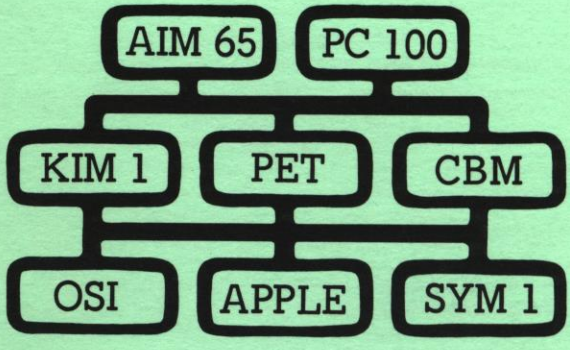
65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

DM 9,50 Nr.36 April 1984



Inhaltsverzeichnis

ASSMOS 1.0 - AIM 65-Assembler für G65SC02 3
Eprommer für 2764 und 27128 am AIM 65 14
CBM 710 Spezial (2) 24
Mapping INPUT für Commodore 64 26
ROFF (3) - Textformatierung für 6502 Mikroprozessoren 38
Datenbank DBMS L84 48
Bücher 37, 58
Aus der Branche 60
Editorial 61
ROM-Umschaltung per Software für AIM 65 62

Diese Zeitschrift wird ab Heft 37 (Juni 1984) umbenannt in:

MICRO MAG

DUO Plott Interface

für MX80 F/T und
ITOH 8510
und COMMODORE-Rechner 30/40/80

Paralleles IEEE 488 - Interface, genormter IEEE-Stecker und Kabel
kompletter Zeichensatz des CBM-Computers
zwei Geräteadressen für Groß-/Grafikmodus und Textmodus
alle Funktionen der Drucker bleiben erhalten, Floppy-Kompatibilität
Deutsche Umlaute, ß und Paragraph
Problemloser Einbau, inklusive deutsches EPSON-Handbuch
Komplettpreis einschl. Kabel, CBM-Grafiksatz und Handbuch incl. MWSt
für EPSON DM 298,--
für ITOH DM 298,--

Umrüstsatz MX80 F/T auf MX80 Typ 3

Aus Ihrem EPSON MX-80 F/T wird der MX-80 Typ 3
Einzelnadelansteuerung, erweiterter Befehlssatz, Elite-Schrift
Preis inkl. MWSt DM 98,--
Komplettpreis Interface, Kabel, Handbuch und Umrüstsatz inkl. MWSt DM 450,--

Für COMMODORE

Deutsche Tastatur mit Original-Tastensätzen (keine Aufkleber)
inkl. deutschem Zeichengenerator
für CBM 8032 DM 98,--
für CBM 8032 und 8096 DM 98,--
nur 8096, ohne Tasten DM 98,--

Speichererweiterung auf 96 K

LOS-kompatibel, inkl. MWSt DM 798,--

ISAM-Routinen
Datenhaltungsprogramm, Indexed Sequential Access Method
siehe Besprechung in Heft 23 des 65xx MICRO MAG, **DM 298,--**

Drucker:

RX 80 DM 998,-- + MWSt

ITOH 8510 DM 1495,-- + MWSt

EPSON-Druck-Computer FX 80 DM 1295,-- + MWSt

Stellberg Computer-Systeme

COMMODORE EPSON C. ITOH SOFTWARE INTERFACE

Blindenaaf 36 5063 Overath Tel.: 022 06 - 66 44

Martin Albrecht, 4350 Recklinghausen
Dipl.-Ing. Rüdiger Wollenberg, 4600 Dortmund 50

ASSMOS 1.0

AIM-65 Assembler für G65SC02

Neben dem Standard Mikroprozessor 6502 sind inzwischen zwei CMOS-Versionen auf dem Markt. Die CPU G65SC02 der Firma GTE untersützt die neuen Befehle

BRA	branch always
PHX, PHY	push X, push Y
PLX, PLY	pull X, pull Y
STZ	store zero into memory
TSB, TRB	test and set/reset memory bits with accumulator

Die Befehle INC und DEC wurden um den Akkumulator-Modus erweitert, für den Befehl BIT sind jetzt auch die Adressierungsarten ZP,X, ABS,X und immediate zugelassen und für den Sprungbefehl wurde JMP (ABS,X) neu implementiert. Neu ist auch die indirekte Adressierung ohne Indexregister. LDA ($A0) ersetzt beispielsweise die beiden Befehle

```
LDY #0
LDA ($A0),Y.
```

Der Befehlssatz der CMOS-CPU von Rockwell geht darüber hinaus. Er umfaßt noch die Funktionen BBRx, BBSx, RMBx und SMBx. Aus Platzmangel konnte eine Implementierung in dem 4 KByte Eprom nicht vorgenommen werden. Diese Befehle sind für Bitmanipulationen in der Zero Page vorgesehen und besitzen nicht die Wichtigkeit der anderen Erweiterungen. – Einzelheiten sind den firmenspezifischen Datenblättern zu entnehmen /1,2/.

Im nachfolgenden Programm sind folgende Änderungen gegenüber dem AIM-Standard wichtig:

- Kleinschreibung von Labels, Befehlen und Kommentaren sowie Assembleranweisungen
- Zulassung aller Buchstaben des deutschen Zeichensatzes
- Listing im 80 Zeichen-Format nach /3/
- Unterstützung der Floppydisk VC-1541 der Fa. Commodore bei Verwendung des Monitors DISKMON 1.0
- Wegfall der Displayausgabe beim ersten Assemblerdurchlauf
- Bei Benutzung des USER-Vektors richtiges Wiederaufsetzen (Carrybit=0) zu Beginn des zweiten Assemblerdurchlaufes

Alle Quelltexte für den ASSMOS-Assembler können in Groß- und Kleinschrift eingegeben werden. Das gilt für die Assembleranweisungen, Mnemonics, die Angabe des Indexregisters (X,Y oder x,y), die Accumulatormodes (A oder a), numerischer Konstanten und natürlich auch der Symbole. Dabei gilt zu beachten, daß START, Start und start drei verschiedene Symbole sind. - Die Assemblerprompts LIST? und OBJ? darf man ebenfalls in Kleinschrift beantworten.

Das neue Formatierprogramm gestattet eine bessere Lesbarkeit des Listings bei Benutzung eines Bildschirmes oder Druckers mit mehr als 60 Zeichen pro Zeile. Die Aufteilung des Listings ist am folgenden Beispiel zu erkennen:

```
0000          ;Beispiel für das neue Listprogramm:
0000 OUTPUT       =$e97a          ;ein Zeichen ausgeben
0000
0000              *=$54fe
54FE              .opt nog,LIS    ;hier Assembleranweisungen
54FE Waldi  A200  ldx #0
```

```
5500 Köter  BD0E55 Lda Hunger,x    ;Zeichen einlesen
5503        48     pha
5504        297F   And #$7f
5506        207AE9 jsr OUTPUT      ;und drucken
5509        E8     inx
550A        68     pla
550B        10F3   BPL Köter       ;Schleife
550D        60     rts
550E
550E Hunger 2020   .byt '  Waldi hat Hunger !',$20+$80
5522        A0
5523
5523               *=$10c          ;Anschluß an Taste F1
010C        4CFE54 jmp Waldi
010F
010F               .end
ERRORS=0000
```

Die im Assembler des AIM 65 vorgesehenen aber nicht realisierten Assembleranweisungen SYM, NOS, CNT, NOC und COU wurden entfernt; ERR und NOE wurden herausgenommen, da ihr Funktionsumfang durch die Anweisungen NOL und LIS abgedeckt wird. - Die Anweisung .PAG erzeugt jetzt eine Zeile von 60 statt von bisher 20 Bindestrichen. Die Option MEM assembliert in den Speicher, bei NOM wird kein Code abgelegt.

Der neue ASSMOS-Assembler gestattet unter DISKMON 1.0 (siehe Heft 35) auch ein Assemblieren von Quellcode, der auf der Floppy VC-1541 von Commodore abgelegt ist. Der Filename kann beliebig lang sein. Zur Unterscheidung werden bei Typefiles die ersten 5 Zeichen, bei Discfiles die ersten 16 Zeichen herangezogen. Es ist sogar möglich, mehrere verkettete Files in einem Arbeitsgang zu einem Objektfile zu assemblieren. Liegen diese Quellfiles auf Floppydisc, so sucht sich der Assembler diese Files selber. Dazu müssen folgende Konventionen beachtet werden:

```
+-->   PROG1        +-->   PROG2        +-->   PROG3
!      ..           !      ..           !      ..
!      ..           !      ..           !      ..
!      ..           !      .FIL PROG3 -+      ..
!      ..           !                         ..
!      .FIL PROG2 -+                          ..
!                                             .END PROG1 -+
!                                                         !
+-------------------------------------------------------+
```

Ist nur ein Quellfile vorhanden, so war es beim Assembler des AIM 65 nicht möglich, hinter .END einen Filenamen anzugeben. Solange nicht von der Floppy assembliert wird, ist dies auch jetzt nicht zwingend. Alte Routinen können unverändert mit dem neuen Assembler übersetzt werden. Beim Arbeiten von der Floppy muß der Filename jedoch unbedingt im END-Statement wiederholt werden. Für den zweiten Durchlauf muß der Assembler das Quellfile erneut von vorne lesen und benötigt dazu den Filenamen im .END-Statement. Ist der Filename irrtümlicherweise einmal vergessen worden, so fordert der Assembler ihn nach seinem ersten Durchlauf an.

Wird er Quelltext vom Band (Tape) eingelesen, hält der Assembler wie gewohnt nach dem ersten Durchlauf an. Er kann mit jeder Taste außer '1' wieder gestartet werden. Taste '1' dient dazu, unter DISKMON 1.0 die Motorsteuerung ein- und auszuschalten. Diese Funktion darf nicht zusammen mit dem Original-Monitor benutzt werden.

Der deutsche Zeichensatz ist auf folgende ASCII-Codes gelegt:

Ä =$5A ä =$7A Ö =$5B ö =$7B Ü =$5C ü =$7C ß =$7E

Der ASSMOS 1.0 ist aus dem Standardassembler des AIM-65 entstanden. Weil mehr als nur punktuelle Eingriffe vorgenommen wurden, ist der Quelltext aus einem Programmlisting abgetippt worden. In der Zukunft können sehr leicht Erweiterungen auf den Rockwell-Befehlssatz und die Ausgabe einer Crossreference- und Symboltafel adaptiert werden.

Literatur:
/1/ GTE Datenblatt: ISO-CMOS G65SCXXX 8-Bit Microprocessor Family GTE Microcircuits, Montenstr. 11, 8000 München 19
/2/ Erweiterter Befehlssatz: CMOS-CPU R65C02, 65xx MICRO MAG Nr. 28, Dezember 1982, S. 16 ff.
/3/ R. Wollenberg: Assembler Formatierer, 65xx MICRO MAG Nr. 32, August 1983, S. 49 ff.

```
0000          ; ASSMOS 1.0
0000          ;geänderter Assembler für CMOS CPU GTE G65SC02
0000          ;und Unterstützung der Floppy VC 1541
0000          ;Alle Änderungen sind kommentiert

0089                .OPT LIS
0089          ;SRCNAM tauscht mit PBUF den Platz
0089 SRCNAM         *=*+5
008E PBUF           *=*+30

D000                .OPT LIS
D000 NEWTOG         =$E9FA          ;Toggle Tape Funktion
D000 WHE2           =$E868          ;Teil von WHEREI
D000 WHERET         =$E850          ;teste, woher Input kommt
D000 ALLF           =$F3C2          ;Floppy File eröffnen
D000 ENDTA          =$E3C4          ;Input File schließen
D000 NORMAL         =$EBFB          ;Filenamen einrichten
D000 NAMO2          =$E8E9          ;Namen mit Blanks auffüllen
D000 LAE            =$A7            ;Länge des Filenamens
D000 CRCK2          =$EA3B          ;CR auf Display
D000 CLFP           =$E3B1          ;Input File schließen

D069                .OPT LIS
D069          A237  LDX #LSMSG-MB
D06B          20C8DF JSR PMSG
D06E          2073E9 JSR REDOUT
D071          20E8DF JSR CAPIT      ;Kleinbuchstaben -> Großbuchstaben
D074          C94E  CMP #'N
D076          D006  BNE NXTPA4

D095                .OPT LIS
D095          A23C  LDX #OBMSG-MB
D097          20C8DF JSR PMSG
D09A          2073E9 JSR REDOUT
D09D          20E8DF JSR CAPIT      ;Kleinbuchstaben -> Großbuchstaben
D0A0          C959  CMP #'Y
D0A2          D019  BNE NXTPA5

D0EF                .OPT LIS
D0EF          ;Initialisieren mit Y=0 statt bisher ACC=0
D0EF P1DUM    A000  LDY #0
D0F1          841F  STY LCDPT
D0F3          8438  STY IFLAGS+1
D0F5          8424  STY ICSB
D0F7          8425  STY ICSE
D0F9          8431  STY IERR
D0FB          8426  STY ICSL
D0FD          8427  STY IEXP
D0FF          8428  STY IEXP+1
D101          8420  STY JLABL
D103          8421  STY JORG
D105          8422  STY JBYWOR
D107          8429  STY ICOLP
D109          ;hier TAY entfernt
D109 H8920    2093E9 JSR INALL
D10C          ;JSR OUTDP1 entfernt, kein Listing beim ersten Durchlauf
D10C          ;dadurch wird der Assembler erheblich schneller

D3AC                .OPT LIS
D3AC H8838    A987  LDA #<OPTDRE
D3AE          A0DD  LDY #>OPTDRE
D3B0          ;Die bisher nicht benutzten Anweisungen SYM,NOS,NOC,CNT und
D3B0          ;COU sowie ERR (=NOL) und NOE (=LIS) wurden entfernt.
```

```
D3B0          A20D    LDX #13          ;7 Anweisungen weniger
D3B2          4C84D2  JMP H9938

D3CC                  .OPT LIS
D3CC H390     8537    STA IFLAGS
D3CE          ;Die Label H303,H304,H305,H306 der nicht implementierten
D3CE          ;Assembler Anweisungen wurden entfernt.

D3FE                  .OPT LIS
D3FE H10MRC   E623    INC PASS
D400          A225    LDX #P2MSG-MB
D402          20C8DF  JSR PMSG
D405          20D9DF  JSR CRLF
D408          AD12A4  LDA INFLG
D40B          C955    CMP #'U'         ;User, nicht warten
D40D          F01B    BEQ H117
D40F          C946    CMP #'F'         ;Floppy, nicht warten
D411          F017    BEQ H117
D413          C94D    CMP #'M'         ;'M', Pass 2
D415          D005    BNE H101
D417          F04F    BEQ HN1
D419 TOG      20FAE9  JSR NEWTOG       ;Toggle Tape Funktion
D41C H101     2073E9  JSR REDOUT       ;Taste holen
D41F          C931    CMP #'1'         ;'1' ?
D421          F0F6    BEQ TOG          ;Toggle Tape
D423          AD12A4  LDA INFLG
D426          C954    CMP #'T'         ;Tape ?
D428          D03E    BNE HN1          ;nein, Pass 2
D42A H117     20D9D6  JSR NFNDNB       ;Filename in der Zeile ?
D42D          9006    BCC H112         ;nein
D42F          2092DF  JSR FLNAME       ;suche Filenamen
D432          4CC9D0  JMP NXTPAS       ;Pass 2
D435 H112     AD12A4  LDA INFLG
D438          C955    CMP #'U'         ;User ?
D43A          D006    BNE FFLA
D43C FFLT     2050E8  JSR WHERET       ;User Input mit C=0
D43F          4CC9D0  JMP NXTPAS       ;Pass 2
D442 FFLA     C946    CMP #'F'         ;Floppy ?
D444          D010    BNE H112N        ;nein
D446          A444    LDY OLDPC        ;Multiple Files ?
D448          D0F2    BNE FFLT         ;> 0 bedeutet Multiple Files
D44A          C587    CMP OBJDEV       ;Output auch Floppy ?
D44C          F0EE    BEQ FFLT         ;File anfordern
D44E          A000    LDY #0
D450          20EFDF  JSR FHELP        ;sonst F=*
D453          4C68D4  JMP HN1
D456 H112N    A200    LDX #0
D458 H116     B589    LDA SRCNAM,X     ;setze Filenamen für Tape
D45A          9D2EA4  STA NAME,X
D45D          2002EF  JSR OUTDP1
D460          E8      INX
D461          E005    CPX #5
D463          D0F3    BNE H116
D465          20B2DF  JSR FLN15        ;suche File
D468 HN1      4CC9D0  JMP NXTPAS       ;Pass 2

D49E                  .OPT LIS
D49E H9917    C93B    CMP #';'
D4A0          F051    BEQ H9936
D4A2          20E8DF  JSR CAPIT        ;auch kleines 'a' testen
D4A5          C941    CMP #'A'
D4A7          D023    BNE H39
D4A9          E8      INX
D4AA          E435    CPX IMAXCL
D4AC          CA      DEX
D4AD          B006    BCS H9965
D4AF          B447    LDY ICRD+1,X
D4B1          C020    CPY #' '
D4B3          D017    BNE H39
D4B5          ;Accumulator Mode: Hinzugekommen sind DEC A und INC A
D4B5          ;JOPTEM ist direkter Index in die Tabelle FORACC, die
D4B5          ;den Offset enthält, der zum Basis-Opcode in JOPBAS
D4B5          ;addiert wird. Acc-Mode ist nicht erlaubt, wenn der Eintrag
D4B5          ;in FORACC negativ ist ($FF).
D4B5 H9965    A410    LDY JOPTEM
D4B7          88      DEY
D4B8          C00D    CPY #13          ;A-mode nur bis JOPTEM=13
```

```
D4BA          B00B    BCS H458
D4BC          B937DF  LDA FORACC,Y    ;JOPTEM ist Index in FORACC
D4BF          3006    BMI H458
D4C1          2022DC  JSR MEMBAK      ;Offset gefunden
D4C4          4C81D4  JMP H459

D51F                  .OPT LIS
D51F H20      202BD7  JSR NFNCMP
D522          ;kein Index, weiter mit H40. Die Sonderbehandlung des
D522          ;JMP Befehls bei H500 entfällt.
D522          9033    BCC H40          ;statt bisher H500
D524          C929    CMP #')'
D526          D00C    BNE H51
D528          E61C    INC JOPTYP
D52A          E61C    INC JOPTYP
D52C          ;keine Sonderbehandlung für JMP mehr
D52C          ;Letztes Zeichen war ')'. Wenn Zeilenende, dann neuer
D52C          ;Indirect-Mode.
D52C          E8      INX
D52D          E435    CPX IMAXCL
D52F          9003    BCC H51

D531          4C99D5  JMP H22A         ;ist neuer Indirect-Mode
D534 H51      B546    LDA ICRD,X
D536          C92C    CMP #','
D538          D003    BNE H203
D53A          ;keine Sonderbehandlung für JMP mehr
D53A          2076D6  JSR CKENDI
D53D H203     B546    LDA ICRD,X
D53F          ;Teste Index X oder Y oder teste auf ' ' bei Indirect-Mode
D53F          20EBDF  JSR CAPIT        ;auch Kleinbuchstaben
D542          C958    CMP #'X'
D544          F00F    BEQ H271
D546          C959    CMP #'Y'
D548          F009    BEQ H27
D54A          ;teste auf ' ' für den neuen Indirect-Mode
D54A          C920    CMP #' '
D54C          F04B    BEQ H22A         ;ist Indirect
D54E          A912    LDA #$12
D550          4C81D6  JMP LTS1Y
D553          ;keine Sonderbehandlung von JMP mehr
D553 H27      E61C    INC JOPTYP

D557                  .OPT LIS
D557 H40      A51D    LDA JNOPV
D559          F003    BEQ H40Z
D55B          ;jetzt mit JMP H41, ist für 'relativ' zu weit
D55B          4CE0D5  JMP H41
D55E H40Z     A510    LDA JOPTEM

D599                  .OPT LIS
D599          ;Behandle Indirect-Mode: setze JOPTYP auf 5
D599 H22A     A905    LDA #5
D59B          851C    STA JOPTYP
D59D          D006    BNE H22C         ;teste nur <= 9
D59F H22      A51C    LDA JOPTYP
D5A1          C906    CMP #6
D5A3          9020    BCC H400
D5A5 H22C     C90A    CMP #10
D5A7          B01C    BCS H400
D5A9          ;haben wir einen JMP ?
D5A9          A50F    LDA JOPBAS
D5AB          C94C    CMP #76
D5AD          D004    BNE H22D
D5AF          A902    LDA #2           ;ja, 2 Bytes erzeugen
D5B1          D01C    BNE JMP2B
D5B3 H22D     A527    LDA IEXP

D5C5                  .OPT LIS
D5C5 H400     A902    LDA #2
D5C7          8530    STA JOPLEN
D5C9          A527    LDA IEXP
D5CB          D013    BNE H41
D5CD          A901    LDA #1
D5CF JMP2B    8530    STA JOPLEN       ;Einsprung für JMP

D5EB                  .OPT LIS
D5EB H45      A8      TAY
D5EC          88      DEY
```

```
D5ED          B958DE LDA KLUDG-1,Y
D5F0          ;Eintrag in KLUDG = 0 bedeutet unerlaubten Operanden
D5F0          F008   BEQ H49

D697                 .OPT LIS
D697 HFIL     20D9D6 JSR NFNDNB
D69A          9003   BCC HFIL1
D69C          2092DF JSR FLNAME
D69F HFIL1    E644   INC OLDPC        ;OLDPC > 0 bedeutet Multiple Files
D6A1          A946   LDA #70
D6A3          4CDBD3 JMP LTS10

D6B0                 .OPT LIS
D6B0          ;erzeuge Zeile mit 60 Bindestrichen bei .PAG
D6B0          A03B   LDY #59
D6B2 HPAG1    A92D   LDA #'-'
D6B4          20BCE9 JSR OUTALL
D6B7          88     DEY
D6B8          10F8   BPL HPAG1

D765                 .OPT LIS
D765 NALPH    B546   LDA ICRD,X
D767          C941   CMP #'A'
D769          9004   BCC K10
D76B          ;lasse alle Zeichen bis $7E, d.h. Kleinbuchstaben und
D76B          ;deutsche Umlaute ($5B bis $5D und $7B bis $7E) zu.
D76B          C97F   CMP #$7F
D76D          90E3   BCC K15
D76F K10      18     CLC
D770 QL01     60     RTS

D908                 .OPT LIS
D908 E20      2065D7 JSR NALPH
D90B          904A   BCC E40
D90D          ;wandle Kleinbuchstaben in Großbuchstaben für Hex-Zahlen
D90D          20E8DF JSR CAPIT
D910          38     SEC
D911          E937   SBC #55
D913 E30      C517   CMP KBASE

D9E1                 .OPT LIS
D9E1 NOPFND   A947   LDA #<OPRNDE
D9E3          A0DE   LDY #>OPRNDE
D9E5          A23F   LDX #63          ;jetzt 63 statt bisher 55 Opcodes
D9E7          20F7D9 JSR OPSRCH

D9FD                 .OPT LIS
D9FD G210     B92A00 LDA ISYM,Y
DA00          ;wandle Kleinbuchstaben in Großbuchstaben für Vergleich
DA00          ;mit der Tabelle der Mnemonics bzw. Assembler Anweisungen
DA00          20E8DF JSR CAPIT
DA03          D10D   CMP (OPRTBL),Y

DA8E                 .OPT LIS
DA8E          ;Ab hier Assembler Formatierer nach
DA8E          ;R. Wollenberg, 65xx Micro Mag Nr. 32, Seiten 49 ff.
DA8E          A533   LDA IPC+1         ;lade aktuellen PC
DA90          A632   LDX IPC
DA92          2042EA JSR WRITAZ        ;Anzeige
DA95          A200   LDX #0
DA97          A520   LDA JLABL         ;Label in der Zeile ?
DA99          F025   BEQ Q32           ;nein
DA9B          C8     INY               ;ja, schreibe ein Blank
DA9C          20E3DF JSR ABLK
DA9F Q34      B546   LDA ICRD,X        ;unterdrücke führende Blanks
DAA1          E8     INX
DAA2          C920   CMP #' '
DAA4          D006   BNE Q33
DAA6          E435   CPX IMAXCL        ;alles abgearbeitet ?
DAA8          F01E   BEQ Q35           ;ja
DAAA          D0F3   BNE Q34           ;nein
DAAC Q33      C8     INY               ;schreibe Label Buchstabenweise
DAAD          20BCE9 JSR OUTALL
DAB0          E435   CPX IMAXCL        ;alles abgearbeitet ?
DAB2          F00C   BEQ Q32           ;ja
DAB4          B546   LDA ICRD,X       ;nächstes Zeichen
DAB6          E8     INX
```

```
DAB7          C920    CMP #' '        ;Blank ?
DAB9          F005    BEQ Q32         ;d.h. Label zu Ende
DABB          C93D    CMP #'='        ;'=' ?
DABD          D0ED    BNE Q33         ;d.h. Label zu Ende
DABF          CA      DEX             ;ein Zeichen zu viel
DAC0 Q32      20E3DF  JSR ABLK        ;fülle Raum auf, für gleich viele Zeichen
DAC3          C8      INY
DAC4          C008    CPY #8          ;acht Zeichen gedruckt ?
DAC6          D0F8    BNE Q32
DAC8 Q35      8619    STX J           ;merke Zahl der abgearbeiteten Zeichen
DACA          ;Ende des 1. Teils des geänderten Formatierers
DACA          ;Neueinstellen von OLDPC entfällt
DACA Q30      A504    LDA LTBSV

DB1E                  .OPT LIS
DB1E Q65      C007    CPY #7
DB20          1006    BPL Q76A

DB28                  .OPT LIS
DB28          ;Ab hier Teil 2 des Assembler Formatierers nach Wollenberg
DB28 Q76A     A000    LDY #0
DB2A          8420    STY JLABL       ;lösche Label
DB2C          A619    LDX J           ;alter Zeiger auf Quelltext-Zeile
DB2E          E435    CPX IMAXCL      ;alles abgearbeitet ?
DB30          B079    BCS Q80         ;ja
DB32 Q73      B546    LDA ICRD,X      ;nein, lies Zeichen
DB34          C920    CMP #' '        ;unterdrücke Blanks
DB36          D007    BNE Q72
DB38          E8      INX
DB39          E435    CPX IMAXCL      ;alles abgearbeitet ?
DB3B          B06E    BCS Q80         ;ja
DB3D          90F3    BCC Q73         ;nein, weitere Blanks ?
DB3F Q72      B546    LDA ICRD,X      ;von Blank verschiedenes Zeichen
DB41          C93B    CMP #';'        ;';' ?

DB43          F031    BEQ Q20
DB45          C927    CMP #$27        ;first ''' ?
DB47          F041    BEQ Q22
DB49          C920    CMP #' '        ;Blank ?
DB4B          F01E    BEQ Q70
DB4D          C93D    CMP #'='        ;'=' ?
DB4F          F004    BEQ Q77         ;ja, schreibe das '='
DB51          C92A    CMP #'*'        ;'*' ?
DB53          D007    BNE Q76         ;nein
DB55 Q77      20BCE9  JSR OUTALL      ;ja, schreibe das '=' bzw. '*'
DB58          E619    INC J           ;Merker, um folgende Blanks zu löschen
DB5A          D007    BNE Q74
DB5C Q76      20BCE9  JSR OUTALL      ;nein, schreibe das Zeichen
DB5F          A900    LDA #0          ;lösche Blank Merker
DB61          8519    STA J
DB63 Q74      C8      INY             ;erhöhe TV-Zeiger
DB64 Q75      E8      INX             ;erhöhe Quelltext Zeiger
DB65          E435    CPX IMAXCL      ;alles abgearbeitet ?
DB67          B042    BCS Q80         ;ja
DB69          90D4    BCC Q72         ;nein
DB6B Q70      A519    LDA J           ;unterdrücke mehrfache Blanks
DB6D          D0F5    BNE Q75
DB6F          20E3DF  JSR ABLK        ;schreibe Blank
DB72          E619    INC J           ;setze Blank Merker
DB74          D0ED    BNE Q74
DB76 Q20      20E3DF  JSR ABLK        ;positioniere auf gleiche Kommentarzeile
DB79          C8      INY
DB7A          C010    CPY #16
DB7C          90F8    BCC Q20
DB7E Q21      B546    LDA ICRD,X      ;schreibe Kommentar
DB80 Q21C     20BCE9  JSR OUTALL
DB83          E8      INX
DB84          E435    CPX IMAXCL      ;alles abgearbeitet ?
DB86          B023    BCS Q80         ;ja
DB88          90F4    BCC Q21         ;nein
DB8A Q22      B546    LDA ICRD,X      ;schreibe String in Quotes
DB8C          20BCE9  JSR OUTALL
DB8F          C8      INY
DB90          E8      INX
DB91          E435    CPX IMAXCL      ;alles abgearbeitet ?
DB93          B016    BCS Q80         ;ja
DB95          B546    LDA ICRD,X      ;nein, nächstes Zeichen des Strings
DB97          C927    CMP #$27        ;zweites ''' ?
```

```
DB99          D0EF    BNE Q22          ;nein, also weiter
DB9B          F0BF    BEQ Q76          ;ja, kommt Kommentar ?
DB9D          ;Ende des 2. Teils des geänderten Formatierers
DB9D Q21A     B546    LDA ICRD,X

DD1B                  .OPT LIS
DD1B OBDL     A541    LDA S2+1
DD1D          2002DC  JSR NUMOBJ
DD20          A540    LDA S2
DD22          2002DC  JSR NUMOBJ
DD25          CA      DEX
DD26          D0F3    BNE OBDL
DD28          20FBDC  JSR CRLFO
DD2B          AD13A4  LDA OUTFL6
DD2E          48      PHA
DD2F          AD12A4  LDA INFL6
DD32          48      PHA
DD33          A587    LDA OBJDEV       ;Object File
DD35          8D13A4  STA OUTFL6
DD38          200AE5  JSR DU11         ;schließen
DD3B          68      PLA
DD3C          20B1E3  JSR CLFP
DD3F          68      PLA
DD40          8D13A4  STA OUTFL6
DD43          60      RTS

DD44                  .OPT LIS
DD44          ;Startadressen der Assembler Anweisungen.
DD44          ;Die Adressen von SYM,NOS,NOC,CNT,COU,ERR,NOE entfallen.
DD44 ASMJMP   9BD2    .WOR H111,H113,H506,H26
DD46          A3D2
DD48          9FD2
DD4A          D6D3
DD4C          A6D6    .WOR HPAG,H10,H301,HFIL
DD4E          E0D3
DD50          9FD3
DD52          97D6
DD54          B5D3    .WOR H323,H302
DD56          B9D3
DD58          CAD3    .WOR H309,H310,H308
DD5A          C6D3

DD5C          C1D3
DD5E          BDD3    .WOR H307
DD60
DD60          ;Assemblerdirektiven SYM,NOS,NOC,CNT,COU,ERR,NOE entfernt
DD60 ASMDIR   4259    .BYT 'BYTWORDBYSKIPAGEND'
DD72          4F5054  .BYT 'OPT'
DD75 ASMDRE   46494C  .BYT 'FIL'
DD78 OPTDIR   4745    .BYT 'GENNO6'
DD7E          4D45    .BYT 'MEMNOMLIS'
DD87 OPTDRE   4E4F4C  .BYT 'NOL'
DD8A OPRNDS
DD8A          ;Die Mnemonics für die neuen Befehle wurden eingefügt:
DD8A          ;STZ,TRB,TSB,PHY,PLY,PHX,PLX,BRA.
DD8A          5354    .BYT 'STZTRBTSB'
DD93          5048    .BYT 'PHYPLYPHXPLXBRA'
DDA2          4144    .BYT 'ADCANDASLBCCBCSBEQ'  ;usw.

DE4A                  .OPT LIS
DE4A          ;Tabelle der Offsets der Basis-Opcodes.
DE4A          ;Die Werte 17,48,56,58 und 101 sind neu hinzugekommen.
DE4A          ;255 bedeutet, daß der Operand nicht erlaubt ist.
DE4A KLTBLP   00      .BYT 0,4,8,12
DE4B          04
DE4C          08
DE4D          0C
DE4E          10      .BYT 16,20,24,28
DE4F          14
DE50          18
DE51          1C
DE52          20      .BYT 32,255
DE53          FF
DE54          11      .BYT 17,48,56,58,101
DE55          30
DE56          38
DE57          3A
DE58          65
```

```
DE59 KLUDG
DE59            ;Die Adressierungsarten sind in ein System von 0 bis 14
DE59            ;unterteilt. Die Einteilung ist wie folgt:
DE59            ;0,4,5,8 und 9 sind nicht belegt;
DE59            ;1=ZP  2=ZP,X  3=ZP,Y  6=( )  7=( ,X)  10=( ),Y  11=#
DE59            ;12=ABS  13=ABS,X  14=ABS,Y
DE59            ;Die Einträge von KLUDG sind Zeiger in die Tabelle
DE59            ;KLTBL. Dort steht für jede der 16 Befehlsklassen als
DE59            ;Halbbyte der Index in die Tabelle KLTBLP der Offsets.
DE59            ;Die Befehlsklassen zu den Befehlen in OPRNDS stehen in
DE59            ;der Tabelle KTMPL.
DE59 FF         FF      .BYT 255,15,31,0,0,47,63,0,0
DE5A 0F         0F
DE5B 1F         1F
DE5C 00         00
DE5D 00         00
DE5E 2F         2F
DE5F 3F         3F
DE60 00         00
DE61 00         00
DE62 4F         4F      .BYT 79,95,111,127,143
DE63 5F         5F
DE64 6F         6F
DE65 7F         7F
DE66 8F         8F
DE67
DE67            ;Tabelle der Indices in die Offset-Tabelle.
DE67            ;jeweils 8 Byte stehen in der Reihenfolge für
DE67            ;ZP ZP,X ZP,Y ( ) ( ,X) ( ),Y # ABS ABS,X ABS,Y
DE67 KLTBL      11      .BYT $11,$99,$01,$01,$00,$10,$00,$00
DE68 99         99
DE69 01         01
DE6A 01         01
DE6B 00         00
DE6C 10         10
DE6D 00         00
DE6E 00         00
DE6F 55         55      .BYT $55,$99,$49,$45,$94,$94,$49,$94
DE70 99         99
DE71 49         49
DE72 45         45
DE73 94         94
DE74 94         94
DE75 49         49
DE76 94         94
DE77 99         99      .BYT $99,$99,$99,$99,$49,$59,$99,$99
DE78 99         99
DE79 99         99
DE7A 99         99
DE7B 49         49
DE7C 59         59
DE7D 99         99
DE7E 99         99
DE7F AA         AA      .BYT $AA,$B9,$99,$99,$99,$99,$99,$99
DE80 B9         B9
DE81 99         99
DE82 99         99
DE83 99         99
DE84 99         99
DE85 99         99
DE86 99         99
DE87 00         00      .BYT $00,$B9,$99,$99,$99,$99,$99,$99
DE88 B9         B9
DE89 99         99
DE8A 99         99
DE8B 99         99
DE8C 99         99
DE8D 99         99
DE8E 99         99
DE8F 44         44      .BYT $44,$99,$99,$99,$99,$99,$99,$99
DE90 99         99
DE91 99         99
DE92 99         99
DE93 99         99
DE94 99         99
DE95 99         99
DE96 99         99
```

```
DE97          29      .BYT $29,$99,$90,$E0,$99,$09,$99,$99
DE98          99
DE99          90
DE9A          E0
DE9B          99
DE9C          09
DE9D          99
DE9E          99
DE9F          33      .BYT $33,$00,$23,$23,$22,$32,$29,$2C
DEA0          00
DEA1          23
DEA2          23
DEA3          22
DEA4          32
DEA5          29
DEA6          2C
DEA7          77      .BYT $77,$99,$69,$67,$99,$96,$69,$9D
DEA8          99
DEA9          69
DEAA          67
DEAB          99
DEAC          96
DEAD          69
DEAE          9D
DEAF          66      .BYT $66,$99,$99,$99,$99,$79,$99,$99
DEB0          99
DEB1          99
DEB2          99
DEB3          99
DEB4          79
DEB5          99
DEB6          99
DEB7
DEB7
DEB7 KTMPL
DEB7          ;Tabelle der Befehlsklassen. Ursprünglich von 1 bis 14,
DEB7          ;jetzt von 1 bis 16. Zuordnung:
DEB7          ;1: LDA, ORA, SBC, CMP, ADC, AND, EOR
DEB7          ;2: STA  ;3: JMP  ;4: JSR  ;5: ASL, LSR, ROL, ROR
DEB7          ;6: CPX, CPY  ;7: BIT  ;8: LDY  ;9: STX  ;10: STY
DEB7          ;11: LDX  ;12: DEC  ;13: INC  ;14: von 21 auf 14 umgesetzt
DEB7          ;alle Branch Befehle  ;15: TSB, TRB  ;16: STZ
DEB7          ;21: alle 'Implied' Befehle
DEB7          10      .BYT 16,15,15
DEB8          0F
DEB9          0F
DEBA          14      .BYT 20,20,20,20,21
DEBB          14
DEBC          14
DEBD          14
DEBE          15
DEBF          01      .BYT 1,1,5,21,21,21,7,21,21,21  ;usw.
DEC0          01
DEC1          05
DEC2          15
DEC3          15
DEC4          15
DEC5          07
DEC6          15
DEC7          15
DEC8          15

DEF7                  .OPT LIS
DEF7 KCODE
DEF7          ;Tabelle der Basis Opcodes
DEF7          64      .BYT $64,$14,$04
DEF8          14
DEF9          04
DEFA          5A      .BYT $5A,$7A,$DA,$FA,$80
DEFB          7A
DEFC          DA
DEFD          FA
DEFE          80
DEFF          61      .BYT 97,33,6,144,176,240,36,48,208,16  ;usw.
DF00          21
DF01          06
DF02          90
DF03          B0
```

```
DF04          F0
DF05          24
DF06          30
DF07          D0
DF08          10

DF37                  .OPT LIS
DF37          ;Tabelle der Offsets bei Accumulator-Mode
DF37 FORACC   FF      .BYT 255,255,255,255,4,255,255
DF38          FF
DF39          FF
DF3A          FF
DF3B          04
DF3C          FF
DF3D          FF
DF3E          FF      .BYT 255,255,255,255,116,52
DF3F          FF
DF40          FF
DF41          FF
DF42          74
DF43          34
DF44          ;Geänderte Texte
DF44 MB               =*
DF44 ERMSG    2A2A    .BYT '**ERROR',$A0
DF4B          A0
DF4C STOERR   5359    .BYT 'SYM TBL '
DF54          4F56    .BYT 'OVERFLO',$57+$80
DF5B          D7
DF5C NUMERR   4552    .BYT 'ERRORS',$3D+$80
DF62          BD
DF63 P1MSG    5041    .BYT 'PASS ',$31+$80
DF68          B1
DF69 P2MSG    5041    .BYT 'PASS ',$32+$80
DF6E          B2
DF6F HDRMSG   4153    .BYT 'ASSMOS 1.',$30+$80
DF78          B0
DF79 LOCM     3D      .BYT '=',$3D+$80
DF7A          BD
DF7B LSMSG    4C49    .BYT 'LIST',$3F+$80
DF7F          BF
DF80 OBMSG    4F424A  .BYT 'OBJ',$3F+$80
DF83          BF
DF84 LSTMSG   4C49    .BYT 'LIST',$2D+$80
DF88          AD
DF89 OBJMSG   4F424A  .BYT 'OBJ',$2D+$80
DF8C          AD
DF8D

DF92                  .OPT LIS
DF92 FLNAME   203BEA  JSR CRCK2       ;CR auf Display
DF95          A000    LDY #0
DF97          A629    LDX ICOLP
DF99 NCHR     B546    LDA ICRD,X      ;Filenamen lesen
DF9B          E435    CPX IMAXCL      ;bis Zeilenende
DF9D          B00E    BCS ISEND
DF9F          C005    CPY #5          ;nur 5 für Tape
DFA1          B003    BCS NON
DFA3          992EA4  STA NAME,Y
DFA6 NON      2002EF  JSR OUTDP1      ;anzeigen
DFA9          E8      INX
DFAA          C8      INY
DFAB          D0EC    BNE NCHR
DFAD ISEND    84A7    STY LAE         ;Länge des Filenamens
DFAF          20E9E8  JSR NAMO2       ;Rest mit Blanks auffüllen für Tape
DFB2 FLN15    AD12A4  LDA INFLG
DFB5          C946    CMP #'F'        ;Floppy ?
DFB7          D005    BNE TP
DFB9          A4A7    LDY LAE         ;Floppy File eröffnen
DFBB          4CEFDF  JMP FHELP
DFBE TP       C954    CMP #'T'        ;Tape ?
DFC0          D003    BNE OTH         ;andere
DFC2          4C2FE3  JMP LOADTA      ;Tape File suchen
DFC5 OTH      4C68E8  JMP WHE2        ;andere
DFC8
DFC8          ;PMSG jetzt wie KEP im Monitor
DFC8          ;spart die ';' hinter den Texten
DFC8 PMSG     BD44DF  LDA MB,X
DFCB          48      PHA
```

```
DFCC          297F    AND #$7F
DFCE          20BCE9  JSR OUTALL
DFD1          E8      INX
DFD2          68      PLA
DFD3          10F3    BPL PMSG
DFD5          60      RTS
DFD6
DFD6 CRLFB    20E3DF  JSR ABLK
DFD9 CRLF     48      PHA
DFDA          8A      TXA
DFDB          48      PHA
DFDC          20F0E9  JSR CRLFS
DFDF          68      PLA
DFE0          AA      TAX
DFE1          68      PLA
DFE2          60      RTS
DFE3
DFE3
DFE3 ABLK     A920    LDA #IHBLK
DFE5          4CBCE9  JMP OUTALL
DFE8
DFE8          ;Umwandlung von Kleinbuchstaben in Großbuchstaben
DFE8 CAPIT    C941    CMP #$41
DFEA          9002    BCC CAPK
DFEC          29DF    AND #$DF
DFEE CAPK     60      RTS
DFEF
DFEF          ;Hilfsroutine für Floppydisk Betrieb:
DFEF          ;altes Input File schließen, neues File eröffnen
DFEF FHELP    A200    LDX #0          ;X=0 Start des Filenamens
DFF1          86A7    STX LAE         ;ab $A438
DFF3          20FBEB  JSR NORMAL      ;Namen einrichten, Länge setzen
DFF6          48      PHA             ;Länge in A
DFF7          20C4E3  JSR ENDTA       ;Input File schließen
DFFA          68      PLA             ;Länge holen
DFFB          85A7    STA LAE
DFFD          4CC2F3  JMP ALLF        ;File eröffnen
E000
E000                  .END
ERRORS=0000
```

