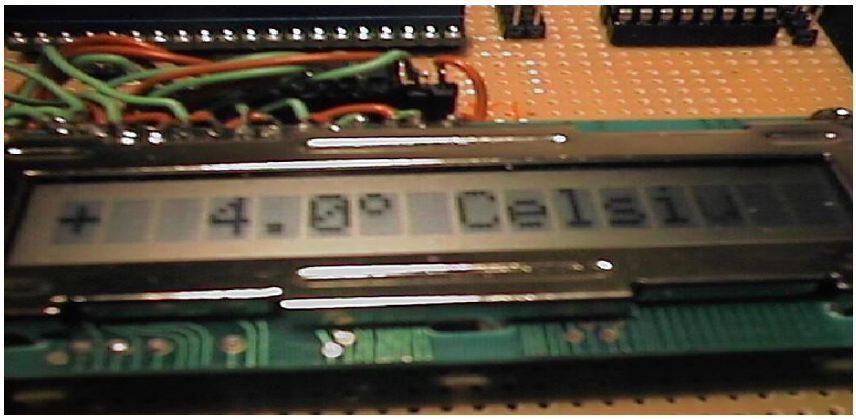
LC – Display



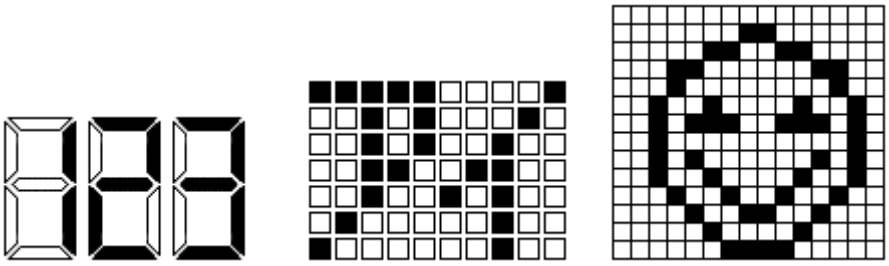
Zumindest habe ich in der Überschrift keinen Fehler gemacht. Immer wieder liest und hört man den Ausdruck LCD – Display oder LCD – Anzeige. Ich wurde schon des öfteren von Zuschauern per Telefon belehrt: „Das heißt doch nicht **L**iquid **C**rystal **D**isplay **D**isplay“. Mittlerweile habe ich es aber kapiert.

Bei meinen Experimenten habe ich lange auf die Verwendung einer externen Ausgabe verzichtet, weil ich eigentlich nicht wußte, was ich damit anfangen sollte. Eine Computerausgabe erschien mir immer ausreichend – und wenn das Projekt fertig war, so lief es ohne Werte auszugeben.

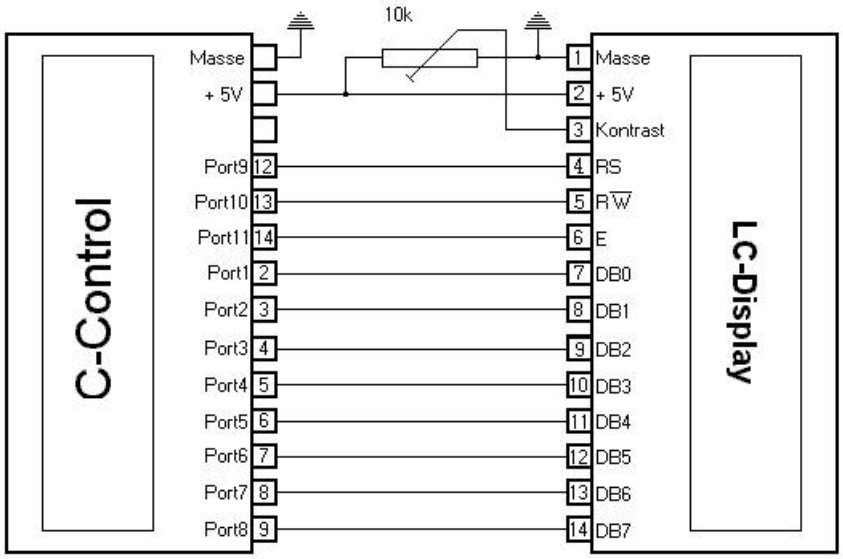
Doch offensichtlich ist es ein beliebter Zusatz, der bei vielen Experimenten zu finden ist. Als mir ein Faxgerät durch übelste Gerüche aus dem Netztrafo unangenehm auffiel, machte ich ein Schlachtfest und neben den Schrittmotoren fiel mir auch ein einzeiliges LC – Display mit 16 Stellen in die Hand. Jetzt wollte ich es doch genauer wissen und begann ohne Beschreibung damit, irgend etwas zur Anzeige zu bringen.

Das Internet war – wie meistens – behilflich. Ein Suchauftrag mit 'LCD' brachte gleich die richtigen Seiten. Das abgebildete Assemblerprogramm verschaffte mir dann die Weisheiten, die nötig sind, um die Ansteuerung der einzelnen Zeichen zu erreichen.

Es gibt die verschiedensten Ausführungen. Eine kleine Auswahl ist hier unten abgebildet.



1. Segment system 2. Dot matrix system (character display) 3. Dot matrix system (graphics display)



Zunächst einmal stellt man fest, dass 11 Digitalports benötigt werden. Mit dem normalen Lallussystem ist dies nicht zu realisieren, da hier nur noch ein einzelner Digitalport für Experimentieraufgaben zur Verfügung steht.

Da aber noch ein Experimentierboard mit frei zugänglichen Pins zur Verfügung steht, konnte das Basteln beginnen. 11 Ports klingt natürlich sehr viel. Doch wenn man bedenkt, dass ein Zeichen aus 5 * 8 Punkten besteht und damit 40 * 16 = 640 Punkte angesteuert werden müssen, ist dies verschwindend gering. Die meiste Arbeit übernimmt daher der LCD – Controller, von denen es wiederum verschiedene Typen gibt.

An 8 von 11 Ports werden die Daten des Zeichens angelegt, das dargestellt werden soll. Streng nach ASCII – Code. Eine 65 auf dem Byteport, also 01000001, zeichnet ein großes A usw.

