Uwe Koch
Frankenstraße 25
5880 LÜDENSCHEID
Tel. (0 23 51) 2 61 92

LOOP 6
1. JAHRGANG

Zeitung für Computer-Bauer, -Anwender, -Programmierer und -Starter DM 3,–

In eigener Sache . . .

1986 – das Jahr der Software.

1985 hatten wir uns als das Jahr der Dokumentation vorgenommen. Wir glauben, daß uns dies gut gelungen ist – es entstanden neue, wesentlich umfangreichere Handbücher und die Zeitschrift *„LOOP"*, die anerkannt besser geworden ist.

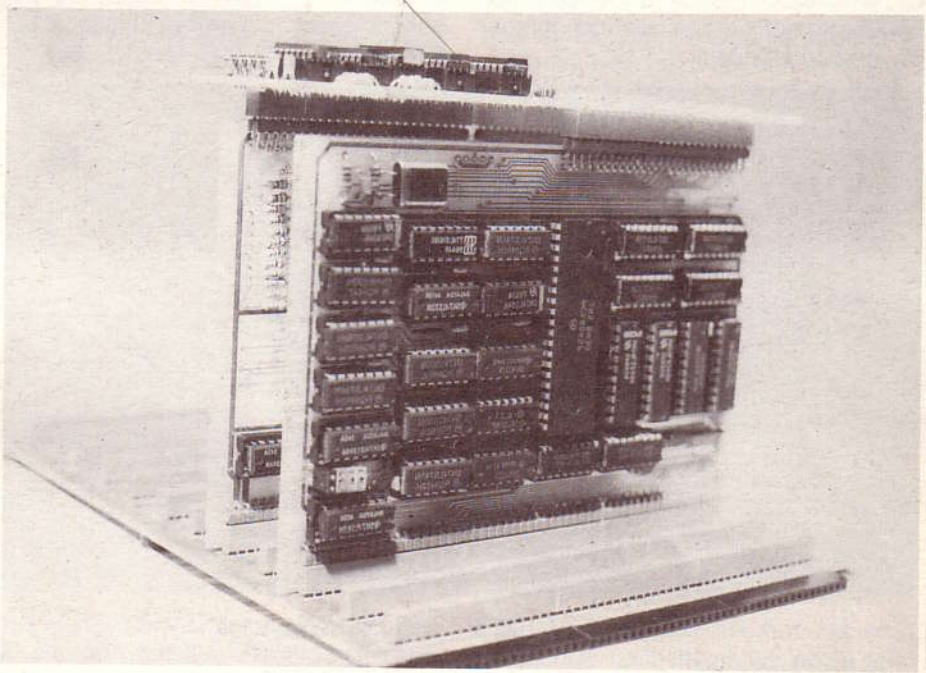
Wir wollen uns mit *„LOOP"* nicht in die endlose Reihe der Computerzeitschriften am Kiosk einreihen. *„LOOP"* wird auch weiterhin nur über Abos und im Direktverkauf von GES vertrieben. Wir wollen jedoch *„LOOP"* möglichst regelmäßig erscheinen lassen – dazu brauchen wir Ihre Mitarbeit. *„LOOP"* soll nicht mit Inseraten am Leben erhalten werden. Das in Fachzeitungen „normale" Verhältnis von 40 bis 50% Inseratenanteil (mehr als 50% läßt die Post nicht zu) können und wollen wir nicht erreichen. *„LOOP"* lebt aber von seinen Abonnenten und von Leuten, die für eine kleine Anerkennung uns ihre fast immer hervorragenden Artikel zur Verfügung stellen. Ein klein wenig tragen hierzu auch die Herausgeber, Rolf-Dieter Klein und Gerd Graf, bei.

Knappe 2.000 Abonnenten haben wir bis heute und 5.000 haben wir uns bis Ende 1986 zum Ziel gesetzt. Also ein Aufruf an alle „mal so"-Leser: bitte senden Sie uns den Abo-Schein.

Helfen, die *„LOOP"* besser zu machen, können Sie alle. Durch Ihre begründete Kritik – wir drucken alle Leserbriefe ab, nicht nur die guten – durch Ihre Mitarbeit, besonders für die Anfänger, und durch Ihre Treue.

Wir versprechen, *„LOOP"* weiter zu verbessern und diese Zeitschrift, obwohl vom Hersteller herausgegeben, nie zu einer Werbeschrift zu machen.

Rolf-Dieter Klein, Gerd Graf



Endlich: Farbe für den NDR-Computer

Die COL256-Baugruppen sind nun lieferbar Seite 2

Der NDR-Computer wieder im Fernsehen!

Jetzt „Rechner modular" – Neue Folgen in Hamburg gedreht

Ab Samstag, 11. 1. 1986, 17.15 Uhr, in Bayern III – über Kabel nach Berlin

Ab Montag, den 6. 1. 1986, 16.45 Uhr in NDR III Näheres Seite 5

Schwerpunkt – für Einsteiger Seiten 6 – 10

Neue Produkte lieferbar!

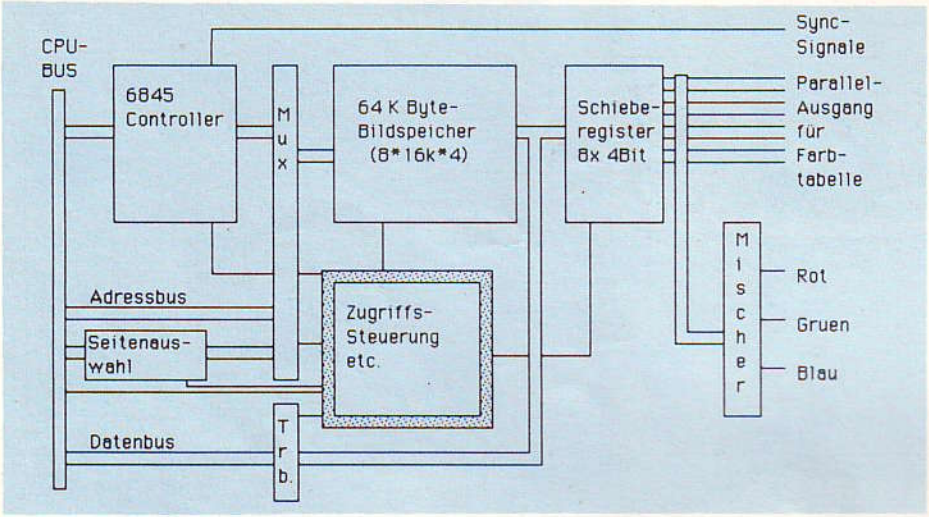
COL256, Relais, Hardcopy-Maus, CPU68000 mit 16 bit und 8 oder 12 MHz sind lieferbar! Genaue Beschreibungen auf den Seiten 22 und 23

Farbe für den NDR-Computer – COL256 – nun lieferbar

Teil 1

von Rolf-Dieter Klein

Farbige Bilder auf dem Bildschirm darstellen zu können, das war bisher mit dem NDR-Klein-Computer nicht möglich. Die Baugruppe COL256 bringt jetzt Farbe in den Computer. Sie besitzt einen eigenen 64KByte großen RAM Speicher und kann damit z.B. **256 mal 256 Punkte mit je 256 verschiedenen Farben pro Bildpunkt darstellen.** Die Baugruppe ist für alle CPUs geeignet; hier soll am Beispiel des 68008 gezeigt werden, wie man mit der Baugruppe arbeitet.



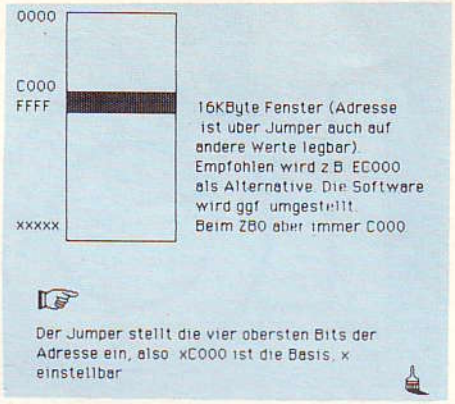
Zunächst zum Hardware-Aufbau. Bild 1 zeigt einen schematischen Schaltplan. Die Erzeugung der Synchronsignale und des Timings übernimmt ein Bildschirmcontroller mit der Bezeichnung MC6845. Dieser Baustein ist schon sehr lange auf dem Markt und eigentlich für Textdarstellung entworfen. Er läßt sich auch ohne große Probleme für Graphik verwenden. Im Gegensatz zu einem Graphikcontroller hat er selbst keinen Zugriff auf den Bildspeicher, sondern sorgt nur dafür, daß die Bildpunkte nacheinander ausgelesen werden. Der Controller besitzt eine Möglichkeit, die Zeilen des Bildschirms zu scrollen. Dazu beginnt er einfach bei einer neuen Startadresse den Bildspeicher auszulesen. In der Schaltung der COL256 ist ein Scroll um je vier Zeilen möglich. Im Controller gibt es dazu spezielle Register (siehe Datenblatt 6845).

Man kann auch horizontal scrollen, ebenfalls um vier Bildpunkte. Wenn man 64KByte als Bildspeicher verwendet, so kann man genau eine Bildseite darstellen. Will man mehrere Bildseiten verwenden, so kann man auf der Baugruppe auch andere Speicher einsetzen, z.B. 41254, die 64K x 4 organisiert sind und hat dann vier Bildebenen, ähnlich wie auf der GDP64. Man kann dann beim Scrollen von einer Bildebene in die nächste scrollen.

Der Bildspeicher ist bei 64KByte mit 8 x 4, also 32 parallelen Ausgängen organisiert, die an 8 Schieberegister mit je 4 Bit führen. Damit kann man also 8 Bit pro Bildpunkt verwenden. Ein kleiner Mischer sorgt dafür, daß man auch auf einfache Weise drei Ausgänge mit analogen Signalen für R (Rot), G (Grün) und B (Blau) erhält. Diese Ausgänge kann man direkt an eine SCART-Buchse eines Farbfernsehers legen, oder an den Eingang eines Farbmonitors mit RGB (analog)-Eingängen. Die notwendigen Sync-Signale werden vom Controller geliefert. Achtung, für den SCART-Anschluß muß man ggf. Monoflops oder Inverter dazwischenschalten. (Im Bausatz enthalten.)

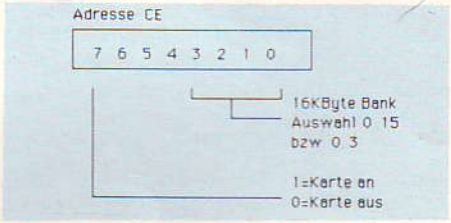
An die 8 Parallel-Ausgänge kann man z.B. eine Farbtabelle (G170) anschließen, mit der man dann 256 aus 262144 Farben darstellen kann. Die Farbtabellen-Baugruppe befindet sich aber noch im Prototypen-Stadium.

Auf der Baugruppe sorgt eine Zugriffsteuerung dafür, daß man jederzeit vom Prozessor aus auf den Bildspeicher zugreifen kann. Dabei wird ein sogenannter transparenter Zugriff durchgeführt, der Prozessor muß also nicht auf den Auslesevorgang des Bildspeichers achten, da beide Zugriffe ineinander verzahnt sind. Damit lassen sich sehr hohe Verarbeitungsgeschwindigkeiten erreichen. Der Bildspeicher ist vom Prozessor aus als 16KByte Speicherbereich sichtbar. Über einen Port (Adresse ØCEH) kann man eine Seitenauswahl treffen, denn der Bildspeicher ist ja mindestens 64K Byte oder maximal 256K Byte groß. Aber über die 16KByte-Fenster kann man auch mit dem Z80 bequem auf den Speicher zugreifen, Bild 2 zeigt das Schema.



Über Brücken kann man die Lage des Fensters bestimmen, jedoch nur über die vier höherwertigen Adressbits (A19 – A16). Das Fenster liegt also immer auf einer C000-Adresse. Über die Banksel-Leitung wird dafür gesorgt, daß entsprechender Hauptspeicher ausgeblendet wird. Achtung, die BANKBOOT-Baugruppe ist noch nicht dafür vorbereitet. Man muß einen Open-Controller-Treiber in die BANKSEL-Leitung legen, damit auf dem Bus mehr als eine Karte diese Leitung schalten kann. Bisher liegt dort ein normaler TTL-Ausgang. (Dies ist im Handbuch der COL256 erläutert.)

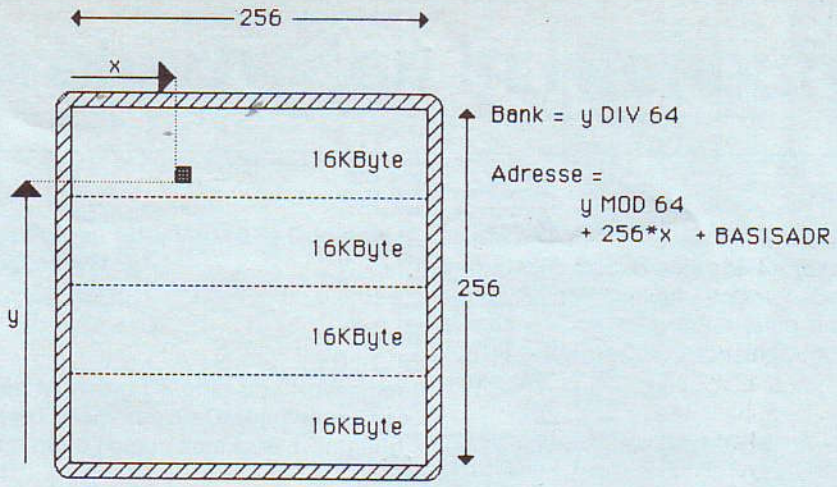
Die SBC3 ist bereits dafür ausgelegt. Bild 4 zeigt die Belegung des Seitenauswahl-



Ports. Bit 7 bestimmt, ob der Speicher eingeblendet werden soll. Ist Bit 7 auf 1, so wird der Speicher eingeblendet. Nach dem RESET ist Bit 7 gelöscht. Die Bits 0 bis 3 bestimmen die 16K Byte-Bank, die im Fenster erscheinen soll.

Das COL256-System erzeugt 256 * 256 Bildpunkte mit 256 Farben, pro Bildpunkt einzeln setzbar. Es besteht aus zwei (Europa-)Karten mit NDR-Bus und einer Verbindungsbaugruppe.

In Bild 6 ist die Aufteilung des Bildschirms gezeigt, wenn man als Startadresse im 6845 Null wählt (Standard kein Scroll). Ein Bildpunkt ist ein Pixel; will man einen Punkt mit x,y-Koordinate setzen, so kann man die Speicheradresse und die Bank-Adresse einfach ausrechnen. Bild 7 zeigt die Belegung eines einzelnen Pixels. Bit 0 und 1 bestimmen die Intensität der Farbe Rot, Bit 2 und 3 von Grün und Bit 4 und 5 die Intensität von Blau. Bit 6 und 7 haben eine besondere Eigenschaft, sie bestimmen die Intensität aller Farben und zwar als niederwertigste Stelle. Damit ist es möglich z.B. 16 verschiedene Grautöne zu erzeugen und auch 16 verschiedene Stufungen der Grundfarben, wobei nur

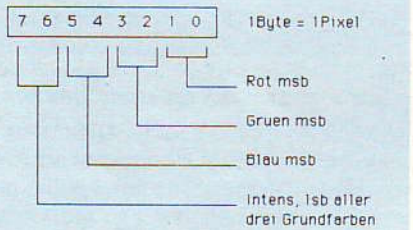


Die angegebenen Formeln gelten für eine Auflösung von 256 x 256 Punkten. Wenn man ein anderes Format verwendet (z.B. 320 x 200), so ändern sich die Formeln entsprechend.

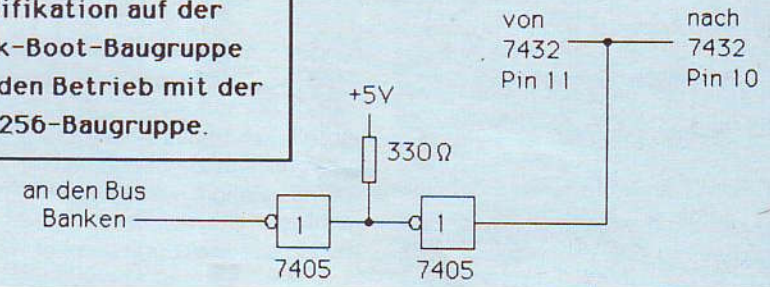
eine leichte Farbverschiebung stattfindet. Die Aufteilung gilt für den eingebauten Mischer. Wenn man eine Farbtabelle verwendet, ist man völlig frei. Übrigens kann man auch 512 Punkte oder 640 Punkte in einer Zeile darstellen, sobald man einen Multiplexer hinter den Parallel-Ausgang schaltet und diesen mit dem Bildpunkttakt umschaltet. Dann hat man aber nur 16 Farben pro Pixel. Auch 512 Zeilen sind möglich, falls man 256K Byte Bildspeicher verwendet und im Controller auf Zeilensprung umschaltet. Man erkennt die enormen Möglichkeiten, die die Baugruppe bietet.

Bild 8 zeigt ein kleines Softwarepaket, mit dem man die Baugruppe in Betrieb nehmen kann. Das Unterprogramm INIT1, das auch INIT2 aufruft, sorgt dafür, daß der Controller richtig eingestellt wird.

Nach Start des Programms wird der Bildschirm auf Grün gelöscht und eine Diagonale von links unten nach rechts oben gezeichnet. Dann wird der Farbcode jedes Bildpunktes invertiert, und dann das