Dipl.-Ing. Wolfgang Krebs, 1000 Berlin 41

Eprommer

für 2764 und 27128 am AIM 65

Das nachfolgende Programm baut auf demjenigen in 65xx MICRO MAG Nr. 20 S. 18 auf. Die Schaltung ist so ausgelegt, daß möglichst wenig Port-Anschlüsse einer VIA 6522 belegt werden. Hier wird der User-Port und die User-VIA des AIM 65 verwendet. Die wenigen Anschlußpins ermöglichen ein einfaches Übertagen auf andere Rechnersysteme mit weniger Portanschlüssen, z.B. einen Commodore, bei dem noch 2 Portanschlüsse des Steckers für den Kassettenrekorder mit verwendet werden müßten. Der Betrieb am AIM 65 an den Pins eines evtl. nicht mehr angeschlossenen Thermodruckers ist ebenfalls möglich (Ändern der VIA-Adressen).

Die EPROMs 2764 und 27128 haben 28 Pin und werden mit 21 Vpp (Pin 1) programmiert. Die Programmierspannung kann während des ganzen Programmiervorganges an Pin 1 anliegen. Ein Byte wird durch einen 50 msec. dauernden Low-Puls an $\overline{\text{PGM}}$ (Pin 27) programmiert. Die Zeit wird vom Timer 2 der User-VIA abgeleitet. Die Zeiten sind für einen 1 MHz-Takt ausgelegt. Statt des DELAY-Unterprogrammes kann auch das gleichnamige im AIM-Monitor verwendet werden. Hier ist es gelistet, um die Dinge für andere Rechner zu vereinfachen..

Die Adressen werden mit den CMOS-Zählern 4040 und 4013 abgezählt und stehen dann am EPROM an. Es wird der kürzeste Weg des Zählens zu der gewünschten Adresse gewählt. Entweder wird ab erreichter Stelle weitergezählt oder mit einem Reset-Impuls wieder von Null an.

Das Programm ist für das RAM des AIM 65 ab Adresse $200 geschrieben. Es ist aber auch EPROM-

fähig. - Hier wird es mit Taste 'F1' aufgerufen. Danach folgt die Abfrage des EPROM-Typs und der Adreßbereiche im Dialog. Es folgt ein Menü, aus dem die einzelnen Funktionen des Eprommers aufgerufen werden können. Mit 'F2' kann gleich in das Menü gesprungen werden. Es stehen hier folgende Funktionen zur Verfügung:

A Adreßbereiche ändern
R Read EPROM ins RAM
P Programmiere EPROM. Es wird jedes Byte mit dem RAM verglichen und nur dann programmiert, wenn beide Bytes ungleich sind. Nach je 256 Byte wird ein P im Display angezeigt.
N Alle Bytes werden programmiert ohne Vergleich. Sonst wie P. Nach N und P wird ein Verify durchgeführt. Das Programmieren kann mit der Taste 'ESC' abgebrochen werden, die man länger drücken muß.
C Check, ob das EPROM im Bereich leer ist ($FF).
V Verify, vergleiche EPROM und RAM im Adreßbereich
M Fülle RAM im Adreßbereich mit hex FF.

Wegen der mehrzeiligen Ausgaben empfiehlt es sich, den Thermodrucker einzuschalten oder mit einem Video-Display zu arbeiten.

Ausblick: Die Erweiterung der Schaltung für größere EPROMS ist einfach möglich. Hierzu sind keine weiteren Anschlußpins nötig, sondern es muß nur der Adreßzähler verlängert und das Programm entsprechend angepaßt werden.

Um auch den 2732A (A für 21V Vpp) programmieren zu können, müssen die Programmierspannung und +5V und 0 Volt an Pin 22 gelegt werden können. Zur Ansteuerung sind noch Ausgänge am 74C174 frei. - Interessant ist auch der 'Intelligent Programming Algorithm' im INTEL-Datenblatt mit +6V an Vcc und +21V an Vpp.

Quellen:
65xx MICRO MAG Nr. 20, S. 18: EPROM-Programmiereinheit für AIM 65
65xx MICRO MAG Nr. 21, S. 47: Feedback
Datenblätter 2764 und 27128 der Firma INTEL
Application Note 'AIM-65 Prom-Programmer' der Fa. Rockwell, 18.2.80

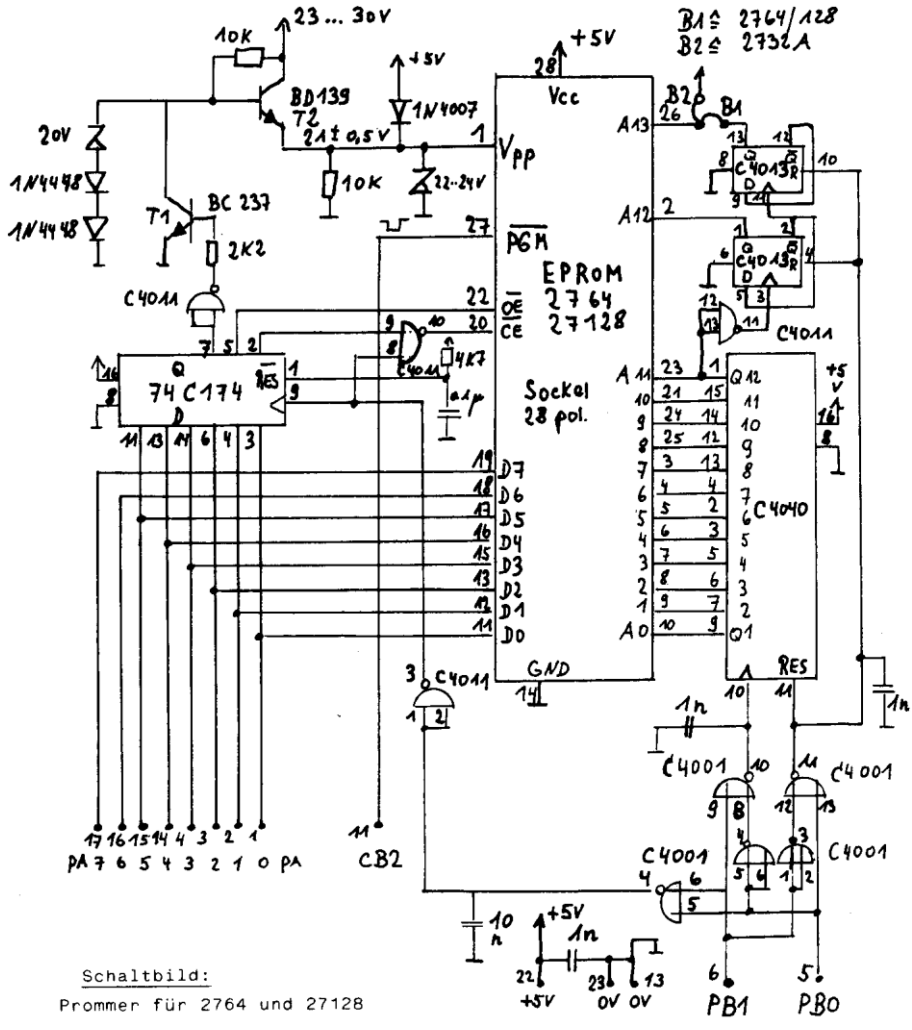
Die Schaltung des EPROMMERs findet sich auf der nächsten Seite. Anschließend daran die Programmliste.

Tabelle: Bedeutung der Steuerinformationen

Read	PA2	PA1	PA0	hex	CB2
Read	0	0	1	01	1
Progr.	1	1	1	07	⌐_⌐
Prog./Ver.	1	0	1	05	1
Tri-State	0	1	0	02	1

Zähler-Betriebszustände:

