaltung (Bild 1.6) besteht aus einem bipolaren Transistor und einem MOSFET. Der Strom durch die LEDs verursacht einen Spannungsabfall an R2, der ab ca. 0,6 V Basis-Emitter-Spannung die Gate-Source-Spannung an T2 so reguliert, dass sich ein konstanter Strom von ca. $I = 0,6/R1$ A ergibt. Über den Steuereingang können die LEDs mit einer Spannung von 5 - 12 V eingeschaltet und mit 0 V ausgeschaltet werden. Wird der Eingang mit einem PWM-Signal beaufschlagt, kann die Helligkeit der LEDs gesteuert werden. Die Versorgungsspannung der LEDs wird nur durch die maximale Drain-Source-Spannung des MOSFET begrenzt. Gegebenenfalls muss der MOSFET gekühlt werden.

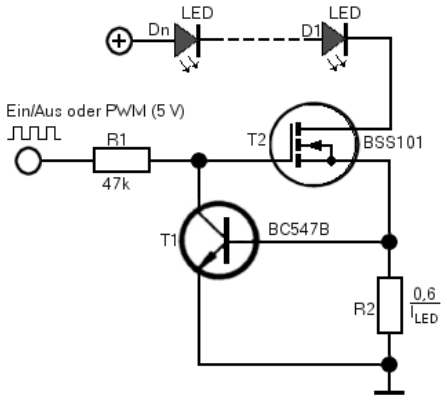

Bild 1.6: Einfache, diskret aufgebaute LED-Konstantstromquelle

Die in vielen Leuchtfarben erhältlichen LED-Stripes (Bild 1.7) erfreuen sich wachsender Beliebtheit. Sind sie doch sehr universell einsetzbar, z. B. als Dekorationsbeleuchtung in Vitrinen, Schränken usw., als optischer Gimmick oder als Warn- bzw. Sicherheitsbeleuchtung, etwa zur Markierung von Trep-

penstufen, als Leitlicht in dunklen Bereichen oder als Warnmarkierung. Die Streifen sind sowohl anreihbar (Verlängerung) als auch an bestimmten Stellen kürzbar. Es gibt sie auf einer starren „Platine“ montiert für ebene Untergründe oder flexibel zum Aufkleben auf Rundungen. Auch im Modellbau, insbesondere in der Modellbahntechnik, sind die vielseitigen Leuchtmodule gut einsetzbar, sei es als Waggon-Innenbeleuchtung oder für die Gebäude- oder Werbebeleuchtung.

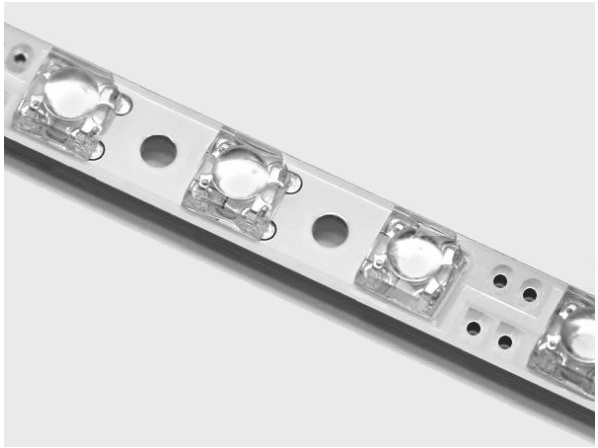

Bild 1.7: Teilbare Superflux-LED-Streifen für vielfältige Anwendungen

Die Kombination von LEDs führt dann zur bekannten Siebensegmentanzeige, wobei der Name streng genommen nicht stimmt, denn es ist fast immer noch eine achte LED für den Dezimalpunkt vorhanden. Diese Anzeige reicht für die Anzeige der Ziffern und auch einiger Buchstaben (z. B. A . . . F für hexadezimale Anzeige).

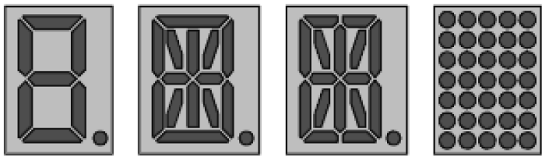

Bild 1.8: LED-Anzeigen mit 7, 14, 16 und 40 Elementen

Sollen mehr Symbole dargestellt werden, kann man zu 14- oder 16-Segment-Anzeigen greifen (auch wieder mit zusätzlichem Dezimalpunkt). Bei der 16-Segment-Anzeige sind gegenüber jener mit 14 Segmenten die beiden Segmente oben und unten nochmals geteilt (Bild 1.8). In der Regel sind die Anzeigen mit gemeinsamer Anode oder mit gemeinsamer Kathode lieferbar (aufpassen beim Schaltungsdesign!).

Noch variabler ist natürlich eine Punktmatrix-Anzeige, bei der sich alle LEDs einzeln ansteuern lassen. Module mit 5x8 LEDs eignen sich gut für alphanumerische Displays, Module mit 8x8 LED sind meist anreihbar und können zu größeren Grafikmatrizen kombiniert werden. Aber auch ein oder zwei Module bieten mannigfache Möglichkeiten und wirken selbst bei reiner Ziffernanzeige oft „eleganter“.

Siebensegment-Displays auf LED-Basis werden auch weiterhin bei Ausgaben von geringer Komplexität eingesetzt. Bezüglich Sichtbarkeit (auch unter ungünstigen Umgebungsbedingungen) und Ablesbarkeit aus großen Entfernungen haben sie immer noch keine Alternative. Zusätzlich zu den bisherigen Farben Rot, Gelb und Grün leuchten sie auch in Weiß und Blau. Außerdem gibt es Jumbo-Anzeigen mit bis zu 30 cm Höhe, bei denen jedes Segment aus etlichen LEDs gebildet wird. Oft ist auch gar keine Ziffernanzeige nötig oder wünschenswert, sondern beispielsweise ein „Bargraph“ aus LEDs – wie man ihn von Aussteuerungsanzeigen kennt. Die Balkenhöhe ist mit einem Blick erfassbar.

Normalerweise wird eine Siebensegmentanzeige an einen Decoder angeschlossen, der den BCD-Code am Eingang in den Siebensegmentcode umsetzt. Leider gibt es keine Decoder mehr auf dem Markt, die auch die Werte von 10. . . 15, entsprechend den Hex-Ziffern A. . . F umsetzen. Man muss hier ggf. selbst einen Decoder (mit GAL o. ä.) programmieren. Deshalb ein kleiner Tipp: Vor etwa

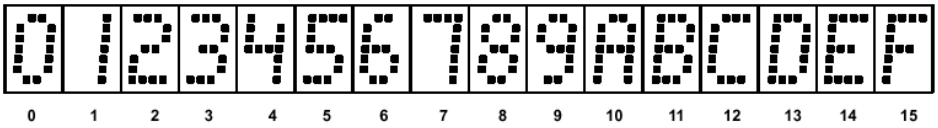

Bild 1.9: Code der Anzeige TIL 311

20 Jahren entwickelte Texas Instruments die Anzeige TIL311, die alle Ziffern, 0...9 und A...F, als LED-Punktmatrix darstellen kann. Bild 1.9 zeigt den Matrix-Code. Die Anzeige wird direkt mit 4-Bit-Binärcode gespeist, denn der Decoder ist in der Anzeige enthalten. Einige Anbieter haben sie immer noch im Programm, leider zu Wahnsinnspreisen (zwischen 20 und 30 Euro/Stück).

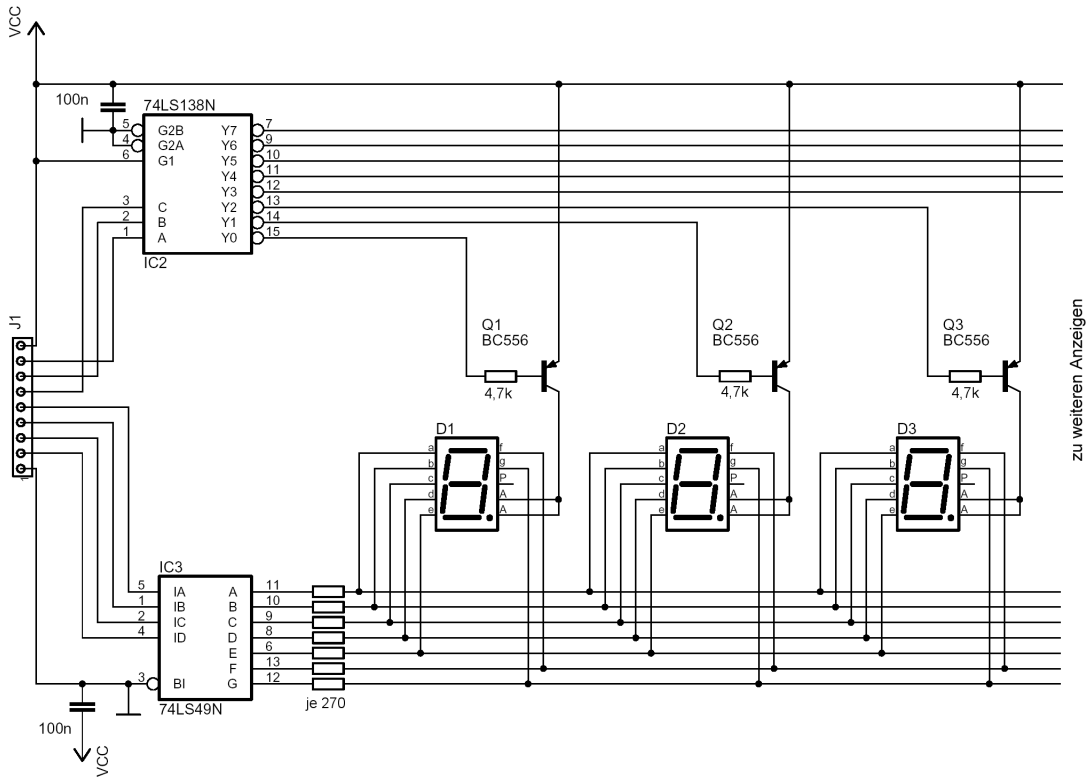

Bild 1.10: Typische Multiplexansteuerung von Siebensegmentanzeigen

1.1.1 Anzeigen multiplexen

Ein anderes Problem ist die Anzahl der benötigten Ausgabeports bei mehrstelligen Anzeigen. Selbst mit Decoder braucht man vier Bit pro Ziffer. Im Microcontrollerbereich findet man daher oft Multiplexlösungen wie in Bild 1.10. Die Anzeigegruppe wird mit acht Bit angesteuert, wobei vier Bit für den Anzeigecode und vier Bit für die Displayauswahl verwendet werden. Da für die Segmente ist sowieso ein Treiber/Verstärker notwendig. [2] Daher kann man – sofern nicht spezielle Darstellungen wie Vorzeichen oder Fehlersymbole benötigt werden – Controller-Leitungen sparen, wenn man einen BCD-zu-Siebensegment-Decoder verwendet. Dieser setzt den binären Code von 0 bis 9 auf die entsprechende Anzeige um. So werden statt sieben Leitungen nur noch vier benötigt. Beachten Sie auch, dass jedes Segment seinen eigenen Vorwiderstand benötigt (270 ... 470 Ohm). Hat man weniger Anzeigen, kann gegebenenfalls der Decoder 74LS138 entfallen und die Transistoren können direkt von den Controller-Ports angesteuert werden.

Damit alle Anzeigen zu leuchten scheinen, müssen die maximal 16 Anzeigen mindestens 25 bis 100 mal jede Sekunde angesteuert werden (Multiplexbetrieb, siehe unten). Da jede Anzeige nur kurzzeitig in Betrieb ist, muss auch der Strom erhöht werden, um den gleichen Helligkeitseindruck wie bei einer statischen Anzeige zu erzeugen.

[2] Der Mikrocontroller könnte beim direkten Anschluss der Segmente nicht genügend Strom liefern oder würde sogar defekt.

Ein ähnliches Problem tritt auch beim Ansteuern von Punktmatix-Anzeigen auf. In einer LED-Matrix sind jeweils die Kathoden und Anoden der LEDs jeweils zeilenweise bzw. spaltenweise miteinander verbunden (Bild 1.11). Der Vorteil besteht darin, dass nur wenige Leitungen nach außen geführt und angesteuert werden müssen. Auch der Verdrahtungsaufwand auf der Platine und die Zahl der benötigten Treiber sowie Vorwiderstände ist geringer. Die Ansteuerung muss dabei zwangsläufig im Multiplexbetrieb erfolgen. Prinzipiell können Zeilen oder Spalten gemultiplext werden. Im folgenden Beispiel erfolgt das Multiplexen der Spalten.

Die Programmierung erfolgt so, dass jeweils immer nur eine Spalte wirklich leuchtet. Die anderen Spalten sind abgeschaltet. Wird nun in schneller Folge jede Spalte einmal eingeschaltet, ergibt sich ein scheinbar vollständiges Bild, bei dem alle LEDs gleichzeitig leuchten – sofern die Wechselfrequenz hoch genug ist. Ab etwa 25 Hz verschmilzt aufgrund der Trägheit des menschlichen Auges eine Bilderfolge zur kontinuierlichen Bewegung, ab 70 bis 100 Hz ist das Bild flimmerfrei. Der prinzipielle Programmablauf ist recht einfach:

```
do forever
  {
  for (SP = 1; SP <= 5; SP++)
    {
    alle Spalten ausschalten;
    Muster für Spalte SP an Zeilen R1 \dots R7 anlegen;
    Spalte SP einschalten;
    sleep(Multiplexzeit);
    }
  }
```

Die Multiplexzeit lässt sich auch recht einfach berechnen. Wenn wir eine Frequenz von 100 Hz annehmen, dann muss die for-Schleife oben 100 mal pro Sekunde durchlaufen werden. Da die Schleife selbst fünf Durchläufe hat, ergeben sich 100 * 5 = 500 Schleifendurchläufe. das ergäbe eine Wartezeit von 1/500 Sekunde, also 2 ms. Da auch noch Zeit für das Ansteuern der Matrix selbst benötigt wird, würde man entweder auf 1,5 ms herunter gehen der die 2 ms beibehalten und sich mit eine etwas geringeren Multiplexfrequenz zufrieden geben.

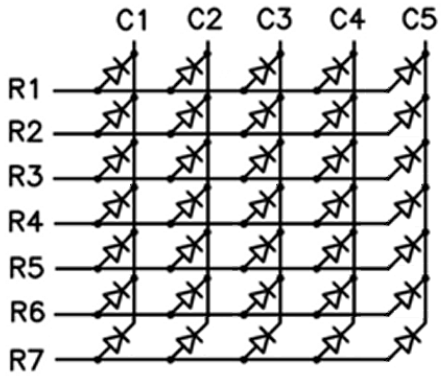

Bild 1.11: Typische LED-Matrix mit 5 Spalten und 7 Zeilen

Da jede der N Spalten einer Matrix immer nur für 1/N der Zeit aktiv ist, muss in dieser Zeit ungefähr die gleiche Lichtmenge abgegeben werden. Nur so erscheint die Anzeige genauso hell wie eine konstant mit Strom versorgte. Durch die LEDs muss also der N-fache Strom fließen. Entsprechend wären die Vorwiderstände bzw. Stromquellen zu dimensionieren. Diese Voraussetzung kann jedoch nicht immer erfüllt werden, denn:

- Der Strom durch eine LED kann nicht beliebig hoch sein – auch nicht bei Impulsbetrieb. Genaue Angaben zu den Maximalwerten liefert das Datenblatt der LED. Fehlt das Datenblatt, kann man das Zehnfache des Dauerstroms als Impulsstom ansetzen. Darüber hinaus werden die Impulsströme zu hoch. Bei einer LED-Matrix mit 15 mA Strom im Dauerbetrieb würde der Impulsstrom schon 150 mA betragen.
- Diese hohen Ströme vertragen die LEDs auch nur ganz kurzzeitig; die Ausschaltzeit wird benötigt, um den Chip um wieder abzukühlen. Die Einschaltzeit muss also wesentlich kürzer als die Ausschaltzeit sein. Bei der 5x7-Matrix im Beispiel ist das Verhältnis durch die for-Schleife auf 1:5 festgelegt.

Wegen des höheren Stroms im Multiplexbetrieb werden auch immer Treiberbausteine oder -transistoren notwendig sein. Die LEDs müssen aufgrund ihrer Kennlinie an einer Stromquelle betrieben werden.[3] Prinzipiell spielt es keine Rolle, ob man Zeilen oder Spalten einer Matrix multiplext, meist spiele weitere Faktoren mit herein. So wäre es bei einer Anzeigematrix mit acht Zeilen und 20 Spalten sicher sinnvoll, die Zeilen zu multiplexen und nicht die Spalten.

Für den Betrieb größerer LED-Matrizen eignet sich die unten gezeigte Porterweiterung mittels Schieberegister. Wer den Aufwand der Ansteuerung scheut, kann auf integrierte Lösungen für das direkte Betreiben einer LED-Matrix an der SPI- oder I^2C-Schnittstelle ausweichen. Beispiele hierfür sind die MAXIM-Bausteine 7219 und 7221 für 8x8-LED-Matrizen. Dieser Hersteller liefert auch ICs für 7x5-Matrizen oder 14- bzw. 16-Segment-Anzeigen.

Beim Multiplexbetrieb muss man ganz besonders auf korrekte Programmierung achten. Die LEDs sind ja immer nur kurze Zeit eingeschaltet und werden, damit sie etwa genauso hell erscheinen wie dauernd leuchtende LEDs, mit wesentlich höherem Strom betrieben. Wenn nun der Multiplexbetrieb aufgrund eines Softwarefehlers zum Erliegen kommt, würden die gerade leuchtenden LED den „Stromtod“ erleiden. Für die Testphase in der Softwareentwicklung sollten Sie deshalb die Ströme stark verringern. Die LEDs leuchten dann zwar wesentlich dunkler, überleben aber einen Softwareabsturz der Steuerung. Ist alles getestet, kann man den Strom durch die LEDs wieder auf die volle Höhe setzen.

Um im Betrieb sicher zu gehen, kann man die Dioden über eine Konstantstromquelle versorgen (siehe Seite 16) oder per Watchdogtimer vorsorgen – der Prozessor wird im Fehlerfall rechtzeitig wieder in einen definierten Zustand versetzt. Eine Hardwarelösung des Problems kann mittels eines retriggerbaren Monoflops erfolgen, das über den A-Eingang des Demultiplexers 74LS138 getriggert wird und die G-Eingänge des 74LS138 auf 0-Pegel zieht, solange es sich im metastabilen Zustand befindet (also getriggert wurde). Fällt das Multiplexen von Seiten des Controllers aus, werden alle Transistoren gesperrt und die Anzeige bleibt dunkel.

1.1.2 Porterweiterungen und intelligente Displays

Für einen Linux-PC wäre prinzipiell das Multiplexen kein Problem, nur haben wir nicht allzu viele Parallelports, und das Auffrischen der Anzeige könnte, je nach anderen laufenden Prozessen, etwas unregelmäßig stattfinden. Sollen anstelle der Siebensegmentanzeigen LED-Zeilen angesteuert werden, reichen die Portleitungen ohne Zusatzhardware auf keinen Fall aus. Deshalb will ich Ihnen hier eine Schaltung vorstellen, die den Prozessor kaum belastet, für eine statische Anzeige sorgt und beliebig erweiterbar ist. Wenn Sie die Binär-zu-Siebensegment-Decodierung in der Software vornehmen, wird auch kein Decoder benötigt.

In der Schaltung kommen Schieberegister vom Typ 4094 zum Einsatz. Diese Bausteine bestehen aus einem 8-Bit-Schieberegister, an dessen Parallelausgänge ein 8-Bit-Latch angeschlossen ist. Man kann also acht Bits seriell in das Schieberegister eintakten und diese dann in das Latch übernehmen. An dessen Ausgangsleitungen steht die binäre Information dann statisch an. Die 4094-Chips sind beliebig hintereinander zu schalten, so dass Sie 8, 16, 24, 32, ... Ausgangsleitungen ansteuern können, aber immer nur drei Portleitungen (Data, Clock, Store) brauchen. Statt des 4094 kann genauso gut auch ein 74HC595-Schieberegister verwendet werden.