Modifikation auf der Bank-Boot-Baugruppe für den Betrieb mit der COL256-Baugruppe.




```
Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 1


0E9C00                  ********************************************
0E9C00                  * UNTERPROGRAMME ZUM BETRIEB DER FARBKARTE  *
0E9C00                  * COL 256                                   *
0E9C00                  * ROLF-DIETER KLEIN 851129     REV 1.1     *
0E9C00                  ********************************************
0E9C00
0E9C00                  * KLEINER AUSZUG AUS DER GROSSEN BIBLIOTHEK
0E9C00
= FFFFFFCC              CRT EQU $FFFFFFCC
= FFFFFFCD              CRTD EQU $FFFFFFCD
= FFFFFFCE              CRTB EQU $FFFFFFCE
0E9C00
0E9C00                  START:          * KLEINER TEST
0E9C00  6100 008A        BSR INIT1      * 6845 INITIALISIEREN
0E9C04  103C 00CC        MOVE.B #$CC,D0     * HINTERGRUND GRUEN
0E9C08  6100 00C6        BSR CLRCOL1
0E9C0C  13FC 00C3        MOVE.B #$C3,COLOR * MAL-FARBE    ROT
0E9C10  000E9EB2
0E9C14  4239 000E9EA8    CLR.B XORMODE      * DOMINANT ZEICHNEN
0E9C1A  4279 000E9EAA    CLR X1
0E9C20  4279 000E9EAE    CLR Y1
0E9C26  33FC 00FF        MOVE #255,X2
0E9C2A  000E9EAC
0E9C2E  33FC 00FF        MOVE #255,Y2   * DIAGONALE
0E9C32  000E9EB0
0E9C36  6100 01AE        BSR DRAW1      *
0E9C3A  3E3C 0001        MOVE #2-1,D7   * NUN BILD ZWEIMAL KOMPLEMENTIEREN
0E9C3E  33FC 0001        MOVE #1,XORMODE   * KOMPLEMENT MODE
0E9C42  000E9EA8
0E9C46  13FC 00FF        MOVE.B #$FF,COLOR * ALLE BITS.
0E9C4A  000E9EB2
0E9C4E                  LP1:
0E9C4E  323C 0000        MOVE #0,D1     * X=0
0E9C52                  LP2:
0E9C52  33C1 000E9EAA    MOVE D1,X1
0E9C58  33C1 000E9EAC    MOVE D1,X2
0E9C5E  33FC 0000        MOVE #0,Y1
0E9C62  000E9EAE
0E9C66  33FC 00FF        MOVE #255,Y2
0E9C6A  000E9EB0
0E9C6E  48E7 4100        MOVEM.L D1/D7,-(A7)
0E9C72  6100 0172        BSR DRAW1 -
0E9C76  4CDF 0082        MOVEM.L (A7)+,D1/D7
0E9C7A  0641 0001        ADD #1,D1
0E9C7E  0C41 0100        CMP #256,D1
0E9C82  6600 FFCE        BNE LP2
0E9C86  51CF FFC6        DBRA D7,LP1
0E9C8A  4E75             RTS            * ENDE
0E9C8C
0E9C8C                  INIT1:
0E9C8C  4279 000E9EA8    CLR XORMODE
0E9C92  13FC 00FF        MOVE.B #$FF,COLOR
0E9C96  000E9EB2
0E9C9A  6100 0004        BSR INIT2      * REV, KEIN LOESCHEN MEHR DANACH
0E9C9E  4E75             RTS
0E9CA0
```

```
0E9CA0                  ********************************************
0E9CA0                  * SYNC PARAMETER ETC INITIALISIEREN, 6845
0E9CA0                  * HIER WIRD DAS FORMAT 256*256
0E9CA0                  * EINGESTELLT
0E9CA0
0E9CA0                  INIT2:
0E9CA0  41FA 001C        LEA TAB(PC),A0    * TABELLE DER CODES

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 2

0E9CA4  4241             CLR D1
0E9CA6  363C 000E        MOVE #LENGTH-1,D3
0E9CAA                  LPP:
0E9CAA  13C1 FFFFFFCC    MOVE.B D1,CRT
0E9CB0  13D8 FFFFFFCD    MOVE.B (A0)+,CRTD
0E9CB6  5241             ADDQ #1,D1
0E9CB8  51CB FFF0        DBRA D3,LPP
0E9CBC  4E75             RTS
0E9CBE
0E9CBE                  TAB:
0E9CBE  6F 40 50 07 4D   DC.B 111,64,80,7,77,0,64,70,0,3,0,0,0,0,0
0E9CC3  00 40 46 00 03
0E9CC8  00 00 00 00 00
0E9CCD
= 0000000F              LENGTH EQU *-TAB
0E9CCD
0E9CCD  00               DS 0
0E9CCE
0E9CCE                  ********************************************
0E9CCE                  * LOESCHEN DES BILDSCHIRMS
0E9CCE
0E9CCE                  CLEAR1:                 * LOESCHEN AUF SCHWARZ
0E9CCE  4280             CLR.L D0
0E9CD0                  CLRCOL1:                * LOESCHEN MIT FARBCODE 0..255
0E9CD0  13FC 00B0        MOVE.B #$B0,CRTB        * CODE IN D0.B
0E9CD4  FFFFFFCE
0E9CD8  6100 002E        BSR CLRP
0E9CDC  13FC 00B1        MOVE.B #$B1,CRTB
0E9CE0  FFFFFFCE
0E9CE4  6100 0022        BSR CLRP
0E9CE8  13FC 00B2        MOVE.B #$B2,CRTB
0E9CEC  FFFFFFCE
0E9CF0  6100 0016        BSR CLRP
0E9CF4  13FC 00B3        MOVE.B #$B3,CRTB
0E9CF8  FFFFFFCE
0E9CFC  6100 000A        BSR CLRP
0E9D00  4239 FFFFFFCE    CLR.B CRTB
0E9D06  4E75             RTS
0E9D08
0E9D08                  CLRP:           * UPRG. 16K BYTE SCHNELL LOESCHEN.
0E9D08  41F9 0000C000    LEA $C000,A0
0E9D0E  1600             MOVE.B D0,D3
0E9D10  E158             ROL #8,D0
0E9D12  1003             MOVE.B D3,D0
0E9D14  3600             MOVE D0,D3
0E9D16  4840             SWAP D0
0E9D18  3003             MOVE D3,D0
0E9D1A  363C 0FFF        MOVE #$1000-1,D3
```

```
0E9D1E                  LOOP:
0E9D1E  20C0             MOVE.L D0,(A0)+
0E9D20  51CB FFFC        DBRA D3,LOOP
0E9D24  4E75             RTS
0E9D26
0E9D26                  ****************************************
0E9D26                  * VERSCHIEDENE PUNKTSETZ-ROUTINEN
0E9D26
0E9D26                  * D0.B = FARBCODE, D1=X ,D2=Y
0E9D26
0E9D26                  SETFDOT1:       * DOMINANT PUNK SETZEN
0E9D26  3F02             MOVE D2,-(A7)
0E9D28  4602             NOT.B D2        * ACHTUNG NICHT NEG VERWENDEN
0E9D2A  EC5A             ROR.W #6,D2     * AAAAAAXX000000AA
0E9D2C  0002 0080        OR.B #$80,D2    * AAAAAAXX100000AA
0E9D30  13C2 FFFFFFCE    MOVE.B D2,CRTB  * SELEKT DURCHFUEHREN

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 3

0E9D36  E44A             LSR.W #2,D2     * 00AAAAAA--------
0E9D38  1401             MOVE.B D1,D2    * 00AAAAAABBBBBBBB
0E9D3A  41F9 0000C000    LEA $C000,A0    * BASIS-ADRESSE COL256
0E9D40  D0C2             ADDA.W D2,A0    * A0=ADRESSE PIXEL.
0E9D42  341F             MOVE (A7)+,D2
0E9D44  1080             MOVE.B D0,(A0)
0E9D46  4239 FFFFFFCE    CLR.B CRTB
0E9D4C  4E75             RTS
0E9D4E
0E9D4E                  SETDOT1:    * TRANSPARENT PUNKT SETZEN (ODERN)
0E9D4E  3F02             MOVE D2,-(A7)
0E9D50  4602             NOT.B D2        * ACHTUNG NICHT NEG VERWENDEN
0E9D52  EC5A             ROR.W #6,D2     * AAAAAAXX000000AA
0E9D54  0002 0080        OR.B #$80,D2    * AAAAAAXX100000AA
0E9D58  13C2 FFFFFFCE    MOVE.B D2,CRTB  * SELEKT DURCHFUEHREN
0E9D5E  E44A             LSR.W #2,D2     * 00AAAAAA--------
0E9D60  1401             MOVE.B D1,D2    * 00AAAAAABBBBBBBB
0E9D62  41F9 0000C000    LEA $C000,A0    * BASIS-ADRESSE COL256
0E9D68  D0C2             ADDA.W D2,A0    * A0=ADRESSE PIXEL.
0E9D6A  341F             MOVE (A7)+,D2
0E9D6C  8110             OR.B D0,(A0)
0E9D6E  4239 FFFFFFCE    CLR.B CRTB
0E9D74  4E75             RTS
0E9D76
0E9D76
0E9D76                  XORDOT1:.   * FARBE KOMPLEMENTIEREN , D0.B = CODE.
0E9D76  3F02             MOVE D2,-(A7)
0E9D78  4602             NOT.B D2        * ACHTUNG NICHT NEG VERWENDEN
0E9D7A  EC5A             ROR.W #6,D2     * AAAAAAXX000000AA
0E9D7C  0002 0080        OR.B #$80,D2    * AAAAAAXX100000AA
0E9D80  13C2 FFFFFFCE    MOVE.B D2,CRTB  * SELEKT DURCHFUEHREN
0E9D86  E44A             LSR.W #2,D2     * 00AAAAAA--------
0E9D88  1401             MOVE.B D1,D2    * 00AAAAAABBBBBBBB
0E9D8A  41F9 0000C000    LEA $C000,A0    * BASIS-ADRESSE COL256
0E9D90  D0C2             ADDA.W D2,A0    * A0=ADRESSE PIXEL.
0E9D92  341F             MOVE (A7)+,D2
0E9D94  B110             EOR.B D0,(A0)
0E9D96  4239 FFFFFFCE    CLR.B CRTB
0E9D9C  4E75             RTS
0E9D9E
0E9D9E                  GETCOL1: * D1/D2  -> D0.B, FARBCODE EINLESEN
0E9D9E  3F02             MOVE D2,-(A7)
0E9DA0  4602             NOT.B D2        * ACHTUNG NICHT NEG VERWENDEN
0E9DA2  EC5A             ROR.W #6,D2     * AAAAAAXX000000AA
0E9DA4  0002 0080        OR.B #$80,D2    * AAAAAAXX100000AA
0E9DA8  13C2 FFFFFFCE    MOVE.B D2,CRTB  * SELEKT DURCHFUEHREN
0E9DAE  E44A             LSR.W #2,D2     * 00AAAAAA--------
0E9DB0  1401             MOVE.B D1,D2    * 00AAAAAABBBBBBBB
0E9DB2  41F9 0000C000    LEA $C000,A0    * BASIS-ADRESSE COL256
0E9DB8  D0C2             ADDA.W D2,A0    * A0=ADRESSE PIXEL.
0E9DBA  341F             MOVE (A7)+,D2
0E9DBC  1010             MOVE.B (A0),D0
0E9DBE  4239 FFFFFFCE    CLR.B CRTB
0E9DC4  4E75             RTS
0E9DC6
0E9DC6                  * OHNE BEREICHSPRUEFUNG, JEDOCH SICHER. A0 IST ERG. ADR.
0E9DC6                  CALCDOT:    * ADRESSE BERECHNEN. D1.W=0.255,D2.W=0.255
0E9DC6  3F02             MOVE D2,-(A7)  * RETTEN Y-KOORDINATE
0E9DC8  4602             NOT.B D2       * ACHTUNG NICHT NEG VERWENDEN
0E9DCA  EC5A             ROR.W #6,D2    * AAAAAAXX000000AA
0E9DCC  0002 0080        OR.B #$80,D2   * AAAAAAXX100000AA
0E9DD0  13C2 FFFFFFCE    MOVE.B D2,CRTB * SELEKT DURCHFUEHREN
0E9DD6  E44A             LSR.W #2,D2    * 00AAAAAA--------
0E9DD8  1401             MOVE.B D1,D2   * 00AAAAAABBBBBBBB

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 4

0E9DDA  41F9 0000C000    LEA $C000,A0   * BASIS-ADRESSE COL256
0E9DE0  D0C2             ADDA.W D2,A0   * A0=ADRESSE PIXEL.
0E9DE2  341F             MOVE (A7)+,D2  * Y ZURUECK.
0E9DE4  4E75             RTS
0E9DE6
```

```
0E9DE6                  ****************************************
0E9DE6                  * BRESENHAM ALGORITHMUS, ZEICHNEN EINER
0E9DE6                  * LINIE AUF DEM BILDSCHIRM.
0E9DE6                  * IN X1,Y1 IST DER STARTPUNKT
0E9DE6                  * IN X2,Y2 DER ENDPUNKT.
0E9DE6                  * COLOR.B ENTHAELT DEN FARBCODE
0E9DE6                  * XORMODE.W =0, DANN DOMINANT
0E9DE6                  * XORMODE.W =1, DANN KOMPLEMENTIEREN
0E9DE6                  * XORMODE.W =2, DANN ODERN
0E9DE6
0E9DE6                  DRAW1:   * VON X1,Y1 NACH X2,Y2
0E9DE6  383C 0001        MOVE #1,D4
0E9DEA  3A39 000E9EAC    MOVE X2,D5
0E9DF0  9A79 000E9EAA    SUB X1,D5
0E9DF6  6A00 0008        BPL ST1
0E9DFA  383C FFFF        MOVE #-1,D4
0E9DFE  4445             NEG D5
0E9E00                  ST1:
0E9E00  3C3C 0001        MOVE #1,D6
0E9E04  3E39 000E9EB0    MOVE Y2,D7
0E9E0A  9E79 000E9EAE    SUB Y1,D7
0E9E10  6A00 0008        BPL ST2
0E9E14  3C3C FFFF        MOVE #-1,D6
0E9E18  4447             NEG D7
0E9E1A                  ST2:
0E9E1A  3605             MOVE D5,D3
0E9E1C  BA47             CMP D7,D5
0E9E1E  6A00 0006        BPL ST3
0E9E22  3607             MOVE D7,D3
0E9E24  4443             NEG D3
0E9E26                  ST3:
0E9E26                  S4:
0E9E26  3239 000E9EAA    MOVE X1,D1
0E9E2C  3439 000E9EAE    MOVE Y1,D2
0E9E32  1039 000E9EB2    MOVE.B COLOR,D0
0E9E38  4A79 000E9EA8    TST XORMODE    * MODE AUSWAEHLEN
0E9E3E  6606             BNE.S XXDR1    * =0 DANN STD-SETZEN
0E9E40  6100 FEE4        BSR SETFDOT1   *    SETZEN DOMINANT
0E9E44  6014             BRA.S XXDR2
0E9E46                  XXDR1:
0E9E46  0C79 0001        CMP #1,XORMODE
0E9E4A  000E9EA8
0E9E4E  6606             BNE.S XXDR11
0E9E50  6100 FF24        BSR XORDOT1    *  XORMODE
0E9E54  6004             BRA.S XXDR2
0E9E56                  XXDR11:
0E9E56  6100 FEF6        BSR SETDOT1    * OR-MODE
0E9E5A                  *
0E9E5A                  XXDR2:
0E9E5A  3039 000E9EAA    MOVE X1,D0
0E9E60  B079 000E9EAC    CMP X2,D0
0E9E66  6700 0028        BEQ S7
0E9E6A                  S5:
0E9E6A  4A43             TST D3
0E9E6C  6B00 0012        BMI S6
0E9E70  D244             ADD D4,D1
0E9E72  33C1 000E9EAA    MOVE D1,X1
0E9E78  9647             SUB D7,D3
0E9E7A  9647             SUB D7,D3

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 5

0E9E7C  6000 FFA8        BRA S4
0E9E80                  S6:
0E9E80  D446             ADD D6,D2
0E9E82  33C2 000E9EAE    MOVE D2,Y1
0E9E88  D645             ADD D5,D3
0E9E8A  D645             ADD D5,D3
0E9E8C  6000 FF98        BRA S4
0E9E90                  S7:
0E9E90  3039 000E9EAE    MOVE Y1,D0
0E9E96  B079 000E9EB0    CMP Y2,D0
0E9E9C  6600 FFCC        BNE S5
0E9EA0  4239 FFFFFFCE    CLR.B CRTB
0E9EA6  4E75             RTS
0E9EA8
0E9EA8
0E9EA8                  * SPEICHERZELLEN
0E9EA8
0E9EA8                  XORMODE: DS.W 1
0E9EAA
0E9EAA                  X1: DS.W 1   * WERDEN NACH AUFRUF VON DRAW1
0E9EAC                  X2: DS.W 1   * ZERSTOERT.
0E9EAE                  Y1: DS.W 1
0E9EB0                  Y2: DS.W 1
0E9EB2
0E9EB2                  COLOR: DS.B 1
0E9EB3
0E9EB3                  END

0E8CE8  Ende-Symboltabelle
```

Ganze wiederholt. Am Schluß muß wieder der Hintergrund grün sein und die Diagonale rot. Damit kann man einen Speichertest durchführen. Es dürfen sonst keine Einzelpunkte sichtbar werden. Abschließend noch ein Trick: Nachdem sich das Grundprogramm wieder gemeldet hat, rufen Sie einmal das Menue IO-Setzen auf und geben auf den Port $FFFFFFCE den Wert $80. Damit wird die Farbkarte wieder eingeblendet. Nun rufen Sie den Texteditor auf, wählen aber zuvor als Textstartadresse den Wert $C000. Nun tippen Sie einen kleinen Text ein und geben dann CTRL-C ein. Auf dem Bildschirm wird nun der Text als Punktmuster sichtbar. So können Sie Speicherbereiche beobachten. Wenn Sie hinter $C000 kein RAM mehr haben, wird auch der System-Stack im Bildspeicher liegen, das erkennt man daran, daß sich laufend Punkte im Speicher ändern. Wenn man selbst Programme für die Baugruppe schreibt, muß man immer genau darauf achten, wo das Programm und der Stack liegen, damit beim Umschalten keine Störungen auftreten.

In den abgedruckten Unterprogrammen wird das automatisch gewährleistet, jedoch darf das Programm nicht im Bereich $C000 bis $FFFF liegen. Übrigens, für CP/M68K gibt es zur Verwendung unter C eine Bibliothek, die das automatisch sicherstellt.

Routinen für den Z80 werden in der nächsten *LOOP* gezeigt, wie auch einfache Anwendungen mit der Baugruppe.

COL256 ist ab Ende Dezember 1985/Anfang Januar 1986 lieferbar. Das System COL256 besteht aus drei Baugruppen.

COL256A und COL256B sind jeweils 100 x 160 mm groß und belegen *zwei* Einbauplätze. COL256C verbindet oben die beiden Baugruppen und erzeugt die SCART und VIDEO-Signale.

Die Preise:

COL256H	Handbuch (HB)	10,–
COL256P	Drei Leiterplatten ohne HB	129,–
COL256B	Komplettbausatz mit HB, enthält drei Baugruppen	598,–
COL256F	Fertiggerät, geprüft, mit HB	748,–

(Fortsetzung folgt)

Dreharbeiten in Hamburg

Neue Folgen über den NDR-Computer fertiggestellt.

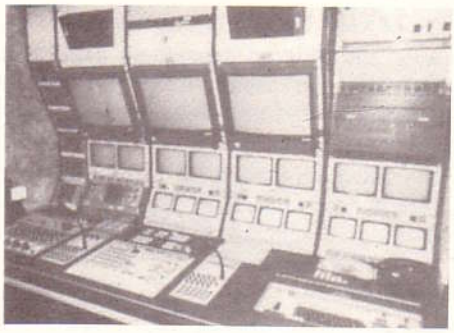
von Gerd Graf

In den Monaten Oktober und November fanden in Hamburg die Dreharbeiten über einige neue Folgen zum NDR-Computer statt.

Neu gezeigt werden:

Die ehemalige „Folge 0", 2 Folgen über das Arbeiten mit HEXMON mit Anwendung eines Lauflichtbeispieles und zwei Folgen über das Arbeiten mit SPS.

Es ist interessant, mit welchem Aufwand hier gearbeitet wurde. Innerhalb des großen Studios (Abmessungen nach meiner Schätzung ca. 40 x 40 Meter) waren ständig 20 bis 30 an den Dreharbeiten beschäftigte Personen da. Produziert wurde, wie auch die ursprünglichen Sendungen, direkt auf Videoband (MAZ). Vor dem Studio stand der neue, große Farbübertragungswagen des Studios Hamburg, den wir im Bild zeigen. Im FÜ1 saß die Bildregie und Bildtechnik, die Tonregie und Tontechnik. Das Ausgangssignal wurde an einen weiteren Wagen abgegeben, in dem die gesamten MAZ (magnetische Aufzeichnungs-Technik) Geräte standen. Von hier wurden auch bereits vorher gedrehte Teile zugespielt und eingemischt.



Der Farbübertragungswagen verfügt über fünf Beta-Cam-Kameras (Sony BVP-3), einen Bosch-Bildmischer mit 10 Eingängen, Bildquellenvorwahl mit Kreuzschiene, elektronische Maskierung und diversen Tricks. Über ein digitales Effektsystem, Gemini 2, können z.B. nichtsynchrone Bildsignale überblendet und Bildsignale eingefroren werden.

Wieder mit dabei ist Thomas Naumann, der sympatische Sprecher. Thomas ist keineswegs ein Mikrocomputer-Bastler – obwohl er durch seine Tätigkeit mit dem NDR-Computer dazu geworden ist, sondern Schauspieler mit Ausbildung an der Staatlichen Schauspielschule Hamburg. Derzeit spielt er am Hessischen Staatstheater in Wiesbaden „Troilus und Kressida".

Mehr Speed bei CP/M 2.2

Erhöhung der STEP-Rate für neue Laufwerke (TEAC1)

Wer das EFLOMON V 1.5b und das CP/M80 besitzt, wird sich bestimmt gewundert haben, daß die Floppy-Laufwerke relativ langsam positionieren. Das liegt daran, daß Rolf-Dieter Klein im EFLOMON bei der Floppy-Routine den langsamsten Wert für die Schrittmotore eingestellt hat. Es befinden sich zwar ein „SPEED.-COM" auf der Diskette, aber man muß es immer vorher laden, um die schnellste Steprate zu bekommen.

Mein Tip: Man liest sich das EFLOMON über den Promer z.B. auf Adresse 8000 ein und sucht sich die Zeile 9C58. Dort steht der Wert 03. Diesen Wert ändert man auf den Wert 00. Mit dem Wert 00 wird dann die schnellste Steprate eingestellt. Original EFLOMON gut aufbewahren!

1CB8

Michael Milzsch

Die Sendetermine des Bay. Fernsehens (B III)

ab 11. 1. 1986, Samstag, 17.15 Uhr

Der Bayerische Rundfunkt nannte uns auf Anfrage folgende (unverbindliche) Sendetermine:

11. 1. Schritt für Schritt (neue Folge)
18. 1. Die Stromversorgungen (4)
25. 1. Startlogik und Taktgenerator (5)
1. 2. Die Zentraleinheit (6)
8. 2. Dem Speicher auf der Spur (7)
15. 2. Hexmon (neu)
22. 2. Lauflichter (neu)
1. 3. Robotersteuerung (10)
8. 3. Schreiben lernen (Folge 11)

Die Sendetermine des NDR III – Am 6. 1. 1986 geht's los!

Rechner modular

Fernsehfolgen	Länge	Termine / Uhrzeit		
Schritt für Schritt	15'	6. 1. *16.45*	13. 1. *9.15*	16. 1. *9.15*
Die Spannungsversorgung	15'	14. 1. *16.30*	20. 1. *9.15*	23. 1. *9.15*
Startlogik und Taktgenerator	15'	21. 1. *17.00*	27. 1. *9.15*	30. 1. *9.15*
Die Zentraleinheit	15'	28. 1. *16.45*	3. 2. *9.30*	6. 2. *9.15*
Dem Speicher auf der Spur	15'	4. 2. *16.30*	10. 2. *9.30*	13. 2. *9.15*
HEMON	15'	10. 2. *17.05*	17. 2. *9.30*	20. 2. *9.15*
Lauflichter	15'	24. 2. *9.30*	27. 2. *9.15*	
Roboter steuern	15'	21. 2. *17.35*	3. 3. *9.30*	6. 3. *9.15*
Schreiben lernen	15'	27. 2. *17.00*	7. 3. *9.00*	10. 3. *9.30*

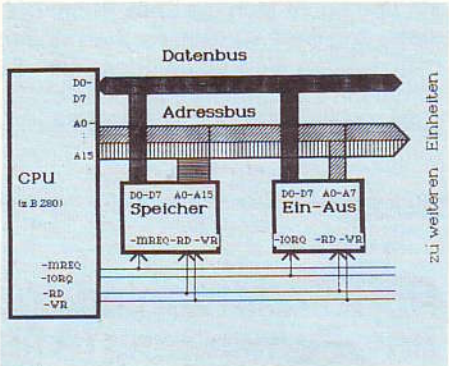
Steuern mit SPS

Fernsehfolgen	Länge	Termine / Uhrzeit		
Der programmierte Draht	15'	24. 4. *16.30*	29. 4. *11.15*	2. 5. *11.05*
Zeit-Zeichen	15'	29. 4. *17.35*	6. 5. *11.10*	9. 5. *11.05*

Für den Anfänger:

(Zweiter Teil)

In der letzten *„LOOP"* haben wir uns die IOE-Baugruppe etwas genauer angesehen. Zur weiteren Verdeutlichung erst einmal ein Blockschaltbild, wie das mit den Ein-/Ausgaben und den Adressen überhaupt prinzipiell funktioniert.



Wir erkennen drei funktionelle Blöcke
- die Zentraleinheit (CPU)
- den Speicher (Memory)
- die Ein-/Ausgaben (I/O)

CPU und Speicher sind für *jeden* Computer eine unbedingte Voraussetzung, auch falls die beiden Einheiten auf einem Chip untergebracht werden.

