4.1 Peripherie ansprechen in C/LCD

4.1.1 Arten von LCDs

Für die Ausgabe numerischer Daten (Zahlen) und alphanumerischer Daten (Texte) durch den Mikrocontroller gibt es mehrere technische Lösungen[1]:

a) VFD-Anzeigen (*vacuum fluorescent displays*)

b) LED-Anzeigen (*light emitting diodes*)

c) LCD-Anzeigen (*liquid crystal displays*)

Die LCD-Anzeigen haben für portable Geräte den Vorteil der geringen Leistungsaufnahme (zumindest dann, wenn keine Hintergrundbeleuchtung ins Spiel kommt).

Unter ihnen (bei den anderen auch) gibt es wiederum verschiedene Arten, wie einzelne Anzeigeelemente zusammengesetzt werden (Form, Menge der Anzeigeelemente) und wie sie angesteuert werden. Es gibt zum Beispiel:

a) mehrstellige Siebensegment-Ziffern-Anzeigen, bekannt aus Uhren und Digitalmultimetern

b) mehrstellige Punkmatrix-Anzeigen zur Darstellung von Ziffern und Buchstaben

c) Grafik-Displays zur Darstellung von Bildern, Zeichen und Zahlen

Hier soll es um die zweite Variante gehen: Eine Punktmatrix-Anzeige für 2 Zeilen mit je 16 Zeichen.

Für die Ansteuerung (von Punktmatrix-Displays) existieren wieder mehrere Varianten:

a) Direktanschluss der Segmente bzw. Pixel. Bei LEDs muss ein Konstantstrom geliefert werden, bei VFDs eine Gitter- und eine Anodengleichspannung, bei LCDs eine Ansteuerspannung auf der einen Seite und eine Rechteckspannung auf der anderen Seite. Multiplexing muss man dann selbst vornehmen. Ein Mikrocontroller kann das natürlich, aber man braucht eine Reihe von Port-Anschlüssen.

b) Display-Controller mit Parallelanschluss. Die Versorgung der Segmente oder Pixel und das Multiplexing werden vom Display-Controller übernommen. Die Verbindung zum Rest der Welt läuft über ein kleines Bussystem.

c) Display-Controller mit RS232-Schnittstelle. Die Versorgung der Segmente oder Pixel und das Multiplexing werden vom Display-Controller übernommen. Die Verbindung zum Rest der Welt läuft über eine RS232-Schnittstelle.

d) Display-Controller mit ı2c-Schnittstelle. Die Versorgung der Segmente oder Pixel und das Multiplexing werden vom Display-Controller übernommen. Die Verbindung zum Rest der Welt läuft über den ı2c-Bus.

Hier wird die zweite Variante ausgewählt: Ein Controller vom Typ Hitachi HD44780 (oder kompatibel, z.B. Samsung KS0073) steuert die LCD-Pixel an und kommuniziert über ein Bussystem parallel mit dem Controller.

4.1.2 Verbindung des LCDs

Die meisten dieser Anzeigen verfügen über eine 14-polige Anschlussleiste (ausgeführt als einreihige oder doppelreihige Pfostenleiste), dazu eventuell noch zwei Anschlüsse für die Hintergrundbeleuchtung. Der Anschluss Nr. 1 ist meistens gesondert gekennzeichnet. Tabelle 1 zeigt die häufigste Anschlussbelegung. D0 bis D7 sind die gemeinsamen Daten- und Adressleitungen (Daten- und Adressbus), RS, R/W und E die Steuerleitungen (Steuerbus). Mit Vss (manchmal auch GND)

[1] Natürlich gibt es noch weitere: Nixie- und NIMO-Anzeigen sowie Bildröhren gelten als technisch veraltet, OLED-Anzeigen werden bezüglich ihrer Lebensdauer als noch nicht tauglich angesehen.

Pin	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Bedeutung	Vss	Vdd	Vee	RS	R/W	E	D0	D1	D2	D3	D4	D5	D6	D7

Tabelle 1: Übliche Belegung der LCD-Anschlüsse

ist der Masseanschluss gemeint ($U = 0\,\mathrm{V}$), mit Vdd der Plusanschluss der Betriebsspannung ($U = 5\,\mathrm{V}$). Vorsicht: Bei manchen LCDs sind Vss und Vdd vertauscht! An Anschluss 3 (Vee) wird die Kontrastspannung U_k angelegt, sie soll zwischen U_- und U_+ einstellbar sein (am besten mit einem $10\,\mathrm{k\Omega}$-Trimmpoti)[2].

Nun gibt es mehrere Arten, das LCD anzusteuern:

a) 8-Bit-Modus, bidirektional (D0–D7, mit R/W)

b) 8-Bit-Modus, unidirektional (D0–D7, ohne R/W)

c) 4-Bit-Modus, bidirektional (D4–D7, mit R/W)

d) 4-Bit-Modus, unidirektional (D4–D7, ohne R/W)

Der 4-Bit-Modus spart gegenüber dem 8-Bit-Modus vier Port-Bits und wird daher sehr oft benutzt. Deswegen wird er hier ausgewählt. Der unidirektionale und der bidirektionale Modus unterscheidet sich nur in der Benutzung einer Leitung (R/W). Wenn man sie mit anschließt, kann man sich aussuchen, welchen der beiden Modi man benutzt.

