LCD-Ansteuerung mit MikroForth

Zu unserer Attiny-Platine wird standardmäßig ein kleines LCD von der Firma Pollin beigelegt. Dieses ist auf eine kleine Platine gelötet, welches sich über eine Steckerleiste mit der PortB-Buchsenleiste der Attiny-Platine verbinden lässt.

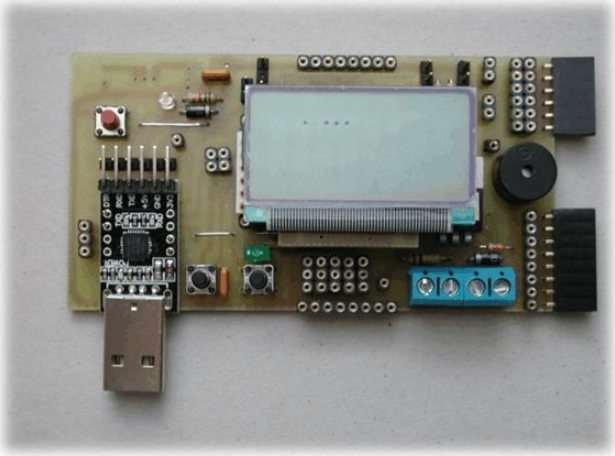

Abbildung 1

Das Pollin-LCD-Modul besitzt als Controller den Hitachi 44780. Dieser Baustein kommt bei sehr vielen Displays zum Einsatz; er gilt als Quasi-Standard.

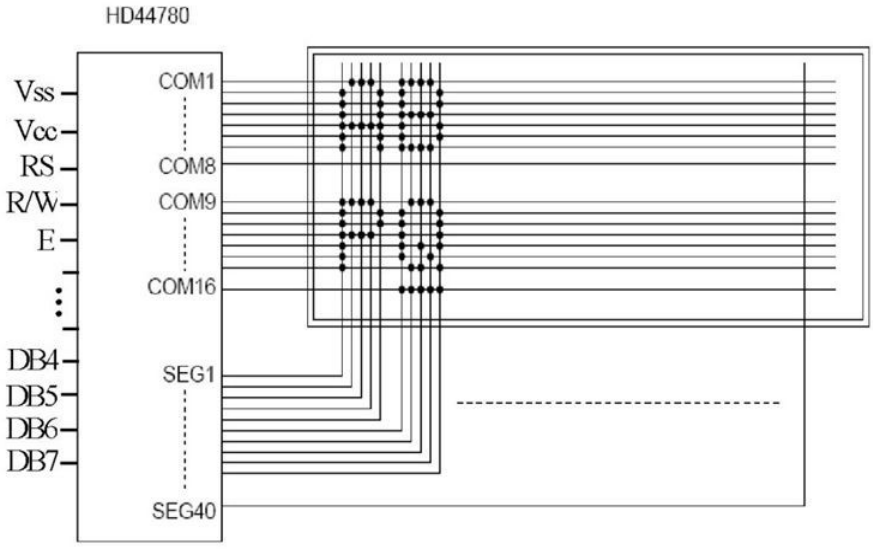

Abbildung 2

Die Anschlüsse auf der linken Seite dienen zur Stromversorgung und Ansteuerung des Displays. Die Bedeutung der einzelnen Anschlüsse sind in der folgenden Tabelle angegeben:

Vss	Anschluss an Masse
Vcc	Anschluss an + 5 V
RS	Wenn RS = 0 ist, dann werden die übertragen Daten als Befehle (instructions) gedeutet, wenn RS = 0 ist, dann werden sie als Codes von darzustellenden Zeichen (data) gedeutet.
R/W	(genauer: $\mathbf{R/\overline{W}}$) Wenn dieses Bit 0 ist, erfolgt ein Schreibvorgang; ist es 1, erfolgt ein Lesevorgang; wir schließen ihn an Masse an (s. u.)
DB0 ... DB7	Über diese Anschlüsse werden die Daten (Befehle oder Zeichen) geschrieben oder gelesen. Die Daten können wahlweise byteweise oder auch nibbleweise (1 Nibble = 4 Bit) übertragen werden. Wir benutzen hier die letztere der beiden Möglichkeiten (s. u.).
E	Über diesen Anschluss takten wir den Informationsfluss, ähnlich wie z. B. bei I2C (s. u.).

An PortB stehen uns insgesamt 8 Bits zur Verfügung; zwei davon sollen für den I2C-Bus (an PortB. 5 und PortB.7) frei gehalten werden. Damit bleiben 6 Bits übrig. RS und E sind zur Steuerung der Kommunikation unverzichtbar; wir schließen sie an PortB.6 bzw. PortB.4 an. Wenn wir auf Lesevorgänge verzichten, können wir R/W direkt an Masse anschließen und benötigen dafür also keinen weiteren Anschluss am Attiny. Die restlichen 4 Anschlüsse von PortB (PortB.0 ... PortB.3) schließen wir an DB4 ... DB7 an.