Ein Computer „läuft" dann auch ohne Ein-/Ausgaben, jedoch hat das wenig Sinn, da er ja nicht nur zum Selbstzweck funktionieren soll. Der minimalste Ein-/Ausgabeblock besteht z.B. aus einem Schalter (Eingabe) und einer Lampe (Ausgabe) – hat auch noch wenig Sinn, den Computer könnte man ja auch durch ein Stück Draht ersetzen. Die unterste sinnvolle Ausbaustufe haben wir z.B. beim Einsteigerpaket: eine einfache Tastatur und eine hexadezimale Anzeige.

Nun zu den im Bild eingezeichneten Verbindungen: zunächst erkennt man den Datenbus. Er ist hier 8 Bit „breit" – dies gilt für die CPU Z80 oder die CPU68008. Beim NDR-Computer ist der Datenbus 8, 16 oder 32 Bit breit. Wie das funktioniert, obwohl man auf der Bus-Leiterplatte nur die Signale D0 bis D7 findet, wird in einem späteren Artikel erläutert.

Die Pfeile deuten auf die Datenflußrichtung hin, die in beide Richtungen gehen muß. Warum? Na klar, die CPU muß Daten *ausgeben* können (z.B. an den Speicher oder an eine Ausgabe) und Daten *einlesen* können (vom Speicher oder einer Eingabe).

Damit es hier zu keinen Kollisionen kommt, regeln die Signale –RD (Read, lese) und –WR (Write, schreibe) die Richtung, immer von der CPU aus gesehen. Der „–" Strich vor dem Signalnamen bedeutet: Dieses Signal ist „Ø"-aktiv. Diese Signale werden an alle Einheiten geführt, die den Datenbus benützen, also alle Speicher, Ein-/Ausgabe-Einheiten und mehr.

Deswegen werden diese Signale, wie auch alle anderen Bussignale, meist (z.B. auf der Vollausbau-CPU mit einem 74LS245) verstärkt. Dies nennt man auch „Pufferung".

Die erwähnten Signale –WR und –RD ordnet man auch dem „Steuerbus" zu – dies jedoch nur, damit Sie den Begriff einmal gehört haben.

Nun ist im Bild noch ein ganz „dicker" Bus eingezeichnet: der Adressenbus. Irgendwie muß die CPU ja auseinanderhalten, wo sie ihre Daten hinschreibt oder von wo herholt. Die Breite des Adressenbusses ist von der gewählten CPU abhängig, beim Z80 ist er 16 Bit breit (A0 – A15) und kann damit $2^{16} = 65535 = 64$ K verschiedene Zustände annehmen. Bei der CPU 68008 ist er schon 20 Signale groß, der „Adressraum" ist hier also 2^{20} oder 1 MByte, der von dieser Baugruppe „direkt" adressiert werden kann.

Z80-Anwender, nicht verzweifeln – es gibt einen ganz einfachen Trick, um auch mit der Z80-CPU soviele Adressen zu erzeugen. Man verwendet dazu einfach einen Ausgabeport (z.B. 4 D-Flip-Flops), der an A16 bis A19 angeschlossen wird. Eine Ausgabe auf diesem Port ermöglicht der CPU „sich selber" die Adressen zu erhöhen.

Man spricht hier von Bank-Umschaltung, wobei der Speicher einfach in 64k Bänke aufgeteilt wurde. Na – funkt's schon? Richtig – auf der *Bank-Boot*-Baugruppe ist diese Bank-Umschalte-Logik (unter anderem) realisiert.

Beim Z80, und bei diesem Beispiel wollen wir bleiben, führt nun der Adressenbus an alle Speicher und *die unteren* 8 Bit des Adressenbusses an die Ein-/Ausgaben. Man kann also $2^8 = 256$ verschiedene Ein-/Ausgaben ansprechen.

Die CPU 68008 kann 65536 I/O-Adressen erzeugen (das macht sie automatisch, sobald auf den Adressbereich FxxxxH zugegriffen wird, siehe Sonderheft 2, der mc Architektur und Sprache, Seite 5 ff). Beim NDR-Bus werden davon jedoch wieder nur die unteren 8 Bit dekodiert.

Diese 256 Aus-/Eingabeadressen werden besonders beim modularen NDR-Computer bald knapp! Vergleichen Sie dazu bitte auch die Übersicht über belegte IO's, die immer wieder aktualisiert in *„LOOP"* veröffentlicht wird.

Dazu müssen wir allerdings etwas in die Hardware einsteigen. Nehmen Sie sich nun das Schaltbild der IOE, z.B. aus dem Handbuch, Seite 14, zur Hand.

Zur Adressdekodierung dient der Baustein 74LS139, verbunden mit dem 74LS85. Den 74LS85 mit den Bits A4 bis A7 haben wir bereits in *LOOP 5* erörtert. Die Signale A0 und A1 werden an den 74LS139 (J6) geführt. Dies ist ein einfacher 2Bit-Binärdekoder, der aus 2 Bit-Eingang (Binärsignal) 4 getrennte Ausgänge erzeugt, von denen immer nur einer aktiv (low) ist. Zwei solche Dekoder sind im Baustein enthalten. An beide Dekoder werden nun die Signale A0 und A1 geführt. Jeder Dekoder hat noch einen sogenannten Freigabe-Eingang (Enable). An den linken Dekoder wird über das Odergatter J8 das Signal READ geführt, an den rechten Dekoder das Signal WRITE. Über das Odergatter wird der Gleichausgang des Dekoders 74LS85 geführt. Also dekodieren die Dekoder nur dann, wenn:

– der 74LS85 „grünes Licht" gegeben hat, also die Adressen A4 bis A7 mit der Stellung der Jumperreihe JMP1 übereinstimmen

– der linke Dekoder dekodiert nur dann, wenn das Signal-RD (READ) aktiv ist.

– und der rechte Dekoder nur dann, wenn das Signal-WRITE (schreibe) aktiv ist.

Von den nun vier möglichen Zuständen werden auf der IOE nur zwei verbraucht. Die zwei Ausgänge des linken Dekoders (der, wie wir gesehen haben, fürs Lesen zuständig ist), werden an den Freigabeeingang (Pin 19) des Bausteins 74LS245 gelegt. Damit wird dieser Baustein freigegeben und die Daten, die an der IOE anstehen, werden auf den Datenbus gelesen. Der rechte Teil des Dekoders ist für die Ausgabe zuständig. Hier werden dann die D-Flip-Flops 74LS374 freigegeben.

Natürlich wird immer nur ein Flip-Flop freigegeben, abhängig von den Adressen A0 und A1. Da nur zwei Ausgänge (der vier möglichen) des Dekoders 139 verwendet werden, werden auch nur die Bitkombinationen ØØ und Ø1 der Adress-Bits A0 und A1 verwendet. Zwei weitere mögliche Bitkombinationen gehen hier verloren und können z.B. für Erweiterungen auf der IOE verwendet werden.

Viel schlimmer ist, daß auf der Baugruppe die Adress-Bits A2 und A3 gar nicht verwendet werden. Das heißt in der Praxis, daß die Bitkombinationen, die an diesen beiden Adressbeinchen anstehen, über-

haupt nicht ausgewertet werden. Nehmen wir einmal an, die IOE-Baugruppe ist auf die Adresse 3 0 eingestellt (z.B. für SPS). Schreiben Sie sich 3 0 hexadezimal einmal binär auf. Das sieht dann so aus: 0011 0000.

Wir haben festgestellt, daß A3 und A2 nicht ausgewertet werden. Die Adresse sieht nun so aus: 0011xx00.

Der Adressdekoder 139 verwendet zwei Adressen. Also wird sich die IOE-Baugruppe angesprochen fühlen, auf den Adressen 0011xx00 und 0011xx01. Da A2 und A3 nicht verwendet werden, kann xx für jede beliebige Binärkombination stehen.

Adresse ‚31H' ist:
0011 00 01 = 31H
0011 01 01 = 35H
0011 10 01 = 39H
0011 11 01 = 3DH

Dadurch werden also Adressen „verschenkt". Bei einer neueren IOE-Baugruppe werden wir alle 8 Adressen ausdekodieren. Es gibt heute schon sehr preisgünstige 8-Bit-Vergleicher, die dies leicht ermöglichen. Ein solcher Baustein ist z.B. der 74LS688.

Nun zur 2. Frage: was passiert, falls durch einen Programmfehler eine Ausgabe auf eine nicht dekodierte Adresse gemacht wird?

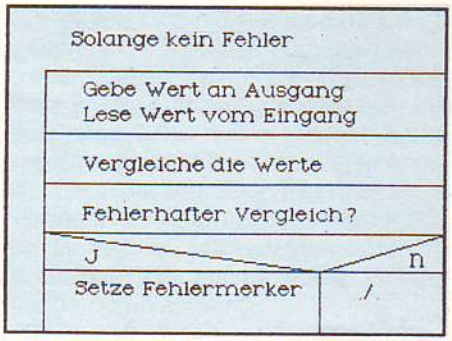
Der CPU ist dies egal. Ganz einfach, die Ausgabe geht verloren. Es wird ja von keiner Dekodierlogik irgendeine Hardware in Bewegung gesetzt, um diese Adresse zu dekodieren und im richtigen Moment die Daten vom Datenbus „aufzufangen". Die Daten tröpfeln also nicht irgendwo hinten aus Ihrem Computer heraus, sondern waren einfach kurzzeitig auf dem Datenbus – niemand hat sich für sie interessiert – und damit hat sich's.

Nun zur nächsten Frage: die Daten beim „In"-Befehl gelangen beim Z80 wie auch beim 8080 oder 8085 immer in den Akkumulator (Register A). Von dort müssen sie vom Programm weiterverarbeitet werden. Dies ist auch ganz sinnvoll, weil man mit dem Akkumulator bei diesen CPU's sehr viel machen kann.

Beim 68008 ist dies wesentlich eleganter gelöst. Hier gibt es nur den generellen MOVE-Befehl und damit kann man die Daten in jedes Register, das einem gerade angenehm erscheint, direkt einlesen.

Nun, wie könnte man eine IOE-Karte einfach prüfen?

Ganz einfach, man verbindet die Eingänge mit den Ausgängen. Ein kleines Testprogramm wird dann so aussehen:



Damit ist der Anfängerartikel Port und die IOE-Baugruppe abgeschlossen. Wir würden uns freuen, die Meinung der Anfänger zu hören, ob Ihnen Artikel dieser Art überhaupt etwas bringen. Die Profis unter Ihnen werden diesen Artikel ja nicht gelesen haben – oder doch?
Aus diesem Artikel geht auch hervor, daß in der Mikroelektronik, besonders wenn es um Ein-/Ausgaben geht, grundlegende Hardware-Kenntnisse unbedingt erforderlich sind. Es gibt also keinen „Nur-Programmierer", der sich überhaupt nicht um Hardware kümmern muß. Eine Ausbildung in dieser Richtung wollen wir ja mit dem offenen NDR-Computer erreichen. Ein Literaturhinweis für Anfänger, die sich mit integrierten Bausteinen beschäftigen wollen: Das Standardwerk ist die Pocket-Guide von von Texas-Instruments, in der die wichtigsten TTL-IC's detailliert erläutert sind.

LOOP für Einsteiger — LOOP für Einsteiger

Das Programmieren von Tabellen

von Andreas Kruse
8014 Neubiberg

Programmieren zum Arbeiten mit Tabellen

Wenn man Programme schreibt, durch die eine größere Anzahl Werte in ein Register (Sound, GDP bei Direktsprache, DA usw.) oder an eine Adresse (IOE usw.) geschrieben werden, möchte man vermeiden, immer wieder denselben Ablauf – (Lade Wert, Gebe Wert an Adresse aus, – Springe auf Verzögerungs UPR) zu programmieren. Die Programme werden so unübersichtlich, zu lang und es macht Mühe, die Werte zu ändern, da man ein längeres Programm durchgehen muß, um vielleicht den Inhalt jeder achten Speicherzeile zu löschen. Abhilfe schafft hier ein Programm, das eine oder mehrere verschiedene Schleifen durchläuft und nacheinander Werte aus einer Tabelle abruft, die an einem beliebigen Platz des Speicherbereichs stehen kann. Die nächste Seite zeigt ein Beipielprogramm für ein Acht-Kanal-Lauflicht, das bis zu 2 hoch 8 = 256 Werte nacheinander ausgeben kann. Auf der IOE-Karte (Adr.: 30h) werden 8 LED nach dem Beispiel der Ampelsteuerung angeschlossen. (s. 1. Sonderheft, S. 42). Das Programm nutzt die Möglichkeiten der indireken Adressierung. Die Adresse der Tabelle wird abgespeichert und bei jedem Schleifendurchlauf um 1 erhöht. Den aktuellen Wert des „Tabellen-Adress-Zählers" lädt man ins Register HL und erhält dann über die indirekte Adressierung den Inhalt der Speicherzelle, deren Adresse in HL steht, im Akku. Dann gibt man diesen Wert aus und ruft (in diesem Beispiel) das „Warte" Unterprogramm der Ampelsteuerung auf. Dann erfolgt der nächste Schleifendurchlauf für den nächsten Wert.

Das Programm läßt sich für die verschiedensten Fälle umschreiben und erweitern. Man kann z.B. in einer Schleife mehrere Werte abfragen und an verschiedene Register (z.B. SOUND-Generator) senden. Persönlich habe ich das Programm hauptsächlich verwendet, um beim Z80-Grundprogramm umfangreichere Figuren mit Kurzvektoren zu zeichnen. Dabei ließ sich auch eingeschränkt bewegliche Grafik realisieren. Im Z80-Grundprogramm läßt sich das Programm selbstverständlich unter Verwendung der „Schleife" und „Endschleife" Befehle vereinfachen.

```
                        TITLE   8 Kanal Lauflicht
                        ;
                        ORG     8100H
                        ;
                        Hauptprogramm:
8100  21 00 82                   LD      HL,8200H        ;Adresse der
                                                          ;Tabelle laden
8103  22 FE 81                   LD      (81FEH),HL      ;und auf
                                                          ;Tabellen-
                                                          ;adresszähler
                                                          ;speichern
8106  3E 10                     LD      A,10H           ;Anzahl
8108  32 FD 81                   LD      (81FDH),A       ;Schleifen-
                                                          ;durchläufe =16
                                                          ;abspeichern
810B  3A FD 81                   LD      A,(81FDH)       ;Start der
                                                          ;Schleife
810E  D6 01                     SUB     1               ;(Schleifen-
                                                          ;zähler Haupt-
                                                          ;programm) -1
8110  32 FA 81                   LD      (81FAH),A       ;abspeichern
8113  2A FE 81                   LD      HL,(81FEH)      ;Wert Tabellen-
                                                          ;adresszähler
                                                          ;laden
8116  7E                        LD      A,(HL)          ;zugehörigen
                                                          ;Wert
                                                          ;in den Akku
8117  D3 30                     OUT     (30H),A         ;ausgeben auf
                                                          ;der IOE
8119  23                        INC     HL              ;Wert Tabellen-
                                                          ;adresszähler +1
811A  22 FE 81                   LD      (81FEH),HL      ;abspeichern
811D  CD 30 81                   CALL    8130H           ;Unterprogramm
                                                          ;aufrufen
8120  C2 0B 81                   JP      NZ,810BH        ;Ende der
                                                          ;Schleife
8123  C3 00 81                   JP      8100H           ;Sprung an Start
                        ;
                        ;
                        Unterprogramm:                  ;(Warteschleife)
8126  3E C8                     LD      A,0C8H          ;äussere Schlei-
                                                          ;fe 200 mal
8128  32 FB 81                   LD      (81FBH),A       ;Zähler 1
812B  3E FF                     LD      A,0FFH          ;innere Schleife
                                                          ;255 mal
812D  32 FC 81                   LD      (81FCH),A       ;Zähler 2
8130  3A FC 81                   LD      A,(81FCH)       ;Zähler 2 innen
8133  D6 01                     SUB     1
8135  32 FC 81                   LD      (81FCH),A       ;-1
8138  C2 3A 81                   JP      NZ,813AH        ;bis Null
813B  3A FB 81                   LD      A,(81FBH)       ;Aussen Zähler 1
813E  D6 01                     SUB     1
8140  32 FC 81                   LD      (81FCH),A
8143  C2 35 81                   JP      NZ,8135H        ;innen Zähler
                                                          ;neu belegen
8146  C9                        RET                     ;zurück ins
                                                          ;Hauptprogramm
                        ;
                        ;       Folgende Zwischenspeicher
                        ;       freihalten !
                        ;
                        ;       81FBH   Schleifenzähler 1 UPR
                        ;       81FCH   Schleifenzähler 2 UPR
                        ;       81FDH   Schleifenzähler HPR
                        ;       81FEH   +
                        ;       81FFH   16 Bit Tabellen-
                        ;               adresszähler
                        ;
                        ;
                        ;       Tabelle:
                        ;
                        ;               LED  1 2 3 4 5 6 7 8
                        ;  8200 01           x
                        ;  8201 02             x
                        ;  8202 04               x
                        ;  8203 08                 x
                        ;  8204 10                   x
                        ;  8205 20                     x
                        ;  8206 40                       x
                        ;  8207 80                         x
                        ;  8208 03           x x
                        ;  8209 06             x x
                        ;  820A 0C               x x
                        ;  820B 18                 x x
                        ;  820C 30                   x x
                        ;  820D 60                     x x
                        ;  820E C0                       x x
                        ;  820F 80                         x
```

8100 HAUPTPROGRAMM

8126 UNTERPROGRAMM

```
no fatal error(s)

SUM=2353
CRC=4E08
```

Von einem der auszog, den Umgang mit Transistoren zu lernen

Erfahrungen mit dem Kompaktkurs Elektronik

von Mike C. Hayduk

Sicher – nach vorhandenen Schaltplänen hatte man ja schon (teils mit Erfolg) diese oder jene Schaltungen zusammengebastelt, eventuell auch selbst eine Platine geätzt. Trotzdem: Die tieferen Geheimnisse jener verschlungenen Stromlaufpläne blieben im Verborgenen. Hier Abhilfe zu schaffen ist das Anliegen des Christiani Kompaktkurses „Elektronik".

Nach dem Auspacken des Kurses (kompakt = 3,5 kg Material und Literatur) weicht die Begeisterung einer kurzen Phase der Verwirrung, bis man die beigefügten Erläuterungen geordnet hat, zuunterst kommt schließlich das Begrüßungsschreiben zum Vorschein. Etwas erschreckt wird man auch von der Bestellkarte für Zusatzgeräte (Preisvergleich einholen!), die aber in den meisten „Elektronikhaushalten" schon vorhanden sein dürften.

Insgesamt gelungen ist meines Erachtens die Einführung mittels Tonbandkassette und einer sogenannten „Flipchart" (ein neuer Schritt nach vorn in der deutsch-englischen Sprachvermählung). Wenngleich man bei Unterbrechungen und beim Zurückblättern etwas Schwierigkeiten hat, Band und „Flipchart" wieder zu synchronisieren, da keine Seitenzahlen auf der Kassette genannt werden. Die als Zeichen zum Umblättern gespielte „Musik" ging mir

schnell „auf den Wecker", aber über Geschmack läßt sich bekanntlich nicht streiten.

Der Aufbau des Experimentiermaterials (gute Baubeschreibung – man lernt sogar über das Löten noch etwas Neues!) nimmt so ungefähr ein verregnetes Wochenende in Anspruch – Unterbrechungen durch Ehefrau, Kinder und/oder Freunde eingerechnet.

Doch nun endlich zum eigentlichen Kursus, der „Lernarbeit" – ohne die geht es nämlich auch bei Christiani nicht, wenn auch „mundgerecht" aufbereitet.

Der Kurs beginnt sehr elementar: beim richtigen Messen und dem Umgang mit dem Vielfachinstrument. Von Anfang an wird konsequent und „professionell" das Diagramm, das Meßprotokoll mitgenommen und erläutert. (Hält man sich daran, ist die Gefahr gering, über einen Abschnitt mit der Bemerkung „kann ich" hinwegzuhuschen.) So wird die Verbindung Theorie – Praxis immer wieder erneuert und aufrecht erhalten.

Über das Aufnehmen von Kennlinien, von Widerständen und Dioden gelangt man zum Verständnis von Schaltungen wie Einweg- oder Brückengleichrichter, Schaltungen mit Zehnerdioden usw.

Große Frustration bereitete das Übersehen einer Druckfehlerberichtigung, die zum Nichtfunktionieren eines Versuches führte, der ganz am Anfang steht (Diodenkennlinie). Der Hinweis darauf sollte sicher auffälliger erfolgen.

Gefallen hat mir besonders, daß man sich bei Christiani nicht scheut, zu vereinfachenden Erläuterungen zu greifen, um komplizierte Verhältnisse auch dem Nicht-Physiker „begreifbar" zu machen (z.B. Verständnis der Transistorfunktion).

Der gesamte Lehrstoff ist in zwei große Teile gegliedert: im ersten Teil werden Bauelemente wie Widerstände, Kondensatoren etc. bis zur Diode abgehandelt. Der zweite (größere) Teil befaßt sich im wesentlichen mit dem dreibeinigen Zentralelement, der gesamten Elektronik, dem Transistor, kleinere Ausflüge führen in die Optoelektronik, in die Temperaturmessung usw.

Arbeitet man den Kurs ganz systematisch und sorgfältig durch, kann man sich geraume Zeit mit der Elektronik beschäftigen, ist dann allerdings auch in der

Lage, einfache Berechnungen zu einer Schaltung durchzuführen, wobei hierzu nicht Hochschulmathematik gefordert wird.

Ergebnis: am Ende sieht man im Transistor nicht mehr das „große Unbekannte mit 3 Anschlüssen", sondern ein universell nützliches und verstehbares Bauelement der Elektronik. Zusammen mit dem qualitativ und quantitativ guten Übungsmaterial, das ja immer für eigene und weitere Experimente zur Verfügung steht: Eine lohnende Investition für Theorie und Praxis.

Der Kompaktkurs „Elektronik" ist bei GES ab Lager lieferbar. Preis: DM 248,–, Bestellcode CELEK.