Am Ausgang (Fan-out: 2 LS-Lasten) lassen sich alle möglichen Anzeigen anschließen:

- Siebensegmentanzeigen mit Decoder (der 4543 erlaubt sogar den Anschluss von LED- und LCD-Anzeigen); zwei Stellen/Schieberegister.
- Siebensegmentanzeigen ohne Decoder (aber mit Treiber als Verstärker, z. B. ULN2803); eine Stelle/Schieberegister.
- Einzel-LEDs mit Treiber; acht LEDs/Schieberegister. Bei Verwendung von Low-Power-LEDs könnten Sie sich sogar den Treiber sparen.
- LED-Strips (mehrere LEDs auf einer harten oder flexiblen Basis, die gemeinsam angesteuert werden) mit Treiber; acht Strips/Schieberegister.
- Prinzipiell sind auch Relais oder andere Aktoren denkbar.

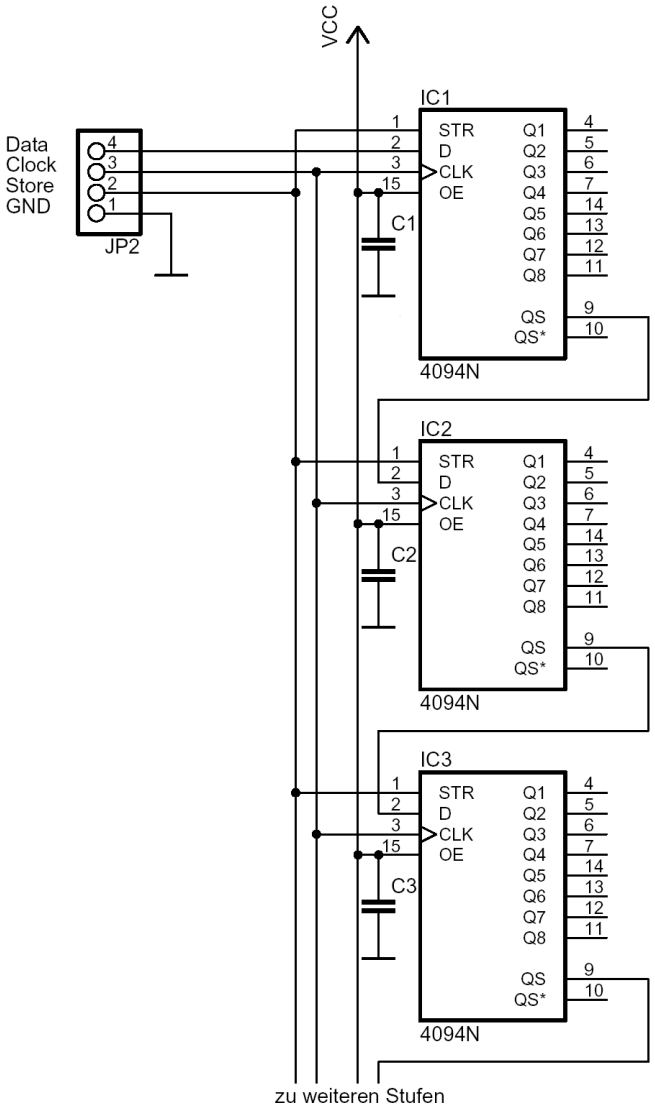

Bild 1.12: Ansteuerung von Anzeigen mittels Schieberegister

Tabelle 1.2: Ansteuerung einer Siebensegmentanzeige mit gem. Anode

Ziffer	Segmente .gfe dcba	Hex	Ziffer	Segmente .gfe dcba	Hex
0	1100 0000	$C0	8	1000 0000	$80
1	1111 1001	$F9	9	1001 1000	$98
2	1010 0100	$A4	A	1000 1000	$88
3	1011 0000	$B0	B	1000 0011	$83
4	1001 1001	$99	C	1111 0000	$F0
5	1001 0010	$92	D	1010 0001	$A1
6	1000 0010	$82	E	1000 0110	$86
7	1111 1000	$F8	F	1000 1110	$8E

Bild 1.12 zeigt die Schaltung ohne angeschlossene Treiber, und in Tabelle 1.2 ist der Siebensegmentcode für die Konvertierung angegeben. Die Ausgabe gemäß der Tabelle ist active low (Segment an = 0), eignet sich also für Anzeigen mit gemeinsamer Anode.

[3]Im einfachsten Fall ist das eine Spannungsquelle mit nachgeschaltetem Vorwiderstand. Besser ist jedoch eine spezielle Schaltung für die Versorgung der LEDs.

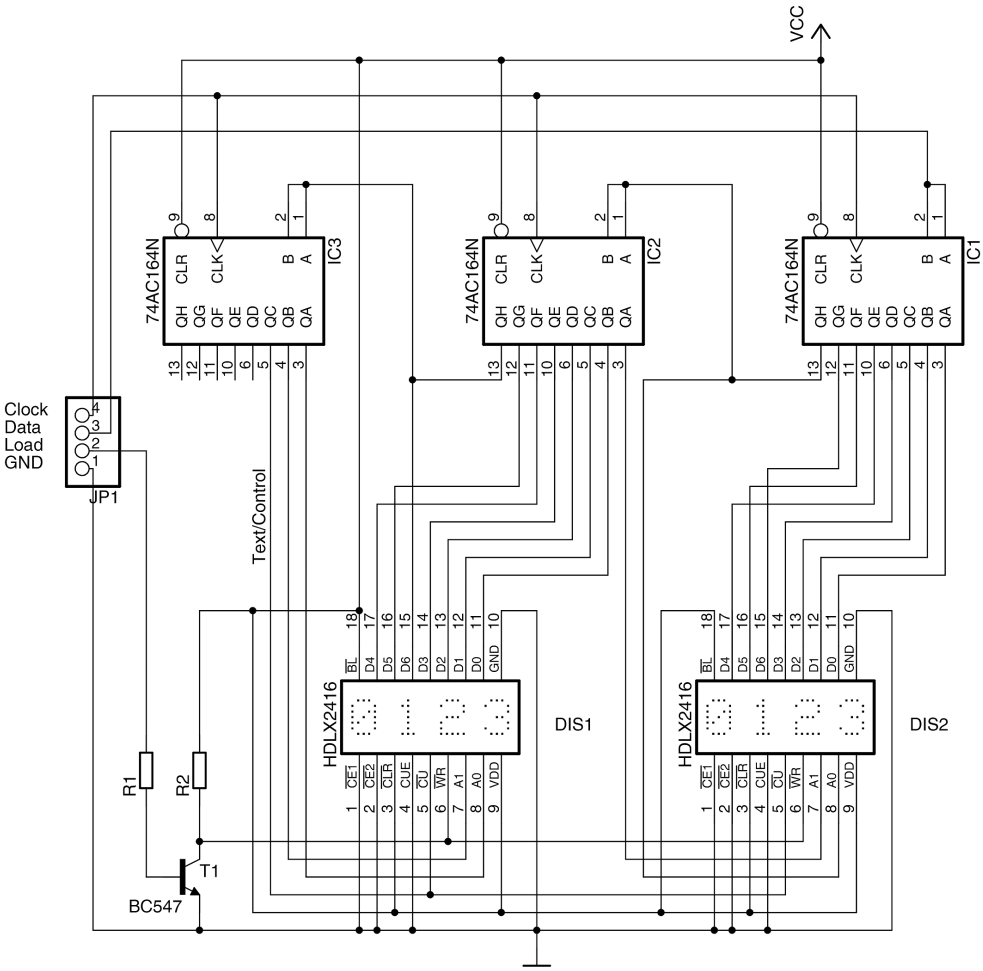

Bild 1.13: Ansteuerelektronik für die HDLX2416

Anstelle der Siebensegmentanzeigen kann man mit dem gezeigten Schema auch alphanumerische Anzeigen ansteuern. Auf dem Markt sind vierstellige 7x5-Matrixdisplays von verschiedenen Herstellern erhältlich, beispielsweise HDLX2416 von HP. Mit zwei dieser Anzeigen lassen sich acht Digits darstellen. Die Bausteine sind horizontal und vertikal anreihbar. Die Anzeige hat sieben parallele Daten- und einige Steuerleitungen. In Bild 1.13 ist die Schaltung einer achtstelligen Anzeige zu sehen.

Alle Bausteine sollten mit (nicht explizit gezeichneten) Kondensatoren (100 nF) gepuffert sein. Die Displays sind auch in helleren Umgebungen aus einiger Entfernung und von der Seite gut zu erkennen. Das Display HDLX2416 verfügt über einen internen Zeichengenerator, zwei Adressleitungen für die Zeichenposition und eine Leitung, die das ausgewählte Zeichen zur Anzeige bringt. Liegt an den Datenleitungen D0...D6 eine Bitkombination zwischen 0 und 127 an, wird ein Zeichen aus einem internen ASCII-Zeichensatz dargestellt. Der Zustand der Leitungen A0 und A1 bestimmt, an welcher der vier möglichen Positionen das Zeichen erscheinen soll. Ein kurzer Impuls an der Schreibleitung WR sorgt für die Übernahme des Zeichencodes. Es bleibt dort so lange stehen, bis es überschrieben wird. Die Chipselect-Leitungen und die Blanking-Leitung sind fest verdrahtet und können bei Bedarf auch vom PC aus gesteuert werden. Die Leitung CU schaltet zwischen ASCII- und Kommandomodus um. In letzterem können Helligkeit und einige Displayparameter eingestellt werden. Alle Leitungen (Daten, Adressen und CU) werden seriell in drei Schieberegister 74LS164 eingetaktet und dann durch Aktivierung der Write-Leitung ins Display übernommen. Einen Überblick der Ansteuermöglichkeiten liefert die Übersicht in Bild 1.14.

Alles, was oben beispielhaft für Siebensegmentanzeigen beschrieben wurde, gilt natürlich analog für LED-Zeilen oder frei gestaltete LED-Anzeigen. Für das Ansteuern von Punktmatrix-Anzeigen reicht die Anzahl der Ports eines Controllers aber niemals aus. Schon eine 8x8-Matrix braucht im Multiplexbetrieb 16 Portleitungen. Eine Zeile mit 20 Zeichenpositionen würde schon die Kapazität eines jeden Controllers sprengen. Bei solchen Anwendungsfällen behilft man sich mit Schieberegistern, die

CUE	$\overline{BL}$	$\overline{CLR}$	$\overline{CE_1}$	$\overline{CE_2}$	$\overline{WR}$	$\overline{CU}$	A_1	A_0	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Function
0	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	Display ASCII
1	1	1														Display Stored Cursor
X	X	0														Reset RAMs
X	0	1														Blank Display but do not reset RAMS and Control Register
X	X	1	0	0	0	0	0	0	Extended Functions Disable	Intensity Control			Master Blank	Digit Blank Disable 0	Digit Cursor 0	Write to Attribute RAM and Control Register
						0	0	1	0 = Enable D_1-D_5 1 = Disable D_1-D_5 D_0 Always Enabled	000 = 100%* 001 = 60% 010 = 40% 011 = 27% 100 = 17% 101 = 10% 110 = 7% 111 = 3%			0 = Display ON 1 = Display Blanked	Digit Blank Disable 1	Digit Cursor 1	DBD_n = 0, Allows Digit n to be blanked DBD_n = 1 Prevents Digit n from being blanked. DC_n = 0 Removes cursor from Digit n DC_n = 1 Stores cursor at Digit n
						0	1	0						Digit Blank Disable 2	Digit Cursor 2	
						0	1	1						Digiit Blank Disable 3	Digit Cursor 3	
X	X	1	0	0	0	1	0	0	Digit 0 ASCII Data (Right Most Character)							Write to Character RAM
						1	0	1	Digit 1 ASCII Data							
						1	1	0	Digit 2 ASCII Data							
						1	1	1	Digit 3 ASCII Data (Left Most Character)							
X	X	1	1	X	X	X	X	X	X	X	X	X	X	X	X	No Change
			X	1	X											
			X	X	1											

0 = Logic 0; 1 = Logic 1; X = Do Not Care; * 000 = 27% for HDLU-2416

Bild 1.14: Wahrheitstabelle für die HDLX2416 (Nach Unterlagen von HP)

zusätzlich ein Latch für ihre parallelen Ausgangsleitungen besitzen (z. B. 4094 oder 74HC595) und hintereinander geschaltet werden können. Egal wie viele Bits auszugeben sind, benötigt der Controller nur drei Leitungen, um die Daten auszugeben: Data (0 oder 1), Schiebetakt und Übernahme ins Latch. Wegen des höheren Strombedarfs muss oft mit Treibern gearbeitet werden. Die Schieberegister erhalten jeweils einen ULN2803 nachgeschaltet und die Zeilen werden mittels eines BC557 (PNP-Transistor) angesteuert. Sein Kollektorwiderstand richtet sich nach den verwendeten Displays (ca. 47 – 120 Ω).

Tipp

- Die Helligkeit von LED-Anzeigen lässt sich – wie schon bei den Motoren gezeigt – ausgezeichnet über PWM steuern. Das funktioniert sogar besser und bequemer als eine lineare Steuerung des LED-Stroms. Die Kennlinie der LED ist nämlich nicht linear.
- Es gibt Siebensegmentanzeige-Typen, bei denen die Anschlusspins rechts und links angeordnet sind, und solche, bei denen die Anschlüsse oben und unten liegen. Je nach Schaltung kann das Platinenlayout durch die Wahl des geeigneten Typs stark vereinfacht werden.

Es ist noch ein Thema zu behandeln, das heutzutage für LEDs große Bedeutung hat: mehrfarbige LED, die wegen der Kombination aus einem roten, einem grünen und einem blauen LED-Chip in einem Gehäuse kurz „RGB-LED“ genannt werden. Solche LEDs sind einzeln oder in Form der oben genannten LED-Strips, aber auch als Punktmatrix-Modul lieferbar. An dieser Stelle geht es nicht um die LEDs selbst, sondern um deren Ansteuerung.

Das menschliche Auge ist nicht für alle Farben gleich empfindlich, vielmehr werden gelb-grüne Töne (also die Mitte des sichtbaren Spektrums) besser wahrgenommen als rote oder blaue. Auch die Helligkeitswahrnehmung des Auges ist nicht linear, sondern eher logrithmisch. Diese Tatsache wird beispielsweise auch von anderen Systemen oder Bildbearbeitungsprogrammen berücksichtigt (etwa in der Funktion „Gammakorrektur“). Ist die LED ganz dunkel, werden kleine Änderungen gut wahrgenommen, ist die LED dagegen schon ziemlich hell, merkt man die Zu- oder Abnahme erst nach mehreren Schritten. Wie man diese Probleme löst, ist im Kapitel über Programmierung von Embedded-Systeme beschrieben.

Inzwischen gibt es auch „schlaue“ Leuchtdioden, z. B. WS2812 von Worldsemi. Es handelt sich um RGB-LEDs mit eingebautem PWM-Controller, die hintereinander geschaltet werden können. Über

eine einzige Leitung wird die Helligkeit jeder RGB-Komponente programmiert. Jede Farbe benötigt acht Bit, was 256 Helligkeitsstufen entspricht. Pro LED müssen daher 24 Bit übertragen werden, was einem Umfang von über 16 Millionen Farbmöglichkeiten entspricht. Mehrere Leuchtdioden werden einfach über die Pins DO und DI hintereinandergeschaltet (Bild 1.15).

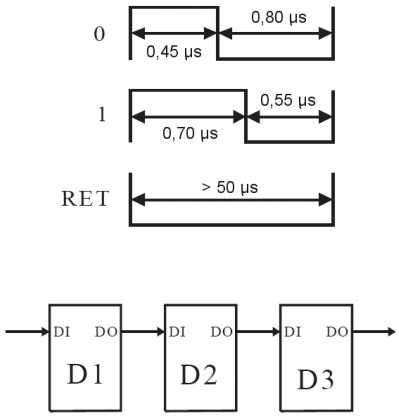

Bild 1.15: Das Ansteuersignal der WS2812-LED

So sind Ketten von bis zu 1024 LEDs möglich, die alle einzeln angesteuert werden können. Die Stromaufnahme jeder LED beträgt ca. 1 mA im ausgeschalteten Zustand. Dies wird für den PWM-Controller benötigt. Bei voller Helligkeit aller drei Farben ist die Stromaufnahme ca. 60 mA bei 5V Versorgungsspannung.

Zur Programmierung der LED dient ein serielles Datensignal. Für jedes der 24 übertragenen Bits wird ein Datenrahmen von 1,25 Mikrosekunden benötigt, was einer Datenrate von 800 KHz entspricht. Dabei werden 0- und 1-Bits durch die unterschiedliche Länge des High-Signals unterschieden. Erst wenn das Eingangssignal länger als 50 Mikrosekunden Low bleibt, übernehmen alle LEDs einer Kette die neuen Werte und stellen die Helligkeit entsprechend ein.

Die Stromversorgung des Steuerkreises der Diode wird über ein RC-Glied abgeblockt, um Störungen fernzuhalten. Die LEDs selbst werden direkt mit 5 V verbunden. Ein Vorwiderstand ist nicht nötig, da der interne Schaltkreis die Strombegrenzung vornimmt(Bild 1.16).

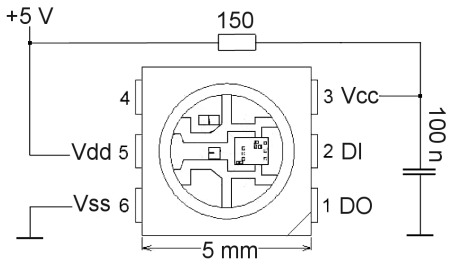

Bild 1.16: Beschaltung der WS2812-LED

1.1.3 Konstantstromquelle

Eine einfache LED-Schaltung mit einem Vorwiderstand reicht für viele Fälle aus, hat aber den Nachteil, dass der Diodenstrom stark von der Versorgungsspannung abhängt. Man muss für jede Spannung einen eigenen Widerstand bestimmen. Dagegen hilft eine Konstantstromquelle. Ihr Ausgangsstrom ist weitgehend unabhängig von der Eingangsspannung und vom Spannungsabfall am Verbraucher. Außerdem kann man (abhängig von der Versorgungsspannung und der LED-Spannung) unterschiedliche LED-Farben oder LED-Anzahl verwenden ohne die Schaltung ändern zu müssen.

Weiterhin sorgt eine Konstantstromquelle auch für einen sicheren Schutz der LEDs vor Hard- und Softwarefehlern beim Ansteuern der LEDs vom Controller aus. Für die Steuerung der LED-Helligkeit wird in der Regel Pulsweitenmodulation verwendet. Hängt sich der Controller beispielsweise durch

einen Softwarefehler auf kann es vorkommen, dass die LEDs einen konstanten Einschaltpegel erhalten (anstelle des vorgesehenen Rechtecksignals) und nach einiger Zeit den Überhitzungstod erleiden.

