LOOP 16

3. JAHRGANG

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

Der SPS-Compiler ist da!

Aus speicherprogrammierbarer Steuerung direkt Z80-Assembler-Quelltext erzeugen! Das neue, sensationelle Produkt von Rolf-Dieter Klein!

1. Was ist SPS?

Die Abkürzung SPS steht für speicherprogrammierbare Steuerung.

„Normale" Mikrocomputer können in industriellen Prozessen nur sehr schwer für Steuerungsaufgaben eingesetzt werden. Für technische Anwender ist es nicht zumutbar, sich mit Carryflags, logischen Verknüpfungen auf Assemblerebene, Ein-/Ausgabebefehlen etc. herumzuplagen. Deswegen wurde die Programmiersprache SPS entwickelt, die es ermöglicht, mit einfachen, logischen Verknüpfungen Steuerungen zu realisieren.

Durch diese Sprache ergeben sich sehr viele Vorteile:

- einfache Programmierbarkeit und Programmänderungen
- hohe Verknüpfungsdichte
- schnelle, automatische Programmdokumentation
- geringer Platzbedarf
- hohe Systemzuverlässigkeit
- einfache Programmvervielfältigung
- Aufteilung einer Steuerung in Einzel-, Gruppen- und Leitsteuerung.

Da SPS nur Kenntnisse in der Schaltalgebra voraussetzt, ist sie wegen ihrer einfachen Programmierung für jeden gedacht, der eine logische Schaltung entwickeln will. Dies gilt auch für Schulen, Labors im Bereich der Steuerungstechnik und der Schaltalgebra.

Eine freiprogrammierbare SPS arbeitet bei der Programmeingabe mit Funktionsgleichungen, wie sie aus der Schaltalgebra (Bool'sche Algebra) bekannt sind. Beispiel: Funktionsgleichung der Schaltalgebra für eine Undverknüpfung

$A0 = E0 \wedge E1$

im Beispiel für SPS

E0 & E1 = A0

SPS kennt die logischen Verknüpfungen und, oder, nicht, Zuweisung, ein Setzen und ein Rücksetzen (z. B. eines Flip-Flops). Als Operanden sind Eingang und Ausgang sowie Merker und Timer vorhanden. Innerhalb einer SPS-Programmzeile können durch Klammerausdrücke fast beliebig tief gestaffelte Formeln stehen. Beispiel:

!NM3 & (E3/E4) & (E2/E8) = SA1 = SA4 = SM2 = SM3

Dieses Beispiel wurde aus einer einfachen Fahrstuhlsteuerung aus dem SPS-Handbuch entnommen.

2. Was ist ein SPS-Compiler?

Dem großen Vorteil der speicherprogrammierbaren Steuerung – die einfache Programmierbarkeit – steht ein gewichtiger Nachteil gegenüber. SPS's sind bisher nur als Interpreter bekannt, d. h., für die Steuerung muß ein SPS-Computer zur Verfügung stehen. Dies bedeutet einen erheblichen Kostenaufwand, so daß SPS-Steuerungen nicht in einfache Systeme Einzug gefunden haben.

Mit dem SPS-Compiler gibt es nun folgende Möglichkeiten:

- Eingabe der SPS-Programme am Bildschirm
- Start des SPS-Compilers
- Der SPS-Compiler erzeugt eine Assembler-Quelldatei im Z80/8080-Code
- Übersetzen dieser Quelldatei z. B. mit M80

Voranzeige

Graf Elektronik wieder auf der Hannover Messe CeBit!

HANNOVER MESSE CeBIT'88
Welt-Centrum Büro · Information · Telekommunikation
16. – 23. MÄRZ 1988
HALLE 6 · F 21

Wie letztes Jahr, finden Sie uns – und viele unserer Software-Partner – wieder auf der Hannover Messe. Am gleichen Stand: Halle 6 Stand F21. Diesmal lohnt sich der Besuch besonders – mehr davon in der nächsten LOOP.

Aus dem Inhalt:

Der SPS-Compiler ist da 16/1
In eigener Sache 16/5
Aktion Steckbrief 16/6
MLA v1.4 für JADOS ist da 16/6
Liefertermine Microsoft 16/8
Messen von analogen Größen 16/10
Datum 16/12
Centronic-Schnittstelle 16/14
Z80-Vollausbau 16/16
Tips und Tricks in dBASEII . . . 16/20
Pascalprogramme auf Eproms 16/20
Tips und Tricks bei HEBAS . . 16/21
Ergänzung zum Skop-Programm AD 10 x 1 . . . 16/22
RUBAFIND-Programm zum Suchen von Bytefolgen 16/23
JUMP per Interrupt 16/26
Objectfile-Format für JADOS . 16/27
Aus der Technik 16/28
Erweiterung der Eprom-Programmierkarte 16/29
Leserbriefe und Kleinanzeigen 16/32

- Linken und Erzeugen eines Eproms, das in einem Ein-Platinen-Computer (z. B. SBC2) lauffähig ist.

Damit lassen sich nun Steuerungen auch für einfache Anwendungen am Bildschirm entwickeln und anschließend in ein Eprom brennen.

Bei Fehlern oder Erweiterungen ist es jederzeit möglich, das SPS-Programm zu modifizieren (es wird mit einem normalen Texteditor geschrieben und als normale, sequentielle Datei gespeichert) und den Übersetzungslauf neu durchzuführen.

3. Kenndaten des SPS-Compilers.

Mit Hilfe des SPS-Compilers ist es möglich (abhängig vom Speicherplatz im Zielsystem), beliebig lange SPS-Programme zu erstellen. Die Beschränkung auf 24 Zeilen in unserem SPS-Interpreter ist damit hinfällig.

Weiter können eine beliebige Anzahl von Ein- und Ausgängen sowie Merker und Timer verwaltet werden. Die Einschränkung ist wiederum nur der Speicherplatz und die I/0-Baugruppen im Zielsystem.

Weiter können die Ein- und Ausgabeports innerhalb des SPS-Programmes beliebig definiert werden, so daß ebenso beliebig Ein-Platinen- oder größere Rechner mit unterschiedlichen Ein-/Ausgabekarten verwendet werden können.

Bild 1 zeigt die dazu notwendigen, neuen Befehle innerhalb des SPS.

eingang[indexbereich1] = portadresse[indexbereich2]

ausgang[indexbereich1] = portadresse[indexbereich2]

timer[index] = einschaltverzögerung,ausschaltverzögerung

frequenz wertangabe

negein

negaus

Achtung Befehle in Kleinschrift eingeben!

Bild 1

indexbereich steht für entweder eine einzelne Zahl, oder ein wertebereich a..b. Die Portadresse wird in Hex mit einem führenden 0x angegeben. Dezimal auch möglich. Beispiel:

```
eingang[0..3] = 0x30[2..5]
ausgang[122] = 0x31[3]
timer[9] = 120,140
frequenz 4000000
```

Die Zeitangaben beim Timer erfolgen in Millisekunden, die CPU-Frequenz wird in Hz angegeben.
Die Frequenzangabe ist wichtig, damit der Compiler die Timer richtig berechnen kann. 4MHz sind aber voreingestellt. Der Schwankungsbereich wird als Kommentar am Ende der Assemblerquelle mit ausgegeben. Die Zykluszeit wird am Ende der Übersetzung mit ausgegeben.

negein stellt negative Logik für alle Eingänge ein, und negaus stellt negative Logik für alle Ausgänge ein.
Dies ist z.B. nötig, wenn man mit Schaltern nach Masse, und LEDs nach +5V arbeitet.

4. Praktischer Einsatz.

Das SPS-Quellprogramm wird – mit den in Bild 1 vorgestellten Befehlen am Anfang – mit einem normalen Texteditor, z. B. mit WordStar, geschrieben und mit der Erweiterung .SPS abgelegt. Danach startet man die Übersetzung mit

SPS name

wobei name für den gewählten Dateinamen steht.

Bild 2 zeigt ein kleines SPS-Programm in Form dieser Quelldatei. Die Anweisung „zilog" am Anfang teilt dem SPS-Compiler mit, daß er Z80-Befehle erzeugen kann; ansonsten werden nur 8080-Befehle erzeugt.

Anschließend wird die Übersetzung gestartet mit

SPS name

```
zilog
negein
negaus
eingang[0..7] = 0x30[0..7]
ausgang[0..7] = 0x30[0..7]
timer[0] = 1000,1000

!E0 & E1 = A0
!E0 = SA1
!E1 = RA1
!E0 & A3 = NA2
!E1 & A2 = NA3
!E0 = T0
!T0 = A4
!PE
```

Bild 2: SPS-Programm

Der SPS-Compiler arbeitet und erzeugt eine Datei namens

name.MAC – die Assembler-Quelldatei.

In unserem Beispiel ist sie in Bild 3 gezeigt. Diese Assembler-Quelldatei kann nun – bei Bedarf – noch erweitert bzw. optimiert werden. Anschließend wird sie mit dem Makroassembler M80 in eine REL-Datei verwandelt. Der Aufruf dazu lautet:

M80 name = name.

Nun kann man mit dem Linker entweder eine COM-Datei, die unter CP/M ablauffähig ist oder eine HEX-Datei zum Brennen in Eproms für einen Ein-Platinen-Computer, z. B. die SBC2-Baugruppe, erzeugen. Die Befehle für die COM-Datei lauten:

L80 name,name/N/E:

```
; Programm test3 durch SPS-Compiler V 1.1 erzeugt.
; von Rolf-Dieter Klein (C) 1987, Muenchen
.z80
 CSEG
start:   ; Programm Start
 LD SP,stack        ; Stack am Ende vom RAM
 JP INIT     ; zuerst alles Initialisieren und definieren
LOOP:        ; Schleifenanfang
 LD A,(s0)   ;Eingang
 LD B,A
 LD A,(s1)   ;Eingang
 AND B
 LD (s8+1),A        ;Ausgang
 LD A,(s0)   ;Eingang
 OR A
 JR Z,p1
 LD (s10+1),A       ;Ausgang
p1:
 LD A,(s1)   ;Eingang
 OR A
 JR Z,p2
 PUSH AF
 XOR A
 LD (s10+1),A       ;Ausgang
 POP AF
p2:
 LD A,(s0)   ;Eingang
 LD B,A
 LD A,(s12)  ;Ausgang
 AND B
 CPL
 LD (s11+1),A       ;Ausgang
 LD A,(s1)   ;Eingang
 LD B,A
 LD A,(s11)  ;Ausgang
 AND B
 CPL
 LD (s12+1),A       ;Ausgang
 LD A,(s0)   ;Eingang
 LD (s17+1),A       ;Timer
 LD A,(s17)  ;Timer
 LD (s13+1),A       ;Ausgang
DATEIN:      ; Einsprung bei Start
 IN A,(030h)        ; Eingang 0
 CPL  ; da negative Logik
 BIT 0,A
 JR Z,p3
 LD A,0FFh
```

Bild 3: Die vom Compiler erzeute Quelldatei

```
 JR p4
p3:
 XOR A
p4:
 LD (s0),A
 LD A,(s8+1)        ; Ausgang 0
 LD (s8),A
 OR A
 LD A,(s9)
 JR Z,p5
 SET 0,A
 JR p6
p5:
 RES 0,A
p6:
 LD (s9),A
 LD IX,s17
 CALL timexec
 IN A,(030h)        ; Eingang 1
 CPL  ; da negative Logik
 BIT 1,A
 JR Z,p7
 LD A,0FFh
 JR p8
p7:
 XOR A
p8:
 LD (s1),A
 LD A,(s10+1)       ; Ausgang 1
 LD (s10),A
 OR A
 LD A,(s9)
 JR Z,p9
 SET 1,A
 JR p10
p9:
 RES 1,A
p10:
 LD (s9),A
 IN A,(030h)        ; Eingang 2
 CPL  ; da negative Logik
 BIT 2,A
 JR Z,p11
 LD A,0FFh
 JR p12
p11:
 XOR A
p12:
 LD (s2),A
 LD A,(s11+1)       ; Ausgang 2
 LD (s11),A
 OR A
 LD A,(s9)
 JR Z,p13
 SET 2,A
 JR p14
p13:
 RES 2,A
p14:
 LD (s9),A
 IN A,(030h)        ; Eingang 3
 CPL  ; da negative Logik
 BIT 3,A
 JR Z,p15
 LD A,0FFh
 JR p16

p15:
 XOR A
p16:
 LD (s3),A
 LD A,(s12+1)       ; Ausgang 3
 LD (s12),A
 OR A
 LD A,(s9)
 JR Z,p17
 SET 3,A
 JR p18
p17:
 RES 3,A
p18:
 LD (s9),A
 IN A,(030h)        ; Eingang 4
 CPL  ; da negative Logik
 BIT 4,A
 JR Z,p19
 LD A,0FFh
 JR p20
p19:
 XOR A
p20:
 LD (s4),A
 LD A,(s13+1)       ; Ausgang 4
 LD (s13),A
 OR A
 LD A,(s9)
 JR Z,p21
 SET 4,A
 JR p22
p21:
 RES 4,A
p22:
 LD (s9),A
 IN A,(030h)        ; Eingang 5
 CPL  ; da negative Logik
 BIT 5,A
 JR Z,p23
 LD A,0FFh
 JR p24
p23:
 XOR A
p24:
 LD (s5),A
 LD A,(s14+1)       ; Ausgang 5
 LD (s14),A
 OR A
 LD A,(s9)
 JR Z,p25
 SET 5,A
 JR p26
p25:
 RES 5,A
p26:
 LD (s9),A
 IN A,(030h)        ; Eingang 6
 CPL  ; da negative Logik
 BIT 6,A
 JR Z,p27
 LD A,0FFh
 JR p28
p27:
 XOR A
p28:
 LD (s6),A
 LD A,(s15+1)       ; Ausgang 6
 LD (s15),A
 OR A
 LD A,(s9)
 JR Z,p29
 SET 6,A
 JR p30
p29:
 RES 6,A
p30:
 LD (s9),A
 IN A,(030h)        ; Eingang 7
 CPL  ; da negative Logik
 BIT 7,A
 JR Z,p31
 LD A,0FFh
 JR p32
p31:
 XOR A
p32:
 LD (s7),A
 LD A,(s16+1)       ; Ausgang 7
 LD (s16),A
 OR A
 LD A,(s9)
 JR Z,p33
 SET 7,A
 JR p34
p33:
 RES 7,A
p34:
 LD (s9),A
 LD A,(s9)  ; Ausgabe
 CPL  ; da negative Logik
 OUT (030h),A
 JP LOOP     ; und alles wiederholen
INIT:
 JP DATEIN   ; Daten einlesen
```

```
 ; da max case selten, wird er bei Maxcycle nicht
 ; voll angerechnet
timexec:     ; Decrementiert (IX).. Timer
 LD A,(IX+3)        ; Check ob Null
 OR (IX+4)   ; Einschaltverz.
 OR (IX+5)
 OR (IX+6)
 SUB 1       ; Trick nur dann carry wenn Ergebnis 0 war
 CCF  ; Nun Carry wenn <> 0
 LD A,(IX+3)        ; Lsb decrementieren
 SBC A,0     ; Subtrahiert 1 wenn <> 0 war
 LD (IX+3),A
 LD A,(IX+4)        ; alle Bytes
 SBC A,0     ; nur Carry
 LD (IX+4),A
 LD A,(IX+5)        ; alle Bytes
 SBC A,0     ; nur Carry
 LD (IX+5),A
 LD A,(IX+6)        ; alle Bytes
 SBC A,0     ; nur Carry
 LD (IX+6),A
 LD A,(IX+11)       ; Check ob Null
 OR (IX+12)  ; Einschaltverz.
 OR (IX+13)
 OR (IX+14)
 SUB 1       ; Trick nur dann carry wenn Ergebnis 0 war
 CCF  ; Nun Carry wenn <> 0
 LD A,(IX+11)       ; Lsb decrementieren
 SBC A,0     ; Subtrahiert 1 wenn <> 0 war
 LD (IX+11),A
 LD A,(IX+12)       ; alle Bytes
 SBC A,0     ; nur Carry
 LD (IX+12),A
 LD A,(IX+13)       ; alle Bytes
 SBC A,0     ; nur Carry
 LD (IX+13),A
 LD A,(IX+14)       ; alle Bytes
 SBC A,0     ; nur Carry
 LD (IX+14),A
 LD A,(IX+1)        ; neuer Wert
 OR A
 JR Z,timto0        ; Dann Eingang auf 0 und Sprung
 LD A,(IX+2)        ; vergleich alter Wert
 OR A ; Nur wenn vorher 0, Flanke da
 JR NZ,tiskip1
 LD A,(IX+7)        ; Einschalt-Timer neu setzen
 LD (IX+3),A
 LD A,(IX+8)        ; Einschalt-Timer neu setzen
 LD (IX+4),A
 LD A,(IX+9)        ; Einschalt-Timer neu setzen
 LD (IX+5),A
 LD A,(IX+10)       ; Einschalt-Timer neu setzen
 LD (IX+6),A
tiskip1:
 LD A,(IX+3)        ; Check ob Null
 OR (IX+4)   ; Einschaltverz.
 OR (IX+5)
 OR (IX+6)
 JR NZ,tiskip2
 LD (IX+0),0ffh     ; Ausgang geht auf 1
tiskip2:
 JR timfin
timto0:      ; Eingang ist auf 0
 LD A,(IX+2)        ; vergleich alter Wert
 OR A ; Nur wenn vorher 1, Flanke da
 JR Z,tiskip3
 LD A,(IX+15)       ; Ausschalt-Timer neu setzen
 LD (IX+11),A
 LD A,(IX+16)       ; Ausschalt-Timer neu setzen
 LD (IX+12),A
 LD A,(IX+17)       ; Ausschalt-Timer neu setzen
 LD (IX+13),A
 LD A,(IX+18)       ; Ausschalt-Timer neu setzen
 LD (IX+14),A
tiskip3:
 LD A,(IX+11)       ; Check ob Null
 OR (IX+12)  ; Einschaltverz.
 OR (IX+13)
 OR (IX+14)
 JR NZ,tiskip4
 LD (IX+0),0        ; Ausgang geht auf 0
tiskip4:
timfin:      ; Final Prozessing
 LD A,(IX+1)
 LD (IX+2),A
 RET  ; Unterprogramm Ende timexec


 DSEG
s0: DEFB 0   ; Eingang 0
s8: DEFB 0,0        ; Ausgang 0
s17: DEFB 0,0,0     ; Timer 0 : value next old
 DEFB 0,0,0,0       ; Stand Einschalt
 DEFW 1814,0        ; Einschaltverz. ca. 1000ms
 ; Ein: min = 948ms, max = 1051ms
 DEFB 0,0,0,0       ; Stand Ausschalt
 DEFW 1814,0        ; Ausschaltverz. ca. 1000ms
 ; Aus: min = 948ms, max = 1051ms
s1: DEFB 0   ; Eingang 1
s10: DEFB 0,0       ; Ausgang 1
s2: DEFB 0   ; Eingang 2
s11: DEFB 0,0       ; Ausgang 2
s3: DEFB 0   ; Eingang 3
s12: DEFB 0,0       ; Ausgang 3
s4: DEFB 0   ; Eingang 4
s13: DEFB 0,0       ; Ausgang 4
s5: DEFB 0   ; Eingang 5
s14: DEFB 0,0       ; Ausgang 5
s6: DEFB 0   ; Eingang 6
s15: DEFB 0,0       ; Ausgang 6
s7: DEFB 0   ; Eingang 7
s16: DEFB 0,0       ; Ausgang 7
s9: DEFB 0   ; Merker fuer Port 0x30
 DEFS 50
stack:
 END start
```

```
Kontaktplan, (C) 1987, Rolf-Dieter Klein 1.1

  E0      E1                                                    A0
 +-[ ]--+-[ ]--+-------------------------------------------+--( )--+
 + E0                                                         SA1  !
 +-[ ]--+-----------------------------------------------------+--( )--+
 + E1                                                         RA1  !
 +-[ ]--+-----------------------------------------------------+--( )--+
 + E0      A3                                                  A2  !
 +-[ ]--+-[ ]--+-------------------------------------------+--(/)--+
 + E1      A2                                                  A3  !
 +-[ ]--+-[ ]--+-------------------------------------------+--(/)--+
 + E0                                                          T0  !
 +-[ ]--+-----------------------------------------------------+--( )--+
 + T0                                                          A4  !
 +-[ ]--+-----------------------------------------------------+--( )--+
 !                                                                 !
 !                                                                 !
```

Bild 4: Der Kontaktfahrplan

Die Befehle zur Erzeugung einer HEX-Datei, zum Brennen in Eproms und für die SBC2-Baugruppe:

L80 /P:0,/D:8000,name,name/N/X/E.

Der SPS-Compiler erzeugt Code- und Datensegmente (CSEG sowie DSEG); dadurch lassen sich mit dem Linker beliebige Adressbereiche für RAM und ROM festlegen.

5. Der Kontaktplan

Natürlich besteht weiterhin die Möglichkeit, einen Kontaktplan auszugeben. Der Aufruf wird durch

KPLAN name

erzeugt.

Dieses Programm erzeugt eine Datei name.PLN, die den Kontaktplan im ASCII enthält. Der Kontaktplan kann dann am Bildschirm oder über einen Drucker ausgegeben werden. Bild 4 zeigt als Beispiel den Kontaktplan der vorgenannten Anwendung.

6. Zusammenfassung

Mit dem SPS-Compiler beginnt eine neue Möglichkeit, kleine, billige Rechnersysteme im industriellen Einsatz – und natürlich auch im privaten Anwendungsbereich – einzusetzen. Der SPS-Compiler läuft sowohl auf dem NDR-Computer mit

Z80/CP/M-Ausbau, als auch auf dem IBM-Personal-Computer.

Zur Übersetzung muß allerdings ein Makroassembler unter CP/M verfügbar sein, d. h., die vom IBM-PC erzeugten Daten müssen auf einen Z80-Rechner übertragen werden, um hier die Assemblierung vorzunehmen.

Der SPS-Compiler ist ein sehr preiswertes Produkt, das sicher sehr viele Freunde finden wird.

Empfohlene Konfiguration für SPS-Compiler:

Rechner mit CP/M 2.2 (mc oder NDR).

11168 SPS-Compiler (NDR) (5¼") 80)
10442 Word-Star Textverarbeitung
11058 Microsoft-Entwicklungspaket (M80)

10349 Promer
11006 ev. Christiani-SPS-Lehrbriefe
ev. EIN/AUS oder IOE2

In eigener Sache

Wie versprochen erscheint diese *LOOP* noch vor Weihnachten, damit Sie über die Feiertage etwas zu lesen haben und wir auch allen unseren Lesern ein frohes Fest sowie ein gutes neues Jahr wünschen können, was wir hiermit in aller Herzlichkeit tun!

Die Nachfrage auf Grund der in der letzten *LOOP* veröffentlichten Preisreduzierungen sowie der neuen Produkte haben uns in den letzten Wochen ganz schön ins Schwitzen gebracht! Dennoch ist diese *LOOP* wieder voll geworden – diesmal liegt das Schwergewicht der Artikel bei den Einsteigern und bei einfacheren Anwendungen – aber, Cracks, keine Angst, wir vergessen Euch nicht!

Einige Leser arbeiten schon kräftig am **LOOP-Wettbewerb,** den wir in Nr. 15 ausgeschrieben haben. Auf Grund der schon eingegangenen Wünsche, die Ferienzeit noch voll ausnützen zu können, verlängern wir den **Einsendeschluß vom 31. Dezember 1987 auf 31. Januar 1988** Das ist aber unwiderruflich der letzte Termin! Also – alle Anwender, die noch nicht gestartet sind, ran an den Lötkolben oder an die Tastatur! Einige Anregungen, wie eine solche Arbeit aussehen sollte, finden Sie auch in dieser *LOOP*. Sehen Sie sich einmal den Artikel „Messen von analogen Größen mit dem Einsteigerpaket" an!

Nun, nochmals ein frohes Fest wünschen Ihnen

Rolf-Dieter Klein **Gerd Graf**

Zweite Software-Partner-Tagung im Sauerland

Das diesjährige Treffen der Programmierer für den NDR-Computer wurde vom Elektronikladen Detmold veranstaltet. Der Ort der Zusammenkunft war Villingen im Sauerland.

Schon am Freitag, den 30. Oktober, waren die meisten der ca. 20 geladenen Software-Partner angereist. Bei einem gemütlichen Zusammensein wurden die ersten Kontakte geknüpft und für ein gegenseitiges Kennenlernen gesorgt.

Am Samstag Vormittag bekam jeder Gelegenheit, seinen Rechner aufzubauen, um seine neuen Programme vorzustellen. Dabei wurden alle Teilnehmer mit Informationen ausgesprochen vollgepumpt. Ab nachmittags wurden Workshops gebildet, um neue Strategien über zukünftige Projekte auszubrüten. Die Köpfe der Software-Partner rauchten und die Zeit verging im Eiltempo.

Aus der Erfahrung der versierten Programmier entstand z. B. ein Pflichtenheft über ein neu zu überarbeitendes JADOS-Betriebssystem. Außerdem wurden die zukünftigen Betriebssysteme für den NDR-Computer besprochen. Es werden in Zukunft für die CPU's 68xxx drei verschiedene angeboten:

- JADOS
- CPM68K
- OS-9.

Durch die gute Idee von unserem Software-Partner Ralf Jork sollen in Zukunft alle Programme in der Form ausgeliefert werden, daß sie mit allen drei Betriebssystemen betrieben werden können. Daß wir die GDP64 in einer neu überarbeiteten Version neben der Baugruppe COL256 und der ACRTC-Baugruppe vertreiben, wurde von allen Teilnehmern befürwortet. Später wurde in einem „Brainstorming" eine große Anzahl von Anregungen von den Software-Partnern entgegengenommen. Diese werden vom Elektronikladen und dem GES-Team sehr ernst genommen und bearbeitet.

Die Erfolge des NDR-Computers werden von den Software-Partnern sehr erfreulich aufgenommen. Es wurde über die neuen Projekte, unter anderen mit der Deutschen Bundespost, vielen Schulen und Großfirmen berichtet, die für die Zukunft auf den NDR-Computer setzen.

Es ist für die Programmierer wesentlich angenehmer, sich an eine große Arbeit zu setzen, wenn sie wissen, daß viele damit arbeiten werden.

Das „tierisch-ernste" Abendprogramm bestand aus einer Mischung von Life-Musik, Bier und mancher, einfach nicht abzuschaltenden „Syntaxanalyse".

Nach dem gemeinsamen Sonntagsfrühstück machten sich die Teilnehmer auf den Weg, um wieder in die verschiedensten Ecken von Deutschland zurückzukehren, von wo sie den Ausflug ins Sauerland begonnen hatten.

Zusammenfassend kann berichtet werden, daß die Software-Partner durch eigenes Engagement und kreatives Tun, einen wesentlichen Beitrag für das gesunde Gedeihen des NDR-Computers geleistet haben.

Wir dürfen ihnen allen nochmals ein Kompliment und ein Dankeschön, in unserem und im Auftrag aller NDR-Computer Anwender aussprechen, und freuen uns schon jetzt auf die 3. Software-Partner-Tagung im nächsten Jahr, die dann wieder im Allgäu sein wird.

LOOP-Termine 1988

LOOP	Redaktionschluß KW/Datum		Auslieferung ab KW/Datum	
17	04	31. 01. 88	6	08. 02. 88
18	12	21. 03. 88	14	05. 04. 88
19	20	16. 05. 88	22	01. 06. 88
20	28	11. 07. 88	30	25. 07. 88
21	36	05. 09. 88	38	19. 09. 88
22	44	01. 11. 88	46	14. 11. 88

Fehlerberichtigung aus LOOP 14 und 15

Das Programm Datei II, von unserem Softwarepartner Manfred Zehner, wird nicht als EPROM-Version, sondern nur als Diskettenversion verkauft. Hier noch einmal die Artikelnummern der verschiedenen Versionen:

11061 Datei II unter JADOS, 3½" 80 SP
11062 Datei II unter JADOS, 5¼" 80 SP
11091 Datei II unter JOGIDOS, 3½" 80 SP
11092 Datei II unte JOGIDOS, 5¼" 80 SP

Der Artikel „FLOMON 4.2 – neue Möglichkeiten" (LOOP 15/S. 12) stammt nicht von Rolf-Dieter Klein, sondern von Herrn Frank Ehrensperger, Filialleiter München. Ehre wem Ehre gebührt!

NEU!

Aktion Steckbrief

NEU!

Wie Sie bereits in *LOOP* 15 lesen konnten, haben wir den ersten unserer Software-Partner vorgestellt. Genau das ist die Aufgabe unserer neuen Serie „Aktion Steckbrief".

Nachdem wir festgestellt haben, daß hinter den von uns betriebenen Programmen wie JADOS, RL-BASIC, HEBAS, SCHACH usw. höchst interessante Personen stecken, möchten wir Ihnen diese nicht vorenthalten. Sie können ab jetzt in jeder weiteren *LOOP*-Ausgabe einen neuen Software-Partner kennenlernen.

Bald werden Sie sehen, daß der Unterschied der vorgestellten Programmierer zu Ihnen oft kleiner ist als erwartet.

Übrigens, als Software-Partner von GES werden alle diejenigen Programmierer bezeichnet, von denen wir ein Programm im Lieferumfang vertreiben.

Die Reihenfolge der Vorstellung in unserer „Aktion Steckbrief" erfolgt strikt nach der Einsendung der jeweiligen Lebensläufe.

Nun aber los: Es geht weiter mit dem 2. Software-Partner, dem frischgebackenen Dipl.-Ing. Uwe Koch, der Ihnen durch eine Vielzahl von Programmvorschlägen bekannt sein dürfte. Außerdem hat er einen großen Beitrag für die JADOS-Tooldiskette geleistet.

- Uwe Koch,
  Frankenstraße 25, 5880 Lüdenscheid,
  Telefon: (02351) 26192
- geboren am 29. September 1960
- Ausbildung: Studium Nachrichtentechnik an der FH Dortmund
  Abschluß: Diplom-Ingenieur (FH)

- Programmieren:
- 1977 wählte ich Informatik im Gymnasium, Programmierung in Basic
- 1979 bekam ich als ersten eigenen Rechner einen in Basic programmierbaren Taschenrechner (SHARP PC 1211). Zusätzlich habe ich am Rechner eines Freundes programmiert (GENIE II)
- Während des Studiums Vorlesungen in FORTRAN und APL
- ab 1984 programmieren in Z80-Assembler auf NDR-System
- ab 1985 programmieren in RDK-Basic auf NDR-System
- ab Mitte 1985 unterrichte ich an der VHS Basic für C64
- ab Ende 1985 programmieren in TurboPascal
- ab 1986 programmieren in 68008-Assembler auf NDR-System
- ab Mitte 1986 programmieren in RL-Basic
- ab 1987 Erstellung von TurboPascal-Programmen für Industrie-Anwendungen auf IBM-XT

● **Produkte:**

Z80:

- BEEP	(RDK-Basic)
- RENUMBER	(RDK-Basic)
- RENEW	(RDK-Basic)
- SOFT-CHARACTER	(EGRUND/RDK-Basic)
- FIGUR	(EGRUND)
- FUNKTIONSPLOT	(RDK-Basic)
- RFLO	(CP/M 2.2)

68008 / 680xx:

- DRUCKE	(EGRUND)
- ASSLIST, PASLIST	(EGRUND)
- EXIT	(CP/M-68K)
- KOPIE	(JADOS)
- HARDCOPY	(JADOS, CP/M-68K, SYSTEMA-DOS)
- SYSTEMTEST	(CP/M-68K)
- ASSLIT	(JADOS)
- PASCAL/S-SYSTEM	(JADOS)
- EGRUND auf BANKBOOT	(68008)

Hardware:

- Volldecodierung für IOE-Centronics

Zum NDR-Computer kam ich 1984, weil ich ein preisgünstiges, ausbaufähiges Z80-System suchte. Die fertigen Rechner, die damals bezahlbar waren (C64, Genie II, Sinclair Spectrum usw.), waren mir zu unflexibel. Ich suchte ein System, das „beliebig" erweitert werden konnte, um es meinen persönlichen Wünschen anzupassen. Das war nur mit einem Selbstbau-System möglich. Unter diesen Systemen waren einige schon in der Grundausstattung für meinen „Hobby-Etat" zu teuer, da sie direkt mit Diskettenlaufwerken und Betriebssystemen ausgestattet werden mußten (mc-CP/M-Computer). Daher fiel meine Wahl auf den NDR-Klein-Computer, von dem ich auch jetzt noch begeistert bin.

Jetzt lieferbar - Jetzt lieferbar

MLA V 1.4 für JADOS ist da!

MLA ist ein Macro-Link-Assembler zum vereinfachten Erstellen von 680xx Assemblerprogrammen. Der Programmierer kann durch die Definition von Macros und globalen Unterprogrammen seine Programme sehr vereinfacht und übersichtlich erstellen. Die einmal erstellten Unterprogramme können immer wieder von neu geschriebenen Programmen benutzt werden. Dieses ist interessant bei Grafikprogrammen für GDP / COL256 / ACRTC / UHR und ähnlichem.

Wie und wann wird MLA 1.4 eingesetzt und benutzt?

MLA wird immer dann eingesetzt, wenn Programme auf Grund der Länge unübersichtlich werden. Ist dieses der Fall, so teilt man das Programm in die wichtigsten Bestandteile auf. Dieses sind Hauptprogramm, Unterprogramm, Texte und Speicherfestlegungen. Weiterhin passiert es sehr oft, daß Programmteile immer wieder unverändert benötigt werden, aber jedesmal mit unterschiedlichen Werten, Registern oder Konstanten. Hierbei sollte man nur einmal an eine Textausgabe über WRITE denken. Mit Macros wird ein solcher Aufruf einmal definiert und dann im Programm nur noch mit einer Zeile aufgerufen. Damit kann die Textlänge stark verkürzt werden und das Programm wird im Gegensatz zu vorher nicht viel länger. Dadurch wird das Programm auch erheblich übersichtlicher und einfacher zu durchschauen.

Was ist denn ein Macro und wie wird es aufgerufen?

Ein Macro ist eine definierbare Abfolge

von Befehlen, in die unterschiedliche Werte, Register und Konstanten bei dem Aufruf eingesetzt werden. Dabei kann es sich zum Beipiel um eine Textausgabe handeln. Der Programmtext selbst ist fast immer gleich und unterscheidet sich nur in den eingesetzten Werten und Konstanten. Also wird diese Befehlsabfolge vorher einmal definiert, aber die Werte und Konstanten durch Variablen ersetzt. In diese Variablen werden dann beim Aufruf des Macro die entsprechenden Werte und Konstanten eingesetzt. Das nebenstehende Beispiel zeigt ein Macro an Hand einer Textausgabe.

MLA 1.4 arbeitet ähnlich dem Assembler in JADOS. Schon beim Aufruf von MLA 1.4 werden die zu übersetzenden Dateinamen und Optionen mit angegeben. Dadurch werden diese lästigen Eingaben im Assemblerlauf gespart und man kann sich anderen wichtigeren Dingen (Kaffee trinken . . .) zuwenden. Da mit MLA keine oder nur selten einzelne Texte übersetzt werden, muß zuvor eine Antwortdatei, ähnlich dem Assembler bei JADOS, erstellt werden. Die Antwortdatei wird beim Start von MLA angegeben. Nach der Antwortdatei werden die Optionen eingegeben. Keine Option bedeutet nur Fehlerausgabe. Die nebenstehende Tabelle gibt die Optionen wieder.