Vom Grundprogramm (Z80) zu ZEAT

Bank-Boot als Speicherkarte mit Kopierfunktion

von Uwe Koch,
Frankenstraße 25,
5880 Lüdenscheid

Heute schreibe ich Ihnen, da ich glaube, etwas Interessantes für die „Nichtprofis" (wie ich) unter den Nachbauern des NDR-Klein-Computers zu haben. Dieser Beitrag richtet sich an alle, die ihren NKC mit Vollausbau-CPU-Z80 und R0A64, aber noch ohne ZEAT oder CP/M betreiben, so wie ich.

Wenn man also schon, je nach Ausbaustufe, über 1200 DM für sein Hobby, ausgegeben hat, dann muß mancher doch erst einmal richtig sparen, um sich so tolle Dinge wie ZEAT oder CP/M leisten zu können. Daher möchte ich solchen Nachbauern empfehlen, in kleinen preisgünstigen Schritten weiter zu machen, ohne auch nur vorübergehend etwas überflüssiges in seinem Rechner zu haben. Wenn man schon die R0A64 mit Grundprogramm und eventuell GOSI, BASIC, Assembler und bis zu 32 kByte RAM hat, dann kann man als ersten Schritt in Richtung ZEAT die Bankboot-Karte kaufen. Denn die Bankboot-Karte ist zunächst einmal eine normale Speicherkarte. Wenn man die voreingestellte Brücke von „16" auf „64" umstellt, hat man Platz für vier 2764-Eproms, also z.B. Grundprogramm, GOSI, BASIC und Assembler. Auf der R0A64 beläßt man die 6264-RAMs und wenn man bei R0A und BANKBOOT die Stecker für die Adressleitungen A16 – A19 offen läßt, funktioniert der Rechner wieder ganz normal.

Damit wären die Hardware-Voraussetzungen für ZEAT geschaffen. Als nächsten kleinen Schritt kauft man sich später die inzwischen recht preiswert gewordenen 6264-RAMs bis man auf der R0A die 64 kByte beisammen hat. Um die unteren 32 Kbyte bis zum Erwerb der ZEAT-EPROMs nicht ungenutzt zu lassen, kann man folgende Routine eingeben.

Nach dem Einschalten muß das Grundprogramm erscheinen. Die nachfolgende Routine bewirkt den Transport aller vier Eproms der BANKBOOT auf die Speicherkarte. Anschließend springt das Programm auf Adresse 0000h der Speicherkarte. Der Transport erfolgt in zwei Portionen von je 32 Kbyte. Hierbei werden jeweils zwei Eproms in den Ram-Speicher A000h kopiert, umgeschaltet auf die Speicherkarte und wieder auf die Ursprungsadresse transportiert.

Das Programm ‚COPY' kopiert den Eprombereich von der Bankboot-Karte in zwei Portionen zu je 16 Kbyte, zunächst in das RAM ab 0C000h auf der R0A und schaltet mit dem OUT-Befehl die Bankboot aus und die unteren 32 k RAM ein, um dann den Bereich von 0C000h wieder auf seine Ursprungsaddresse zu kopieren (diesmal auf der R0A).

Als letztes wird die Adresse 0000h angesprungen, also das Grundprogramm gestartet. Man hat nun die System-Software im RAM stehen und kann zum Beispiel statt GOSI meine Systemerweiterung (Softcharakter, Beep, Renumber, Renew, Figur usw.) von Kassette laden. Vorallem kann man aber nun Korrekturen an der System-Software vornehmen. So sollte man z.B. beim EGOSI2 V1.1 folgende Daten ändern:
3E17 LD HL, 8600h ⟶ LD HL,9000h;
Grenze Namen ⟷ Programme

3E20 LD HL,8FFFh ⟶ LD HL,0FFFFh;
3E38 LD HL,8FFFh ⟶ LD HL,0FFFFh;
3FA3 LD SP,8FFFh ⟶ LD SP,0FFFFh;
Obergrenze des RAM-Bereiches

Dadurch kann man unter GOSI die vollen 32 Kbyte RAM ausnutzen, (etwa 28700 Byte zum Lernen und 3250 Byte für Namen) und nicht nur die 44 Byte in der Originalversion.

Mit der Routine ‚Back' kann man ebenso wie durch einen Reset, wieder in den Eprombereich kommen. Wenn sich im unteren RAM-Bereich schon das Grundprogramm befindet, kann man auch durch „I0 setzen", „Adresse C8", „Data 80" in den RAM-Bereich umschalten.

Ich hoffe, daß auch Sie der Ansicht sind, diese Tips könnten zum schrittweisen Erweitern des NDR-Computers nützlich sein und vielleicht manche *LOOP*-Leser interessieren.

```
                                TITLE     Transport-Routine


                                org       8800h

8800  21 00 00                  ld        hl,0000h        ;von
8803  11 00 A0                  ld        de,0a000h       ;nach
8806  01 FF 3F                  ld        bc,3fffh        ;wieviel
8809  ED B0                     ldir                      ;transport

880B  3E 80                     ld        a,80h           ;umschaltung
880D  D3 C8                     out       (0c8h),a        ;auf R0A 64

880F  21 00 A0                  ld        hl,0a000h       ;von
8812  11 00 00                  ld        de,0000h        ;nach
8815  01 FF 3F                  ld        bc,3fffh        ;wieviel
8818  ED B0                     ldir                      ;transport

881A  3E 00                     ld        a,00h           ;umschaltung
881C  D3 C8                     out       (0c8h),a        ;auf BANKBOOT

881E  21 00 40                  ld        hl,4000h        ;von
8821  11 00 A0                  ld        de,0a000h       ;nach
8824  01 FF 3F                  ld        bc,3fffh        ;wieviel
8827  ED B0                     ldir                      ;transport

8829  3E 80                     ld        a,80h           ;umschaltung
882B  D3 C8                     out       (0c8h),a        ;auf R0A 64

882D  21 00 A0                  ld        hl,0a000h       ;von
8830  11 00 40                  ld        de,4000h        ;nach
8833  01 FF 3F                  ld        bc,3fffh        ;wieviel
8836  ED B0                     ldir                      ;transport

8838  C3 00 00                  jp        0000h           ;Grundprogramm
                                                          ;auf R0A 64


                                end

no fatal error(s)
```

LOCATE für EBASIC

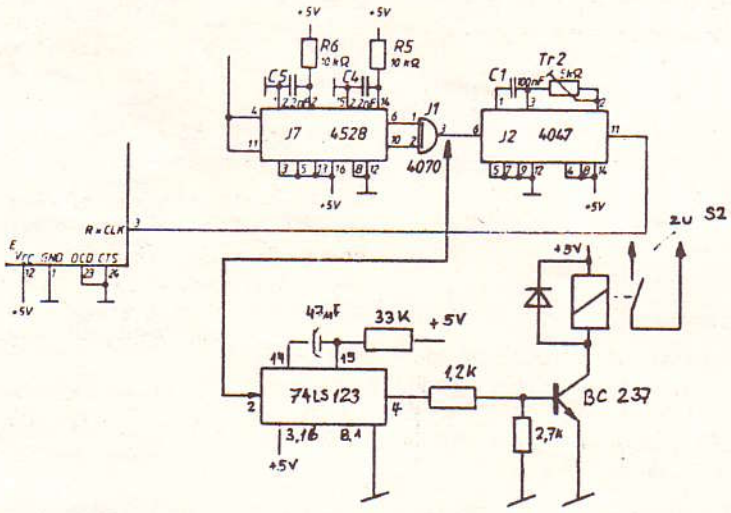
von Dietmar Arnds

Eigentlich ist das Programm kein Programm, vielmehr eine Procedure. Diese Procedure bildet die in vielen BASIC Dialekten bekannte Funktion Locate Zeile, Spalte nach. Die Funktion Locate ist wohl auch unter Namen wie PRINT AT,PRINT „Klammeraffe" oder aber HTAB,VTAB geläufig. Es stellt sich ja doch immer wieder das Problem der tabellenorientierten Eingabe. Nun, wenn man sich beim Aufruf der Procedure auf die Zeilen 1 – 23 beschränkt, kann diese dem Problem der Eingabe Zeile, Spalte ein Ende bereiten. Geht man auf Zeile 24 oder 25, was möglich ist, ergibt sich wohl ein Scroll nach oben. Zeilen größer 25 oder aber Spalten größer 80 werden als Fehler ausgewiesen. Weitere Informationen sind dem Programm als Kommentar beigegeben.

```
PROGRAM LOCATE( INPUT,OUTPUT );
VAR AZEILE,ZEILE,SPALTE:INTEGER;
            EINGABE:REAL;
(*VOR DEM ERSTEN AUFRUF DER PROCEDURE MUSS AZEILE FESTGELEGT WERDEN.DIES IST  *)
(*DIE AKTUELLE CURSORPOSITION.GESCHRIEBEN WIRD IMMER MIT WRITELN.DA NACH      *)
(*JEDER PROCEDURE EINMAL NACH OBEN GESPRUNGEN WIRD.DENN READ UND READLN HABEN *)
(*GRUNDSAETZLICH EIN LINE FEED UND CARRIAGE RETURN ZUR FOLGE.DER CURSOR STEHT *)
(*NACH JEDER EINGABE AM ZEILENANFANG DER NAECHSTEN ZEILE.DAS MACHT AUCH EINE  *)
(*GETRENNTE FUNKTION RECHTS LINKS UEBERFLUESSIG *)
PROCEDURE LOCATE (ZEILE,SPALTE:INTEGER);
VAR N,DZEILE:INTEGER;
 BEGIN
     DZEILE:=ZEILE-AZEILE;(* DIFFERENZ=NEUE ZEILE-ALTE ZEILE *)
     (* NACH OBEN *)
     IF (DZEILE<0) AND (DZEILE>-25) THEN BEGIN FOR N:=AZEILE-1 DOWNTO ZEILE DO
        WRITE(CHR(11));
        AZEILE:=ZEILE;
     END;
     (* NACH UNTEN *)
     IF (DZEILE>0) AND (DZEILE<25) THEN BEGIN FOR N:=AZEILE TO ZEILE-1 DO
        WRITE(CHR(10));
        AZEILE:=ZEILE;
     END;
     (* FEHLER  BEREICH UEBERSCHRITTEN *)
     IF (DZEILE<-25) OR (DZEILE>=25) THEN WRITE('SUBSCRIPT OUT OF RANGE ');
     WRITE(CHR(11));
     (* NACH RECHTS ODER LINKS *)
     IF SPALTE<=80 THEN BEGIN
             FOR N:=1 TO SPALTE-1 DO
                    WRITE(CHR(9));
     END;
     (* FEHLER  BEREICH UEBERSCHRITTEN *)
     IF SPALTE>80 THEN WRITE('SUBSCRIPT OUT OF RANGE ');
END;
(* ENDE DER PROCEDURE  -----  H A U P T P R O G R A M M  -----          *)
BEGIN
WRITELN(CHR(1),'E SCLRSCREEN ');(* CLEARSCREEN *)
WRITELN(CHR(1),'E SSIZE ',17);(* 80 ZEICHEN   *)
WRITELN('                          LOCATEFUNKTION');
  AZEILE:=1;
  REPEAT
    LOCATE (5,1);
     WRITELN('ERSTE EINGABE  :');
    LOCATE (5,17);
     READ(EINGABE);
    LOCATE (6,1);
     WRITELN('ZWEITE EINGABE :');
    LOCATE (6,17);
     READ(EINGABE);
    LOCATE (5,30);
     WRITELN('DRITTE EINGABE :');
    LOCATE (5,46);
     READ(EINGABE);
    LOCATE (6,30);
     WRITELN('VIERTE EINGABE :');
    LOCATE (6,46);
     READ(EINGABE);
    UNTIL EINGABE=0
END.
```

Anmerkung der LOOP-Redaktion: Danke für diese Prozedur! Es geht aber auch einfacher: mit ESC = X Y kann man direkt positionieren. Also: **WRITE(chr(27),'=',chr(zeile+32),chr(spalte+32))**

Automatisches Umschalten von Aufnahme zu Wiedergabe – CAS



Ausgang 4 (Q) des nachtriggerbaren Monoflops bringt über den Transistor das Relais zum Anziehen. Normalstellung und „Aufnahme" schließt über den Relaiskontakt S2.

Bei Wiedergabe wird das Monoflop ständig nachgetriggert, sodaß das Relais abfällt und Schalter 2 öffnet.

Heinz Amgwerd,
Rebbergstraße 13 a,
CH-561/ Wohlen

Was ist beim Starten des Betriebsprogramms ZEAT zu beachten?

Sobald Sie das Menue auf dem Bildschirm haben, müssen Sie ca. 5 Sekunden warten, ehe Sie mit der Eingabe von „3" ZEAT aufrufen. Diese Pause sollten Sie einhalten. Ansonsten besteht die Möglichkeit, daß beim Austesten eines Programms mit dem Aufrufen von Trace das System aussteigt. Der Fehler kommt vom Interrupt-Signal der Floppy-Controller-Karte.

Falls Sie ohne Floppies arbeiten, tritt der Fehler nicht auf. Dann brauchen Sie auch die 5 Sekunden Pause nicht einzuhalten.

Achtung: Der Anschluß des Akustikkopplers und der Zugriff auf eine Datenbank unter ZEAT funktioniert ebenfalls nur bei unterbrochenem Interrupt (oder Ziehen der FLO2). An der Fehlerbehebung wird gearbeitet.

Der Splitter

Rolf-Dieter Klein

Wenn man einen 68000 besitzt, so ist es nötig, zwischen geraden (even) und ungeraden (odd) Adressen zu unterscheiden. Denn der Bus ist in zwei Hälften geteilt. Die eine Hälfte enthält alle geraden, die andere alle ungeraden Speicherzellen.

```
* Split-Routine  fuer EPROM-Programmierung
* beim 68000.
* Rolf-Dieter Klein

quelle   equ $10000   * Quell-Adresse
zieleven equ $14000   * Ziel Even-Eprom
zielodd  equ $16000   * Ziel Odd-Eprom

laenge equ $4000      * 16K-Quelle

start:
 lea quelle,a0        * dort Maschinenprog.
 lea zieleven,a1
 lea zielodd,a2
 move #laenge,d1      * Anzahl Bytes
schleife:
 move.b (a0)+,(a1)+ * even
 move.b (a0)+,(a2)+ * odd
 sub.w #2,d1          * je wort
 bne schleife         * bis alles aufgeteilt.
 rts

Textstart=009000 Fenster=009000 Tor=009000     aber CTRL-J=Hilfe
```

Wenn man nun EPROMs brennen will, so muß man Programme zunächst in gerade und ungerade Hälften teilen. Man benötigt also immer mindestens zwei EPROMs, die man später z.B. auf je eine R0A64 stecken kann, ein EPROM auf die R0A64 für gerade Adressen und eines auf die für ungerade Adressen.

Bild 1 zeigt ein Beispiel für ein Programm, das die Aufteilung durchführt. Das Programm zerlegt einen 16KByte Adress-Bereich in zwei 8K-Bereiche mit dem jeweils geraden und ungeraden Anteil. Hier steht die Quelle auf Adresse $10000. Nach dem Aufruf von Start liegen auf Adresse $14000 die Daten für das gerade (even) EPROM und ab Adresse $16000 die Daten für das ungerade (odd) EPROM. Danach ruft man das EPROM-Programmier-Menue auf und programmiert für jeden dieser Bereiche ein EPROM. Auf diese Weise kann man auch einfach Programme, die für den 68008 (in EPROMs) gedacht waren, auf dem 68000 zum Laufen bringen. So zum Beispiel GOSI und PASCAL/S. Das gilt jedoch nur für Programme, die keinen Zugriff auf IO-Ports durchführen, denn diese Programme müssen modifiziert werden. Alle IO-Adressen werden nämlich beim 68000 mit 2 multipliziert und zusammenhängende IO-Bereiche kommen so z.B. auf gerade oder ungerade Adressen.

Achtung, wenn Sie selbst Programme für den 68008 schreiben, versuchen Sie IO-Adressen zu vermeiden, indem Sie die fertigen Unterprogramme aus dem Grundprogramm verwenden. Dann bleiben Sie kompatibel.

Das Grundprogramm und den Formatierer kann man aus diesem Grunde mit dem Splitter nicht aufteilen.

Den Splitter benötigen Sie immer, wenn Sie auf den 68000 EPROMs programmieren.

Wenn Sie nicht genügend Speicher haben (hier sind 32K Byte mind. erforderlich), so können Sie das Programm auch stückchenweise splittern und beim Programmieren nur Teile dazuprogrammieren.

```
; ***************************************************
; *                                                 *
; *            TRANSPORT ROUTINE                    *
; *                                                 *
; * MIT DIESER KLEINEN ROUTINE IST ES MöGLICH MIT   *
; * DEM ZEAT-BETRIEBSSYSTEM PROGRAMME ZU ERSTELLEN, *
; * WELCHE AUF DIE ROUTINEN DES GRUNDPROGRAMMS ZU-  *
; * RUECKGREIFEN.                                   *
; * NACH DEM ASSEMBLIEREN WIRD DAS GANZE PROGRAMM   *
; * MIT DEM BEFEHL "RUN 100H" AUF DIE BANK E KO-    *
; * PIERT, WO SICH DAS EPROM MIT DEM GRUNDPROGRAMM  *
; * BEFINDET.                                       *
; * DIE SPRUNGADRESSEN WERDEN VOM ASSEMBLER RICH-   *
; * TIG  BERECHNET, FALLS DER PROGRAMMSTART AB      *
; * DER ADRESSE 8800H (ORG 8800H) FESTGESETZT WIRD. *
; * SOLLTE DAS PROGRAMM EINEN GRöSSEREN RAM-        *
; * BEREICH ALS 10 KBYTE BENöTIGEN, SO MÜSSEN       *
; * VOR DEM ASSEMBLIEREN MIT DEM BEFEHL "PROFI ON"  *
; * DIE GESCHÜTZTEN SYSTEMADRESSEN FREIGEGEBEN      *
; * WERDEN.                                         *
; * NACH DEM ABLAUF DES PROGRAMMS KEHRT MAN MIT RE- *
; * SET ZUM FLOMOM ZURÜCK UND WÄHLT DEN MENÜPUNKT   *
; * 2. ES MELDET SICH DAS GRUNDPROGRAMM. DAS PRO-   *
; * GRAMM KANN NUN AB ADRESSE 8800H GESTARTET WER-  *
; * DEN.                                            *
; *                                                 *
; ***************************************************

;
SYSTEM  EQU     00005H
;
;       DEFINITION FÜR SYSTEMVARIABLE
;
WSTARTF EQU     00000H
;
;
;       DEFINITION DER VARIABLEN
;
AQ      EQU     08800H          ;ANFANGSADRESSE IN
                                ;DER QUELLBANK
EA      EQU     0EF00H          ;ENDADRESSE DES
                                ;DATENBLOCKS
AZ      EQU     08800H          ;ANFANGSADRESSE IN
                                ;DER ZIELBANK
NQ      EQU     00H             ;NUMMER DER
                                ;QUELLBANK
NZ      EQU     0EH             ;NUMMER DER
                                ;ZIELBANK
FL      EQU     0F05BH          ;FLOMON BANK-
                                ;EINSPRUNG
;
;
        ORG     0100H           ;DAS PROGRAMM
                                ;BEGINNT
;                               ;BEI 0100H
LADEN:
        LD      HL,AQ           ;LADE ANFANGS-
                                ;ADRESSE
                                ;DER QUELLBANK
        LD      DE,AZ           ;LADE ANFANGS-
                                ;ADRESSE
                                ;DER ZIELBANK
TRANSPORT:
        LD      C,NQ            ;LADE NAME DER
                                ;QUELLBANK
        LD      B,NZ            ;LADE NAME DER
                                ;ZIELBANK
        CALL    FL              ;RUFE BANK
        LD      BC,0080H        ;LADE BC,128D
                                ;DAS UNTERPROGRAMM
                                ;BANK VERSCHIEBT
                                ;DIE DATEN IN
                                ;128 BYTE BLöCKEN
ADDITION:
        EX      DE,HL           ;ADDITION
        ADD     HL,BC           ;DE + 128D
        EX      DE,HL
        ADD     HL,BC           ;ADDITION
                                ;HL + 128D
        LD      BC,EA           ;DATEN
        LD      A,B;            ;TRANSPORT
        XOR     H               ;ABGESCHLOSSEN?
        JP      NZ,TRANSPORT
        JP      WSTARTF
;
; * ACHTUNG:                                       *
; * AUF DER JEWEILIGEN BANK MUSS VON ADRESSE F000 *
; * BIS FFFF AUF JEDEN FALL EIN RAM-SPEICHER      *
; * SEIN, SONST WIRD EIN CARRY ALS ERGEBNIS GE-   *
; * LIEFERT.                                      *
; *                                               *
; ***************************************************
```

Hilfsprogramme für den 68008

von Rüdiger Bäcker

Wer hat das nicht schon erlebt, man möchte ein Programm oder Daten im Speicher verschieben oder als Hex- oder ASCII-Dump auf Bildschirm oder Drucker ausgeben oder Speicherbereiche vergleichen oder, oder, oder . . .

All diese kleinen Probleme lassen sich mit Hilfsprogrammen leicht lösen. Doch woher nehmen und nicht stehlen?

Da sich auch für mich diese Frage stellte, habe ich mich einmal hingesetzt und ein paar Routinen für diese Zwecke geschrieben. Die dabei entstandenen Routinen werde ich hier in der *LOOP* so nach und nach vorstellen.

Die erste Routine aus dieser Reihe stelle ich in dieser Ausgabe vor, es ist das Programm „RUBADUMP", mit dem es möglich ist, beliebige Speicherbereiche auf dem Bildschirm oder dem Drucker auszugeben. Ausgegeben werden dabei die Adresse, der Inhalt in HEX (eigentlich heißt es ja sedezimal) und als ASCII-Zeichen. Da nicht alle Speicherinhalte aus druckbaren ASCII-Zeichen bestehen, werden die nicht druckbaren Speicherinhalte als Punkt dargestellt.