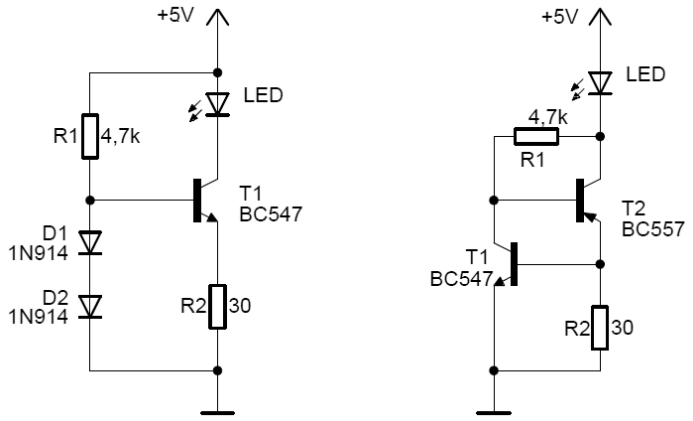

Bild 1.17: Zwei einfache, diskret aufgebaute Konstantstromquellen

Die erste Schaltung (Bild 1.17 links) verwendet zwei Si-Dioden in Durchlassrichtung zur Stabilisierung einer Hilfsspannung von ca. 1,2 V. Die Spannung am Emitterwiderstand R2, der den Strom bestimmt, liegt um ca. 0,6 V niedriger. Hier fallen daher 0,6 V ab. Beim in der Schaltung als Beispiel gewählten Wert ergibt sich nach dem Ohmschen Gesetz ($I = U/R$) zu 0,6/30 = 0,02 A = 20 mA. Statt der beiden Si-Dioden kann auch eine Z-Diode verwendet werden (z. B. 3,3 V). Es muss dann bei der Rechnung die Durchbruchspannung berücksichtigt werden. Gegebenenfalls ist auch eine höhere Versorgungsspannung erforderlich.

Bei einer Diodenstabilisierung wird immer etwas Strom „verschwendet". Die zweite Schaltung (Bild 1.17 rechts) ersetzt die beiden Dioden durch einen Transistor. Die Schaltung hat zwar eine etwas schlechtere Stabilisierung, bildet dafür aber einen reinen Zweipol, weshalb man den gesamten Strom auch durch Entfernen der LED abschalten kann. Die Schaltung verhält sich wie ein Vorwiderstand, der sich automatisch der Spannung anpasst. Diese Schaltung wird auch gerne für Kontroll-LEDs verwendet. Die Schaltungen können je nach Versorgungsspannung (Verlustleistung am Transistor und an R2 beachten!) bis ca. 100 mA liefern.

Anstelle der LEDs kann natürlich auch jeder andere Verbraucher mit einem konstanten Strom versorgt werden. Das Prinzip der linken Schaltung kann auch für Lasten verwendet werden, die gegen Masse liegen. Es werden dann nur der NPN- Transistor durch einen PNP-Typ ersetzt, R2 und die nunmehr in der anderen Richtung gepoltren Dioden (wieder in Durchlassrichtung) liegen an +5 V und die LED (sowie R1) an Masse. Bei der rechten Schaltung kann die LED einfach „nach unten" versetzt werden.

Für höhere Ströme bis ca. 1,2 A kann man einen integrierten Spannungsregler als Konstantstromquelle schalten. Das ist unter *http://www.netzmafia.de/skripten/hardware/lm317/lm317.html* (LM317-Spannungsregler/Stromquelle berechnen) zu finden. Bei höheren Strömen muss man aber auch die Verlustleistung am Regler und am strombestimmenden Widerstand berücksichtigen.

Schön wäre es natürlich, wenn man den Strom nicht durch einen Festwiderstand festgelegt werden müsste, sondern einstellbar wäre. Man könnte natürlich in den gezeigten Schaltungen R2 durch einen Festwiderstand in Reihe mit einem Trimmpoti ersetzen, aber das wäre nur ein erster Schritt – eine Steuerung von Controllerseite muss anders erfolgen.

Bild 1.18 zeigt eine Lösungsmöglichkeit. Die Einstellung des Stroms durch den Lastwiderstand RL (z. B. eine Power-LED) erfolgt mit Hilfe der Referenzspannung Uref am nicht-invertierenden Eingang des Operationsverstärkers und des Widerstands R3. Auch hier gilt, wie schon zuvor, $I_{Last} = Uref/R3$. Beim Einschalten der Schaltung ist der N-Kanal-MOSFET gesperrt, weshalb zwischen MOSFET und R3 Massepotential anliegt, das sich über den Widerstand R2 auf den invertierenden Eingang des Operationsverstärkers (OPV) überträgt. Da mit Uref am nicht-invertierenden Eingang eine deutlich höhere Spannung anliegt, steuert der Ausgang des OPV zunächst voll durch, weshalb der Power-MOSFET öffnet und durch die Last ein Strom fließen kann.

Durch den Strom bildet sich über R3 ein Spannungsabfall, der die Spannung am invertierenden Eingang des OPV anhebt und dazu führt, dass die Spannungsdifferenz zwischen den beiden Eingängen

des OPV abnimmt, was seinerseits zu einer Verringerung der Ausgangsspannung des Operationsverstärkers führt und den MOSFET weniger öffnet. Der daraus resultierende geringere Strom durch die Last führt wiederum zu einem geringeren Spannungsabfall am Widerstand R3 und somit zu einer geringeren Spannung am invertierenden Eingang des OPV. Der OPV regelt den Stromfluss so, dass am invertierenden Eingang die gleiche Spannung liegt wie am nicht-invertierenden Eingang (Uref). OPV und Power-MOSFET arbeiten im linearen Betrieb. Als Folge daraus liegt am Widerstand R3 ebenfalls Uref an, also gilt tatsächlich $I_{Last} = Uref/R3$. R1, R2 und C1 dienen der Filterung der zurückgekoppelten Spannung und verhindern ein Schwingverhalten der Regelschleife.

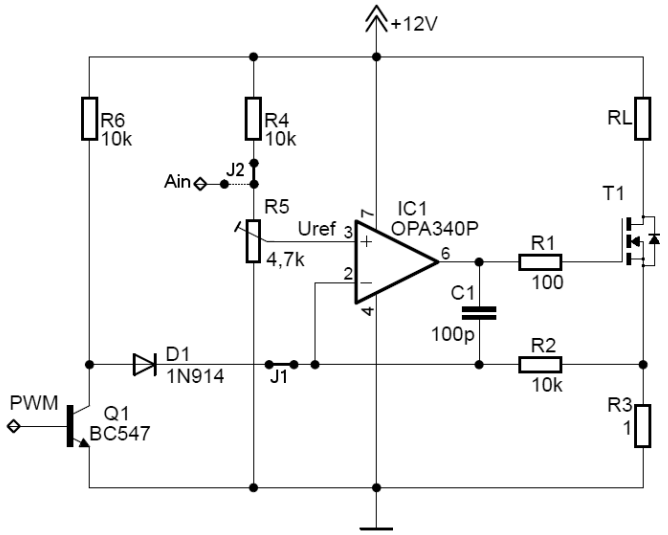

Bild 1.18: Steuerbare Konstantstromquellen

Soll die Konstantstromquelle ein- und ausschaltbar sein, kann die Schaltung einfach durch eine Diode (D1) am invertierenden Eingang des OPV erweitert werden (J1 geschlossen). Legt man die Anode der Diode auf Massepotential, verhält sich die Schaltung wie oben beschrieben, da die Diode sperrt und somit keinen Einfluss hat. Legt man hingegen eine Spannung deutlich oberhalb von Uref an die Anode, fließt über die Diode und R2 und R3 ein Strom. Am invertierenden Eingang des OPV liegt die angelegte Spannung (abzüglich der Diodenspannung). Da diese Spannung größer als Uref ist, geht der Ausgang des OPV auf Massepotential und der Power-MOSFET sperrt. Die Diode wird über eine NPN-Transistorstufe angesteuert, die zum Pegel eines Controllerausgangs kompatibel ist. Wird das Feature nicht benötigt, muss man Jumper J1 öffnen.

Die Schaltung bietet aber noch ein weitere Feature. Alternativ kann das Trimmpoti R5 statt von Vcc mit einer externen Spannung beaufschlagt werden (Jumper J2 umstecken – Verbindung Ain und R5). Diese Spannung kann beispielsweise von einem Analogausgang des Controllers stammen. Nun kann die Referenzspannung Uref von außen vorgegeben und so auch der Strom durch die Last gesteuert werden. Hier könnte die Steuerung auch über einen DMX-Ausgang (0 . . . 10 V) erfolgen. [4]

Wem der diskrete Aufbau zu kompliziert ist, kann zu einem der zahlreichen Buck/Boost-Wandlerchips greifen, die von vielen Halbleiterherstellern angeboten werden. Stellvertretend für alle sei hier ein Baustein vorgestellt.

Der CAT4101 ist eine Konstantstromsenke, die eine Reihe von High-Brightness-LEDs mit bis zu 1 A treiben kann, wobei unter Vollast nur ein sehr geringer Spannungsabfall von 0,5 V auftritt. Der Baustein benötigt keine externe Induktivität, sondern lediglich einen Widerstand und einen Kondensator. Der LED-Strom wird durch einen externen Widerstand am RSET-Pin eingestellt. Der Ausgang kann LED-Reihen mit einen Versorgungsspannung von bis zu 25 V ansteuern. Der Chip gewährleistet ein genau geregelten Strom durch die LEDs, unabhängig von von Versorgungsspannung und LED-Durchlassspannung.

Über den PWM/EN-Eingang kann das Device abgeschaltet oder die LED- Helligkeit mit Hilfe eines externen PWM-Signals gesteuert werden. Der LED-Driver verfügt über einen Übertemperaturschutz, der bei einer Temperatur von mehr als 150° C aktiviert wird. Die interne Logik wird mit 5 V gespeist, der LED-Strang kann mit bis zu 25 V versorgt werden. Gegebenenfalls muss für die Logik ein kleiner

[4] Schlösse man den Eingang Ain an den Ausgang eines Verstärkers an, hätte man eine moderne Variante der Lichtorgel aus den 1980er Jahren.

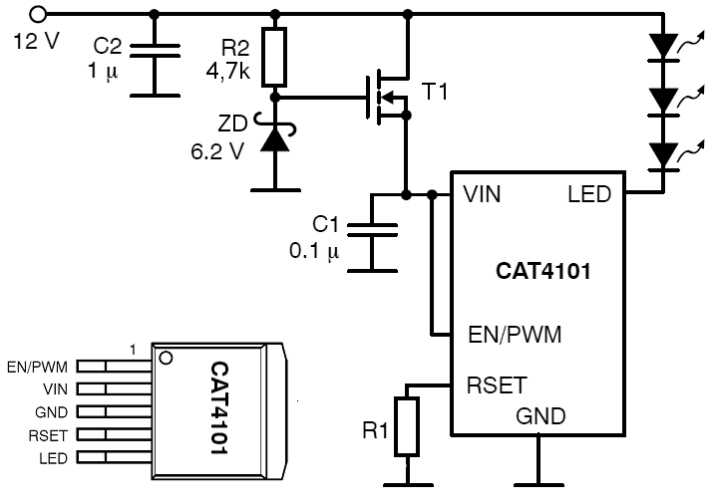

Bild 1.19: Konstantstromsenke CAT4101

Spannungsregler eingesetzt werden. Bild 1.19 zeigt die Beschaltung inklusive Generierung der 5-V-Logikspannung und das Bauteilelayout. Der Strom wird durch R1 festgelegt, dessen Wert sich aus der Faustformel $R1 = 500/I$ ermitteln läßt. Eine Tabelle mit genauen Werten ist dem Datenblatt zu entnehmen.

Genaues lineares Dimmen funktioniert mit PWM-Frequenzen von 100 Hz bis 5 kHz für ein PWM-Tastverhältnis bis zu 1 Prozent. PWM-Frequenzen bis zu 50 kHz werden für ein Tastverhältnis von mehr als 10 Prozent unterstützt. Bei der Kombination von niedrigen PWM-Frequenzen und kleinem Tastverhältnis kann der Chip versehentlich in den Shutdown-Modus wechseln. Dies hat keine Auswirkung auf die Genauigkeit beim Dimmen, weil die Einschaltzeit TPS im Bereich von eine Mikrosekunde liegt. Um das zu vermeiden, sollte die Low-Pulsbreite länger als eine Mikrosekunde sein. Der CAT4101 wechselt in einen „Null Strom Shutdown-Modus“ 5 ms nachdem der Pin EN/PWM auf Low gegangen ist.

1.2 Displays

Üblicherweise werden LC-Displays ja über eigene Mikrocontroller betrieben. Ein PC erledigt diese Aufgabe quasi nebenbei, erfordert sie doch kaum Rechenleistung. Lediglich die Umleitung der gewünschten Prozessanzeigen auf einen Port des Rechners und die Anpassung auf das Datenformat der Anzeige sind notwendig.

Eine Menge an Informationen, Treiber etc. zum Thema „LCD“ finden Sie auf der Webseite *http://ssl.bulix.org/projects/lcd4linux/*. Eigentlich könnte ich hier noch einen Schaltplan platzieren und mit dem nächsten Kapitel weitermachen. Ich finde jedoch, dass die kleinen und doch recht intelligenten Displays es verdient haben, dass man sich ihnen etwas ausführlicher widmet, und Sie sollen auch sehen, wie man sie direkt ansteuert und dass dies, zumindest von der Hardware-Seite, nicht sehr aufwändig ist.

1.2.1 LCD- und OLED-Grundlagen

In diesem Abschnitt geht es um Flüssigkristallanzeigen (LCD Liquid Crystal Display), die in neuerer Zeit von Anzeigen mit anderer Technik abgelöst werden. Bei der neuesten Entwicklung handelt es sich um organische Leuchtdioden, kurz OLED (Organic Light Emitting Diode), deren Lebensdauer aber noch kürzer ist als die der LCDs. Beide Typen werden auf die gleiche Weise angesteuert.

Eine LCD-Anzeige besteht im Prinzip aus zwei Glasscheiben und einer speziellen Flüssigkeit, den „nematischen Phasen“ oder „Flüssigkristallen“ dazwischen. Die Flüssigkeit ist in der Lage, die Polarisationsebene von Licht zu drehen. Dieser Effekt wird durch Anlegen eines elektrischen Feldes beeinflusst. Um ein Feld erzeugen zu können, bedampft man die Platten mit einer hauchdünnen Metallschicht, die außerdem das Anzeigemuster enthält (Siebensegmentanzeige, Punktmatrix, Symbole

etc.). Man versieht anschließend die beiden Glasplatten jeweils mit einer hauchdünnen Polarisationsfolie. Die beiden Folien sind um 90° gegeneinander gedreht. Die obere Folien nennt man „Polarisator", die untere „Analysator" (Bild 1.20). Ohne Flüssigkeit zwischen den Platten könnte das Licht nicht passieren. Die Flüssigkeit dreht ohne angelegtes elektrisches Feld die Polarisationsebene des einfallenden Lichtes um 90°, so dass dieses ungehindert den Analysator passieren kann – das LCD ist durchsichtig. Legt man nun eine Spannung an die aufgedampfte Metallschicht, drehen sich die Kristalle in der Flüssigkeit. Dadurch wird die Polarisationsebene des Lichtes um beispielsweise weitere 90° gedreht. Der Analysator versperrt nun dem Licht den Weg durch das LCD – das LCD ist undurchsichtig.

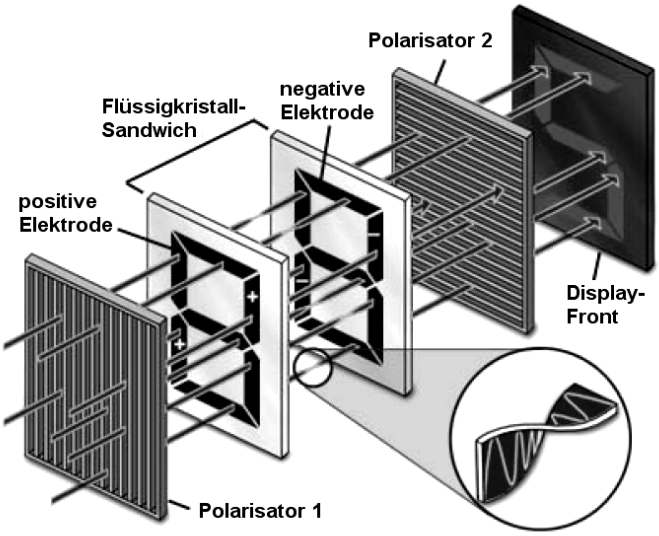

Bild 1.20: Schema einer LCD-Anzeige

Diese ersten, als TN (Twisted-Nematic) bezeichneten LCDs drehten die Polarisationsebene des Lichtes um 90°. Dann kamen STN (Super-Twisted-Nematic); sie drehen die Polarisationsebene des Lichtes um mindestens 180°. Dadurch erreicht man einen besseren Kontrast. Allerdings gab es parallel eine leichte Färbung des Displays (gelb-grün oder grau-blau, je nach Typ). Um diesen Farbeffekt zu kompensieren, verwendet man in der FSTN-Technik eine Filterfolie auf der Außenseite des Displays, was aber nur für beleuchtete Displays sinnvoll ist. Sobald aber das Display beleuchtet wird, spielt die jeweilige Displayfarbe keine Rolle mehr – es dominiert die Farbe des Lichts.

Kleine Displays mit geringem Anzeigenumfang (Voltmeter, Uhren etc.) werden meist statisch angesteuert. Werden die Displays allerdings komplexer, sind für den statischen Betrieb immer mehr Leitungen nötig, und man muss zum Multiplexbetrieb übergehen. Das Display wird in Zeilen und Spalten und Zeile für Zeile nacheinander aktiviert. Durch die Trägheit der Flüssigkeit ergibt sich ein (scheinbar) stehendes Bild. Durch das Multiplexen leiden Kontrast und Helligkeit (sinnvoll nur bei STN).

Jedes LC-Display besitzt eine sog. Vorzugsblickrichtung, von der aus betrachtet das Display einen optimalen Kontrast bietet. Je weiter der Betrachter von dieser Mittelachse abweicht, desto schwächer wird der Kontrast.

Unbeleuchtete Displays besitzen auf der Rückseite einen Reflektor. Transflektive Displays haben auf der Rückseite einen teildurchlässigen Reflektor, der es erlaubt, das Display mit oder ohne rückseitige Beleuchtung abzulesen. Transmissive Displays besitzen gar keinen Reflektor und sind nur mit Hintergrundbeleuchtung ablesbar. Die meisten Displays werden im so genannten Positivmodus produziert. Sie bieten dunkle Zeichen auf hellem Hintergrund. Negativdisplays haben einen dunklen Hintergrund und leuchtende Zeichen (nur mit Beleuchtung sinnvoll anwendbar). Die Beleuchtung erfolgt mit LEDs oder Elektroluminiszenz-Folien. Letztere sind zwar sehr sparsam im Stromverbrauch, benötigen aber einen Inverter zum Erzeugen der notwendigen Hochspannung.

Standard-LCDs haben einen Temperaturbereich von 0 bis +50 °C. Es gibt aber auch solche, die für den Betrieb von −20 °C bis +70 °C ausgelegt sind. Da die Kontrasteinstellung der LCDs temperaturabhängig ist, brauchen solche Displays manchmal eine negative Spannung für die Kontrasteinstellung. Sie sollten auch bei ausgeschaltetem Display auf die maximale Umgebungstemperatur achten.

In keinem Fall darf jedoch die Lagertemperatur eines Displays überschritten werden. Direkte Sonneneinstrahlung kann da mörderisch wirken: Mit zunehmender Temperatur wird es dunkler, was zu noch höherer Wärmeabsorption führt, und schließlich „verdampft“ die im Display enthaltene Flüssigkeit – Exitus!

Die ersten LCDs waren Siebensegmentanzeigen, so wie sie noch heute in einfachen Taschenrechnern und Uhren zu finden sind. Später kamen Textdisplays, welche die ASCII-Zeichen mit einer Punktmatrix aus 5x7 Punkten darstellten. Bis zum Grafikdisplay war es dann nur noch ein kleiner Schritt. Inzwischen finden sich alle Formen und Farben im Angebot der Hersteller. Auch besitzen die meisten Displays eine eingebaute Intelligenz. Das Display selbst ist auf einer Platine mit Controller, Zeichengenerator, ggf. Displayspeicher und Treibern für die Ansteuerung montiert. Die darzustellenden Daten werden über eine parallele oder eine serielle Schnittstelle übertragen. Der wohl bekannteste Controller ist der der HD44780 für Dotmatrixdisplays; er benötigt nur die ASCII-Information und übernimmt dann Zeichendarstellung, Speichern, Cursorpositionierung und Multiplexen. Für Grafikdisplays sind die Typen HD61xxx, SED1520, SED1330 oder T6963 weit verbreitet (wobei hier die Ansteuerung oft noch in Gefrickel ausartet). Weiter unten werde ich auf intelligentere Vertreter dieser Spezies eingehen.