Die restlichen 3 Leitungen sind mit RS, R/W und EN bezeichnet. Sie übernehmen die Steuerung.

R/W (read / write) setzt das Register entweder auf Schreiben oder Lesen. Eine 0 steht für Schreiben, eine 1 entsprechend für Lesen.

RS (register select) legt die Art der Übermittlung in Kombination mit R/W fest.

R/W = 0 : RS = 1 – Ein Zeichen soll in die Anzeige geschrieben werden.
R/W = 0 : RS = 0 – Ein Steuerzeichen (Anzeige löschen z.B.) wird übertragen.
R/W = 1 : RS = 1 – Ein Zeichen soll zurückgelesen werden.
R/W = 1 : RS = 0 – Kann ein neues Zeichen übertragen werden (busy)?

An den Kombinationen erkennen wir bereits, dass wir einen Port einsparen können. Da wir zunächst (und später wahrscheinlich ebenfalls) nur Zeichen schreiben wollen und dieses mit dem langsamen Basic unternehmen (keine busy – Abfrage), kann R/W fest auf Masse gelegt werden.

E (enable) ist das Signal für die Übernahme der Daten in das LC – Display. Hier genügt ein kurzer Impuls mit `pulse en`. In dem Programm wird noch RW als Digitalport geführt. Wenn man weitere Experimente vorhat, so kann damit gespielt werden. Will man nur schreiben, so kann er entfallen.

Das Programmieren kann beginnen.

```
define Daten   byteport[1]
define RS      port[9]
define RW      port[10]
define E       port[11]
define zeichen byte

E  = 0
RW = 0
RS = 0 : Daten = &H0E : pulse E
RS = 0 : Daten = &H01 : pulse E

RS = 1 : Daten = 65 : pulse E
RS = 1 : Daten = 66 : pulse E
```

Dieses einfache Programm sollte bereits 'AB' auf das Display schreiben, der Cursor steht rechts neben dem B, er blinkt nicht.

Zunächst wurde mit dem Kommando (bei RS = 0) &H0E das Display eingeschaltet, der Cursor ist sichtbar und blinkt nicht. Mit dem Kommando &H01 in der nächsten Zeile wird der Inhalt des Displays gelöscht und der Cursor an den Anfang der Zeile gestellt. Jetzt kann man schon andere Funktionen mit verschiedenen Kommandos aufrufen.

Mit &H0F (statt &H0E) wird das Display eingeschaltet und ein blinkender Cursor erzeugt.
Mit &H02 (statt &H01) wird der Cursor an den Anfang gestellt, ohne den Inhalt zu löschen. Um dieses zu testen, sollten Sie die folgenden Zeilen mit einem Hochkomma kurz 'ausRemen'.

Mit dem Kommando &H80 + position stellt man eine Curorposition ein, die das gesendete Zeichen einnehmen soll. Die erste Position ist 0. Bei dem unteren Beispiel beginnt das A auf Position 4.

```
RS = 0 : Daten = &H0E : pulse E
RS = 0 : Daten = &H01 : pulse E
RS = 0 : Daten = &H83 : pulse E

RS = 1 : Daten = 65 : pulse E
RS = 1 : Daten = 66 : pulse E
```

Mit einer kleinen Erweiterung des Programmes können bereits die Daten der seriellen Schnittstelle (über das Terminalprogramm eingegeben) dargestellt werden.

```
RS = 0 : Daten = &H0E : pulse E
RS = 0 : Daten = &H01 : pulse E

#main
get zeichen
RS = 1 : Daten = zeichen : pulse E
Goto main
```

Wir stellen jetzt fest, dass das Display nicht voll beschrieben wird. Die rechte Hälfte oder die zweite Zeile ist verschwunden. Zum Einschalten gibt es wieder einen Befehl: Bit4 (32) ist an, display on (16) ist an, 2 line display (8) ist an. Das Kommando lautet also: 32+16+8=56 oder &H56.

```
RS = 0 : Daten = &H38 : pulse E
```

Fügen wir dieses Kommando in die Initialisierung ein, so wird man erkennen, dass beide Zeilen eingeschaltet sind. Aber es stört noch etwas. Die Anzeige der Zeichen wird nur bis zur Mitte dargestellt, danach passiert nichts mehr. Doch!, wenn man lange genug Zeichen eingibt. Plötzlich werden die neuen Zeichen wieder vorne im Display angezeigt. Zum Erreichen der zweiten Zeile muss ein Offset von 66 oder &H40 eingefügt werden. Gibt man z.B. als Kommando die Zeile ein:

```
RS = 0 : Daten = &H80+&H40 : pulse E
```

Somit werden die Zeichen jetzt in der zweiten Hälfte dargestellt. Man muss das Programm so abändern, dass dieser Positionswechsel automatisch geschieht.

```
define Daten    byteport[1]
define RS       port[9]     ' register select
define RW       port[10]    ' read/write
define E        port[11]    ' enable
define zeichen  byte        ' was wurde gesendet?
define stelle   byte        ' wo ist der Cursor?
define zeile    bit[192]    ' erste oder zweite Zeile?

E  = 0
RW = 0

RS = 0 : Daten = &H0E : pulse E ' displ. on, cursor on
RS = 0 : Daten = &H01 : pulse E ' displ. clear
RS = 0 : Daten = &H38 : pulse E ' zwei Zeilen

#main
get zeichen                              ' serielle?
if stelle = 8 then gosub zeilen_wechsel ' wechsle Zeile
RS = 1 : Daten = zeichen : pulse E       ' schreibe das Zeichen
stelle = stelle + 1                      ' Cursorposition
Goto main

#zeilen_wechsel
zeile = not zeile                        ' toggle Variablenbit
daten = &H80                             ' erste Zeile
if zeile then daten = &H80 + &H40        ' zweite Zeile
RS = 0 : pulse E                         ' uebertrage Kommando
Stelle = 0                               ' Cursor auf 0
Return
```

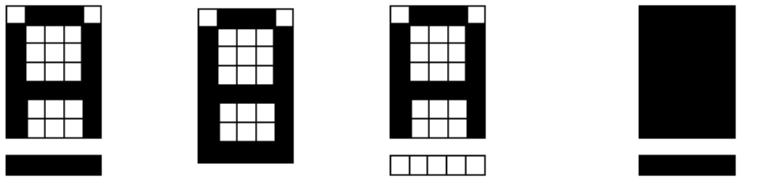
Mit diesem kleinen Programm kann man jetzt (bei geöffnetem Terminalprogramm) über die Tastatur beliebige ASCII – Zeichen eingeben. Am Ende der zweiten Zeile erfolgt ein Umsprung auf die erste Zeile. Mit dem Controller lassen sich noch weitere Spezialitäten ausarbeiten. Man kann die Cursoranzeige ausschalten, den Cursor zum Blinken veranlassen, den Cursor von links nach rechts oder von rechts nach links bewegen.