Wie zu erkennen ist, kann das Listing auch in eine Datei ausgegeben werden.

```
Befehle:                    35  38     Komentare:

.MACRO print                 * Macro heißt "print".
{                            * mit { beginnt die Definition.
move.b #&0,d0                * erste Variable (größe) eintragen.
move.w #&1,d1                * zweite Variable (X-Pos.) eintragen.
move.w #&2,d2                * dritte Variable (Y-Pos.) eintragen.
lea &3(pc),a0                * vierte Variable (Text) eintagen.
move.w #!write,d7            * Funtion WRITE in D7 laden.
trap #1                      * Befehl ausführen.
}                            * Macro "print" abgeschloßen.


.print $21,10,100,test          * Schrift-Größe $21   1.Variable
bra xxx                         * X-Pos. 10           2.Variable
                                * Y-Pos. 100          3.Variable
test: dc.b 'Test Text',0        * Text   test         4.Variable
```

Das hat den Vorteil – bei langen Programmen – nicht gleich 100 Seiten Papier zu bedrucken oder dauernd auf den Bildschirm zu starren, bis eine bestimmte Stelle erreicht ist. Außerdem bleibt das Listing immer vorhanden. Als weiteres kann die Symboltabelle gespeichert werden. Das hat den Vorteil, daß diese, ebenso wie das Listing, immer vorhanden bleibt.

MLA ist kein neuer Assembler, sondern nur ein Programm, das den Grundprogramm-Assembler in seinen Möglichkeiten erweitert und vereinfacht.

```
-P          => Ausgabe des Listing auf Drucker.
-C          => Ausgabe des Listing auf den Bildschirm.
-D          => Ausgabe des Listing in eine Datei.
-S          => Speichern der Symboltabelle als Text.
-U          => Programm nicht speichern.
-FDatei     => Anderer Programmname (Datei.68k).
```

TDS –

eine GEM-ähnliche Benutzeroberfläche für CP/M-68K

Als Benutzer des NDR-Computers mit CP/M-68K haben Sie sich bestimmt schon oft darüber geärgert, mit einer recht unbequemen Systemoberfläche arbeiten zu müssen. Mit TDS wird das nun anders!

Das System ist fast vollständig in Modula 2 geschrieben und arbeitet ähnlich dem von ATARI-ST bekannten GEM. Neben grafischer Darstellung der Directory und der Diskettenlaufwerke sowie Unterstützung der Maus, bietet es zusätzlich Funktionen, für die normalerweise besondere Dienstprogramme benötigt werden.

TDS wird wie ein gewöhnliches Programm gestartet und erfordert keinerlei Systemänderung. Der Rechner muß folgende Anforderungen erfüllen:

- 680xx-System
- Grundprogramm Version ab 4.3
- CP/M-68K
- Hardcopy/Maus-Karte
- Atari-Maus oder serielle Maus

Folgende Funktionen sind in TDS implementiert:

- grafische Darstellung der Directory mit automatischem Ordnersystem für nicht ausführbare Dateien,
- Auswahl einer Datei für eine beliebige Operation durch Selektieren (Anklicken), Tippen überflüssig,
- freies Kopieren zwischen den Laufwerken (User-Bereichen) durch Selektion der Datei(en) und des Ziellaufwerks,
- Info: freier Diskettenplatz, Filegrößen
- Anzeigen von Textdateien, positionieren innerhalb der Datei mit „Scrollbar",
- Hexdump beliebiger Dateien oder des Speichers (mit Hilfe von Scrollbar),
- Drucken von Dateien,
- Möglichkeit der Eingabe einer normalen CP/M-Kommandozeile (keinerlei Einschränkung der normalen CP/M-Funktionen),
- Fehlermeldungen und Infos werden in „Errorboxen" angezeigt,
- Start von Programmen und Submit-Dateien durch Anklicken.

Während seiner Benutzung liegt TDS in der TPA wie jedes andere Programm. Beim Start eines Programms, einer Submit-Datei oder von CP/M-Kommandozeilen wird das System komplett verlassen und der Benutzer befindet sich wieder in der gewohnten CP/M-Umgebung. Nach Beenden des gestarteten Programms lädt sich das System automatisch wieder von der Diskette nach. Aus Geschwindigkeitsgründen empfiehlt es sich daher, TDS auf der RAM-Floppy zu betreiben.

TDS ist eine ergonomische Bedienungsoberfläche, die eine angenehme Benutzung von CP/M ermöglicht.

COL 256

Info über den momentanen Stand der aktuellen Hard- und Software.

Von Kei Thomsen

Die COL 256 ist keine billige Grafikkarte. Die COL kann mehr als man denkt. Mit der COL 256 lassen sich Bilder von 256 x 256 Punkten in 256 Farben bis zu 640 x 512 in 16 Farben darstellen. Da die Grafikkarte nur mit einem einfachen CRT-Controller 6845 ausgestattet ist, der nur für die Ausgabe des Videobildes zuständig ist, erge-

ben sich hierdurch viele Vorteile, aber auch einige Nachteile. Ein schneller Grafikprozessor, wie der ACRTC HD63484 von Hitachi, hat viele fertige Grafikkomandos, die in sich recht schnell sind. Wenn es dabei aber um einzelne Punkte oder ganze Speicherblöcke geht, die es zu verschieben gilt, ist ein solcher Prozessor sehr langsam. Deshalb benutzen große Grafikcomputer zur Erstellung von Werbebildern oder ähnlichem auch nur den ganz einfachen 6845 CRT aus der COL. (Siehe dazu „Traumwelt aus dem Rechner" C'T 11/87, Seite 92 – 98.) Denn bei den Supercomputern kommt es nicht auf eine schnell ausgefüllte Fläche, Zoom oder seitlichem Scroll an, da dieses alles viel zu künstlich aussieht, sondern auf einen schnellen Speicherzugriff auf das Videoram, um die Bilder aufzubauen. Dabei werden keine Linien mehr gezeichnet, sondern jeder Bildpunkt einzeln in seiner Farbe und Helligkeit berechnet und dann gesetzt, um der Realität möglichst nahe zu kommen. Dieses läßt sich mit einem ACRTC nicht durchführen, da dabei die Übergabe viel zu umständlich ist und zu lange dauert. Weiterhin kommt es auf viele Farben an. Mit der COL 256 und der CLUT können 256 von 262144 (neuerdings auch aus 16 Millionen!) gleichzeitig dargestellt werden. Mit einem kleinen Programm läßt sich das sogar auf über 2000 Farben gleichzeitig steigern. Diese Farbenvielfalt läßt sich besonders gut an der Fractal-Demo und den Bildern die vom Amiga überspielt werden, darstellen. Obwohl die Auflösung dabei mit 256 x 256 bzw. 320 x 200 nicht sehr hoch ist, sieht das Bild sehr gut aus, weil durch die feinen Übergänge bei den Farben einzelne Bildpunkte fast nicht mehr zu sehen sind. Wie schon oben erwähnt, läßt sich die Bildauflösung verdoppeln. Dabei wird ein einfacher Trick angewandt. Es wird die 8 Bit Bildinformation (256 Farben) in 2 mal 4 Bit (16 Farben) aufgeteilt. Dadurch liegen bei der Ausgabe von einem Byte gleich 2 Punkte nebeneinander. Es wird also jeder Punkt in 2 halb so breite Punkte aufgeteilt. Ebenso läßt sich die Auflösung in senkrechter Richtung verdoppeln. Dieses geschieht mit dem sogenannten Interlace Mode. Dabei wird das Bild nicht mit 50 Herz, sondern nur mit 25 Herz erneuert. Das Bild wird aber trotzdem mit 50 Herz ausgegeben. Wie das? Ganz einfach. Das Bild besteht aus 2 sogenannten Halbbildern. Beim ersten Durchlauf des Bildstrahls über den Monitor werden alle geraden Zeilen beschrieben, beim zweiten Durchlauf alle ungeraden Zeilen. Dadurch kann das Bild bei dünnen waagerechten Linien zwar etwas flimmern, aber dies stört bei großen Flächen überhaupt nicht.

Ein Editor auf der COL 256 ist inzwischen fertig. Dabei handelt es sich um einen sehr schnellen Editor, mit einer Bildauflösung von 640 x 256 Punkte in 16 Farben. Auf eine Breite von 640 Punkte passen natürlich erheblich größere Zeichen, als auf die GDP mit 512 Punkte. Die Zeichen sind 8 Punkte breit und 12 Punkte hoch. Dadurch sieht der Text wie 80 Zeichen aus, und kann bequem auch aus 2 Meter Entfernung gelesen werden. Obwohl der gesamte Bildschirm bei jedem Scrollvorgang völlig neu aufgebaut werden muß, das sind immerhin 96 KByte Daten, kann mit einem 68008 MHz noch 4 mal in der Sekunde gescrollt werden! Bei einem 68020 16 MHz ist das sogar 12 mal in der Sekunde. Dieser nicht sehr große Zeitunterschied liegt daran, weil der 68008 4 Byte mit einem Langwortzugriff in die COL schreibt und 68020 dazu 4 mal ein Byte übertragen muß. Der Editor ist mit vielen Extras versehen, wodurch das Arbeiten zur Freude wird, speziell bei sehr langen Texten. Dabei ist er mit 8 KByte auch nicht sehr lang, weil alle wichtigen Variablen immer in Registern gehalten werden können, wodurch die hohe Geschwindigkeit beim Scrollen zustande kommt.

Weiter möchte ich hier behaupten, daß die COL bei Textdarstellung mit Scrollen schneller ist als die GDP (ohne Modifikation mit Hardscroll) und sogar schneller als die ACRTC. Diese Behauptung kann ich deshalb aufstellen, weil ich eine schnellere Methode als die altbekannte bei der Textdarstellung entdeckt habe. Näheres dazu später, wenn der Texttreiber für CP/M, JADOS . . . fertig ist.

Anmerkung der Redaktion:

Auch für die Z80-Benutzer ist die COL 256 Baugruppe interessant. Wir arbeiten derzeit an einem speziell für die COL ausgelegten FLOMON, um unter dBase, TURBO-PASCAL etc. die COL 256 zu nutzen.

Microsoft

kündigt Liefertermine für MS-OS 1.0 und MS-OS/2 1.1 an

(München) – Microsoft gab in diesen Tagen die Auslieferterminе für die Betriebssysteme MS-OS/2 1.0 und MS-OS/2 1.1 (einschließlich Presentation Manager und grafische Bedieneroberfläche für OS/2) zur Auslieferung an Software-Entwickler und OEM-Kunden bekannt.

Die Endversion (Final Release) des Microsoft OS/2 1.0-Software-Development-Kits wird im November erstellt. OEM-Kunden steht diese Version von MS-OS/2 ab Dezember zur Verfügung. Es ist zu erwarten, daß die ersten Ankündigungen über die Verfügbarkeit von MS-OS/2 durch die OEM-Kunden in naher Zukunft erfolgt.

Die erste Version von MS-OS/2 1.1 wird Anfang 1988 als Teil des MS-OS/2-Software-Development-Kits an Software-Entwickler ausgeliefert. Die endgültige Version von OS/2 1.1 steht im vierten Quartal 1988 zur Verfügung.

Darüber hinaus wurde bekannt gegeben, daß Microsoft die IBM mit seiner OS/2-LAN-Technologie beliefern wird. Microsoft arbeitet mit der IBM zusammen, weil IBM Elemente der Microsoft-LAN.Technik in ihre für das /2/2 angebotene lokale Netzwerklösung integrieren wird.

Ebenfalls zusammen mit der IBM wird Microsoft Aktivitäten starten, um in der Industrie OS/2-Applikationen zu forcieren. Diese Maßnahmen umfassen die Lieferung von Kopien des OS/2-Betriebssystems und das Development-Tool-Kits sowie die Veranstaltung von Symposien. Während der „Spring Comdex" in Atlanta wird ein Symposium stattfinden, das von Microsoft und der Firma IBM gesponsert ist. OS/2 wird auf dem mc-modular-AT laufen!

Vorankündigung!

Neue Revision der TERM1

Die TERM1 ist auf Grund ihrer vielseitigen Einsetzbarkeit eine der meistgekauften Baugruppen. Diese Tatsache war mit ausschlaggebend für die Verbesserung und Anpassung der TERM1 an den Stand der Technik.

In den folgenden Punkten werden die wichtigsten Änderungen kurz zusammengefaßt:

1. Die Schwingquarze wurden durch Quarzoszillatoren ersetzt.

2. Der POWER ON RESET wurde technisch verbessert und über Jumper einstellbar auf den BUS gelegt.

3. Es sind die Grafikprozessoren GDP 9365, 9366 oder 9367 einsetzbar (die jeweiligen Prozessoren sind über Jumper einstellbar). Default eingestellt ist der GDP9367.

4. Statt 8k EPROM und 2k RAMs sind jetzt folgende Speicher einsetzbar:
   ROM: 8k x 8 und 32k x 8
   RAM: 8k x 8 und 32k x 8

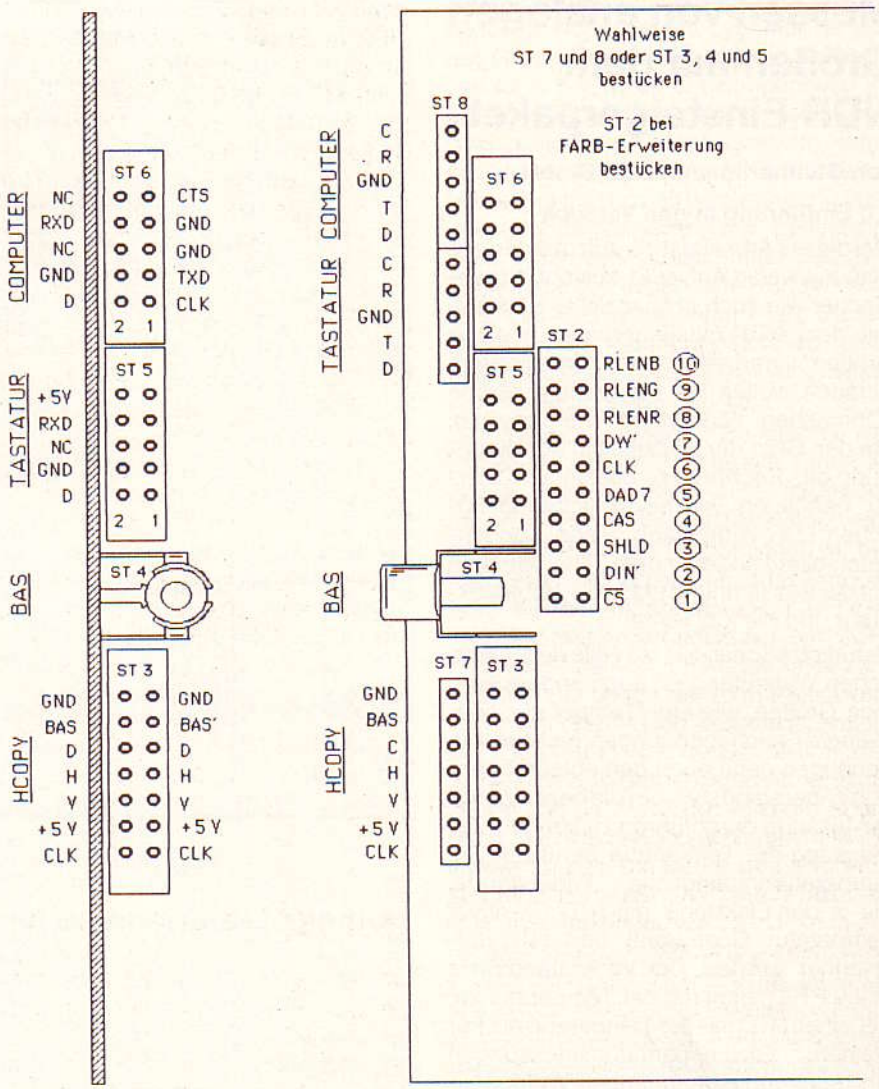
5. Es wird jetzt eine „Fast Version" der TERM mit dem Z80B und den schnellen STIs (6 MHz) angeboten. Außerdem kann die Computerschnittstelle bis 38,4 KBaud betrieben werden.

6. Die TTL-Monitorsignale D (Video), H (Horizontal Synchronsignal) und V (Vertikal Synchronsignal) sind über Jumper invertierbar.

7. Die Hardware zur Erzeugung einer Hardcopy ohne die Hardcopybaugruppe ist vorgesehen (wird softwaremäßig aber noch nicht unterstützt).

8. Stecker zur Farberweiterung vorgesehen.

Außerdem werden die Ein- und Ausgänge der TERM neu überarbeitet. In der folgenden Abbildung sehen Sie die neue Steckerbelegung:



ST8: serielle Schnittstelle zum Computer und zur Tastatur, identisch mit den Steckern der alten TERM

ST7: TTL-Videosignale und BAS-Signal (wahlweise mit ST3 bestücken)

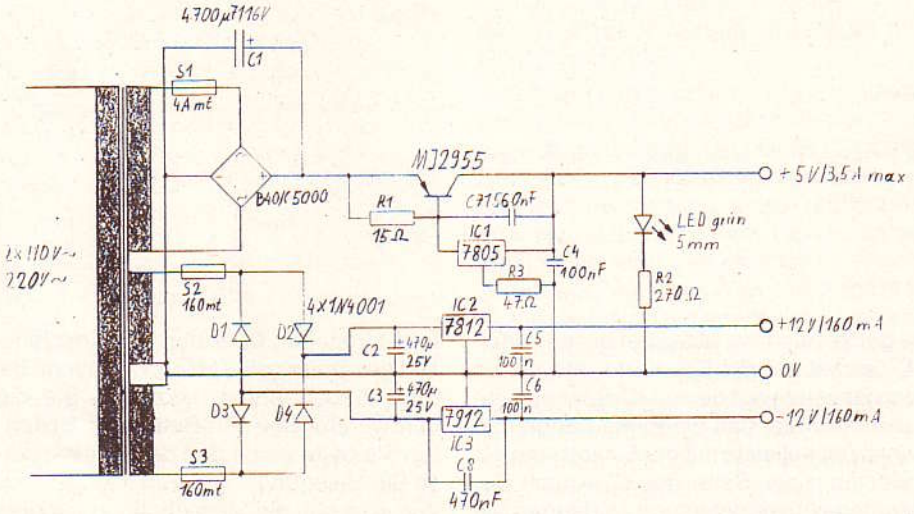
ST6: serielle Schnittstelle zum Computer (DLV11-J-Norm)

ST5: serielle Schnittstelle zur Tastatur (DLV11-J-Norm) ST6 und ST7 sind wahlweise bestückbar mit ST8

ST4: BAS Buchse (Cinch-Buchse eingelötet)

ST3: BAS und sonstige TTl-Video-Signale (auch als Verbindung zur Baugruppe HCOPY)

ST2: Stecker zur Farberweiterung



Angebot:

Solange Vorrat reicht!

Industrienetzteil getestet, ohne Gehäuse und Anschlußkabel; besonders geeignet für Subsysteme (± 12 V), aber auch für NDR-Computer oder sonstige Computersysteme ohne Laufwerke geeignet.

Technische Daten:

Spannungen: + 5 V; 3,5 A
+ 12 V; 160 mA
− 12 V; 160 mA

Schaltplan siehe unten:

Best.-Nr. 60 524

Preis incl. MWSt.: **DM 95,–**

Messen von analogen Größen mit dem NDR-Einsteigerpaket

von Günther Mader, GES GmbH

1.0 Einführung in den Versuch

Ziel dieses Artikels ist es, aufzuzeigen wie man mit wenig Aufwand, sowohl in technischer wie auch in finanzieller Hinsicht, mit dem NDR-Einsteigerpaket analoge Größen verarbeiten kann. In unserem Versuch wollen wir die analoge Größe „Ohmschen Widerstand" heranziehen, um den Grad der Helligkeit in der Umgebung des Rechners zu bestimmen. Hierzu benötigen wir einen lichtempfindlichen LDR-Widerstand, einen Kondensator, einen sogenannten Timerbaustein mit der Bezeichnung LM 555, Kupferlackdraht und die IOE-Karte.

Natürlich können Sie mit Hilfe des „Ohmschen Widerstandes" auch andere analoge Größen, wie eine Temperatur oder kleinere Gleichspannungen, messen. Sie benötigen dann eben den entsprechenden Widerstandstyp, um die beabsichtigte Messung durchführen zu können. Zur Messung der Temperatur benützen Sie temperaturempfindliche Widerstände, die in der Elektronik mit PTC (Positiver Temperatur Coeffizient) und NTC bezeichnet werden. Der Widerstandswert eines PTC nimmt mit der Temperatur zu, der eines NTC mit der Temperatur ab. Für kleinere Gleichspannungsmessungen sind spannungsunabhängige Widerstände, sogenannte „Varistoren", im Fachhandel erhältlich. Zusätzlich angemerkt sei, daß all diese Widerstände in Normreihen eingeteilt werden (E 12, E 24, E 36). Dies ist bei der Entwicklung eigener Meßschaltungen zu beachten. Die Auswertung der eingelesenen Werte entspricht dem anschließend beschriebenen Schema.

1.1 Prinzip des Einlesevorgangs

Wie können nun analoge Größen vom Rechner verarbeitet werden? Im Prinzip wird die Zeitdauer des HIGH- bzw. LOW-Zustandes am IN-Port des Rechners ermittelt, d. h., eine analoge Größe wird in Zeitwerte umgewandelt und kann so vom Rechner interpretiert werden. Wie dies genau vor sich geht, wird in einem folgenden Abschnitt beschrieben.

1.2 Erläuterungen zur benötigten Hardware

Aber nun zur hardwaremäßigen Realisierung der Meßschaltung. Die Bauelemente werden auf die IOE-Karte gelötet, wie es Bild 1.2 zeigt. Der Timerbaustein LM555 sollte nicht direkt auf die IOE-Karte gelötet werden. Man lötet besser einen IC-Sockel auf die IOE-Karte und steckt den Baustein auf diesen IC-Sockel. Bitte beachten Sie, daß die Einkerbungen an einer Sockelleiste mit der Einkerbung auf oder an einer Seite des IC's nach der Montage übereinstimmen. Verbinden Sie die einzelnen Bausteine gemäß Bild 1.2. In Bild 1.2 stehen über den PIN-Anschlüssen des Timers die Zahlen 1 – 8. Wenn Sie Bild 1.5 betrachten, erkennen Sie die hardwaremäßige Pin-Belegung. Beachten Sie beim Verbinden der Bauteile diese Pin-Belegung.

Wie wird nun die Meßschaltung mit dem Rechner verbunden? Dies zeigt Bild 1.4.

```
MACRO-80 3.43   27-Jul-81      PAGE    1

                        .Z80
                        ;*****************************************************************
                        ;**                                                             **
                        ;**              PROGRAMM ZUR AUFNAHME EINES MESSWERTES         **
                        ;**                                                             **
                        ;**  V 1.0                 02.11.1987      (c) Graf Computer    **
                        ;**                                                             **
                        ;*****************************************************************

                                                                                ;von G.Mader

0033                                    clear     equ    0033h
001B                                    prthl     equ    001bh
8000                                    anzfeld   equ    8000h
0009                                    anzeige   equ    0009h


                                                 org 8100h

                                        ;Triggerimpuls fuer Timer setzen

8100'  21 0000          start:          ld hl,0000h          ;HL-Registerpaar mit 0000h laden
8103'  3E 00                            ld a,00h             ;Akkumulator mit 00h laden
8105'  D3 30                            out (30h),a          ;Akkumulator an Port 30h ausgeben
8107'  3E 01                            ld a,01h             ;Akkumulator mit 01h laden
8109'  D3 30                            out (30h),a          ;Akkumulator an Port 30h ausgeben


                                        ;Messwerte einlesen

810B'  DB 30            einlesen:       in a,(30h)           ;Wert der an Port 30h anliegt enlesen
810D'  E6 01                            and 01               ;Verknuepfung
810F'  CA 8116'                         jp z,adresse         ;bedingter Sprung
8112'  23                               inc hl               ;Schleifenzaehler
8113'  C3 810B'                         jp einlesen          ;unbedingter Sprung


                                        ;Anzeige des HL-Registers

8116'  E5               adresse:        push hl              ;HL-Registerinhalt sichern
8117'  CD 0033                          call clear           ;Anzeige loeschen
811A'  DD 21 8004                       ld ix,anzfeld+4      ;Anzeigenfeld wird adressiert
811E'  21 8000                          ld hl,8000h          ;HL-Registerpaar mit Adresse 8000h laden
8121'  36 89                            ld (hl),89h          ;adressierte Speicherzelle mit 'Buchstabe H' laden
8123'  23                               inc hl               ;Folgeadresse laden
8124'  36 C7                            ld (hl),0c7h         ;adressierte Speicherzelle mit 'Buchstabe L' laden
8126'  E1                               pop hl               ;Rueckholen des gesicherten HL-Wertes
8127'  CD 001B                          call prthl           ;HL-Register ins Anzeigenfeld
812A'  CD 0009                          call anzeige         ;Aufruf der Anzeigenroutine
812D'  C3 8100'                         jp start             ;Sprung zum Anfang

MACRO-80 3.43   27-Jul-81      PAGE    1-1


                                        ;Bitte beachten Sie, dass bei der Eingabe von
                                        ;Adressen zuerst das niederwertigere BYTE und
                                        ;dann das höherwertige BYTE eingegeben werden
                                        ;muss.


                        end
MACRO-80 3.43   27-Jul-81      PAGE    S

Macros:

Symbols:
8116'  ADRESSE     0009   ANZEIGE     8000   ANZFELD
0033   CLEAR       810B'  EINLESEN    001B   PRTHL
8100'  START

No Fatal error(s)
```

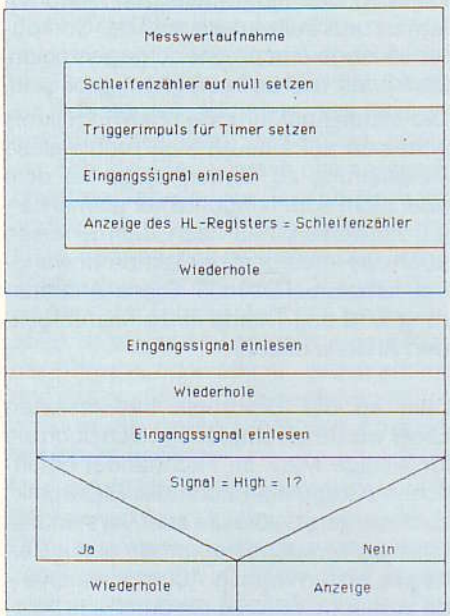


Bild 1.1 Struktogramm mit Programmodul "Einlesen"

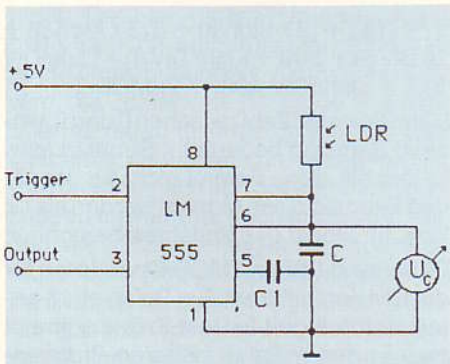


Bild 1.2 Meßschaltung

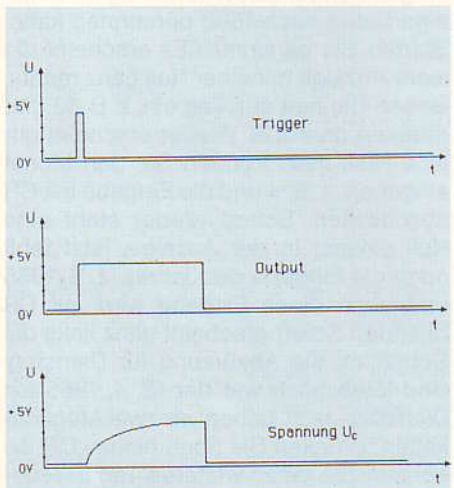


Bild 1.3 Spannungsverläufe

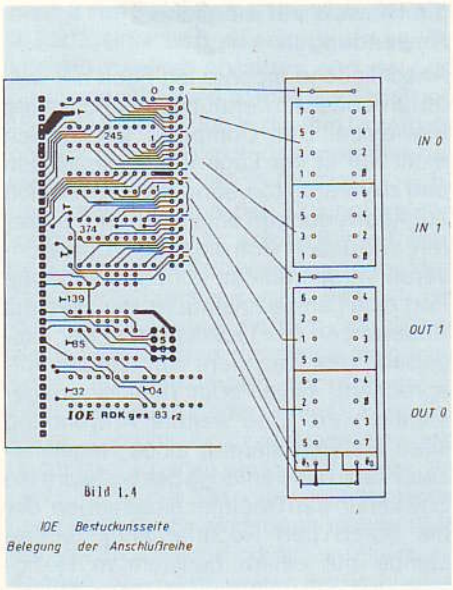


Bild 1.4

IOE Bestückungsseite
Belegung der Anschlußreihe

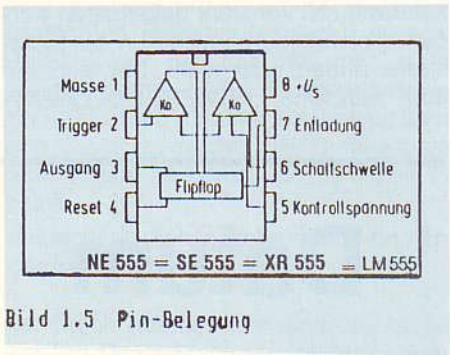


Bild 1.5 Pin-Belegung

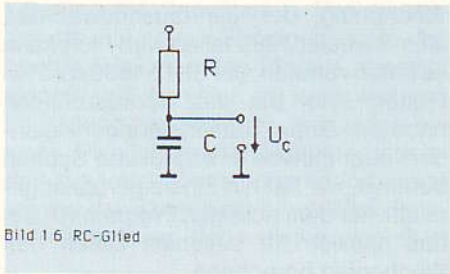


Bild 1.6 RC-Glied

Verbinden Sie „Pin2" mit OUT 0,0. Verbinden Sie „Pin3" mit IN 0,0. Die Jumperpositionen 6 und 7 auf der IOE müssen zur Porteinstellung kurzgeschlossen werden. Außerdem werden e1 und e0 mit Masse verbunden.

Nun eine kurze Erläuterung zu einem wichtigen Bestandteil der Meßschaltung, dem sogenannten „RC-GLIED" (siehe Bild 1.6). Ein „RC-Glied" wird gebildet aus der Reihenschaltung eines ohmschen Widerstandes mit einem Kondensator. Multipliziert man den ohmschen Widerstandswert mit dem Wert der Kapazität (= Kondensator), so ergibt sich eine Zeitkonstante, die man mit dem griechischen Buchstaben τ bezeichnet. Die Zeitkonstante ist kennzeichnend für das jeweilige „RC-Glied". Sie gibt an, wie schnell der Kondensator auf den Wert der Betriebsspannung aufgeladen werden kann. Näheres hierzu im folgenden Abschnitt. Nach dem gleichen Schema bilden wir unser „RC-Glied". Als ohmschen Widerstand benützen wir einen LDR-Widerstand, den kapazitiven Teil des Gliedes bildet ein Kondensator. Legt man an dieses Glied eine Spannung an, in unserem Fall ca. + 5 V, so ist der Kondensator im ersten Augenblick kurzgeschlossen, d.h., es fließt ein hoher Ladestrom, somit liegt keine Spannung am Kondensator. Nun steigt mit der Zeit die Spannung über dem Kondensator an, der Ladestrom sinkt. Nach einer bestimmten Zeit (5 x Zeitkonstante) liegen am Kondensator 99% der Betriebsspannung an. Diesen allmählichen Spannungsanstieg über dem Kondensator verwerten wir in unserem Versuch (siehe Bild 1.3).

Den zweiten wichtigen Bestandteil der Schaltung (Bild 1.2) bildet der Timer. Zu seiner Funktion läßt sich folgendes sagen. Schauen Sie zunächst noch einmal Bild 1.2 an. Ein Triggersignal, wir benützen einen kurzen Rechteckimpuls, setzt ein sogenanntes FLIP-FLOP. Der Ausgang des Timers liefert nun ein HIGH-SIGNAL. Dieses Signal liegt zwichen 2,5 V und 5 V. Die Länge der High-Phase wird durch unser RC-Glied bestimmt. Erreicht die Spannung am Kondensator den Schwellwert $^2/_3$-Betriebsspannung, in unserem Fall ca. + 3 V, so wird die Spannung am Ausgang des Timers LM555 wieder zu null. Die Dauer des „High-Zustandes" kennzeichnet die Pulsbreite des Ausgangsimpulses.

Wichtig für die Erzielung von Meßergebnissen ist eine sorgfältig abgestimmte Hardware. Unter einer sorgfältig abgestimmten Hardware versteht man die Optimierung der Schaltung. So muß z.B. die Zeitkonstante sinnvoll gewählt werden, d.h., sie darf weder zu groß noch zu klein sein. Ist die Zeitkonstante zu groß, beeinflußt der Einlesevorgang die Schnelligkeit des Programmablaufes. Ist die Zeitkonstante zu klein, wird die Messung sehr ungenau. Als Faustregel für die Größe der Zeitkonstante gelten 20 ms.

Wie ermittelt man nun den notwendigen Kondensatorwert? Zunächst mißt man mit einem analogen Vielfachmeßgerät, unter Abdunkelung des LDR, den maximalen Widerstandswert des LDR. Unter Abdunkelung deshalb, weil ein LDR bei Abdunkelung sich seinem maximalen Widerstandswert annähert. Da Zeitkonstante und Widerstandswert nun vorgegeben sind, ermittelt sich „C" aus diesen beiden Größen.

Die allgemein gültige Formel zur Berechnung der Zeitkonstanten lautet:

ZEITKONSTANTE =
Kondensatorwert x Widerstandwert

Aus dieser Formel umgestellt nach „C" ergibt sich unser Kondensatorwert:

KONDENSATORWERT =
Zeitkonstante/Widerstandswert

Hat man kein Vielfachmeßgerät zur Hand, so können Sie auch einen kleinen Kondensatorwert annehmen (z. B. 100nF). Anhand des Programms können Sie erkennen, wie genau Ihre Messung ist und ob Sie den Kondensator größer oder kleiner wählen müssen.

1.3 Erläuterungen zur benötigten Software

Wie Sie sicherlich wissen, benötigt ein Computer zur Ausführung seiner Aufgabe Software. Betrachten Sie Bild 1.1 „Struktogramm Meßwertaufnahme". Zuerst setzen wir unseren Schleifenzähler, in diesem Programm das HL-Registerpaar, auf Null. Anschließend geben wir über den peripheren Port (30h) das Triggersignal für den Timberbaustein aus. Dieser kurze Rechteckimpuls bewirkt ein ebenfalls rechteckiges Ausgangssignal, dessen Pulsbreite vom „RC-Glied" abhängt. Dies wurde in einem vorherigen Abschnitt erläutert. Über den peripheren IN'Port (30h) „fragt" der Computer ab, ob an seinem Eingang High- oder Low-Signal anliegt. Liest der Rechner z. B. 01h, also High-Signal, in den Akkumulator ein, und verknüpft man diesen Wert mit 01h, so läßt sich anhand des Wertes im HL-Register (= Schleifenzähler) aufzeigen, wie oft unser Rechner seinen In-Port nach 01h abgefragt hat, bevor an diesem Low-Signal eingelesen wurde. Dieser hexadezimale Wert ist proportional zur Pulsdauer des Ausgangssignals am Timer.