Inzwischen kommen auch andere Technologien zum Einsatz. Recht neu sind die OLED- und PLED-Displays. Eine OLED ist ein in Dünnfilmtechnik hergestelltes, leuchtendes Bauelement aus organischen Halbleitern, dessen Aufbau dem einer normalen Leuchtdiode (LED) ähnelt. Die OLED-Technik ist vorrangig für die Bildschirmanwendung gedacht. Ein weiteres Einsatzgebiet stellt die großflächige Raumbeleuchtung dar. Für aus Polymeren gefertigte organische LEDs hat sich die Abkürzung PLED durchgesetzt. Der Herstellungsprozess eines OLED-Displays unterscheidet sich grundlegend von dem eines LCD. OLEDs können auf fast jedes Material gedruckt werden. Durch die Verwendung von biegsamen Trägermaterialien eröffnen sie die Möglichkeit, aufrollbare Bildschirme herzustellen und Displays in Kleidungsstücke zu integrieren. Ein weiterer Vorteil der OLED-Bildschirme gegenüber den herkömmlichen Flüssigkristallbildschirmen ist, dass sie ohne Hintergrundbeleuchtung auskommen: Während LCDs nur als farbige Filter wirken, emittieren OLEDs farbiges Licht. Außerdem haben OLED-Anzeigen einen großen Blickwinkelbereich (bis zu 170°) und eine hohe Schaltgeschwindigkeit. Das größte technische Problem stellt die vergleichbar geringe Lebensdauer dar.

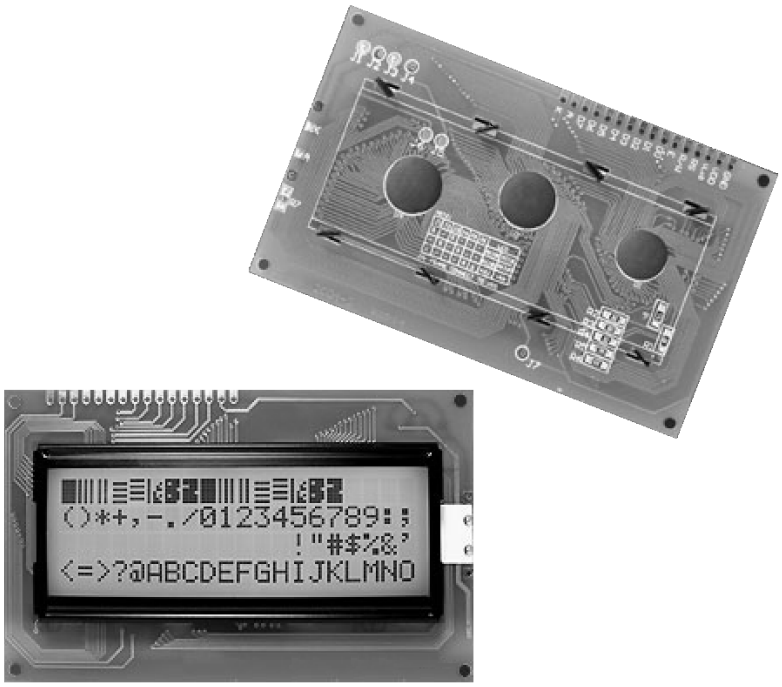

Bild 1.21: LC-Textdisplay mit HD44780

Im Folgenden soll die Anwendung und Ansteuerung der bekanntesten Text-LCDs mit HD44780-kompatiblen Controllern erläutert werden. Der Betrieb ist am Parallelport problemlos möglich, da alle Displays TTL-kompatible Anschlüsse besitzen. Über einen Seriell-Parallel-Wandler wäre auch

ein Anschluss an die serielle Schnittstelle möglich. Bild 1.21 zeigt die Vorder- und Rückansicht solcher Displays, wobei es Typen mit eins bis vier Zeilen à 8 bis 40 Zeichen gibt.

1.2.2 Displaytypen

Je nach Displaytyp kann es vorkommen, dass der Aufbau des LCD-Moduls in Spalten und Zeilen nicht unbedingt mit seiner internen Organisation übereinstimmt. Unproblematisch sind alle 2-zeiligen Displays (8 bis 40 Zeichen pro Zeile) und alle 4-zeiligen mit einer Zeilenlänge über 20 Zeichen, da diese Typen den/die benötigten zusätzlichen Displaytreiber HD44100 enthalten.

Kritisch sind hingegen alle 1-zeiligen Displays mit 8 bis 40 Zeichen/Zeile sowie 4-zeilige Displays mit bis zu 20 Zeichen/Zeile. So entspricht ein zweizeiliges Display mit jeweils 8 Zeichen pro Zeile genau der internen Struktur eines HD44780-Chips, wenn er sich im 5x8-Punkte-Modus befindet. Um Kosten zu sparen, werden oft einfach die beiden Zeilen eines 2x8-Displays mechanisch hintereinander angeordnet (8+8-Modul). Der Controller weiß davon nichts und behandelt die vordere und die hintere Displayhälfte wie zwei getrennte Zeilen; er muss also 2-zeilig initialisiert werden. Wie viele Displaytreiber bei welcher Zeilenlänge benötigt werden, hängt vom verwendeten Chip-Typ ab.

Die Problematik zeigt sich spätestens beim Ansteuern per Programm. So gibt es 4x40-Displays, bei denen die ersten 20 Zeichen in die erste Zeile geschrieben werden, während die Zeichen 21 bis 40 in der dritten Zeile landen. Genauso gehören die zweite und vierte Zeile hintereinander. Vierzeilige Displays mit bis mehr als 20 Zeichen pro Zeile lassen sich nicht mehr mit einem Controller realisieren. Der interne Textpuffer des HD44780 ist zu klein. Deshalb sind diese Displays als zwei unabhängige zweizeilige Displays aufgebaut. Der erste Controller verwaltet die beiden oberen Zeilen und der zweite Controller die unteren. Beide Controller sind parallel am Interfacestecker angeschlossen, wobei jeder Controller seinen eigenen ENABLE-Pin besitzt.

1.2.3 Zeichenadressierung

Der Controller (und damit die LCD-Anzeige) stellt in der Anzeige die Zeichen dar, die sich im internen Textpuffer befinden. Dieser 80 Zeichen lange Puffer wird als „DDRAM“ (Display Data RAM) bezeichnet.

Tabelle 1.3: Adresszuordnung der Display-Anzeigepositionen im Textpuffer (hex)

Typ	1. Zeile	2. Zeile	3. Zeile	4. Zeile	Bemerkung
1x8	00 – 07	–	–	–	
1x16	00 – 0F	–	–	–	einzeiliges LCD
1x16 (8+8)	00 – 07 40 – 47	–	–	–	linke Hälfte rechte Hälfte
1x20	00 – 13	–	–	–	
1x40	00 – 27	–	–	–	
2x8	00 – 07	40 – 47	–	–	
2x12	00 – 0B	40 – 4B	–	–	
2x16	00 – 0F	40 – 4F	–	–	
2x20	00 – 13	40 – 53	–	–	
2x24	00 – 17	40 – 57	–	–	
2x40	00 – 27	40 – 67	–	–	
4x16	00 – 0F	40 – 4F	10 – 1F	50 – 5F	
4x20	00 – 13	40 – 53	14 – 27	54 – 67	
4x40	00 – 27 –	40 – 67 –	– 00 – 27	– 40 – 67	1. Controller 2. Controller

Wurde der Controller einzeilig initialisiert, besitzt das DDRAM einen geschlossenen Speicherbereich mit den Adressen 00h bis 4Fh. Bei zweizeiliger Initialisierung besteht der DDRAM-Bereich aus zwei getrennten Bereichen zu je 40 Zeichen: für die erste Zeile von 00h bis 27h und für die zweite Zeile von 40h bis 67h. Wir merken uns also, dass die zweite Zeile in der Regel bei Adresse 40h beginnt.

Das Display zeigt also immer den Inhalt des internen, 80 Zeichen langen Textpuffers an. Ist das Display kleiner als der Textpuffer, wird nur ein Ausschnitt des Puffers angezeigt. Dieser Ausschnitt ist im Textpuffer verschiebbar (per Programmierung ist einstellbar, ob der Cursor nach rechts oder der Text nach links wandert). Im Folgenden wird nur die Zuordnung von Displaypositionen zu Adressen des Textpuffers ohne Berücksichtigung einer möglichen Verschiebung betrachtet. Tabelle 1.3 zeigt die Zuordnung für die verschiedenen Typen.

1.2.4 Anschluss des LC-Displays

Die nachfolgenden Anschlussbelegungen gelten als Orientierung und treffen auf die meisten Displays zu. Es wird aber immer auch Ausnahmen geben. Der Blick in das Datenblatt des Displays bleibt Ihnen also nicht erspart. Um Anschlussfehler zu vermeiden, sollte man zumindest prüfen, ob der vermeintliche Massepin mit den Masseflächen des Displays verbunden ist. Dann kann man davon ausgehen, dass auch die anderen Pins mit der vermuteten Anschlussbelegung übereinstimmen.

Tabelle 1.4: Anschlussbelegung der LC-Displays

Pin	Symbol	Pegel	Beschreibung
1	Vss	GND	Masse
2	Vdd	+5 V	Versorgungsspannung +5 V
3	Vo	0 ... 1,5 V −2 V ... −5 V	Kontrast Kontrast bei HT-Displays
4	RS	0/1	Register Select
5	R/W	0/1	1: Read, 0: Write
6	E	0/1	1: Enable, 0: Disable
7	D0	0/1	Datenleitung 0 (LSB)
8	D1	0/1	Datenleitung 1
9	D2	0/1	Datenleitung 2
10	D3	0/1	Datenleitung 3
11	D4	0/1	Datenleitung 4
12	D5	0/1	Datenleitung 5
13	D6	0/1	Datenleitung 6
14	D7	0/1	Datenleitung 7 (MSB)
15	LED+	??	LED-Beleuchtung: Pluspol
16	LED-	GND	LED-Beleuchtung: GND

Die LCD-Module sind unterschiedlich kontaktiert. Je nach Modell und Hersteller gibt es Module mit einreihigen oder zweireihigen Kontaktleisten (siehe Bild 1.22) mit 14 oder 16 Polen. Die Pole 15 und 16 sind in der Regel für die Hintergrundbeleuchtung vorgesehen. Hat ein beleuchtetes Display nur 14 Kontakte am Verbinder, so sind die Beleuchtungsanschlüsse an einer anderen Stelle des Displays zu finden. Die Pins 1 bis 14 sind in aller Regel identisch belegt. Die Belegung ist in Tabelle 1.4 aufgelistet.

Vierzeilige Displays mit mehr als 20 Zeichen/Zeile besitzen zwei unabhängige Display-Controller. Einen für die ersten beiden Zeilen und einen für die unteren beiden. Hier gibt es einen zusätzlichen Enable-Pin, der meist nach dem Pin 6 (Enable) eingeschoben wird – das LCD hat dann 17 Pins. Auch bei diesen ist ein Blick ins Datenblatt unerlässlich.

Die Displays benötigen eine Betriebsspannung Vdd von +5 V (± 5%). Der Vss-Pin liegt auf Masse. Die Stromaufnahme ohne Beleuchtung liegt bei 1–5 mA. Außerdem wird eine Spannung zur Einstellung des Displaykontrastes benötigt. Es gibt zwei unterschiedliche Typen von Displays, die auch unterschiedliche Kontrast-Spannungen benötigen:

- So genannte Standard-Displays mit einen Temperaturbereich zwischen 0 und 50 °C. Diese benötigen eine Kontrastspannung zwischen 0 und 1,5 V.
- Großdisplays und Displays für hohe Umgebungstemperaturen (Temperaturbereich: −20 bis 70 °C, „HT-Displays“) benötigen oft eine Spannung zwischen −2 und −5 V, was bei der Ansteuerschaltung zu berücksichtigen ist. Durch Änderung der Kontrast-Spannung lässt sich der Kontrast

passend zum Blickwinkel verändern. Der Kontrast ist leider auch temperaturabhängig. Wer das Display unter verschiedenen Temperaturen betreiben muss, sollte für den Einstellwiderstand ein Potentiometer vorsehen.

Bevor Sie das Verbindungskabel herstellen, sollten Sie als Erstes feststellen, welchen Steckertyp Sie benötigen. Für den zweireihigen Steckverbinder (2x8) wird ein Flachbandkabel mit einem Aderabstand von 1,27 mm verwendet. Für die einreihigen Steckverbinder (1x16) kommt ein Flachbandkabel mit einem Aderabstand von 2,54 mm zum Einsatz. Die Anfertigung der Kabel ist im Prinzip bei beiden Versionen gleich.

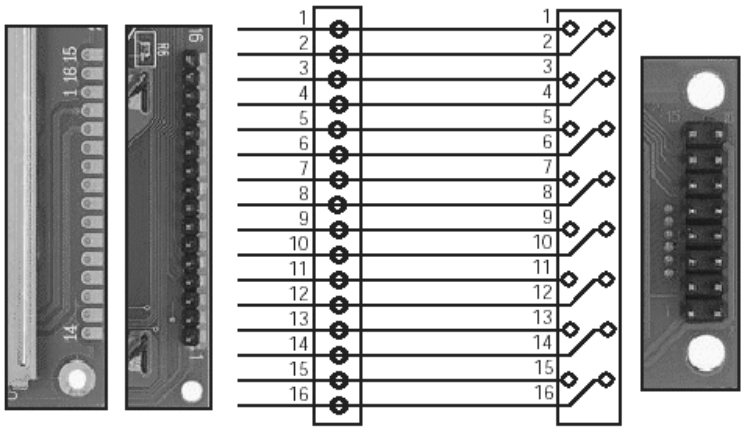

Bild 1.22: Die verschiedenen Anschlussleisten der LCDs

Beide Seiten des Flachbandkabels sind mit einem Pfostensteckverbinder zu versehen, auf dessen Messerkontakte das Flachbandkabel aufgequetscht wird. Wer keine spezielle Quetschzange besitzt, kann das Aufpressen mit Hilfe eines Schraubstocks vornehmen, da hier die Kraftübertragung gleichmäßig über die gesamte Kabelbreite erfolgt. Das Flachbandkabel legen Sie gerade in den Pfostenverbinder und quetschen anschließend beide Hälften des Pfostenverbinders langsam und vorsichtig mit dem Schraubstock zusammen. Die überstehenden Kabelenden schneiden Sie mit einem scharfen Messer direkt am Stecker ab. Das Flachbandkabel ist an einer Seite (eine Ader) farblich markiert. Hierdurch wird Pin 1 gekennzeichnet, so dass ein versehentliches Verdrehen des Steckers vermieden werden kann. Im nächsten Arbeitsschritt ist die Kontaktleiste des LCD-Moduls mit einer Stiftleiste zu versehen. Bei der zweireihigen Stiftleiste ist es besonders wichtig, dass diese Stiftleiste von hinten auf die Platine gelötet wird (Bild 1.22 zeigt die LCDs von hinten, weshalb auch die Nummerierung in der Gegenrichtung zur Schaltzeichnung verläuft). Jetzt kommt der wohl wichtigste Arbeitsschritt, der Anschluss des LCD-Moduls an die Steuerplatine.

Hinweis

Wie man in Bild 1.22 erkennt, kann die Pinfolge je nach Modell unterschiedlich sein. Normalerweise liegen die Pins entsprechend ihrer Nummerierung hintereinander (1, 2, 3 ... 16). Es gibt aber auch eine Pinfolge, bei der sich die beiden Pins 15 und 16 (Hintergrundbeleuchtung) neben dem Pin 1 befinden. Die Abfolge ist dann 16, 15, 1, 2, 3 ... 14. Und es gibt auch Hersteller, die hier die Pins 15 und 16 miteinander vertauschen.

Das Anschlusskabel endet in einer Interfaceplatine für die parallele Druckerschnittstelle des PC (Bild 1.23). Die acht Datenleitungen des LCDs werden auf die Anschlüsse des Datenregisters der Schnittstelle geführt. Die Steuerleitungen enden am zugehörigen Control-Port, wobei folgende Zuordnung gilt (die auch von den meisten LCD-Programmen oder -Treibern verwendet wird):

- Bit 0: E (Enable)
- Bit 1: R/W (Read/Write)
- Bit 2: RS (Register Select)
- Bit 3: Hintergrundbeleuchtung ein/aus

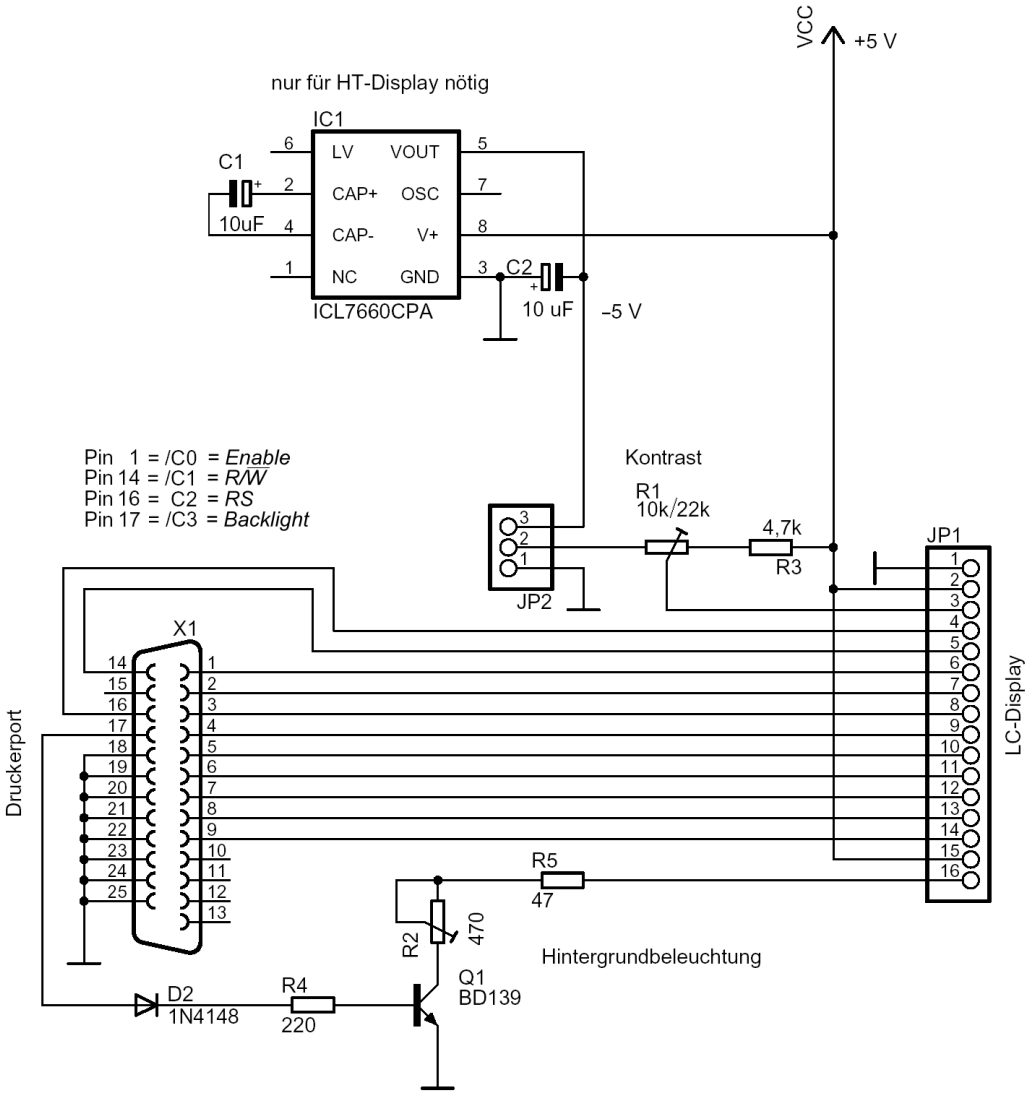

Bild 1.23: LCD-Interface für die Druckerschnittstelle

Zur Versorgung eines Displays mit erweitertem Temperaturbereich oder eines Großdisplays benötigt man eine negative Spannung, die aber so gut wie nicht belastet wird (einige Mikroampere). Dafür ist in der Schaltung die Erzeugung einer negativen Spannung mit dem Ladungspumpen-IC ICL 7660 vorgesehen. Der Baustein benötigt lediglich zwei zusätzliche Tantal-Elkos. Die Wahl der Spannung (-5 V/Masse) erfolgt über einen Jumper. Wer will, kann auch noch einen 5-V-Spannungsregler für die Versorgung des LCDs vorsehen. Mit dem Trimmpoti R2 kann die Helligkeit der Hintergrundbeleuchtung eingestellt werden. Hier kann alternativ für R2 und R5 auch ein passender Festwiderstand vorgesehen werden.