In den meisten Fällen kann man eigene Zeichen im RAM ablegen. Die zugehörigen Kommandos sind in der folgenden Tabelle abgebildet.

Instruction	RS	R/W	B7	B6	B5	B4	B3	B2	B1	B0	Beschreibung	Execution Time

												(when F_{cp} or f_{osc} is 250KHz)
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears Display and returns cursor to the Home Position (Address 00)	80uS = 1.64mS
Return Home	0	0	0	0	0	0	0	0	1	*	Returns cursor to Home Position. Returns shifted display to original position. Does not clear display	40uS = 1.6mS
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets DD RAM counter to increment or decrement (I/D) Specifies cursor or display shift during to Data Read or Write (S)	40uS
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Sets Display ON/OFF (D), cursor ON/OFF (C), and blink character at cursor position	40uS
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves cursor or shifts the display w/o changing DD RAM contents	40uS
Function Set	0	0	0	0	1	DL	N	F	*	*	Sets data bus length (DL), # of display lines (N), and character font (F)	40uS
Set CG RAM Address	0	0	0	1	A_{CG}						Sets CG RAM address. CG RAM data is sent and received after this instruction	40uS
Set DD RAM Address	0	0	1	A_{DD}							Sets DD RAM address. DD RAM data is sent and received after this instruction	40uS
Read Busy Flag & Address	0	1	BF	AC							Reads Busy Flag (BF) and address counter contents	1uS
SIZE=2>Write Data from DD or CG RAM	1	0	Write Data								Writes data to DD or CG RAM and increments or decrements address counter (AC)	40uS
Read Data from DD or CG RAM	1	1	Read Data								Reads data from DD or CG RAM and increments or decrements address counter (AC)	40uS

I/D=1: Increment S=1: Display Shift on data entry S/C=1: Display Shift (RAM unchanged) R/L=1: Shift to the Right DL=1: 8 bits N=1: 2 Lines F=1: 5x10 Dot Font D=1: Display ON C=1: Cursor ON B=1: Blink ON BF=1: Cannot accept instruction	I/D=0: Decrements S=0: Cursor Shift on data entry S/C=0: Cursor Shift (RAM unchanged) R/L=0: Shift to the Left DL=0: 4 bits N=0: 1 Line F=0: 5x7 Dot Font D=0: Display OFF C=0: Cursor OFF B=0: Blink OFF BF=0: Can accept instruction	Definitions: DD RAM: Display data RAM CG RAM: Character generator RAM A_{CG}: CG RAM Address A_{DD}: DD RAM Address(Cursor Address) AC: Address Counter used for both DD and CG RAM Address * Don't Care	Execution Time changes when Frequency changes per the following example: If F_{CP} or f_{osc} is 27 KHz 40uS x 250/270 = 37uS



Wofür benutzt man ein externes Display? Zum Beispiel zur Anzeige von Daten eines Analogsensors. Will man Temperaturen mit 0,5 er Auflösung darstellen, so muss die Programmierung schon ausgefeilt sein, da die CControl keine Fließkommaarithmetik kennt.

Deshalb soll an dieser Stelle noch ein Beispielprogramm gezeigt werden, das mit Kommazahlen umgehen kann. Es kann natürlich auch für ähnliche Zwecke benutzt werden. Hier werden Zufallszahlen im Bereich von 0 bis 255 erzeugt.

```
define daten   byteport [ 1]
define RS      port     [ 9]
define RW      port     [10]
define E       port     [11]
define zahl    word          ' Eingabe
define werte   byte          ' Merker
define i       byte          ' Zaehler

' --- Initialisieren -------------

RW = 0                ' read/write
E  = 0                ' Enable - Impuls
RS = 0                ' register select

daten = &H38 : pulse E ' Initialisierung 2 Zeilen
daten = &H0C : pulse E ' Display einschalten, ohne Cursor
daten = &H01 : pulse E ' Display clear, Cursor HOME

#main
randomize timer
zahl=rand and 255 ' irgendeine Zahl zwischen 0 und 255
gosub anzeige
pause 100
goto main

#anzeige
i = 0
RS = 0 : daten= &H01 : pulse E ' Anzeige loeschen
if zahl < 0 then werte = 45 else werte = 43 : gosub senden '-/+
zahl = abs(zahl)
werte = (zahl/1000)+48
if werte = 48 then i = 1
if werte = 48 then werte = 32 : gosub senden
if zahl < 1000 then werte=(zahl/100)+48 else werte = (zahl-
1000)/100+48
```

```
if werte = 48 and i = 1 then werte = 32 : gosub senden
werte = (zahl mod 100) / 10 + 48 : gosub senden
werte = 46 : gosub senden
werte = (zahl mod 100) mod 10 + 48 : gosub senden
for i = 0 to 8
if i = 2 then gosub position   ' zweite Zeile einschalten
looktab celsius,i,werte        ' Tabelle lesen
RS = 1 :daten = werte : pulse E
next i
return

#senden
RS = 1 :daten = werte : pulse E
return

#position
RS = 0 : daten = 64 + &H80 : pulse E
return

table celsius
32 39 67 101 108 115 105 117 115
tabend
```

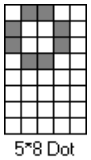
Wenn das Programm gestartet wird, so sieht man eine etwas komische Gradanzeige. Sie wird mit einem Hochkomma dargestellt. Der Zeichensatz enthält kein ° - Zeichen. Das störte mich gewaltig und deshalb erweiterte ich den Zeichensatz mit einer eigenen Gradanzeige. Das CG – RAM hat zwei Bereiche, in die man eigene Dots schreiben kann. Wir begnügen uns hier mit dem CG – RAM (DD – RAM wäre der zweite Bereich). Hier können 8 Zeichen abgelegt werden. Sie liegen dann später von 0 – 7. Statt der Eingabe des ASCII – Wertes können dann einfach diese selbst erstellten Zeichen aufgerufen werden.