Diesen Wert kann man nun verarbeiten. Dazu ist eine Art Eichung notwendig, die von der selbstgestellten Aufgabe abhängig ist. Hier ein Vorschlag für eine zugegebenermaßen unvollkommene Eichung: Man könnte zum Beispiel den ermittelten Höchstwert einem bestimmten Helligkeitswert proportional setzen. Diesen Höchstwert teilt man durch eine Konstante und erhält somit einen Meßwert. Werden nun vom Computer weniger High-Signale ermittelt, so wird eine kleinere Zahl durch diese Konstante geteilt, und man erhält somit einen kleineren Meßwert.

Am Ende des Programms wird durch Aufruf von Unterprogrammroutinen dafür „gesorgt", daß der Inhalt des HL-Registers im Anzeigefeld erscheint. Wenn Sie jetzt den LDR abdunkeln, werden Sie bemerken, daß die Anzeige eventuell zu flackern beginnt. Der Grund ist die Inanspruchnahme des Rechners durch den Einlesevorgang. Durch die Abdunkelung steigt der Widerstandswert, d. h., der Rechteckimpuls wird länger und damit auch der Einlesevorgang. Deshalb wird die Anzeige immer seltener aufgefrischt.

1.4 Hinweis auf mögliche Anwendungen

Anschließend möchte ich noch auf weiterführende Anwendungsmöglichkeiten hinweisen. Der Computer ist nämlich nicht nur in der Lage Werte einzulesen und zu verarbeiten, sondern er kann auch auf Auswertungen entsprechend reagieren. Wie äußert sich dies in Bezug auf unseren Versuch? Man könnte über einen Port eine Lampe ansteuern, die dann auf Änderungen der Lichtstärke in der Umgebung des Rechners von diesem entsprechend heller oder dunkler ausgesteuert wird. Eine weitere Anwendung wäre die Realisierung eines Regelkreises. Hierzu wird anfangs per Software ein Sollwert in den Rechner eingegeben, der die durch den Rechner angesteuerte Lampe mit einem bestimmten Helligkeitswert aufleuchten läßt. Dieser Helligkeitswert soll konstant beibehalten werden, gleichgültig ob sich nun die Lichtstärke ändert oder nicht. Der Rechner führt nun einen ISTWERT-SOLLWERT-Vergleich durch, wobei der Istwert unserem eingelesenen „Zeitwert" entspricht. Ergeben sich nun Differenzen zwischen Istwert und Sollwert, so muß der Computer so lange entsprechend gegenregeln, bis Istwert und Sollwert gleich groß sind.

Die letzten Ausführungen zum Regelkreis scheinen auf eine an sich problemlose Realisierung zu deuten, leider ist dem aber nicht so. Ich möchte nur einmal darauf hinweisen, daß nach dem Einlesen auch das Ansteuern der Lampe einwandfrei funktionieren muß. Diese Ansteuerung wird das Thema eines nachfolgenden Artikels bilden.

Dem an der Elektronik interessierten Leser würde ich empfehlen, sich doch ein für wenige Mark im Fachhandel erhältliches „Grundlagenbuch der Elektronik" zu besorgen, da dieses zum Verständnis unseres Versuches ebenfalls einen Beitrag zu leisten vermag. Auch für kommende Versuche wäre es ein durchaus nützliches Nachschlagewerk.

Datum

Wissen Sie eigentlich schon auf welchen Wochentag der Jahrtausendwechsel, also Silvester 1999 fallen wird? Ich kann es Ihnen verraten: der 31. 12. 1999 wird ein Freitag sein. Um das herauszufinden habe ich keinen „Tausendjährigen Kalender" oder ähnlich umständliche Sachen benötigt, mir hat ein Einsteigerpaket gereicht. Mit dem richtigen Programm kann das nämlich für beliebige Daten den Wochentag berechnen.

Das ist möglich, da unser Kalender, der sogenannte Gregorianische Kalender, den Papst Gregor XIII. 1582 einführte, sehr regelmäßig ist. Bis 1582 galt der Julianische Kalender, in dem immer wieder Änderungen vorgenommen werden mußten, da die Sonne und der Mond sich nicht nach diesem Kalender richten wollten. Seit 1582 werden diese Änderungen durch die Schaltjahre vermieden. Diese werden nach einfachen Regeln ermittelt:

- Alle durch vier teilbaren Jahre sind Schaltjahre (zum Beispiel 1992 / 4 = 498, also ist 1992 ein Schaltjahr).
- Alle durch 100 teilbaren Jahre sind keine Schaltjahre, obwohl sie durch vier teilbar sind (zum Beispiel das Jahr 1900),
- Alle durch 400 teilbaren Jahre sind Schaltjahre, obwohl sie durch 100 teilbar sind (zum Beispiel das Jahr 2000).

Damit wird der Kalender recht einfach berechenbar. Man berechnet folgende Funktion:

x := (Tag + (2 x Monat) + (((3 x Monat) + 3) DIV 5) + Jahr + (Jahr DIV 4) – (Jahr DIV 100) + (Jahr DIV 400) + 2) MOD 7

Dann ist x eine Zahl zwischen 0 und 6, wobei 0 Samstag bedeutet, 1 Sonntag usw. Leider gilt diese Formel nicht für Januar und Februar, diese Monate werden als 13. bzw. 14. Monat des Vorjahres berechnet.

Dabei sind DIV und MOD Operatoren für die Division mit Rest. Ein Beispiel: 13 geteilt durch 5 gibt 2, Rest 3. Das schreibt man in den meisten höheren Programmiersprachen so: 13 DIV 5 = 2 und 13 MOD 5 = 3, also DIV liefert das Ergebnis der Division (ganzzahlig) und MOD den Rest.

DATUM ist ein Programm, das nach der obigen Formel den Wochentag für beliebige Daten nach 1582 berechnen kann. Starten Sie es einmal! Es erscheint die leere Anzeige mit einer Null ganz rechts. Geben Sie nun den Tag ein, z. B. 13 und drücken dann CR. Wieder erscheint nur eine Null. Jetzt können Sie den Monat eingeben, z. B. 4 und die Eingabe mit CR abschließen. Schon wieder steht eine Null einsam in der Anzeige, jetzt fehlt noch die Eingabe des Jahres, z. B. 1965 und auch diese Eingabe wird mit CR beendet. Sofort erscheint ganz links die Schrift di, die Abkürzung für Dienstag. Und tätsächlich war der 13. 4. 1965 ein Dienstag. Jetzt haben wir zwei Möglichkeiten: drücken Sie noch einmal CR, so können Sie einen weiteren Tag berechnen, drücken Sie irgendeine andere Taste, landen Sie wieder im HEXMON.

Schauen wir uns das Programm einmal näher an:

Es gliedert sich in drei große Teile und ein Unterprogramm. Recht einfach ist die Eingabe, da dazu nur Unterprogramme des HEXMON aufgerufen werden. Etwas schwieriger ist die Ausgabe, denn dort muß aus einer Zahl zwischen 0 und 6, die im Registerpaar BC steht, ein Text gemacht werden. Dazu wird einfach zu dieser Zahl die Startadresse einer entsprechenden Tabelle, die TagTabelle heißt, addiert. An dieser Adresse steht nun das erste Zeichen, das einfach in den Anzeigespeicher kopiert wird. Sieben Byte weiter in der Tabelle steht das zweite Zeichen, da es sieben Tage gibt. Also wird zu der Adresse noch einmal sieben addiert und das an dieser Adresse stehende Zeichen in die zweite Stelle des Anzeigespeichers kopiert.

Das eigentlich komplizierte ist aber die Berechnung der Formel. Hier muß mit 16 Bit-Werten gerechnet werden, da z. B. die Jahreszahl nicht in einem Byte dargestellt werden kann. Deshalb wird das Programm etwas länger. Außerdem gibt es keinen Z80 Befehl zum Dividieren oder um den Rest zu berechnen. Dafür mußte ein Unterprogramm her, das dividiere heißt. Da hier dieses Unterprogramm nicht sehr oft aufgerufen wird, muß es auch nicht besonders schnell sein. Es wird die „natürliche Methode" benutzt, solange die zu teilende Zahl größer oder gleich der Zahl, durch die geteilt wird ist, wird diese einfach von der anderen abgezogen. Nun muß nur noch mitgezählt werden, wie oft dies gelungen ist. Anders formuliert: Der Rechner zählt, wie oft er die Zahl, durch die geteilt wird, von der anderen abziehen kann. In unserem Beispiel vom Anfang: 13 durch 5 ist? 13 ist größer als 5, also kann 5 von 13 abgezogen werden, das gibt 8. 8 ist größer als 5, also kann 5 von 8 abgezogen werden, das gibt 3. 3 ist kleiner als 5, also sind wir fertig. Wir können zweimal abziehen und es sind drei übriggeblieben, also 13 DIV 5 ist 2 und 13 MOD 5 ist 3. Auch in diesem Unterprogramm macht es sich unangenehm bemerkbar, daß wir mit 16-Bit Zahlen rechnen müssen, denn der Z80 hat keinen Vergleichsbefehl für solche Zahlen. Man muß also die einzelnen Bytes der beiden Zahlen miteinander vergleichen. Ist das höherwertige Byte der ersten Zahl kleiner als das der zweiten, dann muß die ganze erste Zahl kleiner sein, wir sind fertig. Sind die beiden höherwertigen Byte ungleich, dann ist also das erste größer, und damit ist die ganze erste Zahl größer. Es bleibt noch der Fall, daß die höherwertigen Byte gleich sind, dann müssen wir die niederwertigen Byte vergleichen. Können wir nicht weiter subtrahieren, dann muß das Ergebnis noch in die gewünschten Register gebracht werden. Hier fällt auf, daß der Z80 auch nicht alle denkbaren 16-Bit Transportbefehle kennt, deshalb wird hier ein Umweg über den Stack gemacht. Die Befehlsfolge push hl pop bc bewirkt dasselbe wie ld bc,hl.

Vielleicht an dieser Stelle einmal eine Erklärung zum Stack:

Der Stack von dem hier die Rede ist, ist ein Top-Down-Stack (zu deutsch: Von-Oben-nach-Unten-Stapel). Das bedeutet, daß die ersten Eintragungen in den oberen Regionen des Stapels abgelegt werden und man sich mit jeder weiteren Eintragung immer weiter nach unten bewegt. Diese Vorgehensweise erscheint uns zunächst etwas befremdend, wenn wir von der Vorstellung eines Bücherstapels ausgehen, denn dieser wächst ja von unten nach oben. Vielleicht hilft Ihnen auch folgender Vergleich zum Verständnis des Top-Down-Stapels: Man bezeichnet ihn mitunter auch als „Kellerstapel", weil die deutsche Übersetzung „Von-Oben-nach-Unten-Stapel" doch etwas unbeholfen wirkt, und zum anderen der Stapel bei fortlaufender Beschikkung von einer fiktiven Decke aus in den „Keller" herabwächst. Beiden Stapelarten ist jedoch gemeinsam, daß zuerst alle darüberliegenden (beim Bücherstapel) bzw. alle darunterliegenden (beim Top-Down-Stapel) Ablagen entfernt werden müssen, um an ein bestimmtes Objekt zu gelangen. In beiden Fällen nennt man das auch LIFO-Prinzip (Last In – First Out), da das letztabgelegte (Last In) Objekt in jedem Falle als erstes (First Out) wieder vom Stapel genommen werden muß, um an eventuell höherliegende Objekte (denken Sie an den von der Decke aus herabwachsenden Stapel) gelangen zu können. Genauso arbeitet der Z80 mit seinem Stapel. Auf diesem Stapel liegen (natürlich) nur Zahlen. Diese Zahlen kommen aus den Registern des Prozessors mit dem Befehl „push . . ." und werden nachher auch wieder ins Register gebracht mit dem Befehl: „pop . . .". Der Befehl push ix legt also die Zahl, die im Register IX steht auf den Stapel. Nun kann der Prozessor also nur noch an diese Zahl heran, da sie ganz „unten" im Stapel liegt.

Bekommt er nun den Befehl pop hl, dann ist es ihm egal, aus welchem der Register die zu holende Zahl in den Stapel geschoben wurde, er nimmt sich lediglich die zuletzt abgelegte Zahl des Stapels und gibt sie in das HL-Register. So ist es also möglich, Daten von Registern in andere zu bringen, für die es keinen Transportbefehl (ld) gibt.

```
                ; Programm  DATUM     1.0
                ;
                ; Dieses Programm berechnet zu einem Datum den Wochentag

                .Z80

                Tag       EQU  80F0H  ; Speicher für den Tag
                Monat     EQU  80F1H  ; Speicher für den Monat
                Jahr      EQU  80F2H  ; Speicher für das Jahr (2 Byte)
                x         EQU  80F4H  ; Hilfsvariable

                HoleTaste EQU  000CH  ; HEXMON Unterprogramm zum Abfragen des Tastenfeldes
                GetDez    EQU  002AH  ; Liest eine Dezimalzahl in HL ein
                Clear     EQU  0033H  ; Löscht die Anzeige
                Anzeige   EQU  8000H  ; Adresse des Anzeige-Speichers

                          ORG  1050H

                DATUM:

                ; Zunächst ein Datum einlesen

1050  DD 21 07 80          ld ix,Anzeige+7 ; letzte Stelle der Anzeige steht in IX
1054  21 00 00             ld hl,0         ; Default Wert in HL
1057  CD 2A 00             call GetDez     ; Dezimalzahl einlesen
105A  22 F0 80             ld (Tag),hl     ; Ergebnis speichern, das zweite Byte wird wieder
                                           ; überschrieben
105D  DD 21 07 80          ld ix,Anzeige+7 ; letzte Stelle der Anzeige steht in IX
1061  21 00 00             ld hl,0         ; Default Wert in HL
1064  CD 2A 00             call GetDez     ; Dezimalzahl einlesen
1067  22 F1 80             ld (Monat),hl   ; Ergebnis speichern, das zweite Byte wird wieder
                                           ; überschrieben
106A  DD 21 07 80          ld ix,Anzeige+7 ; letzte Stelle der Anzeige steht in IX
106E  21 00 00             ld hl,0         ; Default Wert in HL
1071  CD 2A 00             call GetDez     ; Dezimalzahl einlesen
1074  22 F2 80             ld (Jahr),hl    ; Ergebnis speichern, diesmal sind zwei Byte gültig
1077  CD 33 00             call Clear      ; Anzeige löschen

                ; Dann daraus den Wochentag berechnen

107A  3A F1 80             ld a,(Monat)    ; Monat in den Akku laden
107D  FE 03                cp 3            ; ist es Januar oder Februar ?
107F  30 09                jr nc,MonOK     ; nein, Korrektur überspringen

                ; Korrektur für Januar und Februar, zur Berechnung der Schalttage

      C6 0C                add a,12        ; Monat um 12 erhöhen
1083  32 F1 80             ld (Monat),a    ; und wieder speichern
1086  21 F2 80             ld hl,Jahr      ; Adresse von Jahr in HL laden
1089  35                   dec (hl)        ; um eins erniedrigen, zum Ausgleich, da Monat um
                                           ; 12 erhöht wurde
                MonOK:
108A  2A F0 80             ld hl,(Tag)     ; den Tag in 1 laden, dabei kommt der Monat nach h
108D  26 00                ld h,0          ; h ist Null, HL enthält jetzt den Tag
108F  ED 5B F1 80          ld de,(Monat)   ; in E steht der Monat
1093  16 00                ld d,0          ; D ist Null, also steht in DE der Monat
1095  19                   add hl,de       ; Monat zu HL addieren
1096  19                   add hl,de       ; noch einmal addieren
1097  22 F4 80             ld (x),hl       ; und Ergebnis Speichern
109A  62                   ld h,d          ; Monat nach HL laden
109B  6B                   ld l,e
109C  19                   add hl,de       ; Monat zu HL addieren
109D  19                   add hl,de       ; noch einmal addieren, jetzt steht 3*Monat in HL
109E  11 03 00             ld de,3
10A1  19                   add hl,de       ; 3 dazu addieren
10A2  11 05 00             ld de,5         ; in DE steht 5
10A5  CD 1C 11             call dividieren ; Unterprogramm zum teilen, in HL steht jetzt
                                           ; ((3*Monat)+3)/5
10A8  ED 5B F4 80          ld de,(x)       ; Zwischenwert nach DE bringen
10AC  19                   add hl,de       ; zum bisherigen Zwischenwert addieren
10AD  ED 5B F2 80          ld de,(Jahr)    ; Jahr nach DE bringen
10B1  19                   add hl,de       ; und auch addieren
10B2  23                   inc hl          ; plus zwei,
10B3  23                   inc hl          ; also zweimal inkrementieren
10B4  22 F4 80             ld (x),hl       ; und wieder speichern
10B7  2A F2 80             ld hl,(Jahr)    ; Jahr in HL
10BA  11 04 00             ld de,4         ; durch 4 teilen,
10BD  CD 1C 11             call dividieren ; gibt die Zahl der Schaltjahre
10C0  ED 5B F4 80          ld de,(x)       ; Zwischenwert holen
10C4  19                   add hl,de       ; zum addieren
10C5  22 F4 80             ld (x),hl       ; und wieder speichern
10C8  2A F2 80             ld hl,(Jahr)    ; Jahr in HL
```

```
10CB  11 64 00          ld de,100       ; durch 100 teilen,
10CE  CD 1C 11          call dividieren ; gibt die Zahl der ausgefallenen Schaltjahre
10D1  54                ld d,h          ; nach DE kopieren
10D2  5D                ld e,l
10D3  2A F4 80          ld hl,(x)       ; Zwischenwert laden
10D6  ED 52             sbc hl,de       ; ausgefallene Schaltjahre abziehen
10D8  22 F4 80          ld (x),hl       ; und wieder speichern
10DB  2A F2 80          ld hl,(Jahr)    ; Jahr in HL
10DE  11 90 01          ld de,400       ; durch 400 teilen,
10E1  CD 1C 11          call dividieren ; gibt die Zahl der Schaltjahre, die nicht ausfielen
10E4  ED 5B F4 80       ld de,(x)       ; Zwischenwert laden
10E8  19                add hl,de       ; zum addieren
10E9  11 07 00          ld de,7         ; durch 7 Teilen,
10EC  CD 1C 11          call dividieren ; der Rest ist die Nummer des Wochentags

                ; Nun das Ergebnis ausgeben
                ; In BC steht eine Zahl zwischen 0 und 6, dabei steht 0 für Samstag, 1 für
                ; Sonntag, .... und 6 für Freitag

10EF  60                ld h,b          ; Ergebnis nach HL bringen
10F0  69                ld l,c          ; da dort gerechnet werden kann
10F1  11 0E 11          ld de,TagTabelle
10F4  19                add hl,de       ; Adrsse des ersten Buchstabens in der Tabelle
                                        ; berechnen, steht in HL
10F5  7E                ld a,(hl)       ; Buchstaben holen
10F6  32 00 80          ld (Anzeige),a  ; und in die Anzeige schreiben
10F9  11 07 00          ld de,7         ; jetzt Adresse des zweiten Buchstabens berechnen,
10FC  19                add hl,de       ; der steht sieben Byte weiter
10FD  7E                ld a,(hl)       ; zweiten Buchstaben holen
10FE  32 01 80          ld (Anzeige+1),a ; auch in die Anzeige schreiben
                Fertig:
1101  CD 0C 00          call HoleTaste  ; Auf Tastendruck warten, damit die Anzeige gelesen
                                        ; werden kann
1104  FE 57             cp 57h          ; Taste CR gedrückt ?
1106  20 03             jr nz,ENDE      ; nein ? Dann Programm beenden.
1108  C3 50 10          jp DATUM        ; sonst noch einmal starten
110B  C3 00 00  ENDE:   jp 0            ; zum HEXMON-Start springen

110E  92 92 AA A1 AA AI 8E  TagTabelle: DB  92h, 92h, 0aah, 0alh, 0aah, 0alh, 8eh
                            ; das heißt  S   S   M   d   M   d   F
1115  88 A3 A3 CF CF A3 AF          DB  88h, 0a3h, 0a3h, 0cfh, 0cfh, 0a3h, 0afh
                            ; das heißt  A   o   o   I   I   o   r
                                         *
                            dividieren:

                            ; Einfaches Unterprogramm zum Dividieren mit Rest
                            ; Für große Zahlen, die durch kleine geteilt werden, ist es nicht besonders
                            ; schnell, es gibt bessere Methoden.
                            ;
                            ; Eine Zahl steht in HL, sie wird durch DE geteilt.
                            ; Das Ergebnis steht wieder in HL, der Rest in BC

111C  F5                    push af         ; Akku und Flags auf den Stack retten
111D  DD E5                 push ix         ; ix auch retten
111F  DD 21 00 00           ld ix,0         ; In IX wird die Anzahl der Schleifendurchläufe
                                            ; mitgezählt
1123  7C                    DivLoop: ld a,h ; Oberes Byte des Divisors in A
1124  BA                    cp d            ; mit dem oberen Byte des Dividenten vergleichen
1125  38 0C                 jr c,DivEnde    ; Wenn kleiner muß gar nicht weiter getestet werden
1127  20 04                 jr nz,Div       ; HL ist größer als DE, also weitermachen
1129  7D                    ld a,l          ; Unteres Byte in A
112A  BB                    cp e            ; mit dem unteren Byte des Dividenten vergleichen
112B  38 06                 jr c,DivEnde    ; HL ist kleiner als DE, also fertig.
112D  DD 23                 Div:  inc ix    ; Mitzählen
112F  ED 52                 sbc hl,de       ; einmal abziehen
1131  18 F0                 jr DivLoop      ; und wiederholen
                            DivEnde:
1133  E5                    push hl         ; Rest nach bc bringen
1134  C1                    pop bc
1135  DD E5                 push ix         ; Zahl der Schleifendurchläufe ist das Ergebnis
1137  E1                    pop hl
1138  DD E1                 pop ix          ; Register wieder herstellen
113A  F1                    pop af
113B  C9                    ret             ; und zurück zum Hauptprogramm

                            END
```

Mit dem Einsteigerpaket: **von Michael Giulini, GES GmbH**

Ansteuerung der Centronicschnittstelle

Hauptprogramm

Dieses Assemblerprogramm ermöglicht es, einen bestimmten Speicherbereich der HEXIO in Hexdump auf einen Drucker auszugeben. Dies erfolgt über die Baugruppe CENT2. Der angeschlossene Drucker muß eine Centronicschnittstelle haben.

Der erste Programmteil dient der Tastatureingabe. Hier muß der Anfangs- und Endwert des auszudruckenden Speicherbereichs eingegeben werden. Dies wird mit den zwei Betriebssystemroutinen GETVON und GETBIS bewerkstelligt. Zuerst wird die Routine GETVON aufgerufen. Dieses Unterprogramm, das sich ab der Adresse 0266 im HexMon befindet, gibt auf die linken drei Stellen der 7-Segment-Anzeige das Wort „von" aus. Auf den rechten vier Anzeigestellen ist eine Zahl zu sehen, die mit der Anfangsadresse des auszudruckenden Speicherbereiches überschrieben werden kann. Nach Drücken der CR-Taste wird der Wert übernommen und in dem 16-Bit Register HL gespeichert. Durch den Befehl „push hl" wird die Anfangsadresse auf den Stapel gegeben, damit er nicht durch den Endwert überschrieben wird. Die Betriebssystemroutine GETBIS hat die gleiche Funktion, wie die GETVON-Routine, mit der einen Ausnahme, daß statt des Wortes „von" das Wort „bis" angezeigt wird. Der so eingegebene Endwert wird in dem Registerpaar BC gespeichert und, nachdem der Anfangswert wieder vom Stapel geholt wurde, auf

Hauptprogramm cent1.:

- Betriebssystemsroutine GETVON aufrufen (Eingabe der Anfangsadresse)
- Betriebssystemsroutine GETBIS aufrufen (Eingabe der Endadresse)
- wiederhole
  - Spaltenzaehler auf 0ah setzen
  - Zeilenvorschub ausgeben (durch Aufruf UP drucken)
  - erste beiden Ziffern von lfd. Adresse durch UP Umwandeln in ASCII-Code wandeln
  - Ausgeben der beiden ASCII-Codes an den Drucker (durch 2x Aufrufen des UP Drucken)
  - letzte beiden Ziffern von lfd. Adresse durch UP Umwandeln in ASCII-Code wandeln
  - Ausgeben der beiden ASCII-Codes an den Drucker (durch 2x Aufrufen des UP Drucken)
  - drei Leerzeichen drucken
  - wiederhole
    - Leerzeichen drucken
    - Speicherinhalt einlesen
    - Aufruf UP Umwandel
    - Aufruf UP Drucken
    - Aufruf UP Drucken
    - Speicheradresse um eins erhöhen
    - Spaltenzähler erniedrigen
  - bis Spaltenzähler = 0
- bis Endwert erreicht ist
- Return auf Drucker ausgeben
- Programmende (Reset)

dem Stapel zwischengespeichert, da er erst am Ende des Hauptprogrammes gebraucht wird.

Ist die Eingabe abgeschlossen, so wird mit dem Ausdruck des Hexdumps begonnen. Zunächst wird festgelegt, daß jede Zeile 16 (hexadezimal 10) Bytewerte beinhalten soll. Danach wird der Code 0ah für Linefeed dem Unterprogramm DRUCKEN übergeben. Dies bewirkt, daß nach jeder vollen Zeile ein Zeilenvorschub ausgelöst wird. Nun werden die ersten zwei Ziffern der Adresse dem Unterprogramm UMWANDEL übergeben und durch dieses in die entsprechenden ASCII-Codes umgewandelt. Dieser ASCII-Code wird darauf durch die Druck-Routine auf den Drucker ausgegeben. Für jedes Zeichen, das gedruckt werden soll, muß die Routine DRUCKEN einmal aufgerufen werden. In der gleichen Weise wird auch der zweite Teil der Adresse zuerst in ASCII-Code umgewandelt und dann gedruckt. Dazu wird der Wert in der Speicherzelle mit der Adresse HL in das Register B übertragen. Nach dem Umwandeln in ASCII-Code wird dieser auf den Drucker ausgegeben. Darauf wird ein Leerzeichen gesendet und der Zeiger HL auf die Speicherzelle um Eins erhöht. Der Spaltenzähler E wird erniedrigt. Sind noch keine 16 Bytewerte in eine Zeile gedruckt (E()oh), so springt das Programm an den Anfang dieses Programmteiles und druckt den nächsten Wert in die gleiche Zeile. Ist die Zeile voll, so wird der Adreßzeiger mit der Endadresse verglichen, die dafür vom Stapel in das Regi-

ster BC geholt wird. Zum Zweck des Vergleichens wird die Endadresse von der laufenden Adresse subtrahiert. Ist das ergebnis größer oder gleich Null, d. h. die Endadresse ist erreicht oder schon überschritten, so springt das Programm zum Label „Ende". Dort wird nach dem Senden eines Returns, das notwendig ist, damit die letzte Zeile auch gedruckt wird, das Programm mit einem Reset beendet. Ist der Endwert noch nicht erreicht, wird das Register BC, das vorher subtrahiert wurde, wieder zur Adresse HL addiert. Nach dem der Endwert wieder auf den Stapel geschrieben wurde, wird eine neue Zeile gedruckt.

Unterprogramm DRUCKEN

Dieses Unterprogramm ist für die korrekte Übertragung der Daten und der Steuersignale zwischen HEXIO und Drucker zuständig. Nachdem der erste ASCII-Wert über den Kanal 48h auf die Schnittstelle ausgegeben wurde, muß das Strobe-Signal kurz auf Null gesetzt werden. Es wird dann gleich wieder auf High gesetzt. Dieses Signal ist notwendig, damit der Drucker das Datenwort übernimmt.

UP Drucken:

ersten ASCII-Code auf Kanal 48h ausgeben
Strobe-Signal auf Low setzen
Strobe-Signal auf High setzen
wiederhole
Busy-Signal einlesen
solange Busy = 1
RÜCKSPRUNG

Darauf wird das Busy-Signal (Bit 7 des Statusregisters) eingelesen. Ist dieses Signal auf High gesetzt, heißt das, daß der Drucker nicht bereit ist weitere Zeichen zu drucken. In diesem Fall fragt das Programm das Busy-Signal in einer Schleife so lange ab, bis der Drucker wieder empfangsbereit ist. Danach wird zum Hauptprogramm zurückgesprungen und das nächste Zeichen ausgegeben.

Das Unterprogramm Umwandeln

Dieses Unterprogramm dient der Umwandlung der hexadezimalen Ziffern 0 – 9 und A – F in ihre ASCII-Codes. Zuerst wird durch den vierfachen Befehl „srl a" das höherwertige Halbbyte bitweise auf die Stelle des niederwertigen Halbbytes geschoben. Dann wird der Wert 0ah (dezimal 10) von diesem Byte abgezogen. Ist das Resultat positiv, so bedeutet dies, daß es sich um einen Buchstaben von A bis F handelt. In diesem Fall wird im Unterprogramm BUCHST der Wert 41h dazuaddiert, um den ASCII-Code zu erhalten. Dieser Wert setzt sich aus 37h, der Differenz zwischen Hex-Wert und ASCII-Code und 0ah, dem Wert, der für die vorherige Prüfung subtrahiert wurde, zusammen.

Ist das Resultat der Prüfung positiv, handelt es sich um eine Ziffer von 0 bis 9. Deswegen wird im Unterprogramm ZAHLEN der Wert 3ah addiert, der sich ebenfalls aus der Differenz zwischen Hex- und ASCII-Wert und dem Wert 0ah zusammensetzt.

Im folgenden Programmteil wird die niederwertige Ziffer des Bytes in seinen ASCII-Code umgewandelt. Dazu wird durch die Maskierung des Bytes mit 0fh das höherwertige Halbbyte abgeschnitten. Vom verbleibenden Wert wird zur Prüfung wieder 0ah subtrahiert und abhängig vom Ergebnis, wie im ersten Teil, der entsprechende Wert addiert.

Nach dem Rücksprung ins Hauptprogramm kann der eben errechnete ASCII-Code ausgedruckt werden.

UP Umwandeln:

Byte um vier Stellen nach rechts verschieben	
Ist Wert >09h ?	
ja	nein
Buchstaben: 41h dazuaddieren	Zahlen: 3ah dazuaddieren
ASCII-Wert in Reg. c ablegen	
höheres Halbbyte auf Null setzen	
Ist Wert >09h ?	
ja	nein
Buchstaben: 41h dazuaddieren	Zahlen: 3ah dazuaddieren
zweiten ASCII-Wert in Reg. D ablegen	

Schematische Programmstruktur:

Eingabe der Anfangs- und der Endadresse über die Tastatur
wiederhole
Zeilenvorschub und Adresse drucken
wiederhole
Hex-Werte in Zeile drucken
bis Zeile voll
bis Endadresse erreicht

```
MACRO-80 3.43   27-Jul-81       PAGE    1


                        .z80
                        ;*****************************************
                        ;* Ansteuerung der Centronics von der HexI/O *
                        ;* 24. November 1987  Michael Giulini        *
                        ;*  centi v 1.0                              *
                        ;*****************************************


                        org 8100h                   ; Programmspeicher ab 8100h
004B                    cent equ 4Bh                ; Datenregister der CENT2
0049                    signal equ  49h             ; Steuerregister der CENT2
0266                    getvon equ 0266h            ; Adresse der Betriebssystemroutine GETVON
0272                    getbis equ 0272h            ; Adresse der Betriebssystemroutine GETBIS


8100'  CD 0266          start:          call getvon     ; Betriebssystemroutine fuer Anfangswert
8103'  E5                               push hl         ; hl retten
8104'  CD 0272                          call getbis     ; Betriebssystemroutine fuer Endwert
8107'  44                               ld b,h          ; HL in BC ablegen
8108'  4D                               ld c,l
8109'  E1                               pop hl          ; Anfangswert herholen
810A'  C5                               push bc         ; Endwert retten

810B'  1E 10            naechstzeile:   ld e,10h        ; 16 Werte in eine Zeile
810D'  0E 0A                            ld c,0ah        ; Der Code fuer Linefeed (0ah) in Reg C ablegen
                                                        ; Reg. C: Uebergaberegister fuer Druckroutine
810F'  CD 8160'                         call drucken    ; Linefeed auf Drucker ausgeben
8112'  44                               ld b,h          ; Reg. B: Uebergaberegister fuer Umwandlungsroutine
8113'  CD 8173'                         call umwandel   ; Up Umwandlung Hex -> ASCII
8116'  CD 8160'                         call drucken    ; Aufruf druckroutine
8119'  4A                               ld c,d          ; zweiten ASCII-Wert an Druckroutine uebergeben
811A'  CD 8160'                         call drucken
811D'  45                               ld b,l
811E'  CD 8173'                         call umwandel
8121'  CD 8160'                         call drucken
8124'  4A                               ld c,d
8125'  CD 8160'                         call drucken
8128'  0E 20                            ld c,20h        ; 3 Leerzeichen ausdrucken (ASCII-Code 20h)
812A'  CD 8160'                         call drucken
812D'  CD 8160'                         call drucken
8130'  CD 8160'                         call drucken

8133'  0E 20            naechstwert:    ld c,20h        ; Leerzeichen
8135'  CD 8160'                         call drucken
8138'  46                               ld b,(hl)       ; Speicherinhalt einlesen
8139'  CD 8173'                         call umwandel
813C'  CD 8160'                         call drucken
813F'  4A                               ld c,d
8140'  CD 8160'                         call drucken
8143'  0E 20                            ld c,20h        ; Leerzeichen
8145'  CD 8160'                         call drucken
8148'  23                               inc hl          ; Speicheradresse um eins erhoehen
8149'  1D                               dec e           ; Spaltenzaehler erniedrigen
814A'  C2 8133'                         jp nz, naechstwert  ; wenn Zeile nicht voll naechster Wert
MACRO-80 3.43   27-Jul-81       PAGE    1-1


814D'  C1                               pop bc          ; Endwert wieder herholen
814E'  ED 42                            sbc hl,bc       ; hl=hl-bc
8150'  F2 8159'                         jp p, ende      ; wenn Resultat groesser Null, dann Ende
8153'  ED 4A                            adc hl,bc       ; hl-Wert wieder herstellen
8155'  C5                               push bc         ; Endwert retten
8156'  C3 810B'                         jp naechstzeile ; weiter mit neuer Zeile

8159'  0E 0D            ende:           ld c,0dh        ; Return (Code 0dh) an Drucker senden
815B'  CD 8160'                         call drucken
815E'  C7                               rst 00h         ; Hallo-Meldung (Reset)
815F'  00                               nop


8160'  79               drucken:        ld a,c          ; erster ASCII-Wert -> Akku
8161'  D3 4B                            out (cent),a    ; auf Centronics-Schnittst. ausgeben
8163'  3E 00                            ld a,00h        ; Strobe-Signal auf Low setzen
8165'  D3 49                            out (signal),a
8167'  3E 01                            ld a,01h        ; Strobe-Signal auf High setzen
8169'  D3 49                            out (signal),a
```

```
816B'  DB 49          busy:      in a,(signal)  ; Busy-Signal einlesen
816D'  FE FF                     cp 0ffh
816F'  CA 816B'                  jp z, busy     ; wenn busy=1 dann warte
8172'  C9                        ret            ; Ruecksprung

8173'  78             uawandel:  ld a,b         ; Hex-Wert in Akku laden
8174'  CB 3F                     srl a          ; Byte viermal bitweise nach links
8176'  CB 3F                     srl a          ; verschieben
8178'  CB 3F                     srl a          ; (hoeheres Halbbyte maskieren)
817A'  CB 3F                     srl a
817C'  D6 0A                     sub 0ah        ; 0ah von Wert subtrahieren
817E'  F4 8195'                  call p,buchst  ; wenn Ergebnis positiv, Aufruf Up Buchstaben
8181'  FC 8192'                  call m, zahlen ; wenn Ergebnis negativ, Aufruf Up Zahlen

8184'  4F                        ld c,a         ; gewandelten Wert in Reg. C speichern

8185'  78                        ld a,b
8186'  E6 0F                     and 00001111b  ; niedrigeres Halbbyte maskieren
8188'  D6 0A                     sub 0ah
818A'  F4 8195'                  call p,buchst
818D'  FC 8192'                  call m, zahlen
8190'  57                        ld d,a         ; gewandelter Wert in Reg. D speichern
8191'  C9                        ret            ; Ruecksprung ins Hauptprogramm

8192'  C6 3A          zahlen:    add a, 3ah     ; ASCII=Hexdump+30h
8194'  C9                        ret

8195'  C6 41          buchst:    add a, 41h     ; ASCII=Hexdump+37h
8197'  C9                        ret

                                 end

      MACRO-80 3.43  27-Jul-81      PAGE   S

Macros:

Symbols:
8195'  BUCHST      816B'  BUSY       0048  CENT
8160'  DRUCKEN     8159'  ENDE       0272  GETBIS
0266   GETVON      8133'  NAECHSTWERT 810B' NAECHSTZEILE
0049   SIGNAL      8100'  START      8173' UMWANDEL
8192'  ZAHLEN

No Fatal error(s)
```

Probeausdruck des Speicherbereiches 8100h-819f durch CENT2

```
8100  CD 66 02 E5 CD 72 02 44 4D E1 C5 1E 10 0E 0A CD
8110  60 81 44 CD 73 81 CD 60 81 4A CD 60 81 45 CD 73
8120  81 CD 60 81 4A CD 60 81 0E 20 CD 60 81 CD 60 81
8130  CD 60 81 0E 20 CD 60 81 46 CD 73 81 CD 60 81 4A
8140  CD 60 81 0E 20 CD 60 81 23 1D C2 33 81 C1 ED 42
8150  F2 59 81 ED 4A C5 C3 0B 81 0E 0D CD 60 81 C7 00
8160  79 D3 48 3E 00 D3 49 3E 01 D3 49 DB 49 FE FF CA
8170  6B 81 C9 78 CB 3F CB 3F CB 3F CB 3F D6 0A F4 95
8180  81 FC 92 81 4F 78 E6 0F D6 0A F4 95 81 FC 92 81
8190  57 C9 C6 3A C9 C6 41 C9 2A 03 E5 F5 2B 75 A2 D0
```

Anmerkung der Redaktion:

Zum Selbstprogrammieren ist das Listing des Monitors im Einsteigerpaket unbedingt erforderich. Es ist im Buch „Mit HEXMON Programme entwickeln" von Rolf-Dieter Klein (Best.-Nr. 10286) enthalten. Dieses Buch gibt's zum Sonderpreis!