Nun müssen nur noch die Anschlüsse D4–D7, RS, R/W und E mit einem Port des Mikrocontrollers verbunden werden. *Leider* gibt es dafür keine Norm und jeder macht es anders. Hier soll eine bestimmte Verbindung festgelegt werden, dargestellt in Tabelle 2 und Abbildung 1.

LCD-Pin	1	2	3	4	5	6	11	12	13	14
Bedeutung	Vss	Vdd	Vee	RS	R/W	E	D4	D5	D6	D7
RN-Board-Pin	9	10	—	3	1	2	5	6	7	8
Bedeutung	—	—	—	PD2	PD0	PD1	PD4	PD5	PD6	PD7

Tabelle 2: Verbindung LCD–Mikrocontroller

4.1.3 Zeichen und Befehle zum LCD schicken

Zum LCD kann man Befehle und Zeichen schicken. Wie schickt man ein Zeichen zum LCD? Zuerst muss die E-Leitung (*enable*=Freigabe) auf 0 sein. Das ist sie standardmäßig. Nun legt man den ASCII-Code des Zeichens an den Datenbus und setzt RS auf 1 (Daten-Modus). Das Zeichen wird übernommen, wenn man jetzt E (*enable*, Freigabe) für mindestens eine viertel Mikrosekunde auf 1 bringt und danach wieder auf 0 setzt. Bei einem Befehl ist es genauso, nur muss hier RS auf 0 sein (Befehls-Modus).

Nach jedem Senden eines Zeichens oder eines Befehls muss man $t_w = 50\,\mu\mathrm{s}$ warten, bis das LCD wieder bereit ist. Bei den Befehlen CLEAR und RETURN HOME muss man etwas länger warten, nämlich $t_w = 1530\,\mu\mathrm{s}$[3].

4.1.4 Der 4-Bit-Modus

Im 4-Bit-Modus werden einfach zwei Halbbytes (*nibbles*) nacheinander zum LCD gesendet. Dabei wird das High-Nibble zuerst gesendet.

[2] Wie die Hintergrund-Beleuchtung angesteuert werden muss, ist von Typ zu Typ unterschiedlich und wird meistens im Datenblatt kurz angesprochen. Oft benötigt man einen Vorwiderstand, manchmal einen Inverter, manchmal eine Konstantstromquelle

[3] Durch Benutzung des R/W-Anschlusses kann man sich die Wartezeit auch sparen: Man fragt nach einem Schreibbefehl mit einem Lesezugriff solange das Busy-Flag ab, bis es 0 ist. Dann kann man wieder Schreiben.

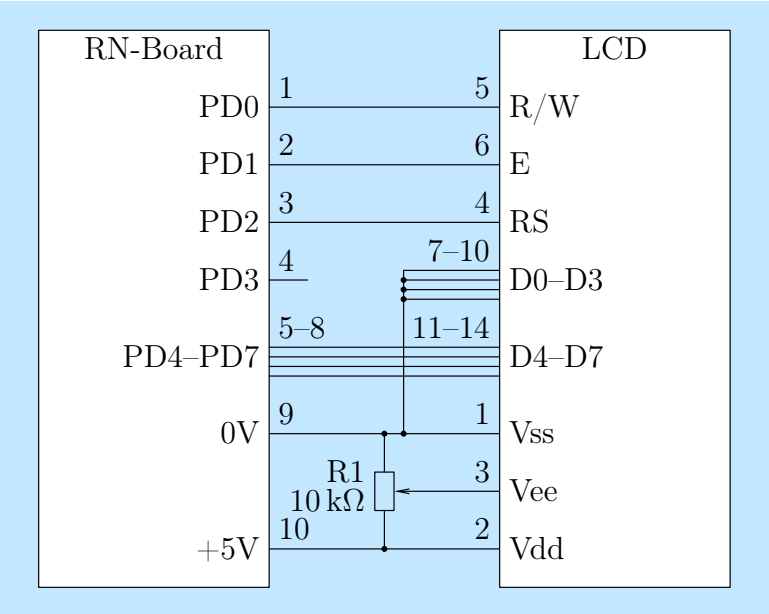

Abbildung 1: Verbindung LCD–Mikrocontroller

4.1.5 Initialisierung

Nach dem Start ist eine feste Befehlssequenz nötig. Ohne sie bleibt das Display leider leer. Für die meisten Module hilft folgende Sequenz:

a) 250 ms warten (je nach Modul auch weniger)

b) Befehl `0b0011` schicken (8-Bit-Modus einschalten)

c) 5 ms warten

d) Befehl `0b0011` schicken (8-Bit-Modus einschalten)

e) 5 ms warten

f) Befehl `0b0011` schicken (8-Bit-Modus einschalten)

g) 5 ms warten

h) Befehl `0b0010` schicken (4-Bit-Modus einschalten)

i) 5 ms warten

j) Befehl `0x28` in zwei Halbbytes schicken (4-Bit-Modus, 2 Zeilen, 5x7-Zeichen)

k) Befehl `0xC0` in zwei Halbbytes schicken (Display an, Cursor an, Blinken aus)

l) Befehl `0x04` in zwei Halbbytes schicken (Eingabemodus: von links, kein Shift)