Zuerst muß das Kommando 'Set CG – RAM Adress' eingeleitet werden:

```
RS = 0 : daten = 64 : pulse E ' Kommando für erste Stelle 0
RS = 0 : daten = 64 + 8 : pulse E ' Kommando für zweite Stelle 1
```

Jetzt werden die Daten für das Bitmuster übertragen, das dargestellt werden soll.

Das Gradzeichen soll so aussehen, wie abgebildet.


5*8 Dot

Eine 1 schaltet das Dot an, eine 0 löscht entsprechend das Dot. Die ersten Zeilen sehen dann binär so aus:

```
&B 0000110 0
&B 0001001 0
&B 0001001 0
&B 0000110 0
&B 00000000
```

```
&B 00000000
&B 00000000
&B 00000000
```

Wenn man sich das Muster genauer anschaut, so kann man das Gradzeichen erkennen. Das Programm dazu ist jetzt relativ einfach zu schreiben.

```
define Daten   byteport[1]
define RS      port[9]
define RW      port[10]
define E       port[11]
define i byte
define j byte
define zeichen byte

E  = 0 : RW = 0
RS = 0 : Daten= &H0C : pulse E ' Anzeige ein, ohne Cursor
RS = 0 : Daten = 64 : pulse E '
RS = 1

for i = 0 to 7
looktab grad,i,daten
pulse E
next i

RS = 0 : Daten = &H01 : pulse E ' Löschen und HOME
RS = 1 : Daten = 0    : pulse E ' 0. Stelle anzeigen
end

table grad &B00001100 &B00010010 &B00010010 &B00001100 0 0 0 0
tabend
```

Fügt man die Tabelle und die Initialisierungsroutine für das Bitmuster in obiges Temperaturprogramm ein, so hat man eine ordentliche Anzeige. Man muß darauf achten, dass die 39 in der Tabelle (für Hochkomma) jetzt mit einer 0 ersetzt wird.

Das Elektronikhobby sollte vor allem Spaß bereiten. Wenn ich meinen Mitarbeitern wieder das Bastelergebnis eines langen Abends vorstelle, so setzen sie immer eins drauf, wohl in der Hoffnung, dass mein Programmiervermögen versagt. Als ich die LC – Anzeige kurz vorführte, kam sofort wieder die Sonderaufgabe: „Wie sieht es denn mit einer Laufschrift aus – und zwar von links nach rechts mit deinem Vornamen, der hat doch genau 8 Buchstaben?“

Ich mußte ein paar Minuten überlegen und kam dann auf die Idee, den Shiftbefehl zu benutzen, mit dem man den Displayinhalt komplett nach rechts oder links um eine Stelle verschieben kann.

In der Tabelle liest man: Shift = Bit4 (16) immer an, Bit3 (Display Shift ein) (8), Bit2 nach links (0), also 24 oder &H18. Der Name wird aus der Tabelle mit den ASCII – Zeichen gelesen.

Zunächst aber wird das Display ausgeschaltet, damit mein Trick nicht sichtbar wurde. Damit das Verschieben des Namens beginnen kann, muß das Display vollgeschrieben sein. Dieses geschieht in der ersten Schleife ohne Anzeige und ohne Pause, damit es schnell geht. Mit dem AND – Befehl werden die Tabellenaufrufe von 0-7 ausmaskiert.

Danach wird die Anzeige eingeschaltet und der Inhalt des Controller – RAM’s immer um eine Position nach links geschoben. Fügt man den Shiftbefehl 'rechts' ein (statt 24 wird 28 (&H1C)

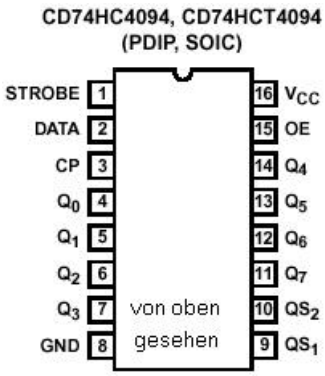
gesendet), so läuft das Ganze natürlich anders herum. Zum Glück waren die Mitarbeiter damit zufrieden und nörgelten nicht daran herum, dass der Name nicht alleine steht.

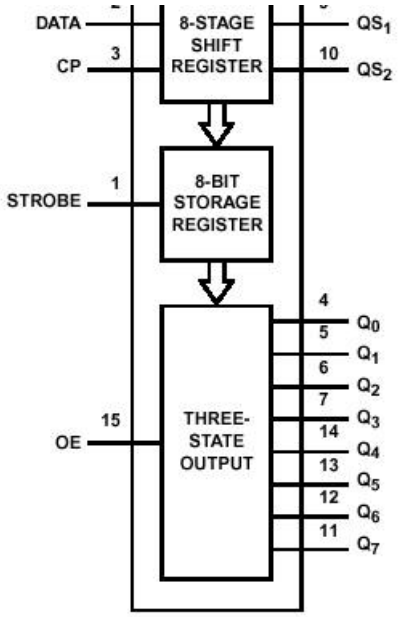
```
define Daten   byteport[1]
define RS      port[9]
define RW      port[10]
define E       port[11]
define i       byte
define zeichen byte

E =0
rw=0
RS = 0 : Daten= &H38 : pulse E ' zwei Zeilen

RS = 0 : Daten= &H08 : pulse E ' Display aus
RS = 0 : Daten= &H01 : pulse E ' Anzeige loeschen

for i=0 to 79
looktab namen,i and 7,zeichen
RS = 1 : Daten= zeichen : pulse E , Daten schreiben
next i

RS = 0 : Daten= &H0C : pulse E ' Display ein, ohne Cursor

#main
pause 10                      ' Pause fuer Anzeige
RS = 0 : Daten= &H18: pulse E ' Shift nach links
goto main

table namen 87 111 108 102 103 97 110 103 ' ASCII: Wolfgang
tabend
```

Am Anfang dieses Kapitels wurde bereits erwähnt, dass die Ansteuerung eines LC – Moduls eine ‚portfressende' Angelegenheit ist. Fast alle Digitalports der CControl werden verbraucht. Das muß auch besser gehen, zumal das Display evtl. im Lallussystem eingeklinkt sein soll. Dann muß alles über die beiden Leitungen SDA und SCL des I2C – Busses gesteuert werden. Zunächst einmal eine Lösung, die mit 3 Ports zu realisieren ist .