Z 80-Vollausbau bis ZEAT

„Checksummen"

von Leo Hepperger

Als NDR-Computer Fan freue ich mich auf jede neue *LOOP*. Leider sind Programme speziell für Z80-CPU dünn gesäht. Daher möchte ich heute auch einen Beitrag leisten.

Beim Lesen des Buches „Rechner modular (von R.-D. Klein), tauchte der Wunsch nach einem Checksummen-Programm auf.

Das folgende Programm gibt Hexdump mit Checksummen am Bildschirm aus, und zusätzlich am Drucker, wenn mit J oder j gewünscht. Programmstart mit CS. Nach Eingabe von Anfangs- und Endadresse werden je 10 Zeilen ausgegeben. Mit W oder w geht es weiter. Mit N oder n startet das Programm nochmal. Mit M oder restlichen Tasten zurück ins GRUND-Menue.

Mit Z80-CPU / EGRUND 2.0 / EZASS 2.1 erstellt.

(c) 11/87 Leo Hepperger, Am Schlag 7, 8031 Eichenau

```
Start-Adresse:CS

End-Adresse:MEMORY4  + 1
F000 CD3000   CS: CALL CLR        ;m.CS starten,CLR v.GRUND loescht Schirm
F003 CD9FF0    CALL TEXT1         ;1.Bildseite
F006 CDB9F0    CALL TEXT2         ;    "
F009 CD2FF1    CALL EINGABE1      ;    "
F00C CD42F1    CALL INVERT        ;    "
F00F CD4AF1    CALL TEXT3         ;    "
F012 CD7AF1    CALL EINGABE2      ;    "
F015 CD42F1    CALL INVERT        ;    "
F018 CD90F1    CALL TEXT4         ;    "
F01B CDBCF1    CALL EINGABE3      ;    "
F01E CD3000    CALL CLR           ;    "
F021 CDD2F1    CALL TEXT5         ;2.Bildseite
F024 DD21CDF1  LD IX,ENDADR       ;1.Byte
F028 CD05F2    CALL HEXASC
F02B 2214F3    LD (MEMORY4),HL    ;v.HEXASC gewandelte ENDADR nach (MEMORY4)
F02E DD218BF1  LD IX,ANFADR       ;1.Byte
F032 CD05F2    CALL HEXASC
F035 010000   SEITE: LD BC,0000H
F038 ED435280  LD (8052H),BC      ;X-Koord.Speicher f.Bildausgabe (v.GRUND)
F03C 11F000    LD DE,00F0H
F03F ED535480  LD (8054H),DE      ;Y-Koord.Speicher f.Bildausgabe (v.GRUND)
F043 0E0A      LD C,0AH           ;10 Zeilen Hexdump Ausgabe
F045 CD3FF2   ZEILE: CALL AUSWORT
F048 CDCDF2    CALL LEER
F04B 110000    LD DE,0000H
F04E 0610      LD B,10H           ;16 Speicherinhalte werden ausgegeben
F050 7E       ZLOOP: LD A,(HL)    ;gewandelte ANFADR (v.HEXASC) nach A
F051 CD56F2    CALL AUSBYTE
F054 CDCDF2    CALL LEER
F057 23        INC HL
F058 C5        PUSH BC
F059 E5        PUSH HL            ;(MEMORY4) ist v.HEXASC gewandelte ENDADR
F05A ED4B14F3  LD BC,(MEMORY4)
F05E 03        INC BC
F05F A7        AND A              ;loescht Uebertrag
F060 ED42      SBC HL,BC          ;Anfangsadr. minus Endadr.
F062 E1        POP HL
F063 C1        POP BC
F064 281F      JR Z,0F085H        ;bei 0 Ausgabeende
F066 83        ADD A,E
F067 5F        LD E,A
F068 3001      JR NC,ZJUMP
F06A 14        INC D
F06B 05       ZJUMP: DEC B
F06C 20E2      JR NZ,ZLOOP        ;naechste Ausgabe bis B = 0
F06E EB        EX DE,HL           ;fuer AUSWORT nach HL
F06F CDCDF2    CALL LEER
F072 CDD5F2    CALL PLUSIST
F075 CD3FF2    CALL AUSWORT
F078 EB        EX DE,HL
F079 CDEAF2    CALL CRLF
F07C CDEAF2    CALL CRLF
F07F 0D        DEC C              ;C = Zaehler fuer 10 Zeilen
F080 3E00      LD A,00H
F082 B9        CP C
F083 20C0      JR NZ,ZEILE        ;neue Zeile bis C = 0
F085 CDF7F2    CALL EINTAST
```

```
F088 FE57      CP 57H              ;"W"
F08A CC06F3    CALL Z,WEITER
F08D FE77      CP 77H              ;"w"
F08F CC06F3    CALL Z,WEITER
F092 FE4E      CP 4EH              ;"N"
F094 CA00F0    JP Z,CS             ;CS = neuer Start
F097 FE6E      CP 6EH              ;"n"
F099 CA00F0    JP Z,CS
F09C C30000    JP 0000H            ;m.M oder anderen ins GRUND-Menue
F09F 21A6F0   TEXT1: LD HL,INHALT1
F0A2 CD1E00    CALL WRITE          ;von GRUND
F0A5 C9        RET

F0A6 70       INHALT1: LD (HL),B   ;keine Befehle Assemb.interprediert falsch
F0A7 00        NOP
F0A8 E0        RET PO              ;Y-Koord.
F0A9 00        NOP
F0AA 33        INC SP              ;Schriftgr.
F0AB 04        INC B               ;Schriftrichtung
F0AC 43        LD B,E              ;"CHECK-SUMMEN"
F0AD 48        LD C,B
F0AE 45        LD B,L
F0AF 43        LD B,E
F0B0 4B        LD C,E
F0B1 2D        DEC L
F0B2 53        LD D,E
F0B3 55        LD D,L
F0B4 4D        LD C,L
F0B5 4D        LD C,L
F0B6 45        LD B,L
F0B7 4E        LD C,(HL)           ;Ende INHALT1
F0B8 00        NOP
F0B9 DD21C9F0 TEXT2: LD IX,INHALT2
F0BD 212500    LD HL,0025H         ;X-Koord.
F0C0 11C500    LD DE,00C5H         ;Y-Koord.
F0C3 3E22      LD A,22H            ;Schriftgr.
F0C5 CD1306    CALL TEXTPRINT      ;von GRUND intern,ermoeglicht mehrere Zeilen
F0C8 C9        RET

F0C9 47       INHALT2: LD B,A      ;"Gibt je 16 Byte mit Checksumme aus
F0CA 69        LD L,C              ;wird Drucker-Ausgabe gewuenscht > J
F0CB 62        LD H,D              ;sonst beliebige Taste druecken"
F0CC 74        LD (HL),H
F0CD 206A      JR NZ,0F139H
F0CF 65        LD H,L
F0D0 2031      JR NZ,0F103H        ;keine Befehle Assemb.interprediert falsch
F0D2 3620      LD (HL),20H         ;nur Text
F0D4 42        LD B,D
F0D5 79        LD A,C
F0D6 74        LD (HL),H
F0D7 65        LD H,L
F0D8 206D      JR NZ,0F147H
F0DA 69        LD L,C
F0DB 74        LD (HL),H
F0DC 2043      JR NZ,0F121H
F0DE 68        LD L,B
F0DF 65        LD H,L
F0E0 63        LD H,E
F0E1 6B        LD L,E
F0E2 73        LD (HL),E
F0E3 75        LD (HL),L
F0E4 6D        LD L,L
F0E5 6D        LD L,L
F0E6 65        LD H,L
F0E7 2061      JR NZ,TEXT3
F0E9 75        LD (HL),L
F0EA 73        LD (HL),E
F0EB 0A        LD A,(BC)
F0EC 0A        LD A,(BC)
F0ED 77        LD (HL),A
F0EE 69        LD L,C
F0EF 72        LD (HL),D
F0F0 64        LD H,H
F0F1 2044      JR NZ,0F137H
F0F3 72        LD (HL),D
F0F4 75        LD (HL),L
F0F5 63        LD H,E
F0F6 6B        LD L,E
F0F7 65        LD H,L
F0F8 72        LD (HL),D
F0F9 2D        DEC L
F0FA 41        LD B,C
F0FB 75        LD (HL),L
F0FC 73        LD (HL),E
F0FD 67        LD H,A
F0FE 61        LD H,C
F0FF 62        LD H,D
F100 65        LD H,L
F101 2067      JR NZ,0F16AH
F103 65        LD H,L
F104 77        LD (HL),A
F105 75        LD (HL),L
F106 65        LD H,L
F107 6E        LD L,(HL)
F108 73        LD (HL),E
F109 63        LD H,E
F10A 6B        LD L,B
F10B 74        LD (HL),H
F10C 203E      JR NZ,0F14CH
F10E 204A      JR NZ,0F15AH
F110 0A        LD A,(BC)
F111 73        LD (HL),E
F112 6F        LD L,A
F113 6E        LD L,(HL)           ;nur Text
F114 73        LD (HL),E
F115 74        LD (HL),H
F116 2062      JR NZ,EINGABE2
F118 65        LD H,L
F119 6C        LD L,H
F11A 69        LD L,C
F11B 65        LD H,L
F11C 62        LD H,D
F11D 67        LD H,A
F11E 65        LD H,L
F11F 2054      JR NZ,0F175H
F121 61        LD H,C
F122 73        LD (HL),E
F123 74        LD (HL),H
F124 65        LD H,L
F125 2064      JR NZ,ANFADR
F127 72        LD (HL),D
F128 75        LD (HL),L
F129 65        LD H,L
F12A 63        LD H,E
F12B 6B        LD L,E
F12C 65        LD H,L
F12D 6E        LD L,(HL)
F12E 00        NOP                 ;Ende INHALT2
F12F 2138F1   EINGABE1: LD HL,PARAM1 ;Drucker ja oder nein
F132 0E01      LD C,01H            ;mit Rahmen
F134 CD2100    CALL READ           ;von GRUND
F137 C9        RET

F138 DF       PARAM1: RST 3        ;X-Koord.
F139 00        NOP
F13A 74        LD (HL),H           ;Y-Koord.
F13B 00        NOP
F13C 220001    LD (0100H),HL       ;22=Schriftgr., 00=immer, 01=max.Zahl
F13F 00        NOP                 ;tatsaechliche Anzahl der Zeichen
F140 00       JANEIN: NOP          ;bei 4A(J) o. 6A(j) Drucker aktiv
F141 00        NOP                 ;Ende PARAM1
F142 3E00     INVERT: LD A,00H     ;INVERTer fuer Bildausgabe
F144 CD3B00    CALL WAIT           ;von GRUND
F147 D370      OUT (70H),A         ;Portadr.vom Bildschirm
F149 C9        RET

F14A 2151F1   TEXT3: LD HL,INHALT3
F14D CD1E00    CALL WRITE          ;von GRUND
F150 C9        RET

F151 25       INHALT3: DEC H       ;X-Koord.
F152 00        NOP
F153 58        LD E,B              ;Y-Koord.
F154 00        NOP
F155 220042    LD (4200H),HL       ;Schriftgr.- u.Richtung  u. "B"
F158 69        LD L,C              ;"(B)itte die Anfangs-Adresse eingeben"
F159 74        LD (HL),H           ;keine Befehle , nur Text
F15A 74        LD (HL),H
F15B 65        LD H,L
F15C 2064      JR NZ,0F1C2H
F15E 69        LD L,C
F15F 65        LD H,L              ;keine Befehle , nur Text
F160 2041      JR NZ,0F1A3H
F162 6E        LD L,(HL)
F163 66        LD H,(HL)
F164 61        LD H,C
F165 6E        LD L,(HL)
F166 67        LD H,A
```

```
F167 73        LD (HL),E
F168 2D        DEC L
F169 41        LD B,C
F16A 64        LD H,H
F16B 72        LD (HL),D
F16C 65        LD H,L
F16D 73        LD (HL),E
F16E 73        LD (HL),E
F16F 65        LD H,L
F170 2065      JR NZ,0F1D7H
F172 69        LD L,C
F173 6E        LD L,(HL)
F174 67        LD H,A
F175 65        LD H,L
F176 62        LD H,D
F177 65        LD H,L
F178 6E        LD L,(HL)
F179 00        NOP                 ;Ende INHALT3
F17A 2183F1    EINGABE2: LD HL,PARAM2;fuer Anfangsadr.
F17D 0E01      LD C,01H            ;mit Rahmen
F17F CD2100    CALL READ           ;von GRUND
F182 C9        RET

F183 D0        PARAM2: RET NC      ;X-Koord.
F184 00        NOP
F185 45        LD B,L              ;Y-Koord.
F186 00        NOP
F187 220004    LD (0400H),HL       ;22=Schriftgr., 00=immer , 04=max.Zahl
F18A 04        INC B               ;tatsaechliche Anzahl der Zeichen
F18B 00        ANFADR: NOP         ;Anfangsadr. 1.Byte
F18C 00        NOP                 ;    "     2. "
F18D 00        NOP                 ;    "     3. "
F18E 00        NOP                 ;    "     4. "
F18F 00        NOP                 ;Ende PARAM2
F190 2197F1    TEXT4: LD HL,INHALT4
F193 CD1E00    CALL WRITE          ;von GRUND
F196 C9        RET

F197 35        INHALT4: DEC (HL)   ;X-Koord.
F198 00        NOP
F199 25        DEC H               ;Y-Koord.
F19A 00        NOP
F19B 220042    LD (4200H),HL       ;Schriftgr.- u.Richtung u. "B"
F19E 69        LD L,C              ;"(B)itte die End-Adresse eingeben"
F19F 74        LD (HL),H           ;keine Befehle , nur Text
F1A0 74        LD (HL),H
F1A1 65        LD H,L
F1A2 2064      JR NZ,0F208H
F1A4 69        LD L,C
F1A5 65        LD H,L
F1A6 2045      JR NZ,0F1EDH        ;keine Befehle , nur Text
F1A8 6E        LD L,(HL)
F1A9 64        LD H,H
F1AA 2D        DEC L
F1AB 41        LD B,C
F1AC 64        LD H,H
F1AD 72        LD (HL),D
F1AE 65        LD H,L
F1AF 73        LD (HL),E
F1B0 73        LD (HL),E
F1B1 65        LD H,L
F1B2 2065      JR NZ,0F219H
F1B4 69        LD L,C
F1B5 6E        LD L,(HL)
F1B6 67        LD H,A
F1B7 65        LD H,L
F1B8 62        LD H,D
F1B9 65        LD H,L
F1BA 6E        LD L,(HL)
F1BB 00        NOP                 ;Ende INHALT4
F1BC 21C5F1    EINGABE3: LD HL,PARAM3;fuer Endadr.
F1BF 0E01      LD C,01H            ;mit Rahmen
F1C1 CD2100    CALL READ           ;von GRUND
F1C4 C9        RET

F1C5 D0        PARAM3: RET NC      ;X-Koord.
F1C6 00        NOP
F1C7 1000      DJNZ 0F1C9H         ;Y-Koord.
F1C9 220004    LD (0400H),HL       ;22=Schriftgr., 00=immer , 04=max. Zahl
F1CC 04        INC B               ;tatsaechliche Anzahl der Zeichen
F1CD 00        ENDADR: NOP         ;Endadr. 1.Byte
F1CE 00        NOP                 ;    "     2. "
F1CF 00        NOP                 ;    "     3. "
F1D0 00        NOP                 ;    "     4. "
F1D1 00        NOP                 ;Ende PARAM3
F1D2 21D9F1    TEXT5: LD HL,INHALT5
F1D5 CD1E00    CALL WRITE          ;von GRUND
F1D8 C9        RET

F1D9 1000      INHALT5: DJNZ 0F1DBH ;X-Koord.
F1DB 1000      DJNZ 0F1DDH         ;Y-Koord.
F1DD 22004D    LD (4D00H),HL       ;Schriftgr.- u.Richtung u. "M"
F1E0 203D      JR NZ,0F21FH        ;"(M) = Menue  N = nochmal  W = weiter"
F1E2 204D      JR NZ,0F231H        ;keine Befehle , nur Text
F1E4 65        LD H,L
F1E5 6E        LD L,(HL)
F1E6 75        LD (HL),L
F1E7 65        LD H,L
F1E8 2020      JR NZ,0F20AH
F1EA 2020      JR NZ,HLOOP
F1EC 4E        LD C,(HL)
F1ED 203D      JR NZ,0F22CH
F1EF 206E      JR NZ,0F25FH
F1F1 6F        LD L,A
F1F2 63        LD H,E
F1F3 68        LD L,B
F1F4 6D        LD L,L
F1F5 61        LD H,C
F1F6 6C        LD L,H              ;keine Befehle , nur Text
F1F7 2020      JR NZ,0F219H
F1F9 2057      JR NZ,0F252H
F1FB 203D      JR NZ,0F23AH
F1FD 2077      JR NZ,0F276H
F1FF 65        LD H,L
F200 69        LD L,C
F201 74        LD (HL),H
F202 65        LD H,L
F203 72        LD (HL),D
F204 00        NOP                 ;Ende INHALT5
F205 0604      HEXASC: LD B,04H    ;macht aus 31-1,aus 41-A,aus 61-a  usw.
F207 0E03      LD C,03H            ;fuer D oder E laden
F209 210000    LD HL,0000H         ; HL loeschen
F20C 110000    HLOOP: LD DE,0000H  ; DE loeschen
F20F DD7E00    LD A,(IX+00H)       ;Byte nach A
F212 CB77      BIT 6,A             ;Test Bit 6, Ziffer o.  Buchstabe ?
F214 2004      JR NZ,HJUMP1
F216 D630      SUB 30H             ;bei Ziffer
F218 180A      JR HJUMP3
F21A CB6F      HJUMP1: BIT 5,A     ;Test Bit 5, Gross.-o. Kleinbuchstabe ?
F21C 2004      JR NZ,HJUMP2
F21E D637      SUB 37H             ;bei Grossbuchstabe
F220 1802      JR HJUMP3
F222 D657      HJUMP2: SUB 57H     ;bei Kleinbuchstabe
F224 CB40      HJUMP3: BIT 0,B     ;B hat 2 Aufg. schieb. o. spring.u.Zaehler
F226 2008      JR NZ,HJUMP4        ;bei 1 springen, bei 0 schieben
F228 CB27      SLA A               ;links schieben
F22A CB27      SLA A               ;    "
F22C CB27      SLA A               ;    "
F22E CB27      SLA A               ;    "
F230 CB49      HJUMP4: BIT 1,C     ;bei 1 nach D, bei 0 nach E
F232 2808      JR Z,HJUMP6
F234 57        LD D,A
F235 19        HJUMP5: ADD HL,DE   ;Ergebn.nach HL
F236 0D        DEC C
F237 DD23      INC IX              ;fuer weitere Byte
F239 10D1      DJNZ HLOOP          ;DEC B u.bei B=0 Ende, (nach 4 Byte)
F23B C9        RET

F23C 5F        HJUMP6: LD E,A
F23D 18F6      JR HJUMP5           ;Ende HEXASC
F23F CDCDF2    AUSWORT: CALL LEER  ;gibt Adr.u.Checksumme aus
F242 CDCDF2    CALL LEER
F245 F5        PUSH AF
F246 7C        LD A,H
F247 CD56F2    CALL AUSBYTE
F24A 7D        LD A,L
F24B CD56F2    CALL AUSBYTE
F24E CDCDF2    CALL LEER
F251 CDCDF2    CALL LEER
F254 F1        POP AF
F255 C9        RET

F256 F5        AUSBYTE: PUSH AF
F257 F5        PUSH AF
F258 1F        RRA
F259 1F        RRA
F25A 1F        RRA
F25B 1F        RRA
```

```
F25C CD65F2    CALL AUSDIGIT
F25F F1        POP AF
F260 CD65F2    CALL AUSDIGIT
F263 F1        POP AF
F264 C9        RET

F265 E60F      AUSDIGIT: AND 0FH
F267 C630      ADD A,30H          ;fuer Ziffer
F269 FE3A      CP 3AH
F26B 3802      JR C,ADJUMP
F26D C607      ADD A,07H          ;fuer Buchstabe
F26F CD73F2    ADJUMP: CALL AUSCHAR
F272 C9        RET

F273 F5        AUSCHAR: PUSH AF
F274 E5        PUSH HL
F275 D5        PUSH DE
F276 C5        PUSH BC
F277 3211F3    LD (MEMORY1),A     ;Zeichen in Memory1
F27A 3E20      LD A,20H
F27C 3212F3    LD (MEMORY2),A     ;Space in Memory2
F27F 3E00      LD A,00H
F281 3213F3    LD (MEMORY3),A     ;Ende Text in Memory3
F284 DD2111F3  LD IX,MEMORY1
F288 2A5280    LD HL,(8052H)      ;X-Koord.
F28B ED5B5480  LD DE,(8054H)      ;Y-Koord.
F28F 010100    LD BC,0001H
F292 09        ADD HL,BC          ;dehnt Ausgabe am Bildschirm
F293 3E11      LD A,11H           Schriftgr.
F295 CD3B00    CALL WAIT          ;aus GRUND
F298 CD1306    CALL TEXTPRINT     ;aus GRUND intern
F29B 3A40F1    LD A,(JANEIN)      ;fuer Drucker ja oder nein
F29E FE4A      CP 4AH             ;"J"
F2A0 CCADF2    CALL Z,DRUCKER
F2A3 FE6A      CP 6AH             ;"j"
F2A5 CCADF2    CALL Z,DRUCKER
F2A8 C1        POP BC
F2A9 D1        POP DE
F2AA E1        POP HL
F2AB F1        POP AF
F2AC C9        RET

F2AD C5        DRUCKER: PUSH BC   ;entspricht CENRONICS aus LOOP 5
F2AE D5        PUSH DE
F2AF E5        PUSH HL
F2B0 0E49      LD C,49H           ;Drucker Portadr.
F2B2 ED40      DJUMP: IN B,(C)    ;Busy-Signal nach B
F2B4 CB40      BIT 0,B
F2B6 20FA      JR NZ,DJUMP
F2B8 3A11F3    LD A,(MEMORY1)     ;MEMORY1 enthaelt aussugebendes Zeichen
F2BB D348      OUT (48H),A
F2BD 06FF      LD B,0FFH
F2BF ED41      OUT (C),B
F2C1 0600      LD B,00H
F2C3 ED41      OUT (C),B
F2C5 06FF      LD B,0FFH
F2C7 ED41      OUT (C),B
F2C9 E1        POP HL
F2CA D1        POP DE
F2CB C1        POP BC
F2CC C9        RET

F2CD F5        LEER: PUSH AF      ;LEERraum
F2CE 3E20      LD A,20H           ;"Space"
F2D0 CD73F2    CALL AUSCHAR
F2D3 F1        POP AF
F2D4 C9        RET

F2D5 F5        PLUSIST: PUSH AF
F2D6 3E2B      LD A,2BH           ;"+"
F2D8 CD73F2    CALL AUSCHAR
F2DB 3E3D      LD A,3DH           ;"="
F2DD CD73F2    CALL AUSCHAR
F2E0 3E20      LD A,20H           ;"Space"
F2E2 CD73F2    CALL AUSCHAR
F2E5 CD73F2    CALL AUSCHAR
F2E8 F1        POP AF
F2E9 C9        RET

F2EA F5        CRLF: PUSH AF
F2EB 3E0A      LD A,0AH           ;"LF"
F2ED CD73F2    CALL AUSCHAR
F2F0 3E0D      LD A,0DH           ;"CR"
F2F2 CD73F2    CALL AUSCHAR
F2F5 F1        POP AF
F2F6 C9        RET

F2F7 E5        EINTAST: PUSH HL   ;Eingabe von Tastatur
F2F8 D5        PUSH DE
F2F9 C5        PUSH BC
F2FA CD2400    CALL CI            ;von GRUND
F2FD CD3B00    CALL WAIT          ;von GRUND
F300 D370      OUT (70H),A        ;Portadr. vom Bildschirm
F302 C1        POP BC
F303 D1        POP DE
F304 E1        POP HL
F305 C9        RET

F306 CD3000    WEITER: CALL CLR   ;von GRUND
F309 E5        PUSH HL
F30A CDD2F1    CALL TEXT5
F30D E1        POP HL
F30E C335F0    JP SEITE
F311 00        MEMORY1: NOP       ;Speicher1 fuer AUSCHAR
F312 00        MEMORY2: NOP       ;Speicher2 fuer AUSCHAR
F313 00        MEMORY3: NOP       ;Speicher3 fuer AUSCHAR
F314 00        MEMORY4: NOP       ;Speicher4 fuer gewandelte ENDADR(v.HEXASC)
F315 00                           ;ENDE  H-Byte von MEMORY4
```

Das Programm >>> CHECKSUMMEN <<< hat sich selbst ausgedruckt !
CALL CRLF auf F07C durch 3-NOP ersetzt, fuer kleinen Zeilenabstand

```
F000  CD 30 00 CD 9F F0 CD B9 F0 CD 2F F1 CD 42 F1 CD  +=  0A89
F010  4A F1 CD 7A F1 CD 42 F1 CD 90 F1 CD BC F1 CD 30  +=  0B38
F020  00 CD D2 F1 DD 21 CD F1 CD 05 F2 22 14 F3 DD 21  +=  0937
F030  8B F1 CD 05 F2 01 00 00 ED 43 52 80 11 F0 00 ED  +=  0731
F040  53 54 80 0E 0A CD 3F F2 CD CD F2 11 00 00 06 10  +=  05F0
F050  7E CD 56 F2 CD CD F2 23 C5 E5 ED 4B 14 F3 03 A7  +=  09D5
F060  ED 42 E1 C1 28 1F 83 5F 30 01 14 05 20 E2 EB CD  +=  06FE
F070  CD F2 CD D5 F2 CD 3F F2 EB CD EA F2 00 00 00 0D  +=  09F2
F080  3E 00 B9 20 C0 CD F7 F2 FE 57 CC 06 F3 FE 77 CC  +=  09E8
F090  06 F3 FE 4E CA 00 F0 FE 6E CA 00 F0 C3 00 00 21  +=  0809
F0A0  A6 F0 CD 1E 00 C9 70 00 E0 00 33 04 43 48 45 43  +=  05E4
F0B0  4B 2D 53 55 4D 4D 45 4E 00 DD 21 C9 F0 21 25 00  +=  054A
F0C0  11 C5 00 3E 22 CD 13 06 C9 47 69 62 74 20 6A 65  +=  055A
F0D0  20 31 36 20 42 79 74 65 20 6D 69 74 20 43 68 65  +=  04D5
F0E0  63 6B 73 75 6D 6D 65 20 61 75 73 0A 0A 77 69 72  +=  05C4
F0F0  64 20 44 72 75 63 6B 65 72 2D 41 75 73 67 61 62  +=  05D4
F100  65 20 67 65 77 75 65 6E 73 63 68 74 20 3E 20 4A  +=  058A
F110  0A 73 6F 6E 73 74 20 62 65 6C 69 65 62 67 65 20  +=  05B0
F120  54 61 73 74 65 20 64 72 75 65 63 6B 65 6E 00 21  +=  0593
F130  38 F1 0E 01 CD 21 00 C9 DF 00 74 00 22 00 01 01  +=  0466
F140  4A 00 3E 00 CD 3B 00 D3 70 C9 21 51 F1 CD 1E 00  +=  05EA
F150  C9 25 00 58 00 22 00 42 69 74 74 65 20 64 69 65  +=  04B2
F160  20 41 6E 66 61 6E 67 73 2D 41 64 72 65 73 73 65  +=  05D2
F170  20 65 69 6E 67 65 62 65 6E 00 21 83 F1 0E 01 CD  +=  05CE
F180  21 00 C9 D0 00 45 00 22 00 04 04 46 30 30 30 00  +=  02FF
F190  21 97 F1 CD 1E 00 C9 35 00 25 00 22 00 42 69 74  +=  04F8
F1A0  74 65 20 64 69 65 20 45 6E 64 2D 41 64 72 65 73  +=  057E
F1B0  73 65 20 65 69 6E 67 65 62 65 6E 00 21 C5 F1 0E  +=  061A
F1C0  01 CD 21 00 C9 D0 00 10 00 22 00 04 04 46 33 31  +=  036C
F1D0  35 00 21 D9 F1 CD 1E 00 C9 10 00 10 00 22 00 4D  +=  0463
F1E0  20 3D 20 4D 65 6E 75 65 20 20 20 20 4E 20 3D 20  +=  03C2
F1F0  6E 6F 63 68 6D 61 6C 20 20 20 57 20 3D 20 77 65  +=  04F2
F200  69 74 65 72 00 06 04 0E 03 21 00 00 11 00 00 DD  +=  02DE
F210  7E 00 CB 77 20 04 D6 30 18 0A CB 6F 20 04 D6 37  +=  0577
F220  18 02 D6 57 CB 40 20 08 CB 27 CB 27 CB 27 CB 27  +=  0642
F230  CB 49 28 08 57 19 0D DD 23 10 D1 C9 5F 18 F6 CD  +=  06A5
F240  CD F2 CD CD F2 F5 7C CD 56 F2 7D CD 56 F2 CD CD  +=  0BFD
F250  F2 CD CD F2 F1 C9 F5 F5 1F 1F 1F 1F CD 65 F2 F1  +=  0AB3
F260  CD 65 F2 F1 C9 E6 0F C6 30 FE 3A 38 02 C6 07 CD  +=  08D5
F270  73 F2 C9 F5 E5 D5 C5 32 11 F3 3E 20 32 12 F3 3E  +=  08AB
F280  00 32 13 F3 DD 21 11 F3 2A 52 80 ED 5B 54 80 01  +=  0653
F290  01 00 09 3E 11 CD 3B 00 CD 13 06 3A 40 F1 FE 4A  +=  04FA
F2A0  CC AD F2 FE 6A CC AD F2 C1 D1 E1 F1 C9 C5 D5 E5  +=  0CEA
F2B0  0E 49 ED 40 CB 40 20 FA 3A 11 F3 D3 48 06 FF ED  +=  07F4
F2C0  41 06 00 ED 41 06 FF ED 41 E1 D1 C1 C9 F5 3E 20  +=  0837
F2D0  CD 73 F2 F1 C9 F5 3E 2B CD 73 F2 3E 3D CD 73 F2  +=  0A29
F2E0  3E 20 CD 73 F2 CD 73 F2 F1 C9 F5 3E 0A CD 73 F2  +=  09EB
F2F0  3E 0D CD 73 F2 F1 C9 E5 D5 C5 CD 24 00 CD 3B 00  +=  08AF
F300  D3 70 C1 D1 E1 C9 CD 30 00 E5 CD D2 F1 E1 C3 35  +=  0ACA
F310  F0 20 20 00 15 F3
```

Z80-CP/M2.2

Tips und Tricks in dBASE II . . .

von Peter Schwarck, Dreyener Str. 67, 4905 Spenge, Tel.: (05225) 1769

Sicher hat es schon viele dBASE II Anwender gestört, daß die Summe im Reportgenerator mit ***GESAMMT*** bezeichnet wird, stört doch dieser unschöne Fehler den Eindruck eines sonst sauberen Ausdruckes. Dem kann jedoch mittels des folgenden Patches abgeholfen werden.

Die Änderungen werden in der Datei 'DBASEOVR.COM' mit dem Debugger 'DDT.COM' durchgeführt. Dazu wird am besten 'DDT.COM' auf die dBASE II Arbeitsdiskette kopiert.