Das Programm ist relokativ und kann aus der Bibliothek aufgerufen werden. Nachdem das Programm mit ‚J' gestartet wurde, kann die Start- und die Endadresse in HEX eingegeben werden. Bild 1 zeigt den Bilschirm nach der Eingabe. Drückt man den ‚CR', so werden die Speicherinhalte ausgegeben. Auf dem Bildschirm sieht das so aus, wie in Bild 2. Die Ausgabe der Zeichen erfolgt über die Routine C02. Da man dabei zwischen Bildschirm- und Druckerausgabe umschalten kann, ist ein Ausdruck auf dem Drucker leicht möglich. Sollte man jedoch einen seriellen Drucker besitzen, so ist überall dort im Programm, wo jetzt C02 steht, S0 einzutragen. Dann muß jedoch die serielle Schnittstelle durch eine andere Routine initialisiert werden (siehe auch SER-Handbuch). Die Umwandlung HEX → ASCII erfolgt in der Routine über eine Tabelle, in der die entsprechenden ASCII-Codes stehen. Es wird vom HEX-Wert jeweils ein Halbbyte abgetrennt. Dazu wird der Wert in D2 mit einer entsprechenden Maske UND-verknüpft. Die Leistungsfähigkeit der Adressierungsarten des 68008 wird dann ausgenutzt, um den entsprechenden Wert in der Tabelle auszuwählen.

Da das Programm die C02-Routine benutzt, ist die Start-/Stopp-Funktion mit CONTROL S & CONTROL Q möglich. Nach der Ausgabe gelangt man mit ‚m' wieder in das Grundprogramm zurück.

In *LOOP 7* erscheint das Programm ‚RUBATRANS 68', mit dem Daten und Programme beliebig verschoben werden können.

```
000400                  ORG $400
000400
000400                  *       R U B A D U M P
000400                  *
000400                  *       PROGRAMM ZUR AUSGABE VON SPEICHERBEREICHEN IN HEX + ASCII
000400                  *       AUF DEM BILDSCHIRM ODER DEM DRUCKER
000400                  *
000400                  *       COPYRIGHT (C) 1985 BY RUEDIGER BAECKER - POSTFACH 4111 - 5820 GEVELSBERG
000400
000400                  KOPF:
000400  55AA0180        DC.L $55AA0180
000404  5275626164756D  DC.B 'Rubadump'
00040B  70
00040C  00000020        DC.L RUBADUMP-KOPF
000410  000002E4        DC.L ENDEA-KOPF
000414  01              DC.B 1
000415  00 00 00        DC.B 0,0,0
000418  00000000        DC.L 0,0
00041C  00000000
000420
= 00000070              ZEILENZ         EQU     $70             * BUFFER FUER ZEILENZAHLER
= 000000F4              ZIELADR         EQU     $F4
= FFFFFF68              TAST            EQU     $FFFFFF68
000420
000420                  RUBADUMP:                       * TEXTE AUSGEBEN
000420  3E3C 0011       MOVE #!CLPG,D7
000424  4E41            TRAP #1
000426  4239 FFFFFF69   CLR.B TAST+1
00042C  4280            CLR.L D0
00042E  4281            CLR.L D1
000430  4282            CLR.L D2
000432  4283            CLR.L D3
000434  1B7C 0000 0070  MOVE.B #0,ZEILENZ(A5)
00043A
00043A  41FA 0272       LEA DUTEXT1(PC),A0
00043E  103C 0042       MOVE.B #$42,D0
000442  123C 0000       MOVE.B #$0,D1
000446  143C 00C8       MOVE.B #200,D2
00044A  3E3C 000A       MOVE #!WRITE,D7
00044E  4E41            TRAP #1
000450
000450  0442 001E       SUB#30,D2
000454  103C 0032       MOVE.B #$32,D0
000458  41FA 025D       LEA TEXT2(PC),A0
00045C  3E3C 000A       MOVE #!WRITE,D7
000460  4E41            TRAP #1
000462
000462  0442 0028       SUB #40,D2
000466  103C 0021       MOVE.B #$21,D0
00046A  41FA 0268       LEA DUTEXT2(PC),A0
00046E  3E3C 000A       MOVE #!WRITE,D7
000472  4E41            TRAP #1
000474
000474  0442 0014       SUB #20,D2
000478  41FA 0262       LEA DUTEXT3(PC),A0
00047C  3E3C 000A       MOVE #!WRITE,D7
000480  4E41            TRAP #1
000482
000482                  GETWD1:                         * START- & ENDADRESSE HOLEN
000482  103C 0021       MOVE.B #$21,D0
000486  143C 00B2       MOVE.B #130,D2
00048A  123C 00FA       MOVE.B #250,D1
00048E  163C 0009       MOVE.B #9,D3
000492  41ED 00F4       LEA ZIELADR(A5),A0
000496  3E3C 000B       MOVE #!READ,D7
00049A  4E41            TRAP #1
00049C  41ED 00F4       LEA ZIELADR(A5),A0
0004A0  1B10            MOVE.B (A0),D4
0004A2  0C04 006D       CMP.B #'m',D4
0004A6  6700 01F4       BEQ ENDDUMP
0004AA  3E3C 001D       MOVE #!WERT,D7
0004AE  4E41            TRAP #1
0004B0  2B40            MOVEA.L D0,A4
0004B2
0004B2                  GETWD2:
0004B2  143C 006E       MOVE.B #110,D2
0004B6  103C 0021       MOVE.B #$21,D0
0004BA  123C 00FA       MOVE.B #250,D1
0004BE  41ED 00F4       LEA ZIELADR(A5),A0
0004C2  3E3C 000B       MOVE #!READ,D7
0004C6  4E41            TRAP #1
0004C8  41ED 00F4       LEA ZIELADR(A5),A0
0004CC  3E3C 001D       MOVE #!WERT,D7
0004D0  4E41            TRAP #1
0004D2. 2A40            MOVEA.L D0,A5
0004D4
0004D4  9BCC            SUBA.L A4,A5
0004D6  220D            MOVE.L A5,D1
0004D8  204C            MOVEA.L A4,A0
0004DA  45FA 01C2       LEA ANFTAB(PC),A2
0004DE  4282            CLR.L D2
0004E0  4280            CLR.L D0
0004E2  3E3C 0014       MOVE #!CLRSCREEN,D7
0004E6  4E41            TRAP #1
0004E8  6100 014E       BSR LHEAD
0004EC  1B3C 0004       MOVE.B #4,D4
0004F0  2B48 00F4       MOVE.L A0,ZIELADR(A5)
0004F4  6100 009E       BSR ADRLOOP
0004F8  49ED 00F4       LEA ZIELADR(A5),A4
0004FC  193C 0010       MOVE.B #16,D4
000500                  LOOP:
000500  1410            MOVE.B (A0),D2
000502  6100 0102       BSR PUTASCII
000506  0202 00F0       AND.B #$F0,D2
00050A  E80A            LSR.B #4,D2
00050C  1032 2000       MOVE.B 0(A2,D2),D0
000510  3E3C 0021       MOVE #!C02,D7
000514  4E41            TRAP #1
000516  1418            MOVE.B (A0)+,D2
000518  0202 000F       AND.B #$0F,D2
00051C  1032 2000       MOVE.B 0(A2,D2),D0
000520  3E3C 0021       MOVE #!C02,D7
000524  4E41            TRAP #1
000526  103C 0020       MOVE.B #' ',D0
00052A  3E3C 0021       MOVE #!C02,D7
00052E  4E41            TRAP #1
000530  0444 0001       SUB #1,D4
000534  0C04 0000       CMP.B #0,D4
000538  6700 000B       BEQ LF
00053C                  LFR:
00053C  51C9 FFC2       DBRA D1,LOOP
000540  4E75            RTS
000542                  LF:
000542  103C 0020       MOVE.B #' ',D0
000546  3E3C 0021       MOVE #!C02,D7
00054A  4E41            TRAP #1
00054C  6100 00D2       BSR GETASCII
000550  4280            CLR.L D0
000552  102D 0070       MOVE.B ZEILENZ(A5),D0
000556  0C00 0044       CMP.B #68,D0
00055A  6700 001A       BEQ NEXTPP
00055E                  PSCHLPP:
00055E  6100 011A       BSR HEND
000562  2B48 00F4       MOVE.L A0,ZIELADR(A5)
000566  1B3C 0004       MOVE.B #4,D4
00056A  6100 0028       BSR ADRLOOP
00056E  183C 0010       MOVE.B #16,D4
000572  6000 FFC8       BRA LFR
000576
000576                  NEXTPP:
000576  1B7C 0000 0070  MOVE.B #0,ZEILENZ(A5)
00057C  103C 000A       MOVE.B #$0A,D0
000580  4283            CLR.L D3
000582  163C 0003       MOVE.B #4-1,D3
000586                  NPP1A:
000586  3E3C 0021       MOVE #!C02,D7
00058A  4E41            TRAP #1
00058C  51CB FFF8       DBRA D3,NPP1A
000590  6000 FFCC       BRA PSCHLPP
000594
000594                  ADRLOOP:
000594  48E7 2080       MOVEM.L A0/D2,-(A7)
000598  49ED 00F4       LEA ZIELADR(A5),A4
00059C  41ED 00F4       LEA ZIELADR(A5),A0
0005A0                  WL:
0005A0  1410            MOVE.B (A0),D2
0005A2  0202 00F0       AND.B #$F0,D2
0005A6  E80A            LSR.B #4,D2
0005AB  1032 2000       MOVE.B 0(A2,D2),D0
0005AC  3E3C 0021       MOVE #!C02,D7
0005B0  4E41            TRAP #1
0005B2  1418            MOVE.B (A0)+,D2
0005B4  0202 000F       AND.B #$0F,D2
0005B8  1032 2000       MOVE.B 0(A2,D2),D0
0005BC  3E3C 0021       MOVE #!C02,D7
0005C0  4E41            TRAP #1
0005C2  0444 0001       SUB #1,D4
0005C6  0C04 0000       CMP.B #0,D4
0005CA  6600 FFD4       BNE WL
0005CE  103C 0020       MOVE.B #' ',D0
0005D2  3E3C 0021       MOVE #!C02,D7
0005D6  4E41            TRAP #1
0005D8  103C 002D       MOVE.B #'-',D0
0005DC  3E3C 0021       MOVE #!C02,D7
0005E0  4E41            TRAP #1
0005E2  103C 002D       MOVE.B #'-',D0
0005E6  3E3C 0021       MOVE #!C02,D7
0005EA  4E41            TRAP #1
0005EC  103C 003E       MOVE.B #'>',D0
0005F0  3E3C 0021       MOVE #!C02,D7
0005F4  4E41            TRAP #1
0005F6  103C 0020       MOVE.B #' ',D0
0005FA  3E3C 0021       MOVE #!C02,D7
0005FE  4E41            TRAP #1
000600  4CDF 0104       MOVEM.L (A7)+, D2/A0
000604  4E75            RTS
000606
000606
000606                  PUTASCII:
000606  0C02 0020       CMP.B #$20,D2
00060A  6D00 000E       BLT PUNKT
```

```
000060E  0C02 007F      CMP.B #$7F,D2
000612  6E00 0006      BGT PUNKT
000616  18C2           MOVE.B D2,(A4)+
000618  4E75           RTS
00061A
00061A                 PUNKT:
00061A  18FC 002E      MOVE.B #'.',(A4)+
00061E  4E75           RTS
000620
000620                 GETASCII:
000620  49ED 00F4      LEA ZIELADR(A5),A4
000624  4285           CLR.L D5
000626  1A3C 000F      MOVE.B #15,D5
00062A                 GETLOOP:
00062A  101C           MOVE.B (A4)+,D0
00062C  3E3C 0021      MOVE #!CO2,D7
000630  4E41           TRAP #1
000632  51CD FFF6      DBRA D5,GETLOOP
000636  4E75           RTS
000638
000638                 LHEAD:
000638  48E7 0080      MOVEM.L A0,-(A7)
00063C  41FA 0070      LEA DUTEXT1(PC),A0
000640                 HSCHL:
000640  1018           MOVE.B (A0)+,D0
000642  0C00 0000      CMP.B #0,D0
000646  6700 000C      BEQ COPYR
00064A  3E3C 0021      MOVE #!CO2,D7
00064E  4E41           TRAP #1
000650  6000 FFEE      BRA HSCHL
000654
000654                 COPYR:
000654  103C 0020      MOVE.B #' ',D0
000658  3E3C 0021      MOVE #!CO2,D7
00065C  4E41           TRAP #1
00065E  41FA 0057      LEA TEXT2(PC),A0
000662                 COPYR1:
000662  1018           MOVE.B (A0)+,D0
000664  0C00 0000      CMP.B #0,D0
000668  6700 000C      BEQ HENDD
00066C  3E3C 0021      MOVE #!CO2,D7
000670  4E41           TRAP #1
000672  6000 FFEE      BRA COPYR1
000676                 HENDD:
000676  4CDF 0100      MOVEM.L (A7)+,A0
00067A                 HEND:
00067A  103C 000D      MOVE.B #$D,D0
00067E  3E3C 0021      MOVE #!CO2,D7
000682  4E41           TRAP #1
000684  103C 000A      MOVE.B #$A,D0
000688  3E3C 0021      MOVE #!CO2,D7
00068C  4E41           TRAP #1
00068E  102D 0070      MOVE.B ZEILENZ(A5),D0
000692  0640 0001      ADD #1,D0
000696  1B40 0070      MOVE.B D0,ZEILENZ(A5)
00069A  4E75           RTS
00069C
00069C                 ENDDUMP:
00069C  4E75           RTS
00069E
00069E                 ANFTAB:
00069E  30 31 32 33 34  DC.B $30,$31,$32,$33,$34,$35,$36,$37,$38,$39,$41,$42,$43,$44,$45,$46
0006A3  35 36 37 38 39
0006A8  41 42 43 44 45
0006AD  46
0006AE                 DUTEXT1:
0006AE  5275626164756D  DC.B 'Rubadump',0
0006B5  7000
0006B7                 TEXT2:
0006B7  28432920313938  DC.B '(C) 1985 by Ruediger Baecker',0
0006BE  35206279205275
0006C5  65646967657220
0006CC  426165636865726
0006D3  00
0006D4                 DUTEXT2:
0006D4  766F6E202D2D3E  DC.B 'von -->',0
0006DB  00
0006DC                 DUTEXT3:
0006DC  626973202D2D3E  DC.B 'bis -->',0
0006E3  00
0006E4                 ENDEA:
0006E4                 END.

0E8DF6  Ende-Symboltabelle
```

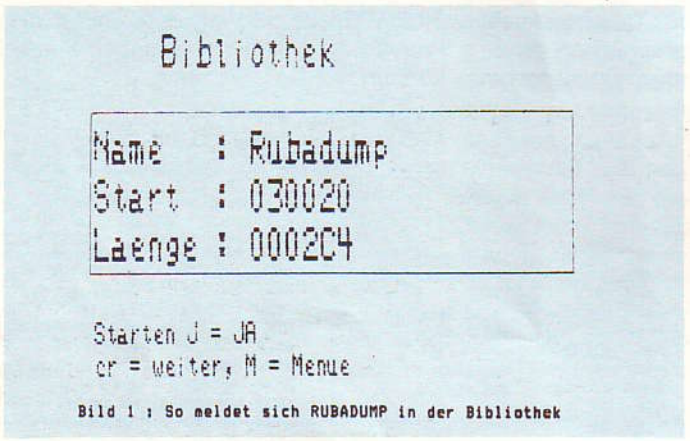


Bild 1 : So meldet sich RUBADUMP in der Bibliothek

Bild 2 : Die Eingabe der Datenbereiche erfolgt in HEX

```
Rubadump (C) 1985 by Ruediger Baecker
00000400 --> 55 AA 01 80 52 75 62 61 64 75 6D 70 00 00 00 20  U...Rubadump...
00000410 --> 00 00 02 E4 01 00 00 00 00 00 00 00 00 00 00 00  ................
00000420 --> 3E 3C 00 11 4E 41 42 39 FF FF FF 69 42 80 42 81  X..NAB9...iB.B.
00000430 --> 42 82 42 83 1B 7C 00 00 00 70 41 FA 02 72 10 3C  B.B..|...pA..r.<
00000440 --> 00 42 12 3C 00 00 14 3C 00 C8 3E 3C 00 0A 4E 41  .B.<...<..X..NA
00000450 --> 04 42 00 1E 10 3C 00 32 41 FA 02 50 3E 3C 00 0A  .B...<.2A..)X..
00000460 --> 4E 41 04 42 00 28 10 3C 00 21 41 FA 02 68 3E 3C  NA.B.(.<.!A..hX
00000470 --> 00 0A 4E 41 04 42 00 14 41 FA 02 62 3E 3C 00 0A  ..NA.B..A..bX..
00000480 --> 4E 41 10 3C 00 21 14 3C 00 82 12 3C 00 FA 16 3C  NA.<.!.<...<...<
00000490 --> 00 09 41 ED 00 F4 3E 3C 00 0B 4E 41 41 ED 00 F4  ..A...X..NAA...
000004A0 --> 18 10 0C 04 00 60 67 00 01 F4 3E 3C 00 1D 4E 41  .....`g...X..NA
000004B0 --> 28 40 14 3C 00 6E 10 3C 00 21 12 3C 00 FA 41 ED  (@.<.n.<.!.<..A.
000004C0 --> 00 F4 3E 3C 00 0B 4E 41 41 ED 00 F4 3E 3C 00 1D  ..X..NAA...X..
000004D0 --> 4E 41 2A 40 9B CC 22 0D 20 4C 45 FA 01 C2 42 82  NA*@..". LE...B.
000004E0 --> 42 80 3E 3C 00 14 4E 41 61 00 01 4E 18 3C 00 04  B.X..NAa..N.<..
000004F0 --> 28 48 00 F4 61 00 00 9E 49 ED 00 F4 18 3C 00 10  (H..a...I....<..
00000500 --> 14 10 61 00 01 02 02 02 00 F0 E8 0A 10 32 20 00  ..a..........2 .
00000510 --> 3E 3C 00 21 4E 41 14 18 02 02 00 0F 10 32 20 00  X.!NA.......2 .
00000520 --> 3E 3C 00 21 4E 41 10 3C 00 20 3E 3C 00 21 4E 41  X.!NA.< X.!NA
00000530 --> 04 44 00 01 0C 04 00 00 67 00 00 08 51 C9 FF C2  .D......g...Q...
00000540 --> 4E 75 10 3C 00 20 3E 3C 00 21 4E 41 61 00 00 D2  Nu.< X.!NAa...
00000550 --> 42 80 10 20 00 70 0C 00 00 44 67 00 00 1A 61 00  B..-.p...Dg...a.
00000560 --> 01 1A 28 4
```

Bild 3 : Eine Ausgabe am Bildschirm

Kaltstart für EFLOMON

```
.z80
aseg
flo equ 0f000h      ; Startadresse im EFlomon
org 0100h
start:              ; Programm Starten
call flo            ; und ausfuehren

 end

; Dieses kleine Programm fuehrt dazu,
; das sich das EFlomon mit einen Kaltstart
; wieder meldet. Das Programm kann man
; dafuer gebrauchen, wenn z.B. das Grund-
; programm auf der Bankboot auf Adresse 2000H
; liegt, und man ein Programm auf Casette
; abspeichern moechte.
```

Linken mit dem Grundprogramm

Rolf-Dieter Klein

Wenn man mehrere getrennt erstellte Assemblerprogramme hat, und möchte diese zu einem großen Programm zusammenbinden, so gibt es normalerweise zwei Möglichkeiten: entweder man kopiert die Quellen alle zusammen und erhält einen sehr großen Quelltext, oder man verwendet einen Linking-Assembler. Die erste Methode ist nicht sehr praktisch, da entweder die neue Gesamtquelle zu groß für den Speicher ist, oder weil man in den unabhängig erstellten Assemblerteilen die gleichen Marken verwendet hat. Sehr beliebt ist z.B der Name LOOP oder Schleife, und der Assembler meldet dann doppelt definierte Marken.

Besser ist die zweite Möglichkeit. Bei einem Linking-Assembler benötigt man normalerweise ein zusätzliches Programm mit dem Namen LINKER (siehe CP/M und C). Wenn man den Assembler des Grundprogramms verwendet, geht das einfacher. Er besitzt eine sogenannte globale Symboltabelle. Das heißt, die Symboltabelle bleibt auch nach der Übersetzung erhalten und wird bei weiteren Übersetzungen in der aktuellen Form verwendet. Dadurch ist es möglich, Programme getrennt zu übersetzen und dennoch Bezüge untereinander zu haben.

```
;
* Hauptprogramm
* als erstes Programm uebersetzen
*

start:
 move #90,d0
 jsr @drehe
 bsr quadrat   * beim ersten uebersetzen undefiniert
 move #100,d0
 jsr @schreite
 move #-180,d0
 jsr @drehe
 bsr dreieck   * beim ersten uebersetzen undefiniert
 rts

ende1

 end


Textstart=009000 Fenster=009000 Tor=009000      aner CTRL-J=Hilfe
```

An einem Beispiel soll gezeigt werden, wie man so etwas macht. Bild 1 zeigt das Hauptprogramm. Es ruft zwei im Hauptprogramm nicht definierte Unterprogramme, QUADRAT und DREIECK auf. Wenn man das Programm übersetzt, so erhält man zwei Fehler vom Assembler gemeldet, denn die Symbole QUADRAT und DREIECK sind noch nicht definiert. Das macht aber gar nichts, denn nun soll das Unterprogramm getrennt dazu übersetzt werden, Bild 2 zeigt das Programm. Es enthält die Definition von QUADRAT. Damit nun das Unterprogramm nicht an die gleiche Stelle gelegt wird, wie das Hauptprogramm, wird eine ORG-Anweisung verwendet. Als Adresse wird dazu ENDE angegeben. Dieser Name ist im Hauptprogramm definiert worden, und zwar als letzte Marke am Programm, die damit den nächsten noch nicht belegten Speicherplatz angibt. Das Unterprogramm QUADRAT wird also dahinter gelegt. Dann wird die Marke ENDE2 definiert, die das Ende des Unterprogramms QUADRAT angibt.

```
;
* Unterprogramm 1
* als zweites uebersetzen

 org ende   * damit neue Code-Bereich

quadrat:
 move #4-1,d3
lp1:
 move #100,d0
 jsr @schreite
 move #90,d0
 jsr @drehe
 dbra d3,lp1
 rts

ende2:     * fuer Unterprogramm 2

 end


Textstart=009000 Fenster=009000 Tor=009000      aner CTRL-J=Hilfe
```

```
;
* Unterprogramm 2
* als drittes uebersetzen

 org ende2   * Hinter Uprg2

dreieck:
 move #3-1,d3
lp1:
 move #100,d0
 jsr @schreite
 move #120,d0
 jsr @drehe
 dbra d3,lp1
 rts

ende3:   * ggf weitere Programme
 end


Textstart=009000 Fenster=009000 Tor=009000      aner CTRL-J=Hilfe
```

Nun kann man den dritten Teil, den Bild 3 zeigt, übersetzen. Er beginnt nun bei der Adresse ENDE2 und wird von da an im Speicher abgelegt. Die Übersetzung der Unterprogramme darf zu keinem Fehler führen, denn alle Symbole im Unterprogramm sind bei uns definiert. Nun muß das Hauptprogramm als letztes nochmals übersetzt werden. Diesmal darf auch hier kein Fehler mehr erscheinen. Dann kann man das Hauptprogramm bei der Marke Start starten und das Bild eines Hauses muß auf dem Bildschirm erscheinen, wenn alles richtig gemacht wurde.