4.1.6 Literatur und Web-Links

- Datenblatt des Controllers: `http://www.adafruit.com/datasheets/HD44780.pdf`
- Beschreibung mit Timing-Diagrammen: `http://sprut.de/electronic/lcd/index.htm`

4.1.7 Beispiel

Im folgenden Assembler-Beispiel wird das LCD initialisiert; anschließend wird der String `Test` ausgegeben. Quelle ist: `http://www.mikrocontroller.net`, AVR-Tutorial, abgewandelt für das benutzte Display.

```
1  .include "/usr/share/avra/m32def.inc"
2  .equ LCD_PORT = PORTC
3  .equ LCD_DDR  = DDRC
4  .equ RS   = PC2
5  .equ E    = PC1
6  ; Daten-Bits des LCDs ab PORTC, Bit 4 (PC4) - im Programmcode fest:
7  ; DB4 = PC4, DB5 = PC5, DB6 = PC6, DB7 = PC7
8
9                                  ; Stackpointer setzen
10         ldi r16, LOW(RAMEND)
11         out spl, r16
12         ldi r16, HIGH(RAMEND)
13         out sph, r16
14                                 ; Ausgabe-PORT benutzen
15         ldi r16, 0xff
16         out LCD_DDR, r16
17
18         rcall lcd_init
19         rcall lcd_clear
20         ldi r16, 'T'
21         rcall lcd_data
22         ldi r16, 'e'
23         rcall lcd_data
24         ldi r16, 's'
25         rcall lcd_data
26         ldi r16, 't'
27         rcall lcd_data
28 ende:
29         rjmp ende
30
31 lcd_data:
32         mov r17, r16
33         andi r16, 0xf0     ; High-Nibble
34         ori r16, 1<<RS     ; RS=1
35         out LCD_PORT, r16  ; Ausgabe an RS sowie DB4 bis DB7
36         rcall lcd_enable
37
38         andi r17, 0x0f     ; Low-Nibble
39         swap r17           ; Nibbles vertauschen statt 4x lsl
40         ori r17, 1<<RS     ; RS=1
41         out LCD_PORT, r17  ; Ausgabe an RS sowie DB4 bis DB7
42         rcall lcd_enable
43         rcall delay50us
44         ret
45
46 lcd_command:
47         mov r17, r16
48         andi r16, 0xf0     ; High-Nibble
49         out LCD_PORT, r16  ; Ausgabe an DB4-7
```

```
50          rcall lcd_enable
51
52          andi r17, 0x0f     ; Low-Nibble
53          swap r17           ; Nibbles vertauschen statt 4x lsl
54          out LCD_PORT, r17  ; Ausgabe an RS sowie DB4 bis DB7
55          rcall lcd_enable
56          rcall delay50us
57          ret
58
59  lcd_enable:
60          sbi LCD_PORT, E
61          nop
62          nop
63          nop
64          nop
65          nop
66          nop
67          nop
68          nop
69          nop
70          nop
71          nop
72          nop
73          cbi LCD_PORT, E
74          ret
75
76  delay50us:
77          ldi r16, 0xff
78  delay50us_:
79          nop
80          dec r16
81          brne delay50us_
82          ret
83
84  delay5ms:
85          ldi r16, 0x84
86  del1:   ldi r17, 0xc9
87  del2:   dec r17
88          brne del2
89          dec r16
90          brne del1
91          ret
92
93  lcd_init:
94          ldi r18, 50          ; 250 ms warten
95  powerupwait:
96          rcall delay5ms
97          dec r18
98          brne powerupwait
99          ldi r16, 0x3<<4      ; Reset/Init. im 8-Bit-Modus
100         out LCD_PORT, r16
101         rcall lcd_enable     ; 1. Mal
102         rcall delay5ms
103         rcall lcd_enable     ; 2. Mal
```

```
104         rcall delay5ms
105         rcall lcd_enable      ; 3. Mal
106         rcall delay5ms
107
108         ldi r16, 0x2<<4       ; 4-Bit-Modus
109         out LCD_PORT, r16
110         rcall lcd_enable
111         rcall delay5ms
112
113         ldi r16, 0b00101000 ; 0x28
114         rcall lcd_command
115
116         ldi r16, 0b00001100 ; 0x0c
117         rcall lcd_command
118
119         ldi r16, 0b00000100 ; 0x04
120         rcall lcd_command
121         ret
122
123 lcd_clear:
124         ldi r16, 1
125         rcall lcd_command
126         rcall delay5ms
127         ret
```