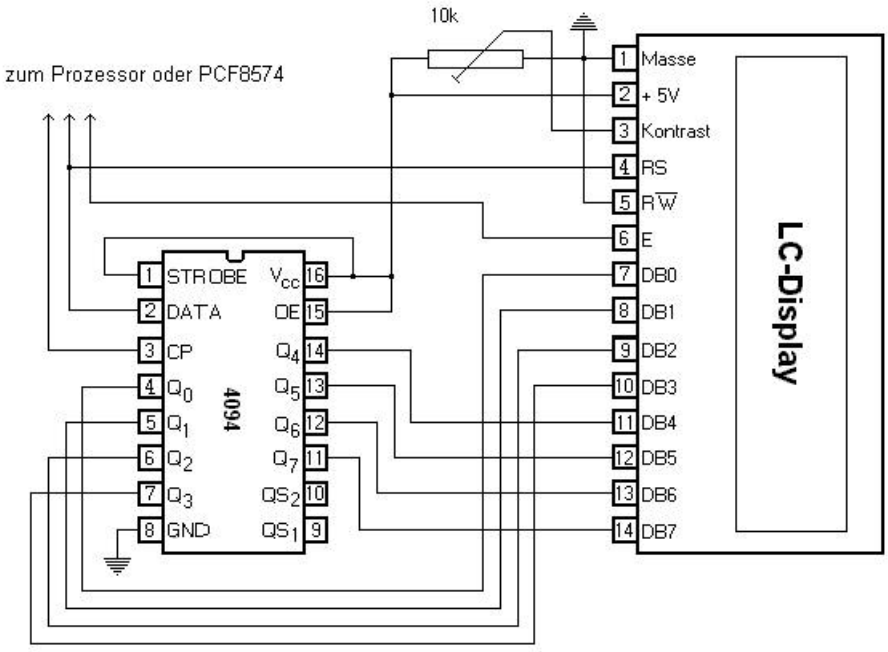
Der Einsatz eines Schieberegisters hilft hier weiter. Die Daten werden seriell in das Register geschoben und stehen dann zur Übernahme in den LCD – Controller bereit.

Ein HCT4094 eignet sich sehr gut für diese Aufgabe. Dieses Schieberegister hat acht Ausgänge (Q0 – Q7), einen Clockeingang CP, einen Dateneingang DATA, einen Übernahmeeingang STROBE, einen Eingang Output Enable OE, der auf +5V gelegt wird. Das Register funktioniert so: An den Eingang DATA wird ein Bit mit 0 oder 1 angelegt. Ein positiver Clockimpuls auf CP schiebt das angelegte Bit in das Register, wenn das STROBE – Signal auf high liegt. Das nächste Bit folgt .....





Das rechte Bild erläutert die Funktionsweise des Schieberegisters. Die Daten werden über cp (Clockpulse) in das 8-Stage SHIFT REGISTER eingetaktet. Ein positiver STROBE übermittelt die Daten in den THREE STATE OUTPUT. Wir können sowohl STROBE, wie auch OE auf logisch 1 legen. Den STROBE – Befehl nutzt man dann, wenn man die einlaufenden Daten nicht sichtbar machen will. Das Problem besteht hier nicht, da die Anzeige erst nach den eingelaufenen 8 Bits eingeschaltet wird. Der data – Eingang hat zwei Funktionen: zum einen transportiert er die Datenbits, zum zweiten schaltet er RS auf den Kommando- (0) oder Schreibmodus (1). Da bei der Realisierung der Anzeige nicht gelesen werden soll, kann RW fest auf Masse gelegt werden. Wer will, der kann ein 10k Potentiometer so anschließen, wie dargestellt. Ich habe darauf verzichtet und direkt einen Anschluß von Pin3 nach Masse geschaltet.



Das Programm zur Ansteuerung ist jetzt relativ einfach zu realisieren. Wir benötigen ein Unterprogramm zum Versenden der Kommandos und ein Unterprogramm zum Schreiben.

Das Programmbeispiel ist einfach gehalten und steuert nur 1 Zeile des Displays. Es ist jedoch leicht anzupassen, wenn man die vorhergehenden Informationen benutzt.

```
define data    port [1]
define cp      port [2]
define E       port [3]
define wert    byte
define i       byte          ' Zaehler


E    = 0    ' Enable - Impuls fuer die Anzeige
CP   = 0    ' Clock für Schieberegister
data = 0    ' Daten Schieberegister oder RS

wert = &H0E : gosub kommando ' Display 1zeilig ein, ohne Cursor
wert = &H01 : gosub kommando ' Display loeschen und HOME

#main
get wert : gosub schreiben ' Wert ueber serielle Schnittstelle
goto main

#kommando
for i=7 to 0 step -1
data = off
if wert and 1 shl i then data = on
pulse cp            ' Uebernahme ins Schiebregister
next i
data = off :pulse e ' Uebernahme ins Display
return

#schreiben
for i=7 to 0 step -1
data = off
if wert and 1 shl i then data = on
pulse cp
next i
data = on : pulse e
return
```

Wer Wert auf kompaktes Programmieren legt, der kann natürlich auch anders programmieren. Da sich die Kommando- und Schreibprozedur ähneln, kann man mit einem Unterprogramm auskommen.

```
define data    port [1]
define cp      port [2]
define E       port [3]
define wert    byte
define i       byte

E = 0 : CP = 0 : data = 0
wert = &H0E : gosub schieben : data = off : pulse e
wert = &H01 : gosub schieben : data = off : pulse e

#main
get wert : gosub schieben : data = on :pulse e
goto main
```