	PB1	PB0
Ruhe	1	1
Reset	1	0
Clock	0	1
Latch	0	0


23 ... 30V
10K
BD139
T2
21 ± 0,5V
+5V
1N4007
20V
1N4448
1N4448
T1
BC 237
2K2
10K
22...24V
C4011
74 C174
RES
D
Q
4K7
0,1µ
+5V
Vcc
28
VPP
1
PGM
27
EPROM
2764
27128
OE
CE
22
20
Sockel
28 pol.
GND
14
B1 ≙ 2764/128
B2 ≙ 2732A
B2
B1
A13
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
D7
D6
D5
D4
D3
D2
D1
D0
C4013
C4040
Q12
Q13
Q14
Q1
PA7 6 5 4 3 2 1 0 PA
CB2
C4001
1n
10n
PB1
PB0
+5V
0V
Schaltbild:
Prommer für 2764 und 27128

```
0002  0000                    ;FILE=BURNA7 21.03.84 KREBS
0003  0000                    ;******************************
0004  0000                    ;
0005  0000                    ; EPROMER FUER 2764 UND 27128
0006  0000                    ; VERSION FUER AIM 65 MIT USER-VIA
0007  0000                    ;
0008  0000                    ;******************************
0009  0000                    ;
0010  0000                          *=$6              ;VARIABLE
0011  0006                    MRLO  *=*+2             ;SPEICHER START
0012  0008                    PRLO  *=*+2             ;PROM START
0013  000A                    MRHI  *=*+2             ;SPEICHER ENDE
0014  000C                    MRADR *=*+2             ;AKTUELLE SPEICHER-ADRRESSE
0015  000E                    PRADR *=*+2             ;AKTUELLE PROM-ADRESSE
0016  0010                    VAR1  *=*+1             ;ARBEITS-VARIABLE
0017  0011                    VAR2  *=*+1             ; *
0018  0012                    TYP   *=*+1             ;PROM-TYPE-VARIABLE
0019  0013                    MENCHR *=*+1            ;ZEICHEN IM MENUE EINGEGEBEN
0020  0014                    FBYTE *=*+1             ;BYTE FUER MFILL
0021  0015                    MITCNT *=*+2            ;MITZAEHLEN ZUR ADRESS-AUSGABE
0022  0017                    CNTH30 *=*+2            ;DELAY-ZEIT
0023  0019                    ;
0024  0019                    VIA   =$A000            ;AIM 65 USER-VIA 6522
0025  0019                    UDRB  =VIA              ;USER-VIA 6522 REGISTER
0026  0019                    UDRA  =VIA+1
0027  0019                    UDDRB =VIA+2
0028  0019                    UDDRA =VIA+3
0029  0019                    T2L   =VIA+8            ;TIMER 2 FUER DELAY
0030  0019                    T2H   =VIA+9
0031  0019                    IFR   =VIA+$D
0032  0019                    ;
0033  0019                          *=$10C            ;AIM F-TASTEN
0034  010C  4C 00 02                JMP START         ;NEU-AUFRUF
0037  010F  4C AC 02                JMP MENUE         ;SPRUNG INS MENUE
0036  0112                    ;AIM 65 ADRESSEN :
0037  0112                    ;CNTH30 =$A417      ;AIM-ADR. IM PROGRAMM DOPPELT
0038  0112                    ADDR  =$A41C            ;ADRESSE BEI EINGABE TO=
0039  0112                    ;
0040  0112                    BLANK =$E83E
0041  0112                    CRLF  =$E9F0
0042  0112                    ;DELAY =$EC0F       ;IM AIM65 DOPPELT
0043  0112                    FROM  =$E7A3            ;EINGABE FROM
0044  0112                    TO    =$E7A7            ;EINGABE TO
0045  0112                    NUMA  =$EA46            ;AUSGABE BYTE HEX
0046  0112                    WRAX  =$EA42            ;AUSGABE DOPPEL-BYTE HEX
0047  0112                    OUTALL =$E9BC           ;AUSGABE ASCII
0048  0112                    INALL =$E993            ;EINGABE ASCII
0049  0112                    RCHEK1 =$E907           ;TASTEN-ABFRAGE =AIM RCHEK
0050  0112                    PSL1  =$EE37            ;EIN '/' INS DISPLAY
0051  0112                    QM    =$E7D4            ;AUSGABE EINES ?
0052  0112                    EQUAL =$E7D8            ;AUSGABE EINES =
0053  0112                    ;
0054  0112                          *=$200            ;PROGRAMM IM AIM65 RAM
0055  0200  20 23 05          START JSR VORBER
0056  0203  D8                      CLD
0057  0204  A0 4D                   LDY #TXT8-TXT1
0058  0206  20 D4 04                JSR TEXT          ;NUR 21V TYPEN
0059  0209  20 F0 E9                JSR CRLF
0060  020C  A0 07                   LDY #TXT2-TXT1
0061  020E  20 D4 04                JSR TEXT          ;ABFRAGE NACH ROM-TYPE
0062  0211  20 F0 E9                JSR CRLF          ;8K ODER 16K

0063  0214  A0 0E                   LDY #TXT3-TXT1
0064  0216  20 D4 04                JSR TEXT
0065  0219  20 44 05          ROMTYP JSR RCHEK
0066  021C  C9 FF                   CMP #$FF          ;NICHT AIM
0067  021E  F0 F9                   BEQ ROMTYP        ;NICHT AIM
0068  0220  C9 38             K8    CMP #'8           ;2764 8K-BYTE ?
0069  0222  D0 04                   BNE K16
0070  0224  A9 20                   LDA #$20          ;TYPE-VARIABLE SETZEN
0071  0226  D0 06                   BNE TVAR
```

```
0072  0228  C9 30        K16    CMP #'0          ;27128 16K-BYTE ?
0073  022A  D0 D4              BNE START        ;TYPE NICHT MOEGLICH
0074  022C  A9 40              LDA #$40         ;TYPE-VARIABLE SETZEN
0075  022E  85 12        TVAR   STA TYP

0077  0230  20 F0 E9     MEMAD  JSR CRLF
0078  0233  A0 00        MAD0   LDY #0
0079  0235  20 D4 04             JSR TEXT
0080  0238  20 A3 E7             JSR FROM         ;MEMORY ADRESS BEREICH
0081  023B  B0 F6              BCS MAD0         ;ABFRAGEN
0082  023D  AD 1C A4             LDA ADDR
0083  0240  85 06                 STA MRLO
0084  0242  AD 1D A4             LDA ADDR+1
0085  0245  85 07                 STA MRLO+1
0086  0247  20 3E E8             JSR BLANK
0087  024A  20 A7 E7             JSR TO           ;RAM BIS ?
0088  024D  B0 E4              BCS MAD0
0089  024F  AD 1D A4             LDA ADDR+1       ;NICHT GROESSER ALS EPROM ?
0090  0252  85 0B                 STA MRHI+1
0091  0254  AD 1C A4             LDA ADDR
0092  0257  85 0A                 STA MRHI
0093  0259  38                    SEC
0094  025A  E5 06                 SBC MRLO
0095  025C  A5 0B                 LDA MRHI+1
0096  025E  E5 07                 SBC MRLO+1
0097  0260  C5 12                 CMP TYP          ;NICHT GROESSER ALS TYPE-VARIABLE
0098  0262  90 0A              BCC START1       ;NOCH EINMAL EINGEBEN
0099  0264  20 F0 E9             JSR CRLF
0100  0267  A0 20                 LDY #TXT4-TXT1   ;EPROM ANFANGS-ADRESSE
0101  0269  20 D4 04             JSR TEXT
0102  026C  30 C2              BMI MEMAD        ;NOCH EIMMAL BEI ADRESS-FEHLER

0104  026E  20 F0 E9     START1 JSR CRLF         ;EPROM ANFANG-ADRESSE MIT
0105  0271  A0 07        ST2    LDY #TXT2-TXT1   ;FROM UEBERNEHMEN
0106  0273  20 D4 04             JSR TEXT
0107  0276  20 A3 E7             JSR FROM
0108  0279  B0 F6              BCS ST2
0109  027B  AD 1C A4             LDA ADDR         ;UND NACH PROM-ADRESSE SCHREIBEN
0110  027E  85 08                 STA PRLO
0111  0280  AD 1D A4             LDA ADDR+1
0112  0283  85 09                 STA PRLO+1
0113  0285  C5 12                 CMP TYP          ;NICHT GROESSER ALS EPROM
0114  0287  90 0A              BCC INIT2
0115  0289  20 F0 E9             JSR CRLF
0116  028C  A0 20                 LDY #TXT4-TXT1   ;NOCH EINMAL BEI ADRESS-FEHLER
0117  028E  20 D4 04             JSR TEXT         ;ADRESS-FEHLER
0118  0291  30 DB              BMI START1

0120  0293  20 F0 E9     INIT2  JSR CRLF
0121  0296  20 9C 02     INIT   JSR INIT0        ;INITIALISIERUNG ALS UP
0122  0299  4C AC 02             JMP MENUE        ;ZUM MENUE-PROGRAMM
0123  029C                     ;

0124  029C  20 2B 04     INIT0  JSR DISABL       ;EPROM-D.-LEITUNGEN AUF HOCH-OHMI
C
0125  029F  20 F1 04             JSR ARES         ;ADRESS ZAEHLER AUF 0000
0126  02A2  A2 03                 LDX #3
0127  02A4  B5 06        INIT1  LDA MRLO,X       ;ADRESS-VARIABLE MIT ANFANGS-
0128  02A6  95 0C                 STA MRADR,X      ;ADRESSEN LADEN
0129  02A8  CA                    DEX
0130  02A9  10 F9              BPL INIT1
0131  02AB  60                    RTS

0133  02AC  20 23 05     MENUE  JSR VORBER       ;INITIALISIERE USER-VIA + CRLF
0134  02AF  A0 2D                 LDY #TXT5-TXT1
0135  02B1  20 D4 04             JSR TEXT         ;BETRIEBSART AUS MENUE WAEHLEN
0136  02B4  20 44 05     MEN1   JSR RCHEK        ;TASTE ?
0137  02B7  F0 FB              BEQ MEN1
0138  02B9  C9 FF              CMP #$FF
0139  02BB  F0 F7              BEQ MEN1
0140  02BD  48                    PHA
0141  02BE  20 F0 E9             JSR CRLF
```

```
0142  02C1  68           PLA
0143  02C2  85 13        STA MENCHR      ;ZEICHEN ABSPEICHERN
0144  02C4  C9 41        CMP #'A         ;ADRESSEN AENDERN
0145  02C6  D0 03        BNE MEN2
0146  02C8  4C 30 02     JMP MEMAD
0147  02CB  C9 4E     MEN2  CMP #'N      ;(NACH)BRENNEN OHNE TESTEN
0148  02CD  D0 03        BNE MEN3
0149  02CF  4C A2 03     JMP NACHB
0150  02D2  C9 52     MEN3  CMP #'R      ;PROM LESEN
0151  02D4  F0 71        BEQ LESEN
0152  02D6  C9 4D        CMP #'M         ;RAM MIT $FF FUELLEN
0153  02D8  D0 03        BNE MF0
0154  02DA  4C 66 03     JMP MFILL1
0155  02DD  C9 43     MF0   CMP #'C      ;CHECK OB EPROM LEER
0156  02DF  F0 45        BEQ TEST
0157  02E1  C9 56        CMP #'V         ;VERGLEICHE EPROM + RAM
0158  02E3  F0 18        BEQ VERGL
0159  02E5  C9 50        CMP #'P         ;NORMAL PROGRAMMIEREN DES EPROM
0160  02E7  D0 AD        BNE INIT        ;SONST NOCH EINMAL
0161  02E9  4C A2 03     JMP PROGR

0163  02EC  A0 60     VERGL1 LDY #TXT9-TXT1  ;VERGLEICHE RAM+EPROM NACH PROGR.
0164  02EE  20 D4 04        JSR TEXT        ;VERIFY
0165  02F1  20 F0 E9        JSR CRLF
0166  02F4  20 9C 02        JSR INIT0       ;ADRESS-POINTER AUF ANFANG
0167  02F7  4C FD 02        JMP VERGL       ;AB 1. ADRESSE
0168  02FA  20 7B 04  VGLNXT JSR NXTADR     ;NAECHSTE ADRESSE VERGLEICHEN
0169  02FD  A0 00     VERGL  LDY #0         ;VERGLEICHEN RAM-EPROM
0170  02FF  B1 0C           LDA (MRADR),Y   ;VERGLEICHE RAM (VAR1) MIT
0171  0301  85 10           STA VAR1        ;EPROM (VAR2)
0172  0303  20 3B 04        JSR ADRESS      ;LETZTE ADRESSE ?
0173  0306  30 05           BMI VGLEND
0174  0308  20 88 04        JSR GLEICH      ;VERGLEICHE EPROM MIT VAR1
0175  030B  F0 ED           BEQ VGLNXT      ;WENN GLEICH NAECHSTE ADRESSE
0176  030D  20 F0 E9  VGLEND JSR CRLF       ;TEST WAR ERFOGREICH
0177  0310  A0 00           LDY #0
0178  0312  20 D4 04        JSR TEXT
0179  0315  20 DB E7        JSR EQUAL       ;MEMORY=PROM
0180  0318  A0 07           LDY #TXT2-TXT1
0181  031A  20 D4 04        JSR TEXT
0182  031D  20 F0 E9        JSR CRLF
0183  0320  4C 96 02        JMP INIT        ;ZUM MENUE-START

0185  0323  20 7B 04  TSTNXT JSR NXTADR     ;CHECK OB EPROM LEER
0186  0326  A9 FF     TEST   LDA #$FF       ;LEER = $FF
0187  0328  85 10           STA VAR1
0188  032A  20 3B 04        JSR ADRESS      ;ADRESSE AUF ADRESS-ZAEHLER
0189  032D  30 05           BMI TSTEND      ;LETZTE ADRESSE ?
0190  032F  20 88 04        JSR GLEICH      ;VERGLEICHE EPROM MIT $FF
0191  0332  F0 EF           BEQ TSTNXT
0192  0334  20 F0 E9  TSTEND JSR CRLF       ;TEST ERFOLGREICH
0193  0337  A0 07           LDY #TXT3-TXT1  ;PROM LEER ANZEICHEN
0194  0339  20 D4 04        JSR TEXT
0195  033C  A0 3E           LDY #TXT6-TXT1
0196  033E  20 D4 04        JSR TEXT
0197  0341  4C 96 02        JMP INIT        ;ZUM MENUE-START

0199  0344  20 7B 04  LSNXT  JSR NXTADR     ;ADRESSE WEITER +1
0200  0347  20 3B 04  LESEN  JSR ADRESS      ;LESE EPROM IN RAM
0201  034A  30 0A           BMI LSEND
0202  034C  20 1E 04        JSR READ        ;EIN BYTE LESEN
0203  034F  A0 00           LDY #0
0204  0351  91 0C           STA (MRADR),Y   ;UND IN DEN RAM SCHREIBEN
0205  0353  4C 44 03        JMP LSNXT
0206  0356  20 F0 E9  LSEND  JSR CRLF       ;LESEN BEENDET
0207  0359  A0 00           LDY #0          ;SCHLUSSTEXT:
0208  035B  20 D4 04        JSR TEXT ;MEMORY GELADEN
0209  035E  A0 44           LDY #TXT7-TXT1
0210  0360  20 D4 04        JSR TEXT
0211  0363  4C 96 02        JMP INIT        ;ZUM MENUE-START
```

```
0213  0366                    ;FUELLE SPEICHER IM BEREICH MIT $FF
0214  0366  A9 FF     MFILL1 LDA #$FF
0215  0368  85 14            STA FBYTE
0216  036A  20 9C 02         JSR INIT0         ;ADRESS-POINTER AUF ANFANG
0217  036D  4C 73 03         JMP MFILL         ;AUCH MIT ANDEREM BYTE IM FBYTE
0218  0370  20 7B 04  MFNXT  JSR NXTADR
0219  0373  20 3B 04  MFILL  JSR ADRESS        ;TESTE OB LETZTE ADRESSE
0220  0376  30 09            BMI MFEND
0221  0378  A5 14            LDA FBYTE         ;BYTE FUER FILL
0222  037A  A0 00            LDY #0
0223  037C  91 0C            STA (MRADR),Y     ;IN RAM SCHREIBEN
0224  037E  4C 70 03         JMP MFNXT
0225  0381  20 F0 E9  MFEND  JSR CRLF          ;NACH LETZTER ADRESSE
0226  0384  A0 67            LDY #TXT10-TXT1
0227  0386  20 D4 04         JSR TEXT          ;MIT $FF GELADEN
0228  0389  A0 44            LDY #TXT7-TXT1
0229  038B  20 D4 04         JSR TEXT
0230  038E  4C 96 02         JMP INIT          ;ZUM MENUE-START

0232  0391  20 7B 04  PRGNXT JSR NXTADR        ;PROGRAMMIERE NAECHSTES BYTE
0233  0394  A5 0E            LDA PRADR
0234  0396  D0 0A            BNE PROGR
0235  0398  A5 13            LDA MENCHR        ;ALLE 256 BYTE P/N INS DISPLAY
0236  039A  20 BC E9         JSR OUTALL
0237  039D  48               PHA
0238  039E  20 44 05         JSR RCHEK         ;ZUM STOPEN MIT <ESC>
0239  03A1  68               PLA
0240  03A2                   NACHB
0241  03A2  20 3B 04  PROGR  JSR ADRESS        ;LETZTE ADRESSE ?
0242  03A5  30 57            BMI PRGEND
0243  03A7  A0 00            LDY #0
0244  03A9  B1 0C            LDA (MRADR),Y     ;LADE BYTE AUS DEM RAM
0245  03AB  85 10            STA VAR1          ;NACH ARBEITS-VARIABLE1

0246  03AD  48               PHA
0247  03AE  20 1A 04         JSR READP         ;VERIFY BYTE
0248  03B1  C5 10            CMP VAR1
0249  03B3  D0 0A            BNE PRO1          ;UNGLEICH -> IMMER PROGRAMMIEREN
0250  03B5  A5 13            LDA MENCHR
0251  03B7  C9 4E            CMP #'N           ;NACHBRENNEN ?
0252  03B9  F0 04            BEQ PRO1
0253  03BB  68               PLA
0254  03BC  4C 91 03         JMP PRGNXT        ;WAR SCHON RICHTIGES BYTE
0255  03BF                   ;
0256  03BF  A9 FF     PRO1   LDA #$FF          ;PROGRAMIERE EIN BYTE
0257  03C1  8D 03 A0         STA VIA+3         ;PORT A AUSGANG
0258  03C4  A9 07            LDA #$07          ;OE+CE=0 PIN 1 AUF 21V
0259  03C6  8D 01 A0         STA VIA+1
0260  03C9  20 FB 04         JSR LATCH         ;INS D-FLIPFLOP
0261  03CC  68               PLA
0262  03CD  8D 01 A0         STA VIA+1
0263  03D0  EA               NOP
0264  03D1  EA               NOP
0265  03D2  EA               NOP
0266  03D3  EA               NOP
0267  03D4  EA               NOP
0268  03D5  AD 0C A0  PRO0   LDA VIA+12        ;50 MSEC. PROGRAMMIER-PULS
0269  03D8  29 DF            AND #$DF          ;CB2 LOW
0270  03DA  8D 0C A0         STA VIA+12
0271  03DD  A5 18            LDA CNTH30+1      ;WERTE AUF STACK RETTEN
0272  03DF  48               PHA
0273  03E0  A5 17            LDA CNTH30
0274  03E2  48               PHA
0275  03E3  A9 C3            LDA #$C3          ;TIMER AUF 50 MSEC.
0276  03E5  85 17            STA CNTH30
0277  03E7  A9 50            LDA #$50
0278  03E9  85 18            STA CNTH30+1
0279  03EB  20 58 05         JSR DELAY         ;ZEIT-VERZOEGERUNG
0280  03EE  68               PLA
0281  03EF  85 17            STA CNTH30        ;ALTE DELAY-WERTE ZURUECK
0282  03F1  68               PLA
0283  03F2  85 18            STA CNTH30+1
```

```
0284  03F4  AD 0C A0        LDA VIA+12       ;PROGRAMMIER-PULS AUS
0285  03F7  09 E0           ORA #$E0         ;CB2 HIGH
0286  03F9  8D 0C A0        STA VIA+12
0287  03FC  D0 93           BNE PRGNXT       ;NAECHSTES BYTE PROGRAMMIEREN
0288  03FE                 ;
0289  03FE  20 F0 E9     PRGEND JSR CRLF      ;NACH LETZTEM BYTE
0290  0401  20 F0 E9           JSR CRLF
0291  0404  A0 07              LDY #TXT2-TXT1  ;PROM GELADEN
0292  0406  20 D4 04           JSR TEXT
0293  0409  A0 44              LDY #TXT7-TXT1
0294  040B  20 D4 04           JSR TEXT
0295  040E  20 F0 E9           JSR CRLF
0296  0411  20 3E 05           JSR PIEP         ;PROGRAMMIEREN FERTIG -> PIEPEN
0297  0414  20 F0 E9           JSR CRLF         ;NACH PROGRAMMIEREN EIN
0298  0417  4C EC 02           JMP VERGL1       ;PROM = RAM ? TEST

0300  041A                 ;-------------------------------
0301  041A                 ; UNTERPROGRAMME
0302  041A                 ;
0303  041A  A2 05        READP  LDX #05          ;VON PROG
0304  041C  D0 02              BNE RE1          ;EIN BYTE LESEN
0305  041E  A2 01        READ   LDX #01          ;VON READ EIN BYTE LESEN
0306  0420  20 2D 04     RE1    JSR DIS1         ;UND X INS D-FLIPFLOP

0307  0423  A2 00              LDX #$00
0308  0425  8E 03 A0           STX VIA+3        ;INPUT
0309  0428  AD 01 A0           LDA UDRA         ;DATA
0310  042B  A2 02        DISABL LDX #02         ;EPROM HOCHOHMIG
0311  042D                 DIS1                ;EIN BYTE INS D-FLIPFLOP
0312  042D  48                 PHA
0313  042E  A9 FF              LDA #$FF         ;PORT AUF AUSGANG
0314  0430  8D 03 A0           STA VIA+3
0315  0433  8E 01 A0           STX VIA+1
0316  0436  20 FB 04           JSR LATCH        ;EIN D-FF TAKT
0317  0439  68                 PLA
0318  043A  60                 RTS
0319  043B                 ;
0320  043B  A5 0D        ADRESS LDA MRADR+1      ;TEST OB LEZTE ADRESSE
0321  043D  C5 0B              CMP MRHI+1
0322  043F  90 0B              BCC ADRES1
0323  0441  D0 06              BNE ENDADR
0324  0443  A5 0A              LDA MRHI
0325  0445  C5 0C              CMP MRADR
0326  0447  B0 03              BCS ADRES1
0327  0449  A9 FF        ENDADR LDA #$FF         ;LETZTE ADRESSE
0328  044B  60                 RTS
0329  044C                 ADRES1
0330  044C  A5 0F              LDA PRADR+1
0331  044E  C5 12              CMP TYP          ;EPROM-ADRESSE GROSSER ALS EPROM?
0332  0450  B0 F7              BCS ENDADR
0333  0452                 ;
0334  0452  38            ACNT0  SEC            ;ADRESSE IN DEN ADRESS-ZAEHLER
0335  0453  A5 0E              LDA PRADR
0336  0455  E5 15              SBC MITCNT       ;AUF DEM SCHNELLSTEN WEG ZAEHLEN
0337  0457  F0 08              BEQ ACNT4
0338  0459  A5 0F              LDA PRADR+1
0339  045B  E5 16              SBC MITCNT+1
0340  045D  10 0D              BPL ACNT1
0341  045F  30 08              BMI ACNT3
0342  0461                 ;
0343  0461  A5 0F        ACNT4  LDA PRADR+1
0344  0463  E5 16              SBC MITCNT+1
0345  0465  F0 11              BEQ ACNT2
0346  0467  10 03              BPL ACNT1        ;WENN ZAEHLER > ADRESSE -> RESET
0347  0469  20 F1 04     ACNT3  JSR ARES         ;ADR-CNT=0000
0348  046C  20 ED 04     ACNT1  JSR CLOCK        ;EIN TAKT +1
0349  046F  E6 15              INC MITCNT       ;IM RAM MITZAEHLEN
0350  0471  D0 DF              BNE ACNT0
0351  0473  E6 16              INC MITCNT+1
0352  0475  4C 52 04           JMP ACNT0
0353  0478                 ;
```

```
0354  047B  A9 00     ACNT2  LDA #0          ;NOCH NICHT AM ENDE
0355  047A  60               RTS
0356  047B             ;
0357  047B  E6 0E     NXTADR INC PRADR       ;NAECHSTE ADRESSE
0358  047D  D0 02            BNE NXTMA
0359  047F  E6 0F            INC PRADR+1
0360  0481  E6 0C     NXTMA  INC MRADR
0361  0483  D0 02            BNE NXTEND
0362  0485  E6 0D            INC MRADR+1
0363  0487  60         NXTEND RTS
0364  0488             ;
0365  0488  20 1E 04   GLEICH JSR READ        ;LESE UND VERGLEICHE
0366  048B  85 11            STA VAR2
0367  048D  C5 10            CMP VAR1        ;MIT VAR1
0368  048F  D0 01            BNE UNGL
0369  0491  60               RTS
0370  0492             ;
0371  0492  68         UNGL   PLA             ;VERGLEICH UNGLEICH
0372  0493  68               PLA
0373  0494  20 F0 E9         JSR CRLF
0374  0497  A0 00            LDY #0
0375  0499  20 D4 04         JSR TEXT        ;MEMORY/PROM
0376  049C  20 37 E8         JSR PSL1
0377  049F  A0 07            LDY #TXT2-TXT1
0378  04A1  20 D4 04         JSR TEXT
0379  04A4  20 F0 E9         JSR CRLF
0380  04A7  A5 0D            LDA MRADR+1
0381  04A9  A6 0C            LDX MRADR
0382  04AB  20 42 EA         JSR WRAX        ;ADRESSE INS DISPLAY
0383  04AE  20 3E E8         JSR BLANK       ;WENN VERGLEICH UNGLEICH
0384  04B1  A5 10            LDA VAR1        ;RAM+EPROM INS DISPLAY
0385  04B3  20 46 EA         JSR NUMA        ;HEX INS DISPLAY
0386  04B6  20 37 E8         JSR PSL1
0387  04B9  A5 11            LDA VAR2
0388  04BB  20 46 EA         JSR NUMA
0389  04BE  20 3E E8         JSR BLANK
0390  04C1  A5 0F            LDA PRADR+1
0391  04C3  A6 0E            LDX PRADR
0392  04C5  20 42 EA         JSR WRAX        ;ADRESSE INS DISPLAY
0393  04C8  20 3E E8         JSR BLANK
0394  04CB  20 7B 04         JSR NXTADR
0395  04CE  20 D4 E7         JSR QM          ;?
0396  04D1  4C B4 02         JMP MEN1
0397  04D4             ;
0398  04D4  B9 6A 05   TEXT   LDA TXT1,Y      ;TEXT-PROGRAMM
0399  04D7  48               PHA
0400  04D8  29 7F            AND #$7F
0401  04DA  20 BC E9         JSR OUTALL      ;1 ZEICHEN INS DISPLAY
0402  04DD  C8               INY
0403  04DE  68               PLA
0404  04DF  10 F3            BPL TEXT
0405  04E1  60               RTS
0406  04E2             ;
0407  04E2  A9 13   EINSCH  LDA #$13
0408  04E4  A6 12            LDX TYP
0409  04E6  E0 08            CPX #$08
0410  04E8  F0 02            BEQ OUT
0411  04EA  A9 10            LDA #$10
0412  04EC  60       OUT     RTS
0413  04ED             ;
0414  04ED  A9 FD     CLOCK  LDA #$FD        ;EIN TAKT AUF DEN ADRESS-ZAEHLER
0415  04EF  D0 0C            BNE LA1
0416  04F1             ;
0417  04F1  A9 00     ARES   LDA #$00        ;ADRESS-ZAEHLER =0000
0418  04F3  85 15            STA MITCNT
0419  04F5  85 16            STA MITCNT+1
0420  04F7  A9 FE            LDA #$FE
0421  04F9  D0 02            BNE LA1
0422  04FB             ;
0423  04FB  A9 FC     LATCH  LDA #$FC        ;EIN TAKT-PULS AUF D-FLIPFLOP
0424  04FD  48       LA1    PHA
```

```
0425  04FE  AD 02 A0          LDA VIA+2
0426  0501  09 03             ORA #$03          ;OUTPUT
0427  0503  8D 02 A0          STA VIA+2
0428  0506  68                PLA
0429  0507  2D 00 A0          AND VIA
0430  050A  8D 00 A0          STA VIA           ;LOW
0431  050D  EA                NOP
0432  050E  EA                NOP
0433  050F  EA                NOP
0434  0510  EA                NOP
0435  0511  EA                NOP
0436  0512  EA                NOP
0437  0513  EA                NOP
0438  0514  EA                NOP
0439  0515  EA                NOP
0440  0516  EA                NOP
0441  0517  EA                NOP
0442  0518  EA                NOP
0443  0519  EA                NOP
0444  051A  EA                NOP
0445  051B  EA                NOP
0446  051C  EA                NOP
0447  051D  09 03             ORA #$03          ;UND PULS WEGNEHMEN
0448  051F  8D 00 A0          STA VIA           ;HIGH
0449  0522  60                RTS
0450  0523                    ;
0451  0523  20 29 05    VORBER JSR INIVIA        ;VIA AUF NICHT AKTIV
0452  0526  4C F0 E9          JMP CRLF          ;UND <CR>,<LF>
0453  0529                    ;
0454  0529                    INIVIA            ;INITIALISIERE VIA NICHT AKTIV
0455  0529  AD 0C A0          LDA VIA+$C
0456  052C  09 E0             ORA #$E0          ;CB2 HIGH
0457  052E  8D 0C A0          STA VIA+$C        ;KEIN PROGRAMM PULS
0458  0531  A9 FF             LDA #$FF          ;PORT A AUSGANG
0459  0533  8D 03 A0          STA VIA+3
0460  0536  A9 02             LDA #$02          ;EPROM HOCH-OHMIG
0461  0538  8D 01 A0          STA VIA+1
0462  053B  4C FB 04          JMP LATCH
0463  053E                    ;
0464  053E                    PIEP              ;1 MAL PIEPEN
0465  053E  A9 07             LDA #$07          ;BELL ASCII-ZEICHEN
0466  0540  20 BC E9          JSR OUTALL
0467  0543  60                RTS
0468  0544                    ;
0469  0544                    RCHEK             ;ENTSPRICHT AIM-RCHEK
0470  0544  20 07 E9          JSR RCHEK1
0471  0547  30 0C             BMI RCH2
0472  0549  C9 60             CMP #$60
0473  054B  90 02             BCC RCH1          ;KEINE KLEINSCHREIBUNG
0474  054D  29 5F             AND #$5F
0475  054F                    RCH1
0476  054F  48                PHA
0477  0550  20 BC E9          JSR OUTALL        ;SPIEGELN ZUM TERMINAL
0478  0553  68                PLA
0479  0554  60                RTS
0480  0555  A9 FF       RCH2  LDA #$FF
0481  0557  60                RTS
0482  0558                    ;
0483  0558  A5 18       DELAY  LDA CNTH30+1     ;NUR FUER ANDERE RECHNER DELAY
0484  055A  8D 08 A0          STA T2L           ;WIE IM AIM65 MONITOR
0485  055D  A5 17             LDA CNTH30
0486  055F  8D 09 A0    DE1   STA T2H
0487  0562  AD 0D A0    DE2   LDA IFR
0488  0565  29 20             AND #$20
0489  0567  F0 F9             BEQ DE2
0490  0569  60                RTS
0491  056A                    ;
0492  056A  4D 45       TXT1  .BYT 'MEMORY',$A0
0492  0570  A0
0493  0571  50 52       TXT2  .BYT 'PROM  ',$A0
0493  0577  A0
```

```
0494  0578  32 37      TXT3   .BYT '2764(8)/27128(0)?',$A0
0494  0589  A0
0495  058A  41 44      TXT4   .BYT 'ADRESSFEHLER',$A1
0495  0596  A1
0496  0597  2D 41      TXT5   .BYT '-A-R-P-C-V-M-N-?',$A0
0496  05A7  A0
0497  05A8  4C 45      TXT6   .BYT 'LEER ',$A1
0497  05AD  A1
0498  05AE  47 45      TXT7   .BYT 'GELADEN ',$A1
0498  05B6  A1
0499  05B7  4E 55      TXT8   .BYT 'NUR 21 VOLT TYPEN ',$A1
0499  05C9  A1
0500  05CA  56 45      TXT9   .BYT 'VERIFY',$A0
0500  05D0  A0
0501  05D1  4D 49      TXT10  .BYT 'MIT $FF ',$A0
0501  05D9  A0
0502  05DA            ;
0503  05DA                   .END

ERRORS = 0000 <0000>
```

OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO

CBM 710 Spezial (2)

Für diesen Rechner steht dem Autor von Seiten des Herstellers nur die kleine Fibel 'Commodore 500/600/700 Series User's Guide' Zur Verfügung. Vielen Lesern wird es nicht anders gehen. Daher mögen folgende Hinweise nützlich sein:

Editieren am Bildschirm, Bildschirmformatierung

Viele Funktionen des Editors werden in der dort beschriebenen Weise durch die Vor-Taste ESCAPE und eine nachfolgende Buchstabentaste gesteuert. Steuerungen dieser Art kann man natürlich auch durch ein PRINT-Statement in der Form PRINT CHR$(xx) erhalten. In der nachfolgenden Tabelle ist xx dabei ein dezimales Argument:

```
19      Cursor HOME
19 19   Voller Bildschirm
18      Reverse On
146     Reverse Off
153     Scroll Down
25      Scroll Up
15      Setze oberste Zeile des Fensters
143     Setze unterste Zeile des Fensters
9       Tabuliere
14      Normaler Zeichensatz ein (=Text-Modus)
142     Graphik-Zeichensatz Ein
7       Glocke Ein/Aus (Toggle)
147     Clear Screen
17      Cursor Down
157     Cursor links
29      Cursor rechts
20      Lösche/Delete Zeichen vor dem Cursor
145     Cursor nach oben
21      Lösche Zeile
149     Insert Zeile
22      Lösche bis Zeilenende
150     Lösche bis Zeilenbeginn
148     Füge Zeichen ein
```

Hat man ein Bildschirmfenster mit ESC-T und ESC-B festgelegt, so möchte man es gelegentlich wieder aufheben. Das erreicht man mit 2x HOME oder gleichwertig mit SYS 59847. Dabei erhält man wieder die Verfügung über den vollen Bildschirm. Innerhalb eines Programmes ist das für einen alternativen Bildaufbau wichtig. Die Funktionen ESC-T und ESC-B sind für die Definition eines Fensters jedoch zu umständlich, wenn man sich durch verschiedene PRINT-Befehle an die Eckpunkte des Fensters herantasten wollte, zumal dabei Information gelöscht werden könnte. Gezielter erreicht man die Definition durch Parameterübergabe (POKE) an folgende Stellen:

POKE	*Funktion*
222	*Linker Rand, Spalten 0 - 78*
223	*Rechter Rand, Spalten 1 - 79*
220	*Oberste Zeile (0 - 25)*
221	*Unterste Zeile des Fensters (0 - 25)*

Mit bis zu vier POKEs erhält man also per Programm ein Bildschirmfenster an beliebiger Stelle. Beim Wechsel von Fenstern: a) Zunächst altes Fenster ggfs. löschen mit Clear Screen, bzw. Informationen stehen lassen, b) Vollen Screen mit 2x HOME oder SYS 59847 herstellen, c) Neues Fenster mit POKE definieren, d) Erst danach Clear Screen im neuen Fenster und Daten dort einschreiben. Alle übrige Information außerhalb des Fensters bleibt dann stehen, z.B. eine Führungsleiste mit Name, Straße, Ort usw..

Die TAB-Funktion

Sie ist im kleinen Anwenderhandbuch nicht beschrieben. Ein TAB wird gesetzt/gelöscht, durch gleichzeitiges Betätigen der Tasten SHIFT und TAB (Toggle-Funktion). Angesteuert wird ein TAB durch die Taste TAB oder per Programm durch PRINT CHR$(9); "Otto". - Nicht nur im Programmverlauf hat diese Funktion ihren Wert, auch beim Editieren von Programmen. Indem man sich anfangs TABs setzt, bewegt man sich schneller über die Zeilen zur Korrektur. Die TAB-Stops werden als Bitmuster in einer Tabelle ab hex 3A1 verwaltet. - Jeweils beim Übergang von BASIC zum Monitor und umgekehrt werden (leider) alle gesetzten TABs gelöscht.

Eine ebenfalls binäre Tabelle ab hex E2-E4 kennzeichnet beim LIST Zeilen ggfs. als Mehrfachzeilen. Die Zero Page ab E5 - FF scheint oberhalb dieser Tabelle frei für Zwecke des Anwenders zu sein.

Außer mit den Tasten ESC-L und -M (Enable/disable Scrolling) kann man Einfluß auf des Scrolling der Bildschirmzeilen auch durch POKE 923,0 oder ,1 nehmen. Beim Arbeiten mit Bildschirmfenstern wird man darauf achten.

INPUT vom Bildschirm

Es ist bekannt, daß das INPUT-Statement verschiedene Sonderzeichen nicht annimmt, die aber durchaus in Adreßdateien und Datenbanken benötigt werden. INPUT reagiert dann mit 'extra ignored' und hackt den String an der kritischen Stelle ab. Insbesondere in reinen Anwendersystemen kann das sehr lästig sein. Man umschifft das entweder durch eine spezielle Assembler-Routine oder durch die Benutzung des GET, das aber langweiliger und umständlicher ist. Wenn man eine Lösung sucht, so muß man auch bedenken, daß Strings, die solche Sonderzeichen enthalten, später auch wieder eingelesen werden müssen, z.B. von der Floppy. Ggfs. muß man dann auch hier mit GET arbeiten. - Bei der Eingabe beliebiger Strings vom Bildschirm her ist nicht unbedingt ein OPEN notwendig. Hierzu eine Möglichkeit:

```
10 SYS 34531: A=PEEK(14)-1: BANK 1 :REM INLIN, LÄNGE, PUFFER
20 FOR I=0 TO A: A$=A$+CHR(64094+I): NEXT :REM HOLE AUS PUFFER
30 BANK 15 :REM SYSTEMBANK
```

Das SYS 34531 ruft dabei die Routine INLIN auf, mit der eine Zeile bis zu 160 Zeichen vom aktiven Kanal eingegeben werden kann. Nach dem CR steht die aktive Länge in Zelle 14. Die nachfolgenden Programmzeilen holen die Bytes vom Eingabepuffer nacheinander in die Stringvariable. Dieser Puffer liegt in Bank 1 ab Adresse 64094.

Tastaturdekodierung, Tastaturbelegung

Wenn man EPROMs der 8K-Typen (2764) brennen kann, dann ist die Anpassung einer ASCII-Tastatur an deutsche Bedürfnisse nicht weiter schwierig. Auch die Tastatur des 710 ist 'dumm', sie enthält keine aktiven Bauteile, wird vielmehr matrixartig vom Rechner dekodiert. Der Schlüssel zur Dekodierung liegt im Festwertspeicher. Hier gibt es drei Haupttabellen: a) ab hex EA29 für normale Tastatur, ungeshiftet, ab EA89 dito geshiftet, b) ab EAE9 für Graphik und Shift sowie c) ab EB49 für ESC-Tasten. Hier finden wir jeweils in 6er-Gruppen ASCII- oder Hexbytes, mit denen die Kreuzungspunkte (=Tasten) dekodiert werden. Ein Eingriff an diesen Stellen führt zu einer anderen Tastaturbelegung. Die 6er-Gruppen beginnen links von der Tastatur her jeweils mit einer Funktionstaste. Ein hex FF bedeutet: keine Belegung. Die Shift-Taste wird mit 00 dekodiert. Bei Bedarf fällt es nicht schwer, die Zuordnung durch ein kleines Programm festzustellen:

```
10 GET A$: IF A$="" "THEN 10
20 PRINT ASC(A$);: GOTO 10
```

R.L.

○□○

Peter W. Arps, 2000 Hamburg 73

Mapping INPUT

für Commodore 64

Einleitung

Ziel war es, das Feldkonzept (protected/unprotected fields) des IBM-Systems 3270 auf dem Commodore 64 nachzubilden; d.h. es können nur Eingaben in Felder gemacht werden, die als Eingabefelder in einer Maske definiert worden sind. Die restlichen Stellen des Bildschirms sind für den Cursor nicht errreichbar.

Der Versuch, dies durch den GET-Befehl zu realisieren, führt zu keinem befriedigenden Ergebnis, da Felder nur sequentiell eingegeben werden können. Es ist also nicht möglich, die Felder wahlweise zu füllen. Ein weiterer Nachteil ist die geringe Geschwindigkeit der Verarbeitung. Den Anwendern des 'Datenbank'–Programmes für DM 99 eines bekannten Softwarehauses ist die Problematik bekannt.

Die Realisierung des Feldkonzepts durch den INPUT-Befehl scheidet aus, weil

1. *der Cursor über den gesamten Bildschirm zu bewegen ist,*
2. *nach 40 Zeichen in der Zeile eine Leerzeile eingefügt wird, die den Bildschirmaufbau zerstört,*
3. *die Insert-Taste das Gleiche bewirkt,*
4. *durch die CLR-Taste ebenfalls der Bildschirmaufbau zerstört wird,*
5. *nach Erreichen der 25. Zeile der Bildschirm nach oben verschoben wird,*
6. *jedes Feld einzeln durch CR übergeben werden muß und weil*
7. *eine Zeile nicht in mehrere Felder unterteilt werden kann.*

Es galt daher, dem BASIC die o.a. Eigenschaften 'abzugewöhnen'. Eine weitere Forderung war es, die maskengesteuerte Eingabe in das BASIC einzubinden, damit die Routinen nicht über SYS-Befehle aufgerufen werden müssen.

Realisierung

Wegen der besonderen Eigenschaften des C-64 war dies alles möglich. Er hat ein BASIC, das sehr

mit vom Benutzer änderbaren Vektoren arbeitet. Bei der Initialisierung des Programms Mapping-INPUT werden die Vektoren für OPEN und INPUT# auf dieses Programm gerichtet. Eine weitere Eigenschaft des C-64 ist es, BASIC und das Betriebssystem im RAM ablaufen zu lassen, das 'unter' dem ROM liegt. Gesteuert wird dies durch den Port der 6510-CPU. Von dieser Möglichkeit wird auch hier Gebrauch gemacht. - Mapping-INPUT stellt dem BASIC eine modifizierte Eingaberoutine zur Verfügung. Diese untersucht die Cursor-Position auf zur Eingabe zulässige Schreibstellen und stellt den Cursor ggfs. auf das nächste Eingabefeld.

Anwendung

Zuerst ist das Programm Mapping-Editor zu laden und mit SYS (49152)(A$,N$) aufzurufen. Die Variable N$ enthält den Maskennamen der zu ändernden Maske. Bei neuen Masken muß sie ein Leerstring sein. Die erstellte/geänderte Maske wird unter dem Namen dieser zweiten Variablen auf die Floppy gespeichert. Nach dem Aufruf kann die Maske beliebig erstellt werden. Eingabefelder werden durch C=§ (Commodore-Taste und Klammeraffe) gekennzeichnet. Ein Eingabefeld kann die Länge von 1-88 Stellen haben. Es sind über 100 Felder möglich. Die Beendigung des Editierens wird durch Shift/Return erreicht. Anschließend wird die Maske auf der Floppy gesichert. Die Farbinformationen werden ebenfalls gesichert. Ist in der Maske nicht mindestens ein Eingabefeld, so wird eine Fehlermeldung ausgegeben und die Maske nicht gesichert. - Das Editieren einer Maske kann mit der Stop-Taste abgebrochen werden.