Wie wird nun ein Byte übertragen? Die Abb. 3 macht dies deutlich; hier wird gezeigt, wie ein Befehls-Byte mit den einzelnen Bits IR0 ... IR7 übertragen wird. Zunächst wird das höherwertige Nibble übertragen, dann das niederwertige. Das E-Bit stellt ein Clock-Signal bereit.

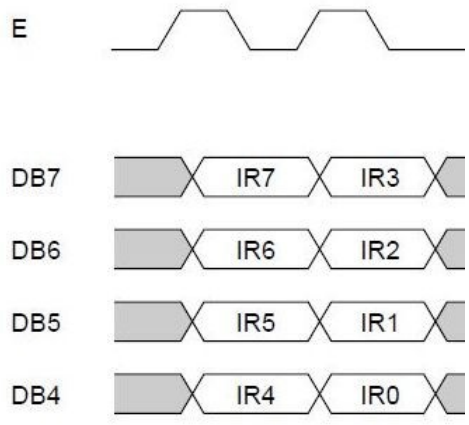

Abbildung 3

Dieses Clock-Signal muss nicht schon vor dem Anlegen der einzelnen Nibbles auf 1 gesetzt worden sein; entscheidend ist vielmehr, dass das es von 1 auf 0 wechselt, wenn die Nibbles bereits anliegen; denn durch diese negative Flanke wird die Verarbeitung dieser Nibbles durch den Hitachi-Controller gestartet. Diese Verarbeitung dauert eine gewisse Zeit; währenddessen kann das Display (genauer der Hitachi-Controller) keine weiteren Befehle/Daten entgegennehmen. Das Display meldet das Ende der Verarbeitungsphase, indem es ein Busy-Flag ausgibt. Da auf unserer Pollin-Platine der R/W-Anschluss fest mit Masse verbunden ist, können wir diese Information mit der Attiny-Platine nicht auslesen. Im Manual des Hitachi-Controllers findet man aber Zeitangaben für die Verarbeitungsphasen. Meist beträgt sie ca. 37 us. Weitere Informationen s. u.

In der folgenden Abb. 4 ist zur Verdeutlichung der BASCOM-Code für die Übertragung eines Zeichens mit dem ASCII-Code z angegeben.

```
Sub Zeichen2lcd(z As Byte)
'Zeichen senden; der Spaltenzeiger wird automatisch um 1 erhöht
  Hnibble = Z AND &B11110000
  Lnibble = Z AND &B00001111
  Shift Hnibble , Right , 4
  PORTB = Hnibble                                    'High-Nibble
  PORTB.6 = 1                                        'RS=1 (zeichen)
  Call Clock
  PORTB = Lnibble                                    'Low-Nibble
  PORTB.6 = 1                                        'RS=1 (Zeichen)
  Call Clock
  PORTB.6 = 0                                        'RS=0
  Waitus Pause                                       '40 us
End Sub

Sub Clock
  PORTB.4 = 1                                        'E=1
  Waitus 1
  PORTB.4 = 0                                        'E=0
End Sub
```

Abbildung 4

Die Übertragung eines Befehls sieht genauso aus; es fehlen hier lediglich die Befehle für das RS-Bit.

Auf zwei spezielle Befehle wollen wir hier noch kurz eingehen: den Befehl zum Löschen des Displays und den zum Positionieren des Cursors.

Zum Löschen wird das Befehlsbyte &B00000001 = 1 benutzt. Hierdurch wird der Inhalt des Displays gelöscht und der Cursor auf die Startposition oben links gesetzt. Bei der von uns gewählten Initialisierung (s. u.) wird übrigens der Cursor nicht auf dem Display angezeigt.

Das Befehlsbyte für das Positionieren des Cursors muss aus der Zeilen- und aus der Spaltennummer berechnet werden. Dabei kann man sich an der Tabelle aus Abb. 5 orientieren. Bei den dort angegebenen Zahlen in den Zellen handelt es sich um die Hex-Adressen der einzelnen Cursorpositionen. Um den Cursor zur Zelle mit der Adresse 45 zu bewegen, muss das Befehlsbyte $80 + $45 = $80 + $40 + $5 = 128 + 64 + 5 = 177 benutzt werden.