Aufruf des DDT.COM mit der zu ändernden Datei:
Eingabe: DDT DBASEOVR.COM

Es erscheint:
DDT VERS 2.2
NEXT PC
A900 0100
—

Der Strich (—) ist das Bereitschaftszeichen von DDT; hier wird ein Befehl des Benutzers erwartet. Entweder blättert man mit dem d-Befehl die gesamte Datei durch, bis im rechten Teil des Bildes der Störenfried gefunden wird oder aber man sucht gleich mit dem s-Befehl die entsprechende Adresse (sofern sie bekannt ist) auf. Hier können wir also auch s1908 eingeben (weil ich die Adresse schon gefunden habe). Befehl und Adresse ohne Leerzeichen oder Bindestrich eingeben.
Eingabe: s1908 (RETURN) drücken
Es erscheint: 1908 6D

An der derzeitigen Cursorposition kann jetzt die Änderung eingegeben werden. Im folgenden gebe ich die alten Inhalte und rechts davon die neuen Inhalte an.

	alt	neu	
1908	6D	74	(RETURN) drücken
1909	74	20	(RETURN) drücken
190A	20	2A	(RETURN) drücken
190B	2A	00	(RETURN) drücken
190C	2A	00	(RETURN) drücken
190D	00	"	(RETURN) drücken

Nach Eingabe von „(RETURN) meldet sich in der folgenden Zeile wieder das DDT Bereitschaftszeichen. DDT kann nun mit 'g0' (g null) verlassen werden. Die Änderungen müssen jetzt nur noch gesichert werden.

Eingabe: g0
Es erscheint das vorgewählte Laufwerksprompt z. B. A).

Gesichert wird mit dem 'save'Befehl des Betriebssystems. Hierzu brauchen wir nun die Größe der Datei 'DBASEOVR.-COM' in KByte, da der 'save'Befehl' immer nur Seiten von 256 Byte ab Adresse 0100h sichert. Ein KByte entspricht 4 Seiten. Die Datei 'DBASEOVR' hat eine Länge von 42 KByte. 4 x 42 = 168 KByte.
Eingabe: save 168 DBASEOVR.COM (RETURN) drücken. – Fertig.

PASCAL und BASIC

Pascalprogramme auf Eproms

von Ralf Pawlowitz
(bei GES, Telefon: 0831/6211)

Die Erlösung

Endlich! Die Zeiten der mühseligen Assemblerprogrammierung sind vorbei. Mußten ehrgeizige Anwender des NDR-Computers bei ihrer Programmierung von epromfähigen Programmen immer auf die Assemblerprogrammierung zurückgreifen, wird ihnen nun besseres gelehrt.
Da sich der NKC hervorragend für Steuerungsaufgaben, Meßwerterfassung etc. eignet und diese Anwendungen meist nur minimale Ausbaustufen erfordern (SBC2 – ROA 256), bietet sich das Unterbringen von Programmen in Eproms, die diese eben beschriebenen Aufgaben erledigen, geradezu an.

Daß eine Hochsprache übersichtlicher und einfacher zu programmieren ist als beispielsweise Assembler, wird den meisten geläufig sein. Turbo Pascal bietet neben direktem Zugriff auf PORT's und Speicherzellen auch komfortable arithmetische Operatoren, boolsche Algebra (Or XOR AND NOT), Links/Rechts schieben von Bytes und viele Schmankerln mehr.

Hard- und Software-Voraussetzungen

Als bestehender Hardwareteil ist auf jeden Fall ein CP/M-Rechner mit Z80 CPU erforderlich, ferner unser allseits bewährter Prommer, um das Programm auf das Eprom zu bringen. Als Programmierwerkzeug sollten wir Turbo Pascal V 3.0 verwenden.

Da unser Programmcode noch geringfügig modifiziert werden muß und später ins Eprom gebrannt wird, bietet sich der Z80 Assembler ZEAT an dieser Stelle an.

Software-Einschränkungen

Trotz der vielen Vorteile muß auch hier mit einigen Tricks gearbeitet werden. Da sich in unserem Minimalsystem außer unserem Programm und ein paar KBytes Ram sonst nichts befindet, muß man bei Aufruf von Standardprozeduren wie WriteLn,-ReadLn, KBD usw. Vorsicht walten lassen. Diese Peripherprozeduren werden bei Turbo Pascal ausschließlich über BDOS Funktionen abgewickelt. BDOS Funktionen erwarten jedoch ein Betriebssystem – in unserem Fall CP/M 2.2 – das hier in der Epromversion nicht existiert.

Selbstverständlich muß auf die Kommunikation mit anderen Baugruppen nicht verzichtet werden, sondern hier werden gestrickte Prozeduren angewandt. Beispielsweise wird beim Ansteuern der GDP64 mit Hilfe des INLINE-Befehls direkt in die Register des Grafikprozessors geschrieben. Trotzdem sollte bei der Programmierung solcher Prozeduren vorsichtig vorgegangen werden, wenigstens so lange die Eigenarten der Epromversionen nicht erforscht sind.

Epromcode – ein Beispiel

Prinzipiell ist es bei jeder Konfiguration möglich, Pascal-Programme in Eproms unterzubringen (mind. 10K Epromplatz). Um jedoch jetzt anhand eines Beispiels die Vorgangsweise zu beschreiben, möchte ich mich auf eine Konfiguration beschränken.

Unser Programm soll auf einer ROA64 ablaufen, der Ramspeicher soll ab der Adresse 8000H verfügbar sein. Die Abbildung zeigt hierzu die Speicheraufteilung.

Als erster Schritt ist das eben erstellte Programm als Com-File auf der Diskette abzuspeichern. Als Startadresse muß hier 2000, als Endadresse 9eff verwendet werden. Anhand dieser Optionen erhalten wir vom Compiler ein speziell für unsere Konfiguration compiliertes Programm, das auf der ROA64 im Eprom-Bereich von 0 – 7fff und im Ram-Bereich von 8000 – 9fff ablaufen kann.

Die Adressen 0 – 1eff des Com-Files enthalten die Pascal RunTime-Libary, die Adressen ab 1fff, das Anwenderpro-

gramm, das variable Größen besitzen kann.

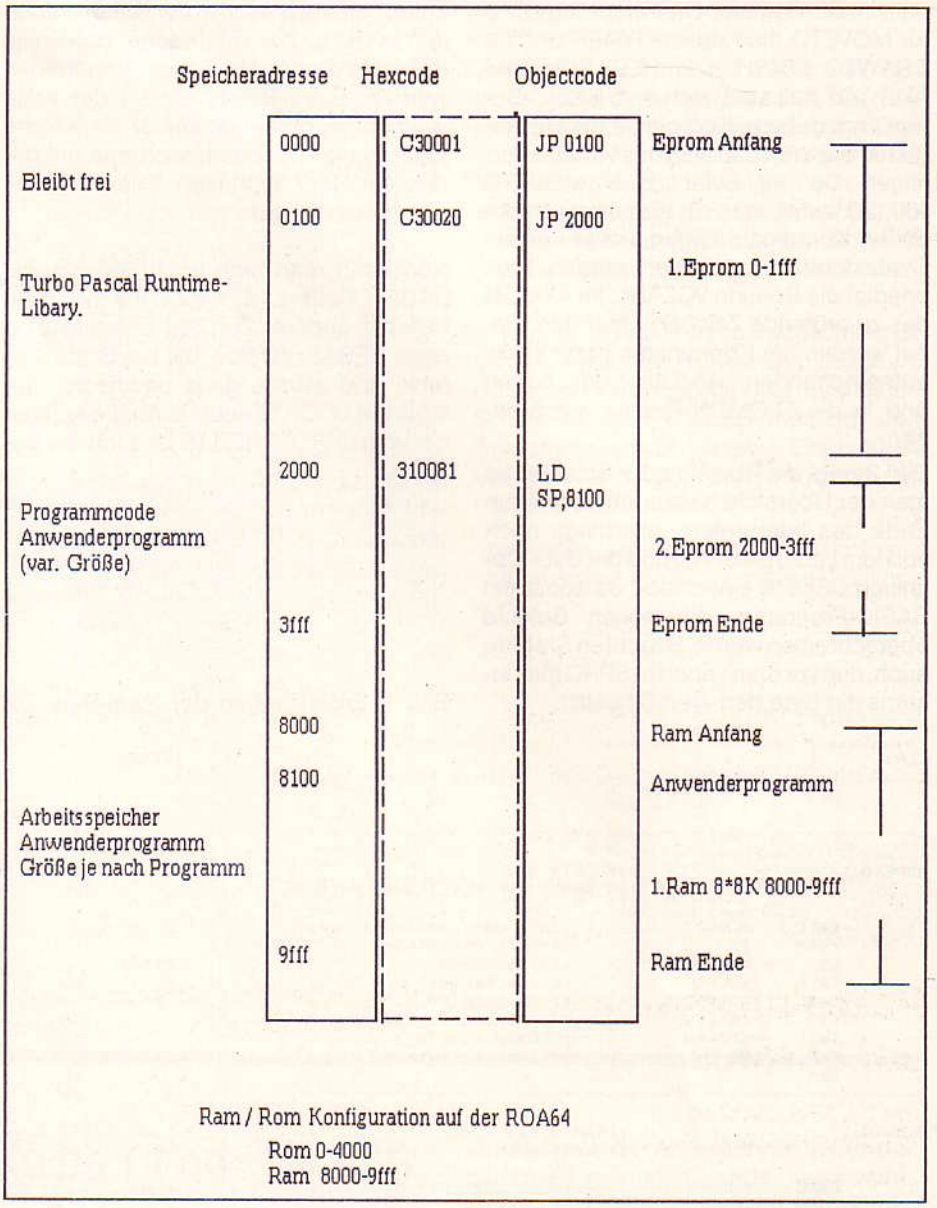
Als nächstes ist es erforderlich, auf der Adresse 0 des Eproms einen Sprung nach 100H einzufügen (JP 0100), da CP/M-Programme immer bei dieser Adresse beginnen.

Die ersten 8 KByte unseres Programmes bestehen aus der RunTime-Libary des Pascal Compilers. Die RunTime-Libary beinhaltet zahlreiche Unterprogramme, die für das richtige Funktionieren unseres Programmes erforderlich sind. Diese Unterprogramme, die in unserem Com-File auf der Diskette stehen, werden nun ab der Epromadresse 0100H – 1fff in das Eprom gebrannt.

Das eigentliche Anwenderprogramm, das direkt nach der Run-Time-Libary (siehe Skizze) beginnt, wird nun in das zweite Eprom, auf die Adresse 0000H gebrannt.

Hier jedoch ist noch eine kleine Änderung erforderlich: Der Pascal Compiler verwendet den Adressbereich unterhalb 0100H als CPU-Stack; weil hier in unserem kleinen Minimalsystem jedoch kein RAM, sondern nur ROM vorhanden ist, müssen wir den Stack auf eine für uns gültige Adresskonfiguration ändern. In unserem Beispiel sitzt der Ramspeicher auf der Adresse 8000H. Dazu muß der Ladebefehl, der auf unserem zweiten Eprom auf der Adresse 0000H stehen muß (LD SP 0100), durch LD SP 8100 ersetzt werden.

Zum Schluß bleibt zu sagen, daß diese eben beschriebenen Änderungen, die sich für die nicht so versierten Anwender unter uns, zunächst etwas kompliziert anhören, sehr wertvoll sein können. Gerade Turbo Pascal ist nicht umsonst eine der weit verbreitesten Programmiersprachen unserer Zeit. Vielleicht hilft dieser Artikel auch, die schon in die Ecke gestellte SBC2 doch noch einmal hervorzuholen und damit etwas „zusammenzuwerkeln". Es soll ja Leute geben, die ihre Kaffeemaschine mit Hilfe ihres Computers steuern.



Literaturverzeichnis: Programmierung des Z80, Rodnay Zaks; Microcomputerzeitschrift mc

Tips und Tricks bei Hebas

Nr. 9

von Dr. Hans Hehl

Welcher BASIC-Befehl soll es sein ... z.B. Grafik-Befehle?

Nachdem für HEBAS der Quellcode zur Verfügung steht, kann grundsätzlich jeder beliebige BASIC-Befehl erstellt werden. So kann man z. B. beliebige Grafik-Befehle unter Ausnützung der schnellen FLOMON-Grafik-Einsprünge beim NDR-Rechner erstellen. Beispielhaft sollen hier die Befehle MOVETO, DRAWTO und CLR (Bildschirm löschen) vorgestellt werden.

Zunächst muß die Tokentabelle erweitert werden. Problemlos geht dies bei der Tabelle TOK2B, also mit den Zwei-Byte-Token 80 XY. Die neuen Befehle werden einfach nach dem BYE-Token und vor dem Label FTXT47 eingefügt. Bild 1 zeigt die Ergänzung mit den neuen Token für MOVETO: 80 96, CLR: 80 97 und DRAWTO 80 98. Es stünde auch das Token 80 90 zur Verfügung, da es nicht verwendet wird (FTAXT47 = Einsprung für Fehlermeldung: nicht vorhanden).

Damit diese Erweiterung berücksichtigt wird, muß die Abfrage in der Routine TOKAW1: CP 16H geändert werden. Da drei neue Befehle hinzukommen, ergibt sich die neue Abfrage zu CP 19H.

Nun müssen die nach dem Befehl eingegebenen Parameter durch Unterroutinen übernommen werden. So sind bei MOVETO und DRAWTO die Koordinaten X und Y zu übernehmen. FLOMON erwartet die X-Koordinate im HL-Register, die Y-Koordi-

nate im DE-Register. Die Einsprünge sind für MOVETO die Adresse F046H und für DRAWTO F049H. Die HEBAS-Routine AUSWO7 holt eine nach dem BASIC-Befehl angegebene Koordinate ins DE-Register. Der Wert darf auch als Variable vorliegen. Da der Befehl z. B. MOVETO 100,120 lautet, muß ein Komma nach der ersten Koordinate folgen, sonst soll ein Syntaxfehler ausgegeben werden. Dies erledigt die Routine VGLAHL, im Akku ist das zu prüfende Zeichen. Über den Stapel werden die Koordinaten dann in die entsprechenden Register geschoben und in die FLOMON-Routine gesprungen.

Bild 2 zeigt die Routinen der Befehle, die man der Übersicht halber am besten am Ende des Interpreters, allerdings noch vor dem Label BASPRO und der Byte-Definition DEFS 8 einschiebt, da sonst ein BASIC-Programm die neuen Befehle überschreiben würde. Beachten Sie bitte auch, daß vor dem Label BASPRO mindestens ein Byte den Wert 0 besitzt.

Zuletzt müssen noch in der Befehlstabelle TOKTB4 an der entsprechenden Stelle der Befehl und das Token eingetragen werden. Beim Befehl entfällt der erste Buchstabe, dieser ist immer am Anfang jeder neuen Buchstabengruppe mit gesetztem Bit 7 enthalten. Bild 3 enthält einen Auszug dazu.

Nun kann man sich auch Befehle wie BAGE X,Y oder LINE Y,Y,Z,U machen oder beliebig andere. Zur Zeit entsteht eine neue HEBAS-Version, die die Grafik-Befehle und etliche ganz praktische Befehle wie LOCATE enthält. Für die Aufsteiger vom EPROM-HEBAS ist auch der Befehl LOADC enthalten, damit die alten Programme vom Kassettenrecorder auf Diskette überspielt werden können. Der Preis für das Up-Date von HEBAS zur Grafik-Version beträgt beim Autor DM 30 (Verrechnungsscheck oder Einzugsermächtigung an Dr. Hehl, Lindenstr. 20, 8059 Wartenberg). Auf Wunsch können gegen Berechnung auch andere Befehle eingebaut werden. Ebenso sind verschieden große HEBAS-Versionen (evtl. mit verringertem oder erweitertem Befehlsumfang) für EPROM-Versionen möglich. So gibt es ein kleines EHEBAS (2 Eprom 2764) für die SBC3 bei der Firma Graf.

```
dw      0               ;80 95  BYE

DW      MOVETO          ;80 96
DW      CLR             ;80 97
DW      DRAWTO          ;80 98

dw      FTXT47          ;  "nicht vorhanden"
```

Bild 1: Ergänzungen der Zwei-Byte-Token-Tabelle TOKB2

```
CLR:
        CALL    CLRSCR          ;Bildschirm löschen
        RET
;-----------------------------------------------------------------

MOVETO:                         ;MOVETO X,Y
                                ;X-Wert nach HL, Y-Wert nach DE

        CALL    AUSWO7          ;Zahl oder Variable holen
        PUSH    DE              ;X-Wert auf Stapel
        LD      A,2CH           ;folgt Komma
        CALL    VGLAHL          ;sonst Fehler
        CALL    AUSWO7          ;Y-Wert holen
        EX      (SP),HL         ;HL retten u. Y-Wert nach HL
        CALL    0F046H          ;FLOMON MOVETO
        POP     HL              ;HL herstellen
        RET
;-----------------------------------------------------------------

DRAWTO:                         ;DRAWTO X,Y
                                ;X-Wert nach HL, Y-Wert nach DE

        CALL    AUSWO7          ;Zahl oder Variable holen
        PUSH    DE              ;X-Wert auf Stapel
        LD      A,2CH           ;folgt Komma
        CALL    VGLAHL          ;sonst Fehler
        CALL    AUSWO7          ;Y-Wert holen
        EX      (SP),HL         ;HL retten u. Y-Wert nach HL
        CALL    0F049H          ;FLOMON DRAWTO
        POP     HL              ;HL herstellen
        RET
```

Bild 2: Routinen für die Befehle CLR, MOVETO und DRAWTO

```
CCCC:                           ;Befehle mit C
        db      0C3H

        db      'LR'            ;CLR = Bildschirm löschen
        db      80H,97H

        db      'LEAR'
        db      97H

-----------------------------------------------------------------

DDDD:                           ;Befehle mit D
        db      0C4H

        db      'RAWTO'         ;DRAWTO X,Y
        db      80H,98H

        db      'ATA'
        db      83H

-----------------------------------------------------------------

MMMM:                           ;Befehle mit M
        db      0CDH

        db      'OVETO'         ;MOVETO X,Y
        db      80H,96H

        db      'AT'
```

Bild 3: Ergänzungen der Befehlstabelle TOKTB4

FÜR 68000-EINSTEIGER

Ergänzung zum Skop-Programm mit dem AD 10 x 1

von Bruno Sontheim

Leider ist beim oftmaligen Kopieren einer der wichtigsten Programmteile „abhanden" gekommen. Es ist das Unterprogramm „Messen", das eine definierte Abtastfrequenz einstellt.

Das kleine Unterprogramm kann am Ende des Programms S/ADW10.ASM eingegeben werden. Natürlich muß es vom Hauptprogramm aufgerufen werden. Fügen Sie dazu bitte folgenden Befehl in die 15. Zeile des Programms ein:

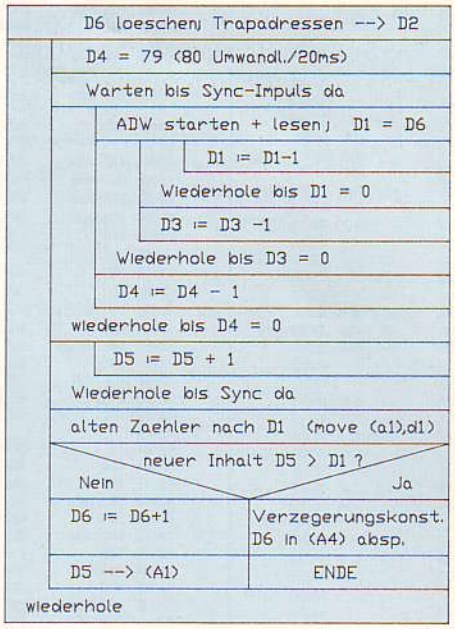
(nach move#30,28(a4)*buffer anfangswert)
BSR MESSEN

Jetzt muß das Ganze nur noch neu assembliert werden.

Wie funktioniert nun diese automatische Anpassung der Abtastfrequenz?

Die GDP liefert genau alle 20 ms einen Synchronisations-Impuls. Mit Hilfe dieses Taktes kann nun die Abtastfrequenz exakt eingestellt werden. Die größte Abtastfrequenz, die das Programm benutzt, beträgt 4 KHz. Das bedeutet, daß in 20 ms 80 analoge Signale in digitale Werte umgewandelt werden müssen. Für eine Umwandlung bleiben also genau 250 us Zeit.

Mit dem AD 10 x 1 sind aber viel schnellere Taktfrequenzen realisierbar. Also muß die verbleibende Zeit in einer Warteschleife „verbraten" werden. Wie oft diese Warteschleife durchlaufen wird ist systemabhängig. Dabei spielen der verwendete Prozessor, wie auch die eingestellten Waitzyklen eine Rolle. Das kleine Unterprogramm wird solange durchlaufen, bis für die Umwandlung der 80 Werte genau 20 ms benötigt werden. Wie oft die Warteschleife durchlaufen werden muß, wird in (A4) festgehalten. Natürlich enthält das UP die gleichen Elemente wie das eigentliche „Leseprogramm". Damit ist für eine ausreichende Genauigkeit gesorgt. Der genaue Ablauf ist dem Struktogramm zu entnehmen.



Welche Möglichkeiten bietet dieses Programm:

Natürlich kann es kein Oszilloskop ersetzen. Aber als Ergänzung für ein Meßlabor ist es durchaus geeignet.

Jeder der sich mit der Elektronik beschäftigt weiß, wie wichtig es ist, den Kurvenverlauf kleiner Frequenzen sichtbar zu machen. Hier ist dieses Programm in Verbindung mit dem AD 10 x 1 eine sinnvolle Erweiterung des Oszilloskops. Es können mühelos Spannungsabläufe mit Frequenzen bis zu 0.2 Hz dargestellt werden.

Eine weitere Anwendungsmöglichkeit ist mit der externen Triggerung gegeben. Damit sind einmalige Impulse darstellbar. So ist es zum Beispiel möglich, den Spannungsverlauf beim Einschalten eines Gerätes zu messen und mit der Hardcopy zu dokumentieren.

Ansonsten bietet das Programm den gleichen Komfort wie ein normales Oszilloskop.

```
messen:
      movem.l   do-d7/a0-a3,-(a7)
      move      #110,d2              * Trap ADW 10x1 lesen
      swap      d2
      move      #28,d2               * Trap Sync-Impuls GDP
      lea       adwstart(pc),a0      * unbenutzter Rambereich
      adda.l    #$3200,a0
      movea.l   a0,a1
      adda.l    #4,a1
      move.l    #8000,(a1)           * 1. Wert fuer Wartezeit
      clr.l     d6                   * Verzoegerungszaehler
mess1:
      move      #79,d4               * 80 Werte lesen
      clr.l     d5
mess2:
      move      d2,d7                * Trap Vektor 28
      trap      #1
      beq.s     mess2
      swap      d2                   * Trap Vektor 110
mess3:
      move      d6,d1                * Verzoegerungszaehler nach d1
      moveq     #0,d3
      move      d2,d7
      trap      #1                   * Wert lesen
mess4:
      dbf       d1,mess4             * hier Verzoegberusngsschleife
      dbf       d3,mess3             * Schleife zur Zeiteinstellung
      move      d0,(a0)              * Wert abspeichern
      dbf       d4,mess3             * wiederhole bis 80 Werte gewandelt
      swap      d2
mess5:
      addq.l    #1,d5                * Warte-Zeitzaehler
      move      d2,d7                * Trap Sync
      trap      #1
      beq.s     mess5                * warten bis Impuls da
      move.l    (a1),d1              * alten Warte-Zeitzaehler holen
      cmp       d5,d1                * neuer groesser als alter?
      bmi       messende             * dann Wert gefunden
      addq.l    #1,d6                * sonst Delaywert + 1
      move.l    d5,(a1)              * neuen Wert abspeichern
      bra       mess1                * weiter
messende:
      move      d6,(a4)              * Ergebnis abspeichern
      clr.l     (a0)
      clr.l     (a1)
      movem.l   (a7)+,d0-d7/a0-a3    * alten Zustand wiederherstellen
      rts
```

RUBAFIND-Programm zum Suchen von Bytefolgen im Speicher

von Rüdiger Bäcker

In der Reihe der Utilitys für den 680xx heute ein Programm zum Suchen von Bytefolgen im Speicher. Damit können einzelne Bytes, Bytesfolgen, Zeichenketten etc. schnell gefunden werden.

Die Eingabe der Adressen von, bis sowie der Suchkriterien erfolgt dabei wieder in der von allen anderen RUBAxxxx-Programmen bekannten Art. Falsche Eingaben werden dabei zurückgewiesen. Mit der Eingabe von 'm' als erstes Zeichen in der Eingabezeile gelangt man zum Grundprogramm bzw. zu JADOS zurück.

```
Rubafind V 1.6

(C) 1987 by Ruediger Baecker

Startadresse --> $400
Endadresse   --> $10000
Suchen nach  --> 'Rubafind'

Ausgabeart ? 1 = Bildschirm, 2 = Drucker

00000404
00000728

Ende der Suche - (CR) = Return
```

Die Ausgabe der Adressen, an denen der Suchwert gefunden wurde, kann wahlweise auf den Bildschirm oder den Drukker erfolgen.

Bild 1 zeigt eine Eingabe. Es wird der ASCII-String 'Rubafind' gesucht. Diese Bytefolge steht auf Adresse $404 und $728.

Bei der Eingabe des Suchwertes ist zu beachten, daß bei HEX=Werten ein "$" und bei ASCII-Werten ein "'" vorangestellt werden muß.

Da das Listing wieder ausführlich dokumentiert ist, kann ich mir weitere Ausführungen sparen.

Zum Schluß noch die Anmerkung, daß alle RUBAxxxx-Programme mittlerweile in der neuesten Version an JADOS angepaßt wurden, weitere Informationen sind gegen eine 0,80 DM Briefmarkte bei mir erhältlich.

```
10.06.87-13.07 Asserint - (C) 1987 by R. Baecker - Listing des Files : 2:RUBAFIND.S
Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 1

020000          *       R U B A F I N D
020000          *
020000          *       ROUTINE ZUM SUCHEN VON ZEICHENKETTEN ETC.
020000          *
020000          *       COPYRIGHT (C) 1987 BY RUEDIGER BAECKER - POSTFACH 4111 - 5820 GEVELSBERG
020000          *
020000          *       V 1.0 - 07.06.87
020000          *       V 1.1 - FEHLERBEHANDLUNG BEI READ - 07.06.87
020000          *       V 1.2 - FEHLERBEHANDLUNG BEI WERT - DTO.
020000          *       V 1.3 - WAHL DER AUSGABEART - 08.06.87
020000          *       V 1.4 - SEITENUMSCHALTUNG MIT SETFLIP ABSCHALTEN - 08.06.87
020000          *       V 1.5 - CTRL S/CTRL Q = STOP/START & HOME IST ABBRUCH - 08.06.87
020000          *       V 1.6 - ENDEMELDUNG BEI ABBRUCH MIT 'm' UNTEN AUSGEBEN - 08.06.87
020000
020000          *******
020000          *** BIBLIOTHEKSKOPF
```

```
020000                  +++++++
020000
020000                  KOPF:                           * BIBLIOTHEKSEINTRAG
020000  55AA0180        DC.L $55AA0180
020004  527562616669 6E DC.B 'Rubafind'
020008  64
02000C  00000020        DC.L RUBAFIND-KOPF
020010  000003D3        DC.L ENDEA-KOPF
020014  01              DC.B 1
020015  00 00 00        DC.B 0,0,0
020018  00000000        DC.L 0,0
02001C  00000000
020020
020020                  +++++++
020020                  +++ VEREINBARUNGEN
020020                  +++++++
020020
020020                  ++++++ ZWISCHENSPEICHER
020020
= 000009DA              BUFFER1         EQU     $9DA    * ADRESSE EINGABEBUFFER
= 00000A02              BUFFER2         EQU     BUFFER1+40
020020
020020                  ++++++ GRUNDPROGRAMMFUNKTIONEN
020020
= 00000066              SETCURXY EQU    102
= 00000022              SETFLIP EQU     34
= 0000000D              CSTS    EQU     13
= 00000032              LST     EQU     50
= 00000031              CRT     EQU     49
= 00000021              CO2     EQU     33
= 0000000C              CI      EQU     12
= 00000010              CLR     EQU     16
= 0000000A              WRITE   EQU     10
= 0000000B              READ    EQU     11
= 0000001D              WERT    EQU     29
= 0000002C              PRINTBX EQU     44
= 00000014              CLRSCREEN EQU   20
020020
020020                  +++++++
020020                  +++ HAUPTPROGRAMM
020020                  +++++++
020020
020020                  RUBAFIND:
020020  42B6            CLR.L D6                        * D6 NIMMT LAENGE DES SUCHSTRINGS AUF
020022  7E10            MOVEQ #CLR,D7                   * BILDSCHIRM LOESCHEN
020024  4E41            TRAP #1
020026  7000            MOVEQ #0,D0                     * SEITENUMSCHALTUNG ABSCHALTEN, DAMIT
```

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

```
020028  7200            MOVEQ #0,D1                     * BILDSCHIRM BEI CI ETC. NICHT BLINKT
02002A  7E22            MOVEQ #SETFLIP,D7               * ABSCHALTUNG UEBER SETFLIP, NORMALMODUS WIRD AM
02002C  4E41            TRAP #1                         * ENDE DES PROGRAMMS DURCH CLRSCREEN ERREICHT
02002E
02002E  41FA 02FB       LEA HEADTXT1 (PC),A0            * TEXTE AUSGEBEN
020032  7043            MOVEQ.B #$43,D0                 * SCHRIFTGROESSE
020034  7200            MOVEQ #0,D1                     * X-POSITION
020036  343C 00C8       MOVE #200,D2                    * Y-POSITION
02003A  7E0A            MOVEQ #WRITE,D7                 * UND AUSGEBEN
02003C  4E41            TRAP #1
02003E  0442 002D       SUBI #45,D2
020042  7021            MOVEQ.B #$21,D0
020044  41FA 0324       LEA HEADTXT5(PC),A0             * NAECHSTER TEXT USW.
02004B  7E0A            MOVEQ #WRITE,D7
02004A  4E41            TRAP #1
02004C  0442 002B       SUBI #40,D2
020050  7021            MOVEQ.B #$21,D0
020052  41FA 02E3       LEA HEADTXT2(PC),A0
020056  7E0A            MOVEQ #WRITE,D7
02005B  4E41            TRAP #1
02005A  0442 0014       SUBI #20,D2
02005E  41FA 02EB       LEA HEADTXT3(PC),A0
020062  7E0A            MOVEQ #WRITE,D7
020064  4E41            TRAP #1
020066  0442 0014       SUBI #20,D2
02006A  41FA 02ED       LEA HEADTXT4(PC),A0
02006E  7E0A            MOVEQ #WRITE,D7
020070  4E41            TRAP #1
020072
020072                  GETW1:                          * DANN DATEN HOLEN (VON, BIS, SUCHSTRING)
020072  7021            MOVEQ.B #$21,D0                 * DAZU WIEDER SCHRIFTGROESSE
020074  7473            MOVEQ #115,D2                   * UND POSITIONEN BELEGEN
020076  323C 00C8       MOVE #200,D1
02007A  7609            MOVEQ.B #9,D3                   * ADRESSE MAX. 9 ZEICHEN LANG
02007C  41ED 09DA       LEA BUFFER1(A5),A0              * ABLAGEPUFFER
020080  7E0B            MOVEQ #READ,D7                  * UND WERT HOLEN
020082  4E41            TRAP #1
020084  0C05 000D       CMPI.B #$D,D5                   * TESTEN, OB LETZTES ZEICHEN CR WAR, WENN NICHT
02008B  66EB            BNE.S GETW1                     * DANN WIEDERHOLEN, DA UEBERLAUF
02008A
02008A  41ED 09DA       LEA BUFFER1(A5),A0              * SONST WERT ERMITTELN
02008E  0C10 006D       CMPI.B #'m',(A0)                * ERSTES EINGEGEBENES ZEICHEN HOLEN UND TESTEN,
020092  6700 0252       BEQ ENDE1                       * OB 'm' GEDRUECKT ? JA, DANN ENDE, SONST MUSS
020096  0C10 0024       CMPI.B #'$',(A0)                * ERSTES ZEICHEN EIN '$' SEIN, DA EINGABE IN HEX
02009A  66D6            BNE.S GETW1                     * ERFOLGEN MUSS ! NEIN, DANN WIEDERHOLEN
02009C
02009C  7E1D            MOVEQ #WERT,D7                  * SONST WERT ERMITTELN
02009E  4E41            TRAP #1                         * STEHT NUN IN D0
02000A0  0C41 0000      CMPI #0,D1                      * SYNTAX-FEHLER AUFGETRETEN ?
02000A4  67CC           BEQ.S GETW1                     * JA, DANN WIEDERHOLEN
02000A6  0C41 0005      CMPI #5,D1                      * NICHT VORHANDENES SYMBOL VERWANDT ?
02000AA  67C6           BEQ.S GETW1                     * AUCH DANN WIEDERHOLEN
02000AC  0C41 FFFF      CMPI #$FFFF,D1                  * INTERNER FEHLER AUFGETRETEN ?
02000B0  67C0           BEQ.S GETW1                     * AUCH DANN WIEDERHOLEN
02000B2  2640           MOVEA.L D0,A3                   * SONST WERT VON IN A3 RETTEN
02000B4
02000B4                 GETW2:                          * NUN BIS HOLEN
02000B4  745F           MOVEQ #95,D2                    * DAZU WIEDER POSITIONEN ETC.
02000B6  7021           MOVEQ #$21,D0
02000B8  323C 00C8      MOVE #200,D1
02000BC  41ED 09DA      LEA BUFFER1(A5),A0
02000C0  7E0B           MOVEQ #READ,D7
02000C2  4E41           TRAP #1
02000C4  0C05 000D      CMPI.B #$D,D5                   * NUN WIEDER TESTEN, OB EINGABE OK ETC.
```