Anschließend kann das Programm Mapping-INPUT geladen werden. Es ist mit SYS 49152 zu initialisieren. Innerhalb eines BASIC-Programmes wird die Maske durch Öffnen eines Files mit der Geräte-Nummer 3, der Sekundäradresse 0 und dem Maskennamen geladen (OPENx,3,0,"Name"). Nach dem CLOSEx kann eine andere Maske mit einem weiteren OPEN geladen werden. Soll nach einem CLOSEx ein OPEN für die gleiche Maske erforderlich sein, so dann dies durch OPENx,3,1 geschehen. Dieses OPEN darf nicht das erste sein (File Not Open Error).

Die Eingabe erfolgt mit dem INPUT#-Befehl. Es gilt die normale Syntax. Die Eingabe ist mit Shift/Return abzuschließen. Der Einhalt der Eingabefelder wird sequentiell in die Variablen übernommen. Je Feld ist eine Variable erforderlich. Dies gilt auch für leere Felder. Sind alle Feldinhalte übernommen, können erneut Eingaben gemacht werden. - Der INPUT#-Befehl beinhaltet kein Löschen des Bildschirms, damit bereits eingegebene Werte erneut übernommen werden können. Vor dem ersten INPUT# ist deshalb der Bildschirm ggfs. zu löschen. Die Insert- und Delete-Tasten wirken auch weiterhin über die gesamte Zeile. Bei mehreren Feldern in einer Zeile führt die Anwendung von INST/DEL zu Verschiebungen der nachfolgenden Feldinhalte. Die Maske bleibt aber unverändert. - Die Datei ist mit CLOSE zu schließen.

Das Quellenprogramm im MAE-Format sowie ein Beispielprogramm können beim Autor auf Diskette bezogen werden (Einzahlung von DM 25 auf Konto 3719 49-208 Postgiroamt Hamburg).

BASIC-AUFRUF: MAPPING-EDITOR UND MAPPING-INPUT PAGE 1

```
100 W=W+1: ON WGOTO 120,190,360
110 :
120 PRINT "CLR CD CD "
130 PRINT TAB(5)"******************************
140 PRINT TAB(5)"*  BILDSCHIRM-MASKEN-EDITOR  *
150 PRINT TAB(5)"******************************
160 :
170 LOAD *"EDIT.MAP",8,1
180 :
190 PRINT "HOME CD CD CD CD CD CD CD CD ": NI$=""
200 INPUT "CD RVS  INPUT MASKE  OFF      ";NI$
210 PRINT "CD RVS  OUTPUT MASKE OFF       ";NI$;: POKE 211,18: INPUT NO$
220 IF NO$="" THEN 190
230 X$=NO$
240 IF NI$=NO$ THEN X$="§:"+NO$
250 :
260 SYS (49152)(NI$,X$): REM  INPUT MASKE
270 :
280 :
290 PRINT "CLR CD CD "
```

```
300 PRINT TAB(5)"****************************
310 PRINT TAB(5)"*  BILDSCHIRM-MASKEN-INPUT  *
320 PRINT TAB(5)"****************************
330 :
340 LOAD *"INPUT.MAP",8,1
350 :
360 SYS (49152): REM  INITALISIEREN
370 OPEN 3,3,0,NO$: REM  LADE MASKE
380 PRINT "CLR "
390 INPUT# 3,A$,B$,C$,D$: REM  JE FELD EINE VARIABLE
400 INPUT# 3,A$,B$,C$,D$: REM  FELDINHALTE BLEIBEN ERHALTEN
410 CLOSE 3
420 :
430 OPEN 3,3,1: REM  OPEN OHNE NEUES ERNEUTES LADEN
440 INPUT# 3,A$,B$,C$,D$: REM  JE FELD EINE VARIABLE
450 CLOSE 3
```

```
0040 ; ********************
0050 ; *                  *
0060 ; *  MAPPING-EDITOR  *
0070 ; *                  *
0080 ; ********************
0090             ;
0100             .BA $C000
0110             ;
0120 R6510       .DE 1
0130 PNT         .DE $14
0140 VAR.PNT     .DE $47
0150 STATUS      .DE $90
0160 VER.FLG     .DE $93
0170 VON         .DE $AC
0180 BIS         .DE $AE
0190 SEC.ADR     .DE $B9
0200 DEVICE      .DE $BA
0210 ANZ.KEY     .DE $C6
0220 BLNK.SW     .DE $CC
0230 CRS.CHR     .DE $CE
0240 BLNK.FLG    .DE $CF
0250 QUOT.SW     .DE $D4
0260 LNG.ZEIL    .DE $D5
0270 CRS.ZEIL    .DE $D6
0280 INS.MODE    .DE $D8
0290 RPT.FLG     .DE $28A
0300 SCRL.DOWN   .DE $292
0310 BGR.COLR    .DE $287
0320 KERNAL      .DE $E000
0330 INS.ZEIL    .DE $E965
0340 PRINT       .DE $E716
0350 BLK.COL     .DE $EA13
0360 GET.CHRS    .DE $E5B4
0370 KL.AUF      .DE $AEFA
0380 KL.ZU       .DE $AEF7
0390 KOMMA       .DE $AEFD
0400 GET.VAR     .DE $AD9E
0410 OPEN.IEC    .DE $F3D5
0420 CLOSE.IEC   .DE $F642
0430 GET.STA     .DE $FFB7
0440 SET.LFS     .DE $FFBA
0450 SET.NAME    .DE $FFBD
0460 TALK        .DE $FFB4
0470 LISTEN      .DE $FFB1
0480 UNTALK      .DE $FFAB
0490 UNLISTEN    .DE $FFAE
0500 SA.SEND.T   .DE $FF96
0510 SA.SEND.L   .DE $FF93
0520 IEC.GET     .DE $FFA5
0530 IEC.PUT     .DE $FFA8
0540 MSG.SEA     .DE $F5AF
0550 MSG.LOAD    .DE $F5D2
0560 PGM.MSG     .DE $FF90
0570 ERROR       .DE $F710
```

```
                    0580 PGM.ERR     .DE $A43B
                    0590 SCREEN      .DE $400
                    0600 COL.RAM     .DE $D800
                    0610 NO.FOUND    .DE $F704
                    0620 STOP.TEST   .DE $FFE1
                    0630 NOT.HALT    .DE $F633
                    0640 PGM.STOP    .DE $A834
                    0650 ADD.1       .DE $FCDB
                    0660 END.TEST    .DE $FCD1          ;CY=1 :END
                    0670 CLEAR       .DE $E9FF
                    0680 MSB.ROUT    .DE $E6B6
                    0690 A.SAVE      .DI END
                    0700 B.SAVE      .DI END+3
                    0710 PR.TEXT     .DE $AB1E
                    0720             ;
                    0730             ;
                    0740             ;--------------------------
                    0750             ;-  VARIABLEN BEARBEITEN  -
                    0760             ;--------------------------
C000- A9 93         0770 START       LDA #$93           ;CLR
C002- 20 16 E7      0780             JSR PRINT
C005- 20 FA AE      0790             JSR KL.AUF         ;SYS(ADR)(A$,B$)
C008- 20 9E AD      0800             JSR GET.VAR        ;A$
C00B- A0 02         0810             LDY #2
C00D- B1 47         0820 LOOP1       LDA (VAR.PNT),Y
C00F- 99 F3 C1      0830             STA A.SAVE,Y
C012- 88            0840             DEY
C013- 10 F8         0850             BPL LOOP1
C015- 20 FD AE      0860             JSR KOMMA
C018- 20 9E AD      0870             JSR GET.VAR        ;B$
C01B- 20 F7 AE      0880             JSR KL.ZU
C01E- A0 02         0890             LDY #2
C020- B1 47         0900 LOOP2       LDA (VAR.PNT),Y
C022- 99 F6 C1      0910             STA B.SAVE,Y
C025- 88            0920             DEY
C026- 10 F8         0930             BPL LOOP2
C028- A9 40         0940             LDA #$40           ;AUSGABE MELDUNGEN
C02A- 20 90 FF      0950             JSR PGM.MSG
C02D- AD F6 C1      0960             LDA B.SAVE         ;FILE-NAME?
C030- D0 06         0970             BNE E0
C032- 20 10 F7      0980             JSR ERROR          ;FEHLT
C035- 4C 3B A4      0990             JMP PGM.ERR
                    1000             ;
C038- A9 00         1010 E0          LDA #0
C03A- 85 93         1020             STA *VER.FLG
C03C- AD F3 C1      1030             LDA A.SAVE         ;OLD FILE?
C03F- F0 3C         1040             BEQ NEW.MAP        ;NO
                    1050             ;
                    1060             ;-----------------
                    1070             ;-  MASKE LADEN  -
                    1080             ;-----------------
C041- AE F4 C1      1090             LDX A.SAVE+1       ;NAME
C044- AC F5 C1      1100             LDY A.SAVE+2
C047- 20 BD FF      1110             JSR SET.NAME FILE-NAME-PARAMETER
C04A- A9 08         1120             LDA #8
C04C- AA            1130             TAX
C04D- A0 60         1140             LDY #$60           ;SEC ADR
C04F- 20 BA FF      1150             JSR SET.LFS FILE-PARAMETER
C052- A6 B9         1160             LDX *SEC.ADR
C054- 20 AF F5      1170             JSR MSG.SEA        ;SEARCHING
C057- 20 D5 F3      1180             JSR OPEN.IEC       ;OPEN IEC-BUC
C05A- A5 BA         1190             LDA *DEVICE
C05C- 20 B4 FF      1200             JSR TALK
C05F- A5 B9         1210             LDA *SEC.ADR
C061- 20 96 FF      1220             JSR SA.SEND.T
C064- 20 D2 F5      1230             JSR MSG.LOAD       ;LOADING
C067- 20 A5 FF      1240             JSR IEC.GET        ;SKIP ADR

C06A- A5 90         1250             LDA *STATUS
C06C- 4A            1260             LSR A
C06D- 4A            1270             LSR A
```

```
C06E- 90 03     1280            BCC MXX
C070- 4C C6 C1  1290            JMP ASCHE
                1300 ;
C073- A0 00     1310 MXX        LDY #ADR1-ADR1
C075- 20 96 C1  1320            JSR UP.GET              ;SCREEN LADEN
C078- A0 08     1330            LDY #ADR3-ADR1
C07A- 20 96 C1  1340            JSR UP.GET              ;FARBRAM LADEN
                1350            ;
                1360            ;----------------------------
                1370            ;-  BETRIEBSSYSTEM INS RAM  -
                1380            ;----------------------------
C07D- A0 00     1390 NEW.MAP    LDY #L,KERNAL
C07F- B4 14     1400            STY *PNT
C081- A2 E0     1410            LDX #H,KERNAL
C083- B6 15     1420 E1         STX *PNT+1
C085- B1 14     1430 E2         LDA (PNT),Y
C087- 91 14     1440            STA (PNT),Y
C089- C8        1450            INY
C08A- D0 F9     1460            BNE E2
C08C- E8        1470            INX
C08D- D0 F4     1480            BNE E1
C08F- A9 60     1490            LDA #$60                ;UNTERDRUECKEN
C091- 8D 65 E9  1500            STA INS.ZEIL            ;LEER-ZEILEN
C094- 8D FF E9  1510            STA CLEAR               ;CLR SCREEN
C097- A9 A9     1520            LDA #$A9                ;LEER-ZEILEN
C099- 8D BB E6  1530            STA MSB.ROUT+5
C09C- A9 27     1540            LDA #$27
C09E- 8D BC E6  1550            STA MSB.ROUT+6
C0A1- 8D C2 E6  1560            STA MSB.ROUT+12
C0A4- A5 01     1570            LDA *R6510
C0A6- 29 FD     1580            AND #$FD                ;UMSCHALTEN
C0A8- 85 01     1590            STA *R6510
C0AA- AD BA 02  1600            LDA RPT.FLG
C0AD- 09 80     1610            ORA #$80                ;REPEAT ALLE TASTEN
C0AF- 8D BA 02  1620            STA RPT.FLG
                1630            ;
                1640            ;--------------------
                1650            ;-  INPUT-ROUTINE  -
                1660            ;--------------------
C0B2- A9 13     1670 HOME       LDA #19
C0B4- 20 16 E7  1680 INPUT      JSR PRINT
C0B7- A9 00     1690            LDA #0
C0B9- 85 D4     1700            STA *QUOT.SW
C0BB- 85 D8     1710            STA *INS.MODE
C0BD- A5 D6     1720            LDA *CRS.ZEIL
C0BF- C9 18     1730            CMP #24
C0C1- B0 EF     1740            BCS HOME
C0C3- A5 C6     1750 WAIT.LOOP  LDA *ANZ.KEY
C0C5- 85 CC     1760            STA *BLNK.SW
C0C7- 8D 92 02  1770            STA SCRL.DOWN
C0CA- F0 F7     1780            BEQ WAIT.LOOP
C0CC- 78        1790            SEI
C0CD- A5 CF     1800            LDA *BLNK.FLG
C0CF- F0 0C     1810            BEQ INPU1
C0D1- A5 CE     1820            LDA *CRS.CHR
C0D3- AE 87 02  1830            LDX BGR.COLR
C0D6- A0 00     1840            LDY #0
C0D8- 84 CF     1850            STY *BLNK.FLG
C0DA- 20 13 EA  1860            JSR BLK.COL
C0DD- 20 B4 E5  1870 INPU1      JSR GET.CHRS
C0E0- C9 03     1880            CMP #3                  ;STOP-TASTE
C0E2- F0 04     1890            BEQ INPU2               ;ENDE OHNE SAVE
C0E4- C9 8D     1900            CMP #$8D                ;SHIFT RETURN
C0E6- D0 CC     1910            BNE INPUT
C0E8- AA        1920 INPU2      TAX
                1930            ;
                1940            ;----------------------------
                1950            ;-  BETRIEBSSYSTEM  IM ROM  -
                1960            ;----------------------------
C0E9- A5 01     1970            LDA *R6510
C0EB- 09 03     1980            ORA #$03                ;ROM EIN
```

```
C0ED- 85 01      1990            STA *R6510
C0EF- AD BA 02   2000            LDA RPT.FLG
C0F2- 29 7F      2010            AND #$7F              ;REPEAT AUS
C0F4- 8D BA 02   2020            STA RPT.FLG
C0F7- E0 03      2030            CPX #3
C0F9- F0 66      2040            BEQ BREAK+3           ;STOP
                 2050            ;
                 2060            ;-------------------------
                 2070            ;-  TEST EINGABEFELDER  -
                 2080            ;-------------------------
C0FB- A9 00      2090            LDA #L,SCREEN
C0FD- A2 04      2100            LDX #H,SCREEN
C0FF- 85 14      2110            STA *PNT
C101- 86 15      2120            STX *PNT+1
C103- A2 04      2130            LDX #4
C105- A0 00      2140            LDY #0
C107- B1 14      2150 LOOP3      LDA (PNT),Y
C109- C9 64      2160            CMP #$64              ;C= @ (STRICH)
C10B- F0 12      2170            BEQ SAVE.MAP          ;IST DA
C10D- C8         2180            INY
C10E- D0 F7      2190            BNE LOOP3
C110- E6 15      2200            INC *PNT+1
C112- CA         2210            DEX
C113- D0 F2      2220            BNE LOOP3
C115- A9 CC      2230            LDA #L,TEXT
C117- A0 C1      2240            LDY #H,TEXT
C119- 20 1E AB   2250            JSR PR.TEXT
C11C- 4C 62 A4   2260            JMP PGM.ERR+39
                 2270            ;
                 2280            ;--------------------
                 2290            ;-  MASKE SICHERN  -
                 2300            ;--------------------
C11F- AD F6 C1   2310 SAVE.MAP   LDA B.SAVE            ;LAENGE
C122- AE F7 C1   2320            LDX B.SAVE+1          ;NAME
C125- AC F8 C1   2330            LDY B.SAVE+2
C128- 20 BD FF   2340            JSR SET.NAME FILE-NAME-PARAMETER
C12B- A9 08      2350            LDA #8
C12D- AA         2360            TAX
C12E- A0 61      2370            LDY #$61              ;SEC ADR
C130- 20 BA FF   2380            JSR SET.LFS FILE-PARAMETER
C133- 20 D5 F3   2390            JSR OPEN.IEC          ;OPEN IEC-BUC
C136- A5 BA      2400            LDA *DEVICE
C138- 20 B1 FF   2410            JSR LISTEN
C13B- A5 B9      2420            LDA *SEC.ADR
C13D- 20 93 FF   2430            JSR SA.SEND.L
C140- A0 04      2440            LDY #ADR2-ADR1
C142- 20 7E C1   2450            JSR UP.PUT
C145- A0 08      2460            LDY #ADR3-ADR1
C147- 20 7E C1   2470            JSR UP.PUT
C14A- 20 AE FF   2480            JSR UNLISTEN
C14D- A5 BA      2490            LDA *DEVICE
C14F- 20 B1 FF   2500            JSR LISTEN
C152- A5 B9      2510            LDA *SEC.ADR
C154- 29 EF      2520            AND #$EF
C156- 09 E0      2530            ORA #$E0
C158- 20 93 FF   2540            JSR SA.SEND.L
C15B- 4C AE FF   2550            JMP UNLISTEN
                 2560            ;
                 2570            ;--------------------
                 2580            ;-  UNTERROUTINEN  --
                 2590            ;--------------------
C15E- 20 33 F6   2600 BREAK      JSR NOT.HALT
C161- 4C 34 AB   2610            JMP PGM.STOP          ;BREAK
                 2620            ;
C164- A2 00      2630 SET.ADR    LDX #0
C166- B9 72 C1   2640 LOOP4      LDA ADR1,Y
C169- 95 AC      2650            STA *VON,X
C16B- C8         2660            INY
C16C- E8         2670            INX
C16D- E0 04      2680            CPX #4
C16F- D0 F5      2690            BNE LOOP4
C171- 60         2700            RTS
```

```
C172- FF 03     2720 ADR1       .SI SCREEN-1
C174- E8 07     2730            .SI SCREEN+1000
C176- FE 03     2740 ADR2       .SI SCREEN-2
C178- E8 07     2750            .SI SCREEN+1000
C17A- 00 D8     2760 ADR3       .SI COL.RAM
C17C- E8 DB     2770            .SI COL.RAM+1000
                2780            ;
C17E- 20 64 C1  2790 UP.PUT     JSR SET.ADR          ;VON/BIS
C181- A0 00     2800 PUT.LOOP   LDY #0
C183- B1 AC     2810            LDA (VON),Y
C185- 20 A8 FF  2820            JSR IEC.PUT
C188- 20 E1 FF  2830            JSR STOP.TEST
C18B- F0 D1     2840            BEQ BREAK
C18D- 20 DB FC  2850            JSR ADD.1 TO VON
C190- 20 D1 FC  2860            JSR END.TEST
C193- 90 EC     2870            BCC PUT.LOOP
C195- 60        2880            RTS
                2890            ;
C196- 20 64 C1  2900 UP.GET     JSR SET.ADR          ;VON/BIS
C199- A5 90     2910 GET.LOOP   LDA *STATUS
C19B- 29 FD     2920            AND #$FD             ;TIME-OUT LOESCHEN
C19D- 85 90     2930            STA *STATUS
C19F- 20 E1 FF  2940            JSR STOP.TEST
C1A2- F0 BA     2950            BEQ BREAK
C1A4- 20 A5 FF  2960            JSR IEC.GET          ;HOL BYTE
C1A7- A0 00     2970            LDY #0
C1A9- 91 AC     2980            STA (VON),Y
C1AB- A5 90     2990            LDA *STATUS
C1AD- 4A        3000            LSR A
C1AE- 4A        3010            LSR A
C1AF- B0 E8     3020            BCS GET.LOOP         ;BAD BYTE
C1B1- 24 90     3030            BIT *STATUS
C1B3- 70 09     3040            BVS GET.END          ;EOF
C1B5- 20 DB FC  3050            JSR ADD.1 TO VON
C1B8- 20 D1 FC  3060            JSR END.TEST
C1BB- 90 DC     3070            BCC GET.LOOP
                3075            .LS
C1BD- 60        3080 RTN        RTS                  ;DATEN GELADEN
                3090            ;
C1BE- 20 AB FF  3100 GET.END    JSR UNTALK
C1C1- 20 42 F6  3110            JSR CLOSE.IEC
C1C4- 90 F7     3120            BCC RTN              ;OK
C1C6- 20 04 F7  3130 ASCHE      JSR NO.FOUND         ;FILE NOT FOUND
C1C9- 4C 3B A4  3140            JMP PGM.ERR
                3150            ;
C1CC- 93 0D 4B  3160 TEXT       .BY $93 13 'KEINE EINGABE-FELDER '
C1CF- 45 49 4E
C1D2- 45 20 45
C1D5- 49 4E 47
C1D8- 41 42 45
C1DB- 2D 46 45
C1DE- 4C 44 45
C1E1- 52 20
C1E3- 49 4E 20  3170            .BY 'IN DER MASKE!!' 13 0
C1E6- 44 45 52
C1E9- 20 4D 41
C1EC- 53 4B 45
C1EF- 21 21 0D
C1F2- 00
                3180 END
                3190 ;          .EN

 END OF PASS
//0000,C1F3,C1F3
```

```
                0040 ; ******************
                0050 ; *                *
                0060 ; *  MAPPING-INPUT *
                0070 ; *                *
                0080 ; ******************
                0090             ;
                0100             .BA $C000
                0110             ;
                0120 R6510       .DE 1
                0130 DEVICE      .DE $13
                0140 VON         .DE $AC
                0150 PNT         .DE VON
                0160 NACH        .DE $AE
                0170 SEC.ADR     .DE $B9
                0180 ANZ.KEY     .DE $C6
                0190 VAR.LNG     .DE $C8
                0200 BLNK.SW     .DE $CC
                0210 BLNK.FLG    .DE $CF
                0220 CRS.CHR     .DE $CE
                0230 SCR.ADR     .DE $D1
                0240 COLUMN      .DE $D3
                0250 QUOT.SW     .DE $D4
                0260 LNG.ZEIL    .DE $D5
                0270 CRS.ZEIL    .DE $D6
                0280 FELD        .DE $D7
                0290 INS.MODE    .DE $D8
                0300 ADR         .DE $FB
                0310 CR.ALL      .DE $FD
                0320 CR.VAR      .DE $C9
                0330 FLD.PNT     .DE $FE
                0340 KERNAL      .DE $E000
                0350 INS.ZEIL    .DE $E965
                0360 PRINT       .DE $E716
                0370 BLK.COL     .DE $EA13
                0380 SCRL.DOWN   .DE $292
                0390 BGR.COLR    .DE $287
                0400 GET.CHRS    .DE $E5B4
                0410 SET.LFS     .DE $FFBA
                0420 PGM.ERR     .DE $A43B
                0430 SCREEN      .DE $400
                0440 COL.RAM     .DE $D800
                0450 STOP.TEST   .DE $FFE1
                0460 ADD.1       .DE $FCDB
                0470 END.TEST    .DE $FCD1
                0480 OS.LOAD     .DE $FFD5
                0490 OS.OPEN     .DE $F34A
                0500 OS.INPUT    .DE $F157
                0510 PRT.RTN     .DE $E674
                0520 CLEAR       .DE $E9FF
                0530 MSB.ROUT    .DE $E6B6
                0540 OPN.VEC     .DE $31A
                0550 INP.VEC     .DE $324
                0560             ;
                0570             ;
                0580             ;----------------------
                0590             ;-  VEKTOREN SETZEN  -
                0600             ;----------------------
C000- A9 3C     0610 INIT        LDA #L,OPEN
C002- A2 C0     0620             LDX #H,OPEN
C004- 8D 1A 03  0630             STA OPN.VEC

C007- 8E 1B 03  0640             STX OPN.VEC+1
C00A- A9 B5     0650             LDA #L,INP.START
C00C- A2 C0     0660             LDX #H,INP.START
C00E- 8D 24 03  0670             STA INP.VEC
C011- 8E 25 03  0680             STX INP.VEC+1
C014- A0 00     0690             LDY #L,KERNAL          ;OS INS RAM
C016- 84 AC     0700             STY *PNT
C018- A2 E0     0710             LDX #H,KERNAL
```

```
C01A- 86 AD     0720 IN1       STX *PNT+1
C01C- B1 AC     0730 IN2       LDA (PNT),Y
C01E- 91 AC     0740           STA (PNT),Y
C020- C8        0750           INY
C021- D0 F9     0760           BNE IN2
C023- E8        0770           INX
C024- D0 F4     0780           BNE IN1
C026- A9 60     0790           LDA #$60              ;UNTERDRUECKEN
C028- 8D 65 E9  0800           STA INS.ZEIL          ;LEERZEILEN
C02B- 8D FF E9  0810           STA CLEAR             ;CLR SCREEN
C02E- A9 A9     0820           LDA #$A9              ;LEERZEILEN
C030- 8D BB E6  0830           STA MSB.ROUT+5
C033- A9 27     0840           LDA #$27
C035- 8D BC E6  0850           STA MSB.ROUT+6
C038- 8D C2 E6  0860           STA MSB.ROUT+12
C03B- 60        0870           RTS
                0880           ;
                0890           ;
                0900           ;----------------------
                0910           ;-  OPEN/MASKE LADEN  -
                0920           ;----------------------
C03C- 20 4A F3  0930 OPEN      JSR OS.OPEN
C03F- 08        0940           PHP
C040- C9 03     0950           CMP #3                ;OPEN1,3,..?
C042- D0 41     0960           BNE RTN1              ;NICHT INPUT-MAPPING
C044- A5 B9     0970           LDA *SEC.ADR          ;ART TESTEN
C046- 29 0F     0975           AND #15
C048- D0 16     0980           BNE MAP.TEST          ;OPEN3,3,1
C04A- A9 08     0990           LDA #8                ;OPEN3,3,0,"NAME"
C04C- AA        1000           TAX
C04D- A0 00     1010           LDY #0
C04F- 20 BA FF  1020           JSR SET.LFS FILE-PARAMETER
C052- A9 00     1030           LDA #0                ;LOAD
C054- A2 00     1040           LDX #L,MAP.FLD
C056- A0 C3     1050           LDY #H,MAP.FLD
C058- 20 D5 FF  1060           JSR OS.LOAD           ;SCREEN + COL.RAM
C05B- 90 0D     1070           BCC INIT.MAP          ;GELADEN
C05D- 4C 3B A4  1080           JMP PGM.ERR           ;FILE NOT FOUND
                1090           ;
C060- AD C4 C1  1100 MAP.TEST  LDA INP.FLD           ;VALID MAP?
C063- 10 05     1110           BPL INIT.MAP          ;JA
C065- 28        1120           PLP
C066- 38        1130           SEC
C067- A9 03     1135           LDA #3                ;FEHLER-NR.
C069- 60        1140           RTS
                1150           ;
                1160           ;----------------------
                1170           ;-  MASKE BEARBEITEN  -
                1180           ;----------------------
C06A- A0 00     1190 INIT.MAP  LDY #ADR1-ADR1
C06C- 20 AC C1  1200           JSR SET.ADR
C06F- A2 00     1210           LDX #0
C071- A0 00     1220 LOOP1     LDY #0

C073- B1 AC     1230           LDA (VON),Y
C075- C9 64     1240           CMP #$64              ;C= '@'
C077- F0 0E     1250           BEQ ERM.FLD
C079- 20 DB FC  1260           JSR ADD.1
C07C- 20 D1 FC  1270           JSR END.TEST
C07F- 90 F0     1280           BCC LOOP1             ;WEITER
C081- A9 00     1290           LDA #0
C083- 85 FD     1300           STA *CR.ALL           ;INPUT
C085- 28        1310 RTN1      PLP
C086- 60        1320           RTS
                1330           ;
C087- A5 AC     1340 ERM.FLD   LDA *VON              ;STORE ADR
C089- 9D C5 C1  1350           STA INP.FLD+1,X
C08C- A5 AD     1360           LDA *VON+1
C08E- 9D C6 C1  1370           STA INP.FLD+2,X
C091- A9 00     1380 LOOP2     LDA #0
C093- 91 AC     1390           STA (VON),Y
```

```
C095- CB          1400              INY
C096- B1 AC       1410              LDA (VON),Y
C098- C9 64       1420              CMP #$64
C09A- F0 F5       1430              BEQ LOOP2
C09C- 98          1440              TYA
C09D- 9D C4 C1    1450              STA INP.FLD,X         ;LAENGE
C0A0- 18          1460              CLC
C0A1- 8A          1470              TXA
C0A2- 69 03       1480              ADC #3                ;NEXT FIELD
C0A4- AA          1490              TAX
C0A5- A9 00       1500              LDA #0
C0A7- 9D C4 C1    1510              STA INP.FLD,X         ;END-MERKER
C0AA- 98          1520              TYA
C0AB- 65 AC       1530              ADC *VON
C0AD- 85 AC       1540              STA *VON
C0AF- 90 C0       1550              BCC LOOP1
C0B1- E6 AD       1560              INC *VON+1
C0B3- D0 BC       1570              BNE LOOP1             ;IMMER
                  1580              ;
                  1590              ;
                  1600              ;--------------------
                  1610              ;-  MAPPING.INPUT  -
                  1620              ;--------------------
C0B5- A5 13       1630 INP.START    LDA *DEVICE
C0B7- C9 03       1640              CMP #3
C0B9- F0 03       1650              BEQ INPU0
C0BB- 4C 57 F1    1660              JMP OS.INPUT
                  1670              ;
C0BE- 98          1680 INPU0        TYA
C0BF- 48          1690              PHA
C0C0- 8A          1700              TXA
C0C1- 48          1710              PHA
C0C2- A5 FD       1720              LDA *CR.ALL
C0C4- D0 67       1730              BNE HOL.CHR           ;NEXT CHARACTER
C0C6- 20 87 C1    1740              JSR MOVE.MAP
                  1750              ;
C0C9- A5 01       1760              LDA *R6510
C0CB- 29 FD       1770              AND #$FD              ;OS UMSCHALTEN
C0CD- 85 01       1780              STA *R6510
                  1790              ;
                  1800              ;--------------------
                  1810              ;-  INPUT-ROUTINE  -
                  1820              ;--------------------
C0CF- A9 13       1830 HOME         LDA #19

C0D1- 20 16 E7    1840 INPUT        JSR PRINT
C0D4- 29 7F       1850              AND #$7F
C0D6- C9 14       1860              CMP #20               ;INS/DEL?
C0D8- D0 03       1870              BNE INPU4             ;NEIN
C0DA- 20 87 C1    1880              JSR MOVE.MAP
C0DD- A9 00       1890 INPU4        LDA #0
C0DF- 85 D4       1900              STA *QUOT.SW
C0E1- 85 D8       1910              STA *INS.MODE
C0E3- A6 D6       1920              LDX *CRS.ZEIL
C0E5- E0 18       1930              CPX #24
C0E7- B0 E6       1940              BCS HOME
C0E9- 18          1950              CLC
C0EA- A5 D1       1960              LDA *SCR.ADR          ;KORRESPONDIERENDES
C0EC- 85 AC       1970              STA *PNT              ;FELD DER MAP LADEN
C0EE- A5 D2       1980              LDA *SCR.ADR+1
C0F0- 69 BF       1990              ADC #H,MAP.FLD-SCREEN
C0F2- 85 AD       2000              STA *PNT+1
C0F4- A4 D3       2010              LDY *COLUMN
C0F6- B1 AC       2020              LDA (PNT),Y
C0F8- F0 04       2030              BEQ WAIT.LOOP         ;EINGABE-FELD
C0FA- A9 1D       2040              LDA #29               ;CRS RIGHT
C0FC- D0 D3       2050              BNE INPUT             ;IMMER
                  2060              ;
```

```
COFE- A5 C6      2070 WAIT.LOOP  LDA *ANZ.KEY
C100- 85 CC      2080            STA *BLNK.SW
C102- 8D 92 02   2090            STA SCRL.DOWN
C105- F0 F7      2100            BEQ WAIT.LOOP
C107- 78         2110            SEI
C108- A5 CF      2120            LDA *BLNK.FLG
C10A- F0 0C      2130            BEQ INPU1
C10C- A5 CE      2140            LDA *CRS.CHR
C10E- AE 87 02   2150            LDX BGR.COLR
C111- A0 00      2160            LDY #0
C113- 84 CF      2170            STY *BLNK.FLG
C115- 20 13 EA   2180            JSR BLK.COL
C118- 20 B4 E5   2190 INPU1      JSR GET.CHRS
C11B- C9 8D      2200            CMP #$8D              ;SHIFT RETURN
C11D- D0 B2      2210            BNE INPUT
                 2220            ;
C11F- A5 01      2230            LDA *R6510
C121- 09 03      2240            ORA #$03              ;ROM EIN
C123- 85 01      2250            STA *R6510
C125- A9 00      2260            LDA #0
C127- 85 C9      2270            STA *CR.VAR
C129- 85 FE      2280            STA *FLD.PNT
C12B- E6 FD      2290            INC *CR.ALL
                 2300            ;
                 2310            ;-----------------------
                 2320            ;-  ZEICHEN-UEBERGABE  -
                 2330            ;-----------------------
C12D- A4 C9      2340 HOL.CHR    LDY *CR.VAR           ;NEUES FELD?
C12F- D0 1F      2350            BNE HOL1              ;NEIN
C131- A6 FE      2360            LDX *FLD.PNT          ;NR FELD
C133- BD C4 C1   2370            LDA INP.FLD,X         ;LAENGE
C136- A8         2380            TAY
C137- BD C5 C1   2390            LDA INP.FLD+1,X       ;ADL
C13A- 85 FB      2400            STA *ADR
C13C- BD C6 C1   2410            LDA INP.FLD+2,X       ;ADH
C13F- 18         2420            CLC
C140- 69 41      2430            ADC #H,SCREEN-MAP.FLD
C142- 85 FC      2440            STA *ADR+1

C144- 88         2450 LOOP4      DEY                   ;FELD-ENDE ERMITTELN
C145- 30 26      2460            BMI HOL.NEU           ;LEER-FELD
C147- B1 FB      2470            LDA (ADR),Y
C149- C9 20      2480            CMP #'
C14B- F0 F7      2490            BEQ LOOP4
C14D- C8         2500            INY
C14E- 84 C8      2510            STY *VAR.LNG          ;SAVE ENDE
C150- A4 C9      2520 HOL1       LDY *CR.VAR           ;CURR. POS.
C152- C4 C8      2530            CPY *VAR.LNG          ;FELD-ENDE?
C154- B0 17      2540            BCS HOL.NEU           ;JA
C156- E6 C9      2550            INC *CR.VAR
C158- B1 FB      2560            LDA (ADR),Y           ;HOL ZEICHEN
C15A- 85 D7      2570            STA *FELD             ;IN ASCII
C15C- 29 3F      2580            AND #$3F
C15E- 06 D7      2590            ASL *FELD
C160- 24 D7      2600            BIT *FELD
C162- 10 02      2610            BPL HOL2
C164- 09 80      2620            ORA #128
C166- 70 02      2630 HOL2       BVS HOL3
C168- 09 40      2640            ORA #64
C16A- 4C 74 E6   2650 HOL3       JMP PRT.RTN           ;ZURUECK
                 2660            ;
C16D- A9 00      2670 HOL.NEU    LDA #0
C16F- 85 C9      2680            STA *CR.VAR           ;LOESCHEN
C171- 18         2690            CLC
C172- A5 FE      2700            LDA *FLD.PNT          ;INDEX ERHOEHEN
C174- 69 03      2710            ADC #3
C176- 85 FE      2720            STA *FLD.PNT
C178- AA         2730            TAX
C179- BD C4 C1   2740            LDA INP.FLD,X         ;ENDE?
C17C- D0 06      2750            BNE HOL4              ;NEIN
C17E- A9 00      2760            LDA #0
C180- 85 FE      2770            STA *FLD.PNT
```

```
C182- 85 FD     2780            STA *CR.ALL
C184- 4C 72 E6  2790 HOL4       JMP PRT.RTN-2       ;RETURN MIT CR
                2800            ;
                2810            ;
                2820            ;--------------------
                2830            ;-  UNTER-ROUTINEN  -
                2840            ;--------------------
C187- 18        2850 MOVE.MAP   CLC
C188- A0 06     2860            LDY #ADR3-ADR1
C18A- 20 90 C1  2870            JSR MOVE            ;SCREEN
C18D- 38        2880            SEC
C18E- A0 02     2890            LDY #ADR2-ADR1
                2900            ;
C190- 08        2910 MOVE       PHP
C191- 20 AC C1  2920            JSR SET.ADR
C194- 28        2930            PLP
C195- A2 04     2940            LDX #4              ;CY=1 ALLES
C197- A0 00     2950            LDY #0              ;CY=0 <>0
C199- B1 AC     2960 MOVE0      LDA (VON),Y
C19B- B0 02     2970            BCS MOVE1           ;STORE BYTE
C19D- F0 02     2980            BEQ MOVE2           ;SKIP BYTE
C19F- 91 AE     2990 MOVE1      STA (NACH),Y
C1A1- C8        3000 MOVE2      INY
C1A2- D0 F5     3010            BNE MOVE0
C1A4- E6 AD     3020            INC *VON+1
C1A6- E6 AF     3030            INC *NACH+1
C1A8- CA        3040            DEX
C1A9- D0 EE     3050            BNE MOVE0
C1AB- 60        3060            RTS
                3070            ;
C1AC- A2 00     3080 SET.ADR    LDX #0
C1AE- B9 BA C1  3090 LOOP3      LDA ADR1,Y
C1B1- 95 AC     3100            STA *VON,X
C1B3- C8        3110            INY
C1B4- E8        3120            INX
C1B5- E0 04     3130            CPX #4
C1B7- D0 F5     3140            BNE LOOP3
C1B9- 60        3150            RTS
                3160            ;
C1BA- 00 C3     3170 ADR1       .SI MAP.FLD
C1BC- E8 C6     3180 ADR2       .SI MAP.FLD+1000
C1BE- 00 DB     3190            .SI COL.RAM
C1C0- 00 C3     3200 ADR3       .SI MAP.FLD
C1C2- 00 04     3210            .SI SCREEN
                3220            ;
                3230 MAP.FLD    .DE INIT+$300
                3240            ; JE FELD 3 BYTES
C1C4- FF        3250 INP.FLD    .BY 255             ;LAENGE
C1C5- 00 00     3260            .SI 0               ;ADRESSE
                3270 ;          .EN

 END OF PASS
//0000,C1C7,C1C7
```