Manche Displays besitzen schon einen internen Vorwiderstand für die LEDs der Hintergrundbeleuchtung, der es erlaubt, an Pin 15 direkt 5 V anzulegen (R5 kann dann entfallen). Wenn Sie den Displaytyp nicht genau kennen, sollten Sie sicherheitshalber davon ausgehen, dass kein interner Vorwiderstand existiert. R5 ist dann unbedingt vorzusehen. Der Betrieb der LED-Beleuchtung ohne oder mit zu kleinem Vorwiderstand führt zur Zerstörung der LED-Beleuchtung. LED-Strom bzw. LED-Spannung lässt sich dem Datenblatt entnehmen. Die meisten weiß beleuchteten Displays haben eine LED-Spannung von 4,2 V. Blaue oder grüne Displays begnügen sich mit 3,3 V, und gelbe LCDs benötigen nur 1,8 V bei einem Strom von 20 bis 100 mA (manchmal auch mehr).

1.2.5 Ansteuerung von LCDs

Der Controller (und damit die LCD-Anzeige) besitzt ein paralleles Interface, bestehend aus:

- Der **8-Bit-Datenbus** dient der eigentlichen Übertragung der ASCII- oder Steuerinformation zum Display und kann nicht nur als 8-Bit-Bus, sondern auch als 4-Bit-Bus geschaltet werden. In diesem Fall werden die Daten in Form zweier „Nibbles“ hintereinander gesendet.
- Die **Enable-Leitung** schaltet das Interface ein (E = 1) oder aus (E = 0). Nur wenn Enable auf 1-Pegel liegt, lässt sich das Display ansprechen. Dies erlaubt die parallele Nutzung der anderen Leitungen (z. B. für das Auslesen einer parallel geschalteten Tastatur oder das Schalten von Relais).

  Mit einer steigenden Flanke an dieser Leitung liest das Display die Werte von RS und R/W ein. Liegt zu diesem Zeitpunkt R/W auf 0-Pegel, liest das Display mit der darauf folgenden fallenden Flanke den am Datenbus anliegenden Wert (Schreibzyklus). War jedoch R/W = 1, legt das Display ein Datenwort auf den Datenbus (Lese-Zyklus), bis die fallende Flanke das Interface wieder deaktiviert.
- Die **RS-Leitung** (RS = Register-Select) legt fest, ob die übertragenen Daten als ASCII-Zeichen in den Textpuffer gelangen (RS = 1) oder als Befehl in ein Steuerregister geschrieben werden (RS = 0).
- Die **R/W-Leitung** (R/W = Read/Write) legt fest, ob Daten zum Display geschrieben (R/W = 0) oder vom Display gelesen werden (R/W = 1). Wird auf die Lesefunktion verzichtet, muss diese Leitung fest mit Masse verbunden werden.

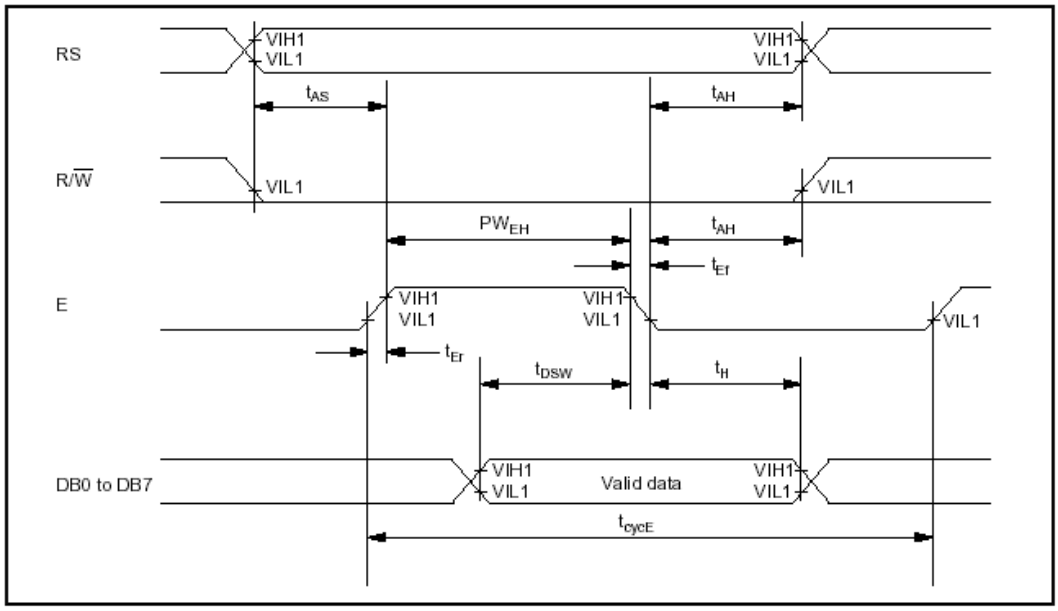

Bild 1.24: Schreibzyklus eines LCDs

Um eine sichere Verarbeitung einzuhalten, sind Mindestzeitabstände zwischen dem Setzen von RS bzw. R/W und der anschließenden Aktivierung der Enable-Leitung einzuhalten. Ebenso müssen alle anderen Leitungen während der fallenden Flanke von Enable stabil bleiben; RS und R/W dürfen den Pegel nur wechseln, wenn Enable auf 0-Pegel liegt. Bild 1.24 zeigt den zeitlichen Ablauf beim Schreiben in den LCD-Speicher, Tabelle 1.5 das Timing.

Tabelle 1.5: Timing beim Schreib- und Lesezugriff

Name	Beschreibung	Dauer
t_{cyce}	Enable cycle time	1 µs
PW_{eh}	Enable pulse width	0,45 µs
t_{as}	Address setup time	0,14 µs
t_{ah}	Address hold time	0,02 µs

Ein übergebenes Datenwort muss vom Display abgespeichert bzw. ein Kommando (z. B. Löschen der Anzeige) ausgeführt werden. Der Display-Prozessor benötigt daher interne Verarbeitungszeiten, die bei der Ansteuerung zu beachten sind. Während dieser Zeit kann das LCD keine neuen Daten

oder Kommandos entgegennehmen. Bevor man also einen neuen Schreibzugriff tätigt, sollte man testen, ob das Display mit dem letzten Befehl schon fertig ist. Die Bereitschaft des Display-Prozessors kann mittels eines Lesezugriffs ermittelt werden. Liegt RS beim Lesezugriff auf 0-Pegel, wird das Statusregister ausgegeben, andernfalls ein Zeichen aus dem Textpuffer (DDRAM). Beim Lesen des Steuerregisters enthält das höchstwertige Bit das so genannte Busy-Flag. Ist dieses Bit = 1, ist der Controller noch beschäftigt und kann keine weiteren Operationen ausführen. Man fragt somit das Busy-Flag so lange ab, bis es wieder 0 ist – oder wartet einfach genügend lange ab.

Es gibt nur ein knappes Dutzend Befehle, die nun einzeln erläutert werden:

- Display löschen

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Zeit
0	0	0	0	0	0	0	0	0	1	2,25 ms

  Das gesamte Display (also eigentlich der ganze Textpuffer) wird gelöscht, und der Cursor an den Anfang des Textpuffers gesetzt.

- Cursor Home

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Zeit
0	0	0	0	0	0	0	0	1	*	2,25 ms

  Der Cursor wird an den Anfang des Textpuffers gesetzt. Falls der vom Display angezeigte Abschnitt des Textpuffers nicht am Textpufferanfang lag, wird das Display wieder dorthin geschoben.

- Entry Mode Set

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Zeit
0	0	0	0	0	0	0	1	I/D	S	0,08 ms

  Mit dem I/D-Bit (Increment/Decrement) stellen Sie ein, ob der Cursor nach dem Schreiben eines Zeichens in den Textpuffer nach rechts (Increment) oder links (Decrement) zur nächsten Position geht (I/D = 0: links, ID = 1: rechts).

  Mit dem S-Bit lässt sich die Shift-Funktion des Displays ein- (S = 1) und ausschalten (S = 0). Ist diese Funktion ausgeschaltet, wandert der Cursor nach dem Schreiben eines Zeichens zur nächsten Position im Textpuffer und auf dem Display. Verlässt der Cursor den angezeigten Bereich, ist er nicht mehr sichtbar.

  Ist Shift dagegen eingeschaltet, dann wandert der Cursor zwar immer noch im Textpuffer, aber gleichzeitig verschiebt sich der im Display angezeigte Bereich des Textpuffers um eine Stelle in Gegenrichtung, so dass der Cursor scheinbar still steht, während der Text im Display wandert.

- Display ein/aus

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Zeit
0	0	0	0	0	0	1	D	C	B	2,25 ms

  Das Bit D schaltet das Display insgesamt ein oder aus. Dabei wird der Textpuffer nicht verändert, sondern nur bei D = 0 nicht mehr angezeigt (D = 0: Display aus, D = 1: Display ein).

  Das Bit C schaltet den Cursor ein und aus. Der Cursor wird bei C = 1 an der nächsten Schreibposition angezeigt (C = 0: Cursor aus, C = 1: Cursor ein).

  Mit Bit B wird der Cursor zwischen Unterstrich und blinkendem Block umgeschaltet (B = 0: Unterstrich, B = 1: blinkender Block).

- Cursor/Display-Shift

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Zeit
0	0	0	0	0	1	S/C	R/L	*	*	0,08 ms

Mit dieser Funktion kann der angezeigte Ausschnitt des Textpuffers oder der Cursor um eine Position nach rechts oder links verschoben werden. S/C steuert den Cursor (S/C = 0: Verschieben des Cursors, S/C = 1: Verschieben des Displays) und R/L bestimmt die Richtung (R/L = 0: nach links, R/L = 1: nach rechts).

- Function Set

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Zeit
0	0	0	0	1	DL	N	F	*	*	0,08 ms

Einstellen grundlegender Funktionen beim Initialisieren. Mit DL wird das Interface auf 8 Bit oder 4 Bit Wortbreite eingestellt (DL = 0: 4-Bit-Modus, DL = 1: 8-Bit-Modus).

Mit dem N-Bit kann auf einzeiligen oder zweizeiligen Betrieb umgestellt werden (abhängig vom Typ). In der Regel wird hier (jenseits der 1x8-LCDs) der zweizeilige Modus nötig sein (N = 0: einzeilig, N = 1: zweizeilig).

Das F-Bit schaltet zwischen der Darstellung mit 5x8-Punktmustern und 5x11-Punktmustern um. Die meisten Displays arbeiten mit 5x8-Punktmustern (F = 0: 5x8-Darstellung, F = 1: 5x11-Darstellung).

- CG RAM Address Set

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Zeit
0	0	0	1	A	A	A	A	A	A	0,08 ms

Das interne Adressregister des Character-Generator-RAMs wird mit einer Adresse (in den Bits 0...5) geladen. Die nachfolgenden Daten werden unter dieser Adresse abgelegt.

- DD RAM Address Set

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Zeit
0	0	1	A	A	A	A	A	A	A	0,08 ms

Das interne Adressregister des Daten-RAMs wird mit einer Adresse (in den Bits 0...5) geladen. Die nachfolgenden Daten werden unter dieser Adresse abgelegt. Dieser Befehl ermöglicht also ein freies Positionieren des Cursors. Beachten Sie die Tabelle 1.3 bei der Wahl der gültigen Adressen.

- Busy-Flag/Address Read

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Zeit
0	1	BF	A	A	A	A	A	A	A	0,08 ms

Auslesen des Busy-Flags (BF = 1: Display ist beschäftigt, BF = 0: Display ist empfangsbereit) und des Standes des internen Adressregisters.

- CG RAM/DD RAM Data Write

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Zeit
1	0	D	D	D	D	D	D	D	D	0,12 ms

Schreiben von Daten in den Display-Zeichenpuffer oder eines Punktmusters für ein frei zu definierendens Zeichen in das Character-Generator-RAM. Das Definieren eines Zeichens wird etwas später behandelt. Dieser Befehl folgt immer direkt auf einen Befehl zum Setzen der DD-RAM- oder CG-RAM-Adresse.

Sollen Daten an beliebiger Stelle auf das Display gebracht werden, können Sie wie folgt vorgehen: Bei RS = 0 erfolgt der Befehl „Set DD-RAM Address“. Es sind mit diesem Befehl 128 Positionen adressierbar. Bei RS = 1 kann man nun die gewünschten ASCII-Daten zum Display schicken. Wurde die automatische In- oder Dekrementierung gewählt, rückt die Position bei jedem Zeichen automatisch um eine Stelle weiter, ohne dass dafür ein besonderer Befehl erteilt werden müsste.

- CG RAM/DD RAM Data Read

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Zeit
1	1	D	D	D	D	D	D	D	D	0,12 ms

Lesen von Daten in den Display-Zeichenpuffer oder eines Punktmusters für ein frei zu definierendes Zeichen aus dem Character-Generator-RAM. Dieser Befehl folgt immer direkt auf einen Befehl zum Setzen der DD-RAM- oder CG-RAM-Adresse.

1.2.6 Eigene Zeichen definieren

Die meisten LCD-Module eröffnen dem Anwender die Möglichkeit, acht Zeichen des Zeichensatzes (mit den Codes 0 bis 7) frei zu definieren, wenn beispielsweise Sonderzeichen oder Sinnbilder auf dem Display erscheinen sollen. Beim Programmieren des Moduls geht man davon aus, dass es korrekt initialisiert wurde und dass bei allen Lese- oder Schreibaktionen auf das Busyflag geachtet (bzw. genügend lange gewartet) wird.

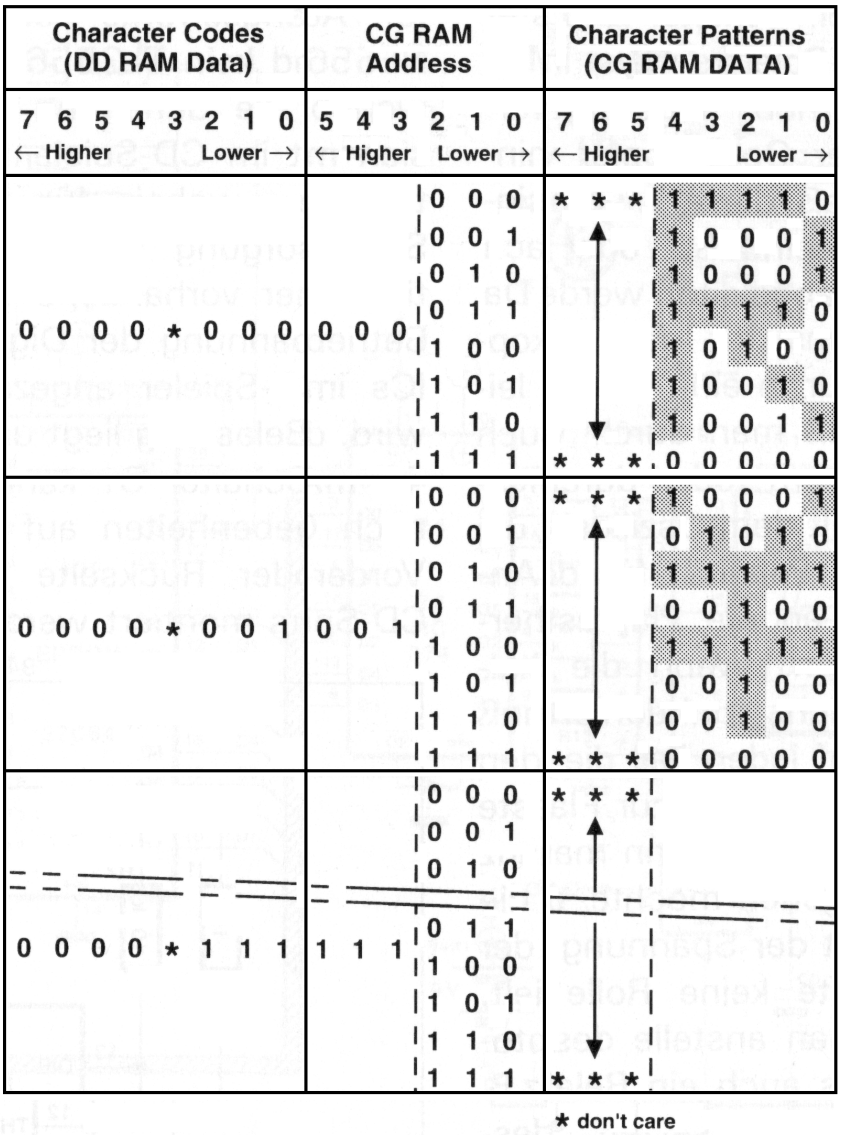

Bild 1.25: Definition eigener Zeichen

Der interne Speicher eines Moduls ist großzügiger bemessen, als es eigentlich notwendig wäre. So können 64 Bytes als gewöhnlicher Speicher verwendet werden, aber auch zur Definition weiterer Zeichen. Das geschieht folgendermaßen: Bei RS = 0 erfolgt der Befehl „Set CG-RAM Address“ (Code 01AAAAAA), der eine von 64 Positionen festlegt. Nun können die gewünschten Daten bei RS =

1 geschrieben werden. Auch hier wirkt sich das Auto-Inkrement oder -Dekrement aus. Nach dem Schreiben eines Bytes rückt die Speicheradresse um eine Position weiter (beziehungsweise zurück).

Wenn man das Display mit einer Matrixgröße von 5x7 Punkten pro Zeichen betreibt, können maximal acht Zeichen definiert werden, im 5x11-Punkte-Modus nur vier. Das Modul sollte auf Auto-Inkrement eingestellt sein. Nun gibt man beispielsweise Befehl 40h (0100 0000) ein, was einen Schreib- oder Lesezugriff auf die CG-RAM-Adresse 00h bedeutet. Danach schickt man die Daten für den Aufbau des Zeichens. Es wird mit der obersten Zeile begonnen; ein 1-Bit ergibt bei der dazugehörenden Stelle einen Punkt. Die drei höchstwertigen Bits der Zeichendefinition haben keine Funktion (die Zeichen sind ja nur 5 Pixel breit). In Bild 1.25 ist dies verdeutlicht. Die anderen Zeilen schreibt man auf die gleiche Art und Weise. Sie können alle acht Zeichen in einem Rutsch schreiben. Die neunte Zeile ist nämlich die erste Zeile des zweiten Zeichens. Auf diese Weise lassen sich alle 64 Zeilen der acht Zeichen definieren. Soll nur ein Zeichen definiert werden, muss die Adresse jeweils ein Vielfaches von 8 sein (also 00h, 08h, 10h, 18h, 20h, 28h, 30h, 38h). Die achte Zeile eines jeden Zeichens besteht ausschließlich aus Nullen, da sie für den Cursor reserviert ist. Die Zeichen werden dann über die Codes 0 bis 7 erreicht.