Die Unterprogramme könnten natürlich auch selbst Aufrufe in andere Programme enthalten, die erst später definiert werden, dann muß einfach jedes Programm insgesamt zweimal übersetzt werden. Wichtig ist dabei aber, daß man zunächst alle Programme in einer bestimmten Reihenfolge aufruft, und beim zweiten Mal in genau der gleichen Reihenfolge. Ferner ist auch die Reihenfolge der Unterprogramme nicht beliebig, wenn man die ORG-Anweisung so verwendet wie es hier getan wurde, denn die Adressen der ORG-Anweisung müssen unbedingt vor der Übersetzung definiert sein. Doch dies kann man ja einmal festlegen. Damit man bei jeder Änderung einer der Programme nicht alle Programme neu übersetzen muß, empfiehlt es sich, beim Test zunächst feste Adressen bei der ORG-Anweisung zu verwenden, und die Abstände der Programme groß genug zueinander zu halten, z.B. UPRG1 bei ORG $10000 und UPRG2 bei $11000, und das Hauptprogramm bei $A000 o.ä.

Eine Marke kommt in den Unterprogrammen doppelt vor: LP1, dies führt jedoch nicht zu einem Fehler, da man in jedem Teilprogramm beliebige Namen verwenden kann. Nur die Namen für die gegenseitigen Bezüge dürfen nicht doppelt vorkommen.

Grundprogramm 4.2 – 4.3

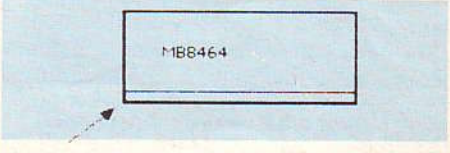
Achtung, es gibt immer noch einige Kunden, die noch die Version 4.2 des Grundprogramms besitzen. Gegenüber der Version 4.3 funktioniert aber dort der Befehl DATEN SPEICHERN und DATEN LESEN auf Kassette nicht. Wer noch die alte Version besitzt, sollte sie nun umtauschen.

Fehler im Handbuch CPU68000

Im Bestückungsplan (S. 12) ist der Widerstand R5 (4,5k) über IC4 zu weit nach rechts gerutscht. Er muß um einen Lötpunkt nach links versetzt werden, kommt also mit einem Anschluß zwischen die Kondensatoren C2 und C5. Im Foto der fertig bestückten Baugruppe (Bild 16, Seite 15 und Bild 17, Seite 16 des Handbuches) ist die *richtige* Lage zu erkennen.

Pin 1 gesucht

In letzter Zeit stellten wir fest, daß unsere Kunden mit dem RAM-Baustein (R8) MB8464 Probleme mit der Bezeichnung von PIN 1 haben. Der PIN 1 ist mit einer Nut auf der Oberfläche gekennzeichnet. In der folgenden Zeichnung ist ersichtlich, wo der PIN 1 des Bausteins sitzt.



An dieser Stelle befindet sich der PIN 1

Fehler im PASCAL/S

Rolf-Dieter Klein

Ein Fehler, der in der Version 3.1 von PASCAL/S steckt, hat vielleicht manchem Kopfzerbrechen gemacht: Wenn man beim Scarlaren-Datentyp in einer IF-Anweisung o.ä. eine Variable von diesem Typ direkt mit einer Konstanten vergleicht, wird ein Syntax-Fehler gemeldet.

Beispiel:
IF WERT = EINS THEN . . .
Wobei
WERT : (EINS, ZWEI, DREI);

Abhilfe schafft eine zusätzliche Hilfsvariable, in die man die Konstante zuweist und dann die Abfrage durchführt, wie es Bild 1 zeigt.

Keine Abhilfe schafft die Abfrage IF ORD(WERT)=ORD(EINS) THEN . . ., denn die Konstante wird hier nicht als Aufzähltyp erkannt, folglich bleibt auch ORD() undefiniert. Dies gilt aber nur bei Vergleichen. Man kann übrigens auch je einer INTEGER-Variablen mit ORD() den Wert zuweisen und dann den Vergleich ausführen.

Der Fehler steckt schon im Original-PASCAL/S und blieb daher unentdeckt. In Zeile 854 des PASCAL-Listings steckt schon eine Ungenauigkeit. Dort steht

```
program test(input,output);

type zahlen = (eins,zwei,drei);
var
 wert : zahlen;
 hilfswert : zahlen;

begin
 wert := eins;
 hilfswert := eins;
 if wert=hilfswert then writeln('ok');
end.
```

```
Textstart=009000 Fenster=009000 Tor=009000      aber CTRL-J=Hilfe
```

TAB[0].ref := t. Das kann aber nicht sein, gemeint war wohl TAB[t0].ref :=t. Dies führt jedoch zu einem Wert von .ref ungleich Null, was bei Konstanten ungewöhnlich ist. In Zeile 1296 wird nun x.ref auf 0 gesetzt auch bei Scalaren, die Konstante sind, damit muß der Vergleich bei 1482 (x.ref=y.ref) immer fehlschlagen, wenn man Konstante und Variable vom Typ Scalar miteinander vergleicht. Bei der Zuweiung passiert nichts, denn dort wird bei „stantyps" in Zeile 1534 der Vergleich von .ref nicht durchgeführt.

Abhilfe könnte z.B. die Entfernung derAbfrage in 1482 (x.ref=y.ref) schaffen, da nur die Fehlerbehandlung abgeschwächt wird und sonst keine Folgen zu befürchten sind. Im Assembler-Listing 3.1 sind dies die Zeilen 1903 bis 1907 (Adresse $14CE bis $14E5 relativ). Man kann sie einfach durch NOP ersetzen.

Eine neue Revision ist geplant, dauert jedoch noch eine Weile, da die vorgeschlagene Änderung nicht sauber ist und nach einem besseren Weg gesucht wird, der auch den Fehler mit Scalaren in Records (siehe Listing-Documentation) beseitigt.

Wer übrigens noch solche Fehler kennt, möge sie uns bitte nennen.

Einführung in C Teil 3

Rolf-Dieter Klein

Heute wollen wir uns einmal mit rekursiven Unterprogrammen beschäftigen. Rekursiv bedeutet, daß sich Unterprogramme selbst aufrufen können.

Nicht alle Sprachen erlauben es, rekursive Porgramme zu konstruieren. In BASIC ist das zum Beispiel nicht (oder nur sehr schwer) möglich.

Aufgabe ist es, einen Baum zu zeichnen. Dieser soll aus einem Stamm bestehen, an dem sich Äste befinden. Dabei soll die Graphik mit Hilfe der Unterprogramme des Grundprogramms ausgeführt werden.

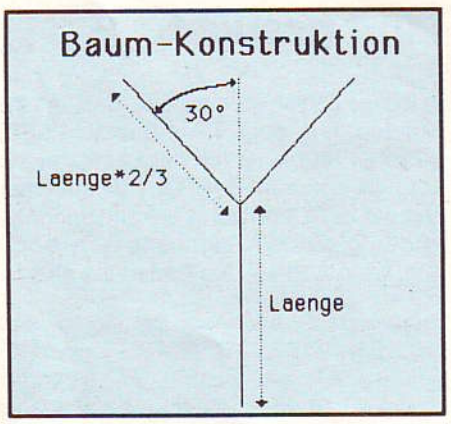


Bild 1 zeigt die Konstruktionsvorlage.

Der Stamm sei „Länge" lang. Am Stamm sollen zwei Äste gewachsen sein, die jeweils „Länge * 2/3" lang sind. Die Äste schließen einen Winkel von je 30 Grad zur Stammrichtung ein. Damit daraus ein Baum entsteht, soll jeder Ast wieder eine Baum-Struktur haben, also der Stamm eines weiteren Baumes sein.

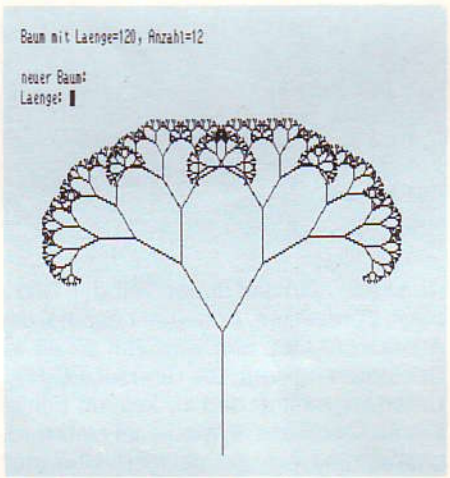


Bild 2 zeigt ein vollständiges Beispiel.

Zur Lösung der Aufgabe ist es am Besten, sich die wichtigsten Unterprogramme des Grundprogramms als Assembler-Teil vorzubereiten. Dies sind SCHREITE und DREHE. Mann könnte die Unterprogramme auch direkt von C aus aufrufen, jedoch ginge das auf Kosten der Geschwindigkeit.

Bild 3 zeigt das Assemblerlisting. (Siehe auch *LOOP* 4 und 5).

Das C-Programm ist in Bild 4 dargestellt.

Das Unterprogramm trägt den Namen Baum. Baum besitzt zwei Parameter, nämlich die Länge und eine Anzahl. Länge gibt die Länge des Stammes an und Anzahl die Anzahl der Verästelungen.

```
 1: * Assemblerprogramme fuer Turtle-Geometrie
 2:
 3: .globl _initturt,_schreite,_drehe
 4:
 5: _initturt:    * initturt()
 6:  movem.l d3-d7/a3-a6,-(a7)
 7:  move #23,d0  * Grundprog. Start
 8:  trap #3      * ueber BIOS
 9:  move.l a4,grund
10:  movem.l (a7)+,d3-d7/a3-a6
11:  rts
12:
13: _schreite:  * schreite(schrittzahl)
14:  link a6,#0
15:  movem.l d3-d7/a3-a6,-(a7)
16:  move.w 8(a6),d0
17:  move #1,d7   * schreite
18:  movea.l grund,a0
19:  jsr $420(a0) * Grundprogramm
20:  movem.l (a7)+,d3-d7/a3-a6
21:  unlk a6
22:  rts
23:
24: _drehe:   * drehe(winkel)
25:  link a6,#0
26:  movem.l d3-d7/a3-a6,-(a7)
27:  move.w 8(a6),d0  * Winkel als Prameter
28:  move #2,d7   * Drehe
29:  movea.l grund,a0
30:  jsr $420(a0) * Grundprogramm
31:  movem.l (a7)+,d3-d7/a3-a6
32:  unlk a6
33:  rts
34:
35: err: rts  * falls init vergessen
36:
37: grund: dc.l err
38:
39:  end
40:
41:
```

Im Unterprogramm wird als erstes geprüft, ob Anzahl größer als Null ist. Wenn nein, so wird kein Befehl ausgeführt und das Unterprogramm gleich wieder verlassen. Dies ist der Fall, wenn keine Äste vorhanden sind (dann gibt es auch keinen Stamm). Wenn Anzahl größer als Null ist, so wird zunächst der Stamm gezeichnet. Dies geschieht mit der Anweisung Schreite. Dann wird eine Drehung um 30 Grad durchgeführt. Dahin soll der weitere Teilbaum kommen. Dazu wird BAUM erneut, also rekursiv aufgerufen. Dabei wird

```
1:
2: /* recursive Routinen in C
3:    Beispiel:  Baum ausgeben
4:    Rolf-Dieter Klein 851128  */
5:
6: #include <stdio.h>
7:
8: EXTERN initturt();
9: EXTERN schreite();
10: EXTERN drehe();
11:
12: baum(laenge,anzahl)
13: int laenge,anzahl;
14: {
15:  if (anzahl > 0) {
16:   schreite(laenge);
17:   drehe(30);
18:   baum((laenge*2)/3,anzahl-1);
19:   drehe(-60);
20:   baum((laenge*2)/3,anzahl-1);
21:   drehe(30);
22:   schreite(-laenge);
23:  }
24: }
25:
26: main()
27: {
28:  int laenge,anzahl;
29:  char buffer[80]; /* fuer gets */
30:  initturt(); /* hier wird Grundprog. Adresse bestimmt */
31:  printf("\001 E @clrscreen\n");
32:  printf("\001 E @setflip 0 0\n\001 E @newpage 0 0\n\001 E @grapoff\n");
33:  do {
34:   printf("\nneuer Baum:\n");
35:   printf("Laenge: "); gets(buffer); sscanf(buffer,"%d",&laenge); printf("\n");
36:   printf("Anzahl: "); gets(buffer); sscanf(buffer,"%d",&anzahl); printf("\n");
37:   printf("\001 E @clrscreen\n");
38:   printf("Baum mit Laenge=%d, Anzahl=%d\n",laenge,anzahl);
39:   printf("\001 E @hebe\n\001 E @auf=y 0 255 0\n\001 E @senke\n");
40:   baum(laenge,anzahl);
41:   printf("\001 E @setflip 0 0\n\001 E @newpage 0 0\n\001 E @grapoff\n");
42:  } while (laenge != 0);
```

als neue Länge der Wert Länge*2/3 angegeben, dadurch werden die Äste zur Spitze hin kleiner und als Anzahl wird der Wert Anzahl –1 angegeben. Denn jeder Teilbaum besitzt weniger Äste als der ganze Baum. Danach wird die Drehung wieder rückgängig gemacht und der zweite Teilbaum im Winkel von –30 Grad angesetzt. Dazu wird Baum nochmals rekursiv aufgerufen. Dann wird die Drehung wieder rückgängig gemacht. Nun muß noch ein Schreite-Befehl um –Länge ausgeführt werden, so daß nach Aufruf von Baum wieder die Ausgangsposition der Zeichen-Schildkröte erreicht ist. Dies ist wichtig, damit auch jeder Teilbaum auf einfache Art an die Ast-Stellen angefügt werden kann. Bei jedem rekursiven Aufruf wird übrigens wieder neuer Speicherplatz für Länge und Anzahl reserviert, sonst würde die Rekursion nicht arbeiten, doch dies geschieht in C automatisch korrekt.

```
B)a:as68 -s a: turtle.s

B)a:c baum

B)A:CP68 -I A: BAUM.C BAUM.I

B)A:C068 BAUM.I BAUM.1 BAUM.2 BAUM.3 -F

B)ERA BAUM.I

B)A:C168 BAUM.1 BAUM.2 BAUM.S

B)ERA BAUM.1

B)ERA BAUM.2

B)A:AS68 -L -U -S A: BAUM.S

B)a:clink baum turtle

B)A:LO68 -R -U+NOFLOAT -O BAUM.68K A:S.O BAUM.O TURTLE.O .0 .0 .0 .0 .0 .0 .0 A:
CLIB

B)
```

Das Hauptprogramm liest die beiden Werte, Länge und Anzahl, von der Tastatur ein und ruft Baum auf. Ferner wird der Bildschirm gelöscht und das Zweiseiten-Umschalten unterdrückt. Achtung, die Eingabezeile wird hier mit GETS() eingelesen und nicht direkt mit SCANF(), da sonst bei einem Eingabefehler das Programm nicht mehr stoppt.

Bild 5 zeigt noch wie man die einzelnen Programme übersetzt (siehe auch *LOOP* 4).

(Fortsetzung folgt).

```
Anfang:  DC.L $55AAØ18Ø
         DC.B  'TEXTKURZ'
         DC.L  Start
         DC.L  End-Anfang
         DC.B  1
         DC.B  0,0,0
         DC.L  0,0

Start:   Moveq #12Ø,D1          ; Leerzeichen
         Moveq #ØD,D2           ; CR

         Move #!Getstx,D7       ; Textanfang nach DØ
         Trap #1
         Movea.l DØ,AØ          ; Textanfang nach AØ

         Moveq #1,D3

LoopØ:   Move.b (AØ)+,D4        ; Textende ?
         Beq.s Fertig

         Cmp.b D4,D1            ; Leerzeichen ?
         Bne.s LoopØ            ; neues Zeichen holen

         Lea -1(AØ),A1          ; Ja = Adresse merken

Loop1:   Cmp.b (AØ)+,D1         ; = Leerzeichen
         Beq.s Loop1

         Cmp.b -(AØ),D2         ; Nein , ist es CR ?
         Bne.s LoopØ            ; nur bei CR/LF Leerzeichen löschen

         Movea.l AØ,A2          ; Adresse für weitere Überprüfung

         Exg A1,A3              ; Adressen merken und weitere
         Exg A2,A4              ; Leerzeichen suchen
         Dbra D3,LoopØ

                                ; Jetzt :
                                ; A1 = Anfangsadresse 1. Bereich
                                ; A2 = Endadresse 1. Bereich
                                ; A3 = Anfangsadresse 2. Bereich
                                ; A4 = Endadresse 2. Bereich

Fertig:  Tst D3                 ; Wenn D3 = 1 , dann keine Leerz.
                                ;      D3 = Ø , nur 1 Bereich zu
                                ;               verschieben
         Bmi.s Loop2            ;      D3 = -1, 1 Bereich
         Bgt.s Ende             ;               und weiter suchen.

         Exg A1,A3
         Exg A2,A4
         Movea.l AØ,A3

Loop2:   Move.b (A2)+,(A1)+     ; Text nachruecken (Bereich)
         Beq.s Ende
         Cmpa.l A2,A3
         Bne.s Loop2

         Movea.l A1,A3
         Addq #1,D3
         Bra.s LoopØ            ; und weiter prüfen

Ende:    Move #!Putstx,D7       ; neues Textende setzen
         Trap #1
         Rts

End:
End
```

Der Leerzeichen-Killer

Das nachfolgende Programm verkürzt Textzeilen, d.h. Leerzeichen am Ende einer Zeile werden gelöscht. Diese Leerzeichen entstehen beim Arbeiten mit dem Editor und häufigem Gebrauch der Delete-Taste und der „Suche und Ersetze"-Routine.

Gerade der Anfänger wird erstaunt sein, wie sich seine alten Programme und Texte verkürzen und somit weiterer Speicherplatz für Ergänzungen und andere Aufgaben frei wird.

Das Programm kann nach dem Verlassen des Editors über die Bibliotheksfunktion aufgerufen werden. Textanfang und Textende werden erkannt und das Textende nach erfolgter Beseitigung der Leerzeichen korrigiert.

Die Bearbeitung eines Textes von 512 Zeilen, die alle mit Leerzeichen aufgefüllt wurden (80 Zeichen x 512 = 40 kB) dauert unter 1 Sec.!

Wahrscheinlich lassen sich noch schnellere und kürzere Routinen entwickeln, die meiner Meinung nach in das Grundprogramm integriert werden sollten. Wenn nach jedem Verlassen des Editors automatisch der Text gekürzt werden würde, wäre dem Anfänger mit wenig „RAM" sicher geholfen.

IN & OUT

Leser fragen – Fachleute antworten

Stellen Sie auch Ihre Fragen an „loop"

Sehr geehrter Herr Klein, sehr geehrter Herr Graf,

als Betreiber des NDR-Computers habe ich folgende Frage an die *LOOP*-Redaktion:

Wann und eventuell wo wird es Turbo-Pascal für den 68008-Ausbau geben? Gibt es das vielleicht schon irgendwo?

Ansonsten: Weiter so!!! Ich bin gern bereit, für mehr Umfang auch mehr zu bezahlen. Es ist ja sicherlich sehr schwierig bei solch einem komplexen System, alle Benutzer mit allen möglichen (und unmöglichen!) Ausbaustufen zufrieden zu stellen.

Horst Oelfke,
Ahornweg 30, 2858 Schiffdorf-Sellstedt

Antwort *LOOP:*

Herzlichen Dank für Ihre Anerkennung. Soweit wir in Erfahrung bringen konnten, wird Borland das Turbo-Pascal für den 68000 im Lauf des Jahres 1986 vorstellen. Wir hoffen, daß dieser Termin tatsächlich eingehalten wird und vielleicht der 68020 mit Arithmetik-Prozessor unterstützt wird. Dann wäre das ohnehin hervorragende Turbo-Pascal unschlagbar. Die Aussagen, die der deutsche Importeur gemacht hat, sind allerdings ohne Gewähr.

Seit Erscheinen der ersten *LOOP* beziehe ich diese Zeitschrift, da sie meiner Meinung nach die einzige Möglichkeit zur Information über Tips, Tricks und Neuerungen bietet.