□○□

Bücher

Schnell, G; Hoyer, K.: Mikrocomputer-Interfacefibel. Braunschweig: Vieweg 1984, 175 S. DM 32,–ISBN 3 528 04248 6. Es handelt sich um ein hervorragend konzipiertes Lehr- und Arbeitsbuch, das für alle wichtigen Interfaces Schaltungsvorschläge und Interfaceprogramme enthält, die leicht verständlich und weitgehend prozessorunabhängig sind. Der Text ist klar formuliert und sollte eigentlich von jedem zu Rate gezogen werden, der vor einer Schnittstellenfrage steht. Wir haben folgende Hauptkapitel: Die Nahtstelle zwischen Hard- und Software, Passive Datenempfänger, Passive Datengeber, Aktive Datengeber, D/A- und A/D-Wandler, Geräte mit Quittungsverkehr, Externe Bussysteme, Interfacebausteine für Parallel- und Seriellbetrieb. Es werden auch der IEEE 488-Bus und der VMEbus abgedeckt.

Dipl.-Ing. Rüdiger Wollenberg, 4600 Dortmund 50

ROFF (3)

Textformatierung für 6502-Mikroprozessoren

(Anm. d. Hrsg.: Heft 33 enthielt die Beschreibung dieses Programms und die Liste seiner Befehle. Heft 34 brachte dann das Quellprogramm. Im vorliegenden Heft kommt der Aufsatz mit den Hintergründen des Programmaufbaues zum Abschluß.)

3.0 Programmaufbau

Im Hinblick auf eine gute Programmdokumentation hat sich in der letzten Zeit neben den Ablaufplänen und Struktogrammen die Beschreibung mittels Pseudocode durchgesetzt. Ein Pseudocode ist eine Hilfssprache, deren Sprachelemente nicht genormt sind, deren Umsetzung in eine konkrete Programmiersprache allerdings nur wenige Zeilen benötigt.

In diesem Artikel wurde nicht nur ein fertiges Programm vorgestellt, sondern vielmehr die Anregung zur individuellen Betätigung auf einem sehr interessanten Softwaregebiet eröffnet. Das hat die Notwendigkeit zur Folge, nicht nur das Assemblerprogramm zu veröffentlichen, sondern mit Hilfe das Pseudocodes eine didaktisch aufgebaute Einführung in das Formatierprogramm zu geben. Durch die ausführliche Beschreibung und den strukturierten Programmaufbau ist dann jeder einigermaßen erfahrene Programmierer in der Lage, eigene Ergänzungen für seine individuellen Bedürfnisse einzufügen, wenn die vorgestellten Kommandos nicht ausreichen sollten.

3.1 Pseudocode

Hauptziel der Verwendung des Pseudocodes ist die Benutzung von programmtechnischen Strukturelementen. So bedeutet die Schreibweise

```
for (i=0,i=10,i=i+1)
  S=S+i
  Q=Q+i*i
...,
```

daß die Schleife, die über Einrückungen gekennzeichnet ist, 11mal durchlaufen wird, wobei der Parameter i die Werte 0,1,2...10 annimmt. Der erste Ausdruck in der Klammer kennzeichnet den Startwert, der zweite den Endwert und der dritte die Veränderung des Laufparameters. Im Gegensatz zu BASIC wird die Schleife kein einziges Mal durchlaufen, wenn der Startwert größer als der Endwert ist.

Das Beispiel zeigt zwei weitere Konventionen. Kleingeschriebene Worte kennzeichnen Pseudosprachelemente und -variablen, wenn z.B. ein Laufindex durch ein Mikroprozessorregister verwirklicht wird, während groß geschriebene Worte auf tatsächlich existierende Variablen- und Unterprogrammnamen hinweisen. Die bekannten höheren Programmiersprachen besitzen für die Kennzeichnung des Blockendes spezielle Strukturbezeichnungen wie NEXT, ENDDO, ENDIF usw.. Hier ist die Blockung einzig und allein durch die Art der Einrückungen gekennzeichnet. Alle Zeilen gleicher Einrückungstiefe gehören zu einem Programmblock. - Eines der am meisten benutzten Strukturelmente ist die IF-THEN-ELSE Struktur:

```
if (INBUF(1)='.')
    call COMAND
else
    call TEXT  .
```

Der erste eingerückte Block wird ausgeführt, wenn die Bedingung nach dem if erfüllt ist, der zweite, wenn die Abfrage nicht zutrifft. Auf jeden Fall wird das Programm nach dem else-Block weiter fortgeführt. Der Hollerith-Bezeichner '.' wird als Kodierung für das entsprechende ASCII-Zeichen aufgefaßt. Weiterhin werden die logischen Symbole and und or verwendet. Für das Ungleich-Zeichen wird das () benutzt, wie z.B. in der while-Schleife

```
while (C()EOF).
```

Kommentare werden wie in der Assemblersprache durch ein Semikolon abgetrennt.

3.2 Verwendete Variablen

Die im folgenden aufgeführten Variablen haben für das Formatierprogramm eine zentrale Bedeutung und werden kurz vorgestellt.

Binärvariablen:

```
FILL  -Zeilen Auffüllen falls yes (0)          ;init=yes
SKIFLG-Überspringe Textausgabe falls yes(()0)  ;init=no
```

Zählvariablen:

```
LSVAL -laufender Zeilenzwischenraum            ;init=1
INVAL -laufende linke Einrückung               ;init=0
RMVAL -laufender rechter Rand                  ;init=PAGWID
TIVAL -temporäre linke Einrückung              ;init=0
CEVAL -Anzahl zu zentrierender Zeilen          ;init=0
ULVAL -Anzahl zu unterstreichender Zeilen      ;init=0
FAVAL -Anzahl Fett zu schreibender Zeilen      ;init=0
SPVAL -Anzahl der Leerzeilen                   ;init=1
PLVAL -Seitenlänge                             ;init=PAGLEN
CURPAG-laufende Seitennummer                   ;init=0
NEWPAG-nächste Seitennummer                    ;init=0
```

Bufferspeicher:

```
INBUF -Speichert Eingangszeile
OUTBUF-Speichert Text bei Zeilenauffüllung
WRDBUF-Speichert isolierte Worte zwischen
HEADER-Speichert Kopfzeile                     ;init=NEWLIN
FOOTER-Speichert Fußzeile                      ;init=NEWLIN
```

3.3. Unterprogramme

Im folgenden werden die verwendeten Unterprogramme im Pseudocode vorgestellt. Leider ist die benutze Typenradmaschine nicht in der Lage, die Kleiner- und Größerzeichen (< und >) darzustellen. Stattdessen mußten die runden Klammern verwendet werden.

Das Programm FORMAT initialisiert zuerst die verwendeten Variablen und liest dann über das Unterprogramm GETLIN die Eingabe zeilenweise und speichert sie in den Eingangsbuffer INBUF. Es erfolgt eine Zerlegung in Text und Formatierungskommandos.

```
FORMAT ;Hauptprogramm
  call INIT
  while (GETLIN(INBUF)()EOF)
    if (INBUF(1)='.')
      call COMAND(INBUF)
    else
      call TEXT(INBUF)
  if (LINENO()0)
    call SPACE(HUGE)
  stop
 end
```

Mit COMAND werden die Kommandos identifiziert und die entsprechenden Routinen angesprungen. Im Assemblerprogramm wird statt der linearen Programmierung eine Sprungleiste und Unterprogrammtechnik verwendet. Über GETVAL werden die Parameterwerte der Formatierkommandos vorher gelesen. Die Konstante HUGE steht symbolisch für eine große Zahl, hier 127.

```
COMAND(INBUF) ;Kommando-Interpretation
  VAL=GETVAL(INBUF,ARGTYP)
  if (INBUF(2)='F' and INBUF(3)='I')
    call BREAK
    FILL = yes
  else if (INBUF(2)='N' and INBUF(3)='F')
    call BREAK
    FILL = no
  else if (INBUF(2)='B' and INBUF(3)='R')
    call BREAK
  else if (INBUF(2)='L' and INBUF(3)='S')
    call SET(LSVAL,VAL,ARGTYP,1,1,HUGE)
  else if (INBUF(2)='S' and INBUF(3)='P')
    call SET(SPVAL,VAL,ARGTYP,1,0,HUGE)
    call SPACE(SPVAL)
  else if (INBUF(2)='U' and INBUF(3)='L')
    call SET(ULVAL,VAL,ARGTYP,1,0,HUGE)
    call UNDON
  else if (INBUF(2)='H' and INBUF(3)='E')
    i=1
    while (INBUF(i)()BLANK and INBUF(i)()NEWLIN)
      i=i+1
    while (INBUF(i)=BLANK)
      i=i+1
    if (INBUF(i)=''' or INBUF(i)='"')
      i=i+1
    j=1
    while (INBUF(i)()NEWLIN)
      HEADER(j)=INBUF(i)
      j=j+1
      i=i+1
    HEADER(j)=INBUF(i)
  else if (INBUF(2)='F' and INBUF(3)='O')
    i=1
    while (INBUF(i)()BLANK and INBUF(i)()NEWLIN)
      i=i+1
    while (INBUF(i)=BLANK)
      i=i+1
    if (INBUF(i)=''' or INBUF(i)='"')
      i=i+1
    j=1
    while (INBUF(i)()NEWLIN)
      FOOTER(j)=INBUF(i)
      j=j+1
      i=i+1
    FOOTER(j)=INBUF(i)
  else if (INBUF(2)='P' and INBUF(3)='L')
    call SET(PLVAL,VAL,ARGTYP,PAGLEN,
             M1VAL+M2VAL+M3VAL+M4VAL+1,HUGE)
    BOTTOM=PLVAL-M3VAL-M4VAL
  else if (INBUF(2)='I' and INBUF(3)='N')
    call SET(INVAL,VAL,ARGTYP,0,0,RMVAL-1)
    TIVAL=INVAL
  else if (INBUF(2)='R' and INBUF(3)='M')
    call SET(RMVAL,VAL,ARGTYP,PAGWID,TIVAL+1,HUGE)
  else if (INBUF(2)='T' and INBUF(3)='I')
```

```
    call BREAK
    call SET(TIVAL,VAL,ARGTYP,0,0,HUGE)
  else if (INBUF(2)='B' and INBUF(3)='P')
    if (LINENO()0)
      call SPACE(HUGE)
    call SET(CURPAG,VAL,ARGTYP,CURPAG+1,0,HUGE)
    NEWPAG=CURPAG
  else if (INBUF(2)='F' and INBUF(3)='A')
    call SET(FAVAL,VAL,ARGTYP,1,0,HUGE)
    call FATON
  else if (INBUF(2)='N' and INBUF(3)='E')
    call BREAK
    PARAM=0
    call SET(PARAM,VAL,ARGTYP,4,0,HUGE)
    lineno=max(LINENO,M1VAL+M2VAL)
    if (BOTTOM(=lineno)
      call SPACE(PARAM)
    else if (BOTTOM-lineno(PARAM)
      call SPACE(PARAM)
  else if (INBUF(2)='P' and INBUF(3)='P')
    call BREAK
    lineno=max(LINENO,M1VAL+M2VAL)
    if (BOTTOM(=lineno)
      call SPACE(PARAM)
    else if (BOTTOM-lineno(PARAM)
      call SPACE(PARAM)
    else
      call SPACE(1)
  else if (INBUF(2)='C' and INBUF(3)='E')
    call BREAK
    call SET(CEVAL,VAL,ARGTYP,1,0,HUGE)
  else if (INBUF(2)='B' and INBUF(3)='S')
    SKIFLG=yes
  else if (INBUF(2)='E' and INBUF(3)='S')
    SKIFLG=no
  return
  end
```

Das Unterprogramm GETVAL erfaßt die numerischen Parameter eines Formatierbefehles. Es sind neben Absolutparametern auch relative Angaben möglich. Die Funktion ctoi ist im Assemblerprogramm nicht als Unterprogramm abgebildet. Die Befehle, die den hexadezimalen Wert eines Parameters bestimmen, sind linear ausprogrammiert. In ARGTYP wird der erste Parameterwert für die spätere Verwendung gerettet, um den Unterschied von relativen und absoluten Datenangaben zu speichern.

```
GETVAL(INBUF,ARGTYP) ;Hole Commandoparameter
  i=1
  while (INBUF(i)()BLANK and INBUF(i)()NEWLIN)
    i=i+1 ;Anweisung übergehen
  while (INBUF(i)=BLANK)
    i=i+1 ;Blanks übergehen
  ARGTYP=INBUF(i)
  if (ARGTYP='+' or ARGTYP='-')
    i=i+1
  VAL=ctoi (INBUF,i) ;Parameterwert bestimmen
  return
  end
```

SET ersetzt den Parameter PARAM durch den gewünschten Wert, der dem eingegebenen absoluten, relativen oder vordefinierten entsprechen kann. Es wird keine Prüfung vorgenommen, ob der zuverlässige Wertebereich nicht überschritten ist..

```
SET(PARAM,VAL,ARGTYP,DEFVAL,MINVAL,MAXVAL) ;Parameterset.
  if (ARGTYP=NEWLIN)
    PARAM=DEFVAL ;Standardvorgabe
  else if (ARGTYP='+')
    PARAM=PARAM+VAL ;Relativ +
  else if (ARGTYP='-')
    PARAM=PARAM-VAL ;Relativ -
  else
    PARAM=VAL ;Absolut
  PARAM=min(PARAM,MAXVAL)
  PARAM=max(PARAM,MINVAL)
  stop
  end
```

Für das Verständnis des Seitenumbruchs werden an dem folgenden Bild die Variablen und Konstanten des Programms erläutert. Die erste Formularzeile wird nicht beschrieben. In die Zeile M1VAL wird die Kopfzeile gedruckt. Es folgen bis zur ersten Druckzeile M2VAL Leerzeilen. Der Wert der aktuellen Textzeile ist in der Variablen LINENO gespeichert. Die letzte Druckzeile hat den Wert BOTTOM, es folgen wieder M3VAL Leerzeilen, bis die Fußzeile gedruckt wird. Ein Formular besitzt insgesamt PLVAL Zeilen. Diese Variable ist per Formatierkommando änderbar, während die anderen Werte vor der Assemblierung festgelegt werden. Die Variable CURPAG enthält die gegenwärtige Seitennummer, während in NEWPAG die nächste abgelegt ist.

```
              1   ...
          M1VAL   Kopfzeile
                  ...
     M1VAL+M2VAL  ...
                  Druckanfang
                  .
                  .

      LINENO

                  .
         BOTTOM   Druckende
                  ...
    BOTTOM+M3VAL  ...
                  Fußzeile
BOTTOM+M3VAL+M4VAL ...
    = PLVAL
```

Die Beachtung der Formulargrenzen erfordert spezielle Ausgabeprogramme. SPACE erzeugt Leerzeilen, PUT sorgt für die Ausgabe des INBUF-Buffers und TUP für den Druck des Inhalts vom OUTBUT-Vektor.

```
SPACE(n) ;Leerzeilen ausgeben entsprechend ACCU-Inhalt
  call BREAK
  if (LINENO)BOTTOM)
    return
  if (LINENO=0)
    call PHEAD
  call SKIP (min(n,BOTTOM+1-LINENO))

  LINENO=LINENO+n
  if (LINENO)BOTTOM)
    call PFOOT
  return
  end
```

Die Routine SKIP ist sehr einfach aufgebaut. Es wird die Ausgabe von Freizeilen ohne jede Prüfung vorgenommen.

```
SKIP(n) ;Erzeugt leere Zeilen entsprechend ACCU-Inhalt
  for (i=1,i(=n,i=i+1)
    call PUTC(NEWLIN)
return
end
```

In der Routine PUT wird, wie schon erwähnt, die Ausgabe des INBUF-Vektors vorgenommen. Zusätzlich werden aber auch eventuelle Einrückungen ausgeführt, Zeilenabstände gesetzt und Zeilennummern vergeben. Am Blattanfang wird wie in SPACE der Ausdruck der Kopfzeile eingesteuert, während am Blattende die Ausgabe der Fußzeile veranlaßt wird.

Im Assemblercode sind zwei weitere Funktionen TUP und TUPLIN aufgenommen worden, die in der Wirkung dem PUT und dem PUTLIN gleichkommen, sich aber auf den Zeilenbuffer OUTBUF statt INBUF beziehen. Es gibt sicher bessere Möglichkeiten, um die Parameterübergabe auf Assemblerebene zu optimieren. Die hier beschriebene Lösung zeichnet sich aber durch einen einfachen Aufbau aus.

```
PUT(INBUF) ;INBUF mit Zwischenräumen und Einrück. ausgeb.
  if (LINENO=0 or LINENO)BOTTOM)
    call PHEAD
  for (i=1,i(=TIVAL,i=i+1)
    call PUTC(BLANK)
  TIVAL=INVAL
  call PUTLIN(INBUF)
  call SKIP(min(LSVAL-1,BOTTOM-LINENO))
  LINENO=LINENO+LSVAL
  if (LINENO)BOTTOM)
    call PFOOT
  return
  end
```

Die PUTLIN bzw. TUPLIN-Routinen besitzen praktisch nur eine for-Schleife.

```
PUTLIN(INBUF) ;Gibt INBUF aus
  for (i=1,INBUF(i)()NEWLIN,i=i+1)
    call PUTC(INBUF(i)
  call PUTC(NEWLIN)
  return
  end
```

Die Routine PUTC sorgt für die Ausgabe eines ASCII-Zeichens im Accumulator auf den Drucker. Sie ist für die unterschiedlichen Rechensysteme zu modifizieren. Per Formatierkommando kann der laufende Ausdruck übersprungen werden. Ein Blankersatz wird zusätzlich hier in in BLANK umgesetzt.

```
PUTC(n) ;Schreibt den Accuinhalt auf den Drucker
  call PHXY
  if (SKIFLG=no)
    if (ACCU=SEMIBL)
      ACCU=BLANK

       call OUTALL
    return
    end
```

PHEAD steuert die Ausgabe des Formularkopfes und aktualisiert die Seitennummer. Es werden nach dem oben beschriebenen Muster die Freizeilen und der Druck der Kopfzeile vorgenommen. PFOOT übernimmt die gleiche Aufgabe für den Formularfuß. Um ein einheitliches Druckbild zu gewährleisten, wird in beiden Fällen der Unterstreich- und Fettschreibmodus direkt ausgeschaltet und nach dem Ausdruck bei Bedarf wieder eingeschaltet. Der Platzhalter PAGNUM='#' bewirkt, daß an dieser Stelle die aktuelle Seitennummer eingesetzt wird. Im Maschinenprogram ist dafür die Routine PUTDEC eingeführt worden, die den Wert einer Speicherzelle dezimal ausgibt.

```
PHEAD ;Seitenkopf ausgeben und Seitennummer erhöhen
  call UNDOFD
  call FATOFD
  CURPAG=NEWPAG
  NEWPAG=NEWPAG + 1
  if (M1VAL)0)
    call SKIP(M1VAL-1)
    for (i=1,HEADER(i)()NEWLINE,i=i+1)
      if (HEADER(i)=PAGNUM)
        call PUTDEC(CURPAG)
      else
        call PUTC(HEADER(i))
  call SKIP(M2VAL)
  LINENO=M1VAL+M2VAL+1
  if (ULVAL)0)
    call UNDOND
  if (FAVAL)0)
    call FATVAL
  return
  end

PFOOT ;Seitenfuß ausgeben
  call UNDOFD
  call FATOFD
  call SKIP(M3VAL)
  if (M4VAL)0)
    for (i=1,FOOTER(i)()NEWLIN,i=i+1)
      if (FOOTER(i)=PAGNUM)
        call PUTDEC(CURPAG)
      else
        call PUTC(FOOTER(i))
    call SKIP(M4VAL-1)
  if (ULVAL)0)
    call UNDOND
  if (FAVAL)0)
    call FATOND
  return
  end
```

Die Druckausgabe wird von TEXT gesteuert. Es werden die erforderlichen Einrückungen vorgenommen und die Textausgabe aktiviert. Im Nofill-Betrieb wird der INBUF-Vektor direkt ausgegeben. Im Fill-Modus wird mittels GETWRD wortweise eine Übergabe an PUTWRD ausgelöst. Paßt ein Wort nicht mehr in die Ausgabezeile hinein, wird eine Ausgabe erzwungen. - Ist der Fettschrift oder der Untersteichmodus eingeschaltet, muß geprüft werden, ob gerade die letzte Zeile in dieser Betriebsart ausgedruckt wurde, damit das Ausgabegerät ausgeschaltet werden kann.

```
TEXT ;Drucke Textzeilen
  if (INBUF(1)=BLANK or INBUF(1)=NEWLIN)
    call LEADBL(INBUF)
  if (CEVAL)0)
    call CENTER
```

```
    call PUT(INBUF)
    CEVAL=CEVAL-1
  else if (INBUF(1)=NEWLIN)
    call PUT(INBUF)
  else if (FILL=no)
    call PUT(INBUF)
  else
    for (I=1,GETWRD(INBUF,I,WRDBUF))0, )
      call PUTWRD(WRDBUF)
  if (ULVAL)0)
    ULVAL=ULVAL-1
    if (ULVAL=0)
      call UNDOFF
  if (FAVAL)0)
    FAVAL=FAVAL-1
    if (FAVAL=0)
      call FATOFF
  return
  end
```

Weist eine Eingabezeile führende Leerzeichen auf, oder handelt es sich um eine leere Zeile, wird durch LEADBL ein Zeilenabbruch vorgenommen und die Variable TIVAL gesetzt, so daß die Einrückungen vor der nächsten Ausgabe berücksichtigt werden.

```
LEADBL ;Führende Leerzeichen löschen und TIVAL setzen
  call BREAK
  for (i=1,INBUF(i)=BLANK,i=i+1)
  if (INBUF(i)()NEWLIN)
    TIVAL=i-1
  for (j=1,INBUF(i)()NEWLIN,j=j+1)
    INBUF(j)=INBUF(i)
    i=i+1
  INBUF(j)=NEWLIN
  return
  end
```

In der Zentrierroutine wird der Wert für TIVAL dermaßen angepaßt, daß der Text in der Mitte der Zeile ausgeschrieben wird.

```
CENTER ;Centrierroutine
  for (i=1,WRDBUF(i)()NEWLIN,i=i+1)
  if ((RMVAL+TIVAL-i)/2)0)
    TIVAL=(RMVAL+TIVAL-i)/2
  else
    TIVAL=0
  return
  end
```

Die Routinen GETWRD und PUTWRD arbeiten im Fill-Modus zusammen. GETWRD isoliert Worte, imdem es ein weiteres Wort ab der Position I von INBUF in den Wortpuffer WRDBUF überträgt. Es wird die Wortlänge im Accumulator zurückgegeben. Das Ende einer Zeile wird durch den Wert Null gekennzeichnet.

```
GETWRD(INBUF,I,WRDBUF) ;Wortübertragung
  i=I
  while (INBUF(i)=BLANK)
    i=i+1
  i=1
```

```
  while (INBUF(i)()NEWLIN and INBUF(i)()BLANK)
    WRDBUF(j)=INBUF(i)
    i=i+1
    j=j+1
  WRDBUF(j)=NEWLIN
  I=i
  ACCU=j-1
  return
  end
```

Die isolierten Worte werden von PUTWRD empfangen. Sollte das Wort nicht mehr in die Zeile hineinpassen, bricht PUTWRD die laufende Zeile ab und veranlaßt eine Ausgabe. Die Daten in WRDBUF werden anschließend in den Ausgabebuffer eingeschrieben.

```
PUTWRD(WRDBUF) ;Bringt WRDBUF nach OUTBUF
  W=1
  while (WRDBUF(W)()NEWLIN)
    W=W+1
  W=W-1
  LAST=W+OUTP+1
  LLVAL=RMVAL-TIVAL
  if (OUTW+W)LLVAL or LAST)=MAXOUT-4)
    LAST=LAST-OUTP
    NEXTRA=LLVAL-OUTW+1
    call SPREAD(OUTBUF,OUTP,NEXTRA,OUTWDS)
    if (NEXTRA)0 and OUTWDS)1)
      OUTP=OUTP+NEXTRA
    call BREAK
  for (i=1,WRDBUF(i)()NEWLIN,i=i+1)
    OUTBUF(OUTP+i)=WRDBUF(i)
  OUTP=LAST
  OUTBUF(OUTP)=BLANK
  OUTW=OUTW+W+1
  OUTWDS=OUTWDS+1
  return
  end
```

SPEAD verschiebt die Worte der Ausgabezeile nach rechts, wobei im Wechsel eine Leerzeichenauffüllung entweder von rechts oder von links vorgenommen wird.

```
SPREAD(OUTBUF,OUTP,NEXTRA,OUTWDS) ;Rechten Rand ausricht.
  DIR=1-DIR
  NE=NEXTRA
  NHOLES=OUTWDS-1
  i=OUTP-1
  j=min(MAXOUT-1,i+NE) ;Platz für NEWLIN lassen
  while (i(j)
    OUTBUF(j)=OUTBUF(i)
    if (OUTBUF(i)=BLANK)
      if (DIR=0)
        NB=(NE-1)/NHOLES+1
      else
        NB=NE/NHOLES
      NE=NE-NB
      NHOLES=NHOLES-1
      for ( ,NB)0,NB=NB-1)
        j=j-1
        OUTBUF(j)=BLANK
```

```
    i=i+1
    j=j+1
  return
  end
```

Ein Seitenabbruck wird dann erforderlich, wenn im Fill-Modus Daten in den OUTBUF-Buffer eingespeichert worden sind.

```
BREAK ;Seitenabbruch
  if (OUTP)0)
    OUTBUF(OUTP)=NEWLIN
    call PUT(OUTBUF)
  OUTP=0
  OUTW=0
  OUTWDS=0
  return
  end
```

Die Routine zur Steuerung der Escapesequenzen des Druckers ist im folgenden für die Unterstreichungseinschaltung aufgelistet. In UNDOND wird die Einschaltung direkt an der Maschine vorgenommen.

```
UNDON ;Unterstreichen einschalten
  if (OUTP=0)
    call UNDOND
  else
    OUTBUF(0(OUTP)=ESCAPE
    OUTBUF(OUTP+1)='t'
    OUTBUF(OUTP+2)=BLANK
    OUTP=OUTP+2
  return
  end

UNDOND ;Unterstreichen direkt einschalten
  call PUTC(ESCAPE)
  call PUTC('t')
  return
  end
```

3.4 Programmverflechtung

Die Programme sind folgendermaßen ineinander verschachtelt:

```
FORMAT
    INIT
    GETLIN
    COMAND
        GETVAL
        BREAK
        SET
        SPACE
        UNDON,FATON
            PUTC
    TEXT
        LEADBL
            BREAK
        CENTER
        PUT
            PHEAD
            PUTC
            PUTLIN
                PUTC
            SKIP
            PFOOT
        GETWRD
        PUTWRD
            SPREAD
            BREAK
        UNDOFF,FATOFF
            PUTC
SPACE
    BREAK
        TUP
            PHEAD
            PUTC
            TUPLIN
                PUTC
            SKIP
            PFOOT
    PHEAD,PFOOT
        UNDOFD,FATOND
            PUTC
        SKIP
        PUTDEC
            PUTC
        PUTC
        UNDOND,FATOND
            PUTC
    SKIP
        PUTC
```

Roland Löhr

Datenbank DBMS L84

Vorbemerkungen

In Hinsicht auf die Ausführungszeiten werden Datenbankprogramme wohl vorwiegend in Assembler oder in einer Compilersprache geschrieben. Die Inbetriebnahme eines CBM 710 mit seinen Programm- und Datenspeichern von jeweils etwa 64 KB warf beim Autor die Frage auf, ob man diese doch schon beachtlichen Speichermöglichkeiten nicht für eine residente Datenbank nutzen könne. Dabei interessierte auch, ob man in BASIC zu befriedigenden Bearbeitungs- und Antwortzeiten kommt.