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

```
02000C8  66EA           BNE.S GETW2
02000CA
02000CA  41ED 09DA      LEA BUFFER1(A5),A0
02000CE  0C10 006D      CMPI.B #'m',(A0)
02000D2  6700 0212      BEQ ENDE1
02000D6  0C10 0024      CMPI.B #'$',(A0)
02000DA  66D8           BNE.S GETW2
02000DC
02000DC  7E1D           MOVEQ #WERT,D7
02000DE  4E41           TRAP #1
02000E0  0C41 0000      CMPI #0,D1                      * SYNTAX-FEHLER AUFGETRETEN ?
02000E4  67CE           BEQ.S GETW2                     * JA, DANN WIEDERHOLEN
02000E6  0C41 0005      CMPI #5,D1                      * NICHT VORHANDENES SYMBOL VERWANDT ?
02000EA  67C8           BEQ.S GETW2                     * AUCH DANN WIEDERHOLEN
02000EC  0C41 FFFF      CMPI #$FFFF,D1                  * INTERNER FEHLER AUFGETRETEN ?
02000F0  67C2           BEQ.S GETW2                     * AUCH DANN WIEDERHOLEN
02000F2  2B40           MOVEA.L D0,A4                   * BIS IN A4
02000F4
02000F4                 GETW3:                          * NUN SUCHSTRING HOLEN
02000F4  744B           MOVEQ #75,D2
02000F6  7011           MOVEQ #$11,D0
02000F8  323C 00C8      MOVE #200,D1
02000FC  7622           MOVEQ #34,D3                    * SUCHSTRING MAX. 34 ZEICHEN LANG
02000FE  41ED 09DA      LEA BUFFER1(A5),A0              * WIRD AB A0 ABGELEGT
020102  42B4            CLR.L D4
020104  7E0B            MOVEQ #READ,D7
020106  4E41            TRAP #1                         * IN D4 STEHT ANZAHL EINGEGEBENER ZEICHEN
020108  0C05 000D       CMPI.B #$D,D5                   * UND WIEDER TESTEN
02010C  66E6            BNE.S GETW3
02010E
02010E  41ED 09DA       LEA BUFFER1(A5),A0              * SUCHSTRING STEHT NUN AB (A0)
020112  0C10 006D       CMPI.B #'m',(A0)                * ERSTES EINGEGEBENES ZEICHEN HOLEN UND TESTEN,
020116  6700 01CE       BEQ ENDE1                       * OB 'm' GEDRUECKT ? JA, DANN ENDE.
02011A
02011A  0C10 0027       CMPI.B #$27,(A0)                * WENN ASCII-STRING, DANN ANFANG MIT "'"
02011E  6700 00E0       BEQ FIND3                       * IN DIESEM FALL ANDERE ROUTINE
020122
020122                  FIND0:
020122  0C10 0024       CMPI.B #'$',(A0)                * HEX-WERT ?
020126  666B            BNE.S FIND1                     * NEIN, DANN TESTEN, OB BINAEREINGABE
020128  43ED 0A02       LEA BUFFER2(A5),A1              * SONST IN HEX WANDELN
02012C  5344            SUBQ #1,D4
02012E  88FC 0002       DIVU #2,D4                      * ASCII-BYTES DOPPELT SO VIELE WIE HEX
020132  3C04            MOVE D4,D6                      * LAENGE RETTEN
020134  5346            SUBQ #1,D6                      * UND ANGLEICHEN
020136  4B44            SWAP D4                         * REST IN UNTERES WORT
02013C  0C44 0000       CMPI #0,D4                      * REST MUSS 0 SEIN, DA NUR GERADER WERT OK
02013C  6600 FFB6       BNE GETW3                       * WENN NICHT, EINGABE WIEDERHOLEN
020140  4B44            SWAP D4                         * SONST WIEDER DREHEN
020142
020142  D1FC 00000001   ADDA.L #1,A0                    * $ UEBERSPRINGEN
020148
020148                  FINDOLP1:                       * UND DANN IN HEX WANDELN, ABLAGE IN BUFFER 2
020148  1018            MOVE.B (A0)+,D0                 * BYTE IN D0
02014A  6126            BSR.S PUTHEX                    * UND WANDELN
02014C  1200            MOVE.B D0,D1                    * UNTERES HALBBYTE RETTEN
02014E  1018            MOVE.B (A0)+,D0
020150  6100 0020       BSR PUTHEX
020154  E909            LSL.B #4,D1                     * OBERES HALBBYTE AN RICHTIGE STELLE SCHIEBEN
02015A  D200            ADD.B D0,D1                     * UND ZUSAMMENFASSEN
02015B  12C1            MOVE.B D1,(A1)+                 * DANN IN BUFFER ABLEGEN
02015A  51CC FFEC       DBRA D4,FINDOLP1                * BIS ALLES GEWANDELT
02015E
02015E  41ED 09DA       LEA BUFFER1(A5),A0
```

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

```
020162  43ED 0A02       LEA BUFFER2(A5),A1
020166  3B06            MOVE D6,D4
020168
020168                  FINDOLP2:                       * DANN WIEDER ZURUECK IN BUFFER 1
020168  10D9            MOVE.B (A1)+,(A0)+              * ABLEGEN
02016A  51CC FFFC       DBRA D4,FINDOLP2
02016E  6000 00A8       BRA FIND
020172
020172                  PUTHEX:                         * IN HEX WANDELN
020172  0C00 0039       CMPI.B #'9',D0                  * TESTEN, OB GROESSER 9
020176  6E06            BGT.S PUTHEX1                   * JA, DANN VERZWEIGEN
020178  0400 0030       SUBI.B #$30,D0                  * SONST $30 SUBTRAHIEREN
02017C  4E75            RTS
02017E
02017E                  PUTHEX1:                        * WENN WERT VON $A - $F, $37 SUBTRAHIEREN
02017E  0C00 0046       CMPI.B #'F',D0                  * TESTEN, OB GROESSER F
020182  6E06            BGT.S PUTHEX2
020184  0400 0037       SUBI.B #$37,D0
020188  4E75            RTS
020BA
020BA                   PUTHEX2:                        * WENN WERT VON 'a - f', MUSS $57 SUBTRAHIERT
020BA  0400 0057        SUBI.B #$57,D0                  * WERDEN
020BE  4E75             RTS
020190
020190                  FIND1:
020190  0C10 0025       CMPI.B #'%',(A0)                * BINAER-WERT ?
020194  6600 FF5E       BNE GETW3                       * NEIN, DANN WIEDERHOLEN, DEZIMALEINGABE VERB.
020198  0C04 0009       CMPI.B #9,D4                    * BEI BINAEREINGABE SIND NUR FOLGENDE LAENGEN
02019C  6D00 FF56       BLT  GETW3                      * MOEGLICH : 9, 17, 25, 33 (INC. %)
02001A0  671C           BEQ.S FIND1LP0                  * WENN 9, DANN DORT WEITER
0201A2  0C04 0011       CMPI.B #17,D4                   * WENN NICHT KLEINER 9, DANN 17 ODER GROESSER
0201A6  6D00 FF4C       BLT  GETW3                      * KLEINER 17 UND GROESSER 9, DANN FALSCH
0201AA  6712            BEQ.S FIND1LP0                  * WENN GLEICH, DANN DORT WEITER
0201AC  0C04 0019       CMPI.B #25,D4                   * WENN NICHT KLEINER 17, DANN 25 ODER GROESSER
0201B0  6D00 FF42       BLT  GETW3                      * KLEINER 25 UND GROESSER 17, DANN FALSCH
0201B4  670B            BEQ.S FIND1LP0                  * WENN GLEICH, DANN DORT WEITER
0201B6  0C04 0021       CMPI.B #33,D4                   * WENN NICHT KLEINER 25, DANN MUSS 33 SEIN.
```

```
0201BA  6600 FF38      BNE   GETW3                 * SONST AUCH FALSCH, ALSO WIEDERHOLEN
0201BE
0201BE                 FINDILP0:                   * NUN IN HEX WANDELN
0201BE  7E1D           MOVEQ #WERT,D7              * IN D0 STEHT DAN WERT ALS HEX, ABER IN
0201C0  4E41           TRAP #1                     * FALSCHER REIHENFOLGE !
0201C2  0C41 0000      CMPI #0,D1                  * SYNTAX-FEHLER AUFGETRETEN ?
0201C6  67F6           BEQ.S FINDILP0              * JA, DANN WIEDERHOLEN
0201C8  0C41 0005      CMPI #5,D1                  * NICHT VORHANDENES SYMBOL VERWANDT ?
0201CC  67F0           BEQ.S FINDILP0              * AUCH DANN WIEDERHOLEN
0201CE  0C41 FFFF      CMPI #$FFFF,D1              * INTERNER FEHLER AUFGETRETEN ?
0201D2  67EA           BEQ.S FINDILP0              * AUCH DANN WIEDERHOLEN
0201D4
0201D4  BBFC 0008      DIVU.B #8,D4                * IN D4 STEHT DANN DIE ZAHL DER BYTES,
0201D8  43ED 09DA      LEA BUFFER1(A5),A1          * DIE ALS HEXWERT IN BUFFER 1 MUESSEN
0201DC  D3FC 00000005  ADDA.L #5,A1                * RUECKWAERTS, DA IN D0 FALSCHE REIHENFOLGE
0201E2  4211           CLR.B (A1)                  * ENDEKENNUNG
0201E4  5344           SUBQ #1,D4                  * FUER DBRA
0201E6  3C04           MOVE D4,D6                  * LAENGE IN D6 RETTEN
0201E8
0201E8                 FINDILP1:
0201E8  1300           MOVE.B D0,-(A1)             * UNTERES BYTE IN BUFFER
0201EA  E098           ROR.L #8,D0                 * UND NAECHSTES BYTE NACH UNTEN SCHIEBEN
0201EC  51CC FFFA      DBRA D4,FINDILP1            * BIS ALLES UEBERTRAGEN, DANN NOCH RETRANSFER
0201F0  41ED 09DA      LEA BUFFER1(A5),A0          * IN BUFFER1
0201F4  3806           MOVE D6,D4                  * FUER RETRANSFER WIEDER IN D4
0201F6
0201F6                 FINDILP2:
```

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

```
0201F6  10D9           MOVE.B (A1)+,(A0)+
0201F8  51CC FFFC      DBRA D4,FINDILP2
0201FC  6000 001A      BRA FIND                    * UND ZUR HAUPTSUCHROUTINE
020200
020200                 FIND3:                      * ASCII-STRING SUCHEN
020200  43ED 09DA      LEA BUFFER1(A5),A1          * START DES BUFFER IN A1, FUER TRANSFER
020204  D1FC 00000001  ADDA.L #1,A0                *  ' UEBERSPRINGEN '
02020A  5744           SUBQ #3,D4                  * FUER DBRA UND DIE BEIDEN ''
02020C  3C04           MOVE D4,D6                  * LAENGE IN D6 RETTEN
02020E
02020E                 FIND3LP0:                   * UND UM EIN BYTE NACH UNTEN SCHIEBEN
02020E  12D8           MOVE.B (A0)+,(A1)+
020210  51CC FFFC      DBRA D4, FIND3LP0
020214  6000 0002      BRA FIND
020218
020218                 FIND:
020218  6100 00BE      BSR SETAUS                  * AUSGABEART SETZEN
02021C  7E14           MOVEQ #CLRSCREEN,D7         * ERST BILDSCHIRM LOESCHEN
02021E  4E41           TRAP #1                     * HAUPTROUTINE, IN D6 STEHT LAENGE DES SUCH-
020220  41ED 09DA      LEA BUFFER1(A5),A0          * STRINGS -1, SUCHSTRING STEHT IN PUFFER 1
020224  4285           CLR.L D5                    * ZAEHLER FUER GEFUNDENE BYTES
020226  5246           ADDQ #1,D6                  * IN D6 DANN TATS. LAENGE
020228
020228                 FINDLP:
020228  7E0D           MOVEQ #CSTS,D7              * TESTEN, OB TASTATUR GEDRUECKT
02022A  4E41           TRAP #1
02022C  0C00 0000      CMPI.B #0,D0                * BEI TASTENDRUCK D0.B = <> 0
020230  6600 00D0      BNE GETTASTE                * JA, DANN ZEICHEN HOLEN
020234  B9CB           CMPA.L A3,A4                * TESTEN, OB ENDADRESSE ERREICHT
020236  6700 00B4      BEQ ENDE                    * JA, DANN ENDE
02023A  121B           MOVE.B (A3)+,D1             * WERT AUS SPEICHER IN D1
02023C  B210           CMP.B (A0),D1               * WERT IM SPEICHER GLEICH 1. WERT SUCHSTRING ?
02023E  66E8           BNE.S FINDLP                * NEIN, DANN WEITER
020240
020240  200B           MOVE.L A3,D0                * WENN JA, DANN GEFUNDENE ADRESSE
020242  0480 00000001  SUBI.L #1,D0                * IN D0
020248  D1FC 00000001  ADDA.L #1,A0                * UND A0 ZEIGT AUF 2. WERT DES SUCHSTRINGS
02024E
02024E                 FINDLP1:                    * NUN TEST, OB REST AUCH GLEICH
02024E  5245           ADDQ #1,D5
020250  BC45           CMP  D5,D6                  * ERST TESTEN OB ALLES GEFUNDEN
020252  6700 0010      BEQ ENDFIND                 * WENN JA, DANN ENDFIND1
020256  B10B           CMPM.B (A3)+,(A0)+          * SONST WEITER VERGLEICHEN
020258  67F4           BEQ.S FINDLP1               * SOLANGE GLEICH
02025A
02025A  41ED 09DA      LEA BUFFER1(A5),A0          * WENN NICHT GANZ GLEICH, DANN BUFFERADR. IN A0
02025E  4285           CLR.L D5                    * ZAEHLER FUER GEFUNDENE BYTES LOESCHEN
020260  6000 FFC6      BRA FINDLP                  * UND WEITERSUCHEN
020264
020264                 ENDFIND:                    * BEREICH MIT GLEICHHEIT GEFUNDEN, ADR. ANZEIGEN
020264  6100 000C      BSR SHOWADR                 * DORT WIRD ADRESSE GEWANDELT UN AUSGEGEBEN
020268  41ED 09DA      LEA BUFFER1(A5),A0          * NUN WIEDER AUF BEGINN DES SUCHSTRINGS ZEIGEN
02026C  4285           CLR.L D5                    * ZAEHLER FUER GEFUNDENE BYTES LOESCHEN
02026E  6000 FFB8      BRA FINDLP                  * UND WEITERSUCHEN
020272
020272                 SHOWADR:                    * GEFUNDENE ADRESSE ANZEIGEN, STEHT IN D0
020272  48E7 FFF8      MOVEM.L D0-D7/A0-A4,-(A7)   * REGISTER RETTEN
020276  41ED 0A02      LEA BUFFER2(A5),A0          * BUFFERADRESSE
02027A  7E2C           MOVEQ #PRINT8X,D7           * NUN ALS ACHTSTELLIGEN HEXWERT IN BUFFER
02027C  4E41           TRAP #1                     * (A0) -> AUF ENDE DES BUFFERS
02027E  10FC 000D      MOVE.B #$D,(A0)+            * NOCH CR UND
020282  10FC 000A      MOVE.B #$A,(A0)+            * LF ANFUEGEN
020286  4210           CLR.B (A0)                  * UND 0 ALS ENDEKENNUNG
020288  41ED 0A02      LEA BUFFER2(A5),A0          * BUFFERADRESSE WIEDER HOLEN
```

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 6

```
02028C  6100 0008      BSR SCHREIBE                * UND UEBER CO2 AUSGEBEN
020290  4CDF 1FFF      MOVEM.L (A7)+,D0-D7/A0-A4   * REGISTER ZURUECK
020294  4E75           RTS                         * AUFRUFENDER ROUTINE, DORTHIN ZURUECK
020296
020296                 SCHREIBE:                   * TEXT UEBER CO2 AUSGEBEN
020296  1018           MOVE.B (A0)+,D0             * STEHT AB (A0), HOLEN NACH D0
020298  0C00 0000      CMPI.B #0,D0                * 0 = ENDEKENNUNG
02029C  6708           BEQ.S ENDSCHREIBE           * DANN RETURN
02029E  7E21           MOVEQ #CO2,D7               * SONST AUSGEBEN
0202A0  4E41           TRAP #1
0202A2  6000 FFF2      BRA SCHREIBE                * UND WEITER
0202A6
0202A6                 ENDSCHREIBE:                * ZURUECK ZU HAUPTPROGRAMM
0202A6  4E75           RTS
0202A8
0202A8                 SETAUS:                     * AUSGABEART SETZEN
0202A8  48E7 FFFE      MOVEM.L A0-A6/D0-D7,-(A7)   * ERSTMAL REGISTER RETTEN
0202AC
0202AC                 SETAUS1:
0202AC  41FA 00FC      LEA AUSTXT(PC),A0
0202B0  7011           MOVEQ #$11,D0
0202B2  7200           MOVEQ #0,D1
0202B4  7414           MOVEQ #20,D2
0202B6  7E0A           MOVEQ #WRITE,D7
0202B8  4E41           TRAP #1
0202BA  7E0C           MOVEQ #CI,D7
0202BC  4E41           TRAP #1
0202BE  0C00 0031      CMPI.B #'1',D0
0202C2  6700 0010      BEQ SETCO
0202C6  0C00 0032      CMPI.B #'2',D0
0202CA  6700 0010      BEQ SETLO
0202CE  6000 FFDC      BRA SETAUS1
0202D2  4E75           RTS                         * UND RETURN
0202D4
0202D4                 SETCO:                      * AUSGABE AUF BILDSCHIRM SETZEN
0202D4  7E31           MOVEQ #CRT,D7               * CO2-AUSGABE AUF BILDSCHIRM UMLENKEN
0202D6  4E41           TRAP #1
0202D8  6000 0006      BRA SETRET
0202DC
0202DC                 SETLO:                      * AUSGABE AUF DRUCKER SETZEN
0202DC  7E32           MOVEQ #LST,D7
0202DE  4E41           TRAP #1
0202E0
0202E0                 SETRET:
0202E0  4CDF 7FFF      MOVEM.L (A7)+,A0-A6/D0-D7   * REGISTER ZURUECK
0202E4  4E75           RTS
0202E6
0202E6                 ENDE1:                      * ENDE BEI ABBRUCH
0202E6                 MOVEQ 0,D1                  * DAZU VORHER CURSORPOSITION SETZEN
0202E6  7414           MOVEQ #20,D2                * DAMIT ENDEMELDUNG UNTEN ERSCHEINT
0202E8  7E66           MOVEQ #SETCURXY,D7
0202EA  4E41           TRAP #1
0202EC
0202EC                 ENDE:
0202EC  7E31           MOVEQ #CRT,D7
0202EE  4E41           TRAP #1
0202F0  41FA 0095      LEA ENDETXT1(PC),A0
0202F4  6100 FFA0      BSR SCHREIBE
0202F8  7E0C           MOVEQ #CI,D7
0202FA  4E41           TRAP #1
0202FC  7E14           MOVEQ #CLRSCREEN,D7
0202FE  4E41           TRAP #1
020300  4E75           RTS
020302
```

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 7

```
020302                 GETTASTE:                   * TASTE GEDRUECKT -->
020302  7E0C           MOVEQ #CI,D7                * TESTEN, OB HOME GEDRUECKT
020304  4E41           TRAP #1
020306  0C00 000F      CMPI.B #$0F,D0
02030A  6700 FFE0      BEQ ENDE                    * JA, DANN ABBRECHEN
02030E  0C00 0013      CMPI.B #$13,D0              * CTRL S ?
020312  6700 0006      BEQ WARTE                   * DANN WARTEN BIS CTRL Q
020316  6000 FF10      BRA FINDLP                  * SONST WEITER
02031A
02031A                 WARTE:                      * WARTEN, BIS CTRL Q GEDRUECKT
02031A  7E0C           MOVEQ #CI,D7                * TASTE HOLEN
02031C  4E41           TRAP #1
02031E  0C00 0011      CMPI.B #$11,D0              * CTRL Q ?
020322  66F6           BNE.S WARTE                 * NEIN, DANN WEITERHIN WARTEN
020324  6000 FF02      BRA FINDLP                  * SONST WEITER
020328
020328                 HEADTXT1:
020328  5275626166696E DC.B 'Rubafind V 1.6',0
02032F  64205620312E36
020336  00
020337                 HEADTXT2:
020337  53746172746164 DC.B 'Startadresse -->',0
02033E  7265737365202D
020345  2D3E00
020348                 HEADTXT3:
020348  456E6461647265 DC.B 'Endadresse   -->',0
02034F  7373652020202D
020356  2D3E00
020359                 HEADTXT4:
020359  53756368656E20 DC.B 'Suchen nach  -->',0
020360  6E61636820202D
020367  2D3E00
02036A                 HEADTXT5:
02036A  28432920313938 DC.B '(C) 1987 by Ruediger Baecker',0
020371  37206279205275
020378  65646967657220
02037F  42616563686572
020386  00
020387
020387                 ENDETXT1:
020387  0D 0A 0A 456E  DC.B 13,10,10,'Ende der Suche - <CR> = Return ',0
02038C  64652064657220
020393  5375636865202D
02039A  203C43523E203D
0203A1  2052657475726E
0203A8  2000
0203AA
0203AA                 AUSTXT:
0203AA  41757367616265 DC.B 'Ausgabeart ? 1 = Bildschirm, 2 = Drucker',0
0203B1  617274203F2031
0203B8  203D2042696C64
0203BF  73636869726D2C
0203C6  2032203D204472
0203CD  75636B657200
0203D3
0203D3
= 000203D3             ENDEA EQU *
0203D3                 END.
```

JUMP per Interrupt

von Curt Michaelis,
Jankeweg 25, 2000 Hamburg 71

Der 68000 hat einen sehr mächtigen Befehlssatz – so kann man es überall lesen, und tatsächlich bieten die vielfältigen Adressierungsarten einen Komfort, den wir bei früheren Prozessoren vermißt haben. Sobald man aber erst einmal mit dem Programmieren begonnen hat, bemerkt man auch die Lücken im Programmiermodell – vor allem gibt es, von TRAPV und CHK abgesehen, überhaupt keine bedingten Unterprogrammaufrufe!

Nun kann man sich solche allerdings mittels PEA und Bcc zusammensetzen und als Makros installieren – allerdings nur für programmzählerrelative Adressierung (ohne Index) mit der Reichweite vorzeichenbehafteter 16-Bit Zahlen (erst bei dem „erwachsenen" 68020 entfällt die letztgenannte Beschränkung). JMP und JSR gibt es dagegen mit allen hier sinnvollen Adressierungsarten, dafür nun aber wieder nicht mit Konditionierung.

Mit Rücksicht darauf, daß das Motorola-Konzept offenbar dazu einlädt, einen besonderen Supervisor-Speicher einzurichten, wird man allerdings die Systemunterprogramme möglichst über TRAPs anspringen.

Leider gibt es innerhalb einer Liste von 256 Vektoren nur 20, die damit zu erreichen sind; setzt man nun noch den 68010 oder 68020 ein, um die Basis der Liste versetzen zu können, ist dies ein einigermaßen mißlicher Umstand. Beim Montieren der neuen IOE 2-Karte, mit den Schaltern und Leuchtdioden, die im Gehäuse versteckt eine undefinierte Aufgabe erfüllen sollen, ist mir nun die Idee gekommen, IC4 als Register so zu schalten, daß beim Schreiben darin ein Interrupt ausgelöst wird, über den dann alle 256 Vektoren zu erreichen sind!

Ich wollte es genau wissen und habe den Fädelstift in die Hand genommen:

Entfernt man von der IOE 2-Karte IC2 und verbindet man unter IC4 die Eingänge mit den Ausgängen – also die Pins 2 mit 3, 4 mit 5, 6 mit 7, 8 mit 9, 12 mit 13, 14 mit 15, 16 mit 17, 18 mit 19, außerdem Pin 1 von IC4 mit Pin 6 von IC9, so bekommt man ein Register, aus dem man lesen kann, was man vorher hineingeschrieben hat. Damit nun beim Schreiben ein Interrupt angefordert wird, verbindet man Pin 10 von IC6 mit dem nächstliegenden Stift von JMP 5 (der wird aber nicht gesteckt!). Um auch noch die Vektornummer durch die Interruptquittung abrufen zu können, nimmt man die Leseleitung von Pin 6 IC9 wieder ab und verbindet statt dessen Pin 1 von IC4 mit Pin 6 von IC10 sowie mit Steckerstift 54. JMP 4 muß gesteckt werden!

Nun muß natürlich noch die Interruptquittung auf die Busleitung 54 gebracht werden. Auf der CPU 68-Karte ist dazu Pin 8 von IC8 an Steckerstift 54 anzuschließen. Damit sich die CPU nun nicht doch den Autovektor holt, muß /VPA abgehängt werden; hierzu nimmt man die CPU vorsichtig aus der Fassung, biegt Pin 39 hoch, verbindet ihn irgendwie mit Pin 40 und steckt sie so wieder ein.

Ist nun eine FLO 3-Karte vorhanden, muß die zur Ruhe gebracht werden; sie aktiviert nämlich ganz unnötigerweise die /INT-Leitung. Hierzu ist JMP 4 zu entfernen (fehlt im Schaltbild!); zweckmäßigerweise knipst man auch gleich R15 heraus – der ist nämlich ein reiner Konstruktionsfehler: r liegt parallel zu RN3 auf der IOE 2 – sowie zu R10 auf der CPU 68K – wenn JMP 2 gesteckt wird, kommt auch noch R9 hinzu, so daß für das treibende Gatter ein Kollektorvorwiderstand von nur 250 Ohm resultiert!

Wie funktioniert's? Nun, es gibt sehr gute Gründe dafür, daß der Prozessor im Supervisor-Status keine Interruptverarbeitung aufnimmt; es steht nur nirgends geschrieben! Bevor Sie aber den User-Status nach SR laden, müssen Sie den User-Stack einrichten. Am einfachsten ist es, ihn an den Supervisor-Stack anzuhängen durch MOVE A7,USP. Wollen Sie den User-Stack dagegen lieber beispielsweise in einen isolierten Ram-Bereich legen, müssen Sie auch noch die Rücksprungadresse dorthin verfrachten. Veranläßt man nun irgendwann vorher oder nachher den Interrupt, holt sich der Prozessor den Vektor mit der Nummer aus $ffffff4a. Das kann so aussehen:

```
        lea     $a6000,a0       ; Basis User-Stack
        move.l  (a7),(a0)       ; Rücksprungadresse dorthin
        move    a0,usp          ; Stackpointer
        move    #$2400,sr       ; Aufruf User-Modus,Int.Lev. 4
        .
        .
        move    #56,d7          ; Parameter für Editor-Aufruf
        move.b  #33,$ffffff4a   ; Vektor-Nummer für Trap 1. Interrupt!
        rts

        Auch bedingte Unterprogrammaufrufe kann man damit verwirklichen, indem
man

interrupt:
        move.b  d6,$ffffff4a    ; Vektornummer in d6
        rts                     ; zurück unter Bcc Interrupt

mit
        pea     *+8(PC)         ; Kein Druckfehler! Rücksprungadresse
        bcc     interrupt       ; Unterprogrammaufruf dazu
```

anspringt. Alles klar?

Leider stehen im ROM hinter den meisten Vektornummern lauter schöne, runde Nullen; um Nutzen aus dem Verfahren ziehen zu können, müßte man erst einmal ein neues brennen.

Damit nun aber keine Mißverständnisse aufkommen: wenn keine weiteren Vorkehrungen getroffen werden, funktioniert der Interrupt mit Autovektor so nicht mehr; jeder zugelassene Interrupt wird sich die in $ffffff4a stehende Zahl als Vektornummer holen. Man kann aber verhältnismäßig einfach mit A2 (Adressleitung) umschalten und sogar ein zweites Register einrichten, das bei Interrupts aus anderer Quelle abgefragt wird; schließlich könnte man in Betracht ziehen, das Interrupt-Register als sozusagen Pico-Coprozessor auf einer neuen CPU-Karte zu installieren.

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

Objectfile-Format für JADOS

von Klaus Janßen

Ich freue mich, nun endlich ein einheitliches Format für Objectfiles vorstellen zu können. Bereits in der *LOOP* 14 wurde ein entsprechender Hinweis (siehe Schriftwechsel Husemann – Janßen) abgedruckt.

Das vorliegende Format berücksichtigt im besonderen Maße die Entwicklung relokativer Programme (Programme, die an keine feste Startadresse gebunden sind). Damit steht dem Einsatz höherer Programmiersprachen auf dem 68000-System theoretisch nichts mehr im Wege. Damit es nicht bei reiner Theorie bleibt, muß nun der nächste Schritt getan werden. Dieser besteht zunächst darin, einen Linker zu programmieren, der aus den Objectfiles ablauffähige .68K-Files erzeugt. Nun nutzt ein Linker leider wenig, wenn es kein Übersetzerprogramm gibt, das das vorliegende Objectfile-Format unterstützt. Daher soll auf jeden Fall schnellstmöglich ein komfortabler Assembler entwickelt werden, der Objectfiles erzeugen kann. Da ich mit der Entwicklung eines solchen Assemblers zeitlich überfordert bin, wäre es schön, wenn einer der Softwarepartner den Assembler programmiert, während ich gleichzeitig den Linker „auf die Beine stelle".

Wer das Objectfile-Format unterstützen will, kann die vollständige Beschreibung bei GES kostenlos beziehen.

Und nun das Format in Kurzfassung:

Definition des Object-Formates

Das Object-File besteht aus einer Anzahl von Records, die den Maschinencode und die Symbole sowie die zugehörigen Adressen enthalten. Ein Record besteht formal aus einem Recordtyp, einer Recordlänge und dem Recordinhalt; siehe nachstehende Abbildung.

```
-------------------------///-------
¦  TYP ¦    LÄNGE    ¦  I N H A L T ¦
-------------------------///-------
```

Der Typ identifiziert eindeutig die Bedeutung des Records und besteht aus einem Byte. Die Länge gibt an, wieviele Bytes das Record hinter der Längenangabe umfaßt. Die Länge selbst wird mit 2 Bytes angegeben, so daß Records bis zu 64k enthalten können.

Die im folgenden mit „Name" bezeichneten Einträge sind wie folgt spezifiziert: Ein Name besteht aus den Buchstaben "A" .. "Z", "a" .. "z", "0" .. "9" und "_. Zwischen Groß- und Kleinbuchstaben wird unterschieden. Ein Name umfaßt immer 16 Zeichen. Kürzere Namen werden von einer binären 0 beendet.

Die einzelnen Records im Objectfile müssen in folgender Reihenfolge stehen:

MHEADER
GLOBLAB – Records
GLOBVAR – Records
EXTLAB – Records
EXTVAR – Records
weitere Records
MODEND

Das erste Record eines Object-Files muß den Modulnamen enthalten, das letzte Record definiert das Modulende.

Im JADOS-System muß ein Object-File mit der Extension .OBJ gekennzeichnet sein.

Modulanfang-Record (MHEADER)

Dies muß das erste Record eines Object-Files sein. Es identifiziert das Object-File mit einem Namen. In einer Library wird es z.B. benötigt, um den Anfang eines Moduls finden zu können. Optionell kann eine Startadresse mitgegeben werden.

Typ (1 Byte):	$01
Länge (2 Byte):	$001A
Modulname (16 Bytes):	Name des Moduls
Dateilänge (4 Bytes):	Länge der Objectdatei in Bytes
CPU-Typ (1 Byte):	1 = 68000/08-Prozessor, 2 = 68020-Prozessor
Reserve (1 Byte):	$00
Startadresse (4 Bytes):	Offset zum Modulanfang

Modulende-Record (MODEND)

Dies muß das letzte Record eines Object-Files sein. Es dient lediglich dazu, das Ende eines Object-Files zu markieren.

Typ (1 Byte):	$02
Länge (2 Byte):	$0000

Maschinencode-Record (MCODE)

Dieses Record enthält Maschinencode. Es dürfen mehrere MCODE-Records im Object-File enthalten ein.

Typ (1 Byte):	$03
Länge (2 Bytes):	
Code:	Maschinencode relativ zur Adresse 0

Der Maschinencode muß relativ zur Adresse 0 übersetzt sein.

Globales Label-Record (GLOBLAB)

Dieses Record enthält den Namen und die relative Adresse bezüglich 0 einer Prozedur.

Typ (1 Byte):	$04
Länge (2 Bytes):	$0014
Name (16 Bytes):	Name einer Prozedur
Adresse (4 Bytes):	Offset des Prozedurlabels zum Modulanfang

Externes Label-Record (EXTLAB)

Das EXTLAB-Record enthält den Namen einer extern definierten Prozedur sowie die Startadresse des zugehörigen Hilfslabels relativ zum Modulanfang.

Typ (1 Byte):	$05
Länge (2 Bytes):	$0018
Name (18 Bytes):	Name einer externen Prozedur
Labeladresse (4 Bytes):	Offset des Hilfslabels zum Modulanfang
Fixupadresse (4 Bytes):	Offset zum Modulanfang, an dessen Stelle der Linker die Adreßdifferenz einträgt

Globale Varibale-Record (GLOBVAR)

Dieses Record enthält den Namen und die relative Adresse bezüglich 0 einer globalen Variable.

Typ (1 Byte):	$06
Länge (2 Bytes):	$0014
Name (16 Bytes):	Name einer globalen Variable
Adresse (4 Bytes):	Offset des Datenlabels zum Modulanfang

Externe Variable-Record (EXTVAR)

Das EXTVAR-Record enthält den Namen einer extern definierten Variable sowie die Startadresse des zugehörigen Hilfslabels relativ zum Modulanfang.

Typ (1 Byte):	$07
Länge (2 Bytes):	$0014
Name (16 Bytes):	Name einer externen Variable
Labeladresse (4 Bytes):	Offset des Hilfslabels zum Modulanfang

Lokales Label-Record (LOCLAB)

Das LOCLAB-Record dient zur Unterstützung von Debuggern, indem es die lokal definierten Labels mit den zugehörigen Adressen angibt.

Typ (1 Byte):	$08
Länge (2 Bytes):	$0014
Name (16 Bytes):	Name eines lokalen Labels
Adresse (4 Bytes):	Offset des Labels zum Modulanfang

Lokal Variable-Record (LOCVAR)

Das LOCVAR-Record dient zur Unterstützung von Debuggern, indem es die lokal definierten Variablen mit den zugehörigen Adressen angibt.

Typ (1 Byte):	$09
Länge (2 Bytes):	$0014
Name (16 Bytes):	Name einer lokalen Variable
Adresse (4 Bytes):	Offset des Labels zum Modulanfang

Zeilennummern-Record (LINNUM)

Das LINNUM-Record dient zur Unterstützung von Debuggern, indem es die Zeilennummern des zugehörigen Quellfiles in Relation zur Programmadresse angibt.

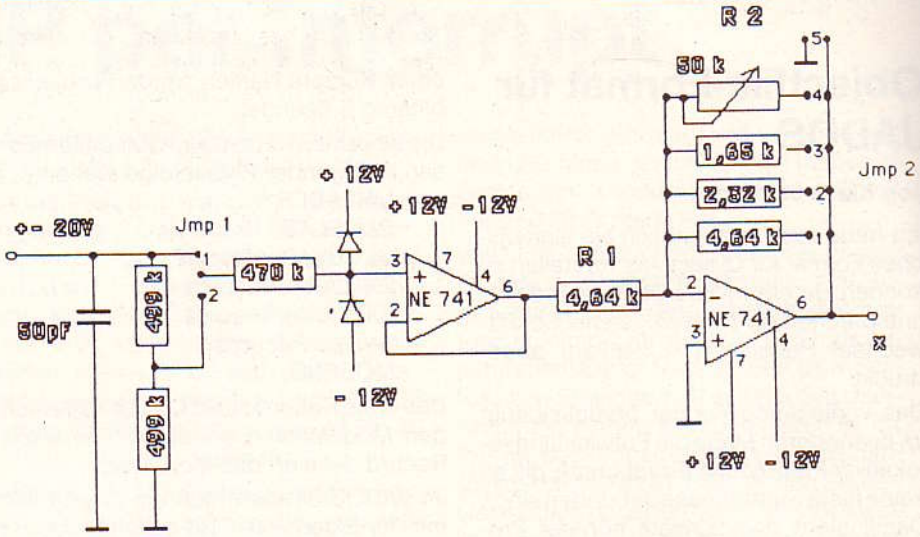
Type (1 Byte):	$0A
Länge (2 Bytes):	$0006
Zeilennummer (2 Bytes):	
Adresse (4 Bytes):	Programmzähler-adresse relativ zur Adresse 0

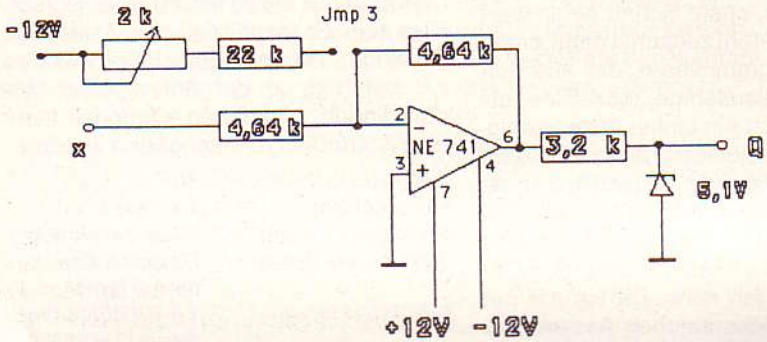
Kommentar-Record (COMMENT)

Hiermit können Kommentare in das Object-File eingebaut werden.