Erfreut habe ich auch festgestellt, daß die Seitenanzahl erhöht wurde. Meiner Meinung nach war dieser Schritt auch längst fällig, da der Preis – auch unter Berücksichtigung der geringen Auflagenstärke – deutlich zu hoch war (ist?), wenn einmal dieser Preis mit dem Inhalt/Umfang ins Verhältnis gesetzt wird. Ein weiterer Kritikpunkt ist wohl das Erscheinen von Artikeln, die mit dem Inhalt dieser Zeitschrift nichts zu tun haben. So ist wohl der Berghüttenartikel eher dazu geeignet, den Herausgebern der *LOOP* (und deren Freunde) weniger das Interesse am NKC als das ausschließliche geschäftliche Interesse zu bescheinigen.

In der Hoffnung, daß diese Ausrutscher der Vergangenheit angehören und auch die inhaltliche Qualität noch weiter ansteigt (weniger Werbung und dafür mehr Tips, Tricks und kleinere Programmlistings), möchte ich diese Zeitschrift für ein weiteres Jahr abonnieren. Die damit fälligen 20 DM lege ich diesem Brief bei.

In der letzten *LOOP* (4) steht auf der Titelseite die CPU-Karte mit der 68000 CPU abgebildet, mit dem Stecker für Waitzyklen auf dem dritten Platz.

Mich interessiert, welche Möglichkeiten der Anwender hat, bzw. welche Änderungen vonnöten sind, um diese Karte völlig ohne Waitzyklen zu betreiben. Zwar könnte man z.B. ausschließlich statische Speicher benutzen und bei den Eproms CMOS Tpyen mit 150 ns Zugriffszeit einsetzen. Als Besitzer einer Detmolder Floppykarte gäbe es auch dort keine Probleme, da diese von sich aus Waitzyklen erzeugt. Aber wie sieht es mit den anderen Baugruppen wie der GDP aus? Evtl. wäre diese Problematik noch mal Thema für die *LOOP*, bevor der Verkauf dieser Karte im großen Stil anläuft und dann erst das große Ärgern kommt.

In der neuesten Ausgabe der *LOOP* (5) stand ein Leserbrief von jemandem, der Schwierigkeiten mit der Detmolder DIN TAST im Zusammenhang mit dem NKC hat.

Vielleicht könnten Sie meine Erfahrungen diesbezüglich in einem Artikel verwerten:

Die Probleme treten dadurch auf, weil beim Senden eines Zeichens von der Tastatur mit dem Strobe Signal ein Flip Flop gesetzt wird, das erst nach dem Abholen des Zeichens von der KEY durch das Grundprogramm wieder zurückgesetz wird. In der Zwischenzeit kann die Tastatur beliebig viele Zeichen senden, die dann verloren gehen. Erst nach dem Zurücksetzen des Flip Flops wird das nächste von der Tastatur kommende Zeichen auch wieder richtig eingelesen.

Die Detmolder DIN TAST ist nun in der Lage lange Zeichenketten auf Tastendruck loszuschicken. Um nun die Übergabe zu synchronisieren gibt es folgende Möglichkeit: Eine freie Ader des 12adrigen Tastaturkabels wird an Stift 1 des *seriellen* Ein-/Ausgangs gelötet. Dieser Eingang führt direkt zum Tastaturprozessor, der hier mit einem Low Pegel in seiner Arbeit unterbrochen werden kann. Dieser Eingang liegt noch über einem Widerstand nach + 5V, was durch Auftrennen der Brücke BR2 geändert werden muß. Die Brücke liegt deutlich sichtbar auf der Bestückungsseite rechts oberalb des Tastatureproms. Auf der anderen Seite des Kabels wird diese Ader an Stift 15 (letzter Stift) der KEY angeschlossen. Von der Lötseite aus gesehen liegt direkt gegenüber diesem Stift der Ausgang Q des 74LS74 (Pin 8). Diese beiden Punkte werden mit einem Widerstand verbunden. Ich habe einen 2k2 Ohm Widerstand eingelötet, aber der Wert ist unkritisch.

Die so entstehenden Möglichkeiten sind enorm. So läßt sich bei entsprechender Belegung einer Taste z.B. folgender Code auf einmal übertragen:

^KX4^M1^M1^M3^MJ der folgendes bewirkt: es wird aus dem Editor gegangen und in dem Menüpunkt Optionen Assembleroptionen und dort nur Fehlerausgabe gewählt, dann wird ins Bibliotheksmenü gesprungen und dort das erste Programm z.B. PASCAL gestartet. Die Geschwindigkeit, wie durch das Menü gehoppt wird, ist atemberaubend.

Übrigens liefert Detmold inzwischen eine neue Version des Tastatureproms bei den neuen Tastaturen aus, die generell eine sehr langsame Ausgabe der Tastatur bewirkt und dadurch zwar auch keine Zeichen verliert aber quälend langsam ist. Ich weiß leider nicht, ob auf Wunsch noch die alte Version erhältlich ist – nachfragen.

Ein weiteres Problem, das sicherlich schon bei vielen aufgetreten sein dürfte, ist das plötzliche, wiederholte Ausgeben des zuletzt eingegebenen Zeichens von der Tastatur, besonders bei Netzstörungen. Ursache ist das offen hängende Beinchen des XOR für den Tastaturstrobe. Dieses freie Beinchen fängt sich allerlei Störungen ein und interpretiert sie dann als Strobe Signal der Tastatur, worauf die alten Daten wieder gültig gemacht werden. Für Besitzer einer Tastatur mit pos. und neg. Strobe besteht die Lösung darin, daß die Brücke Js eingelötet wird und das entsprechende andere Strobe Signal der Tastatur angeschlossen wird. Besitzer einer Tastatur ohne diese Möglichkeit müßten vom freien Beinchen (Pin 10 des 74LS86) einen Widerstand nach + 5 V löten.

In der Hoffnung Ihnen vielleicht geholfen zu haben, verbleibe ich mit freundlichem Gruß.

Folkert Hallenga
Hufelandstr. 16/239, 3000 Hannover 91

Antwort *LOOP:*

Zunächst einmal herzlichen Dank für Ihre technischen Ausführungen, die sicher viele Anwender der DIN-Tastatur des Elektronikladens Detmold interessiert.

Das offene XOR-Beinchen ist mit der neuen Version der KEY behoben.

Nun zu Ihrer Kritik: schon in „eigener Sache" haben wir auf die Problematik der Kosten hingewiesen. Derzeit ist *„LOOP"* noch ein reines Zuschußunternehmen, wir hoffen, daß sich dies im nächsten Jahr ändern wird. Auch glauben wir, daß wir durchaus mal einen anderen Artikel als nur über NDR- und mc-Computer in der *„LOOP"* veröffentlichen dürfen.

Da wir im Allgäu in der glücklichen Lage sind, dort zu arbeiten, wo andere Urlaub machen, wollen wir die Möglichkeiten, die sich dadurch ergeben, auch gerne weitergeben. Da die angesprochene Berghütte nicht kostenlos vermietet wird, ist Ihre Kritik bezüglich des geschäftlichen Interesses berechtigt. Sofort nach Erscheinen der letzten *LOOP* hat sich ein *LOOP*-Leser aus dem hohen Norden ins Auto gesetzt, und ist 1000 km zu uns gefahren, um eine Woche auf der Hütte zu verbringen. Aufgrund des Artikels ist die Hütte bereits bis Januar total ausgebucht.

Nun, wir werden versuchen, die *LOOP* noch dicker und noch besser zu machen. Und hoffen natürlich auf Ihre Mitarbeit.

Sehr geehrter Herr Graf,

das in Ihrer Zeitung *LOOP* Nr. 5 abgedruckte Programmlisting „Sqash", auf Seite 11, läuft nicht.

Zwischen den Zeilen 806 und 811 fehlt die Anweisung IF LAENGE = 1.

Sicher würden sich andereAbonnenten der Zeitung *LOOP* darüber freuen, wenn Sie in der nächsten Ausgabe den Hinweis abdrucken würden.

Hans Berger,
Simonstraße 60, 851/ Fürth

Antwort *LOOP:*

Herzlichen Dank für diesen Hinweis. Die Zeile war bis zur vorletzten Kontrolle noch drin und ist beim sogenannten Umbruch verschwunden.

Wir haben daraus gelernt und werden in Zukunft Programme, die wir veröffentlichen, vor Drucklegung noch einmal abtippen und ausprobieren lassen.

Sehr geehrte Damen und Herren,

seit einigen Tagen habe ich die *LOOP* abonniert. Ich muß Ihnen sagen, daß ich die *LOOP* sehr gut finde und würde mich freuen, sie alle 2 Monate zu erhalten, mit noch mehr Tips und vielleicht mit Programmen. Vielen Dank für den Tip in *LOOP 3* mit der Resettaste der CPU Z80 Vollausbau.

Sehr geehrte Redaktion, ich habe auch gleich ein Problem, und zwar habe ich mir einen Drucker gebraucht gekauft, den Schneider NLQ 401. Er arbeitet beim Druck einwandfrei, aber ich sehe keine Möglichkeit für die Sonderfunktionen wie z.B. Zeichenverdichtung, Zeichenvergrößerung usw., vielleicht können Sie mir da helfen. Ferner würde ich es begrüßen, wenn bei einer neuen BASIC-Version die Befehle Cursor rauf, runter, rechts und links mit eingebaut würden.

Zuletzt habe ich noch einige Fragen. Wie ist es mit der CPU Z80-Vollausbau und der CPU68K, gibt es eine Möglichkeit, sie zusammen auf dem Bus zu betreiben oder auszutauschen, vielleicht durch einen Schalter? Gibt es ein Gehäuse für den Bus4? Wann gibt es die Farbgrafik? Was benötige ich zum Betreiben eines Akustikkopplers alles? Was benötige ich zum Betreiben eines Plotters?

Zuletzt noch eine Bitte, ich suche im Raum 8671 Hof/Saale NDR-Klein-Computer-Nachbarn für Informationsaustausch, vielleicht können Sie mir da helfen.

Für Ihr Bemühen auf all meine Fragen, verbleibe ich mit den besten Grüßen

Hans Federowsky
U.-H.-Reuth 41, 8671 Feilitzsch

Antwort *LOOP:*

Die Frage, ob der Drucker Sonderzeichen und Sonderfunktionen kann, müssen wir an die Leser weitergeben. Vielleicht kann jemand Herrn Federowsky helfen und ihm dies direkt mitteilen.

Bei einer Basicerweiterung, die sicher kommen wird, wird ein Bildschirmeditor mit eingebaut. Die gleichzeitige Betriebsmöglichkeit CPU Z80 und der CPU68K wird derzeit von uns getestet und gegebenenfalls in der *LOOP* veröffentlicht. Für den BUS4 setzen Sie bitte unser Gehäuse ein, das bereits zur Aufnahme des BUS4 vorbereitet ist. Die Farbgrafik – siehe Artikel in diesem Heft.

Zum Betreiben eines Akustikkopplers benötigen Sie eine serielle Schnittstelle SER und ein Programm zum Betrieb derselben, z.B. das ZEAT-Betriebssystem oder das von uns in *LOOP 3* veröffentlichte Modem-Programm für den Z80. Ein weiteres, universelles Terminalprogramm ist in Vorbereitung. Probieren Sie doch einmal unsere Datenbank aus. Telefon-Nummer: 0831/69330 und schreiben Sie eine Mitteilung rein, wenn es geklappt hat.

Einen Akustikkoppler bieten wir selbstverständlich auch an, er kostet derzeit DM 298,–.

Die serielle Schnittstelle können Sie zum Betrieb eines Plotters verwenden. Es gibt auch Plotter mit Centronix-Schnittstelle.

Weiterhin viel Erfolg mit dem NDR-Computer!

Sehr geehrte Redaktion!

Ich bin begeisterter Anhänger des Z80 und CP/M.

Leider vermisse ich beim Arbeiten mit einigen Programmen wie Word-Star und Turbo Pascal die Darstellung in inverser Schrift. Auch bei eigenen Programmen würde ich diese Darstellung gerne benutzen. Was kann ich tun?

Auf der Systemdiskette sind noch einige Fehler (z.B. Sysgen80, MOVCPM80, Ramfloppyunterstützung). Sind diese Fehler schon behoben und wie korrigiere ich sie?

Ist es vorgesehen CP/M 80 gegen CP/M Plus zutauschen?. Wer baut oder besitzt einen NDR-Computer im Raum SL – FL?

Für die Beantwortung meiner Fragen wäre ich sehr dankbar.

Holger Lindemann
Berliner Straße 78, 2380 Schleswig

Anwort *LOOP:*

Die inverse Schriftdarstellung bereitet beim GDP einige Schwierigkeiten. Wir sind jedoch dabei, das FLOMON zu ändern, so daß eine Darstellung, z.B. durch Unterstreichen, möglich wird. Ebenso wird die Scroll-Geschwindigkeit erhöht. Rolf-Dieter Klein ist zuversichtlich, diese Arbeiten bis Januar durchgeführt zu haben. Die erwähnten Fehler bitten wir uns etwas detaillierter zuzusenden, damit wir Ihnen hier weiterhelfen können.

Ein Einsatz von CP/M-Plus ist vorläufig nicht vorgesehen.

Aus der Technik

Fehler im PROMER-Handbuch

Seite 6, vierte Zeile unten:

Falsch: man mißt an PIN6

Richtig: man mißt an PIN1

Grund: PIN6 zeigt das invertierte Signal.

Achtung Fehler ROA64

Bei etwa fünfzig ausgelieferten Platinen und Bausätzen ROA64 (Lieferung im November 1985) ist folgender *Fehler* enthalten:

Verbindung zwischen zwei Leiterbahnen unter J2. Damit wird eine Verbindung zwischen allen Speichern Pin 13 und Pin 15 erzielt. Die *(falsche)* Verbindung ist auch *(leider)* im Handbuch, Seite 4, Bestücksplan ersichtlich.

Abhilfe: Leiterbahnverbindung durchkratzen, geht ohne Auslöten des Sockels, da halb unter dem Steg.

Wir bitten um Entschuldigung für diesen Fehler!

KONTAKTE

Suche in der Schweiz Kontakt mit NDR-Computer-Anwender.

Heinz Amgwerd
Rebbergstraße 13 a, CH-5610 Wohlen,
Telefon 057 222 76 42

Kontakte Raum 8990 Lindau.

Helmut Kulmus
Im Baumgarten 1, 8995 Weissenberg,
Telefon 08389/1075

Suche *dringend* Kontakte zu NDR-Nutzern im Raum Saarland/Pfalz.

Martin Mayländer
Saarwerderstraße 4, 6600 Saarbrücken 6
Telefon 0681/854201

Wer hat Arithmetik-Programme für den Z80?

Christof Hübner
Lessingstraße 2, 6729 Rheinzabern

FLOH MARKT

Verkaufe:

SBC2, Grundprogramm, POW5V, kleine Tastatur

Telefon: (08392) 1362

Verkaufe günstig:

NDR-Computer mit 32 kB fertig aufgebaut: CPU Z80, R0A64, GDP65K, CAS, KEY, EGRUND2, EBASIC2, TAST2, NE2, BUS3, TAXAN-Monitor, ITT-Recorder und diverse Literatur.

Klaus Finke
Prof.-Westermann-Str. 1A,
2807 Achim-Baden, Tel. 04202/70544

Verkaufe: NDR 68008-System Operator-Tastatur, NMC 200-Netzteil, 2 * Bus, 2 * R0A, CAS, IOE, usw., Centronic, Prommer, Pascal, sofort betriebsbereit, 1500,— DM incl. Gehäuse.

Telefon 030/267 42 26 – 16.00 Uhr

PASCAL-Progr. zum Generieren des Eproms der ELZET 80 DIN-TAST. Programmiere auch Eproms nach Wunschbelegung, DM 25,— bzw. DM 30,— + Eprom.

Achim Scheffel
Kohlseestraße 11, 6090 Rüsselsheim 5

LOOP - AKTUELL

Seminare in der Filiale Hamburg

7. 12. 1985: Vorstellung einer Terminal-Emulation mittels zweier CP/M-Computer. Einführung in Turbo-Pascal.

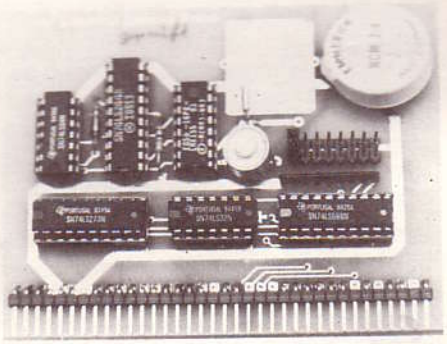
14. 12. 1985: C – was ist das? Einführung in eine neue Programmier-Sprache.

21. 12. 1985: Wie arbeitet man mit Wordstar? Anpassung einer Programmier-Sprache an ein vorhandenes System, z.B. Turbo-Pascal.

1986 finden weitere Seminare statt. Die Themen stehen zur Zeit noch nicht fest. Bitte rufen Sie uns unter folgender Nummer an: Tel.: 040/388151.

Die UHR-Baugruppe ist lieferbar

Endlich – die lang erwartete Echtzeituhr für den NDR-Computer ist nun ab Lager lieferbar. Sie funktioniert mit allen Konfigurationen und stellt Uhrzeit, Tag, Monat und Jahr zur Verfügung.



Die Uhr muß nur einmal eingestellt werden – über einen eingebauten Akku wird die Zeit und das Datum auch nach dem Ausschalten des NDR-Computers weitergeführt.

Die Abfrage der Daten ist sehr einfach und wird im Handbuch genau beschrieben. Das 68K-Grundprogramm unterstützt mit einer speziellen Routine bereits die UHR.

Die Preise:

UHRH	Nur Handbuch	10,00
UHRP	Leiterplatte ohne Handbuch	39,50
UHRB	Bausatz, komplett mit Handbuch	129,00
UHRF	Fertiggerät, komplett, geprüft	189,00

Aktuelle Software-Versionen

Hier der Stand vom Dezember 1985

Bezeichnung	Versionsnummer
EHEX2,EHEX	1.1a
EGRUND2,EGRUND	2.0
EGOSEI2,EGOSI	1.1
EBASIC2,EBASIC	1.5
EZASS2,EZASS	2.1
EFLOMON	1.5b
ESPS2,ESPS	1.2
EASS0–3,EG68000	4.3
EPASCAL,EPASCAL68	3.1
EGOSIC,EGOSI68	3.1
EUFORM68	1.0
EJOGIDOS	2.3
EJOGIMON	2.0
EDEMO	1.1
EZEAT	
CP/M2.2 (Z80)	2.2
Turbo-Pascal	3.0
Strukta	1.6
HEBAS	1.1
EDATED	4.0
EFIBU	2.0
EFRAGE	3.0
EBIO	1.0
EQPROM	1.0
EBOOT68	1.0

Serviceverbesserung – Telefondienst mittwochs bis 20.00 Uhr

Besonders für berufstätige Kunden haben wir zunächst probeweise für die Monate Dezember 85 und Januar 86 einen Abend-Telefondienst eingerichtet. Sie können technische Fragen beantwortet bekommen und natürlich auch bestellen. Beachten Sie bitte, daß nur eine Amtsleitung besetzt ist, also auch bei Freizeichen etwas länger läuten lassen oder nochmal versuchen.

Impressum:

LOOP Zeitung für Computerbauer

Herausgeber: Gerd Graf

Redaktion: Rolf-Dieter Klein, Gerd Graf

Gestaltung und Druck:
Karl-Heinz Rieder, Kempten

Herstellung und Anzeigenverwaltung:
GES GmbH
Magnusstraße 13, 8960 Kempten

Anzeigenpreisliste 1/84

LOOP IDEENWETTBEWERB

Anwendungen mit dem Einsteigerpaket

1. Preis: ein Warengutschein im Wert von DM 400,–

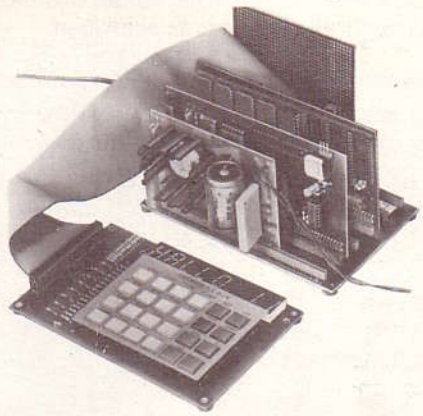
2. Preis: ein Warengutschein im Wert von DM 200,–

3. bis 5. Preis: Warengutscheine im Wert von DM 100,–.

Das Einsteigerpaket ist eine preisgünstige Möglichkeit, schnell und tief in die Mikroelektronik einzusteigen.

Viele Anwender wünschen noch mehr mit dem Paket E machen zu können! Auch wir denken darüber nach, das Einsteigerpaket um eine weitere, preisgünstige Ein-/Ausgabe-Einheit zu erweitern, um damit besonders in Richtung „Steuerung" noch mehr zu tun!

Mit kleinen Erweiterungen, z.B. einem Timer 556, der als einfacher A/D-Wandler geschaltet werden kann oder einigen Schalttransistoren lassen sich sicher eine enorme Anzahl von „Versuchen" mit dem Paket E aufbauen.



Wir denken daran, diese Versuche in einem dicken Handbuch zusammenzustellen und damit den immer mehr werdenden Anfängern zu helfen. Jeder Teilnehmer am Wettbewerb wird in diesem Handbuch namentlich erwähnt. Einige Denkanstöße: Thermometer, auch für Langzeitmessungen, einfaches Luxmeter (Beleuchtungsstärke) mit darauffolgenden Steuerungen, Feuchtefühler, z.B. für Blumenkästen, erweiterbar zur automatischen Gießanlage, Geschwindigkeitsmessungen mit zwei Fotozellen, universelle Digitaluhr, Stoppuhren mit verschiedenen Zwischenzeitmessungen, Ansteuerung für Modelleisenbahnanlagen (Weichenstraßen, Signalstellung).

Nach Abschluß des Wettbewerbs erstellen wir eine neue Universal-IOE-Baugruppe, die die gewünschten Hardware-Zusätze enthalten wird.

Teilnahmebedingungen:
Die Versuche sollen mit dem Paket E und *eventuell kleinen* Hardware-Änderungen möglich sein.