Erste Versuche mit Daten und ihren notwendigen Verkettungsinformationen verliefen so ermutigend, daß nach und nach das hier vorgelegte Datenbanksystem entstand, mit dem man tatsächlich gespeicherte Information einzeln oder in Begriffspaaren, auch INSTRING, in Sekundenschnelle aufschlägt. Recht frühzeitig gab es folgende Planungsüberlegungen:

a) Information soll bei Veränderungen aller Art nicht sortiert werden müssen. Information soll so gespeichert bleiben, wie sie historisch gewachsen oder eingegeben worden ist. Stattdessen sollen bei Änderungen nur die Verknüpfungszeiger auf die Information 'umgehängt' werden, denn Sortierungen und Bewegungen (Move) sind zeitaufwendig.

b) Es sollte eine Kurzwegtechnik gefunden werden, so daß Daten nicht immer von Anfang an durchsucht werden müssen. Es wurde daher ein 'Key' für alle Anfangsbuchstaben der Suchbegriffe geschaffen (az%(j,a)), mit dem das Item mit der jeweils niedrigsten alphabetischen Sortierung sofort adressiert wird. Bei Orten findet man z.B. hier den Zeiger auf 'Aachen', von dem man sich schnell weiter über 'Aalen' zu 'Aschaffenburg' durchhangelt. Die Suche ist damit nach Beginnbuchstaben segmentiert.

c) Je nach Länge der Eintragungen/Begriffe sollen viele Datensätze aufschlagbar sein. Bei einem Datenspeicher von 64 KB, drei Begriffen und einer durchschnittlichen Länge von 10 Zeichen sind etwa 2000 Datensätze in residenter Datenhaltung möglich, womit man vielen Ansprüchen gerecht wird.

d) Das Datenbanksystem soll vielseitig und an Bedürfnisse anpaßbar sein. Jedes Informationssystem hat seine Kopfinformation z.B. mit Name/Ort/Rechnungsnummer usw.. Diese Kopfinformation soll im Text der Bedienerführung immer wieder auftauchen. Es soll dort also nicht einfach '1., 2. und 3.' stehen. In der Anlage der Variablen (Felder) ist weiterhin dafür Sorge zu tragen, daß man möglichst mühelos mit einer unterschiedlichen Zahl von Suchbegriffen und Attributen arbeiten kann, wenn man die Dialoge anpaßt.

e) Keine Information steht für ewig, schon wegen nach der Eingabe entdeckter Fehler oder weil sich Merkmale und Attribute ändern. Bei Veränderungen sind die Zeiger umzuhängen, entleerte Datensätze sollen mit neuer Information wieder gefüllt werden können.

f) Bei der Entwicklung eines Programmes muß man im weiteren Verlauf darauf achten, daß man auf möglichst allen Befehlsebenen eine gleichmäßige Bedienerführung erreicht. Funktionstasten sollten also gleichmäßig belegt sein. Außerdem sollte der Bildschirm eine Bedienerführung enthalten. Neben den Daten sind also die Bearbeitungsmöglichkeiten auf der jeweiligen Befehlsebene anzuzeigen.

Mit der vorstehenden Aufzählung wichtiger Ziele ist verbunden, daß sie nach und nach im Programm verwirklicht wurden. Es mag natürlich einzelne Punkte geben, wo sich ein Anwender anders geführt sehen möchte. Wichtig scheint zunächst, daß eine dokumentierte Logik vorgelegt wird, auf der man aufbauen kann, und daß man ein arbeitsfähiges System findet. – Das Programm wurde noch ohne ein 'Toolkit' oder 'SM-Kit' für den CBM 710 entwickelt, gleichwohl ist es weitgehend strukturiert und stellt im Sinne kurzer Bearbeitungszeiten zeitkritische Unterprogramme möglichst an den Anfang, die einmalige Initialisierung, die Menüs und die Kommentare an den Schluß. Es wurden etliche Besonderheiten des verbesserten BASIC ausgeschöpft, insbesondere

Zeilenlängen bis 160 Zeichen, IF..THEN..ELSE und an manchen Stellen auch DISPOSE und INSTRING. Diese Eigenheiten muß man auf anderen Maschinen etwas anders auffangen.

Datenbanken in der Anwendung

Datenbanken sind für alle Bereiche des Lebens eine nützliche Einrichtung, die inzwischen den meisten Betreibern eines Computers zur Verfügung stehen könnte. Es geht in erster Linie um das Wiederauffinden früher eingespeicherter Information. Hier liegt ihre Hilfe, denn Such-, Übersetzungsvorgänge sowie Zuordnungen gehören zu den zeitaufwendigsten Tätigkeiten im privaten und beruflichen Bereich.

Simples Beispiel einer Datenbank ist ein zweisprachliches Lexikon z.B. für Computerbegriffe in Englisch und in Deutsch. Eine Datenbank sollte eine Suche und Zuordnung in beiden Begriffsreihen zulassen und möglichst auch Attribute mitführen, wie 'umgangssprachlich, kaufmännisch, technisch' usw.. Datenbanken lassen sich ebenso für verwaltende Dateien, Inhaltsverzeichnisse für Sachgebiete usw. einrichten. Daneben gibt es natürlich Datenbanken, gegen deren Einrichtung und Benutzung sich der Bürgerprotest zu Recht wendet.

Aufbaumöglichkeiten

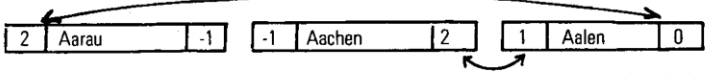
Beim Design einer Datenbank kann man verschiedene Wege gehen. Zumeist werden die Daten extern auf Floppy abgespeichert und für jeden Arbeitsgang des Suchens und Bearbeitens in den Rechner geholt. Das gilt auch für die mit den Daten verbundene Zeigerinformation. Für solche Datenbänke bietet sich die indexsequentielle Speicherungsform an, speziell auch die in relativen Dateien.

Hier wurde ein anderer Weg beschritten:Die Daten und Zeiger werden extern als sequentielle Datei gespeichert und bei der Inbetriebnahme der Datenbank komplett eingelesen. Am Ende des Arbeitstages oder zu einem anderen gewünschten Zeitpunkt wird die Information zurückgeschrieben, wenn eine Update-Variable auf eingetretene Veränderungen hinweist.

Die residente Haltung der Daten im Rechner hat den Vorteil, daß nicht immerzu auf die Diskette zugegriffen werden muß und daß die Information speziell beim Weiter- oder Zurückblättern schneller zur Verfügung steht. Eine Beschränkung ist darin zu sehen, daß man derzeit auf der Floppy eine größere Speichermöglichkeit hat, als im RAM des Rechners. Diese Beschränkung ist aber in Abhängigkeit von der Konfigurierung zu sehen. Inzwischen gibt es oft die Möglichkeit, weitere 64K-Bänke oder eine RAM-Floppy anzusprechen. Megabyte-Speicher sind bald keine Seltenheit mehr.

Auch ist es dem Anwender unbenommen, zusätzlich zur residenten Hauptinformation mit weiteren relativen Dateien zu arbeiten, die die Information ergänzen. Aus dem Index des laufenden Eintrages könnte man dabei direkt auf den RECORD übergehen.

Es wurde schon erwähnt, daß die eigentlichen Daten nicht sortiert werden. Stattdessen wird ein System von Zeigern geführt, daß die Information in auf- und absteigender Folge verkettet. Bei Änderungen jeder Art werden also nur die Zeiger wieder in Reihe gebracht. Man muß sich die Zeiger wie zwei Ösen an einem Datengefäß vorstellen: Die linke Öse weist auf den Vorgänger, die rechte auf den Nachfolger. Nachstehend ein Bild für eine Datenbank mit 3 Ortsnamen. Ein linker Zeiger mit –1 heißt: Es gibt keinen niedrigeren Vorgänger, linkes Ende der Tabelle. Ein rechter Zeiger mit –1 bedeutet entsprechend: Es gibt keinen höheren Nachfolger.



Die Verkettungsinformation wird in den Integer-Arrays (weil platzsparend) l%(i,j) und r%(i,j) geführt ('l' wie 'links', 'r' wie 'rechts'). Die Verkettung nach beiden Seiten erlaubt ein Vor- und Zurückblättern. Im Gegensatz dazu stehen anderswo schon veröffentlichte Zeigersysteme, die nur ein Weiterblättern erlauben; eine Beschränkung für den anwendernahen Datengebrauch.

Die vorliegende Datenbank erlaubt nach der Menüführung zwei Suchbegriffe, die unabhängig voneinander verkettet sind, sowie ein mitlaufendes Attribut (unverkettet). Dazu nachfolgend ein Beispiel für Daten und Zeiger:

Zeile	l%(i,0)	Eintrag	r%(i,0)	l%(i,1)	Eintrag	r%(i,1)	Attribut
0	2	Ida	1	-1	Bonn	2	5100
1	0	Max	-1	2	Rom	-1	I-1000
2	-1	Bert	0	0	Hof	1	8599

Die Array-Variablen sind N$(i,0) und N$(i,1) sowie M$(i) für das Attribut. Hier steht 'i' für den Laufindex, 0 für das erste Datenarray und 1 für das zweite. Diese Variablen werden in den Zeilen ab 46000 dimensioniert. Sofern man mit einer anderen Zahl von verketteten Suchbegriffen oder Atrributen arbeiten möchte, so liegen die notwendigen Eingriffe in das Programm vor allem hier. Und dann ist natürlich auch die Benutzerführung zu ändern. Die Logik der Verkettung in den Zeilen 200-510 und die des Blätterns und Suchens kann dann mit anderen Parametern für i und j unverändert übernommen werden. Das bedeutet Anpaßbarkeit an die Wünsche des Benutzers. - Für Anpassungen ist auch noch einmal auf das Array az%(j,a) hinzuweisen, das je Anfangsbuchstaben der Hauptsuchbegriffe auf das Item mit dem niedrigsten Eintrag hinweist.

Zur Abkürzung von Suchvorgängen bei den Variablen arbeitet das Programm mit nur etwa 30 Variablen, wobei den meisten eine feste Rolle zugeschrieben ist. Nur wenige, wie a, k und w werden für die unterschiedlichsten Aufgaben benutzt.

Leistungen der Datenbank, Menues und Bedienung

Im Sinne einer möglichst strukturierten Programmierung unter BASIC werden Entscheidungen in den Menues fast durchgehend nach dem Prinzip ON .. GOTO vollzogen. Das 'ON' ist dabei einer der jeweils erlaubten Tastendrücke.

Bei der Inbetriebnahme wird die Angabe des Tagesdatums verlangt, das später mit abgespeichert wird. Sodann erscheint das Hauptmenue, das in den Zeilen ab 40000 erzeugt wird. Es bietet die in nachstehender Tabelle gelisteten Bearbeitungsmöglichkeiten.

Hier soll nicht jedes Untermenue besprochen werden, weil die Benutzerführung meistens sehr einfach erfolgt. Wichtig sind vor allem die Suchmenues mit Tastendruck '4' und '9'. Ersteres verlangt die Vorgabe voller Begriffe, die im einfachen Stringvergleich aufgesucht werden können, auch paarweise. Ist ein Begriff aufgeschlagen, so kann der Datensatz wahlweise gedruckt werden. Man macht z.B. eine Liste aller Einträge zu 'Meier/Köln'. Vor und nach dem Ausdrucken kann in beiden Begriffsreihen vorwärts und rückwärts geblättert werden, um die benötigte Information besser sichten und einkreisen zu können. Natürlich ist auch eine Veränderung möglich. In diesem Fall kann die alte und die neue Information für das Änderungsjounal gelistet werden. Sofern ein Datensatz ganz gelöscht wird (er wird aus der Verkettung ausgehakt), wird automatisch das Attribut 'leer' vergeben. Durch eine Suche bei den Attributen mit 'leer' kann man Datensätze später neu füllen. Dieser Vorgang wurde nicht automatisiert, obwohl man das bei der Aufnahme weiterer Items (Befehl '2' des Hauptmenues) berücksichtigen könnte. -- Die Hauptgruppe '9' nutzt den INSTRING-Befehl des BASIC. Auch hier kann man paarweise suchen. Es ist bemerkenswert, wie schnell man auch bei schwierigen Suchvorgaben mit INSTRING zum Ziel kommt, speziell auch durch die Funktion 'W' zum Weitersuchen mit den gleichen Vorgaben. Auch hier können alle gefundenen Datensätze wahlweise gelistet werden.

Schlußbemerkungen, Anregungen

Nachdem das Grundgerüst nun aufgebaut ist, kann man sich zu Ergänzungen Gedanken machen: Der CBM 710 hat 20 Funktionstasten. Man könnte Sie für die Datenerfassung und -bearbeitung mit Standard-Texten belegen. Man wird ggfs. auch Programme schreiben, die eine vorhandene andere Datei in Datenbankbegriffe aufbereitet und sie zur automatischen Verkettung einliest, ohne daß Daten noch einmal getippt werden müssen. Es wird auch keine Schwierigkeiten bereiten, Merkmale (Attribute) in einem binären Array zu führen. Ein einfaches Integer-Array kann 16 Merkmale führen, in einem Stringarray kann man bei entsprechender logischer Aufbereitung fast

beliebig viele Merkmale führen. Man wird dann viel z.B. mit MID$ arbeiten. - Man findet leicht eine Benutzerführung, die das Abzählen von Bits überflüssig macht.

Das nachstehende Programm arbeitet mit dem INPUT-Befehl, womit dessen Beschränkungen wirksam bleiben. Bei Bedarf und Geduld ist es jedoch möglich, eine modifizierte Eingabe zu finden, die alle Zeichen akzeptiert, ohne daß man GET bemühen muß. - Im Sinne einer sicheren Benutzerführung wird man ferner von den Möglichkeiten des CBM 710 Gebrauch machen, mit Eingabefenstern zu arbeiten, aus denen der Anwender nicht herausspringen kann.

---

1 Datei anlegen. Hier erfolgen die Grundeintragungen zu den Suchbegriffen und zum maximalen Umfang der Datenbank. Die laufende Eingabe von Datensätzen schließt sich an. Diese wird abgebrochen, wenn in das erste Feld drei Klammeraffen eingegeben werden.

2 Items einfügen. Ab nächst freier Position werden neue Datensätze aufgenommen und ebenso wie bei 1 sofort verkettet.

3 Items löschen. Bei den beiden Gruppen der unabhängig verketteten Suchbegriffe kann ein Begriff aus der Verkettung ausgekoppelt werden, so daß er beim Suchen oder Blättern übergangen wird. Er bleibt aber über den zweiten Suchbegriff aufschlagbar z.B. für eine erledigte Angelegenheit. Er kann ggfs. mit der Veränderungsfunktion unter 4. und 9. wieder eingekoppelt werden.

4 Suchen. Die Suche erstreckt sich auf volle bekannte Begriffe. Das Menue und die Funktionen werden weiter unten erklärt.

5 Datei einlesen. Alle Variablen werden gelöscht. Es kann eine Datenbank von der Diskette geladen bzw. es kann bei laufendem Programm eine andere Datei geladen werden. Wurden in der zuerst geladenen Veränderungen vorgenommen, so wird zunächst zur Abspeicherung aufgefordert.

6 Datei abspeichern. Der Inhalt einer Datenbank wird einschließlich aller Zeiger auf Floppy abgespeichert, und zwar grundsätzlich in der Form von Strings.

7 Attribut ändern. Hier können nur Attribute auf einen neuen Stand gebracht werden.

8 Ausdrucken. Diese Funktion ist nicht implementiert. Sie kann je nach Bedarf ausgefüllt werden.

9 Instring-Suche. Es kann in allen Feldern einzeln oder auch zusammen in zweien nach Merkmalen gesucht werden. Zum Menue und zur Handhabung weiter unten.

Q Quit. Man Verläßt das Programm. Ein Wiederstart ist mit CONT unter BASIC möglich.

Tabelle: Bearbeitungsmöglichkeiten im Hauptmenue

Menue der Suchmöglichkeiten mit Taste 4 des Hauptmenues

Q Quit, Rückkehr zum Hauptmenue

2 Es werden zwei Begriffe zugleich in den beiden verketteten Arrays gesucht.

X Doppelte Suche im linken verketteten Array und bei den Attributen.

Y Doppelte Suche im rechten verketteten Array und bei den Attributen.

L Suche nur im linken Array
R Suche nur im rechten verketteten Array.
E Suche nur bei den Attributen.

W Weiterblättern im linken Array.
A Zurückblättern ebendort.

H Weiterblättern im rechten Array.
U Zurückblättern ebendort.

V Verändern von Items. Es können in allen drei Begriffen an laufender Stelle Veränderungen vorgenommen werden.

D Drucke Eintrag. Ausgabe einer Zeile auf Drucker (Protokoll).

9 Übergang zur Instring-Suche. Wie im Hauptmenue mit Taste 9.

Menue der Instring-Suche mit Taste 9 des Hauptmenues

2 Es werden zwei unabhängige Instrings zugleich in den beiden verketteten Arrays gesucht. Man kann den Beginnbuchstaben und die 1. Absuchstelle besonders angeben. Als Default sind der Buchstabe 'a' und Stelle 1 programmiert.

X Doppelsuche im linken Array und beim Atrribut, sonst wie bei '2'.

Y Doppelsuche wie vor mit dem rechten Array.

E Instringsuche nur bei den Attributen.

L Instringsuche im linken und
R im rechten Array

Q Quit. Rückkehr zum Hauptmenue.

Unter-Menue, wenn ein Item gefunden wurde:

D Drucke

W Weitersuchen mit alten Vorgaben.

V Eintritt in die Veränderungsfunktion.

N Neuen Instring vorgeben.

G Grundstellung der Instringsuche anspringen.

Q Quit, Rückkehr ins Hauptmenue.