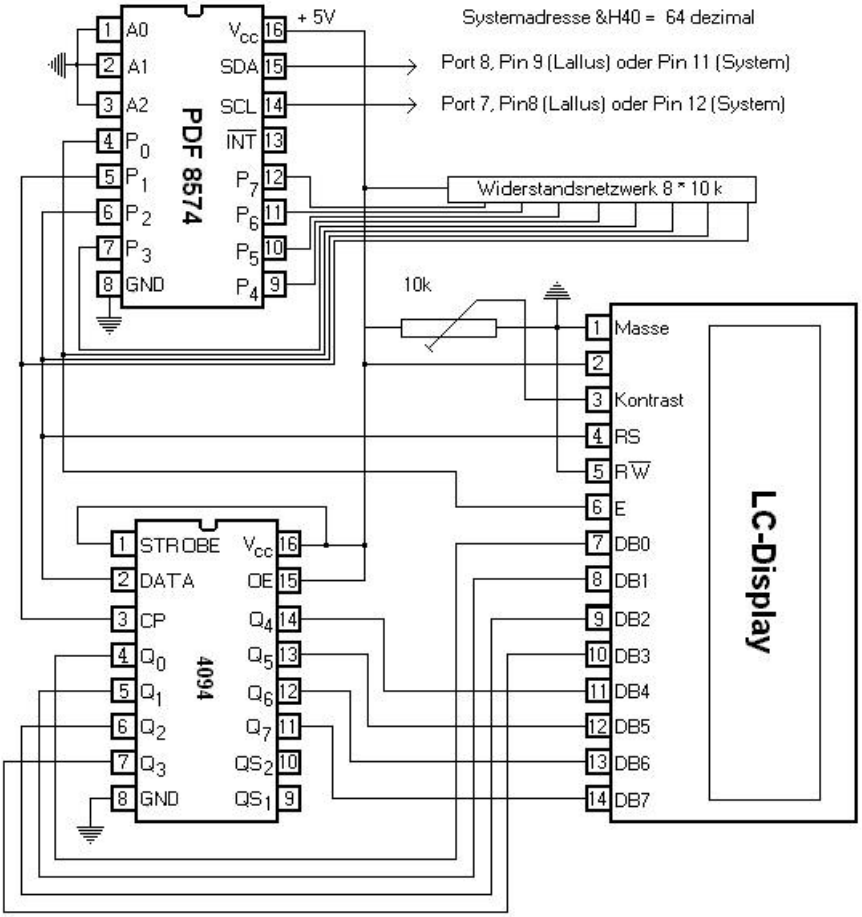
```
#schieben
for i=7 to 0 step -1
data = off
if wert and 1 shl i then data = on
pulse cp
next i
return
```

Es ist einzusehen, dass die Anzeige jetzt nicht mehr so schnell sein kann, wie beim Einsatz der Datenübermittlung mit eigenen Ports. Zunächst muß das Zeichen über die Schleife geschickt werden, dann die Übernahme in das Display. BASIC ist kein Geschwindigkeitsweltmeister. Natürlich könnte man mit einem Assemblerprogramm hier wieder schneller arbeiten.

Irgendwann kommt die Einpassung einer LC – Anzeige in das Lallussystem. Da hier alle Ports weitgehend für das System reserviert sind, muß man die Realisierung über den I2C – Bus vornehmen. Aus 11 Ports am Anfang, über drei Ports mit Hilfe des Schieberegisters werden dann nur noch 2 Digitalports – allerdings muß ein weiterer Chip – der PCF8574 – eingesetzt werden.

Hier werden dann 3 Ports des PCF benötigt, um das Schieberegister zu bedienen. Es gehört schon eine Portion Programmierkenntnis dazu, um das Register mit Hilfe des I2C – Busses richtig zu programmieren. Es sollte hier auch die Assemblerroutine I2CASM.S19 oder i2c_sysd.s19 benutzt werden, um die Anzeige zu beschleunigen. Das Ergebnis ist die Anzeige des Wortes Lallusprojekt, Bildschirm löschen, wieder diese Anzeige usw.

Bei der ersten Anwendung wird lediglich die Anzeige zur Darstellung von Buchstaben gezeigt. Bei der zweiten Anwendung werden dann echte Meßwerte verarbeitet.




```
' Programmname I2CLCD.BAS
define wert  byte     ' Variable
define i     byte     ' Zaehlervariable
define j     byte     ' Zaehlervariable
define zeile bit[192] ' Zeilenumschaltung

define data 1         ' data und RS
define cp   2         ' Clock Schieberegister
define e    4         ' Clock LC - Display
define adr &H40       ' Systemadresse PCF8574
define pos byte       ' Positionszaehler

wert=&H38:gosub kommando ' zwei Zeilen
wert=&H0C:gosub kommando ' Display ein, ohne Cursor
wert=&H01:gosub kommando ' Display clear, Cursor HOME

#main
For j=0 to 12
if pos = 8 then gosub zeile_schalten
looktab tabelle,j,wert : gosub schreiben
pos = pos + 1
next j
pause 50
wert=&H01 : gosub kommando
```

```
gosub zeile_schalten
goto main

#kommando
for i=7 to 0 step -1
'sys &H0101 adr,0,0
if wert and 1 shl i then sys &H0101 adr,data,0
sys &H0101 adr,cp,0
sys &H0101 adr,0,0
next i
sys &H0101 adr,e,0 ' RS = 0
sys &H0101 adr,0,0
return
#schreiben
for i=7 to 0 step -1
'sys &H0101 adr,0,0
if wert and 1 shl i then sys &H0101 adr,data,0
sys &H0101 adr,cp,0
sys &H0101 adr,0,0
next i
sys &H0101 adr,e + data,0 ' RS = 1
sys &H0101 adr,0,0
return

#zeile_schalten
zeile = not zeile
if zeile then wert = &H80 + &H40 else wert=&H80
gosub kommando
pos = 0
return

table tabelle
76  97  108  108  117  115  112  114  111  106  101  107  116
'L  a    l    l    u    s    p    r    o    j    e    k    t
tabend
'syscode "i2casm.s19" oder "i2c_sysd.s19"
```