Typ (1 Byte):	$0B
Länge (2 Byte):	$0050
Kommentar (80 Bytes):	Kommentartext

Aus der Technik





Messen von Spannungen am AD8 x 16

von Nikolaus Bischof, GES GmbH

Manchmal ist es doch ärgerlich, wenn man mit dem preiswerten AD-Wandler AD8 x 16 Spannungen messen will, die sich nicht im Meßbereich von 0 bis 5 V bewegen. Handelt es sich um positive Spannungen, so kann man sich unter Umständen noch mit Spannungsteilern weiterhelfen, sofern die Quelle über eine bestimmte Leistung verfügt. Wenn es sich aber um negative Spannungen handelt, ist schon etwas mehr Schaltungsaufwand notwendig. Hier nun eine Schaltung, die Sie den Eingängen vorschalten können und mit der dann Spannungen von ± 20 V bis ± 200 mV gemessen werden können.

Beschreibung der Schaltung

Der Eingang (max. ± 20 V) hat einen definierten Eingangswiderstand von 1 MOhm und 60 pF. An JMP1 kann eingestellt werden, ob das Signal über einen Vorteiler 1 : 2 laufen solle (Stellung 1: Signal 1 : 1; Stellung 2: Signal 1 : 2). Der 470K Widerstand und die beiden Dioden gegen + 12 V und – 12 V dienen zum Schutz der Schaltung. Wenn Sie also aus Versehen eine Spannung von >20 V anlegen, kann das die Schaltung nicht zerstören. Der folgende Operationsverstärker ist als Impedanzwandler geschaltet, der dazu benötigt wird, um den nachfolgenden Verstärker oder Abschwächer anzusteuern. Dieser folgende Verstärker kann sowohl als Verstärker als auch als Abschwächer arbeiten. Dabei verhält sich die Verstärkung nach folgender Bedingung:

$$R1/R2 = -U1/U2$$

Sind beide Widerstände gleich, wird das Signal 1 : –1 übertragen (Jumperstellung JMP2/1). Wird Jumperstellung 2 gewählt, wird das Signal 1 : –2 abgeschwächt. Jumperstellung 4 dient dazu, mit Hilfe des Potentiometers die Verstärkung einzustellen. Sie können hier auch einen oder mehrere Widerstände einsetzen, um beliebige Verstärkungen einstellen zu können; z. B. Verstärkung –5 entspricht einem Widerstand von 4,64K x 5 = 23,2k usw. Am Ausgang dieses Verstärkers dürfen maximal ± 2,5 V oder 0 bis 5 V anliegen.

Die Schaltung mit dem letzten Operationsverstärker sorgt dafür, daß am Eingang des AD-Wandlers 0 bis + 5 V anliegen. Dazu muß einmal die Invertierung des vorigen Verstärkers wieder aufgehoben werden und außerdem bei bipolarem Betrieb eine Offsetspannung von + 2,5 V dazuaddiert werden. Das Potentiometer von 2k dient dazu, den Offset von –2,5 V einzustellen. Der Abgleich erfolgt dadurch, daß JMP2/5 und JMP3 gebrückt wird und das Potentiometer abgeglichen wird, bis am Ausgang Q + 2,5 V anliegen. Anschließend sollen Sie JMP2/5 wieder öffnen.

Die Zenerdiode 5,1 V dient wieder nur als Schutzdiode. Den Ausgang Q können Sie direkt mit einem der Eingänge des ADC0816 verbinden.

Einstellungen:

± 20 V	JMP1/2	JMP2/3	JMP3
± 10 V	JMP1/1	JMP2/3	JMP3
± 5 V	JMP1/2	JMP2/1	JMP3
± 2,5 V	JMP1/1	JMP2/1	JMP3
± < 2,5 V	JMP1/1	JMP2/4 *	JMP3
0 bis + 20 V	JMP1/2	JMP2/2	–
0 bis + 10 V	JMP1/1	JMP2/2	–
0 bis + 5 v	JMP1/1	JMP2/1	–
0 bis < + 5 V	JMP1/1	JMP2/4 *	–

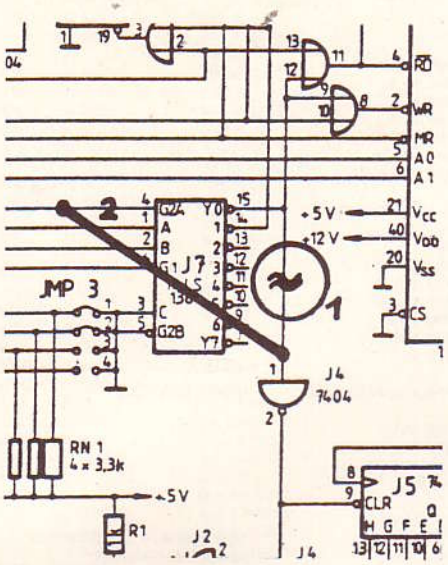
* = Abgleich des 50k Potentiometers auf den entsprechenden Spannungsbereich

– = Jumper offen

Die Schaltung läßt sich problemlos auf dem Lochrasterfeld der AD8 x 16 aufbauen. Der Preis für die Bauelemente liegt bei günstigem Einkauf unter 7,– DM.

CPU68K mit 0 WAIT-STATES

Die CPU68K (Prozessor 68008) bringt gegenüber dem Z80 doch ein Plus an Geschwindigkeit. Dieser Geschwindigkeitsgewinn wird meistens durch Einfügen von WAIT-Zyklen wieder heruntergesetzt wobei in der Regel nur die IO-Zugriffe zu-



sätzliche WAIT-Zyklen benötigen. Sollten Sie die FLO3 besitzen, können Sie durch eine einfache Änderung die FLO3 so umbauen, daß diese bei jedem IO-Zugriff entsprechende WAIT-Zyklen einfügt. Dann können Sie auf der FLO3 die WAIT-Zyklen (JMP2) für die IOs einstellen und auf der CPU den WAIT-Stecker ziehen. Diese Änderung funktioniert allerdings nur bei CPUs, die auch einen externen WAIT erkennen. Die CPU68K r2 hat diese Möglichkeit nicht, kann aber problemlos nachgerüstet werden (siehe Abb.).

Abb. 1: Änderung auf der FLO3 für externen IO-WAIT.

Abb. 2: Änderung der CPU68K r2 für externen WAIT.

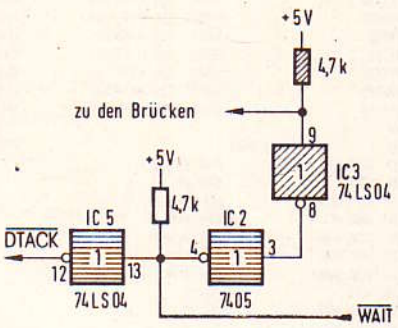


Bild 8. Das ist die Zusatzschaltung (die schon auf der Platine im Layout vorbereitet ist), mit der die CPU auf WAIT reagieren kann

Erweiterung der EPROM-Programmierkarte

für 27256er EPROMs!

von Hans Heinrich May, Möllsteiner Straße 8, 6551 Siefersheim

Um für meine erweiterte ROA64-Karte oder die neue ROA256-Karte EPROMs der Serie 27256 zu programmieren, muß die EPROM-Karte erweitert werden. Wie das geht, beschreibt der folgende Artikel.

Um den Umbau zu vereinfachen, habe ich die Programmierung der „kleinen" EPROMs nicht mehr vorgesehen. So kann man mit dieser Erweiterung „nur" noch die EPROMs 2764, 27128 und 27256 programmieren. Dies halte ich nicht für nachteilig, da die 2764er so billig sind, daß der verschenkte Speicherplatz bei kleineren Steuerungen nicht ins Gewicht fällt. So, nun zur Hardware:

Um beim Einbau des Rechners auf den umständlichen Dil-Stecker verzichten zu können, habe ich einen 9pol. D-Min-Stecker als Codier-Stecker verwendet. Die hier angegebenen IC-Nummern beziehen sich auf die Schaltungsbeschreibung in dem Buch „Die Prozessoren 68000 und 68008" von Rolf-Dieter Klein. Für Leute, die dieses Buch nicht haben, folgende Zuordnung: IC 9 = LT3; IC 2 = MV1; IC 5 = B2; IC 3 = DK2; im mc-Heft „Schaltpläne und Unterlagen".

Es werden zwei neue ICs gebraucht: IC 4' = 4077; IC 9' = 74LS273.

Die Pins 26 und 28 des EPROM-Sockels müssen von Vcc 5V und untereinander getrennt werden.

Um die Adressen A13 und A14 auf den EPROMMER zu bringen, habe ich die Pins 6, 12, 9 des IC 9 von der übrigen Steuerung getrennt. Damit liegen auf dem IC 9 nur noch Adressen. Für die Steuerleitungen habe ich auf das IC 9 das IC 9' huckepack aufgelötet. Hierzu werden die Pins 19, 2, 16, 5, 15, 6, 12, 9 und 11 hochgebogen. Alle anderen Pins werden mit denen von IC 9 verlötet. Außerdem muß auf das IC 4 das IC 4' gelötet werden. Hier sind die Pins 3, 4, 10, 11 und 12 hochzubiegen. Bei IC 1 sollen die Pins 10 und 13 hochgebogen werden. Die Transistoren T2 und T3 können entfallen.

Das waren die Änderungen und nun zur Neuverdrahtung:

Pin 6	IC 9	Pin 26	EPROM-Sockel	
Pin 12	IC 9	Pin 6	Codier-Stecker	
Pin 19	IC 9'	Pin 5	IC 2 Pin 1	IC 1
Pin 2	IC 9'	Pin 22	EPROM-Sockel	
Pin 16	IC 9'	Pin 4,5	IC 4	
Pin 12	IC 9'	Pin 11	IC 1	
Pin 9	IC 9'	Pin 3	IC 1	
Pin 11	IC 9'	Pin 12	IC 3	
Pin 17	IC 5	Pin 9	Codier-Stecker	
Pin 11	IC 1	Pin 13	IC 1 Katode LED	
Pin 6	IC 4	Pin 12	IC 4'	
Pin 1	IC 2	Pin 5	Codier-Stecker	
Pin 11	IC 4'	Pin 1	Codier-Stecker	
Pin 4	IC 1	Pin 6	Reed-Relais	
Pin 8	Reed-Relais	Pin 28	EPROM-Sockel	
Pin 1,2	Reed-relais	Vcc 5V		
Pin 14	Reed-Relais	Vcc 6V		
Pin 2	Codier-Stecker	Vcc 5V		
Pin 3	Codier-Stecker	Pin 20	EPROM-Sockel	
Pin 4	Codier-Stecker	Pin 27	EPROM-Sockel	

Das waren die neu zu verdrahtenden Anschlüsse. Nun zur Verdrahtung des Codier-Steckers:

Für die Eproms 2764 und 27128

Pin 4	Pin 5
Pin 3	Pin 7

Für den Eprom 27256

Pin 1	Pin 3
Pin 4	Pin 6
Pin 2	Pin 9

Die Belegung der Pins des Codier-Steckers:

Pin 1	CE/nicht von EXNOR-IC
Pin 2	Vcc 5V
Pin 3	CE/nicht
Pin 4	Progr/A14
Pin 5	Progr
Pin 6	A14
Pin 7	GND
Pin 8	Frei
Pin 9	Kennung 27256

Die Adressen A10, A11, und A12 können auf dem DIL-Stecker fest verdrahtet werden. Soweit die Hardware-Änderungen.

Durch die Hardware-Änderungen muß auch die Software geändert werden. Die im Grundprogramm enthaltenen Menues „EPROM prog. und EPROM lesen" laufen jetzt nicht mehr. Dies habe ich in meinem neuen Programm berücksichtigt. Die Programmierroutine ist in Anlehnung an das Programm „Quickprom 68" von Rüdiger Becker aus der mc 11/1985, von Seite 84 bis 87 geschrieben. Das Programm ist voll verschiebbar und kann auf Wunsch auf Diskette oder als Eprom von mir bezogen werden. – Viel Spaß mit dem tollen NDR-Computer!

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

0E9C00          * PROMMER FUER 27256'er EPROMS
0E9C00          * COPYRIGHT (C) 1987 BY :
0E9C00          * HEINER MAY  WOELLSTEINER STRASSE 8  6551 SIEFERSHEIM
0E9C00          * NACH ARTIKEL IN DER MC NR.:11 / 1985 SEITE 84 - 87
0E9C00          * VON RUEDIGER BECKER
0E9C00
0E9C00          KOPF:
```

```
0E9C00  55AA0180          DC.L #55AA0180
0E9C04  50524F4D4D4552    DC.B 'PROMMER '
0E9C0B  20
0E9C0C  00000020          DC.L GETPAR-KOPF
0E9C10  00000620          DC.L ENDEA-KOPF
0E9C14  01                DC.B 1
0E9C15  00 00 00          DC.B 0,0,0
0E9C1B  00000000          DC.L 0,0
0E9C1C  00000000
0E9C20                    DS 0
0E9C20
```

```
= 000000F4                BUFFER  EQU $F4
= 00000110                BUFFER1 EQU $110
= FFFFFFF80               IPPORT  EQU $FFFFFF80        ;DATEN VOM PROMMER
= FFFFFFF80               OPPORT  EQU $FFFFFF80        ;DATEN ZUM PROMMER
= FFFFFFF81               BUSYP   EQU $FFFFFF81        ;BUSY VOM PROMMER
= FFFFFFF81               KENN    EQU $FFFFFF81        ;KENNUNG 27256
= FFFFFFF81               ADRP1   EQU $FFFFFF81        ;ADRESSE ZUM PROMMER L-WORT
= FFFFFFF82               ADRP2   EQU $FFFFFF82        ;ADRESSE ZUM PROMMER H-WORT
= FFFFFFF83               STEUP   EQU $FFFFFF83        ;STEUERPORT
0E9C20
0E9C20              GETPAR:                            ;PROGRAMMSTART HAUPTPROGRAMM
0E9C20  3E3C 0011         MOVE #!CLPG,D7
0E9C24  4E41              TRAP #1
0E9C26  41FA 048A         LEA TEXT1(PC),A0
0E9C2A  303C 0032         MOVE #$32,D0
0E9C2E  323C 000A         MOVE #10,D1
0E9C32  343C 00C8         MOVE #200,D2
0E9C36  3E3C 000A         MOVE #!WRITE,D7
0E9C3A  4E41              TRAP #1
0E9C3C  41FA 04BB         LEA TEXT2(PC),A0
0E9C40  303C 0031         MOVE #$31,D0
0E9C44  0442 0032         SUB #50,D2
0E9C48  3E3C 000A         MOVE #!WRITE,D7
0E9C4C  4E41              TRAP #1
0E9C4E  41FA 0494         LEA TEXT3(PC),A0
0E9C52  303C 0021         MOVE #$21,D0
0E9C56  0442 001E         SUB #30,D2
0E9C5A  3E3C 000A         MOVE #!WRITE,D7
0E9C5E  4E41              TRAP #1
0E9C60              WAHL:
0E9C60  3E3C 000C         MOVE #!CI,D7
0E9C64  4E41              TRAP #1
0E9C66  0C00 004C         CMP.B #'L',D0
0E9C6A  6700 030A         BEQ GETPARL                ;SPRINGE ZUM LESEN
0E9C6E  0C00 0050         CMP.B #'P',D0
0E9C72  6700 0160         BEQ GETPARP                ;SPRINGE ZUM PROGRAMMIEREN
0E9C76  0C00 004D         CMP.B #'M',D0
0E9C7A  6700 0434         BEQ SCHLUSS                ;ZURUECK ZUM GRUNDPROGRAMM
0E9C7E  6000 FFE0         BRA WAHL
0E9C82              PRGLP:                             ;PROGRAMMIERSCHLEIFE
0E9C82  0C10 00FF         CMP.B #$FF,(A0)            ;VERGLEICH OB $FF
0E9C86  6700 0090         BEQ NEXTBYTE               ;DANN WEITER
0E9C8A  1010              MOVE.B (A0),D0             ;BYTE NACH D0 ZUM TESTEN
0E9C8C  13C0 FFFFFF80     MOVE.B D0,OPPORT           ;BYTE AN PROMMER
0E9C92  4200              CLR.B D0
```

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

```
0E9C94  13C5 FFFFFF81     MOVE.B D5,ADRP1            ;ADRESSE 0 - 7
0E9C9A  E05D              ROR #8,D5                  ;RECHTSROTIEREN
0E9C9C  13C5 FFFFFF82     MOVE.B D5,ADRP2            ;ADRESSE 8 - 14
0E9CA2  E15D              ROL #8,D5                  ;LINKSROTIEREN
0E9CA4  0204 00C0         AND.B #%11000000,D4        ;AUSMASKIEREN
0E9CA8  0004 0001         OR.B #%00000001,D4         ;OE/NICHT AUF 1
0E9CAC  13C4 FFFFFF83     MOVE.B D4,STEUP            ;AN STEUERPORT
0E9CB2  4E71              NOP                        ;KOENNEN
0E9CB4  4E71              NOP                        ;VIELLEICHT
0E9CB6  4E71              NOP                        ;ENTFALLEN
0E9CB8  0004 0002         OR.B #%00000010,D4         ;UND PROGRAMIERIEPULS
0E9CBC  13C4 FFFFFF83     MOVE.B D4,STEUP
0E9CC2  0204 00C1         AND.B #%11000001,D4        ;TRIGGER RUECKSETZEN
0E9CC6  13C4 FFFFFF83     MOVE.B D4,STEUP
0E9CCC              TRREADY:                           ;WARTE,BIS PRM.PULS FERTIG
0E9CCC  0839 0000         BTST #0,BUSYP              ;TEST,OB PRM.PULS FERTIG
0E9CD0  FFFFFF81
0E9CD4  6600 FFF6         BNE TRREADY                ;WENN NICHT,DANN ZURUECK
0E9CD8  0204 00C0         AND.B #%11000000,D4        ;RUECKSETZEN VON OE/NICHT
0E9CDC  13C4 FFFFFF83     MOVE.B D4,STEUP
0E9CE2  4E71              NOP                        ;SIEHE
0E9CE4  4E71              NOP                        ;OBEN
0E9CE6  0004 00C4         OR.B #%11000100,D4         ;SETZEN VON CE/NICHT(NUR FUER 27256)
0E9CEA  13C4 FFFFFF83     MOVE.B D4,STEUP
0E9CF0  4E71              NOP
0E9CF2  4E71              NOP
0E9CF4  1039 FFFFFF80     MOVE.B IPPORT,D0           ;INHALT EPROM LESEN
0E9CFA  0204 00C0         AND.B #%11000000,D4        ;RUECKSETZEN CE/NICHT
0E9CFE  13C4 FFFFFF83     MOVE.B D4,STEUP
0E9D04  B010              CMP.B (A0),D0              ;RICHTIG?
0E9D06  6700 0010         BEQ NEXTBYTE               ;DANN WEITER NEXTBYT
0E9D0A  5242              ADDQ #1,D2                 ;WENN NICHT,PRM.PULSE ERHOEHEN
0E9D0C  0C02 0003         CMP.B #3,D2
0E9D10  6E00 0092         BGT ERROR                  ;EPROM DEFEKT
0E9D14  6000 FF6C         BRA PRGLP
0E9D18              NEXTBYTE:
0E9D18  524E              ADDQ #1,A6                 ;SCHLEIFENZAEHLER
0E9D1A  5245              ADDQ #1,D5                 ;ADRESSE 'NACH' ERHOEHEN
0E9D1C  D1FC 00000001     ADDA.L #1,A0               ;ADRESSE 'VON' ERHOEHEN
0E9D22  BC88              CMP.L A0,D6                ;ALLES PROGRAMMIERT?
0E9D24  6700 000B         BEQ ENDE                   ;DANN ENDE
0E9D28  4282              CLR.L D2                   ;NEIN,DANN PULSZAEHLER LOESCHEN
0E9D2A  6000 FF56         BRA PRGLP
0E9D2E              ENDE:                              ;WENN EPROM GUT
0E9D2E  260E              MOVE.L A6,D3               ;SCHLEIFENZAEHLER RETTEN
0E9D30  04B3 00000001     SUB.L #1,D3                ;AUF RICHTIGEN WERT BRINGEN
0E9D36  3E3C 0010         MOVE #!CLR,D7
0E9D3A  4E41              TRAP #1
0E9D3C  41FA 040F         LEA OKTXT1(PC),A0
0E9D40  303C 0022         MOVE #$22,D0
0E9D44  323C 000A         MOVE #10,D1
0E9D48  343C 0096         MOVE #150,D2
0E9D4C  3E3C 000A         MOVE #!WRITE,D7
0E9D50  4E41              TRAP #1
0E9D52              ANZAHL:                            ;ANZAHL DER PROG. BYTES
0E9D52  41FA 0404         LEA OKTXT1(PC),A0
0E9D56  343C 0032         MOVE #50,D2
0E9D5A  303C 0022         MOVE #$22,D0
0E9D5E  323C 000A         MOVE #10,D1
0E9D62  3E3C 000A         MOVE #!WRITE,D7
0E9D66  4E41              TRAP #1
0E9D68  41ED 0110         LEA BUFFER1(A5),A0
0E9D6C  2003              MOVE.L D3,D0
0E9D6E  3E3C 0046         MOVE #!PRINTBD,D7
```

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

```
0E9D72  4E41              TRAP #1
0E9D74  41ED 0110         LEA BUFFER1(A5),A0
0E9D78  303C 0021         MOVE #$21,D0
0E9D7C  323C 012C         MOVE #300,D1
0E9D80  343C 0032         MOVE #50,D2
0E9D84  3E3C 000A         MOVE #!WRITE,D7
0E9D88  4E41              TRAP #1
0E9D8A              ENDE1:
0E9D8A  13FC 0000         MOVE.B #%00000000,STEUP    ;PROMMER RUECKSETZEN
0E9D8E  FFFFFF83
0E9D92  3E3C 000C         MOVE #!CI,D7
0E9D96  4E41              TRAP #1
0E9D98  0C00 0057         CMP.B #'W',D0
0E9D9C  6700 0052         BEQ NEU2
0E9DA0              ENDE2:
0E9DA0  6000 FE7E         BRA GETPAR
0E9DA4              ERROR:                             ;WENN EPROM NICHT ZU PROGRAMMIEREN
0E9DA4  260E              MOVE.L A6,D3               ;RETTEN SCHLEIFENZAEHLER
0E9DA6  04B3 00000001     SUB.L #1,D3
0E9DAC  3E3C 0010         MOVE#!CLR,D7
0E9DB0  4E41              TRAP #1
0E9DB2  41FA 037B         LEA PRGMERR(PC),A0
0E9DB6  303C 0033         MOVE #$33,D0
0E9DBA  323C 000A         MOVE #10,D1
0E9DBE  343C 0064         MOVE #100,D2
0E9DC2  3E3C 000A         MOVE #!WRITE,D7
0E9DC6  13FC 0000         MOVE.B #%00000000,STEUP    ;PROMMER RUECKSETZEN
0E9DCA  FFFFFF83
0E9DCE  4E41              TRAP #1
0E9DD0  6000 FF80         BRA ANZAHL
0E9DD4              GETPARP:                           ;PROGRAMMSTART PROGRAMMIEREN
0E9DD4  3E3C 0011         MOVE #!CLPG,D7
0E9DD8  4E41              TRAP #1
0E9DDA  41FA 0391         LEA PTXTP1(PC),A0
0E9DDE  303C 0043         MOVE #$43,D0
0E9DE2  343C 00C8         MOVE #200,D2
0E9DE6  323C 000A         MOVE #10,D1
0E9DEA  3E3C 000A         MOVE #!WRITE,D7
0E9DEE  4E41              TRAP #1
0E9DF0              NEU2:                              ;AB HIER NEUES PROGRAMM
0E9DF0  0B39 0001         BTST #1,KENN               ;ABFRAGE OB 27256ER EPROM
0E9DF4  FFFFFF81
0E9DF8  6600 001B         BNE SPRUNG                 ;WENN NICHT SPRUNG
0E9DFC  41FA 03CB         LEA PTXTPB(PC),A0
0E9E00  303C 0021         MOVE #$21,D0
0E9E04  323C 000A         MOVE #10,D1
0E9E08  343C 00A0         MOVE #160,D2
0E9E0C  3E3C 000A         MOVE #!WRITE,D7
0E9E10  4E41              TRAP #1
0E9E12              SPRUNG:                            ;AB HIER ALTES PROGRAMM
0E9E12  41FA 036E         LEA PTXTP3(PC),A0          ;VON-TEXT
0E9E16  303C 0021         MOVE #$21,D0
0E9E1A  323C 000A         MOVE #10,D1
0E9E1E  343C 008C         MOVE #140,D2
0E9E22  3E3C 000A         MOVE #!WRITE,D7
0E9E26  4E41              TRAP #1
0E9E28  41FA 035F         LEA PTXTP4(PC),D7          ;BIS-TEXT
0E9E2C  0442 0014         SUB #20,D2
0E9E30  3E3C 000A         MOVE #!WRITE,D7
0E9E34  4E41              TRAP #1
0E9E36  41FA 0358         LEA PTXTP5(PC),D7          ;NACH-TEXT
0E9E3A  0442 0014         SUB #20,D2
0E9E3E  3E3C 000A         MOVE #!WRITE,D7
0E9E42  4E41              TRAP #1
```

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

```
0E9E44  41ED 00F4         LEA BUFFER(A5),A0
0E9E48  343C 008C         MOVE #140,D2
0E9E4C  323C 00B4         MOVE #180,D1
0E9E50  303C 0021         MOVE #$21,D0
0E9E54  363C 0008         MOVE #8,D3
0E9E58  3E3C 000B         MOVE #!READ,D7             ;VON-EINGABE
0E9E5C  4E41              TRAP #1
0E9E5E  41ED 00F4         LEA BUFFER(A5),A0
0E9E62  4280              CLR.L D0
0E9E64  3E3C 001D         MOVE #!WERT,D7
0E9E68  4E41              TRAP #1
0E9E6A  2240              MOVEA.L D0,A1
0E9E6C  41ED 00F4         LEA BUFFER(A5),A0
0E9E70  343C 0078         MOVE #120,D2
0E9E74  323C 00B4         MOVE #180,D1
0E9E78  303C 0021         MOVE #$21,D0
0E9E7C  363C 0008         MOVE #8,D3
0E9E80  3E3C 000B         MOVE #!READ,D7             ;BIS-EINGABE
0E9E84  4E41              TRAP #1
0E9E86  41ED 00F4         LEA BUFFER(A5),A0
0E9E8A  4280              CLR.L D0
0E9E8C  3E3C 001D         MOVE #!WERT,D7
0E9E90  4E41              TRAP #1
0E9E92  2C00              MOVE.L D0,D6
0E9E94  41ED 00F4         LEA BUFFER(A5),A0
0E9E98  343C 0064         MOVE #100,D2
0E9E9C  323C 00B4         MOVE #180,D1
0E9EA0  303C 0021         MOVE #$21,D0
```

```
0E9EA4  363C 0008          MOVE #8,D3
0E9EA8  3E3C 000B          MOVE #'READ,D7              ;NACH-EINGABE
0E9EAC  4E41               TRAP #1
0E9EAE  41ED 00F4          LEA BUFFER(A5),A0
0E9EB2  4280               CLR.L D0
0E9EB4  3E3C 001D          MOVE #'WERT,D7
0E9EB8  4E41               TRAP #1
0E9EBA  2A00               MOVE.L D0,D5
0E9EBC  41FA 02B9          LEA PTXTP6(PC),A0
0E9EC0  343C 0032          MOVE #50,D2
0E9EC4  323C 000A          MOVE #10,D1
0E9EC8  303C 0021          MOVE #$21,D0
0E9ECC  3E3C 000A          MOVE #'WRITE,D7
0E9ED0  4E41               TRAP #1
0E9ED2            TASTE:
0E9ED2  3E3C 000C          MOVE #'CI,D7
0E9ED6  4E41               TRAP #1
0E9EDB  0C00 0046          CMP.B #'F',D0
0E9EDC  6700 FF12          BEQ NEU2
0E9EE0  0C00 0057          CMP.B #'W',D0
0E9EE4  6700 FF0A          BEQ NEU2
0E9EEB  0C00 004D          CMP.B #'M',D0
0E9EEC  6700 FEB2          BEQ ENDE2
0E9EF0  0C00 0053          CMP.B #'S',D0
0E9EF4  6600 FFDC          BNE TASTE
0E9EFB  4284               CLR.L D4                    ;LOESCHE STEUERREGISTER
0E9EFA  0004 0001          OR.B #%00000001,D4          ;OE/NICHT AUF 1
0E9EFE  13C4 FFFFFF83      MOVE.B D4,STEUP
0E9F04  0004 0040          OR.B #%01000000,D4          ;VPP AN
0E9F08  13C4 FFFFFF83      MOVE.B D4,STEUP
0E9F0E  0B39 0001          BTST #1,KENN                ;ABFRAGE OB 27256ER EPROM
0E9F12  FFFFFF81
0E9F16  6600 000C          BNE SPRUNG1                 ;WENN NICHT SPRUNG
0E9F1A  0004 00C1          OR.B #%11000001,D4          ;VCC AUF 6V (NUR 27256)
0E9F1E  13C4 FFFFFF83      MOVE.B D4,STEUP
0E9F24            SPRUNG1:
```

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

```
0E9F24  3E3C 0010          MOVE #'CLR,D7
0E9F29  4E41               TRAP #1
0E9F2A  41FA 028F          LEA PTXTP7(PC),A0
0E9F2E  303C 0032          MOVE #$32,D0
0E9F32  343C 0064          MOVE #100,D2
0E9F36  3E3C 000A          MOVE #'WRITE,D7
0E9F3A  4E41               TRAP #1
0E9F3C  2049               MOVEA.L A1,A0               ;VON-ADRESSE
0E9F3E  2606               MOVE.L D6,D3                ;BIS-ADRESSE
0E9F40  5243               ADDQ #1,D3
0E9F42  4282               CLR.L D2
0E9F44  2C7C 00000001      MOVEA.L #$1,A6              ;VORBESETZUNG DES SCHLEIFENZAEHLERS
0E9F4A  6000 FD36          BRA PRGLP                   ;PROGRAMMIERSTART
0E9F4E
0E9F4E            LESEN:
0E9F4E  13C5 FFFFFF81      MOVE.B D5,ADRP1             ;ADRESSE 0 - 7
0E9F54  E05D               ROR #8,D5                   ;RECHTSROTIEREN
0E9F56  13C5 FFFFFF82      MOVE.B D5,ADRP2             ;ADRESSE 8 - 14
0E9F5C  E15D               ROL #8,D5                   ;LINKSROTIEREN
0E9F5E  12B9 FFFFFF80      MOVE.B IPPORT,(A1)          ;INHALT EPROM LESEN
0E9F64  5245               ADDQ #1,D5                  ;ADRESSE 'VON' ERHOEHEN
0E9F66  D3FC 00000001      ADDA.L #1,A1                ;ADRESSE 'NACH' ERHOEHEN
0E9F6C  BC85               CMP.L D5,D6                 ;ALLES PROGRAMMIERT?
0E9F6E  6700 FE30          BEQ ENDE2                   ;DANN ENDE
0E9F72  6000 FFDA          BRA LESEN
0E9F76            GETPARL:                             ;PROGRAMMSTART LESEN
0E9F76  3E3C 0011          MOVE #'CLPG,D7
0E9F7A  4E41               TRAP #1
0E9F7C  41FA 0273          LEA PTXTL1(PC),A0
0E9F80  303C 0043          MOVE #$43,D0
0E9F84  343C 00C8          MOVE #200,D2
0E9F88  323C 000A          MOVE #10,D1
0E9F8C  3E3C 000A          MOVE #'WRITE,D7
0E9F90  4E41               TRAP #1
0E9F92  41FA 0270          LEA PTXTL3(PC),A0           ;VON-TEXT
0E9F96  303C 0021          MOVE #$21,D0
0E9F9A  323C 000A          MOVE #10,D1
0E9F9E  343C 008C          MOVE #140,D2
0E9FA2  3E3C 000A          MOVE #'WRITE,D7
0E9FA6  4E41               TRAP #1
0E9FA8  41FA 025E          LEA PTXTL4(PC),A0           ;BIS-TEXT
0E9FAC  0442 0014          SUB #20,D2
0E9FB0  3E3C 000A          MOVE #'WRITE,D7
0E9FB4  4E41               TRAP #1
0E9FB6  41FA 0257          LEA PTXTL5(PC),D7           ;NACH-TEXT
0E9FBA  0442 0014          SUB #20,D2
0E9FBE  3E3C 000A          MOVE #'WRITE,D7
0E9FC2  4E41               TRAP #1
0E9FC4  0B39 0001          BTST #1,KENN
0E9FC8  FFFFFF81
0E9FCC  6600 0010          BNE WEITER
0E9FD0  41FA 013D          LEA TEXT4(PC),A0
0E9FD4  0442 0028          SUB #40,D2
0E9FD8  3E3C 000A          MOVE #'WRITE,D7
0E9FDC  4E41               TRAP #1
0E9FDE            WEITER:
0E9FDE  41ED 00F4          LEA BUFFER(A5),A0
0E9FE2  343C 008C          MOVE #140,D2
0E9FE6  323C 00B4          MOVE #180,D1
0E9FEA  303C 0021          MOVE #$21,D0
0E9FEE  363C 0008          MOVE #8,D3
0E9FF2  3E3C 000B          MOVE #'READ,D7              ;VON-EINGABE
0E9FF6  4E41               TRAP #1
0E9FFB  41ED 00F4          LEA BUFFER(A5),A0
```

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 6