Tabelle: Bearbeitungsmöglichkeiten in den Suchmenues

Programm-Liste zu DBMS L84

```
10 REM VERKETTETE DATEI COPYRIGHT 1984 BY ROLAND LOEHR
20 GOSUB49100:GOTO40000:REM SYSTEMDATUM, DANN MENUE
30 :
100 PRINT"";E$(J):INPUTN$:A=ASC(N$):IFN$="$$$"THENDISPOSEGOSUB:GOTO40000
110 RETURN
200 I=AZ%(J,A):IFI<>-1THEN300:REM ES GIBT HIER FRUEHERE EINTRAEGE
210 AZ%(J,A)=N:IF(L%(N,J)=-1ANDR%(N,J)=-1)THENRETURN:REM EINZIGER EINTRAG
220 K=A:REM UNTEN IM ALPHABET SUCHEN
230 K=K-1:IFK=-1THEN250:ELSEIFAZ%(J,K)<>-1THENI=AZ%(J,K):GOTO300:REM VERKNUEPFU
NGSMOEGLICHKEIT BEI NIEDRIGEREN ANFANGSBUCHSTABEN SUCHEN
240 GOTO230
250 K=A:REM OBEN IM ALPHABET SUCHEN
260 K=K+1:IFK=256THEN220:ELSEIFAZ%(J,K)<>-1THENI=AZ%(J,K):GOTO300:REM VERKNUEPF
UNGSMOEGLICHKEIT BEI HOEHEREN ANFANGSBUCHSTABEN SUCHEN
270 GOTO260
300 IFN$<>N$(I,J)THEN400:REM UNGLEICH
310 IFR%(I,J)=-1THENR%(N,J)=-1:L%(N,J)=I:R%(I,J)=N:RETURN:REM GLEICH/DAHINTER
320 RV=I:I=R%(I,J):IFN$<N$(I,J)THENR%(N,J)=R%(RV,J):L%(N,J)=RV:L%(I,J)=N:R%(RV,
J)=N:RETURN:ELSEGOTO310:REM NACH RECHTS, BIS FOLGEELEMENT -1 ODER GROESSER
400 IFN$>N$(I,J)THEN500:ELSEAZ%(J,A)=N:REM ALPHA-ZEIGER UMHAENGEN WEIL <
410 IFL%(I,J)=-1THENL%(N,J)=-1:L%(I,J)=N:R%(N,J)=I:RETURN
420 RV=I:I=L%(I,J):IFN$>N$(I,J)THENR%(I,J)=N:L%(RV,J)=N:L%(N,J)=I:R%(N,J)=RV:RE
TURN:ELSE410:REM DAZWISCHEN, SOBALD GROESSER ALS VORGAENGER
500 IFR%(I,J)=-1THENR%(N,J)=-1:R%(I,J)=N:L%(N,J)=I:RETURN:ELSE RV=I:I=R%(I,J):R
EM DAHINTER HAENGEN WENN AM ENDE -1 ODER RECHTS WEITER SUCHEN
510 IFN$(I,J)>N$THENR%(RV,J)=N:L%(N,J)=RV:R%(N,J)=I:L%(I,J)=N:RETURN:ELSE500:RE
M EINHAENGEN, SOBALD KLEINER

600 K=0:IFN$=N$(I,J)THENK=1:RETURN:REM K = WAHRHEITSVARIABLE
610 I=R%(I,J)
620 IFI=-1THENDISPOSEGOSUB:PRINT"NICHTS GEFUNDEN":GOTO34053:REM AM ENDE
630 IFN$(I,J)>N$THENK=-1:RETURN:ELSEGOTO600:REM ABBRUCH/WEITER
700 GETK$:IFK$=""THEN700:ELSEK=ABC(K$):RETURN
710 :
1000 GETN$:IFN$=""THEN1000:ELSEIFN$="J"THENRETURN:ELSEDISPOSEGOSUB:GOTO40000
1010 GETK$:IFK$=""THEN1010
1020 FORK=1TOLEN(K1$):IFK$=MID$(K1$,K,1)THENDISPOSEFOR:ELSENEXT
1030 RETURN
2000 :

31000 GOSUB46000:REM INITIALISIERUNG
31010 L%(0,0)=-1:L%(0,1)=-1:R%(0,0)=-1:R%(0,1)=-1:REM ECKPUNKTE VORBESETZEN
31020 N=0:REM 1.ELEMENT FUER ANZEIGE
31050 PRINT"":GOSUB46250:N=-1
31157 :
31300 N=N+1:FORJ=0TO1:GOSUB100:N$(N,J)=N$:GOSUB200:PRINT:NEXT
31310 GOSUB100:M$(N)=N$
31320 GOSUB46250:GOTO31300
31399 :
32000 PRINT:GOSUB46250
32100 GOSUB49200:FORJ=0TO1:GOSUB100:N$(N,J)=N$:GOSUB200:PRINT:NEXT
32110 GOSUB100:M$(N)=N$
32150 N=N+1:GOTO32000
32199 :
33000 PRINT"SOLL EIN ITEM GELOESCHT WERDEN? J/N":GOSUB1000
33010 PRINT:PRINT"ERHOEHTE UMSICHT IST GEBOTEN":PRINT
33050 INPUT"DATENSATZ-NR. ?";K1$:I=VAL(K1$):GOSUB45000:REM ANZEIGE
33060 PRINT"WO/WIE SOLL GELOESCHT WERDEN BZW. QUIT"
33070 PRINT"L ";E$(0),"R ";E$(1)
33075 PRINT"2 ";E$(0);" UND ";E$(1)," ZUGLEICH LOESCHEN + ATTRIBUT = LEER"
33076 PRINT"N NEUESTEN EINTRAG KOMPLETT LOESCHEN","Q QUIT"
33080 K1$="LR2NQ":GOSUB1010:W=K:REM BEFEHL ENTSCHL.
33085 PRINT"SOLL WIRKLICH GEAENDERT WERDEN? ";GOSUB700:IFK$<>"J"THEN40000
33090 ONWGOTO33200,33300,33100,33150,40000:GOTO33080
33100 GOSUB33180:GOTO34053:REM AN STELLE I DOPPELT AUSHAENGEN
33150 L1=N-1:I=N-1:GOSUB33180:N=L1:GOTO34053:REM NEUESTER RECORD AUSGEHAENGT
33180 L1=N:N=I:J=0:GOSUB33405:N$(I,J)=" ":J=1:GOSUB33405:N$(I,J)=" ":M$(I)="LEE
R":N=L1:GOSUB49200:RETURN:REM AUSHAENGEN AN STELLE I
33200 J=0:GOTO33350
33300 J=1
```

```
33350 GOSUB33405:GOSUB49200:GOTO34053:REM ITEM AUSHAENGEN
33399 :
33405 A=ASC(N$(I,J)):W=AZ%(J,A):IFW<>1THEN33410:ELSEIFR%(I,J)=-1THENAZ%(J,W)=-1
:GOTO33410REM ALPHA-ZEIGER AUF 1.ZEILE ERMITTELN
33407 K=ASC(N$(R%(I,J),J)):IFK=ATHENAZ%(J,A)=R%(I,J):ELSEAZ%(J,A)=-1:REM UMHAEN
GEN DES ALPHA-ZEIGERS AUF NIEDRIGSTEN EINTRAG BZW. -1 = UNBESETZT
33410 IF(L%(I,J)=-1 AND R%(I,J)=-1)THEN RETURN:REM 1. ELEMENT
33420 IF(L%(I,J)<>-1 AND R%(I,J)<>-1)THEN33460:REM MITTENDRIN
33430 IFL%(I,J)=-1THENL%(R%(I,J),J)=-1:RETURN:REM KEIN VORGAENGER
33440 R%(L%(I,J),J)=-1:RETURN:REM KEIN NACHFOLGER, ITEM ABGEHAENGT
33460 L%(R%(I,J),J)=L%(I,J):REM IM NACHFOLGER
33470 R%(L%(I,J),J)=R%(I,J):RETURN:REM AUSHAENGEN IM VORGAENGER
33599 :
33699 :
34000 PRINT"ES GIBT FOLGENDE SUCHMOEGLICHKEITEN":PRINT
34010 PRINT"QUIT"
34020 PRINT"2 ";E$(0);" UND ";E$(1);" ZUSAMMEN VORGEBEN"
34025 PRINT"X ";E$(0);" UND ";E$(2);" ZUSAMMEN VORGEBEN"
34027 PRINT"Y ";E$(1);" UND ";E$(2);" ZUSAMMEN VORGEBEN"
34030 PRINT"L ";E$(0);" VORGEBEN ODER R ";E$(1)
34035 PRINT"W AUFSTEIGEND A ABSTEIGEND IN ";E$(0);" BLAETTERN"
34037 PRINT"U AUFSTEIGEND H ABSTEIGEND IN ";E$(1)
34039 PRINT"K LOESCHEN EINES ITEMS"
34040 PRINT"W AUFSTEIGEND A ABSTEIGEND IN ";E$(0);" BLAETTERN"
34041 PRINT"V VERAENDERN VON ITEMS
34043 PRINT"E ";E$(2);" ABSUCHEN   D DRUCKE EINTRAG"
34045 PRINT"9 UEBERGANG ZUR INSTRING-SUCHE"
34049 I=0:PRINT:GOTO34053:REM SKIP ON FIRST ENTRY
34050 PRINT:PRINT"ENDE      ";
34053 REM WIEDEREINSPRUNGSPUNKT
34056 PRINT" KOMMANDO EINGEBEN  2 X Y L R Q W A U H K V E D 9":PRINT
34058 K1$="2XYLRQWAUHKVED9":GOSUB1010
34060 ONKGOTO34100,34400,34450,34200,34300,40000,34500,34600,34700,34800,33060,
37200,34850,34098,39100
34070 GOTO34050
34098 GOSUB39990:GOTO34056:REM DRUCKEN/MENUE
34099 :
34100 J=0:GOSUB100:L1=AZ%(J,A):IFL1<>-1THENL1$=N$:GOTO34110:REM 1. BEGRIFF
34105 PRINT"KEIN EINTRAG ";:GOTO34053
34110 J=1:GOSUB100:L2=AZ%(J,A):IFL2=-1THEN34105:ELSEL2$=N$:REM 2. BEGRIFF
34120 W=0:I=L2:J=1:N$=L2$:REM ZUERST RECHTS SUCHEN
34122 GOSUB600:IFK=-1THEN34105:ELSEL2=I:W=K:REM K,W WAHRHEITSVARIABLE
34125 J=0:I=L1:N$=L1$:REM DANN RECHTS
34130 GOSUB600:IFK=-1THEN34105:ELSEL1=I:REM DEN STAND BEWAHREN
34140 IFI=L2THENL1=I:GOTO34190:REM BEIDE STIMMEN UEBEREIN
34145 IFW=1THEN34165:REM DANN LINKS AUFWAERTS SUCHEN
34150 L3=L2:REM NIEDRIGSTE UEBEREINSTIMMUNG IN L2 BEWAHREN
34154 I=L3:J=1
34155 N$=L2$:GOSUB610:REM RECHTS HOCHPRUEFEN
34160 IFI=L1THEN34190:ELSEIFK<>-1THEN34155:REM BIS ZUM ENDE RECHTS
34163 :
34165 J=0:I=L1:REM RUECKUEBERPRAGUNG FUER LINKS
34170 N$=L1$:GOSUB610:REM LINKS HOCHPRUEFEN
34175 IFI=L3THEN34190:REM BEIDE STIMMEN UEBEREIN
34178 IFK=-1THEN34105:REM KEIN NACHFOLGER
34180 L1=I:W=0:GOTO34150:REM RECHTS AB BASIS HOCHPRUEFEN
34185 :
34190 GOSUB45000:REM ANZEIGEN
34192 GOSUB39500:REM BEFEHLSMENUE + BEFEHL HOLEN
34195 ONKGOTO34198,34199,37200,40000,34100,34000
34196 GOTO34192
34198 GOSUB39990:GOTO34192:REM DRUCKEN, DANN MENUE
34199 W=0:L2=R%(L3,1):L1=R%(I,0):IFL1=-1ORL2=-1THEN34050:ELSEGOTO34154
34200 J=0:GOTO34900:REM LINKES ARRAY  ABSUCHEN
34300 J=1:GOTO34900
34399 :
34400 J=0:GOSUB100:L2$=N$:I=AZ%(J,A):IFI=-1THEN34105:REM LINKES ITEM GEF.?
34405 J=2:GOSUB100:L1$=N$:N$=L2$:REM ATTRIBUT HOLEN/SICHERN/RESTORE LINKES ITEM
34410 J=0:GOSUB600:IFK=-1THEN34050:REM ELSE LINKS GEFUNDEN
34420 IFL1$=M$(I)THEN34430
34425 I=R%(I,J):IFI=-1THEN34105:ELSE34410:REM OUT/WEITER
```

```
 34430 GOSUB45000:REM ANZEIGE
 34440 GOSUB39500:REM MENUE/BEFEHL
 34445 ONKGOTO34448,34425,37200,40000,34400,34000:GOTO34440
 34448 GOSUB39900:GOTO34440:REM DRUCKEN/MENUE
 34450 J=1:GOSUB100:L2$=N$:I=AZ%(J,A):IFI=-1THEN34105:REM LINKES ITEM GEF.?
 34455 J=2:GOSUB100:L1$=N$:N$=L2$:REM ATTRIBUT HOLEN/SICHERN/RESTORE LINKES ITEM
 34460 J=1:GOSUB600:IFK=-1THEN34050:REM ELSE LINKS GEFUNDEN
 34470 IFL1$=M$(I)THEN34430
 34475 LF=R%(I,J):IFI=-1THEN34105:ELSE34460:REM OUT/WEITER
 34480 GOSUB45000:REM ANZEIGE
 34490 GOSUB39500:REM MENUE/BEFEHL
 34495 ONKGOTO34498,34475,37200,40000,34400,34050:GOTO34490
 34498 GOSUB39900:GOTO34490:REM DRUCKEN/MENUE
 34499 :
 34500 J=0:RV=I:I=R%(RV,J):REM WEITERBLAETTERN LINKES ARRAY
 34510 IFI=-1THENI=RV:I=RV:GOTO34050:ELSEGOTO34980
 34600 J=0:RV=I:I=L%(RV,J):GOTO34510:REM ZURUECKBL. LINKS
 34700 J=1:RV=I:I=R%(RV,J):GOTO34510:REM WEITERBL. RECHTS
 34800 J=1:RV=I:I=L%(RV,J)rGOTO34510:REM ZURUECKBL. RECHTS
 34849 :
 34850 J=2:PRINT"ATTRIBUT EINGEBEN":GOSUB100:REM ATTRIBUT HOLEN/ABSUCHEN
 34855 INPUT"SUCHEN AB ITEM NR. ";K1$:I=VAL(K1$)-1
 34860 I=I+1:IFI=NTHEN34053:REM ABBRUCH AM ENDE
 34870 IFN$<>M$(I)THEN34860:REM NICHTS GEFUNDEN
 34875 GOSUB45000:REM ANZEIGEN
 34880 GOSUB39500:REM BEFEHLSMENUE UND BEFEHL
 34890 ONKGOTO34898,34860,37200,40000,34850,34000:GOTO34880
 34898 GOSUB39990:GOTO34880:REM DRUCKEN, DANN MENUE
 34899 :
 34900 GOSUB100:I=AZ%(J,A):REM LINKES/RECHTES ARRAY ABSUCHEN
 34905 IFI=-1THENPRINT"KEIN EINTRAG":I=N-1:GOTO34053
 34910 IFN$=N$(I,J)THEN34980
 34950 IFN$(I,J)>N$THENPRINT"KEIN EINTRAG":I=N-1:GOTO34053
 34955 RV=L%(I,J):I=R%(I,J)
 34960 IFI=-1THENI=RV:GOTO34050:ELSEGOTO34910
 34980 GOSUB45000:GOTO34053:REM ANZEIGE/MENUE
 34999 :
 35000 PRINT"LADEN EINER DATENBANK":PRINT
 35005 IFE7$="UPDATED"THENPRINT,"DIE DATEI MUSS ERST ABGESPEICHERT WERDEN!!! /TA
S1E":GOSUB700:GOTO40000
 35010 PRINT"DATENDISKETTE IN DRIVE 1 LEGEN"
 35020 PRINT"LADEBEREIT ?":GOSUB49300:REM "J" ABWARTEN
 35025 CLR:REM ALLE VARIABLEN LOESCHEN
 35027 GOSUB49100:REM DATUM HOLEN
 35030 PRINT:INPUT"NAMEN DER DATENBANK EINGEBEN";NA$
 35040 DOPEN#8,(NA$),D1:E7$="":REM UPDATE-FLAG ZURUECKNEHMEN
 35050 INPUT#8,E1$,E2$,E3$,E4$,E5$:REM BESCHREIBUNGEN
 35060 INPUT#8,K$:U=VAL(K$):INPUT#8,K$:N=VAL(K$):REM DIMENSIONS
 35070 GOSUB46040:REM ARRAYS FORMATIEREN
 35075 INPUT#8,E$(0),E$(1),E$(2):REM SUCHBEGRIFFE
 35080 FORI=0TON-1:INPUT#8,N$(I,0),N$(I,1),M$(I),K$,K1$,L1$,L2$:L%(I,0)=VAL(K$):
R%(I,0)=VAL(K1$):L%(I,1)=VAL(L1$):R%(I,1)=VAL(L2$):NEXT
 35090 FORI=0TO255:INPUT#8,K$,K1$:AZ%(0,I)=VAL(K$):AZ%(1,I)=VAL(K1$):NEXT:REM AL
PHA-POINTER
 35100 DCLOSE#8:GOTO40000
 35999 :

 36000 PRINT"ABSPEICHERN DER RESIDENTEN DATENBANK":PRINT
 36010 REM VARIABLE AB ZEILE 50070 ERKLAERT
 36050 PRINT"DATENDISKETTE IN DRIVE 1 LEGEN !!":GOSUB49300:GOSUB49200
 36060 IFE7$="UPDATED"THENNA$="$"+NA$
 36100 DOPEN#8,(NA$),D1,W:REM DATEINAME MAX 16 ZEICHEN, DRIVE 1
 36110 K$=CHR$(13):PRINT#8,E1$+K$;E2$+K$;E3$+K$;E4$+K$;E5$
 36120 PRINT#8,STR$(U)+K$;STR$(N)+K$;E$(0)+K$;E$(1)+K$;E$(2):REM PARAMETER ALS S
TRING
 36150 FORI=0TON-1:PRINT#8,N$(I,0)+K$;N$(I,1)+K$;M$(I)+K$;STR$(L%(I,0))+K$;STR$(
R%(I,0))+K$;STR$(L%(I,1))+K$;STR$(R%(I,1)):NEXT
 36160 FORI=0TO255:PRINT#8,STR$(AZ%(0,I))+K$;STR$(AZ%(1,I)):NEXT:REM ALPHA-ARRAY
 36170 DCLOSE#8:E7$="":NA$=RIGHT$(NA$,LEN(NA$)-1):GOTO40000
 36999 :
```

```
37000 PRINT"AENDERN EINES ATTRIBUTES":IFI>(N-1)THENI=N-1
37003 IFI=-1THENI=0
37005 GOSUB45000
37010 PRINT:PRINT"WELCHE ITEM-#":INPUTI:PRINT:IFI>N-1THEN37010
37020 GOSUB45000
37025 GOSUB46260:REM DRUCKEN?
37030 PRINT:INPUT"NEUES ATTRIBUT";N$
37035 PRINT:PRINT"RICHTIG ? J/N"
37037 K1$="JN":GOSUB1010:ON KGOTO37040,37010:GOTO37035
37040 M$(I)=N$:GOSUB45000:GOSUB46260:GOTO34053:REM DATUM UPDATE/DRUCKEN?
37200 GOSUB46260:REM DRUCKEN?
37205 PRINT"WO SOLL GEAENDERT WERDEN BZW. QUIT"
37210 PRINT"L ";E$(0),"R ";E$(1),"E ";E$(2),"QUIT"
37220 K1$="LRAQ":GOSUB1010
37230 ONKGOTO37300,37400,37250,40000
37250 J=2:GOTO37030
37300 J=0:GOTO37410
37400 J=1
37410 L1=N:PRINT"SOLL WIRKLICH GEAENDERT WERDEN? J/N"
37420 GOSUB700:IFK$<>"J"THEN34053:ELSEN=I:GOSUB33405:GOSUB100:REM HOLE TASTE UN
D STRING, AUSHAENGEN
37435 N$(I,J)=N$:L2=I
37440 GOSUB200:N=L1:I=L2:GOSUB45000
37450 GOSUB46260:GOSUB49200:GOTO34053:REM UPDATE
37999 :
38000 STOP
39000 IFE7$="UPDATED"THENPRINT:PRINT,"DIE DATEI MUSS ABGESPEICHERT WERDEN!!!"
39050 STOP:GOTO40000:REM WARMSTART MOEGLICH MIT CONT
39100 PRINT"              INSTRING-SUCHEN":PRINT
39115 REM WIEDEREINSPRUNGSPUNKT
39116 PRINT"2 ";E$(0);" UND ";E$(1);" ZUSAMMEN VORGEBEN"
39117 PRINT"X ";E$(0);" UND ";E$(2);" ZUSAMMEN VORGEBEN"
39118 PRINT"Y ";E$(1);" UND ";E$(2);" ZUSAMMEN VORGEBEN"
39120 PRINT"L ";E$(0),"R ";E$(1),"E ";E$(2),"Q QUIT"
39130 K1$="2XYLREQ":GOSUB1010
39150 ONKGOTO39600,39700,39800,39200,39300,39400,40000
39160 GOTO39115
39199 :
39200 J=0:GOTO39310
39300 J=1
39310 GOSUB100:GOSUB 39900:REM STRING HOLEN/ANFANGSBEDINGUNGEN
39320 A=INSTR(N$(I,J),N$,W):IFA<>0THENGOSUB45000:GOTO39340:REM GEFUNDEN
39330 I=R%(I,J):IFI=-1THENPRINT:PRINT"ENDE":PRINT:GOTO39116:ELSE39320:REM END/W
EITER
39340 GOSUB39500:REM BEFEHL HOLEN
39348 ONKGOTO39970,39330,37200,40000,39310,39115
39350 GOTO39340
39399 :

39400 J=2:GOSUB100:I=-1:REM ALLE ATTRIBUTE ABSUCHEN
39410 I=I+1:IFI=NTHENPRINT"ENDE DER TABELLE":GOTO39116
39420 A=INSTR(M$(I),N$,1):IFA=0THEN39410:REM WEITER SUCHEN
39430 GOSUB45000:REM ANZEIGEN
39440 GOSUB39500:REM MENUE/BEFEHL
39450 ONKGOTO39490,39410,37200,40000,39480,39100:GOTO39440
39480 GOSUB100:GOTO39400:REM NEUES ATTRIBUT ABSUCHEN
39490 GOSUB39990:GOTO39440:REM DRUCKE/NAECHSTER BEFEHL
39499 :
39500 PRINT:PRINT:REM MENUE, WENN EIN ITEM GEFUNDEN
39505 PRINT"D DRUCKEN    W WEITERSUCHEN    V VERAENDERN"
39510 PRINT"Q QUIT       N NEUER STRING    G GRUNDSTELLUNG DER SUCHE"
39530 K1$="DWVQNG":GOSUB1010:RETURN:REM BEFEHL ENTSCHLUESSELN
39599 :
39600 J=0:GOSUB100:L1$=N$:GOSUB39900:L2=W:REM RETTE STRING, BEGINN/STELLE
39610 J=1:GOSUB100:GOSUB39930:REM 2. STRING
39620 A=INSTR(N$(I,0),L1$,L2):IFA<>0THENGOTO39680:REM GEFUNDEN
39630 I=R%(I,0):IFI=-1THENPRINT:PRINT"ENDE":PRINT:GOTO39115:ELSE39620:REM END/W
EITER
39680 A=INSTR(N$(I,1),N$,W):IFA=0THEN39630:REM RECHTS NICHTS
39685 GOSUB45000:REM 2 UEBEREINSTIMMUNGEN
39690 GOSUB39500:REM MENUE + BEFEHL
```

```
 39695 ONKGOTO39698,39630,37200,40000,39600,39100:GOTO39690
 39698 GOSUB39990:GOTO39690
 39699 :
 39700 J=0:GOSUB100:L1$=N$:GOSUB39900:L2=W:REM RETTE STRING + PARMS
 39710 J=2:GOSUB100:GOSUB39930
 39720 A=INSTR(N$(I,0),L1$,L2):IFA<>0THENGOTO39780:REM GEFUNDEN
 39730 I=R%(I,0):IFI=-1THENPRINT:PRINT"ENDE":PRINT:GOTO39115:ELSE39720:REM END/W
EITER
 39780 A=INSTR(M$(I),N$,W):IFA=0THEN39730:REM HIER NICHTS
 39785 GOSUB45000:REM 2 UEBEREINSTIMMUNGEN
 39790 GOSUB39500:REM MENUE + BEFEHL
 39795 ONKGOTO39798,39730,37200,40000,39700,39100:GOTO39790
 39798 GOSUB39990:GOTO39870
 39799 :
 39800 J=1:GOSUB100:L1$=N$:GOSUB39900:L2=W:REM RETTE STRING + PARMS
 39810 J=2:GOSUB100:GOSUB39930
 39820 A=INSTR(N$(I,1),L1$,L2):IFA<>0THENGOTO39880:REM GEFUNDEN
 39830 I=R%(I,1):IFI=-1THENPRINT:PRINT"ENDE":PRINT:GOTO39115:ELSE39820:REM END/W
EITER
 39880 A=INSTR(M$(I),N$,W):IFA=0THEN39830:REM HIER NICHTS
 39885 GOSUB45000:REM 2 UEBEREINSTIMMUNGEN
 39890 GOSUB39500:REM MENUE + BEFEHL
 39895 ONKGOTO39898,39830,37200,40000,39800,39100:GOTO39890
 39899 :
 39900 PRINT"ANFANGSBUCHSTABEN EINGEBEN, AB DEM GESUCHT WERDEN SOLL/DEFAULT A"
 39910 GOSUB700:IFK=13THENK=65:REM DEFAULT 'A'
 39920 I=AZ%(J,K):IFK=256THENDISPOSEGOSUB:PRINT"ENDE":GOTO39115
 39925 IFI=-1THENK=K+1:GOTO39920
 39930 PRINT"SUCHEN AB BUCHSTABENSTELLE NR./DAUFAULT 1."
 39940 GOSUB700:W=K:IFW=13THENW=1:ELSEIFW=0THENW=1:REM DEFAULT
 39950 RETURN
 39970 GOSUB39990:GOTO39340:REM DRUCKEN, DANN MENUE
 39990 OPEN4,4:PRINT#4,N$(I,0),N$(I,1),M$(I):CLOSE4:RETURN:REM AUSDRUCKEN
 39999 :
 40000 PRINT"           MENUE DATENBANK ROLAND LOEHR":PRINT
 40010 I=0
 40025 PRINT,"";NA$,E7$:PRINT
 40050 PRINT"1 DATEI ANLEGEN":PRINT
 40100 PRINT"2 ITEMS EINFUEGEN":PRINT
 40200 PRINT"3 ITEM LOESCHEN":PRINT
 40300 PRINT"4 SUCHEN":PRINT
 40400 PRINT"5 DATEI EINLESEN":PRINT
 40500 PRINT"6 DATEI ABSPEICHERN":PRINT
 40600 PRINT"7 ATTRIBUT AENDERN":PRINT
 40700 PRINT"8 AUSDRUCKEN":PRINT
 40750 PRINT"9 INSTRING-SUCHE":PRINT
 40800 PRINT"Q QUIT":PRINT
 41000 K1$="123456780":GOSUB1010 .
 41100 ONKGOTO31000,32000,33000,34000,35000,36000,37000,38000,39000,39100
 41200 GOTO40000
 41299 :
 42000 PRINT"ES GIBT 2 UNABHAENGIGE SUCHBEGRIFFE + 1 MITLAUFENDEN":PRINT:RETURN
 43000 PRINT"NAME DER DATENBANK    ":INPUTNA$:PRINT

 43010 INPUT"ERSTMALIG ERSTELLT AM    ";E1$:E2$=E1$:PRINT
 43020 INPUT"ERSTELLT VON    ";E3$:PRINT
 43030 PRINT"BESCHREIBENDER TEXT":INPUTE4$:PRINT
 43040 INPUT"NAME DER GGFS. MITLAUFENDEN RELATIVEN DATEI";E5$
 43050 RETURN
 43500 INPUT"WIE HEISSEN DIE SUCHBEGRIFFE ";E$(0),E$(1):PRINT
 43510 INPUT"WIE HEISST ER MITLAUFENDE BEGRIFF   ";E$(2):PRINT:RETURN
 43599 :
 43800 PRINT:PRINT"              KEIN EINTRAG BITTE EINE TASTE DRUECKEN"
 43810 GETN$:IFN$=""THEN43810:ELSE RETURN
 45000 PRINT"ITEM-#    ";I;PRINT"";E$(0),,E$(1),,E$(2):PRINT
 45010 PRINTN$(I,0),,N$(I,1),,M$(I):RETURN
 45099 :
 46000 PRINT"ANLAGE EINER DATENBANK":PRINT
 46010 GOSUB43000:GOSUB42000:REM DIALOG FUEHREN
 46020 GOSUB43500
```

```
 46030 INPUT"MAXIMALE GROESSE DER DATENBANK  ";N$:U=VAL(N$)-1:REM UMFANG
 46035 GOSUB46040:GOSUB46080:RETURN
 46040 PRINT"ETWAS WARTEN BITTE"
 46050 DIMN$(U,1):DIMM$(U):REM 2 SUCHBEGRIFFE, NEBENBEGRIFF
 46060 DIML%(U,1):DIMR%(U,1):REM LINKE UND RECHTE POINTER
 46070 DIMAZ%(1,255):RETURN:REM POINTER DER BEGINNE NACH BUCHSTABEN
 46079 :
 46080 FORJ=0TO1:FORI=1TO255:AZ%(J,I)=-1:NEXTI,J:REM INITIALISIERUNG
 46090 RETURN
 46099 :
 46250 PRINT:PRINT"                DATENERFASSUNG,ABBRUCH MIT '$$$' - SATZ ",N:RET
URN
 46260 PRINT:PRINT"ZEILE DRUCKEN ? J/N":GOSUB700:IFK$="J"THENGOSUB39990:RETURN:E
LSERETURN
 49100 INPUT"EINGABE SYSTEMDATUM DD.MM.JJ   ";E6$:RETURN
 49200 E2$=E6$:E7$="UPDATED":RETURN:REM UEBERNAHME DES AENDERUNGSDATUMS
 49299 :
 49300 REM EINLEGEN DER DISKETTE ABWARTEN
 49310 PRINT:INPUT"FERTIG   J/N";N$
 49320 IFN$<>"J"THENDISPOSEGOSUB:GOTO40000:ELSE RETURN
 49999 :
 50000 REM N=NAECHST FREIER PLATZ IN DEN ARRAYS
 50005 REM I=LAUFVARIABLE, ZEIGER AUF DATENSAETZE
 50008 REM J=0...2, ZEIGER-AUF ARRAY-TEILE
 50009 REM K, W=ALLG. VARIABLE, WAHRHEITSVARIABLE
 50010 REM A=ASCII(N$) DES 1. BUCHSTABENS DER EINGABE
 50020 REM N$= LAUFENDER EINGABESTRING
 50030 REM N%(U,J)=ARRAY DER EINGEGEBENEN STRINGPAARE, SUCHTABELLE
 50040 REM L%(I,J)= ARRAY DER ZEIGER AUF VORELEMENT. -1: NIEDRIGSTES ELEMENT
 50050 REM R%(I,J)= DITO AUF FOLGEELEMENT, RECHTER ZEIGER, -1: ES FOLGT NICHTS
 50060 REM RV=ZEIGER AUF VOR-ELEMENT IM ELEMENT I
 50070 REM AZ%(J,A)=ALPHA-ARRAY FUER NIEDRIGSTEN EINTRAG UNTER BUCHSTABEN
 50080 REM E1$=DATUM DER ERSTMALIGEN ERSTELLUNG
 50090 REM E2$=DATUM DER LETZTEN VERAENDERUNG
 50100 REM E3$=NAME DES AUTORS DER DATENBANK
 50110 REM E4$=BESCHREIBENDER TEXT DER DATENBANK
 50115 REM E5$=NAME EINER GGFS. ANZUSPRECHENDEN RELATIVEN DATEI
 50117 REM E6$=TAGESDATUM
 50118 REM E7$=UPDATE-TEXT
 50120 REM U=MAXIMALER UMFANG DER DATEI, FUER DIMN$(U,J)
 50140 REM E$(0)= TEXT DES 1, SUCHBEGRIFFES 'LINKS'
 50150 REM E$(1)= TEXT DES 2. SUCHBEGRIFFES 'RECHTS'
 50160 REM E$(2)= TEXT DES MITLAUFENDEN BEGRIFFES, ATTRIBUTES
 50999 :
 60000 J=0:FORI=0TON-1:PRINT"ZEILE ";I,L%(I,J),R%(I,J),N$(I,J):NEXT:PRINT
 60100 J=1:FORI=0TON-1:PRINT"ZEILE ";I,L%(I,J),R%(I,J),N$(I,J),M$(I):NEXT:PRINT
 60200 END
READY.
```

Bücher

Schumny, H.: Mikroprozessoren 6502, 6800, 8080,Z80, 9900. Braunschweig, Vieweg 1983, 240 S. DM 48,--, ISBN 3 528 04235 4. Das Buch gliedert sich in einen Grundlagenteil (Zahlensysteme, Kodierungen usw.) und in einen Arbeitsteil, in dem vor allem die Befehlssätze des 6502 und des Texas 9900 gruppiert und verglichen werden. Es sind jeweils viele anschauliche Darstellungen enthalten. Der Rezensent glaubt aber, daß dieses Buch, das vorwiegend wohl aus der Schulungsarbeit des Autors hervorgegangen ist, bedauerlicherweise eigentlich zu spät kommt: Einerseits hat Texas die Produktion von Mikromputern eingestellt, so daß die 9900er wohl nur noch für Steuerungszwecke Interesse haben, andererseits sind die Beispiele unter dem Vorrang einer hexadezimalen Kodierung ausgeführt, wie wir sie bei Systemen wie KIM-1, Alpha 1 oder TM 990/189 noch zu beherrschen lernen mußten. Wenn wir auch diese lehrrreichen Zeiten nicht missen mögen: Aus heutiger Sicht sollte man mehr den Assembler oder den Gebrauch der Sprachen betonen.

Peckham, H.: BASIC für den IBM Personal Computer. Hamburg 1984 bei McGraw-Hill, 348 S., DM 35,--, ISBN 3 89028 001 3. Der Leser wird von den allereinfachsten Hantierungen am Gerät über viele einfache Beipiele mit dem BASIC dieses Computers vertraut gemacht. Es hat die Abschnitte: Einführung in Computer und BASIC, Bekanntwerden mit dem PC, Einführung in BASIC Grafik und Musik, Arithmetik, E/A, Entscheidungen und Verzweigungen, Schleifen und Funktionen (Unterprogramme), Indizierte Variable, Zeichenkettenvariable, Zufallszahlen. Jedes Kapitel ist didaktisch sauber gegliedert in Lernziele, Entdeckungsübungen, Diskussion, Programmbeispiele, Aufgaben. Der Stoff ist druck-graphisch sauber und lesbar gegliedert. Man kann dieses Buch dem erwachsenen Betreiber eines PC empfehlen, der als Erstanwender neu auf Sprache und Computer zukommt. Er wird zielsicher in die Anfangsgründe eingeweiht und nicht mit überflüssigem belastet, das den Lerneifer stören könnte.

Fox, A.; Fos, D.: BASIC ganz einfach. Hamburg 1984 bei McGraw-Hill, 264 S. DM 24,80, ISBN 3 89028 001 3. Es ist ein Buch für den ganz blutigen Anfänger, vielleicht mehr für einen jungen Leserkreis. Der Stoff ist erzählerisch dargestellt und von vielen ganz lustigen Zeichnungen aufgelockert, in der Hoffnung, daß der Leser bei der Stange bleibt. Es ist ein sorgfältig gesetztes und gut aufgemachtes Buch. Man hätte diesen Aufwand eigentlich mehr dem vorerwähnten für den IBM-PC gewünscht.

Vieweg Programmbibliothek Mikrocomputer: Nr. 7 PC-1500-Sammlung 1, Nr. 8 Programme für den PC-1251 und Nr. 9 PC-1500-Sammlung II, Braunschweig 1984, 70 bis 102 S., DM 19,80 bis DM 24,80. Für die beliebten Rechner werden BASIC-Programme veröffentlicht für mathematische und naturwissenschafliche Probleme, Graphische Darstellungen, Spiele (Nr. 7),für den kommerziellen und naturwissenschaftlichen Bereich (Nr. 8) und für Uhr, Kalender, Ranglisten, Testauswertungen, Notendurchschnitt, Geometrie (Bd. 9). Die Beschreibungen sind klar.

Kaier, E.: BASIC-Wegweiser für den Apple II/IIe. Braunschweig: Vieweg 1984, 200 S. mit zahlr. Abb. und 80 vollst. Programmen, DM 32,--, ISBN 3 528 04259 1. Das Buch vermittelt akktuelles Grundlagenwissen zur Datenverarbeitung, enthält einen kompletten Programmierkurs in Applesoft-BASIC und beschreibt ausführlich das DOS und die technischen Eigenschaften der Geräte, die mit dem Apple benutzt werden können. Mit seinem ausführlichen Sachwortregister und dem klar gegliederten Inhaltsverzeichnis dürfte dieses Buch eine wertvolle Ergänzung zu den Systemhandbüchern des Herstellers sein.

Heilborn, J.; Talbot, R.: VC 20 Anwenderhandbuch. Hamburg: McGraw-Hill Book Company, 388 S. DM 32,--, ISBN 3 89028 004-8. Es handelt sich um die deutsche Übersetzung eines entsprechenden amerikanischen Buches und es gleicht in seinem Aufbau anderen bewährten Büchern, z.B. dem 'PET/CBM Personal Computer Guide'. Wir finden folgende Kapitel: Vorstellung, Bedienung und Programmierung des VC 20, Spielsteuerungen, Grafiken, Klänge, Peripheriegeräte, Systemarchitektur, Speicherbelegung, Fehlermeldungen, BASIC-Statements und -Funktionen. Daneben sind viele Tabellen und Übersichten enthalten. Enthalten sind auch die IEC-Bus-Befehle für die Peripherie, insbesondere für die Floppy. Der Stoff ist grafisch übersichtlich angeordnet und gut lesbar. Er geht weit über den der Anwenderhandbücher vom Hersteller hinaus, so daß der Anfänger hier ein solides Übungsbuch und der Fortgeschrittene ein handliches Nachschlagewerk findet.

Kahlig, P : Assembler-Programmierung von Mikroprozessoren (8080, 8085, Z80) mit dem ZX81. Braunschweig, Vieweg 1983, 186 S. DM 38,--, ISBN 3 528 04238 9. Mit dem dort beschriebenen Poor Man's Development System wird eine Hand-Assemblierung auf dem Billigrechner möglich. Das ist sicher nicht komfortabel, es mag aber sehr lehrreich sein, etliche der etwa 100 Programmbeispiele durchzuarbeiten, die Lösungen aus dem Bereich der vier Grundrechenarten enthalten und damit verständlich sind. Alle Beispiele sind einmal für den 8080/8085 und daneben für den Z80 ausgeführt und diskutiert. Im Anhang sind auf etwa 35 Seiten die vergleichbaren Source-Statements der Prozessortypen einander gegenübergestellt, in aufsteigender Sortierfolge für 1 bis zu 4 Bytes sind die Objectcodes mit den zugehörigen Mnemonics aufgeführt als Hilfe für das Assemblieren und für das Disassemblieren.

Diesem Heft ist ein Prospekt des te-wi Verlages beigefügt, der den Lesern zur Beachtung empfohlen wird.

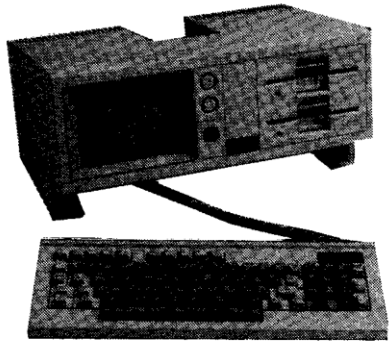
Aus der Branche

Commodore präsentierte neue Geräte auf der Hannover-Messe, insbesondere einen IBM-kompatiblen Commodore PC. Es handelt sich hier um einen Rechner mit CPU 8088, optionalem 8087 für Arithmetik, 256 KB RAM im Grundausbau (intern auf 512 KB erweiterbar,extern bis 764 KB) und zwei Floppies DS/DD mit je 320 KB Speichermöglichkeit. Zusammen mit einem 7"-Monitor sind alle Einheiten in einem formschönen Gehäuse untergebracht, das auch zum Transport eignet, die frei bewegliche Tastatur mit 84 Tasten wird dann in eine Nische des Hauptgehäuses geschoben. Zu den Merkmalen gehören weiterhin: Betriebssystem MS-DOS 1.25 in 24 KB, Möglichkeit, eine RAM-Floppy von 90 KB zu fahren, 40 oder 80 Zeichen/Zeile auf dem Bildschirm (per Software), voll grafikfähiger Bildschirm mit 320x200 Pixel bei 4 sw-Werten, 640x200 Pixel bei zwei sw-Werten. An Anschlüssen sind RS 232, Centronics sowie Systembus vorgesehen. Dieses Gerät soll im Werk Braunschweig hergestellt werden und bei DM 6000 kosten. Für den Spätherbst ist eine Desktop-Version vorgesehen mit einem größeren Bildschirm. In den ersten Datenblättern wird auch von einer Color-Karte gesprochen. - Ein Commodore-Händler konnte das Gerät kürzlich vorübergehend erproben und dabei feststellen, daß es auch vom IBM-PC beschriebene Disketten liest und verarbeitet, womit ein hoher Grad der Kompatibilität gegeben sein dürfte. - Der Vorstoß der Fa. Commodore in neue Gefilde trägt der Tatsache Rechnung, daß der IBM-PC weltweit Standards prägt. Mit seinem Preis ist dieses Produkt jedoch auch eine wettbewerbliche Herausforderung, speziell auch im deutschen Markt, wo Commodore seine führende Stellung behaupten möchte.