Sie müssen gut dokumentiert und besonders für den Anfänger geeignet sein. Der Vorteil eines Mikro-Computers soll hier vortreten. Bei komplexeren Programmen ist es möglich, Unterroutinen bereits im Betriebssystem vorzusehen.

Programme müssen wie folgt übergeben werden:

Prinzipielle Beschreibung

Strukturprogramm

Listing, wenn möglich mit ZEAT oder Macro-Assembler und auf Datenträger.

Literaturhinweise müssen angegeben werden.

Mit der Teilnahme am Wettbewerb gehen die Rechte an den eingesandten Schaltungen und Programmen an GES über. Der Rechtsweg ist ausgeschlossen.

Einsendeschluß: 31. Januar 1986.

Das Grundprogramm Z80 und 68008 wird erweitert

Aufruf zur Mitarbeit!

1986 planen wir eine Revision der Grundprogramme. Dies ist besonders beim 68000 wegen Einführung des 16- und 32-Bit-Rechners sowie durch Einführung der Hardcopy, schnellere Scroll-Geschwindigkeit usw. generell nötig.

Vorschläge, die Sie gerne im Grundprogramm hätten, senden Sie bitte umgehend an die *LOOP*-Redaktion.

Verbesserung des Basic-Interpreters Z80

Der Basic-Interpreter EBASIC soll endlich verbessert werden. Eingebaut wird ein Bildschirmeditor, Ansteuerung der COL256 und vermutlich On-Interrupt-GOSUB.

Weitere Vorschläge, besonders gewünschte Kompatibilität (Microsoft-Basic) bitte an die *LOOP*-Redaktion.

REL – Die Relais-Baugruppe für den NDR-Computer ist ab Lager lieferbar

Die REL (Relais-Interface) ist eine Baugruppe, die eine Beschaltung von 8 externen Geräten (z.B. Modelleisenbahn, Heizungsregler etc.) vom Computer aus ermöglicht. Die beschalteten Geräte sind galvanisch von der Computerspannung getrennt; d.h., es kann auch Wechselstrom geschaltet werden.

Es sind zusätzlich noch 8 verschiedene Bits abfragbar.

Die Baugruppe REL kann an jeder beliebigen Stelle auf dem Bus eingesetzt und von jeder Prozessorkarte (SBC2, CPU Z80, CPU68K und CPU 68000) aus angesprochen werden.

Um eine Baugruppe ansprechen zu können, ist die Einstellung einer Portadresse notwendig. Bei der REL-Karte ist hierfür der 8fache DIP-Schalter vorgesehen. Damit lassen sich 256 verschiedene Portadressen für die REL-Karte einstellen. Dadurch ist die Verwendung von mehreren REL-Karten gleichzeitig möglich.
Wie bereits erwähnt, ermöglicht die REL-Karte das Ansprechen von 8 verschiedenen Relais. Diese acht Relais können von einander unabhängig geschaltet werden.

Nach RESET oder nach dem Einschalten des Computers sind alle Relais abgefallen (ausgeschaltet).

Da oft die POW 5V zur Stromversorgung des Computers verwendet wird und die REL-Karte relativ viel Strom zieht (bis zu 0,68 Ampere), ist die Möglichkeit eingeplant worden, eine externe Stromversorgung für die Relais anzuschließen.

Technische Daten

Spannung: + 5 V

Stromaufnahme: je nach Anzahl der angezogenen Relais bis zu 680 mA

Busformat: jeder NDR-Bus

Größe der Leiterplatte: 100 mm x 81 mm

Anzagl der Relais: 8

Schaltleistung der Relais: 60 Watt

Schaltstrom der Relais:
2 Ampere (DC), 1 Ampere (AC)

Schaltspannung der Relais:
30 Volt (DC), 125 Volt (AC)

Preise:

RELH	Handbuch REL	DM 10,00
RELP	Leiterplatte REL	DM 39,50
RELB	Komplettbausatz incl. Handbuch	DM 218,00
RELF	Fertiggerät, geprüft	DM 258,00

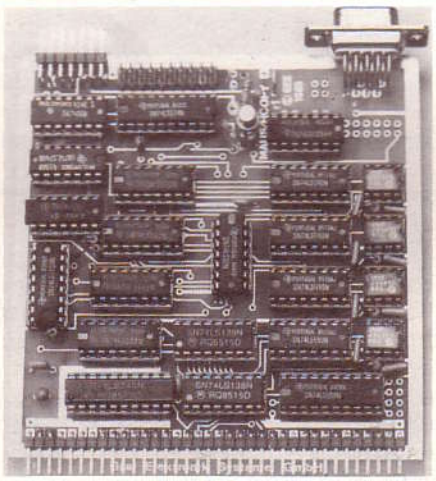
Hardcopy-Maus – ab Lager lieferbar

Was nützen die schönsten Grafiken, die interessantesten Bildschirmausgaben, wenn man sie nicht „schwarz auf weiß nach Hause tragen kann?"

Das direkte Auslesen des Bildwiederholspeichers auf der GDP bereitet aufgrund von Timingproblemen des Controller-Bausteines GDP9366 Schwierigkeiten. Das Problem ist nur durch einen wesentlich erhöhten Bauteileaufwand auf der GDP zu lösen.

In der Zukunft wird es sicher auch wichtig sein, Hardcopys nicht nur von der GDP, sondern von anderen Video-Signalen (z.B. TV-Kamera) zu erstellen. Deshalb haben wir uns entschlossen, eine eigene Baugruppe für die Hardcopy vorzustellen, die Baugruppe Hardcopy-Maus-Fadenkreuz.

Die ausschließliche Verwendung von TTL-Bausteinen ermöglicht eine Benutzung der Baugruppe mit allen bisherigen CPU-Platinen (SBC2, CPUZ80 und CPU68K). Die Durchführung der grafischen Funktionen (Hardcopy und Fadenkreuz) erfolgt im Zusammenspiel mit der GDP64K-Platine und einer Ansteuerungsbaugruppe für einen Drucker (z.B. SER oder IOE+CENT). Abgesehen von den genannten Baugruppen genügt ein



Minimalsystem zum Betrieb der Karte.

Durch die Verwendung moderner und platzsparender Bausteine gelang es, auf der Platine drei wichtige Funktionen unterzubringen:

- Erstellung einer Hardcopy,
- Ansteuerung einer Maus,
- Ausgabe eines flimmerfreien Fadenkreuzes und
- Anschluß eines A/D-Wandlers zur Digitalisierung von Bildern.

Unter einer Hardcopy versteht man die Ausgabe des aktuellen Bildschirminhalts auf einen Drucker. Bisher gab es keine Möglichkeit, die durch die GDP-Baugruppe erzeugten Texte oder Grafiken auf einen Drucker auszugeben. Die Freude über gelungene Grafiken oder ähnliches blieb auf die kurze Betrachtung am Bildschirm beschränkt. Zusammen mit einem eigenen Programm und einem grafikfähigen Drucker (z.B. EPSON) erlaubt die HCOPY/MAUS-Platine nun die Fixierung eines Bildes auf Papier.

Als Maus bezeichnet man ein kleines Kästchen, daß bei der Bewegung auf einer flachen Unterlage dem Computer Informationen über die Bewegungsrichtung und die zurückgelegte Entfernung liefert. Die Umsetzung der Bewegung kann rein mechanisch mit einer Rollkugel oder auf optischem Wege erfolgen. Optische Mäuse arbeiten zwar genauer und verschleißärmer, doch bildet der wesentlich höhere Preis einen unangenehmen Nachteil. „Intelligente" Mäuse liefern dem Computer die Bewegungsinformation fertig aufbereitet über eine serielle Schnittstelle. Dieser Komfort besitzt allerdings auch seinen Preis. Die HCOPY/MAUS-Platine ermöglicht den Anschluß einer preisgünstigeren Maus oder wahlweise eines noch günstigeren Trackballs. Eine einfache Maus oder ein Trackball besitzt 4 TTL-Ausgänge entsprechend den vier Bewegungsrichtungen. Anhand der Signale dieser Ausgänge ermittelt die HCOPY/MAUS-Baugruppe, gesteuert durch das entsprechende Programm, die durchgeführte Bewegung.

Zur Erledigung grafischer Arbeiten benötigt man oft ein Fadenkreuz, um beispielsweise eine Positionierung auf eine bestimmte Stelle vornehmen zu können. Das Fadenkreuz der HCOPY/MAUS-Baugruppe arbeitet, im Gegensatz zum Fadenkreuz der GDP64K-Baugruppe, flimmerfrei und führt daher zu einer geringeren Ermüdung des Benutzers.

Über einen zusätzlichen Port besteht die Möglichkeit einen A/D-Wandler zur Digitalisierung von Video-Signalen anzuschließen.

Technische Daten:
Betriebsspannung: + 5 Volt
Stromaufnahme: ca. 550 mA
Bus Format: NDR-Klein-Bus 36polig
Größe der Leiterplatte: 100 x 105 x 1,5 mm
Anschluß der Apple-Maus: 9pol. Cannon-Stecker (auf Seiten der Maus)
Anschluß an GDP64K-Platine:
2 x 7pol. Stiftleiste
Anschluß an A/D-Wandler:
2 x 10pol. Stiftleiste

Preise:

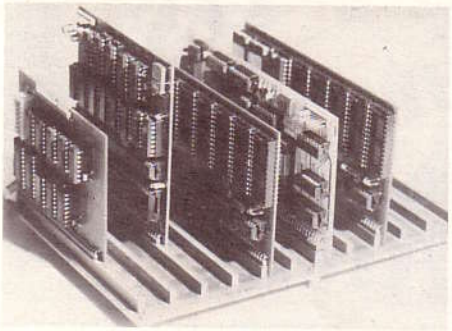
Zum NDR-Computer:

MAUSNDRH	Handbuch (HB)	10,00
MAUSNDRP	Leiterplatte ohne HB	39,50
MAUSNDRB	Komplettbausatz mit HB	198,00
MAUSNDRF	Fertiggerät, geprüft, mit HB	258,00

Zum mc-CP/M-Computer:

MAUSMCH	Handbuch (HB)	10,00
MAUSMCP	Leiterplatte ohne HB	39,50
MAUSMCB	Komplettbausatz mit HB	240,00
MAUSMCF	Fertiggerät, geprüft, mit HB	298,00

Echte 16 bit für den NDR-Computer CPU 68000 nun ab Lager lieferbar!



Die bereits in *LOOP 4* vorgestellte CPU 68000 ist nun, komplett mit Dokumentation, ab Lager lieferbar.

1. Wozu dient die CPU 68000?

Die CPU 68000 ist eine zentrale Baugruppe für den NDR-Computer. Sie enthält den Mikroprozessor, das wichtigste Steuerelement in einem Computer. Eine CPU-Baugruppe muß also in jedem Computer vorhanden sein. Für den NDR-Computer gibt es verschiedene CPU-Baugruppen; die CPU 68000 ist eine der leistungsfähigsten. Durch den 16-bit Datenbus des Mikroprozessors 68000 ist der Aufbau etwas komplexer als beim Mikroprozessor 68008. Allerdings ergibt sich eine Geschwindigkeitssteigerung auf etwa das doppelte, da nur noch ein Buszugriff für zwei Datenbytes durchgeführt werden muß. Dadurch eignet sich die CPU 68000 für Aufgaben, bei denen die Leistung der CPU68K nicht ausreicht.

Die CPU 68000 dient ebenso wie die CPU68K auch zum Betrieb mit dem Betriebssystem CP/M68K, ist durch ihren höheren Durchsatz jedoch für viele Programme besser geeignet.

Die Busplatinen BUS3 und BUS4 sind für den Einbau der CPU 68000 vorbereitet. Dazu muß an der dafür vorgesehenen Stelle eine doppelte Busleiste eingesetzt werden. Alternativ können auch zwei schwarze, einreihige Busleisten nebeneinander eingesetzt werden. Außerdem müssen die Leitungen D0 – D7, –RD und –WR zwischen den Reihen der Doppelbuchsenleiste aufgetrennt werden (siehe Prinzipbeschreibung).

2. Technische Daten

Spannung: + 5 V

Stromaufnahme: ca. 400 mA

Busformat: NDR-Bus 108-polig (Spezial-Ausführung für 16 bit)

Größe der Leiterplatte: 145 mm x 78 mm

CPU: 68000
- 32-bit Daten- und Adressregister
- 16 Megabyte linearer Adressbereich
- 56 leistungsfähige Befehlstypen
- Operationen auf 5 Haupt-Datentypen
- 14 Adressierungsarten

Taktfrequenz: 12 MHz (!)

Datenleitungen: 16

Adressleitungen: 21, aufgeteilt in 2 x 20 für die beiden Bushälften (s. Kapitel 3)

Ansprechbarer Speicher: 2 MByte

Ansprechbarer I/0-Bereich: 512 Byte in einem 128 KByte-Bereich, jedoch nur 256 Byte verwendet

3. Der Mikroprozessor 68000

Der Mikroprozessor 68000 ist ein moderner Prozessor, der nicht nur übersichtliche, sondern auch mächtige Befehle, wie eine eingebaute Multiplikation und Division besitzt. Man benötigt bei ihm nur wenige Befehle, um komplizierte Vorgänge zu beschreiben.

Der Mikroprozessor 68008 ist eine „abgemagerte" Version des 68000. Der 68008 hat 20 Adressleitungen und 8 Datenleitungen, der 68000 sogar 24 Adressleitungen und 16 Datenleitungen. Damit kann der 68000 bis zu 16 MByte Speicher ansprechen. Darin lassen sich sehr umfangreiche Programme und Datenmengen ablegen.

Beide Prozessoren stimmen in der Programmierlogik überein. Man braucht also keinen neuen Befehlssatz zu erlernen, wenn man vom 68008 auf den 68000 aufsteigt. Allerdings gibt es bei Programmen im EPROM Probleme, da die Speicheradressierung etwas unterschiedlich ist.

Bei Programmen im RAM gibt es dieses Problem nicht. Sie müssen sowieso erst geladen werden und dabei legt der 68000 sie automatisch in geeigneter Form ab.

Der 68000 benötigt wegen der Datenbusbreite von 16 bit genau passenden Speicher, also einen Speicher, der ebenfalls 16 bit parallel liefern kann. Da der Prozessor dann in einem Atemzug gleich zwei Bytes vom Speicher holen kann, ist er noch ein Stück schneller als der 68008. Durch diese Anforderung mußten wir uns etwas überlegen, damit es möglich wurde, auch mit dem 68000 alle Speicher- und Peripheriekarten des NDR-Computers zu benutzen. Dies wird im Handbuch zur CPU 68000 beschrieben.

Preise:

CPU 68000H Handbuch, komplett	DM 20,00
CPU 68000D Datenbuch der CPU 68000	DM 10,00
CPU 68000P Leiterplatte ohne Handbuch (mitbestellen)	DM 49,50
CPU 68000B Bausatz, komplett	DM 498,00
CPU 68000F Fertiggerät, geprüft	DM 595,00
EG 68000 Grundprogramm 68000, benötigt 2 x ROA64	DM 185,00
EB 68000 BOOT-Eproms für CP/M68K	DM 50,00
EGOSI68 GOSI-Compiler für 68000	DM 185,00
EPASCAL68 PASCAL-Compiler	DM 155,00

Achtung: „Aufsteiger" von CPU68K: Bei Zusendung Ihrer Rechnungskopie erhalten Sie für von uns bezogene EPROM-Software (z.B. EASSØ–3, EPASCAL) beim Kauf der CPU 68000 und der entsprechenden Programme 50% Gutschrift auf den von Ihnen bezahlten Betrag für diese Eproms. Die Eproms bleiben bei Ihnen, wir benötigen nur die Rechnungskopie oder die Angabe der Rechnungs-Nummer.

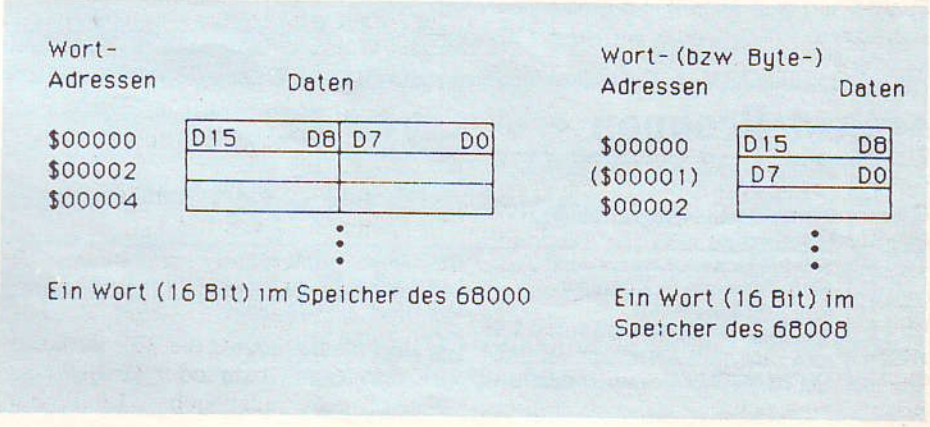
Beispiel: Sie bestellen

CPU 68000B	DM 498,00
EG 68000	DM 185,00
	DM 683,00
Gutschrift für EASSØ-3, 50 % aus DM 185,00	– DM 92,50
	DM 590,50

Der CPU68000-Bausatz ist leider etwas teuer, da wir die *12 MHZ-CPU* verwenden. Auf vielfachen Wunsch werden wir ab Januar eine 8 MHZ-Version dieses Bausatzes vorstellen.

Preise der 8 MHZ-Version:

CPU68000-8P	Leiterplatte	49,50
CPU68000-8B	Bausatz mit HB	298,00
CPU68000-8F	Fertiggerät, geprüft, mit HB	378,00



Theorie und Praxis rund um den NDR-Computer

Mikroelektronik
Einführung

NEU

4 Kursteile (je ca. 70 Seiten im Format A4), DM 38,- je Kursteil

Der Kurs ist auf die HEXIO abgestimmt und ist für alle geeignet, die ihre ersten Schritte in Z 80-Maschinenprogrammierung machen.

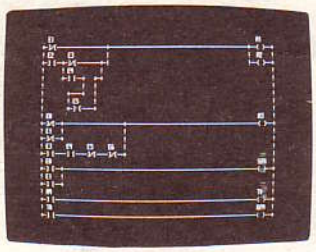
Nach diesem Kurs sind Sie in der Lage, eigene Programme zu schreiben und die Arbeitsweise des Z 80 zu verstehen.

Der Kurs ist in verschiedene Fachgebiete aufgeteilt und bringt eine Menge Aufgaben, Beispielprogramme und Übungen.

Aus dem Inhalt: Was ist ein Mikroprozessor? * Inbetriebnahme des Computers * Planung von Programmen * Aufbau der CPU * Speicher und Adressen * Datentransfer * Lauflicht * Breakpoints * Hilfsfunktionen * Logo-Elemente * Strukturiertes Programmieren * Label & Call.

SPS-Programmierung

NEU



4 Kursteile (je ca. 70 Seiten im Format A4), DM 38,- je Kursteil

Dieser Kurs zeigt Ihnen, wie SPS programmiert wird, die Normung, die Anwendungsmöglichkeiten und die verschiedenen Darstellungsarten.

Sie lernen spielend leicht, Relais- und Schützensteuerungen in SPS-Programme umzusetzen.

Beispielprogramme, Aufgaben und Übungen geben Ihnen die praktischen Erfahrungen und zeigen, wie SPS professionell eingesetzt wird. Nutzen Sie Ihren NDR-Computer für diese moderne Technik voll aus.

Der Kurs ist in folgende Fachgebiete gegliedert: Steuerungstechnik * Digitaltechnik * Methoden zur Beschreibung von Steuerungsaufgaben * Programmierung * Übungen und Tafeln.

ZEAT-Betriebssystem

NEU

Das Betriebssystem beeinhaltet in drei EPROMs: Z 80-2-Pass-Assembler, Disassembler, Editor, Debugger, Telefonmodem-Programm, FLOMON 1,5, ausserdem eine ausführliche Dokumentation zum Preis von DM 198,– .

Das Betriebssystem ZEAT benötigt 64-K-RAM (dynamische RAM-Karte). Die EPROMs werden in die BANKBOOT-Karte eingesteckt und sind sofort betriebsbereit. Programmieren Sie Ihren NDR-Computer mit einem Profi-Assembler.

Das Textverarbeitungsprogramm hat volle Bildschirmeditierung und kann neben der Programmeditierung auch zum Textschreiben eingesetzt werden.

Z 80-Assembler-Programmierung

4 Kursteile (je ca. 70 Seiten im Format A4), DM 38,– je Kursteil

Der Kurs ist auf das ZEAT-Betriebssystem abgestimmt und zeigt Ihnen in leicht verständlicher Art, wie der NDR-Computer in Z 80-Assembler programmiert wird, bringt reichhaltig Übungsbeispiele und Anwendungen. Sie werden erstaunt sein, wie leicht diese Art der Programmerstellung ist. Und Sie lernen, wie man die serielle Schnittstelle bedient und Daten über Telefon übertragen kann.

Die Fachgebiete dieses Lehrgangs sind: Systembeschreibung * Betriebssystem * Programmierung * Testen * Modemprogramm * Listings, Tafeln und Tabellen.

Christiani

Hier abtrennen und im Umschlag einsenden an: Dr.-Ing. P. Christiani GmbH, Techn. Lehrinstitut und Verlag, Postfach 35 69189, 7750 Konstanz

Bestellcoupon

	Preis je Teil	Gesamtpreis
☐ Einführung mit dem NDR-Computer (4 Kursteile)	DM 38,–	DM 152,–
☐ Z 80-Assembler-Programmierung (4 Kursteile)	DM 38,–	DM 152,–
☐ SPS-Programmierung mit dem NDR-Computer (4 Kursteile)	DM 38,–	DM 152,–
☐ Kompakt-Kurs BASIC (angepasst an das RDK-BASIC)	DM 198,–	DM 198,–
☐ ZEAT-Betriebssystem (3 EPROMs mit Dokumentation)	DM 198,–	DM 198,–

Name, Vorname

Straße

PLZ, Ort

Datum Unterschrift 86189