```
0E9FFC  42B0               CLR.L D0
0E9FFE  3E3C 001D          MOVE #'WERT,D7
```

```
0EA002  4E41               TRAP #1
0EA004  2640               MOVEA.L D0,A3
0EA006  41ED 00F4          LEA BUFFER(A5),A0
0EA00A  343C 0078          MOVE #120,D2
0EA00E  323C 00B4          MOVE #180,D1
0EA012  303C 0021          MOVE #$21,D0
0EA016  363C 000B          MOVE #8,D3
0EA01A  3E3C 000B          MOVE #'READ,D7              ;BIS-EINGABE
0EA01E  4E41               TRAP #1
0EA020  41ED 00F4          LEA BUFFER(A5),A0
0EA024  4280               CLR.L D0
0EA026  3E3C 001D          MOVE #'WERT,D7
0EA02A  4E41               TRAP #1
0EA02C  2C00               MOVE.L D0,D6
0EA02E  41ED 00F4          LEA BUFFER(A5),A0
0EA032  343C 0064          MOVE #100,D2
0EA036  323C 00B4          MOVE #180,D1
0EA03A  303C 0021          MOVE #$21,D0
0EA03E  363C 0008          MOVE #8,D3
0EA042  3E3C 000B          MOVE #'READ,D7              ;NACH-EINGABE
0EA046  4E41               TRAP #1
0EA04B  41ED 00F4          LEA BUFFER(A5),A0
0EA04C  4280               CLR.L D0
0EA04E  3E3C 001D          MOVE #'WERT,D7
0EA052  4E41               TRAP #1
0EA054  2240               MOVEA.L D0,A1
0EA056  2A0B               MOVE.L A3,D5
0EA05B  41FA 01BC          LEA PTXTL6(PC),A0
0EA05C  343C 0032          MOVE #50,D2
0EA060  323C 000A          MOVE #10,D1
0EA064  303C 0021          MOVE #$21,D0
0EA068  3E3C 000A          MOVE #'WRITE,D7
0EA06C  4E41               TRAP #1
0EA06E            TASTEL:
0EA06E  3E3C 000C          MOVE #'CI,D7
0EA072  4E41               TRAP #1
0EA074  0C00 0053          CMP.B #'S',D0
0EA078  6600 FFF4          BNE TASTEL
0EA07C  4284               CLR.L D4                    ;LOESCHE STEUERREGISTER
0EA07E  0B39 0001          BTST #1,KENN
0EA082  FFFFFF81
0EA086  6600 0022          BNE EPR2764
0EA08A  0004 0004          OR.B #%00000100,D4          ;CE/NICHT AUF 0 (27256)
0EA08E  13C4 FFFFFF83      MOVE.B D4,STEUP
0EA094  41FA 0079          LEA TEXT4(PC),A0
0EA098  303C 0021          MOVE #$21,D0
0EA09C  323C 000A          MOVE #10,D1
0EA0A0  343C 003C          MOVE #60,D2
0EA0A4  3E3C 000A          MOVE #'WRITE,D7
0EA0AB  4E41               TRAP #1
0EA0AA            EPR2764:
0EA0AA  5246               ADDQ #1,D6
0EA0AC  6000 FEA0          BRA LESEN
0EA0B0            SCHLUSS:
0EA0B0  4E75               RTS
0EA0B2
0EA0B2            TEXT1:
0EA0B2  50726F6D6D6572     DC.B 'Prommer von Heiner May',0
0EA0B9  20766F6E204B65
0EA0C0  696E6572204D61
0EA0C7  7900
0EA0C9            TEXT2:
```

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 7

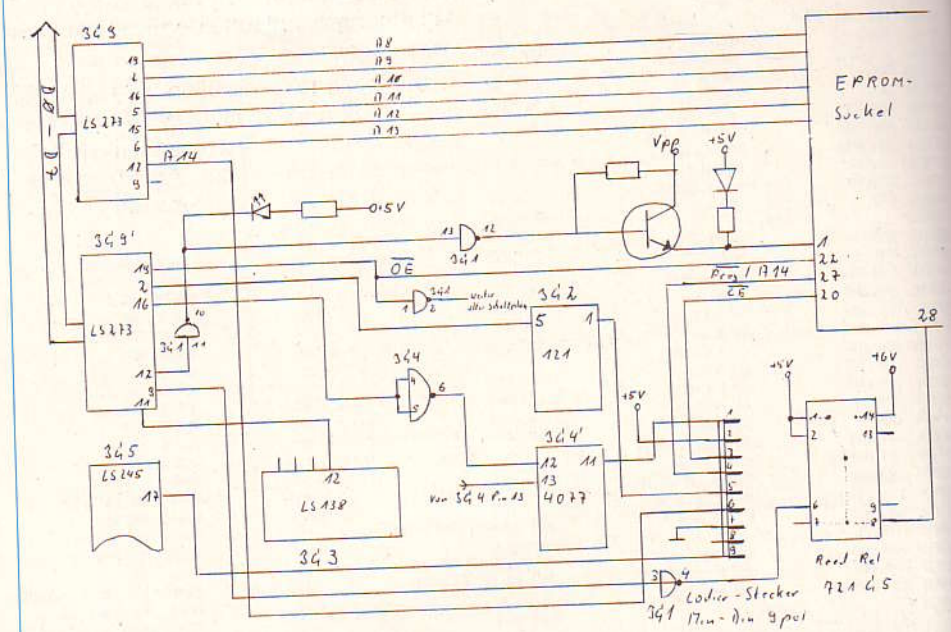
```
0EA0C9  46756572203237     DC.B 'Fuer 2764 und 27256 EPROMs',0
0EA0D0  36342075 6E6420
0EA0D7  3237323536 2045
0EA0DE  50524F4D7300
0EA0E4            TEXT3:
0EA0E4  4C657365 6E203D    DC.B 'Lesen = L    Programmieren = P   Menue = M',0
0EA0EB  204C2020202050
0EA0F2  726F6772616D6D
0EA0F9  696572656E203D
0EA100  205020204D65
0EA107  6E756520204D
0EA10E  00
0EA10F            TEXT4:
0EA10F  457320776972 64    DC.B 'Es wird ein 27256 EPROM gelesen',0
0EA116  2065696E203237
0EA11D  323536204550 52
0EA124  4F4D2067656C65
0EA12B  73656E00
0EA12F            PRGMERR:
0EA12F  4570726F6D206E     DC.B 'Eprom nicht zu programmieren!',0
0EA136  696374207A75
0EA13D  2070726F677261
0EA144  6D6D696572656E
0EA14B  2100
0EA14D            OKTXT:
0EA14D  4570726F6D2067     DC.B 'Eprom gut!',0
0EA154  75742100
0EA158            OKTXT1:
0EA158  50726F6772616D     DC.B 'Programmierte Byte :',0
0EA15F  6D696572746520
0EA166  4279747320 3A00
0EA16D            PTXTP1:
0EA16D  50726F6D6D6572     DC.B 'Prommer 2764 + 27256',0
0EA174  203237362420 2B
0EA17B  20323732353600
0EA182            PTXTP3:
0EA182  766F6E20202000     DC.B 'von   ',0
0EA189            PTXTP4:
0EA189  62697320202000     DC.B 'bis   ',0
0EA190            PTXTP5:
0EA190  6E616368202000     DC.B 'nach  ',0
0EA197            PTXTP6:
```

```
0EA197  533A5374617274  DC.B 'S:Start  F:Fehler  W:weiter M:Menue',0
0EA19E  2020463A466568
0EA1A5  6C65722020573A
0EA1AC  77656974657220
0EA1B3  4D3A4D656E7565
0EA1BA  00
0EA1BB                  PTXTP7:
0EA1BB  42697474652077          DC.B 'Bitte warten!',0
0EA1C2  617274656E2100
0EA1C9                  PTXTP8:
0EA1C9  457320736F6C6C  DC.B 'Es soll ein 27256 EPROM progr. werden ',0
0EA1D0  2065696E203237
0EA1D7  32353620455052
0EA1DE  4F4D2070726F67
0EA1E5  722E2077657264
0EA1EC  656E202100
0EA1F1                  PTXTL1:
0EA1F1  4C6573656E2032          DC.B 'Lesen 2764 + 27256',0
0EA1F8  373634202B2032
0EA1FF  3732353600
0EA204                  PTXTL3:
0EA204  766F6E00                DC.B 'von',0

Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984, Seite 8

0EA208                  PTXTL4:
0EA208  62697320202000          DC.B 'bis   ',0
0EA20F                  PTXTL5:
0EA20F  6E61636B202000          DC.B 'nach  ',0
0EA216                  PTXTL6:
0EA216  533A5374617274          DC.B 'S:Start ',0
0EA21D  2000
0EA21F  00                      DS 0
0EA220                  ENDEA:
0EA220                                  END.

0E8E86  Ende-Symboltabelle
00CC72  Ende-Debug-Tabelle
```



Achtung: Fehler auf der ROA256/1M

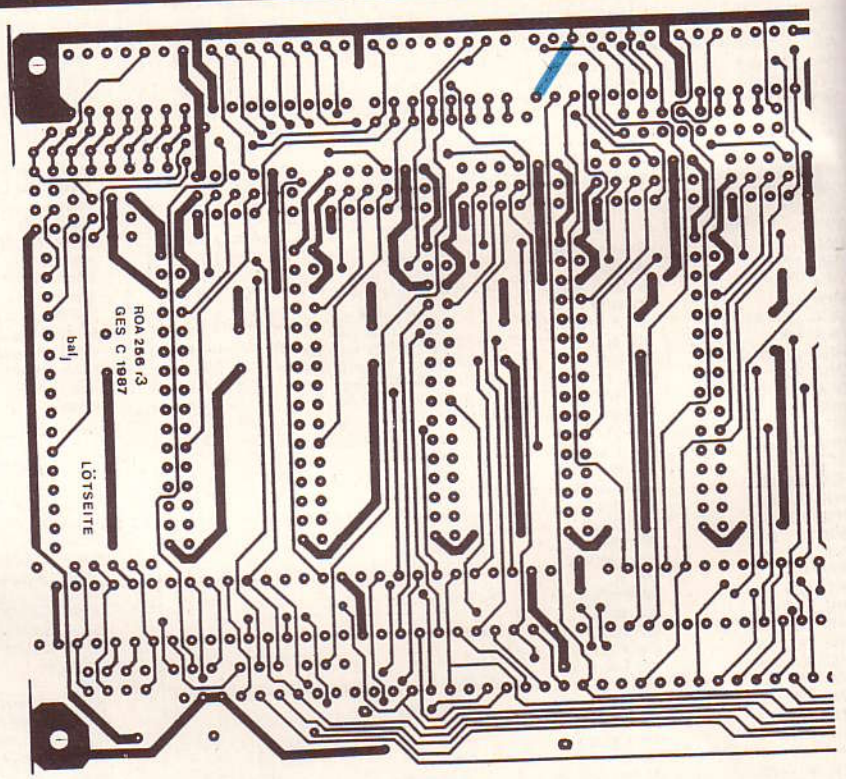
Leider hat sich noch nachträglich ein Fehler bei der Leiterplatte der ROA256/1M r3 herausgestellt. Dieser Fehler stört die Funktion der Baugruppe so gut wie nie, sollte aber doch aus Gründen der Störsicherheit behoben werden. Der Fehler macht sich erst bemerkbar (wenn er überhaupt auftreten sollte), wenn Sie die WAIT-Logik der ROA256/1M benutzen. Bei der Revision r4 ist dieser Fehler bereits behoben.

Fehlerbeschreibung:

Der Baustein 74 LS 38 (J3) hat keine Masseverbindung an Pin 7.

Fehlerbehebung:

Stellen Sie mit Hilfe von Kupferlackdraht oder isolierter Schaltlitze die Verbindung her (J3 Pin 7 ⟵⟶ J3 Pin 10); siehe Abbildung.



Briefe - Kontakte - Kleinanzeigen

Sehr geehrte Herren,

ich interessiere mich für Messen, Steuern und Regeln. Wenn ich z.B. eine Temperatur messen und diese über einen längeren Zeitraum aufzeichnen und als Kurve oder Diagramm darstellen möchte, was brauche ich dazu? 1. Als Hardware. 2. Gibt es Software oder Anleitungen für den Anfänger? Ich haben den NKC mit ZEAT und Grundprogramm, Basic und SPS auf der ROA64 in Betrieb. Bitte schreiben Sie mir, ob es irgendwelche Bücher, Zeitschriften etc. für den Betrieb eines Plotters am NDR-Computer git. Ich baue mir einen Plotter selbst, was für mich „mechanisch" keine Schwierigkeit ist; ich bin Werkzeugmacher. Aber ich habe von der Software-Seite keinerlei Ahnung. Ich bin also dankbar für jeden kleinen Bericht oder über Handbücher mit Tips zum Programmieren für Anfänger. Was brauche ich noch an Hardware? Vielen Dank im voraus.

Winfried Helfmeier,
Hauptstr. 32, 4796 Salzkotten-Verne

Antwort LOOP:

Grundsätzlich besteht ein Plotter aus zwei Schrittmotoren, die für die X- sowie Y-Achse verantwortlich sind. Die Ansteuerungslogik des Computers muß zunächst einmal die Schrittmotoren des Plotters ansteuern. Dazu empfiehlt sich eine Karte, die in der Lage ist, entsprechende Leistungen zu steuern, z. B. unsere Baugruppe ROB2. Sofern die benötigten Ströme mit den Motorströmen übereinstimmen, sollte diese Ansteuerung keine Schwierigkeiten von Seiten der Hardware machen.

Etwas komplexer wird die Software. Zunächst müssen Sie sich in der Literatur mit der Ansteuerung von Schrittmotoren vertraut machen. Hierzu gibt es speziell beim Franzis-Verlag einiges an Literatur in Fachzeitschriften bzw. in Büchern. Sehen Sie sich doch einmal in einer gut sortierten Buchhandlung oder Bücherei um.

Sodann müssen Sie die X- und Y-Werte, die der Plotter ausgeben soll, in Ansteuerungsimpulse für die Motoren umwandeln. Auch dies sollte – zumindest in einer einfachen Version – keine allzugroßen Probleme machen.

Nun zu Ihren Fragen bezüglich Messen, Steuern, Regeln. Beachten Sie dazu bitte den Artikel bezüglich der Temperaturmessung mit dem Einsteigerpaket – die Messung über einen Timer ist zwar etwas ungenau, dafür aber sehr einfach zu realisieren. Besser arbeiten Sie mit einer Analog-Digitalwandlerkarte, z. B. der A/D8 x 16. Über einen temperaturabhängigen Widerstand erzeugen Sie eine Spannung, deren Verlauf dem Temperaturverlauf proportional ist.

Nun können Sie mit einem beliebigen Programm – z. B. unter Assembler in ZEAT oder mit einer Hochsprache – die Werte an der Karte einlesen und abspeichern. Danach oder gleichzeitig können Sie ein Diagramm ausgeben. Sehr sinnvoll ist hier auch der Einsatz der Echtzeituhr, damit Sie zu bestimmten Zeitpunkten speichern können.

Computer und Finanzamt
(Bezug auf *LOOP* 12)

So wie aus vielen Leserbeiträgen in anderen Computer-Zeitschriften, läßt sich aus der Zuschrift von Herrn Welsch erkennen, daß „das Finanzamt" für viele Mitbürger auch so eine Art von Black Box ist. So wie für andere ein Computer.

Jeder hat eine irgendwie geartete Vorstellung, wie es aussieht, aber niemand versteht wie es funktioniert. Anders kann ich mir nicht erklären, daß die Anerkennung der Kosten für einen Computer durch ein Finanzamt wie ein Sieg über die Bürokratie empfunden wird. Dabei sind die Regeln, nach denen die Anerkennung oder Ablehnung zu entscheiden ist, eigentlich ganz einfach.

In den allermeisten Fällen wird es wohl um eine Berücksichtigung bei den Werbungskosten zum Arbeitslohn (Einkünfte aus nichtselbständiger Arbeit) gehen. Und hier sind verschiedene Möglichkeiten denkbar. Beispiele:

1. Der Computer steht am Arbeitsplatz und wird dort beruflich eingesetzt. – Keine Frage, die Kosten sind anzuerkennen. Der Bearbeiter beim Finanzamt muß aber in die Lage versetzt werden, dies zu erkennen. Zum Beispiel durch eine unmißverständliche Bescheinigng des Arbeitgebers und eine genaue Beschreibung dessen, was mit dem Computer gemacht wird.

2. Der Computer steht zu Hause und wird in vollem Umfang (oder mehr als 90 Prozent) beruflich eingesetzt. – Die Kosten sind auch hier anzuerkennen. Hier gilt das gleiche wie zu 1. Vielleicht kommt auch der Bearbeiter des Finanzamtes zu einer Besichtigung, um sich persönlich von der beruflichen Nutzung des Computers zu überzeugen. Bevor er kommt, wird er sich anmelden und einen Termin vereinbaren.

3. Der Computer steht zu Hause und wird zu beruflichen und privaten Zwecken genutzt. Zu welchen Anteilen ist nicht exakt feststellbar, da sich dies ständig verändert. Auf jeden Fall beträgt der private Nutzungsanteil aber mehr als 10 Prozent. – Hier sagt die Rechtsprechung des Bundesfinanzhofes (= oberste Instanz bei Steuerrechtssachen) ganz klar: wenn nicht eindeutig und leicht festzustellen ist, wie groß der berufliche Nutzungsanteil ist (wer hat schon einen Betriebsstundenzähler am Computer?), dann ist gar nichts abzugsfähig. Auch kein geschätzter Kostenanteil. Die Finanzämter müssen die Entscheidung des Bundesfinanzhofes als geltendes Recht (= wie ein Gesetz) anwenden. Der Bearbeiter beim Finanzamt hat hier keinen Ermessensspielraum.

4. Der Computer ist Hobby (so wie für andere die Amateurfunkerei oder sonst etwas). Ab und zu wird auch mal ein berufliches Problem in ein Programm umgesetzt, das vielleicht sogar auch mal den staunenden Kollegen vorgeführt wird. Außerdem ist die Beschäftigung mit dem Computer dazu nütze, die Allgemeinbildung in diesem Bereich zu verbessern.

Und die Kenntnisse gehen ja meist auch sehr viel weiter. Auf jeden Fall hat der Hobby-Computerist keine Berührungsängste mehr, wenn an seinem Arbeitsplatz so ein Gerät installiert wird. – Ein Hobby liegt in der Privatsphäre eines jeden. Kosten sind nicht abzugsfähig, auch kein geschätzter Anteil. Auch dann nicht, wenn die privat erworbenen Kenntnisse im Beruf hin und wieder nützlich sind. Hier hat der Bearbeiter ebenfalls keinen Ermessenspielraum.

Ich gehe davon aus, daß sich viele Leser von *LOOP* in der letzten Gruppe wiederfinden. An sie die Frage: Glauben Sie, daß der Amateurfunker dann die Kosten für sein Hobby steuerlich geltend machen kann, wenn er sein Geld als Radio- und Fernsehtechniker verdient? Oder kennen Sie einen Möbeltischler, der seine private Drechselbank steuerlich geltend gemacht hat? Daß beide in ihren Berufen von den privat gemachten Erfahrungen profitieren, wird wohl jeder anerkennen. Aber das reicht für die Abzugsfähigkeit der Kosten für das Hobby nicht aus.

Anerkennung oder Nichtanerkennung der Kosten für Computer und Peripherie sind also nicht von Glück oder Pech abhängig.

Norbert Kappes
Am Ring 10, 5550 Bernkastel-Kues

Betrifft Microsoft Basic- und Assembler-Entwicklungspaket.

Aufgrund Ihres Artikels „Ein neuer Software-Knüller für CP/M2.2-Anwender" bestellte ich dieses Software-Paket, das auch den Basic-Interpreter MBASIC beinhaltet. Alles in allem bin ich mit dieser Software, die mir die Entwicklung von Programmen sehr komfortabel gestaltet, sehr zufrieden.

Beim Erstellen eines Programmes fiel mir vor kurzem ein, meiner Meinung nach fehlerhaftes Arbeiten einer Funktion des Basic-Interpreters auf. Bei dieser Funktion handelt es sich um die INT(x)-Funktion, die die größte Integerzahl liefert.

Folgendes war mir aufgefallen: In meinem Programm wurde die Funktion folgendermaßen benutzt: Y = INT(K-J). Die Werte für K und J wurden vorher über eine INPUT-Anweisung über Tastatur eingegeben.

Für die Werte K = 1.3 und J = 0.3 berechnet die Funktion für Y den Wert 0, also INT(1.3 – 0.3) = 0, was ja wohl falsch sein dürfte, da INT(1) im allgemeinen 1 ist. Derselbe Effekt zeigte sich bei dem Wertepaar K = 1.8 und J = 0.8.

Dieses fehlerhafte Arbeiten tritt aber nur auf, wenn die Zahlenwerte (hier für K und J) mittels Tastatur eingegeben werden. Auch im Befehls-Modus (Basic meldet sich mit Ok) mit der Eingabezeile PRINT INT(1.3 – 0.3) wird 0 für das Ergebnis berechnet.

Werden die Zahlenwerte für K und J aber nicht über Tastatur eingegeben, sondern im Programmablauf errechnet, so arbeitet die Funktion INT richtig. Um dies zu

verdeutlichen, füge ich diesem Schreiben ein Listing eines kleinen Testprogrammes bei.

Benutzt man die Funktion CINT(x) anstelle von INT(x), so kann man das oben beschriebene Problem umgehen. Es ist nur zu berücksichtigen, daß bei CINT eine Rundung der Stellen hinter dem Komma vorgenommen wird, was bei INT nicht der Fall ist.

Ich würde mich freuen, wenn Sie zu dem beschriebenen Problem Stellung nehmen könnten und es in einer der nächsten Ausgaben der Zeitschrift *LOOP* veröffentlichen würden, damit auch andere Benutzer des MBASIC-Interpreters davon in Kenntnis gesetzt werden.

Peter Kesa
Marquardstraße 30, 6400 Fulda

Antwort LOOP:

Sehr geehrter Herr Kesa,
das von Ihnen beschriebene Problem deutet auf ein Rundungsproblem hin, wie es in Basic-Interpretern üblich ist.

Bitte prüfen Sie zunächst, ob Sie die Variablen, die Sie einlesen, als normal oder doppelt genau definiert haben; danach sollten Sie prüfen, ob dieser Fehler auch dann auftritt, wenn Sie vorher einen „Roundbefehl" gegeben haben.

Generell sollte nach arithmetischen Operationen ein klares Runden stattfinden, um solche Probleme zu vermeiden. Vielleicht können Sie uns kurz vom Ergebnis informieren.

Gerne werden wir Ihren Tip in der nächsten *LOOP* veröffentlichen.

Kleinanzeigen

Zu verkaufen: Geh3, Bus3, CPU Z80, DRAM 128, ROA64, Bank-Boot, Key, FDC, SIO, EPROMPER, große Tastatur, Monitor, 2 x Floppy 5¼", Hexio, IOE2, Netzteil, EPROMs: Ehex, EFlomon, EBasic, EGrund 2000 kompl. mit Listings, NP ca. 4.000,– DM, kompl. für DM 1.800,– evtl. auch einzeln. Burt Wanzke, Gernsheimer Str. 14, 60800 Groß-Gerau, Tel.: (06152) 82671 ab 20.00 Uhr.

Public-Domain-Software für NDR-Computer (CP/M) z. B. Sprachen: FORTH 83 für Z80- und 68k-systeme; Small C, XLISP usw. je Disk DM 19,95 (incl. MWSt., Versand gegen Vorkasse oder per Nachnahme) außerdem Anwendungen, Utilities, Spiele. Bestellung oder Info bei: Roland Steyer, Software-Hardware-Entw., Kappenbergweg 2, 6200 Wiesbaden 15.

Verkaufe Paket für Einsteiger: SBC2, IOE, BUS1, POW5V, HEXIO, EHEX und Handbuch dazu für DM 390,–. Außerdem CAS und Schroff-Gehäuse. Alle Baugruppen fertig aufgebaugt und funktionsfähig; Abgabe evtl. auch einzeln. Rupert Schöttler, Biedersteiner Straße. 30A, 8000 München 40, Tel.: (089) 3614794.

Verkaufe: NDR-Klein-Computer mit SBC 3, Z80H/8MHz, 64KRam, 240KRamfloppy, 2 LW 3½", Schaltnetzteil, PC-Gehäuse, Drucker EPSON LX 86, Software: WordStar 3.0, dBASEII, CP/M2.2 u.a.; viel Literatur. Neupreis für alles: ca. 5.400,– DM jetzt nur 3.500,– DM, Alter 1 – 1½ Jahre, Tel.: (05225) 1769.

NDR 68000/08: Superprogramme für JASOS 2.1 vom JADOS-Entwickler: ✶ schnellen Editor JEDI ✶ 26 Phrasen- und 10 Dateimakros, umfangreiche Diskettenoperationen, Inhaltsverzeichnis, 2. Text ansehen, Textgröße bis 780 KByte, u.v.m. **Preishammer! 59,– DM.** DEMO gegen Disk + 5,– Briefmarken. ✶ Disassembler 50,– ✶ Reversi 39,– ✶ Diskettenmonitor nur 35,–, Preise + Porto: 3,– (Scheck) / 6,– (NN) auf Disk 3½" oder 5¼" unter JADOS. K. Janßen, Hannixweg 74, 4150 Krefeld 1.

Verkaufe: NDR-Computer betriebsbereit SBC2 + SP. Erw., POW5, BUSII + 2III, GDP64K, KEY, CAS, MON1, Basic, TAST1 + Geh., Trafo, Sonderhefte + Bücher + LOOP 0 – 15, VB 680,– DM. Andreas Krenn, Maieranger 8, 8220 Traunstein-Kammer, Tel.: (0861) 2125.

Verkaufe Speicherkarte RAM256 betriebsfertig bestückt mit 256KByte dyn. RAM für 140,– DM und Matrixdrucker SEIKOSHA GP-550A mit Centronics-Anschluß und Software für Z80 und 68008 für 280,– DM. U. Koch, Frankenstr. 25, 5880 Lüdenscheid, Tel.: (02351) 26192.

Verkaufe: SBC3 (8MHz – 16 kByte RAM), BUS3, IOE, HEXIO, HEXMON, GDP64K, Gehäuse3, LOOP 1 – 15; **alle Platinen sind mit Handbücher, fertig aufgebaut und funktionsfähig;** Verhandlungsbasis: DM 750,–; einzeln ca. ½ Bausatzpreis. Oliver Vogel, Tel. (07153) 31699 (von 17.30 Uhr – 20.30 Uhr).

NEU! Tool-Disk für NDR-Computer unter CP/M68k!! Die Programme auf dieser Diskette unterstützen die Zusammenarbeit zwischen NDR-Grundprogramm und CP/M68k. Sie wurden von einem NDR-User mit der CPU 68008 erstellt und gut dokumentiert. Diese Disk für DM 19.95 bei Ronald Steyer, Softwareversand, Kappenbergweg 2, 6200 Wiesbaden 15

Leerdisketten QUAD Density Double sided. Ideal für das 80-Track-Laufwerk des NDR-Computers. Das besondere Angebot! Und nun die Preise: Stück DM 3,50, 1 Karton (10 Stück) DM 29,–. Bestellung oder Info bei Ronald Steyer, Softwareversand, Kappenbergweg 2, 6200 Wiesbaden 15.

KONTAKTE

Die VHS Schöneberg in Berlin führt einen Kurs **„Einführung in die Mikrocomputertechnik"** durch – auf der Basis des NDR-Computers (Z80-System). Ort: **Crelle 21 e.V., Crellestraße 21, 1000 Berlin 62, an 6 Abenden und 2 Wochenenden von 19.00 – 21.00 Uhr, ab 8. Februar 1988.** Anmeldung ab 9. Januar 1988 in der VHS Schöneberg, Kyffhäuserstraße 23, 1000 Berlin 30. Kosten: DM 60,–, ermäßigt DM 30,–.

In Kiel gibt es den **Selbstbau-Computer-Club Kiel e. V.** der sich vornehmlich mit dem NDR-Computer beschäftigt. Kontaktadresse: Jost-Reimer Hoof, Wilhelm-Ivens-Weg 87, 2305 Heikendorf, Telefon: (0431) 242070

Suche NDR-Klein-Computer-User im Raum Wien. Bitte schreibt mir! Ich hoffe, daß genügend Österreicher es gewagt haben, das Ding zu bauen, damit es für einen Mini-Club reicht. Meine Adresse: Gotthold Schaffner, Argentinierstr. 36/6, 1040 Wien.

Ergänzung zu Windowverwaltung

Windowverwaltung für alle FLOMON-Versionen.

Bei der jetzigen Version von Windowverwaltung V 1.1 wird zum Betrieb das FLOMON V 3.2 benötigt.

Um Ihnen das aufwendige Übersetzen des Programmes zu ersparen, um auch mit anderen FLOMON-Versionen arbeiten zu können, bieten wir jetzt die Windowverwaltung in einer neuen Version V 1.2 an.

Darauf finden Sie die entsprechenden Programme für die jweiligen FLOMON-Versionen V 1.5, V 1.6, V 3.0, V 3.1, V 3.2, V 4.1 und V 4.2. (V 4.2 nur für Terminalbetrieb).

Service für die Schon-Besitzer: Senden Sie uns Ihre alte Version mit einem frankierten Rückumschlag – wir spielen Ihnen kostenlos die neueste Version auf!

Frage an die LOOP-Leser – Thema Kassettenrecorder

Die CAS-Schnittstelle hat mit einigen Kassettenrecordern Probleme bereitet. Diese Probleme waren mit der Einführung des Digital-Kassetten-Recorders, der einwandfrei funktionierte, behoben (siehe Katalog).

Leider wird dieser Kassettenrecorder nun nicht mehr produziert. Deshalb, um unseren neuen Kunden Empfehlungen geben zu können, die Frage an die LOOP-Leser, die mit der CAS arbeiten:

- *Welchen Kassettenrecorder (Fabrikat, Typ, Preis, wo erhältlich) verwenden Sie?*
- *Funktioniert alles? Hinweise, Tips und Tricks?*

Schon jetzt herzlichen Dank für Ihre Zuschriften!

In letzter Minute aus der Technik:

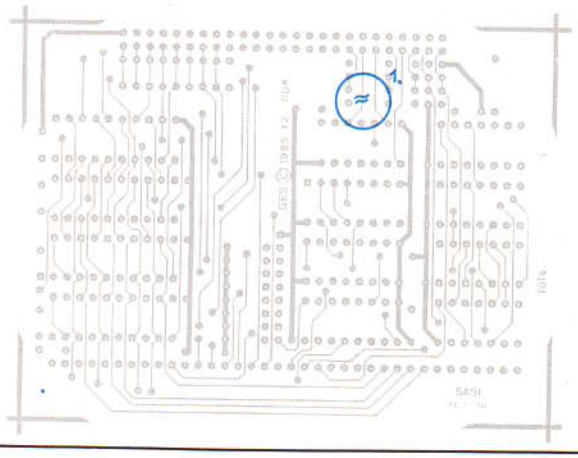
Änderung SASI – OWL

Um die OWL-Platte sicherer zu machen, sollte eine Änderung an der SASI vorgenommen werden.

Fehlerbeschreibung: Es kann vorkommen, daß die Festplatte bei häufigem Ein-/Ausschalten „abstürzt" und das Direktory löscht.

Fehlerbehebung:

1. Auf der SASI-Baugruppe wird der RESET der SASI zum 50-poligen Stecker getrennt.
2. Sollte die Festplatte nicht mehr formatierbar sein, können Sie im Notfall (wenn sonst nichts mehr hilft) die Festplatte mit dem sogenannten „FACTORY FORMAT" formatieren. Dazu müssen Sie die definierte Konstante „FORMATC" (jetzt festgesetzt auf 4) im Formatierer auf den Wert 2 abändern und dann die Festplatte neu formatieren. Dieses „FACTORY FORMAT" sollten Sie aber nur durchführen, wenn die Festplatte sonst nicht mehr formatierbar ist.



BESTELLKARTE

Ich / Wir bestelle(n) unter Anerkennung Ihrer Geschäfts- und Lieferungsbedingungen folgende Artikel:

Stück	Bestell-Nr.	Bezeichnung	Einzelpreis
	10 834	GES-Katalog	10,-
	10061	LOOP-Abo	20,-

Adresse (umseitig) nicht vergessen!

Datum

Unterschrift
Bei Minderjährigen die des gesetzl. Vertreters

BESTELLKARTE

Ich / Wir bestelle(n) unter Anerkennung Ihrer Geschäfts- und Lieferungsbedingungen folgende Artikel:

Stück	Bestell-Nr.	Bezeichnung	Einzelpreis
	10 834	GES-Katalog	10,-
	10061	LOOP-Abo	20,-

Adresse (umseitig) nicht vergessen!

Datum

Unterschrift
Bei Minderjährigen die des gesetzl. Vertreters

Neue Preise – neue Produkte!

SPS-Compiler

Best.-Nr.	Beschreibung	Preis DM
11168	SPS-Compiler, 5¼" 80 Spuren, für NDR oder mc-Computer	298,00
11169	dto., 3½", 80 Spuren	298,00
11170	SPS-Compiler, IBM-Format	298,00
	Hinweis: Die erzeugte MAC-Datei ist nur mit einem Z80-Assembler zu übersetzen; der Compiler erzeugt derzeit noch **keinen** für IBM ablauffähigen Code.	

Zubehör zum SPS-Compiler

Best.-Nr.	Beschreibung	Preis DM
11058	Microsoft-Entwicklungspaket mit M80-Assembler, Linker usw., 5¼", 80 Spuren	248,00
11059	dto., 3½", 80 Spuren	248,00
10442	Word-Star Textverarbeitung, um die SPS-Quelle zu erzeugen, 5¼", 80 Spuren **nur noch**	99,00
10668	dto., 3½", 80 Spuren	139,00
11006	CHRISTIANI-Lehrgang SPS Band 1 – 4	152,00
10349	PROMER-Fertiggerät	179,00
10348	PROMER-Bausatz	99,00
10625	IOE2-Fertiggerät	148,00
10626	IOE2-Bausatz	98,00
10289	IOE-Fertiggerät	74,00
10288	IOE-Bausatz	49,90

Buch „Mit HEXMON Programme entwickeln"

Best.-Nr.	Beschreibung	Preis DM
10286	Das Buch erläutert das im Einsteigerpaket verwendete Monitor-Programm. Alle Unterprogramme sind einzeln aufgeführt und dokumentiert. Dieses Buch ist für jeden, der eigene Programme erstellt, ein „Muß"!	20,00

MLA Makro-Link-Assembler

Best.-Nr.	Beschreibung	Preis DM
11060	Makro-Link-Assembler, 5¼", 80 Spuren	98,00
11093	dto., 3½", 80 Spuren	98,00

TDS Benutzeroberfläche unter CP/M68K

Best.-Nr.	Beschreibung	Preis DM
11174	TDS 5¼", 80 Spuren	98,00
11175	dto. 3½" 80 Spuren	98,00

HEBAS auf EPROM für SBC3

Best.-Nr.	Beschreibung	Preis DM
11103	EHEBAS-Eproms	98,00

Bitte Porto nicht vergessen

BESTELL-POSTKARTE

GRAF computer

Graf Elektronik Systeme GmbH
Postfach 1610

8960 Kempten

Anschrift:

Lieferform: ☐ Nachnahme ☐ Vorkasse
☐ Bankeinzug

Bankeinzug: Hiermit ermächtige ich die Firma GES GmbH, den Rechnungsbetrag für die auf dieser Karte angegebenen Bestellungen von meinem Konto:

BLZ _____ Konto-Nr. _____

Bank: _____

abzubuchen. Falls mein Konto die erforderliche Deckung nicht aufweist, besteht seitens des kontoführenden Kreditinstitutes keine Verpflichtung zur Einlösung.

Datum _____ Unterschrift _____

Bitte Porto nicht vergessen

BESTELL-POSTKARTE

GRAF computer

Graf Elektronik Systeme GmbH
Postfach 1610

8960 Kempten

Anschrift:

Lieferform: ☐ Nachnahme ☐ Vorkasse
☐ Bankeinzug

Bankeinzug: Hiermit ermächtige ich die Firma GES GmbH, den Rechnungsbetrag für die auf dieser Karte angegebenen Bestellungen von meinem Konto:

BLZ _____ Konto-Nr. _____

Bank: _____

abzubuchen. Falls mein Konto die erforderliche Deckung nicht aufweist, besteht seitens des kontoführenden Kreditinstitutes keine Verpflichtung zur Einlösung.

Datum _____ Unterschrift _____