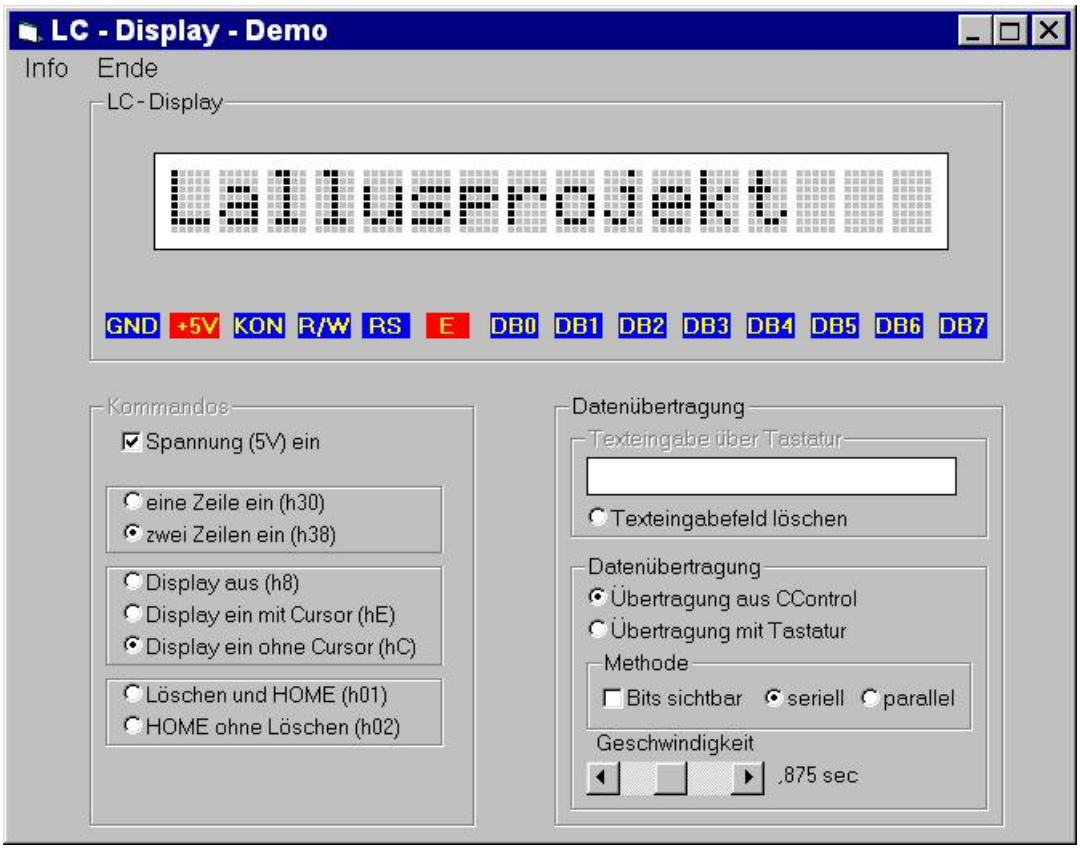
Die Spielerei mit einem LC – Display brachte sicherlich einige Programmierkenntnisse und bringt hoffentlich viel Spaß. Vor allem die Verwendung eines undokumentierten Bauteils aus dem alten Faxgerät brachte mir viel Freude. Ein Blick in das Internet – schon war alles klar.

Jetzt wird es noch wilder. Wer kein Display dieser Art zur Verfügung hat, der kann sich ein Softwaredisplay von der CD holen.
Das Programm dazu heißt LCDDEMO.EXE. Es ist in Visual Basic 3 geschrieben und simuliert eine 16 stellige Anzeige. Die Inhalte und Steuerkommandos werden von der CControl geliefert.

Man kann hier so programmieren, als hätte man eine echte Anzeige vor Augen. Natürlich werden hier keine Ports geschaltet, die Werte für das Programm werden den mit dem `print` – Befehl über die serielle Schnittstelle geliefert. In dem Basic – Programm sind nur die echten ASCII – Werte bis 127 darstellbar. Die Unterscheidung zwischen Kommando und Schreibbefehl wird daher so dargestellt: Alle Werte unter 128 sind Kommandos, alle Werte über 128 entsprechen dem Schreibbefehl Wert – 128.

Es wird eine definierte einfache Datenübertragung genutzt. Die CControl sendet einen Wert über die serielle Schnittstelle. Danach wird auf die Antwort von Visual Basic gewartet. Mit

`get wert` wird dieses erreicht. Visual Basic sendet ein „A“, wenn die Verarbeitung beendet ist.



So sieht der Bildschirm aus. Das Wort 'Lallusprojekt' in der Anzeige wurde von der CControl geliefert. Auch die Wahl, ob Cursor oder ohne Cursor usw. wurde vom Programm geschickt. Lediglich die 5Volt – Spannung muss noch angeklickt werden.

In dem Programm sind alle ASCII – Werte von 32 bis 127 abgespeichert. Zusätzlich wurde das ° Zeichen ((ASCII 0) und das Cursorzeichen (ASCII 1) als Sonderzeichen eingefügt. Es handelt sich dabei um die beiden letzten Zeichen.
Die untenstehende Tabelle zeigt die Werte:

Das Programm zur Steuerung ist relativ einfach:

```
' Programmname : LCDVBAS.BAS
define wert    byte
define i       byte
```

```
print &H38   ' zwei Zeilen einschalten
get wert     ' warten auf Antwort
print &Hc    ' Display ein, ohne Cursor
get wert     ' warten auf Antwort

#main
looktab tabelle,i,wert ' Wert aus Tabelle lesen
print wert+128         ' Schreiben (+128)
get wert               ' warten auf Antwort
i = i + 1
if i = 13 then gosub Zeile_loeschen
goto main

#Zeile_loeschen
i = 0                  ' Zaehler zuruecksetzen
print "1"              ' Kommando loeschen und HOME
get wert               ' warten auf Antwort
return

table tabelle
76  97  108  108  117  115  112  114  111  106  101  107  116
'L  a    l    l    u    s    p    r    o    j    e    k    t
tabend
```

Natürlich kann das Programm auch von Hand bedient werden, um die einzelnen Funktionen kennenzulernen. Es muss hierfür der Optionbutton ***Übertragung mit Tastatur*** angeklickt werden.Mit ***Bits sichtbar*** kann man die angelegten Muster beobachten. Zum einen ist die serielle Übertragung, zum anderen die parallele Übertragung mit 8 eigenen Ports darstellbar. Mit dem Scrollbalken läßt sich die Pausenzeit einstellen.