Commodore 8296: Dieses Gerät ist eine Aufwärtsentwicklung des erfolgreichen 8032 im SK-Gehäuse. Es hat 128 KB RAM in zwei Bänken, wobei sich die Festwertspeicher der Grundbank ganz oder teilweise ausblenden lassen. Das Betriebssystem ist LOS-96 und der Interpreter BASIC 4.0. Damit dürfte für vorhandene Programme des 8032/8096 Aufwärtskompatibilität gegeben sein. Dieses Gerät ist als Arbeitsplatzcomputer in den Betrieben vorgesehen.

Ebenfalls neu ist der CBM Z8000, ein Mehrplatzsystem für 2 und später 8 und mehr Arbeitsplätze. Die weiteren Merkmale sind: 256 KB RAM, 32 KB ROM-Standard, 128 KB Bildschirm-RAM, BASIC, PILOT, Assembler, C-Compiler optional, Bildschirm mit 1024x1024 Punkten, UNIX-7-angepaßtes Betriebssystem COHOS, IEEE 488-Interface für die Commodore-Peripherie, zwei RS 232-Schnittstellen, Anschluß von Maus und Coprozessor Z8070 sowie von externen Massenspeichern (Winchester, Floppy). - Commodore sieht für dieses Gerät große Chancen während der nächsten Jahre im kommerziellen Bereich.

Weitere Produktneuheiten bei Commodore: Home-Computer C16 (für Schüler und Studenten) mit 16 KB RAM, BASIC 2.5, Home- und Semiprofi-Computer C 264 für Heim- und Berufsanwendungen mit 64 KB RAM und BASIC 3.5, bis zu 32 KB zusätzliches ROM für Softwareoptionen. Neu sind auch der Color-Matrixdrucker MCS 801 (7 Farben) und der Typenraddrucker DPS 1101, beide für den seriellen Commodore-Bus ausgelegt. - Der C 64 kann jetzt auch zu etwa DM 250 Btx-fähig gemacht werden (Bildschirmtext). Als weitere Voraussetzungen werden ein Fernsehner mit CEPT-Dekoder und ein Floppy-Laufwerk genannt. - Commodore wird ferner seine Sachbuchreihe für Computer ausbauen.



Commodore PC

Central Computer Unit CCU 8086. Die ancon Ingenieurgesellschaft mbH in Berlin 33, Aug.-Viktoriastr. 63 präsentierte in Hannover ihre CCU 8086 am Apple, ITT 3030 und CBM 8032 unter CP/M 86 und MS-DOS 2.0. Es handelt sich um eine Karte mit CPU 8086, Urlader, PIO 8522 und 128 KB RAM in der Grundversion zu DM 1200. Auf der Karte kann bis 512 KB RAM erweitert werden, eine zweite Karte mit bis zu 512 KB kann über den Prozessorbus betrieben werden. Die CCU ermöglichst damit die Inbetriebnahme von Software, die für den IBM-PC entwickelt wurde, auf einem vorhandenen Host-Rechner, was in vielen Fällen interessant ist und den Vorteil des billigeren Aufbaues hat, weil die vorhandene Systemausstattung weiterbenutzt wird. Die Software wird auf Disketten für den Host geliefert. Die CCU kann auch als RAM-Floppy betrieben werden. Damit sind schnellere Zugriffe auf einen Massenspeicher möglich.

Editorial

Die diesjährige Hannover-Messe hatte einen sehr großen Zuspruch durch das Publikum. Insbesondere in den Hallen für Büro- und Kommunikationstechnik drängten sich die Besucher. Nach wenigen Jahren schon ist der Mikroprozessor für alle modernen Geräte bereits eine Selbstverständlichkeit geworden, und das Angebot ist sowohl für die Textbearbeitung wie aber auch für das eigentliche Computern und für das Messen und Regeln unübersehbar groß. Gleichwohl lassen sich einige Trends ablesen.

Textbearbeitungssysteme waren anfangs recht teuer. So lag es nahe, persönliche Computer um leistungsfähige Wordprozessorprogramme zu bereichern, um preiswerte Systemkonfigurationen aufzubauen. Ein Schönheitsfehler war es häufig, daß die zum Programm gehörenden Bedienungshandbücher keine ganz unkomplizierte Hantierung ermöglichten. Auch die Ausgabe auf Schönschreibdrucker war häufig an individuelle Anpaßschaltungen gebunden.- Heute stehen dem Computerbetreiber anschlußfertige reine Schreibwerke und Kombinationen zwischen Schreibwerk und Schreibmaschine zur Textbearbeitung zur Verfügung, meistens mit V24- oder Centronics-Schnittstelle, und in einer Preislage ab etwa DM 1000. Andererseits findet man seitens der Büromaschinenindustrie leistungsfähige Schreib-Arbeitsplätze ab etwa DM 5000 mit Zeilendisplay, residentem RAM-Speicher und externer Mikro-Floppy, mit denen die Büroarbeit erheblich erleichtert, beschleunigt und von unnötigen Wiederholungen befreit werden kann. Und das mit einer einfachen Hantierung.

Selbst wenn man in der Ausrüstung eines Schreib-Arbeitsplatzes nicht so hoch gehen will oder muß: Die Vereinfachungen sind sowohl für gelegentliche und für intensive Schreiber gegeben. Die schon selbstverständlichen Zeilen-Displays (oft auch mit LCD-Anzeige) ermöglichen eine Korrektur von Eingabefehlern noch vor dem Abdruck einer Zeile.

Es muß schon verwundern, daß auch gut verdienende Großbetriebe die Möglichkeiten noch wenig nutzen. Man schaue z.B. einmal auf die Arbeitstische in Banken und Sparkassen. Oft findet man da noch Typenhebelmaschinen. Andere Betriebe lassen ihre Ingenieure noch jeden Text von Hand vorschreiben, ehe er ins Schreibbüro geht, in der Auffassung, man fände eher einen zusätzlichen Ingenieur als eine tüchtige Schreibdame. Statt daß man nun den Ingenieuren gleich eine Schreibmaschine an den Arbeitsplatz stellt. Selber zu schreiben ist bestimmt nicht unstandesgemäß, unsere Jugendlichen akzeptieren die Tastatur eines Computers als ganz selbstverständlich. Warum nicht auch diejenigen, die schon länger im Beruf stehen? Es gibt sicher die Gefahr, daß wir in unseren Lande zu lange an verkrusteten Strukturen festhalten. Nötig ist das nicht.

Vor einem Jahr wurde der IBM-PC auch in unseren Markt eingeführt. Zugleich fand man auf der Hannover-Messe schon eine Reihe von Anbietern für kompatible Systeme. In diesem Jahr muß man berichten, daß man die Zahl der angebotenen kompatiblen Geräte kaum noch überschauen kann. Obwohl der IBM-PC eine sehr konservative Architektur hat, ist er weltweit zu einem großen Erfolg geworden. Eine Unzahl von Betrieben unterstützt ihn mit Hardware zur Ergänzung und mit leistungsfähigen Programmen. Es gibt daneben eine zunehmende Zahl von Zeitschriften für den PC. Und wenn man BYTE aufschlägt, die große amerikanische Fachzeitschrift für Mikrocomputer, so findet man sowohl im redaktionellen Teil wie auch in den Anzeigen ein Vorherrschen von Themen und Anpreisungen für den PC von IBM. Man geht wohl richtig in der Annahme, daß ähnliches auch hier eintritt, zumal der hier bedeutendste Anbieter Commodore eine zweite Produktlinie mit den kompatiblen Commodore-PCs aufbaut (siehe in diesem Heft).

Viele Computerbetreiber werden sich also künftig auch auf diesem Marktsektor umschauen, um Möglichkeiten zu nutzen. Dabei wird zu bedenken sein, daß die vielen Anbieter das Wort Kompatibilität durchaus unterschiedlich buchstabieren. Die Fragen fangen bei den Diskettenformaten an, setzen sich bei 8- oder 16-poligem Datenbus fort und münden in die Frage, unter welchem Betriebssystem und welcher Version ein Programm lauffähig ist. Hier wird man bei aller verbesserten Architektur, die von verschienenen Anbietern verwirklicht wurde, sehr aufmerksam wählen müssen. Und man wird sicher gut daran tun, nicht zu sehr auf Außenseiter zu setzen, denn es ist immer gut, wenn das eigene System vom Markt gut unterstützt ist und wenn man beim eigentlich immer einmal eintretenden Maschinenbruch am nächsten Tag mit geliehenen Aggregaten weiterarbeiten kann.

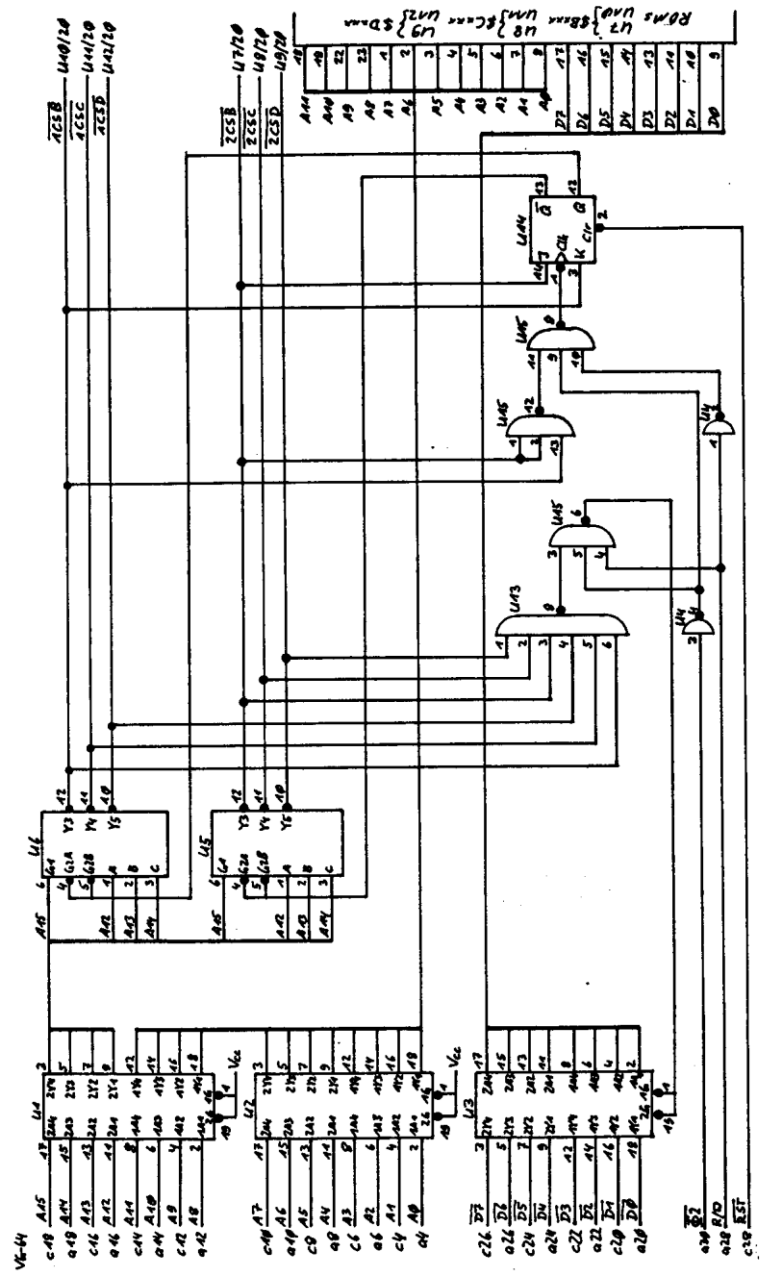
Mit dem vorliegenden Heft wird der sechste Jahrgang dieser Zeitschrift komplettiert. Angefangen beim KIM-1 wurden seither schon einige Rechnergenerationen intensiv betreut. Die Entwicklung geht weiter. Der Herausgeber hat sich daher entschlossen, dieser Zeitschrift

ab Heft 37 (Juni 1984) den neuen Titel MICRO MAG

zu geben. Damit wird der Tatsache Rechnung getragen, daß diese Bezeichnung der Zeitschrift in der Leserschaft schon bisher geläufig war und daß zweitens schon in den vergangenen Jahren eine Ausweitung der Themen über den spezifischen 6502-Bereich hinaus vorgenommen wurde. Der Leser wird wie bisher die wichtigen Geräte und Interfaces betreut sehen und daneben weitergreifende Themen finden, die der rasanten Entwicklung Rechnung tragen.

Von der 2. Auflage des 'Das Buch 1-6 des 65xx MICRO MAG' stehen nur noch wenige restliche Exemplare zur Lieferung zur Verfügung. Nach deren Auslieferung wird keine Neuauflage erfolgen, weil die dort abgehandelten Themen und Lösungen so aktuell und wichtig nicht mehr sind.

□□□



ROM-Umschaltung (Schaltplan)

Assembler

R65C02-Assembler

für AIM 65 und kompatible Systeme. 2-Pass-Assembler für den kompletten (!) Befehlssatz mit zusätzlichem Komfort. Assemblerliste formatiert, Assemblierung mit Offset für virtuelle Speicherbereiche. Der Assembler läuft auf der herkömmlichen 6502. - 2 EPROMs für Speicherbereiche Ihrer Wahl DM 195,--

6805/68705 Cross-Assembler

für AIM 65 und kompatible Systeme. 2-Pass-Assembler mit allem gewohnten Komfort und 6502-Syntax. Erzeugt aus den Mnemonics von Motorola 6805-Code. 2 EPROMs wie vor DM 280,--

6805/68705-Assembler unter FORTH

wie in Heft 35 beschrieben: 1-Pass-Assembler mit Verarbeitung von Symbolen und Labeln. Codeablage für virtuelle Speicherräume. 2 EPROMs für AIM 65 DM 100,--

Mathe-ROM für 6502

Implementierung nach Peter Rix (s. Hefte 28/29). Fließkommaarithmetik und höhere mathematische Funktionen wie in Microsoft-BASIC für AIM-65-FORTH und für jedes 6502-Assemblerprogramm (20 S. Dokumentation mit Einsprungspunkten und Argumenten), für Sockel $D000 DM 124,30

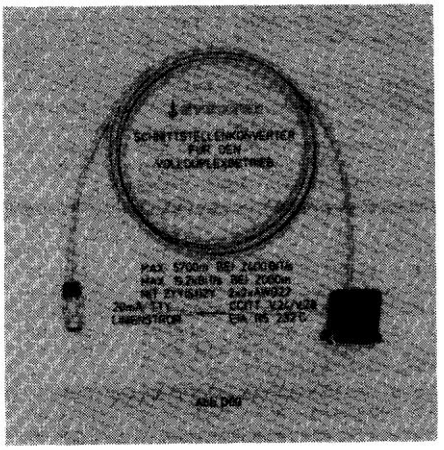
Floppy-Schnittstelle w. Beschaffungsschwierigkeiten nicht lieferbar.

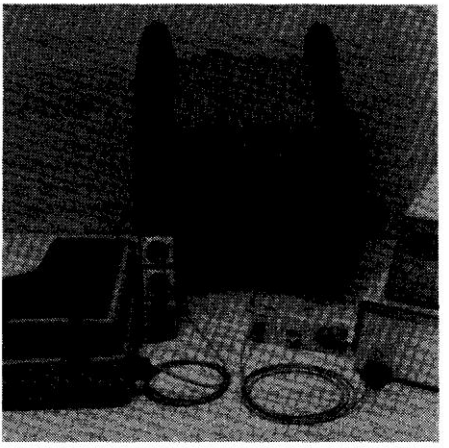
Roland Löhr, Hansdorfer Str. 4, D-2070 Ahrensburg
Tel.: 04 102 - 55 816

Kleinanzeigen

Geld sparen durch Selbstbau: Speichererweiterungen, Ramkarten, Epromkarten u. Programmiergeräte, Rombox etc.; z.B. 80 Zeichenkarte für VC20, Leerplatine incl. Software, Bauanleitung und Schaltplan 99 DM; 64 kBytes dyn. Ramkarte 99 DM; **Katalog für CBM, VC20 und C-64** gegen 2 DM. Roßmöller Datentechnik, Finkenweg 1, 5309 Meckenheim.

6502 Erweiterung: 2x16 RAM statisch á 150,-, 16/32k Eprom 10 Super I/O 2x6551, 1x6522 C-MOS Uhr 256 BYT C-MOS Ram 300,-. Mini DCR-Controller mit Software 250,-, 2 Mini-DCR Laufwerke á 180,-. Video-Platine 295,-, I/O-Platine 2x6522 100,-, Bus-Platine 9 Steckplätze 100,-, 10x SN 7405 á 2,50. Harald Stock, Hermann-Hesse-Str. 26, 6052 Mühlheim 2, Tel. 06108/1692.





V.24/V.28 oder RS 232 C — 20 mA Konverter eingebaut in einem serienmäßigen Subminiatur „D" Steckergehäuse
Gebrauchsmuster angemeldet!

Haben Sie weit auseinanderliegende Datengeräte mit Standard V.24/V.28 oder RS 232 C Schnittstelle? ... dann ist 20 mA Current Loop die Lösung für Ihr lokales Netzwerk.

Den Konverter liefern wir:

Technische Daten:

- V.24/V.28 oder RS 232 C — 20 mA Current Loop.
- Galvanische Trennung
- Umschaltbar aktiv/passiv
- Betriebsanzeige: Senden, grüne LED
  Empfangen, rote LED
- Maximale Leitungslänge 5700 Meter, Kabel 2YY (ST) 2Y2 x 2 x AWG22 (0,64); 58nF/km; 114 Ohm/km, 2,4 k Bit/sec sind dann noch möglich.
- Maximale Übertragungsrate bei gleichem Kabel: 19,2 k Bit/s bis 2000 m aktiv senden
  und bis 290 m passiv senden.
- Einspeisung: +/— 12 V aus angeschlossenem Datengerät oder über Schalterdosennetzgerät SKNT 20, das zusätzlich in die Wandsteckdose der fest verlegten 20 mA Datenleitung eingebaut wird.
- Stromverbrauch: +12 V ca. 60 mA max.
  —12 V ca. 30 mA max.
- Abmessungen: 49 x 54 x 15 mm
  AMP Subminiatur D Stecker 25polig, lieferbar mit Stiften oder Buchsen.
- Auf Wunsch mit flexibler Leitung 7 x 0,14 und Rundstecker Amphenol Tuchel 7polig.
  Vier Adern Daten, drei für externe Stromversorgung.

Preise incl. MWSt

Konverter ohne Kabel	DM 342,00
Konverter mit 2,5 m 7 x 0,14 wie Abb. DO9:	DM 438,90
Netzgerät SKNT 20 für Einbau in Installationsdose:	DM 171,00

Wir beraten und planen für Sie,
Ihr lokales Netzwerk!
Fordern Sie bitte weiteres Informationsmaterial an.

STECKER
INGENIEURBÜRO

5000 Köln 60 (Niehl)
Postfach 60 07 66
Delmenhorster Straße 20
Telefon (02 21) 7 12 40 18

Neue Telefon-Nummer

n der MC (Franzis-Verlag) ist ab 1/84 das Rechnersystem MC 65 beschrieben.
Eine enge (lizensierte) Verwandschaft zum AIM 65 ist nicht zu uebersehen....
Wir versenden ab Bremen alle notwendigen Teile : (Preise inkl. MWST)

MC 65 - CPU Leerkarte aus erstklassigem Material m. Lötstopmaske	79.00
MC 65 - Monitorprogramm (2 x 4 kByte) inkl. overlay-sourcelist.	78.00
Adressdekoder TTL - PROM (512 x 8)	29.00
TMS 4500 (Texas Instruments)	72.00
TMS 4164 - 150 nSek. (8 Stck = 64 kByte RAM) Texas Instruments	190.00
Rockwell-CPU 65C02 (!), 6522, 6532, Quarz 8 MHz HC18U,TMS 4500	176.00
Standard - 6502, 6522, 6532, Quarz 8 MHz HC 18 U,TMS 4500	144.00
Komplettbaus.MC 65 CPU m.Sockeln u.allen Teilen inkl. Mon.-Eproms	540.00
MC 65 - Busplatine f.MC65CPU+3xBus+MC-Terminal+Netzteil Leerkarte:	48.00
MC 65 - Epsio-Karte (Eprom-burner 27/2516 - 27128) mit 6551 (RS232 sowie 6522 (User-Port) lieferbar ab ca. Mitte April 84 Leerkarte	79.00
MC 65 - Eprom-burner Software (im Eprom 2532) auf $B000 ab 4/84	48.00
Eurokarten-Fertignetzteil +5V/5A,+12V/2A,-5V&-12V .5A einschiebbar	168.00
Commodore - Floppy 1541 komplett m. Kabel & SERVICE-MANUAL (!!!)	798.00
Steders 2-pass 6502-Assembler m.Offset(Eprom),AIM65-Dokument.erf.	72.00
Steders Text Editor Ext.(Zeilenorientiert) 4K Eprom ($B000)	67.00
65C02-Assembler (Löhr) f. Rockwell 65C02 (2 Eproms $CD00 - $DFFF)	195.00
dito im "Paket" mit Rockwell 65 C 02 - CPU f. Aufrüstung AIM 65	230.00

Ass. läuft auf MC65 und AIM65,liefert offset u. Symbol-Tabelle
Inbegriffen sind 6 Monate Software-Pflege d. Autor sowie 65C02-Unterlagen!

6502-Steuerplatine MC 2/82 Leerk.f.2*6522,2*2716,1 kByte RAM	73.00
6502-Steuerplatine n. C'T Leerkarte f.6502,6532,Eprom(m.Schalt)	35.00
MC TERMINAL 80x24 RS232 u. 20 mA (MC 1&2/83) Leerkarte :	78.00
Betriebsprogramm 4.7 Wordstar-kompatibel/ADM3A m. ESCAPE-Sequ	48.00
Betriebsprogramm 3.6 wie in MC 1&2/83, gut f. AIM 65 u.MC 65	20.00
Betriebsprogramm-Sourcelisting 3.6 od. 4.7 Originaldruck	50.00
Charactergenerator US und Deutsch umschaltbar im 4 kByte Eprom	28.00
Teilbausatz: TMS 9995,9902,Quarz,Video-Controler 6845	148.00
Cherry-Tastatur G80-0246/247 deutsch oder US	225.00
Gehäuse f. Tastatur, orig. cherry	47.50
Schaltnetzteil Apple-Form +5Volt/5A,+12V/4A,-5V/.5A,-12V/.5A	166.00
ASCII -Tastatur stabil Gr/Kleinschr. Apple-Typ m. Metallrahmen	166.00
stab. Kunststoff-Gehäuse Apple II-Form f. ob. Tastatur + Netzteil	166.00
Text- und GrafikDRUCKER CP 80,80Zeichen/Sekunde in Schönschrift	950.00
RS 232 & 20 mA-Schnittstelle f. CP 80 m. eig. CPU/2k Buffer	245.00
3 Farbbandkassetten für Matrixdrucker CP 80	52.00
Video-Monitor 12" TAXAN 18/20MHz grün Kompaktgerät m.SERVICE-MAN.	360.00 *)
Philips-Kompaktrechner Z80 A (4 MHz) 2010-1 bzw. 2012 mit :	
2 Laufwerken 5-1/4 " a 160 kByte u. CP/M 2.2,WORDSTAR und CALCSTAR	5370.00
2 Laufwerken 5-1/4 " a 640 kByte u. CP/M 2.2,WORDSTAR und CALCSTAR	6550.00
Euro-Busadapterkarten f.Service 100x230mm m.DIL-Schalt.&Lötstopmaske	
---> 64 polige Leerkarte aus erstklassigem Platinenmaterial	58.00
Satz Steckverbinder plus 64 Siemens-DIL-Schalter m. Klemme f.Oszil	44.00
---> 96 polige Leerkarte fuer 96 DIL-Schalter oder Bruecken	78.00
Satz Steckverbinder plus 96 Siemens-DIL-Schalter m.Klemme f.Oszill	62.00
Bücher:Mit Computern steuern 6502 (Feichtinger, Franzis-Verlag)	38.00
Programmierpraxis m.d.6502 (Feichtinger, Franzis-Verlag)	19.80
Original-Dok. Rockwell AIM 65 komplett neue Auflage m. 4 Buechern	48.00
AIM 65 Handbuch in deutsch	28.00

Wir liefern nicht nur diese Fachbücher - Katalog gegen Freiumschlag !

==========

Sonderaktion durch Grosseinkauf bis 25.4.84: Text-u.Grafikdrucker ->CP 80<-
m. 9-Nadel-Kopf,bidirektional&druckwegopt.verstellbarer Traktor&Einzelblatt (!)
mit der erstklassigen Korrespondenzschrift (durch Carbonband) mit Centronics-
schnittstelle,1 Farbband,deutschem Handbuch u.dickem SERVICE-MANUAL 888.--DM

*) Aktionspreis für bis zum 25.4.84 hier eingehende Bestellungen: TAXAN-Monitor 335,-- DM

HANS-JOACHIM REGGE · BREMEN · ELEKTRONIK

Fesenfeld 57 · 2800 Bremen 1
Telefon (0421) 71114

R. S. Microcomputer GmbH
Gagernstraße 4
8580 Bayreuth
Telefon (0921) 68877

HARD + SOFT

Bücher

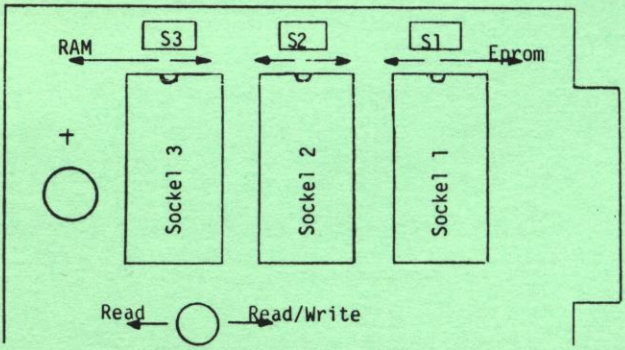
Bestell-Nr.	Titel	ca. Seiten	Preis DM
93701	**CBM Serie 700 DIN ROM-RAM-I/O Assemblerlisting, BASIC CBM 720, Operatingsystem CBM Serie 700 DIN-Tastatur,** kommentierte ROM-Umschaltroutinen m. Beispielen, Erklärung des Cartridge-Anschlusses und RS232-Schnittstelle	400	69,--
93702	**CBM Serie 700 ASCII ROM-RAM-I/O Assemblerlisting, Operatingsystem CBM Serie 700 m. ASCII-Tastatur**	200	49,--
93703	**CBM Serie 8000 DIN ROM-RAM-I/O Assemblerlisting, BASIC und Operatingsystem DIN-Tastatur, CBM 8032/8096/8296**	400	69,--
93704	**Commodore C 64 ROM-RAM-I/O Assemblerlisting, BASIC und Operatingsystem**	400	69,--
93705	**Commodore Floppy 8250 DOS ROM-RAM-I/O Assemblerlisting, Bus- und Disk-Controller**	400	79,--
93710	**Vergleichsliste Adressen CBM 8000/710/720** sortiert nach CBM 8000, lückenlose Liste der Zero-Page- u. RAM-Adressen CBM Serie 700, kommentiert	50	29,--
93720	**Cartridgemodul für CBM 700 mit Kassettenbetriebssystem**	100	39,--

Kurzbeschreibung: O.a. Preise incl. 7% MWSt. Die Bücher sind mehrfarbig ausgeführt, im Rücken verleimt und für Ringbuchablage vorgelocht. Sie enthalten ein ausführlich kommentiertes strukturiertes Assemblerlisting. Darin sind alle Adressen und Prozessorbefehle mit den dazugehörigen Menmonics gelistet und kommentiert. Nach Möglichkeit wurden Original Commodore-Label verwendet. Die Dokumentationen werden durch Crossreferenzlisten vervollständigt, für alle vorkommenden Labels, Adresse, Wert und Zeilennummern. Außerdem sind alle vorkommenden Zero-Page-, RAM- und I/O-Adressen, deren Belegung und bitweise Bedeutung mit Kommentar enthalten. Autoren: R. Schineis und O.-M. Braun

Cartridge-Platine

für CBM 600 / 700

Rechner-Speichererweiterung bis zu 24 K-Byte RAM oder ROM in Bank 15 mit Datensicherung. Das Modul ist durch den Benutzer frei wählbar, auch gemischt, mit drei Eproms '2764' (je 8 K-Byte) oder statischen RAMs 'HM 6264' o.ä. bestückbar. Die einzelnen Sockel sind mit den Adressen (hex) belegt: 2000-3FFF, 4000-5FFF und 6000-7FFF. Die Schalter dienen dem Schreibschutz bzw. der Batteriepufferung. Preis DM 198,--+MWSt.



65xx MICRO MAG

COMPUTING · SOFTWARE · HOBBY

Herausgeber:
Dipl.-Volkswirt Roland Löhr
Hansdorfer Straße 4
D–2070 Ahrensburg
Tel.: 04 102 - 55 816

65xx MICRO MAG erscheint zweimonatlich, jeweils Mitte Februar, April usw..COPYRIGHT 1984 by Roland Löhr. Alle Rechte vorbehalten, auch die des auszugsweisen Nachdruckes, der Übersetzung, der fotomechanischen Wiedergabe und die der Verbreitung auf magnetischen und sonstigen Trägern. Beiträge, die nicht besonders gekennzeichnet sind, stammen vom Herausgeber. - Offsetdruck: Druckartist Gerhard M. Meier, Hamburg 70.

Bezugsbedingungen: Abonnement ab laufender Ausgabe für 6 Hefte DM 54,-- (Inlandsendpreis). Ausland/foreign via surface mail DM 59,--, USA air 26 Dollar. Abonnements laufen bis auf Widerruf mit Kündigungsmöglichkeit bis zu zwei Wochen vor deren Ablauf.

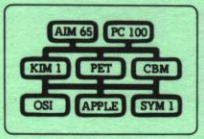
Nachliefermöglichkeiten: Hefte 1-6 und 7-13 sind als Buch nachlieferbar (s. Anzeige unten). Hefte 14-35 können alle als Einzelhefte zu DM 7,80/St. + DM 2,50 je Sendung geliefert werden. Einzelpreis bei 10 und mehr Heften je Sendung DM 6,--.

Private Besteller werden um Überweisung/Scheck (auch Auslandsschecks) zusammen mit der Bestellung gebeten. Konto Roland Löhr, Nr. 654 70-202 Postscheckamt Hamburg, BLZ 200 100 20.

Leser-Service des Herausgebers

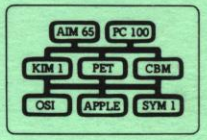
Das Buch 1-6 ist nur noch kurze Zeit lieferbar!

Das Buch 1-6 des 65xx MICRO MAG



230 Seiten, DM 26,--

Das Buch 7-13 des 65xx MICRO MAG



340 Seiten, DM 42,--

FORTH User's Manual
Rockwell-Handbuch für das FIG-FORTH des AIM 65. Mit der Erläuterung des Befehlssatzes auch für andere FORTH-Betreiber geeignet. Ca. 300 S., engl. DM 30,--

Mathe-ROM nach P. Rix für FORTH des AIM + Assemblerprog. m. Doku DM 124,30

Vorstehende Preise inkl. MWSt, zuzüglich DM 2,50/Sendung + ggfs. NN + DM 2,--

Assembler für R65C02, MC6805 und 6809: Siehe im Heftinneren.