Display position	1	2	3	4	5	6	7	8
DDRAM address	00	01	02	03	04	05	06	07
	40	41	42	43	44	45	46	47

Abbildung 5

Bevor nun die ersten Befehle oder Zeichen übertragen werden, muss das Display initialisiert werden. Insbesondere muss deutlich gemacht werden, dass die Übertragungen im Nibble-Modus und dass das Display mit 5*8-Zeichenmatrix arbeitet und 2 Zeilen besitzt. Übersichtlich sind die erforderlichen Schritte dazu in Abb. 24 des Manuals mit kurzen Erläuterungen dargestellt. Mit den Werten N = 1 (2 Zeilen), F = 0 (5*8-Zeichenmatrix), I/D = 1 (Auto-Inkrement des Positionszeigers nach dem Schreiben eines Zeichens), S = 0 (kein Shift) müssen wir demnach folgende Befehls-Nibbles (RS = 0) übertragen und dabei auf die angegebenen Wartezeiten achten:

	Einschalten	mehr als 15 ms nach Erreichen der Spannung 4,5 V
0011		mehr als 4,1 ms
0011		mehr als 100 us
0011		mehr als 37 us
0010		mehr als 37 us
0010 1000	4-bit-Operation, 2-Zeilen-Display mit 5*8-Zeichen	mehr als 37 us
0000 1100	Display an, Cursor aus	mehr als 37 us
0000 0001	LCD löschen	mehr als 37 us
0000 0110	Auto-Inkrement bei Cursor-Position	mehr als 37 us

Unter http://www.g-heinrichs.de/attiny/forth/lcd.zip finden Sie ein BASCOM-Programm, welches zeigt, wie man allein mit der Übertragung von Steuer- und Zeichenbytes Texte auf dem LCD anzeigen, löschen oder auch den Cursor positionieren kann; es diente mir hauptsächlich zum Austesten.

Bei unserem Forth-Compiler (http://www.g-heinrichs.de/wordpress/index.php/attiny/forth/) gibt es keine LCD-Befehle im Grundvokabular. Hier stellen die oben vorgestellten Vorgehensweisen die einzige Möglichkeit dar, Zeichen auf dem LCD anzuzeigen. Der folgende Quelltext gibt einige grundlegende Forth-Wörter an, mit denen Zeichen und Zahlen auf dem LCD dargestellt werden können (Quellcode unter http://www.g-heinrichs.de/attiny/forth/lcd.zip). In dieser Datei findet man auch eine Vokabulardatei `forthvoc.vok`, die bereits diese LCD-Wörter enthält. Soll Ihr Forth-Compiler mit dieser erweiterten Vokabulardatei arbeiten, müssen sie die bisher benutzte Vokabulardatei in dem Verzeichnis des Forth-Compilers durch die neue Vokabulardatei ersetzen.

```
( LCD-Test: gibt "ABC Z" in der 1. Zeile und "#" in der 2. Zeile auf Pollin-Display aus, blinkt )
: lcd_RS 6 swap outPortB ;
: lcd_E 4 swap outPortB ;
: lcd_Clock 1 lcd_E 0 lcd_E ;
: lcd_out 16 / swap . 1 lcd_RS lcd_Clock . 1 lcd_RS lcd_Clock 0 lcd_RS ;
: lcd_code . lcd_clock ;
: lcd_clear 0 lcd_code 1 lcd_code 2 waitms ;
: lcd_cursor 64 * + 128 + 16 / swap lcd_code lcd_code 1 waitms ;
: lcd_digit 48 + lcd_out ;
: lcd_number 100 / swap lcd_digit 10 / swap lcd_digit lcd_digit ;
: lcd_init1 3 lcd_code 5 waitms 3 lcd_code wait1ms 3 lcd_code wait1ms ;
: lcd_init2 2 lcd_code wait1ms 2 lcd_code 8 lcd_code wait1ms 0 lcd_code 12 lcd_code wait1ms 0 lcd_code 1 lcd_code wait1ms 0 lcd_code 6 lcd_code wait1ms ;
: lcd_init lcd_init1 lcd_init2 ;

: textausgabe  65 lcd_out 66 lcd_out 67 lcd_out 32 lcd_out 90 lcd_out 3 1 lcd_cursor 35 lcd_out ;
: main lcd_init begin textausgabe 2 wait lcd_clear 1 wait 0 until end ;
```

Die entscheidenden Wörter sind:

Wort	Kommentar	Stack
`lcd_RS`	setzt das RS-Bit beim Display auf n; n muss 0 oder 1 sein	(n –)
`lcd_clock`	erzeugt ein Clock-Signal für das Display	(–)
`lcd_out`	gibt das Zeichen mit dem Code n an der aktuellen Cursorposition auf dem Display aus	(n –)
`lcd_init`	initialisiert das Display	(–)
`lcd_clear`	löscht das Display	(–)
`lcd_number`	gibt die Zahl n (dezimal) an der aktuellen Cursorposition auf dem Display aus	(n –)
`lcd_cursor`	setzt den Cursor in die Position “Spalte x, Zeile y”; dabei beginnt die Zählung mit 0	(x y –)

Quellen

https://cdn-shop.adafruit.com/datasheets/HD44780.pdf
https://www.mikrocontroller.net/articles/AVR-GCC-Tutorial/LCD-Ansteuerung