Befindet sich das LCD-Modul im 5x11-Modus, können nur vier Zeichen definiert werden. In diesem Fall umfasst ein Zeichen zehn Zeilen, die elfte bleibt für den Cursor frei. Die Zeilen 12 bis 16 sind jeweils für die Definition der Zeichen nicht notwendig und können als freier Speicher genutzt werden. Die 17. Zeile ist dann die erste des zweiten Zeichens usw.

1.2.7 Initialisierung des Displays

Bevor das Display benutzt werden kann, muss es vom PC aus initialisiert werden. Der Displaycontroller kennt eine Power-on-Reset-Funktion, welche die meisten Displays nutzen. Diese läuft beim Anlegen der Betriebsspannung automatisch ab und bringt das Display in einen „fail safe“-Zustand:

- 8-Bit-Interface (DL = 1)
- 1-zeiliges Display (N = 0)
- 5x8-Punkt-Matrix (F = 0)
- Display aus, Cursor aus, Cursorblinken aus (D = 0, C = 0, B = 0)
- Displayshift aus (S = 0)
- Cursor wandert bei jedem neuen Zeichen nach rechts (I/D = 1)

Tabelle 1.6: Initialisierung mit 8-Bit Interface

Power On **Mindestens 15 ms warten!**				
RS	R/W	Befehl (hex)	Beschreibung	Wartezeit
0	0	30	Interface auf 8 Bit setzen	4100 µs
0	0	30	Interface auf 8 Bit setzen	4100 µs
0	0	30	Interface auf 8 Bit setzen	100 µs
0	0	38	8-Bit-Modus, 2-zeilig, 5x8-Matrix	100 µs
0	0	01	Display löschen	2250 µs
0	0	02	Cursor Home	2250 µs
0	0	06	Cursor nach rechts, kein Display shift	100 µs
0	0	0E	Display & Cursor ein, Strich-Cursor	100 µs
Fertig!				

Während der ca. 15 ms langen Reset-Prozedur kann das Display keine Befehle von außen annehmen, das Busy-Flag ist gesetzt. Nach dem Reset ist das Display ausgeschaltet. Eine Initialisierungsroutine muss das Display aus diesem Zustand in den gewünschten Betriebsmodus bringen. Eine sichere Routine bringt das Display aus jedem Zustand in den gewünschten Betriebsmodus. Letztlich wird diese Routine auch nach einem Reset des steuernden Prozessors (der kein Power-on-Reset sein muss)

durchlaufen. Für die Initialisierung werden einige Steuerbefehle zum Display geschrieben (RS = 0, R/W = 0). Die Tabelle 1.6 beschreibt den Ablauf der Initialisierung für ein 8-Bit-Interface.

Bei den ersten beiden Befehlen lässt sich das Busy-Flag nicht testen, es muss also unbedingt gewartet werden! Nach jedem der folgenden Befehle kann entweder auf Busy-Flag = 0 getestet werden, oder man muss die entsprechende Zeit abwarten.

Die Initialisierung im 4-Bit-Modus erfolgt analog. Auch hier wird zweimal der Befehl 30h gesendet. Danach jedoch 20h, um den 4-Bit-Mode einzuschalten und anschließend die Befehle jeweils in Form von zwei Werten, wobei immer nur die Bits D4...D7 verwendet werden (D0...D3 sind immer 0): 20h, 80h, 00h, 80h, 00h, 10h, 00h, 20h, 00h, E0h.

1.2.8 Display-Software

Das folgende Listing enthält einige Funktionen zum Ansteuern eines LCD, die als Bausteine in ein Anwendungsprogramm eingefügt werden können. Sie sollen auch den Weg zu eigenen Experimenten aufzeigen. Vergessen Sie nicht, im Hauptprogramm zu Beginn die Ports zu belegen (mit `ioperm(BASE, 3, 1)`) und vor Beendigung des Programms wieder freizugeben (mit `ioperm(BASE, 3, 0)`). Das Programm benötigt neben den `#include`-Anweisungen noch folgende Konstantendefinitionen:

```
/* Bassisadressen der Parallelports */
#define LPT1 0x378
#define LPT2 0x278
#define LPT3 0x3BC

/* aktuell verwendeter Port */
#define BASE LPT1
#define STATUS  BASE+1
#define CONTROL BASE+2
```

Tabelle 1.7: Funktionen zum Steuern des LCD

`lcd_backlight()`	Hintergrundbeleuchtung schalten (on=0: aus, on=1: ein)
`lcd_write_command()`	Kommando an das LCD ausgeben
`lcd_putchar()`	Buchstaben auf dem LCD darstellen
`lcd_puts()`	Zeichenkette auf dem LCD darstellen
`lcd_init()`	LCD initialisieren
`lcd_goto()`	Cursor auf die Position (Zeile, Spalte) positionieren
`lcd_clear()`	LCD löschen
`lcd_home()`	Cursor home
`lcd_entry_mode()`	Entry-mode setzen (Bit 1: Cursorbewegung rechts/links, Bit 0: Zeilenshift ein/aus)
`lcd_cursor()`	Cursor-Mode setzen (Bit 1: ein/aus, Bit 0: Blink/Unterstrich)

Die in Tabelle 1.7 aufgelisteten LCD-Funktionen umfassen die wichtigsten Aktionen mit dem Display. Hinzu kommt die Funktion `Strobe()`, die den Strobe-Impuls zur Datenübernahme erzeugt. Die genaue Parameterversorgung entnehmen Sie bitte dem Listing.

```
void Strobe(void)
  /* Strobe-Impuls ausgeben */
  {
  outb(inb(CONTROL) | 1, CONTROL);     /* Set Strobe */
  usleep(2);
  outb(inb(CONTROL) & 0xFE, CONTROL);  /* Reset Strobe */
  usleep(2);
  }

void lcd_backlight(int on)
  /* Hintergrundbeleuchtung ein/aus */
  {
  if (on) outb(inb(CONTROL) | 4, CONTROL);
  else outb(inb(CONTROL) & 0xFB, CONTROL);
  }
```

```
void lcd_write_command(unsigned char value)
  /* Kommando an das LCD ausgeben */
  {
  outb(inb(CONTROL) | 2, CONTROL); /* RS = Command */
  outb(value, BASE);
  Strobe();
  outb(inb(CONTROL) & 0xFD, CONTROL); /* RS = Data */
  usleep(100);
  }

void lcd_putchar(unsigned char value)
/* Buchstaben auf dem LCD darstellen */
  {
  outb(inb(CONTROL) & 0xFD, CONTROL); /* RS = Data */
  outb(value, BASE);
  Strobe();
  usleep(100);
  }

void lcd_puts(char *s)
  /* String auf dem LCD darstellen */
  {
  while (*s != '\0')
    { lcd_putchar(*s); s++; }
    }

void lcd_init(void)
  /* LCD initialisieren */
  /* dran denken: lcd_write_command() wartet selbst schon 100 us */
  {
  lcd_write_command(0x30); /* Interface auf 8 Bit setzen */
  usleep(4100);
  lcd_write_command(0x30); /* Interface auf 8 Bit setzen */
  usleep(4100);
  lcd_write_command(0x30); /* Interface auf 8 Bit setzen */
  usleep(50);
  lcd_write_command(0x38); /* 8-Bit-Modus, 2-zeilig, 5x8-Matrix */
  usleep(50);
  lcd_write_command(0x01); /* Display loeschen */
  usleep(2200);
  lcd_write_command(0x02); /* Cursor home */
  usleep(2200);
  lcd_write_command(0x06); /* Cursor nach rechts, kein Display shift */
  lcd_write_command(0x0E); /* Display und Cursor ein, Strich-Cursor */
  lcd_backlight(1);        /* Licht an */
  }

void lcd_goto(int row, int column)
  {
  /* Cursor auf [row, column] positionieren */
  if(row == 2) column += 0x40;
  lcd_write_command(0x80 | column);
  usleep(2200);
  }

void lcd_clear(void)
  /* LCD loeschen */
  {
  lcd_write_command(0x01);
  usleep(2200);
  }

void lcd_home(void)
  /* Cursor nach links oben */
  {
  lcd_write_command(0x02);
  usleep(2200);
  }

void lcd_entry_mode(int mode)
  /* Entry-mode setzen (Bit 1: Cursorbewegung rechts/links,
     Bit 0: Zeilenshift ein/aus */
  {
  lcd_write_command(0x04 + (mode%4));
```

```
    usleep(100);
    }

  void lcd_cursor(int cursor)
    /* Cursor-Mode setzen (Bit 1: ein/aus, Bit 0: Blink/Unterstrich */
    {
    lcd_write_command( 0x0C + (cursor%4));
    usleep(2200);
    }
```

Eigentlich lohnt es sich nicht, die Software selbst zu schreiben. Es existieren nämlich neben vielen anderen zwei leistungsfähige Programmpakete zum Betrieb von LCDs unter Linux: LCD4Linux (*ssl.bulix.org/projects/lcd4linux/*) und LCDproc (*lcdproc.omnipotent.net*). Beide bieten gute und umfangreiche Dokumentationen über die unterstützten Display-Controller, die Verkabelung und die Softwarekonfiguration. Viele Distributionen bringen LCD4Linux bereits mit, so dass sich die Software per Paketmanager einrichten lässt. Um beispielsweise LCD4Linux zu konfigurieren, bearbeiten Sie die Datei `/etc/lcd4linux.conf`. Eine genaue Beschreibung der Syntax finden Sie in der Manpage oder im Howto von LCD4Linux. Beide Dokumente stehen auch in deutscher Sprache zur Verfügung. Der Schaltplan in Bild 1.23 ist – bis auf das Schalten der Hintergrundbeleuchtung – LCD4Linux-konform. Vor dem Start von LCD4Linux muss das `ppdev`-Modul geladen sein, sonst fehlt Ihnen möglicherweise das Device `/dev/parport0`. Als Root können Sie auch direkt über die Ports auf das LCD zugreifen. Solange Sie noch an der Konfiguration basteln, sollten Sie LCD4Linux auch nicht als Daemon, sondern als Vordergrundprogramm starten (Option: −F) und sich mit der Option −vvv sämtliche Debug-Informationen geben lassen.

1.3 Grafikdisplays

Die LC-Grafikdisplays funktionieren prinzipiell genauso wie die Textdisplays, doch sind hier die Bildpunkte einzeln ansprechbar. Je nach Hersteller gibt es auch kombinierte Text/Grafikdisplays. Die Ansteuerung der Displays unterscheidet sich ebenfalls von Typ zu Typ. Manche Modelle warten mit hoher Intelligenz auf und lassen sich per ASCII-Befehl ansteuern, andere füllen den Bildspeicher, wie wir das oben beim Character-Generator-RAM gesehen haben. Die Versorgung mit Daten erfolgt bei den einen parallel (8 Bit), bei den anderen seriell (Data und Clock). Insgesamt also eine bunte Mischung. Je nach Modell erfolgt das „Füttern“ des Displays auch noch über mehrere Speicherseiten, zwischen denen umgeschaltet werden muss. Alles in allem sind solche Displays softwaremäßig nicht ganz trivial anzusteuern.

Als Alternative habe ich mir die Grafikeinheiten von Electronic Assembly herausgesucht, die relativ preiswert und leicht erhältlich sind – und vor allem hochintelligent (*http://www.lcd-module.de/deu/rs232/rs232.htm*). Die Module sind per RS232-Schnittstelle anzusteuern, und einige von ihnen werden sogar mit Touch-Screen geliefert. Diese **LCD-Grafikeinheiten** sind intelligente Grafikdisplays mit zwei oder drei eingebauten Zeichensätzen sowie diversen Grafikfunktionen. Sie erlauben einen zeit- und kostensparenden Einsatz. Außer 5-V-Versorgung und RS-232-Schnittstelle sind keine weiteren Signale erforderlich. Zum Betrieb ist auch keine weitere Software oder ein spezieller Treiber nötig. Alle Grafikroutinen sind im integrierten High-Level-Grafikkontroller des Displays integriert. Dieser ermöglicht mit wenigen Befehlen den Aufbau eines übersichtlichen und ansprechenden Bildschirms. In der Serie sind Grafik-LCDs mit den Auflösungen 120x32, 128x64, 128x128, 240x64 und 240x128 Pixeln lieferbar. Ein Beispiel (240x64) zeigt Bild 1.26.

Bild 1.26: LCD-Grafikdisplay EA GE240-6KCV24

Das Display ist für +5V Betriebsspannung ausgelegt. Die Datenübertragung erfolgt seriell asynchron (8 Datenbits, 1 Stoppbit, keine Parity) im RS-232-Format mit V-24-Pegeln (±10V) oder mit 5V-CMOS-Pegeln. Die Baudrate kann über drei Lötbrücken zwischen 1200 und 115 200 bps eingestellt werden.

Neben den Datenleitungen (TX, RX) stehen die Handshake-Leitungen RTS und CTS zur Verfügung. Bei kleinen Datenmengen ist deren Auswertung jedoch nicht erforderlich.

Zusätzlich sind an einer Lötleiste acht E/A-Ports vorhanden, die als Ausgang oder Eingang individuell beschaltet werden können. Mögliche Anwendungen dafür sind das Schalten externer Komponenten über entsprechende Treiber (maximaler Strom: 10 mA) oder das Einlesen von Tasten bzw. Schaltern. Das dafür notwendige Pull-up-Widerstandsnetzwerk findet sogar noch Platz auf der Platine.

Es gibt drei verschiedene Zeichensätze: 4x6, 6x8 und 8x16 Pixel. Jeder Zeichensatz kann dabei um die Faktoren 1, 2, 3 oder 4 in Höhe und Breite skalieren. Mit dem größten Zeichensatz (8x16 Pixel) lassen sich somit bei 4-fach Zoom (ergibt 32x64 Pixel) bildschirmfüllende Worte und Zahlen darstellen. Zusätzlich können bis zu 16 eigene Zeichen definiert werden, die so lange erhalten bleiben, bis die Versorgungsspannung abgeschaltet wird.

Bei diversen Befehlen kann als Parameter ein Mustertyp (0...7) eingestellt werden. So lassen sich rechteckige Bereiche, Bargraphs und sogar Texte mit unterschiedlichen Mustern verknüpfen und darstellen. Bild 1.27 zeigt die zur Verfügung stehenden Füllmuster.

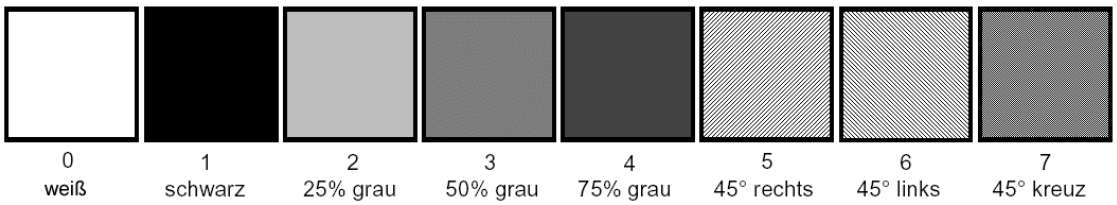

Bild 1.27: Füllmuster des LCD-Grafikdisplays

Für die Programmierung der Grafikeinheit existieren dank der eingebauten Intelligenz komplexere Befehle, beispielsweise das Zeichnen eines Rechtecks oder eines Bargraphen. Der Ursprung des Koordinatensystems liegt in der linken oberen Ecke des Displays. Um beispielsweise ein Rechteck von (0,0) nach (64,15) zu zeichnen, werden über die serielle Schnittstelle folgende Bytes gesendet: `'R' 0x00 0x00 0x40 0x0F`. Zeichenketten lassen sich ebenfalls pixelgenau platzieren. Das Mischen von Text und Grafik ist jederzeit möglich. Jeder Befehl beginnt mit einem Befehlsbuchstaben, gefolgt von einigen Parametern.

Alle Befehle und deren Parameter, z. B. Koordinaten, werden immer als Folge von Bytes erwartet. Dazwischen dürfen keine Trennzeichen, etwa Leerzeichen oder Tabs, verwendet werden. Die Befehle benötigen auch kein wie immer geartetes Abschlusszeichen. Dazu ein Beispiel mit dem „Z"-Befehl. Dieser Befehl schreibt an die Koordinate (xpos, ypos) eine Zeichenkette, die mit einem Nullbyte abgeschlossen ist (wie bei C üblich). Dabei wird der vorher eingestellte Textausgabe-Modus (Art der Ausgabe, Richtung etc.) berücksichtigt. Die Koordinatenangaben beziehen sich auf die linke obere Ecke des ersten Zeichens. die Variablen `xpos` und `ypos` werden am besten als `unsigned char` definiert. Das folgende Listing schreibt an die Koordinate (6, 11) die Zeichenkette „Linux".

```
xpos = 6; ypos = 11;
strcpy(text,"Linux");
fprintf(serdev,"Z%c%c%s",xpos, ypos, text);
```

Recht interessant sind auch die Bargraph-Befehle. Zuerst werden bis zu acht Bargraphs definiert, die nach links, rechts, oben oder unten ausschlagen können. Der bei Vollausschlag vom Bargraphen beanspruchte Bereich kann individuell festgelegt werden. Des weiteren werden Anfangswert (kein Ausschlag) und Endwert (Vollausschlag) definiert. Mit diesen Werten ist die Skalierung des Graphen festgelegt. Gezeichnet wird der Balken immer im Inversmodus mit dem vorgegebenen Muster (siehe oben). Der Hintergrund bleibt somit in jedem Fall erhalten. Mit einem „B"-Befehl wird der Bargraph definiert; auf dem Display ist er dann noch nicht zu sehen. Zum Beispiel:

```
unsigned char str[11] = {'B','O',1,10,20,20,50,4,20,1,0};
...
fprintf(serdev,"%s",str);
```

Es wird der Bargraph Nummer 1 definiert, der nach oben ausschlägt. Bei Vollausschlag nimmt er einen rechteckigen Bereich von (10,20) bis 20,50) ein. Anfangs- und Endwert entsprechen einer Messwertanzeige von 4 bis 20 mA. Es wird Muster 1 verwendet. Die Null am Ende wird nicht ausgegeben, sondern markiert das Ende der Zeichenkette.