Auf einige Spezialitäten, wie Blinken, Shiften oder eigene Charakter erstellen wurde hier verzichtet, da dies noch einige Stunden mehr Programmierarbeit gekostet hätte.

Natürlich kann das Programm so abgewandelt werden, dass echte gemessene Werte angezeigt werden. Es muß dafür lediglich der Wert auf die serielle Schnittstelle mit einem `print` – Befehl ausgegeben werden. Dafür hier ein kleines Programm, das die gemessenenen Temperaturen eines DS1621 über die serielle Schnittstelle an Visual Basic liefert. Das Assemblerprogramm „i2c_sysd.s19“ muß vorhanden sein.

Auch hier wurde wieder ein einfaches Übertragungsprotokoll installiert. Die CControl wartet auf ein Zeichen (A) von Visual Basic, bevor sie weiterarbeiten kann.

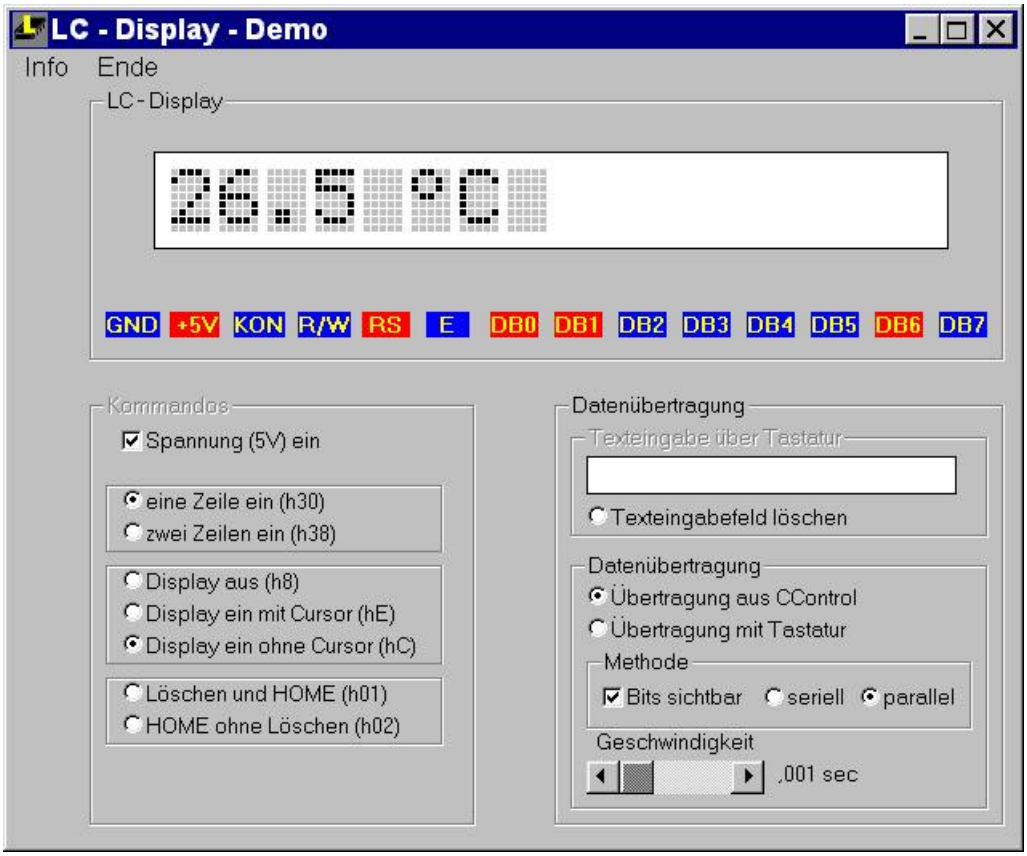
```
' LCDVTEMP.BAS
define aus1 byte
define aus2 byte
define wert byte

print "48" ' eine Zeile ein
get wert
print "12" ' Display ein ohne Cursor
get wert

sys &H0101 &H92,&HAC,0,0,0 ' kontinuierlich wandeln
sys &H0101 &H92,&HEE,0,0,0 ' Wandlung ein
sys &H0101 &H92,&HAA,0,0,0 ' Lesemodus ein
```

```
#main
sys &H0101 &H92,0,0,0,3 ' zwei Bytes von DS 1621 lesen
print aus1/10 + 48 + 128
get wert
print aus1 mod 10 + 48+128
get wert
print 46 + 128 ' Dezimalpunkt
get wert
print (aus2 =128)*-5 +48 +128
get wert
print 32 + 128 ' Leerzeichen
get wert
print 128      ' Character 0 (Gradzeichen)
get wert
print 67+128   ' Character C
get wert
pause 50
print "1"      ' Display loeschen + Home
get wert
goto main

'syscode "i2c_sysd.s19"
```

Das Programmfenster sollte dann so aussehen, wenn alles ordnungsgemäß läuft.
Noch einmal zur Erinnerung:

Im Visual Basic- Programm Spannung (5V) ein anklicken, dann Übertragung aus CControl anklicken, angebotene Schnittstelle wählen, dann CControl starten.

An der Abbildung ist zu erkennen, dass gerade das Zeichen C (DB0 =1, DB1=2 und DB6=64 = ASCII 67) übertragen wurde. RS steht auf 1 = Schreibbefehl.


LC - Display - Demo
Info
Ende
LC - Display
GND
+5V
KON
R/W
RS
E
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
Kommandos
Spannung (5V) ein
eine Zeile ein (h30)
zwei Zeilen ein (h38)
Display aus (h8)
Display ein mit Cursor (hE)
Display ein ohne Cursor (hC)
Löschen und HOME (h01)
HOME ohne Löschen (h02)
Datenübertragung
Texteingabe über Tastatur
Texteingabefeld löschen
Datenübertragung
Übertragung aus CControl
Übertragung mit Tastatur
Methode
Bits sichtbar
seriell
parallel
Geschwindigkeit
,001 sec