Befehlstabelle EA GE128-6N3V24												
Befehl												**Anmerkung**
Funktionen zur Textausgabe												
Text-Modus	T	R L O U	n1	mst								R/L/O/U: Zeichenkette nach (R)echts,(L)inks,(O)ben, (U)nten schreiben; n1: Verknüpfungsmodus für Textausgabe 1=setzen; 2=löschen; 3=invers; 4=replace; 5=invers replace; mst: Muster Nr. 0..7 verwenden;
Font einstellen	F	n1	n2	n3								Font Nr. n1 einstellen; n1=1: 4x6 Font; n1=2: 6x8 Font; n2=3: 8x16 Font n2+n3=Zoomfaktor (1..4); n2=X-Faktor; n3=Y-Faktor;
ASCII-Zeichen setzen	A	x1	y1	n1								Das Zeichen n1 wird an Koordinate x1,y1 gesetzt. (Bezug links oben)
Zeichenkette ausgeben	Z	x1	y1	...	NUL							Eine Zeichenkette (...) an x1,y1 ausgeben; Zeichen ´NUL´ ($00)=Ende
Zeichen definieren	E	n1	daten ...									n1=Zeichen Nr.; daten=Anzahl Bytes je nach akt. Font
Grafik-Befehle mit Verknüpfungsmodus												
Grafik-Modus	V	n1										n1: 1=setzen; 2=löschen; 3=invers; 4=replace; 5=invers replace;
Punkt setzen	P	x1	y1									Ein Pixel an die Koordinaten x1, y1 setzen
Gerade zeichnen	G	x1	y1	x2	y2							Eine Gerade von x1,y1 nach x2,y2 zeichnen
Gerade weiter zeichnen	W	x1	y1									Eine Gerade vom letzten Endpunkt bis x1, y1 zeichnen
Rechteck zeichnen	R	x1	y1	x2	y2							Ein Rechteck zeichnen; x1,y1,x2,y2 = gegenüberliegende Eckpunkte
Rundeck zeichnen	N	x1	y1	x2	y2							Ein Rechteck mit runden Ecken zeichnen; x1,y1,x2,y2 = Eckpunkte
Bereich m. Füllmuster	M	x1	y1	x2	y2	mst						Ein Bereich mit Muster mst (0..7) zeichnen; x1,y1,x2,y2 = Eckpunkte
sonstige Grafik-Befehle												
Display löschen	D	L										Gesamten Displayinhalt löschen (auf Weiß setzen);
Display invertieren	D	I										Gesamten Displayinhalt invertieren;
Display füllen	D	S										Gesamten Displayinhalt füllen; (auf Schwarz setzen);
Bereich löschen	L	x1	y1	x2	y2							Einen Bereich löschen; x1,y1,x2,y2 = gegenüberliegende Eckpunkte
Bereich invertieren	I	x1	y1	x2	y2							Einen Bereich invertieren; x1,y1,x2,y2 = gegenüberliegende Eckpunkte
Bereich füllen	S	x1	y1	x2	y2							Einen Bereich füllen; x1,y1,x2,y2 = gegenüberliegende Eckpunkte
Box zeichnen	O	x1	y1	x2	y2	mst						Ein Rechteck mit Füllmuster mst (0..7) zeichnen; (immer replace)
Rundbox zeichnen	J	x1	y1	x2	y2	mst						Ein Rundeck mit Füllmuster mst (0..7) zeichnen; (immer replace)
Bargraph zeichnen	B	nr	wert									Den Bargraph mit der ´nr´ (1..8) auf den neuen ´wert´ setzen
Bildbereich uploaden	U	x1	y1	daten ...								Einen Bildbereich nach x1,y1 laden; Daten des Bildes siehe Bildaufbau
Kontroll- / Definitions-Befehle												
Bargraph definieren	B	R L O U	nr	x1	y1	x2	y2	aw	ew	mst		Einen Bargraph nach L(inks), R(echts), O(ben), U(nten) mit der ´nr´ (1..8) definieren. x1,y1,x2,y2 sind das umschließende Rechteck des Bargraphs. aw,ew sind die Werte für 0% und 100%. mst=Muster Nr. (0..7)
Display Control	C	n1										n1=0:Display aus, Inhalt bleibt erhalten; n1=1:Display ein, Inhalt sichtbar
Select / Deselect Grafikkontroller	K	S	n1									Display mit Adresse n1 (n1=0..3; n1=4: alle) aktivieren
		D										Display mit Adresse n1 (n1=0..3; n1=4: alle) deaktivieren
I/O-Port schreiben	Y	n1	n2									n1=0..7: I/O-Port n1 rücksetzen (n2=0); setzen (n2=1); invertieren (n2=2) n1=8: Alle 8 I/O-Ports entsprechend n2 (=8-Bit Binärwert) einstellen
Sende-Befehle												
Hardcopy	H	x1	y1	x2	y2							Es wird der angegebene Bildinhalt angefordert. Zuerst werden die Breite und Höhe in Pixel und dann die eigentlichen Bilddaten gesendet.
I/O-Port lesen	X	n1										n1=0..7: I/O-Port <n1> einlesen (1=H-Pegel=5V, 0=L-Pegel=0V) n1=8: Alle 8 I/O-Ports I/O0..I/O7 als 8-Bit Binärwert einlesen
Displaytyp abfragen	?											Es wird der Displaytyp abgefragt. Zurückgesendet werden 3 Bytes: 128, 64, `V` (128x64 Pixel Auflösung, vertikal organisiert)

Bild 1.28: Befehlsübersicht des LCD-Grafikdisplays GE128-6N3V24
(Quelle: Electronic Assembly)

Ebenfalls mit dem „B"-Befehl, aber diesmal mit nur zwei Parametern – der Nummer des Graphen und dem aktuellen Wert – wird die Anzeige veranlasst. Der aktuelle Wert muss zwischen den voreingestellten Anfangs- und Endwerten liegen. Mittels `fprintf(serdev,"B%c%c", 1, 12)` erhält man beispielsweise einen Balken von halber Maximalhöhe.

Zudem lassen sich bis zu vier Displays an einer einzigen seriellen Schnittstelle anschließen. Die jeweilige Displayadresse (0…3) wird über Lötbrücken eingestellt. Da beim Parallelschalten der Handshakeleitungen (RTS) und der Datenleitungen (TXD) die Ausgänge gegeneinander arbeiten würden, muss durch zusätzliche Hardware eine Entkopplung sichergestellt werden (*wired and* bei den Steuerleitungen, *wired or* bei den Datenleitungen). Die Umschaltung zwischen den Displays erfolgt per Befehl, wobei mit dem Wert 255 (0xFF) alle Displays adressiert werden.

Bild 1.28 zeigt eine Befehlsübersicht der Displayfamilie. Andere Displays des Herstellers verfügen über ähnliche Befehle, manchmal wird dem jeweiligen Befehl ein ESC-Zeichen vorangestellt.

Bleibt noch nachzutragen, dass die Displays nicht nur mit einem einfachen E/A-Port, sondern auch als Komplett-Kit mit Touch-Panel geliefert werden. Bei allen Versionen des Typs *EA KITxxTP* bietet das integrierte Touch-Panel 10 x 6 Felder, die von 1 bis 60 durchnummeriert sind (Zeile 1: 1…10, Zeile 2: 11…20 usw.). Die Bedieneinheiten unterstützen das Touch Panel recht komfortabel. Mehrere Touch-Felder lassen sich beispielsweise zu einer größeren Gesamt-Taste zusammenfassen. Dazu werden zwei Felder angegeben: das linke obere und das rechte untere Feld (z. B. wird die Region mit den Feldern 13, 14, 15, 23, 24, 25 durch die Felder 13 und 25 definiert). Die Taste wird dann passend auf das Display gezeichnet und und ihr ein 8-Bit-Return-Code zugewiesen. Der Return-Code 0 kennzeichnet eine deaktivierte Taste. Beim Berühren der Touch-Tasten können diese automatisch invertiert und durch einen Summer die Berührung signalisiert werden. Gleichzeitig sendet das Modul den Return-Code der Taste über die serielle Schnittstelle – oder es wird ein internes Touch-Makro mit der Nummer des Return-Codes gestartet. Die Touch-Panels sind übrigens auch einzeln lieferbar.

Mittlerweise werden von verschiedenen Herstellern auch Touch-Displays geliefert, die keine Unterteilung in Tasten mehr besitzen, sondern eine homogene Fläche. Mit derartigen Displays lassen sich dann Benutzeroberflächen gestalten, die nicht nur die Eingaben von Text und Zahlen per Stift erlauben, sondern auch durch Gesten – wie es die Firma Apple mit seinen Telefonen und Tabletcomputern eindrucksvoll demonstriert.

1.4 Touchscreen

Ein Touchscreen ist ein kombiniertes Ein- und Ausgabegerät, bei dem durch Berührung des Bildschirms Eingaben getätigt werden können. Der Bildschirm wird mit dem Finger oder einem Stift berührt, worauf der Touchscreen die Position an den Rechner übermittelt.

1.4.1 Resistive Touchscreens

Resistive Touchscreens erkennen die Berührung durch leichten Druck auf eine Spezialfolie. Der Bildschirmist mit einer elektrisch leitenden Folie bedeckt. Darüber liegt eine zweite Folie, die durch mikroskopisch kleine Abstandshalter von der ersten Folie getrennt wird. Bei einer Berührung des Touchscreens wird die zweite Folie heruntergedrückt und kommt ein Kontakt zustande. Die Schichten bilden so einen Spannungsteiler, an dem der elektrische Widerstand gemessen wird, um die Position der Druckstelle zu ermitteln (Bild 1.29).

Es muss immer zweimal gemessen werden, einmal in X-Richtung und einmal in Y-Richtung. Erst dann ist die Position in der Fläche feststellbar. Bei der Realisierung gibt es zwei Grundprinzipien:

Four-Wire (Vier-Draht), wie es oben geschildert wurde, ist das einfachste und älteste Verfahren. Die Spannung wird abwechselnd im X- und Y-Richtung an die leitfähigen Schichten angelegt. Daher sind nur vier Anschlussdrähte erforderlich. Four-Wire hat den Nachteil schnell nachlassender Präzision bei der Erfassung der Druckstelle. Die äußere Polyesterschicht des Touchscreens wird durch die Benutzung mechanisch belastet, wodurch die leitfähige Beschichtung ihrer Innenseite an Gleichmäßigkeit verliert.

Five-Wire vermeidet das Nachlassen der Präzision, indem die äußere leitfähige Schicht nicht als Maß für die Position der Druckstelle herangezogen wird. Sie dient nur zum Weiterleiten der Spannung von der unteren Schicht und ist über einen zusätzlichen fünften Draht angeschlossen. Die vier übrigen Anschlüsse befinden sich an den Kanten der unteren Schicht. Zur Messungen werden jeweils zwei benachbarte Kanten direkt verbunden und dann an die beiden Kantenpaare eine Spannung angelegt. Danach wird die Kombination der Kanten gewechselt und die zweite Messung durchgeführt.

1.4.2 Kapazitiver Touchscreen

Ein kapazitiver Touchscreen besitzt als Deckschicht ein mit durchsichtigem Metalloxid beschichtetes Glassubstrat. Dabei liegt ein kleiner Abstand zwischen den horizontal und vertikal aufgedruckten Leiterbahnen, wodurch eine Matrix aus vielen Kondensatoren entsteht. Die eine Gruppe der Leiterbahnen bildet die sogenannten Drive-Lines, an die der Reihe nach Wechselspannung angelegt wird, wodurch eine Potentialdifferenz zur anderen Gruppe Leiterbahnen, den Sense-Lines, entsteht. An diesen wird die Spannung gegenüber den Drive Lines gemessen.

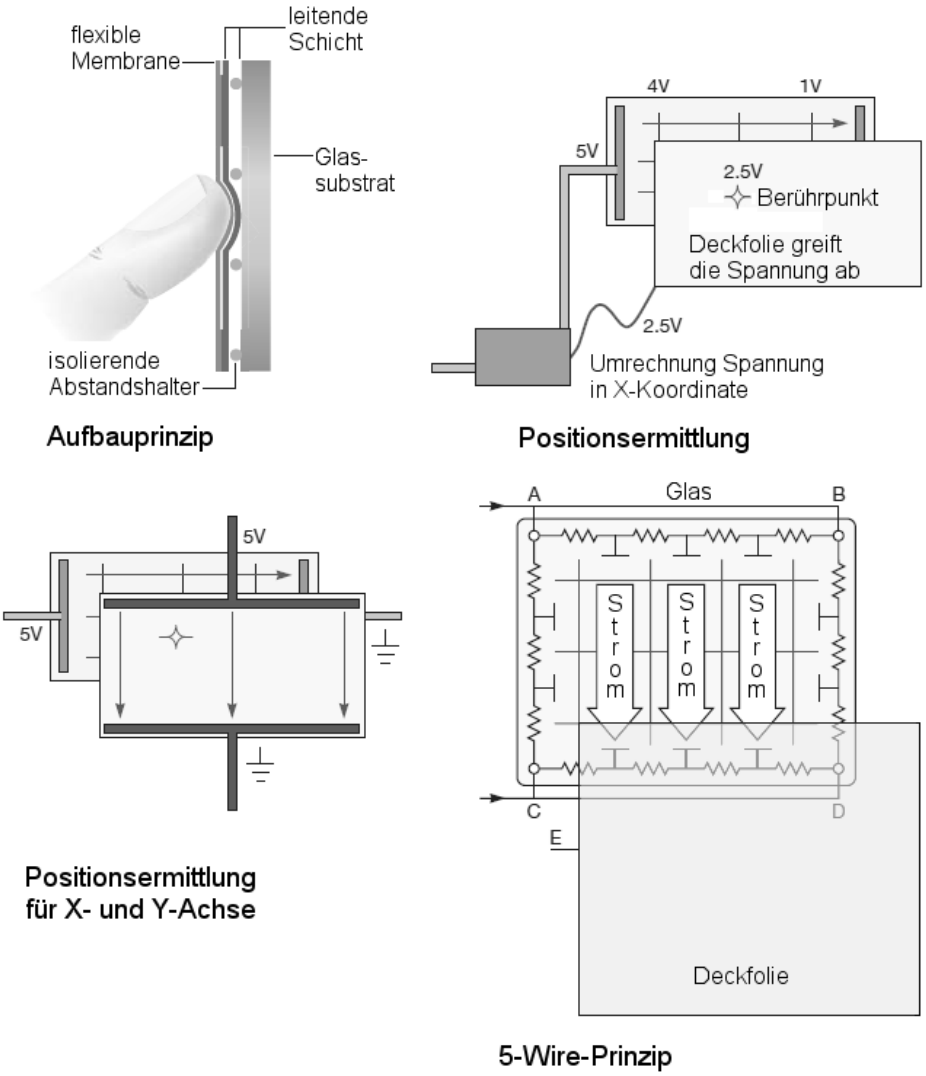

Bild 1.29: Arbeitsweise restisiver Touchscreens

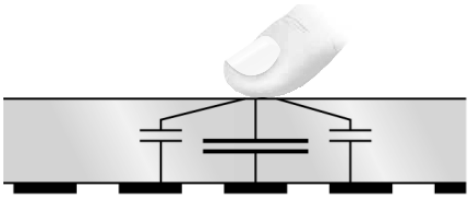

Bild 1.30: Arbeitsweise kapazitiver Touchscreens

Die „Projected Capacitive Touchscreens“ machen sich den Effekt der kapazitiven Kopplung zu Nutze. Bei Annäherung des Fingers verändert dieser das elektrische Feld im Bereich des Berührpunktes, was Verschiebungsströme verursacht, die Aufschluss über die Nähe des Fingers geben. Die Kalibrierung des Touchscreens ist dafür verantwortlich, ab welchem Wert die Störspannung als Berührung des Displays interpretiert wird. Dazu muss aber ein Finger den Touchscreen berühren, mit Handschuhen oder per Stift klappt es nicht (Bild 1.30).

Bei einer anderen Technik kommt eine leitfähige Beschichtung der transparenten Fläche zum Einsatz. Weitere Schichten werden nicht benötigt. Es werden die vier Ecken mit einer Wechselspannung beaufschlagt, die ein schwaches kapazitives Feld erzeugt. Das Auflegen des Fingers bedingt einen Spannungsabfall an diesem Punkt und damit einen Stromfluss zwischen den vier Ecken und dem Betätigungspunkt. Das Verhältnis der an den Ecken zu messenden Ströme ermöglicht das Feststellen der Fingerposition. Diese Systeme sind sehr robust.

1.4.3 Induktive Touchscreens

Induktive Touchscreens (Bild 1.31) haben gegenüber den anderen beiden Verfahren den Nachteil, dass sie sich nur über spezielle Eingabestifte (mit einer integrierten Spule) nutzen lassen, eine Technik, die von Grafiktabletts übernommen wurde.

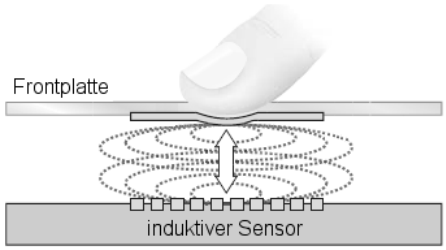

Bild 1.31: Arbeitsweise induktiver Touchscreens

Jedoch bieten sie auch einige Vorteile gegenüber den anderen Techniken:

- Eine aufliegenden Hand erzeugt keine Reaktion des Touchscreens, sondern nur der Stift.
- Die Bildschirmoberfläche kann (übrigens auch bei kapazitiven Touchscreens) aus Glas oder einem anderen robusten Material bestehen, da keine mechanische Verformung benötigt wird.
- Die Stiftposition kann sogar ermittelt werden, wenn der Stift die Oberfläche nicht berührt, sondern sich in einem (geringen) Abstand über ihr befindet.
- Die Induktion kann verwendet werden, um zusätzliche Elemente des Stiftes zu betreiben, beispielsweise Knöpfe.

1.4.4 Akustische Touchscreens

Diese Technik wertet die Laufzeit von Schallwellen aus, die von vier Mikrofonen, die sich an den vier Bildschirmecken des Touchscreens aufgenommen werden. Wird der Touchscreen berührt, entsteht ein Geräusch. Aus den Laufzeitunterschieden zwischen den einzelnen Mikrofonen wird die Berührungsposition bestimmt.

Die Oberflächenwellentechnik (Surface Wave-Touchscreen) arbeitet mit Ultraschall und akustischer Pulserkennung. Hier laufen Ultraschallwellen über die Display-Oberfläche und werden bei Berührung des Touchscreens teilweise absorbiert. Jede Position auf dem Glas erzeugt einen einzigartigen Klang. Mehrere Transducer, die an den Kanten des Touchscreens angebracht sind, nehmen den Klang auf und der Controller ermittelt damit die Position.

1.4.5 Optische Touchscreens

Hier sitzt eine Reihe von LEDs und lichtempfindlichen Sensoren an den Kanten des Bildschirms. So entsteht ein Gitter aus winzigen Lichtschranken. Unterbricht ein Finger oder Stift das Gitter an einer Stelle, kann die Elektronik den Punkt der Berührung ermitteln. Das klappt aber nicht so genau wie bei den anderen Typen. Schmutz auf den Sensoren kann ebenso zu Positionsfehlern führen wie eine Unterbrechung der Lichtschranke durch andere Finger oder Handschmuck. Deshalb wird diese Technik nur bei besonders großen Bildschirmen eingesetzt.

1.4.6 Touchpad

Dies gehört eigentlich nicht direkt zu den Touchscreens. Bei vielen Notebooks dient ein berührungsempfindliches Feld, das Touchpad, zur Steuerung des Mauszeigers auf dem Bildschirm. Es funktioniert ähnlich wie ein kapazitiver Touchscreen und wird aus diesem Grund hier mit erwähnt: Die Berührung mit dem Finger beeinflusst ein schwaches elektrisches Feld und die Elektronik ermittelt daraus die Position. Das funktioniert wie bei allen kapazitiven Touchscreens nur mit dem Finger, nicht mit Handschuhen oder einem Stift.

1.5 Elektronisches Papier, bistabile Displays

Bistabile Displays, Electronic Paper Displays (EPD), e-paper genannt, sind elektronische Displays, die mindestens die gleichen Eigenschaften wie herkömmlich bedrucktes Papier aufweisen. Minimaler Energiebedarf und eine gute Lesbarkeit unter Sonneneinstrahlung verschaffen dem elektronischen Papier Vorteile gegenüber LCDs. Als Trägermaterial dient in der Regel Kunststoff. Vielfach lassen sich die elektronischen Displays platzsparend zusammenrollen.

Der Ursprung der bistabilen Displays liegt in den 1970er Jahren, als im Forschungszentrum PARC von Xerox das erste elektronische Papier entwickelt wurde. Mittlerweile arbeitet eine Vielzahl von Unternehmen an der Entwicklung von elektronischem Papier. Bistabile Displays lassen sich einsetzen für Industrielle Anzeigen, Infotafeln für Fahrgastinformationen, Hinweisschilder, für die Auszeichnung von Ware, als Displays für mobile Terminals oder als Werbetafeln. In Konsumgütern werden sie bei E-Books, in Spielzeugen, bei Mobiltelefonen oder als digitale Bilderrahmen verwendet.

1.5.1 Elektrophoretische Displays (EPD)

Elektrophoretische Displays bzw. elektronisches Papier sind Displays, die bedrucktes Papier nachempfindet. Sie reflektieren Licht wie normales Papier und können den Inhalt z. B. einer Buchseite dauerhaft mit minimalem Stromverbrauch anzeigen. E-Papier ist meist flexibel und kann gebogen werden. Das Display besteht in der Regel aus elektrisch leitendem Kunststoff, der kleine Kügelchen enthält, in denen Pigmente auf elektrische Spannung reagieren. Nick Sheridon hatte am PARC in den 1970er Jahren zuerst eine Art elektronisches Papier entwickelt und nannte es „Gyricon". Es besteht aus kleinen, statisch geladenen Kügelchen, die auf der einen Seite schwarz und auf der anderen weiß sind. Die Anzeige auf dem Papier erfolgt durch ein elektrisches Feld, das die Kügelchen nach oben oder unten dreht.

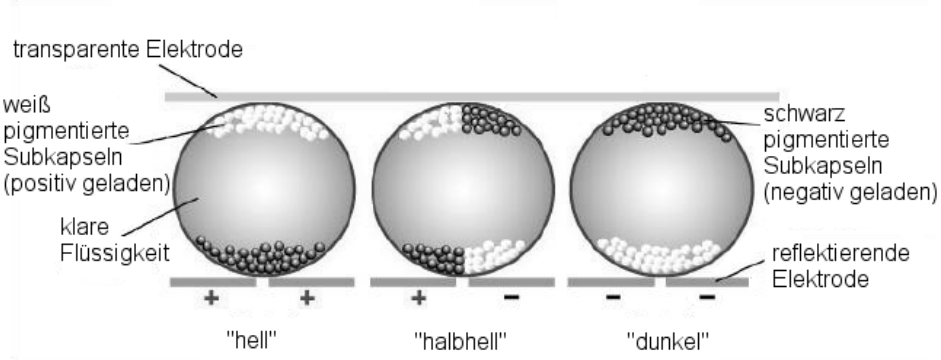

Bild 1.32: Schema eines Elektrophorese-Displays (Quelle: Lemme, Helmuth, Displays zum Aufrollen Elektronik 16/2003)

20 Jahre später verwendete Joseph Jacobson von E-Ink eine andere Form von flüssigkeitsgefüllten Mikrokapseln, in denen sich weiße und schwarze Partikel befinden (Bild 1.32). Diese Partikel sind unterschiedlich geladen und lagern sich je nach angelegter Spannung und dem daraus resultierendem elektrischen Feld an den jeweils gegenüberliegenden Polen der Kapseln an. Je nachdem, ob die weißen Partikel an der Oberseite der Mikrokapsel sind (helles „Pixel") oder unten blieben (dunkles „Pixel"). Diese Displaytechnik erlaubt wegen der Verwendung von Mikrokapseln flexiblen Kunststoff anstelle von Glas als Trägermaterial. Die Kapseln sind zwischen zwei Folien aus einem Polyester-Kunststoff eingebettet, wobei die Folien mit durchsichtigem Indiumzinnoxid (Indium Tin Oxide) beschichtet ist. Indiumzinnoxid ist leitend, dient als gemeinsame, großflächige Elektrode und wird auch bei den anderen hier vorgestellten Technologien sowie bei LC-Displays verwendet. Das gesamte bildgebende Laminat erreicht dabei ein Dicke von ca. 0,1 – 0,3 mm. Vorteilhaft ist, dass die Verteilung der Partikel nach dem Abschalten des Feldes erhalten bleibt, was Displays mit papierähnlichen Eigenschaften ermöglicht. LG.Philips stellte im Mai 2007 ein biegsames, farbiges E-Paper auf E-Ink-Basis vor.

Ähnlich funktioniert die Technik der Firma SiPix, die diese unter dem Namen „MicroCup" vermarktet. Im Gegensatz zu E-Ink verwendet SiPix ein geringeres Rastermaß und kann die Displays in Drucktechnik kostengünstig herstellen. Hier werden keine runden Kapseln verwendet, sondern regelmäßige vier- oder sechseckige Zellen (sogenannte Cups), die mit einer gefärbten Trägerflüssigkeit

gefüllt werden (schwarz, rot, grün, blau oder golden) in der sich elektrisch geladene weißen Pigmente befinden. Wenn die weißen Pigmente auf der hinten liegenden Elektrode liegen, ist nur die Trägerflüssigkeit zu erkennen und stellt somit ein dunklen bzw. farbigen Pixel dar. Liegen die weißen Partikel an der oberen (sichtbaren) Elektrode an, wird die Trägerflüssigkeit verdrängt und der Punkt erscheint weiß (Bild 1.33). Der Nachteil gegenüber der Lösung von E-Ink sind die fehlenden schwarzen Partikel, daher ergibt sich ein geringerer Kontrast. Quelle: http://www.sipix.com/company/news/images/sipix_activematrix.pdf

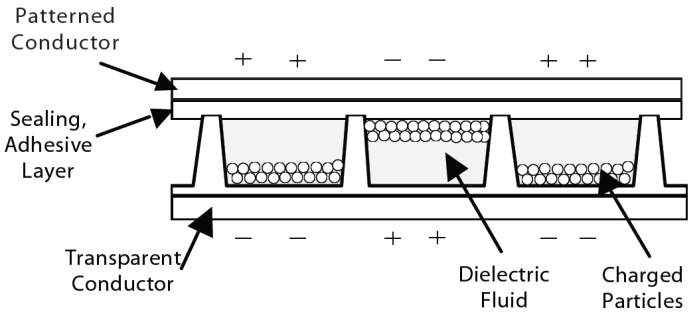

Bild 1.33: Schematicher Aufbau eines MicroCup Electronic Paper

Elektronisches Papier bietet gegenüber herkömmlichen Displays einen hohen Kontrast und einen breiten Betrachtungswinkel, es ist sehr dünn, biegsam sowie leicht und lässt sich in allen Größen und Formen herzustellen. Für portable Lesegeräte ist der geringe Stromverbrauch wichtig, Es muss nur zum Ändern des Bildinhalts (Seitenwechsel) Strom fließen. E-Paper ist bei normalem Raumlicht ebenso wie in hellem Sonnenschein lesbar. Nachteilig ist, dass derzeit die Grauwert- bzw. Farbwertauflösung sehr gering ist und der Seitenwechsel noch relativ träge verläuft, weshalb Videos und Animationen noch nicht darstellbar sind.

1.5.2 Elektrochrome Displays (ECD)

Elektrochrome Displays basieren auf speziellen Kunststoffen, z. B. Polythiophene, die sich unter Stromfluss verfärben. Dabei tritt, als Folge einer reversiblen Oxidation oder Reduktion des Materials, eine Veränderung der Absorptions- bzw. Reflexionseigenschaften auf.

Ein ECD besteht aus drei Schichten: Einem PET-Substrat (Plastikfolie), das mit einem organischen Leiter (PEDOT) beschichtet ist. Als zweite Schicht folgt das als Folie ausgebildete Elektrolyt, das von einer dritten Schicht, einem ebenfalls mit einem organischen Leiter beschichteten, durchsichtigen PET-Substrat, bedeckt wird. Die Schicht aus PEDOT hat dabei zwei Funktionen: als Zuleitung, um Strom zu der zu verfärbenden Fläche zu leiten und als elektrochromes Material. Um eine Verfärbung zu erzeugen, wird eine Spannung zwischen der oberen und unteren PEDOT-Schicht angelegt. Die positiv geladenen Ionen wandern zur negativ geladenen Elektrode und die negativ geladenen Ionen zu der positiv geladenen Elektrode. An der negativen Elektrode können nun aufgrund der positiven Ionen Elektronen auf das elektrochrome Material fließen und es somit reduzieren (analog zu der positiven Elektrode, an der die Ionen oxidiert werden). Als Folge davon erhöht sich die Lichtabsorption im roten Bereich, das Material erscheint dunkelblau (Bild 1.34). Kehrt man die Stromrichtung um, läuft die Reaktion umgekehrt ab, und das Material entfärbt sich wieder.

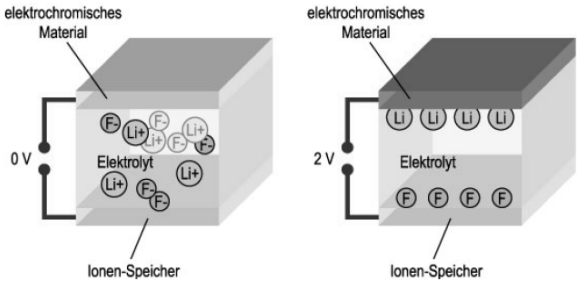

Bild 1.34: Schema eines Elektrochromen Displays (Quelle: Lemme, Helmuth, Displays zum Aufrollen Elektronik 16/2003)

Durch Aufbringung von Strukturen auf die Substratoberfläche sind beliebige Anzeigen möglich. Die Firma Siemens hat bereits auf der CeBIT 2003 unter dem Namen „ePYRUS“ verschiedene rollbare Displays auf elektrochromer Basis bis zur DIN-A4-Größe vorgestellt, dieses aber bis heute nicht zur Serienreife gebracht. Derzeit erreicht „ePYRUS“ nur eine Metastabilität (im Gegensatz zur Bistabilität von E-Ink), was bedeutet, dass die Verfärbung nur für eine gewisse Zeit anhält, danach muss das Bild wieder neu aufgebaut werden.

Elektrochrome Gläser werden dagegen bereits heute schon in der Bauindustrie als schaltbare Fensterscheiben bzw. Glasflächen oder auch Folien verwenden, die wahlweise durchsichtig oder milchig sind (z. B. die Rückwand der ICE-Fahrerkabine). Dieses Verfahren wird ebenfalls für automatisch abblendende PKW-Rückspiegel verwendet.

1.5.3 Cholesteric LCD

Die Firma Kent Displays hat die normale LCD-Technik weiter entwickelt und bistabile Displays auf den Markt gebracht, die ohne weiteren Energieaufwand ihre Informationen beibehalten. Hierbei ist der Aufbau identisch mit einem normalen LCD, der verwendete Flüssigkristall weist aber bistabile Zustand-Charakteristika auf (Verwendung von cholesterische Phasen anstelle von nematischen Phasen).

Das „cholesteric liquid crystal“ (ChLCD) muss bei der Verwendung in kleine Kapseln von etwa 10 – 12 Mikrometer Durchmesser eingeschlossen werden, die in einer wässrigen Lösung schwimmen. Das Material hat zwei stabile Zustände:

- **planar**: einfallendes Licht wird reflektiert,
- **focal conic**: die Kapseln erscheinen fast durchsichtig und der dunkle Hintergrund wird sichtbar.

In der aktuellen Entwicklung ist es gelungen, die elektrisch steuerbare Flüssigkeit, bestehend aus „cholesteric reactive mesogen“ mit einer spannungsabhängigen Blauverschiebung herzustellen. Prinzipiell sind damit auch farbige, flexible Displays möglich.

1.5.4 Elektrowetting Displays

Eine weitere Entwicklung ist das sogenannte Electrowetting-Verfahren (übersetzt etwa „elektrischer Kapillareffekt“), das auf der Veränderung von Oberflächeneigenschaften von Flüssigkeitssystemen durch ein elektrisches Feld basiert. Das Anlegen einer elektrischen Spannung beziehungsweise eines elektrischen Feldes beeinflusst die Oberflächenspannung von Wassertropfen derart, dass eine Verformung stattfindet und sich somit die Oberflächeneigenschaften verändern. Dieser Effekt wird beim Electrowetting genutzt. Das Grundprinzip ist in Bild 1.35 zu sehen: Ein Wassertropfen befindet sich auf einer hydrophoben (wasserabweisenden) Schicht, dass durch ein Dielektrikum von einer Elektrode getrennt ist. Eine zweite Elektrode steckt in diesem Wassertropfen. Im spannungslosen Zustand (obere Abbildung) ist der Wassertropfen aufgrund der wasserabstoßenden Schicht kontrahiert. Legt man jedoch eine Spannung an, zerfließt der Tropfen (untere Abbildung). Der Kontaktwinkel Θ zwischen der hydrophoben Schicht und dem Wassertropfen sinkt mit zunehmender Spannung. Für Tropfen der Größe 1 mm beträgt diese Spannung etwa 10 V bis 20 V in Anhängigkeit von den Schichteigenschaften. Die Verformung erfolgt praktisch leistungslos, da kein direkter Stromfluss durch die beiden Elektroden zustande kommt.

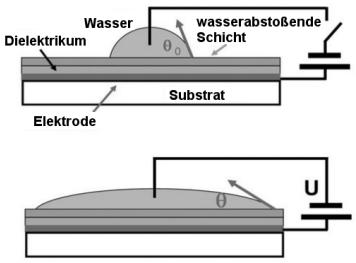

Bild 1.35: Schema des Elektrowettings

Ähnlich arbeitet das Verfahrens von Philips. Hier gibt es einzelne, mit Wasser gefüllte Kammern, in denen sich farbige Öltropfen befinden. Aufgrund der hydrophoben bzw. lipophoben Eigenschaften der beiden Flüssigkeiten, erfolgt keine Durchmischung, sondern eine scharfe Abgrenzung. Im Grundzustand bedeckt der (flache) Öltropfen eine reflektierende Grundschicht und es ist nur die Eigenfarbe des Öls sichtbar. Wird eine Spannung angelegt, so zieht sich der Tropfen auf einen Bruchteil der ursprünglich bedeckten Fläche zusammen und die reflektierende Grundschicht wird sichtbar.

Prinzipiell sind zwei Arten zu unterscheiden, die sich jedoch in ihren typischen Anwendungen praktisch nicht überschneiden:

- Ein-Kammer-System: Es wurde eingeführt von Liquavista. Dieser Ansatz besitzt Vorteile bei kleinen Pixeln und ist videofähig, jedoch nicht bistabil.
- Zwei-Kammer-System: Es wurde entwickelt von der Firma adt. Diese Displays sind bistabil und eher für Pixel größer als 1 mm geeignet.

Als einziges von allen hier vorgestellten Verfahren, erfüllt das Ein-Kammer-System bereits in einem frühen Entwicklungsstadium die Fähigkeit zur Bildwiedergabe in Videogeschwindigkeit. Die ersten Prototypen erreichten Schaltgeschwindigkeiten von 10 bis 13 ms, was einer Schaltfrequenz von ca. 100 Hz entspricht.

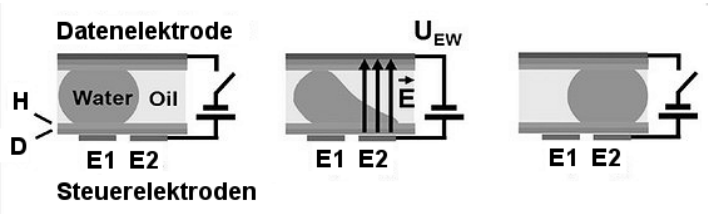

Bild 1.36: Prinzip der bistabilen Electrowetting-Technik von adt

Ein bistabiles Electrowetting-Display nach dem Zwei-Kammer-System besteht aus zwei gegenüberliegenden Elektroden (Daten- und Steuer-Elektrode in Bild 1.36), die jeweils mit einem dünnen Dielektrikum D) und einer hydrophoben Schicht H bedeckt sind. Prinzipiell funktioniert ein solches System mit Öl und Wasser wie oben gezeigt. Sobald ein elektrisches Feld an einer Seite des Tropfens anliegt (z. B. zwischen Daten-Elektrode und E1), fließt der Öltropfen in Richtung E2. Die vollständige Positionsverschiebung des Tropfens geschieht nach dem Abschalten der Spannung („Droplet Driven Display“). Legt man eine Spannung zwischen Daten-Elektrode und E2 an, fließt das Öltröpfchen wieder zurück.

Durch Verdecken eines Teils des Displays, beispielsweise über E2, wird der Tropfen nur sichtbar, wenn er über E1 steht. Einfallendes Licht wird dann vom Öltropfen in der jeweiligen Farbe reflektiert. Im aanderen Zustand wird das Licht durch eine reflektierende Schicht auf dem Substrat ohne Einfärbung direkt zum Betrachter zurückgeworfen.

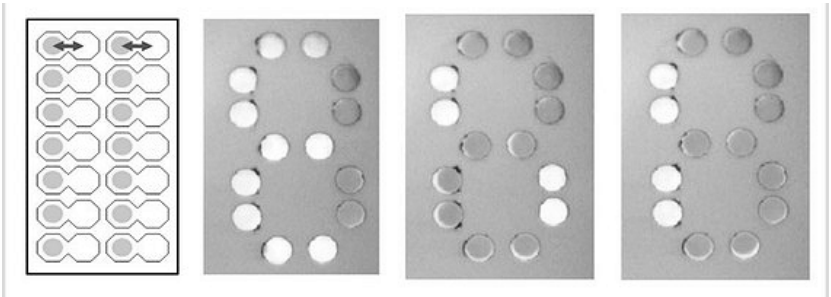

Bild 1.37: Pixellayout und Zahlendarstellung mittels Maske eines Ewetting-Displays (Pressebild von adt)

Der größte Vorteil dieses Prinzips liegt in der Bistabilität, Energie ist nur bei einer Änderung des Bildinhaltes nötig. Eine mechanische Barriere zwischen den beiden Kammern führt zu einer mechanisch bistabilen Tropfenposition, die ohne jeglichen Refresh auskommt. Die Pixelgröße kann zwischen 0,5 mm und 10 mm variieren. Momentan liegt die Zeit für das vollständige Verschieben eines Tropfens im Bereich von bis zu einer Sekunde, daher eignet sich diese Displaytechnik nur für Statusanzeigen (on/off) oder großflächige Informationsanzeigen.

Bild 1.37 zeigt ein bistabiles Electrowetting-Display mit 14 Pixeln, die nebeneinander in zwei Reihen mit jeweils sieben Zeilen angeordnet sind. Durch diese Anordnung lassen sich eine Siebensegment-Anzeige sowie Statusinformation oder zwei Bargraphen visualisieren. Für jedes Pixel findet die Bewegung der Tropfen bistabil von links nach rechts und umgekehrt in mehreren Schritten statt. Das gesamte Display wird mit sieben Zeilen- und acht Spaltenleitungen angesteuert.

Anhang

A.1 Literatur

- Dieter Zastrow: *Elektronik*, Vieweg-Verlag
- G. Koß, W. Reinhold, F. Hoppe: *Lehr- und Übungsbuch Elektronik*, Fachbuchverlag Leipzig
- U. Tietze, Ch. Schenk: *Halbleiter-Schaltungstechnik*, Springer-Verlag
- Helmut Lindner: *Taschenbuch der Elektrotechnik und Elektronik*, Hanser
- Lothar Sabrowsky: *Transistor-Schaltverstärker für beliebige Verwendung*, Franzis
- Siegfried Wirsum: *DC-Stromversorgung*, Pflaum
- E. Prohaska: *Digitaltechnik für Ingenieure*, Oldenbourg
- Ch. Siemers, A. Sikora: *Taschenbuch der Digitaltechnik*, Hanser
- Don Lancaster: *Das CMOS-Kochbuch*, VMI Buch AG
- Don Lancaster: *TTL-Cookbook*, Sams Publishing
- Hans-Dieter Stölting, Eberhard Kallenbach: *Handbuch Elektrische Kleinantriebe*, Hanser
- Elmar Schrüfer: *Elektrische Messtechnik*, Hanser
- Konrad Etschberger (Hrsg.): *CAN Controller Area Network*, Hanser
- *Zeitschrift Elektor*, Elektor-Verlag, Aachen
- *Elrad-Archiv 1977–1997 DVD*, eMedia GmbH, Hannover

A.2 Links

Das Elektronik-Kompendium: *http://www.elektronik-kompendium.de/*

Kabel- und Stecker-FAQ: *http://www.kabelfaq.de/*

Maxim-Datenblätter: *http://www.maxim-ic.com* und *http://datasheets.maxim-ic.com*

Datenblätter aller Art: *http://www.datasheets.org.uk/* und *http://www.alldatasheet.com*

Einführung in SPS: *http://www.studet.fh-muenster.de/~diefrie/einfh.html*

Stichwortverzeichnis